IMPLEMENTASI PRINSIP *ATOMIC HABITS* DAN ALGORITMA *RANDOM FOREST* PADA APLIKASI *HABIT TRACKER*

SKRIPSI

WINDI YULINDA
2022230015

PROGRAM STUDI INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS PRABUMULIH
2026

IMPLEMENTASI PRINSIP *ATOMIC HABITS* DAN ALGORITMA *RANDOM FOREST* PADA APLIKASI *HABIT TRACKER*

Untuk Memenuhi Salah Satu Syarat Memperoleh Gelar Sarjana
Program Studi Informatika Universitas Prabumulih

WINDI YULINDA
2022230015

FAKULTAS ILMU KOMPUTER
PROGRAM STUDI INFORMATIKA
UNIVERSITAS PRABUMULIH
2026

HALAMAN PENGESAHAN

IMPLEMENTASI PRINSIP *ATOMIC HABITS* DAN ALGORITMA *RANDOM FOREST* PADA APLIKASI *HABIT TRACKER*

SKRIPSI

Diajukan Sebagai Salah Satu Syarat untuk Memperoleh Gelar Sarjana Program Studi Informatika Universitas Prabumulih

Oleh :

Windi Yulinda

2022230015

Prabumulih, 22 Juni 2026

Pembimbing I	**Pembimbing II**
Iwan Setiawan, S.Kom., M.Kom.	Nur Aini H. S.Kom., M.Kom.
NUPTK. 1541769670130303	NUPTK. 2434764664300022

Ketua Program Studi Informatika

Iwan Setiawan, S.Kom., M.Kom.

NUPTK. 1541769670130303

Dipindai dengan CamScanner

HALAMAN PERSETUJUAN

Skripsi ini dengan judul "Implementasi Prinsip *Atomic Habits* dan Algoritma *Random Forest* pada Aplikasi *Habit Tracker*" telah dipertahankan dan disetujui di hadapan Tim Penguji Skripsi Program Studi Informatika Fakultas Ilmu Komputer Universitas Prabumulih pada tanggal 22 Juni 2026.

Prabumulih, 22 Juni 2026

Tim Penguji Tugas Akhir

Ketua:

Iwan Setiawan, S.Kom., M.Kom ( )

NUPTK. 1541769670130303

Anggota:

1. Nur Aini H, S.Kom., M.Kom ( )

   NUPTK. 2434764664300022

2. Khana Wijaya, S.Kom.,M.Kom ( )

   NUPTK. 7146768669130363

Mengetahui,

Dekan Fakultas Ilmu Komputer	**Ketua Program Studi Informatika**
(Khana Wijaya, S.Kom.,M.Kom)	(Iwan Setiawan, S.Kom., M.Kom)
NUPTK. 7146768669130363	NUPTK. 1541769670130303

Dipindai dengan CamScanner

PERNYATAAN INTEGRITAS

Saya yang bertanda tangan di bawah ini

Nama : Windi Yulinda

NIM : 2022230015

Program Studi : Informatika

Menyatakan dengan sebenarnya bahwa, skripsi ini benar-benar merupakan hasil karya saya dan tidak terdapat karya orang lain, apabila di kemudian hari terbukti atau dapat dibuktikan bahwa skripsi ini hasil *plagiasi*, maka saya bersedia menerima sanksi sesuai dengan ketentuan yang berlaku atas perbuatan tersebut.

Prabumulih, 22 Juni 2026

Yang membuat pernyataan,

Windi Yulinda

NIM.2022230015

Dipindai dengan CamScanner

ABSTRAK

Implementasi Prinsip *Atomic Habits* dan Algoritma *Random Forest* pada Aplikasi *Habit Tracker*

Windi Yulinda, 2022230015, 2026

Mahasiswa sering menghadapi kesulitan dalam mempertahankan kebiasaan belajar yang konsisten, sehingga diperlukan sistem pendukung yang mampu membantu membangun rutinitas positif secara terarah. Penelitian ini bertujuan mengembangkan aplikasi *Habit Tracker* berbasis Android yang mengintegrasikan prinsip *Atomic Habits* serta algoritma *Random Forest* untuk memprediksi konsistensi kebiasaan pengguna. Prinsip *Cue, Craving, Response, dan reward* diwujudkan dalam fitur notifikasi, antarmuka aplikasi, penanda aktivitas, dan sistem poin. Pengembangan sistem dilakukan menggunakan metode *Agile* untuk memastikan proses iteratif dan fleksibel, sedangkan pengujian kualitas aplikasi diterapkan melalui pendekatan *Statement* . Hasil penelitian diharapkan menghasilkan aplikasi yang mampu memberikan dukungan adaptif, memantau perkembangan kebiasaan, serta membantu mahasiswa membangun kebiasaan positif secara lebih terstruktur dan berkelanjutan.

Kata kunci: *Habit Tracker, Atomic Habits, Random Forest, Agile, Statement.*

ABSTRACT

Implementation of Atomic Habits Principles and Random Forest Algorithm in a Habit Tracker Application

Windi Yulinda, 2022230015, 2026

Students often struggle to maintain consistent study Habits, underscoring the need for a support system that helps establish positive routines in a structured way. This research aims to develop an Android -based Habit Tracker application that integrates the principles of Atomic Habits and the Random Forest algorithm to predict Users' Habit consistency. The principles of Cue, Craving, Response, and reward are realized through notification features, the application's interface, activity markers, and a points system. The system was developed using the Agile method to ensure an iterative and flexible process, and the application's quality was tested using a Statement approach. The expected outcome of the research is an application that provides adaptive support, monitors Habit development, and assists students in building positive Habits in a more structured and sustainable way.

Keywords: *Habit Tracker, Atomic Habits, Random Forest, Agile, Statement.*

KATA PENGANTAR

Segala puji bagi Allah Subhanahu wa Ta'ala atas segala nikmat, rahmat, dan karunia-Nya sehingga penulis dapat menyelesaikan skripsi berjudul "Implementasi Prinsip *Atomic Habits* dan Algoritma *Random Forest* pada Aplikasi *Habit Tracker*" ini dengan baik dan tepat waktu. Shalawat serta salam semoga tercurah kepada Nabi besar Muhammad Shallallahu 'alaihi Wasallam, keluarganya, sahabat, dan seluruh pengikutnya hingga akhir zaman. Skripsi ini disusun sebagai salah satu syarat untuk mendapatkan gelar Sarjana Komputer (S.Kom) di Program Studi Informatika, Fakultas Ilmu Komputer, Universitas Prabumulih. Dalam proses penyusunannya, penulis banyak mendapatkan bantuan, bimbingan, motivasi, dan dukungan dari berbagai pihak. Oleh karena itu, penulis mengucapkan terima kasih yang sebesar-besarnya kepada:

1. Ibu Dr. Yuniar Pratiwi, S.Si., M.Si selaku Rektor Universitas Prabumulih sekaligus yang telah memberikan izin untuk melakukan penelitian di Universitas Prabumulih.
2. Bapak Andi Christian, S.Kom., M.Kom selaku Wakil Rektor Universitas Prabumulih.
3. Bapak Khana Wijaya, S.Kom., M.Kom selaku Dekan Fakultas Universitas Prabumulih.
4. Ibu Suhartini, S.Kom., M.Kom selaku Wakil Dekan Bidang Keuangan Fakultas Ilmu Komputer Universitas Prabumulih.
5. Ibu Nur Aini Hutagalung, S.Kom., M.Si., M.Kom selaku Wakil Dekan Bidang Kemahasiswaan Fakultas Ilmu Komputer Universitas Prabumulih sekaligus pembimbing II.
6. Bapak Iwan Setiawan, S.Kom., M.Kom selaku Ketua Program Studi Informatika sekaligus pembimbing I.
7. Ibu Riza Kartina, S.Kom., M.Kom selaku pembimbing akademik.
8. Bapak dan Ibu Dosen Program Studi Informatika Fakultas Ilmu Komputer Universitas Prabumulih.
9. Staf Administrasi Rektorat Universitas Prabumulih yang telah memberikan izin dan membantu dalam penelitian ini.

10. Bapak/Ibu staf administrasi Fakultas Ilmu Komputer Universitas Prabumulih.

Jazakumullahu khayran katsīran. Semoga Allah Subhanahu wa Ta'ala membalas segala kebaikan dari semua pihak yang telah memberikan bantuan dan dukungan hingga skripsi ini terselesaikan. Penulis menyadari bahwa skripsi ini masih jauh dari sempurna. Oleh karena itu, kritik dan saran yang membangun sangat diharapkan guna perbaikan di masa mendatang. Penulis berharap semoga skripsi ini dapat memberikan manfaat, pengetahuan, serta kontribusi bagi penulis sendiri maupun pembaca. Aamiin.

Prabumulih, Juni 2026

Windi Yulinda
2022230015

Dipindai dengan CamScanner

DAFTAR ISI

HALAMAN JUDUL ii
ABSTRAK v
ABSTRACT vi
KATA PENGANTAR vii
DAFTAR ISI ix
DAFTAR TABEL xii
DAFTAR GAMBAR xiv
DAFTAR LAMPIRAN xvii
DAFTAR SINGKATAN xviii
BAB I 1
PENDAHULUAN 1
1.1 Latar Belakang 1
1.2 Rumusan Masalah 3
1.3 Batasan Masalah 3
1.4 Tujuan penelitian 4
1.5 Manfaat Penelitian 5
1.6 Sistematika Penulisan 5
BAB II 8
TINJAUAN PUSTAKA 8
2.1 Pengertian Implementasi 8
2.2 Pengertian Prinsip 8
2.3 Pengertian *Atomic Habits* 8
2.4 Pengertian *Random Forest* 9
2.5 Pengertian Aplikasi 12
2.6 Pengertian *Habit Tracker* 12
2.7 Pengertian Android 13
2.8 Pengertian *Figma* 13
2.9 Pengertian *Visual Studio Code* 13
2.10 Pengertian *Flutter* 14

2.11 Pengertian *Dart* 14
2.12 Pengertian Android *Studio* 15
2.13 Pengertian *SQLite* 15
2.14 Penelitian Terdahulu 15
2.15 Kerangka Berpikir Penelitian 19
BAB III 22
METODE PENELITIAN 22
3.1 Objek dan Jadwal Penelitian **22**
3.1.1 Objek Penelitian **22**
3.1.2 Waktu dan Jadwal Penelitian **23**
3.1.3 Sejarah, Visi Misi, dan Tujuan Universitas Prabumulih **24**
3.2 Metode Penelitian **26**
3.3 Sumber Data **26**
3.4 Teknik Pengumpulan Data **27**
3.4.1 Teknik Analisis Data Kuesioner 29
3.5 Metode Pengembangan Sistem **30**
3.6 Alat Bantu Perancangan **37**
3.6.1 Alat Bantu Utama 37
3.6.2 Alat Bantu Desain 44
3.6.3 Alat Bantu *Debugging* dan Pengembangan 44
3.7 Pengujian Sistem **45**
BAB IV 47
ANALISA DAN PERANCANGAN 47
4.1 Analisa Sistem **47**
4.1.1 Analisa dan Evaluasi Sistem Berjalan 47
4.1.2 Analisa Kebutuhan 48
4.2 Perancangan Sistem (Desain) yang diusulkan **52**
4.2.1 Perancangan Sistem (Logis) 52
4.2.2 Perancangan *Database* 73
4.2.3 Perancangan Antarmuka (*UI*) 77
BAB V 99
HASIL DAN PEMBAHASAN 99

5.1 Spe sifikasi Perangkat **99**
5.1.1 Perangkat Keras yang digunakan 99
5.1.2 Perangkat Lunak yang digunakan 100
5.2 Implementasi *Database* **101**
5.3 Tampilan Antarmuka **105**
5.4 Hasil Pengujian Sistem **128**
5.4.1 Pengujian Modul *Login* 128
5.4.2 Pengujian Modul *Register* 138
5.4.3 Pengujian Modul Tambah *Habit* 154
5.4.4 Pengujian *Input Progress Habit* 166
5.4.5 Pengujian Modul Statistik *Habit* 178
5.4.6 Pengujian Modul *Random Forest AI* 191
5.5 Pembahasan 201
BAB VI 204
KESIMPULAN DAN SARAN 204
6.1 Kesimpulan 204
6.2 Saran 205
DAFTAR PUSTAKA 206
LAMPIRAN 211

DAFTAR TABEL

Tabel 2. 1 Penelitian Terdahulu 16
Tabel 3. 1 Jadwal Penelitian..... 23
Tabel 3. 2 *Sprint* Pengembangan Sistem 35
Tabel 3. 3 Simbol-simbol *Use Case Diagram* 38
Tabel 3. 4 Simbol *Activity Diagram* 40
Tabel 3. 5 Simbol *Class Diagram*..... 42
Tabel 4. 1 Kebutuhan Perangkat Lunak (*Software*)..... 50
Tabel 4. 2 *Perangkat Pengembangan (Developer)*..... 51
Tabel 4. 3 Perangkat Pengguna (User)..... 51
Tabel 4. 4 *Users* 73
Tabel 4. 5 *Habit*s 74
Tabel 4. 6 *Habit*s Log..... 75
Tabel 4. 7 *Reset Password* 75
Tabel 5. 1 Perangkat Keras yang digunakan..... 99
Tabel 5. 2 Perangkat Lunak yang digunakan..... 100
Tabel 5. 3 Identifikasi Statement Fungsi Login 129
Tabel 5. 4 Cyclomatic Complexity Fungsi Login..... 136
Tabel 5. 5 Test case Fungsi Login 137
Tabel 5. 6 Statement Coverege Fungsi Login..... 137
Tabel 5. 7 Identifikasi Statement Fungsi Register 140
Tabel 5. 8 Cyclomatic Complexity Fungsi Register 150
Tabel 5. 9 Test case Fungsi Register..... 152
Tabel 5. 10 Statement Coverage Fungsi Register 153
Tabel 5. 11 Identifikasi Statement Fungsi Tambah Habit..... 156
Tabel 5. 12 Cyclomatic Complexity Fungsi Tambah Habit 163
Tabel 5. 13 Test case Fungsi Tambah Habit..... 164
Tabel 5. 14 Statement Covarege Fungsi Tambah Habit 165
Tabel 5. 15 Identifikasi Statement Fungsi Input Progress Habit 168
Tabel 5. 16 *Cyclomatic Complexity* Fungsi *Input Progress Habit* 175
Tabel 5. 17 *Test case* Fungsi *Input Progress Habit*..... 177

Tabel 5. 18 Statement Covarege Fungsi Input Progress Habit 177
Tabel 5. 19 Identifikasi Statement Fungsi Statistik 179
Tabel 5. 20 Cyclomatic Complexity Fungsi Statistik 188
Tabel 5. 21 Test case Fungsi Statistik..... 189
Tabel 5. 22 Statement Covarege Fungsi Statistik 190
Tabel 5. 23 Identifikasi *Statement* Fungsi Rekomendasi *Random Forest* 192
Tabel 5. 24 Cyclomatic Complexity Fungsi Rekomendasi Random Forest 198
Tabel 5. 25 Test case Fungsi Rekomendasi Random Forest..... 199
Tabel 5. 26 Statement Covarege Fungsi Rekomendasi Random Forest 200

DAFTAR GAMBAR

Gambar 2. 1 Kode Training Random Forest 11
Gambar 2. 2 Hasil Training Random Forest 11
Gambar 2. 3 Kerangka Berfikir Penelitian 19
Gambar 3. 1 Tahapan Metode *Agile* 32
Gambar 4. 1 *Use Case Diagram* 53
Gambar 4. 2 Activity Splash dan Cek Session 54
Gambar 4. 3 Activity *Login* Pengguna 55
Gambar 4. 4 Activity *Register* Pengguna 56
Gambar 4. 5 Activity *Reset Password* 57
Gambar 4. 6 Activity *Home* 58
Gambar 4. 7 Activity Add *Habit* 59
Gambar 4. 8 Activity *Checklist Habit* 61
Gambar 4. 9 Activity Edit *Habit* 63
Gambar 4. 10 Activity Hapus *Habit* 64
Gambar 4. 11 Activity Kelola *Reminder Habit* 65
Gambar 4. 12 Activity Statistik *Habit* 66
Gambar 4. 13 Activity Rekomendasi *AI* 67
Gambar 4. 14 Activity *Logout* 70
Gambar 4. 15 *Class Diagram* 71
Gambar 4. 16 Halaman *SplashScreen* 77
Gambar 4. 17 Halaman *Log In* 78
Gambar 4. 18 Halaman *Sign In* 79
Gambar 4. 19 Halaman Lupa *Password* 80
Gambar 4. 20 Halaman *Reset Password* 81
Gambar 4. 21 Halaman *Home* 82
Gambar 4. 22 Halaman *Cue* 83
Gambar 4. 23 Halaman *Craving* 84
Gambar 4. 24 Halaman *Response* 85
Gambar 4. 25 Halaman *Reward* 86
Gambar 4. 26 Halaman Tambah Habit 87

Gambar 4. 27 *Habit List* 88
Gambar 4. 28 Halaman Edit *Habit* 90
Gambar 4. 29 Halaman Hapus *Habit* 91
Gambar 4. 30 Halaman Statistik Harian 92
Gambar 4. 31 Halaman Statistik Mingguan 93
Gambar 5. 1 Tabel Users 101
Gambar 5. 2 Tabel Habits 101
Gambar 5. 3 Tabel Habit_logs 103
Gambar 5. 4 Tabel Reset Password 104
Gambar 5. 5 Tabel Hubungi Customer Service 104
Gambar 5.6 Tampilan halaman SplashScreen 106
Gambar 5.7 Tampilan halaman Login 107
Gambar 5.8 Tampilan halaman Register 108
Gambar 5.9 Halaman Lupa Password 109
Gambar 5.10 Halaman Reset Password 110
Gambar 5.11 Halaman Home 111
Gambar 5.12 Halaman Cue 112
Gambar 5.13 Halaman Craving 113
Gambar 5.14 Halaman Response 114
Gambar 5.15 Halaman Reward 115
Gambar 5.16 Tampilan halaman Tambah Habit 116
Gambar 5.17 Tampilan halaman Input progress Habit 117
Gambar 5.18 Halaman Edit Habit 119
Gambar 5.19 Halaman Hapus Habit 120
Gambar 5. 20 Tampilan halaman Statistik Harian 121
Gambar 5. 21 Tampilan Statistik Mingguan 122
Gambar 5. 22 Tampilan Analisis Habit 123
Gambar 5. 23 Tampilan Habit yang akan dianalisis 124
Gambar 5.24 Proses Analisis AI 125
Gambar 5. 25 Proses Random Forest Voting 126
Gambar 5.26 Hasil Prediksi Random Forest 127
Gambar 5. 27 Flowchart fungsi Login 131

Gambar 5. 28 Flowgraph fungsi Login 134
Gambar 5. 29 Flowchart fungsi Register 143
Gambar 5. 30 Flowgraph fungsi Register 147
Gambar 5. 31 Flowchart fungsi Tambah Habit..... 157
Gambar 5. 32 Flowgraph fungsi Tambah Habit..... 160
Gambar 5. 33 Flowchart fungsi Checklist Habit..... 170
Gambar 5. 34 Flowgraph fungsi Input Progress habit Habit 173
Gambar 5. 35 Flowchart fungsi Statistik Habit..... 181
Gambar 5. 36 Flowgraph fungsi Statistik Habit..... 185
Gambar 5. 37 Flowchart fungsi Rekomendasi Random Forest 193
Gambar 5. 38 Flowgraph fungsi Rekomendasi Random Forest 196

DAFTAR LAMPIRAN

Lampiran 1 Surat Permohonan Penelitian..... 211
Lampiran 2 Surat Balasan Permohonan Penelitian..... 212
Lampiran 3 Dokumentasi Wawancara & Observasi..... 213
Lampiran 4 Daftar Pertanyaan Wawancara 214
Lampiran 5 Pertanyaan Kuisioner..... 216
Lampiran 6 Data koding jawaban responden Tema 1..... 218
Lampiran 7 Data koding jawaban responden Tema 2..... 221
Lampiran 8 Data koding jawaban responden Tema 3..... 223
Lampiran 9 Data koding jawaban responden Tema 4..... 226
Lampiran 10 Data koding jawaban responden Tema 5..... 228

DAFTAR SINGKATAN

OS	: *Operating System*
UI	: *User Interface*
UX	: *User Experience*
VS Code	: *Visual Studio Code*
CLI	: *Command Line Interface*
IDE	: *Integrated Development Environment*
RDMS	: *Relation Database Management System*
AI	: *Artificial Intelligence*
Prodi	: Program Studi
UNPRA	:Universitas Prabumulih
STIE	: Sekolah Tinggi Ilmu Ekonomi
STIT	: Sekolah Tinggi Ilmu Teknik
STMIK	: Sekolah Tinggi Manajemen Informatika dan Komputer
FEB	: Fakultas Ekonomi dan Bisnis
FT	: Fakultas Teknik
FASILKOM	: Fakultas Ilmu Komputer
UML	: *Unified Modeling Language*
DB	: *Database*
CRUD	: *Create, Read, Update, Delete*
IPTEKS	: Ilmu Pengetahuan, Teknologi, dan Seni

BAB I
PENDAHULUAN

1.1 Latar Belakang

Perkembangan teknologi di era digital telah mendorong lahirnya berbagai inovasi yang mempermudah aktivitas manusia di berbagai bidang, termasuk pendidikan. Teknologi *mobile*, terutama aplikasi berbasis Android , berkembang pesat karena fleksibilitas, kemudahan akses, dan kemampuannya memberikan layanan yang bersifat personal. Selain itu, kemajuan di bidang kecerdasan buatan atau *Artificial Intelligence*, khususnya *Machine Learning*, memungkinkan sistem digital tidak hanya berfungsi sebagai media pencatatan, tetapi juga dapat menganalisis data pengguna dan menghasilkan prediksi berdasarkan pola aktivitas yang terbentuk.

Dalam dunia pendidikan tinggi, mahasiswa merupakan salah satu kelompok pengguna teknologi yang sangat aktif, terutama dalam penggunaan *smartphone*. Namun, pemanfaatan teknologi oleh mahasiswa masih banyak digunakan untuk keperluan komunikasi, hiburan, dan media sosial, sementara pemanfaatannya sebagai alat bantu untuk memantau pembentukan kebiasaan positif belum dilakukan secara optimal. Padahal, teknologi memiliki potensi besar untuk membantu mahasiswa dalam mencatat aktivitas, mengelola rutinitas, memantau perkembangan kebiasaan, serta mengevaluasi konsistensi aktivitas harian secara lebih terstruktur. Salah satu pendekatan yang relevan untuk diterapkan dalam konteks ini adalah prinsip *Atomic Habits* yang diperkenalkan oleh James Clear, yang menekankan bahwa perubahan perilaku besar dapat dibangun melalui kebiasaan kecil yang dilakukan secara bertahap melalui tahapan *Cue*, *Craving*, *Response*, dan *reward* (Clear, 2018).

Berdasarkan kuesioner kualitatif yang diberikan kepada mahasiswa dan terlampir, diketahui bahwa sebagian besar mahasiswa menghadapi tantangan dalam mengatur kebiasaan belajar dan aktivitas harian. Responden mengungkapkan bahwa mereka belum secara rutin melakukan aktivitas belajar mandiri, pengelolaan waktu masih kurang teratur, serta sering menunda atau melewatkan kegiatan akademik yang telah direncanakan. Selain itu, mahasiswa juga menyatakan belum memiliki media atau aplikasi khusus yang dapat membantu mencatat kebiasaan,

memberikan pengingat, menampilkan perkembangan aktivitas, serta memberikan evaluasi terhadap kebiasaan yang dilakukan. Temuan ini menunjukkan bahwa permasalahan utama bukan terletak pada kurangnya akses terhadap teknologi, melainkan belum tersedianya alat bantu digital yang secara khusus mendukung pemantauan pembentukan kebiasaan positif secara terarah dan berkelanjutan.

Untuk menjawab kebutuhan tersebut, diperlukan sebuah aplikasi yang tidak hanya berfungsi sebagai pencatat aktivitas, tetapi juga dapat membantu pengguna memantau dan mengevaluasi pola kebiasaan yang dilakukan. Aplikasi *Habit Tracker* dapat menjadi alat bantu bagi mahasiswa untuk mencatat kebiasaan, menginput progress yang telah dilakukan, melihat perkembangan melalui statistik, serta memperoleh umpan balik berdasarkan data aktivitas pengguna. Dalam penelitian ini, prinsip *Atomic Habits* diterapkan ke dalam fitur aplikasi, yaitu *Cue* melalui fitur notifikasi, *Craving* melalui tampilan antarmuka yang menarik, *Response* melalui tombol penanda kebiasaan, dan *reward* melalui sistem poin dan *Diamond Badge* apabila berhasil mencapai atau melebihi target *progress* yang telah ditentukan. Dengan penerapan prinsip tersebut, aplikasi diharapkan dapat membantu mahasiswa dalam memantau proses pembentukan kebiasaan positif secara lebih terstruktur.

Selain menerapkan prinsip *Atomic Habits*, penelitian ini juga memanfaatkan algoritma *Random Forest* sebagai bagian dari fitur analisis kebiasaan pengguna. *Random Forest* digunakan untuk menganalisis data progress pengguna selama tujuh hari terakhir, seperti target yang ditentukan, progress yang dicapai, tingkat pencapaian target, lama kebiasaan dijalankan, serta konsistensi aktivitas pengguna sehingga dapat menghasilkan prediksi tingkat konsistensi pengguna. Dengan adanya fitur prediksi tersebut, pengguna dapat memperoleh gambaran mengenai pola kebiasaan yang sedang dijalankan, sehingga aplikasi tidak hanya berperan sebagai media pencatatan, tetapi juga sebagai alat bantu evaluasi berbasis data.

Oleh karena itu, penelitian ini berjudul "Implementasi Prinsip *Atomic Habits* dan Algoritma *Random Forest* pada Aplikasi *Habit Tracker*" yang berfokus pada pengembangan aplikasi berbasis Android sebagai alat bantu bagi mahasiswa dalam memantau pembentukan kebiasaan positif. Penelitian ini menggabungkan pendekatan pembentukan kebiasaan melalui prinsip *Atomic Habits* dengan

pendekatan teknologi menggunakan algoritma *Random Forest* untuk memberikan prediksi konsistensi berdasarkan data aktivitas pengguna. Dengan adanya aplikasi *Habit Tracker* ini, mahasiswa diharapkan memiliki media yang dapat membantu mencatat, memantau, mengevaluasi, dan memperoleh gambaran konsistensi kebiasaan positif secara lebih terarah dan berkelanjutan.

1.2 Rumusan Masalah

Berdasarkan identifikasi masalah, maka rumusan masalah dalam penelitian ini adalah:

1. Bagaimana merancang dan membangun aplikasi *Habit Tracker* berbasis Android sebagai alat bantu pemantauan pembentukan kebiasaan positif mahasiswa?
2. Bagaimana mengimplementasikan prinsip *Atomic Habits* ke dalam fitur aplikasi *Habit Tracker*?
3. Bagaimana memanfaatkan algoritma *Random Forest* untuk memberikan prediksi atau penilaian terhadap konsistensi kebiasaan pengguna berdasarkan data aktivitas pada aplikasi *Habit Tracker*?

1.3 Batasan Masalah

Agar penelitian lebih terarah, maka ruang lingkup permasalahan dibatasi pada:

1. Penelitian difokuskan pada mahasiswa Program Studi Informatika Universitas Prabumulih angkatan 2022 dan 2023 sebanyak 75 mahasiswa sebagai responden kuesioner untuk menggali kebutuhan pengguna terhadap aplikasi *Habit Tracker* sebagai alat bantu pemantauan pembentukan kebiasaan positif.
2. Algoritma *Machine Learning* yang digunakan adalah *Random Forest*, yang penggunaannya dibatasi pada fitur prediksi atau penilaian tingkat konsistensi kebiasaan berdasarkan data aktivitas yang terdapat pada aplikasi *Habit Tracker*.
3. Prinsip pembentukan kebiasaan positif mengacu pada empat kaidah perubahan perilaku dalam buku *Atomic Habits* karya James Clear, yaitu :

a. Isyarat/petunjuk (*Cue*) : diimplementasikan dalam bentuk notifikasi.
b. Keinginan (*Craving*): diimplementasikan melalui tampilan antarmuka yang menarik.
c. Tanggapan (*Response*): diimplementasikan melalui fitur input progress kebiasaan.
d. Hadiah (*Reward*): diimplementasikan dalam bentuk poin *Diamond Badge*.

4. Aplikasi yang dibangun berbasis *mobile* Android dengan fitur utama berupa pencatatan kebiasaan, pengingat aktivitas, input progress kebiasaan, statistik perkembangan, sistem poin dan *Diamond Badge*, serta prediksi konsistensi menggunakan algoritma *Random Forest*.
5. Penelitian ini berfokus pada perancangan, pembangunan, dan pengujian teknis aplikasi. Penelitian ini tidak membahas pengukuran peningkatan konsistensi mahasiswa dalam jangka panjang setelah penggunaan aplikasi.

1.4 Tujuan penelitian

Tujuan yang ingin dicapai dalam penelitian ini adalah:

1. Merancang dan membangun aplikasi *Habit Tracker* berbasis Android sebagai alat bantu bagi mahasiswa dalam memantau pembentukan kebiasaan positif.
2. Mengimplementasikan prinsip *Atomic Habits* ke dalam fitur aplikasi *Habit Tracker*, yaitu:
   a. Isyarat/petunjuk (*Cue*): diimplementasikan dalam bentuk notifikasi.
   b. Keinginan (*Craving*): diimplementasikan melalui tampilan antarmuka yang menarik.
   c. Tanggapan (*Response*): diimplementasikan melalui tombol penanda kebiasaan.
   d. Hadiah (*Reward*): diimplementasikan dalam bentuk poin dan *Diamond Badge*.
3. Memanfaatkan algoritma *Random Forest* untuk memberikan prediksi atau penilaian tingkat konsistensi kebiasaan pengguna berdasarkan data aktivitas pada aplikasi *Habit Tracker*.

1.5 Manfaat Penelitian

Penelitian ini diharapkan dapat memberikan manfaat baik dari sisi teori maupun praktik, sebagai berikut:

1. Manfaat Teoritis
   a. Menambah wawasan dalam bidang ilmu komputer, khususnya penerapan *Machine Learning* dalam pengembangan aplikasi berbasis kebiasaan.
   b. Memberikan kontribusi ilmiah terkait integrasi prinsip pembentukan kebiasaan melalui *Atomic Habits* dengan teknologi analitik berbasis algoritma *Random Forest*.
   c. Menjadi referensi bagi penelitian selanjutnya yang berkaitan dengan pengembangan aplikasi *Habit Tracker* sebagai alat bantu pemantauan kebiasaan positif.

2. Manfaat Praktis
   a. Membantu mahasiswa dalam mencatat, memantau, dan mengevaluasi pembentukan kebiasaan positif secara lebih terstruktur.
   b. Menyediakan solusi digital berupa aplikasi *Habit Tracker* yang dapat digunakan sebagai alat bantu pemantauan kebiasaan harian mahasiswa.
   c. Memberikan gambaran kepada pengguna mengenai tingkat konsistensi kebiasaan berdasarkan data aktivitas yang tercatat pada aplikasi.
   d. Menjadi referensi bagi pengembang aplikasi serupa dalam mengintegrasikan prinsip *Atomic Habits* dengan algoritma *Machine Learning*.

1.6 Sistematika Penulisan

Sistematika penulisan skripsi ini adalah sebagai berikut:

BAB I PENDAHULUAN

Berisi latar belakang, rumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian, dan sistematika penulisan.

BAB II TINJAUAN PUSTAKA

Membahas teori-teori yang relevan, penelitian terdahulu, serta dasar-dasar mengenai *Atomic Habits*, *Habit Tracker*, dan algoritma *Random Forest.*

BAB III METODE PENELITIAN

Menjelaskan metode penelitian yang digunakan, mulai dari jenis penelitian, teknik pengumpulan data, perancangan sistem, hingga tahapan implementasi algoritma *Random Forest.*

BAB IV ANALISA DAN PERANCANGAN

Membahas analisa dan perancangan sistem yang dikembangkan. Pada bab ini dijelaskan analisa sistem yang sedang berjalan, analisa kebutuhan sistem, serta perancangan sistem yang diusulkan. Perancangan sistem meliputi perancangan *Unified Modeling Language (UML)* yang terdiri dari *Use Case Diagram, Activity Diagram,* dan *Class Diagram,* perancangan basis data (*Database*), serta perancangan antarmuka pengguna *(UI/UX)* aplikasi *Habit Tracker.*

BAB V HASIL DAN PEMBAHASAN

Membahas hasil implementasi dan pembahasan dari sistem yang telah dikembangkan. Pembahasan meliputi spesifikasi perangkat yang digunakan, yaitu perangkat keras (*hardware*) dan perangkat lunak (*Software*), implementasi basis data (*Database*), implementasi tampilan antarmuka (*user interface*), serta hasil pengujian sistem. Hasil pengujian disajikan dalam bentuk program yang dihasilkan, *Flowchart* alur sistem, dan potongan kode (*coding*) untuk memastikan sistem berjalan sesuai dengan perancangan yang telah dibuat.

BAB VI KESIMPULAN DAN SARAN

Berisi kesimpulan yang diperoleh berdasarkan hasil analisis, perancangan, implementasi, dan pengujian sistem yang telah

dilakukan dalam penelitian ini. Selain itu, bab ini juga memuat saran yang dapat dijadikan sebagai bahan pertimbangan untuk pengembangan sistem lebih lanjut serta rekomendasi bagi peneliti selanjutnya.

UNIVERSITAS PRABUMULIH
UNPRA
2000

BAB II
TINJAUAN PUSTAKA

2.1 Pengertian Implementasi

Implementasi adalah kegiatan yang direncanakan dan dilaksanakan untuk mencapai tujuan yang telah ditetapkan. Kegiatan yang dilaksanakan akan dilihat sejauh mana peranannya dalam mencapai tujuan yang hendak dicapai (Harmita & Aly, 2023). Implementasi merupakan kegiatan yang dilaksanakan untuk menilai, mengevaluasi dan mengukur apakah kebijakan tersebut dapat berjalan dengan semestinya dan membutuhkan penilaian atau tidak (Puspasari & Dafit, 2021).

Berdasarkan kedua teori tersebut, dapat disimpulkan bahwa implementasi merupakan kegiatan yang telah direncanakan dan dilaksanakan untuk mencapai tujuan tertentu. Selain itu, implementasi berperan dalam menilai dan mengevaluasi sejauh mana kebijakan atau program berjalan sesuai dengan harapan.

2.2 Pengertian Prinsip

Prinsip diartikan sebagai suatu pernyataan yang fundamental (mendasar) atau dijadikan sebagai acuan untuk berpikir mengenai kebenaran umum maupun individual (Arifah et al., 2024). Prinsip merupakan persoalan yang mendasar yang terdapat pada sebuah konsep atau teori (Nuraeni & Mujahidin, 2021).

Berdasarkan kedua teori tersebut, dapat disimpulkan bahwa Prinsip adalah landasan mendasar yang berfungsi sebagai pedoman dalam proses berpikir dan pengembangan suatu konsep atau teori, sehingga menjadi titik rujukan dalam menentukan kebenaran secara umum maupun individual.

2.3 Pengertian *Atomic Habits*

Atomic Habits merupakan kebiasaan kecil yang merupakan bagian dari sistem yang lebih besar. Kebiasaan bisa diibaratkan pedang yang mempunyai dua sisi, karena dapat menguntungkan dan merugikan. Kebiasaan merupakan bunga majemuk dalam perbaikan diri. Menjadi 1% lebih baik setiap hari dapat membantu untuk menghasilkan keuntungan dalam jangka panjang yang besar (Aulia &

Robiansyah, 2023). *Atomic Habits* adalah kebiasaan-kebiasaan yang merupakan bagian dari sistem yang lebih besar. Sama seperti atom-atom yang membentuk molekul, *Atomic Habits* adalah unsur pembentuk hasil-hasil yang luar biasa (Clear, 2018).

Berdasarkan kedua teori di atas, dapat disimpulkan bahwa *Atomic Habits* adalah kebiasaan kecil yang menjadi bagian dari sistem yang lebih besar dan memiliki dampak signifikan terhadap perubahan diri dalam jangka panjang. Setiap kebiasaan kecil berperan layaknya atom yang saling terhubung dan membentuk hasil besar melalui proses yang konsisten dan berkelanjutan. Konsep ini menekankan bahwa perubahan kecil yang dilakukan secara terus-menerus akan menghasilkan transformasi besar dalam kehidupan seseorang.

2.4 Pengertian *Random Forest*

Random Forest adalah algoritma klasifikasi berbasis *ensemble learning* yang membangun banyak *decision tree* dan menggabungkan prediksi mereka untuk meningkatkan akurasi dan stabilitas model (Asrul, 2025). *Random Forest* adalah metode dalam analisis yang terdiri dari beberapa pohon keputusan sebagai *classifier* (Larasati et al., 2022).

Berdasarkan kedua teori di atas, dapat disimpulkan bahwa *Random Forest* merupakan algoritma klasifikasi yang terdiri dari sejumlah pohon keputusan (*decision tree*) yang digabungkan untuk menghasilkan prediksi yang lebih akurat dan stabil. *Random Forest* bekerja dengan menggabungkan banyak *decision tree* untuk menghasilkan prediksi akhir yang lebih akurat dan stabil. Secara umum, proses kerja *Random Forest* meliputi langkah-langkah berikut:

1. *Bootstrap Sampling*

   Dataset asli diambil secara acak dengan pengembalian (*sampling with replacement*) untuk menghasilkan beberapa subset data. Setiap subset akan digunakan untuk melatih satu *decision tree* .
2. Pembangunan banyak pohon keputusan (*Decision Tree*). Setiap *tree* dilatih menggunakan subset data yang berbeda, sehingga struktur keputusan yang terbentuk akan bervariasi antara satu *tree* dan *tree* lainnya.

3. *Random Feature Selection*

   Pada setiap pemecahan node dalam *tree*, hanya sebagian fitur (kolom) yang dipilih secara acak sebagai kandidat split. Strategi ini membuat setiap pohon memiliki tingkat keragaman yang tinggi dan mencegah model mengalami *overfitting*.

4. Proses *Voting*

   Setelah seluruh *tree* selesai dilatih, setiap *tree* akan memberikan prediksi terhadap data uji. Prediksi akhir diambil berdasarkan mayoritas suara (*majority voting).*

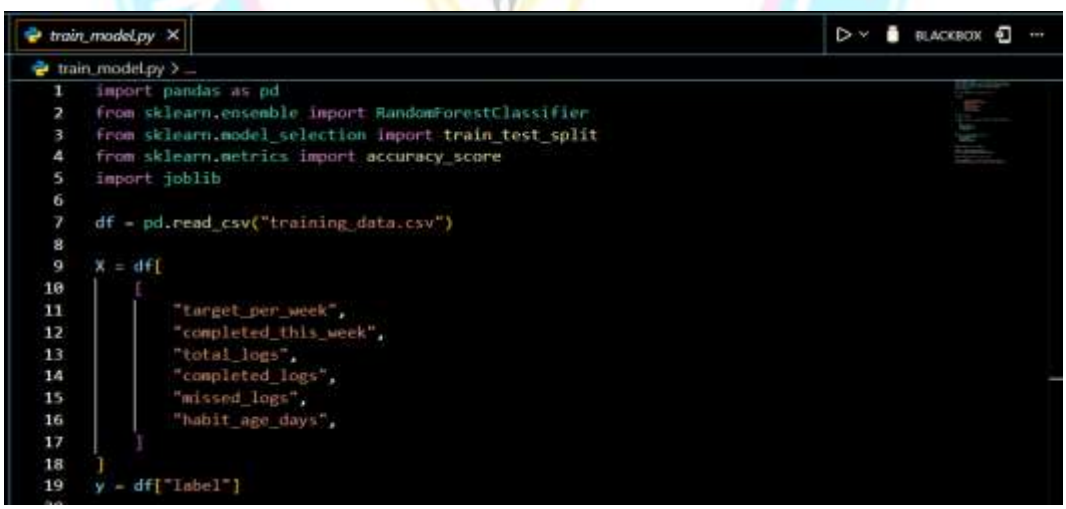
   Misalnya:

   1) 7 *tree* memprediksi "konsisten"
   2) 3 *tree* memprediksi "tidak konsisten" Maka hasil akhir adalah "konsisten".

5. Hasil Prediksi Akhir

   Prediksi gabungan dari seluruh *decision tree* terbukti lebih stabil dan akurat dibandingkan prediksi dari satu *tree* tunggal.

   Berikut contoh implementasi *Random Forest* untuk prediksi konsistensi menggunakan data sederhana.



```python
X_train, X_test, y_train, y_test = train_test_split(
    X,
    y,
    test_size=0.2,
    random_state=42,
    stratify=y,
)

model = RandomForestClassifier(
    n_estimators=200,
    max_depth=8,
    random_state=42,
)

model.fit(X_train, y_train)

y_pred = model.predict(X_test)
acc = accuracy_score(y_test, y_pred)

print(f"Akurasi model: {acc:.2f}")

joblib.dump(model, "random_forest_model.pkl")
print("Model disimpan ke random_forest_model.pkl")
```

Gambar 2. 1 Kode *Training Random Forest*

Gambar di atas merupakan contoh kode pelatihan algoritma *Random Forest* yang digunakan untuk memprediksi tingkat konsistensi kebiasaan pengguna pada aplikasi *Habit Tracker*. Dataset yang digunakan memuat beberapa fitur, yaitu target_per_week, completed_this_week, total_logs, completed_logs, missed_logs, dan *Habit*_age_days. Fitur-fitur tersebut digunakan sebagai variabel input, sedangkan kolom label digunakan sebagai target prediksi untuk menentukan apakah kebiasaan pengguna termasuk konsisten atau tidak konsisten.

```
PROBLEMS    OUTPUT    TERMINAL    PORTS

PS C:\Users\windi\ml_service>  python train_model.py
Akurasi model: 1.00
Model disimpan ke random_forest_model.pkl
```

Gambar 2. 2 Hasil *Training Random Forest*

Gambar di atas merupakan contoh hasil pelatihan model *Random Forest*. Output tersebut menunjukkan bahwa model berhasil dilatih dan disimpan ke dalam file *random_forest_model.pkl*. File model tersebut selanjutnya dapat digunakan

sebagai dasar dalam proses analisis atau pengembangan aturan prediksi konsistensi kebiasaan pada aplikasi *Habit Tracker*.

2.5 Pengertian Aplikasi

Aplikasi merupakan suatu perangkat lunak (*Software*) atau program komputer yang beroperasi pada sistem yang dibuat serta dikembangkan untuk melakukan perintah tertentu, Istilah aplikasi sendiri diambil dari Bahasa Inggris "*Application*" yang dapat diartikan sebagai penerapan atau penggunaan. Secara harfiah, aplikasi merupakan suatu penerapan perangkat lunak atau *Software* yang dikembangkan untuk tujuan melakukan tugas-tugas tertentu (Nasution M. Adlan Al Hawari et al., 2023). Istilah aplikasi berasal dari bahasa inggris "*application*" yang berarti penerapan, lamaran ataupun penggunaan. Sedangkan secara istilah, pengertian aplikasi adalah suatu program yang siap untuk digunakan yang dibuat untuk melaksanakan suatu fungsi bagi pengguna jasa aplikasi serta penggunaan aplikasi lain yang dapat digunakan oleh suatu sasaran yang akan dituju (Nasution M. Adlan Al Hawari et al., 2023).

Berdasarkan kedua teori di atas, dapat disimpulkan bahwa aplikasi merupakan perangkat lunak yang dirancang dan dikembangkan untuk menjalankan perintah atau fungsi tertentu sesuai kebutuhan pengguna. Aplikasi berfungsi sebagai sarana penerapan teknologi yang membantu manusia dalam menyelesaikan tugas-tugas secara lebih efektif dan efisien melalui sistem komputer.

2.6 Pengertian *Habit Tracker*

"Habit Tracker berarti pelacak kebiasaan merupakan alat yang digunakan untuk membantu mengingat perilaku masa depan yang direncanakan dan mempertahankan posisi dari waktu ke waktu, serta memprediksi perubahan perilaku berikutnya serta dapat membantu individu mengubah kebiasaan secara stabil agar dapat mencapai tujuan tertentu" (Mufidah, 2024) yang dikutip oleh (Muslimah & Subando, 2025). *Habit Tracker* dapat diartikan sebagai alat bantu pengingat, monitoring kebiasaan secara sengaja agar individu mendapatkan kontinuinitas dari aktivitas yang mereka lakukan. Serta individu dapat

meningkatkan motivasi dan penguatan nilai dalam pembentukan aktivitas positif (Muslimah & Subando, 2025b).

Berdasarkan kedua teori tersebut, dapat disimpulkan bahwa *Habit Tracker* adalah alat yang membantu memantau, mengingat, dan mengevaluasi kebiasaan yang dilakukan secara terus-menerus.

2.7 Pengertian Android

Android merupakan *mobile Operating System (OS)* yang dikembangkan oleh google (R. Fauzi et al., 2022). Android adalah perangkat *mobile* berbasis Java di mana Android adalah tempat untuk menjalankan aplikasi dan menyediakan *platform* terbuka/*open source* bagi para pengembang (Aulianti et al., 2021).

Berdasarkan kedua teori di atas, dapat disimpulkan bahwa Android merupakan sistem operasi *mobile* yang dikembangkan oleh *Google* dan bersifat *open source*, sehingga memungkinkan para pengembang untuk membuat dan menjalankan berbagai aplikasi di dalamnya.

2.8 Pengertian *Figma*

Figma adalah salah satu *tools* berbasis *web site* yang digunakan untuk mendesain kapan saja dan di manapun melalui internet (Nistrina & Lestari, 2024). *Figma* adalah aplikasi desain grafis berbasis *web* yang memungkinkan pengguna untuk membuat desain antarmuka pengguna/*User Interface (UI)* dan pengalaman pengguna/User Experience *(UX)* yang interaktif (Surianto et al., 2023).

Berdasarkan kedua teori tersebut, dapat disimpulkan bahwa *Figma* adalah aplikasi desain berbasis *web* yang memungkinkan pembuatan desain antarmuka *(UI/UX)* secara interaktif dan fleksibel, karena bisa diakses dan digunakan kapan saja serta di mana saja dengan koneksi internet.

2.9 Pengertian *Visual Studio Code*

Visual Studio Code yang lebih dikenal dengan *vs code* adalah sebuah editor kode gratis, didefinisikan ulang, lintas *platform* yang berjalan secara *native* di sistem operasi *OS X*, *Linux*, dan *Windows* (Safaat, 2021). *Visual Studio Code* adalah

kode editor sumber yang dikembangkan oleh *Microsoft* untuk *Windows*, *Linux* dan *mac OS*. *Visual Studio Code* memudahkan dalam penulisan *code* yang mendukung beberapa jenis bahasa pemrograman yang digunakan dan memberi variasi warna sesuai dengan fungsi dalam rangkaian *code* tersebut (Firnando et al., 2023).

Berdasarkan kedua teori tersebut, dapat disimpulkan bahwa *Visual Studio Code* merupakan editor kode gratis dari *Microsoft* yang kompatibel dengan berbagai sistem operasi dan mendukung beragam bahasa pemrograman.

2.10 Pengertian *Flutter*

Flutter adalah suatu *Framework* yang bersifat *open source*, dikembangkan oleh *Google* untuk membangun antarmuka (*user interface*) (P. R. Setiawan et al., 2025). *Flutter* adalah sebuah teknologi yang dibuat oleh *Google* untuk membangun aplikasi *mobile* (Android , *iOS*), *web* dan desktop (*Windows, Mac, Linux*) sekaligus dengan hanya melalui satu kode sumber saja, kode ini akan di-*compile* secara *native* (Safaat, 2021).

Berdasarkan kedua teori tersebut, dapat disimpulkan bahwa *Flutter* adalah *Framework open source* yang dibuat oleh *Google*, digunakan untuk mengembangkan antarmuka aplikasi *mobile*, *web* , dan desktop dengan satu basis kode.

2.11 Pengertian *Dart*

Dart adalah bahasa pemrograman terstruktur *open source* untuk membuat aplikasi *web* berbasis *browser* yang kompleks (Sofi & Dharmawan, 2022). *Dart* adalah bahasa pemrograman yang dikembangkan oleh *google* untuk kebutuhan dalam membuat aplikasi Android atau *mobile*, *front-end*, *web* , *IoT, back-end (CLI),* dan *Game*. *Dart* menerapkan konsep pemrograman berorientasi objek (*OOP*) di mana struktur kode berada dalam *class* yang di dalamnya berisi *method* maupun variabel (Suryana, 2021).

Berdasarkan kedua teori di atas, dapat disimpulkan bahwa *Dart* adalah bahasa pemrograman *open source* yang dikembangkan oleh *Google* untuk

membangun aplikasi *web, mobile*, dan berbagai *platform* lainnya, dengan menerapkan konsep pemrograman berorientasi objek (*OOP*).

2.12 Pengertian Android *Studio*

Android *Studio* adalah sebuah *Integrated Development Environment (IDE)* yang dipakai untuk membuat sebuah aplikasi*(Krismon et al., 2022).* Android *Studio* merupakan *Software* yang mempunyai *Integrated Development Environment (IDE),* yang digunakan untuk membuat aplikasi berbasis Android yang bergantung pada *Intellij IDEA* (Annisa et al., 2022).

Berdasarkan kedua teori di atas, dapat disimpulkan bahwa Android *Studio* adalah *Integrated Development Environment (IDE)* yang digunakan untuk mengembangkan aplikasi berbasis Android dan dibangun berdasarkan *IntelliJ IDEA.*

2.13 Pengertian *SQLite*

SQLite merupakan sistem manajemen basis data *relasional (RDBMS)* ringan dan portabel yang disematkan langsung dalam perangkat Android, sehingga tidak memerlukan *server* eksternal untuk menyimpan data (Kurnia, 2025). *SQLite* merupakan sebuah *embedded Database* yang artinya *SQLite* tertanam langsung pada aplikasi sehingga cocok digunakan untuk penyimpanan *end user*. Fitur-fitur yang disediakan oleh *SQLite* antara lain ringan, tidak memerlukan proses instalasi, bersifat *open source*, dan dapat digunakan pada berbagai macam *platform* (Nugraha & A. Susetyo, 2023).

Berdasarkan kedua teori di atas, dapat disimpulkan bahwa *SQLite* merupakan sistem manajemen basis data relasional yang bersifat ringan, portabel, dan tertanam langsung dalam aplikasi tanpa memerlukan *server* eksternal, sehingga cocok digunakan untuk penyimpanan data pada perangkat pengguna.

2.14 Penelitian Terdahulu

Penelitian terdahulu merupakan acuan penting dalam menyusun suatu penelitian karena memberikan gambaran mengenai hasil-hasil yang telah dilakukan

oleh peneliti sebelumnya. Melalui kajian ini, peneliti dapat mengetahui sejauh mana topik yang dikaji telah diteliti, pendekatan yang digunakan, serta hasil yang diperoleh sehingga dapat dijadikan dasar untuk mengembangkan penelitian yang lebih baik dan relevan.

Selain itu, penelitian terdahulu juga membantu peneliti dalam mengidentifikasi kesenjangan penelitian (*research gap*) yang dapat dijadikan peluang untuk memberikan inovasi baru. Dengan adanya tinjauan terhadap penelitian sebelumnya, penelitian ini diharapkan memiliki landasan teori yang kuat serta arah pengembangan yang jelas. Beberapa penelitian yang relevan dengan topik ini disajikan pada tabel berikut.

Tabel 2. 1 Penelitian Terdahulu

No	Judul Artikel	Penulis (Tahun)	Rangkuman
1	*Designing A Character Education Development Based On Atomic Habit By James Clear*	(Fahmi & Hidayah, 2025)	Penelitian ini mengembangkan pendidikan karakter berbasis *Atomic Habits* melalui empat tahap: mengidentifikasi nilai karakter, membentuk kebiasaan kecil, menerapkan empat hukum perubahan perilaku (*Cue*, *Craving*, *Response*, *reward*), dan melakukan evaluasi berkelanjutan. Pendekatan ini efektif dalam membangun karakter siswa melalui perubahan kecil yang konsisten.
2	*Habit Tracker* Sebagai Media Cetak untuk Membiasakan Aktivitas Produktif Santriwati di Pondok Pesantren	(Muslimah & Subando, 2025)	*Habit Tracker* merupakan media cetak yang efektif untuk mendampingi santriwati dalam membentuk kebiasaan positif dan meningkatkan produktivitas. Selain sebagai pengingat aktivitas, alat ini juga berfungsi sebagai sarana refleksi diri dan penguatan spiritual. Dengan pendekatan teori *behavioristik* dan nilai-nilai Islam, media ini membantu memperkuat pembentukan karakter yang telah dilakukan secara verbal oleh pengasuh pesantren serta mendorong kemandirian dan disiplin santriwati dalam menjalani aktivitas sehari-hari secara konsisten.
3	Implementasi Metode *Random Forest* pada Kategori	(Mahmuda, 2024)	Penelitian ini menemukan bahwa jumlah penayangan video pada kanal *YouTube* berbanding lurus dengan jumlah subscribers. Metode klasifikasi *Random Forest* dengan parameter mtry

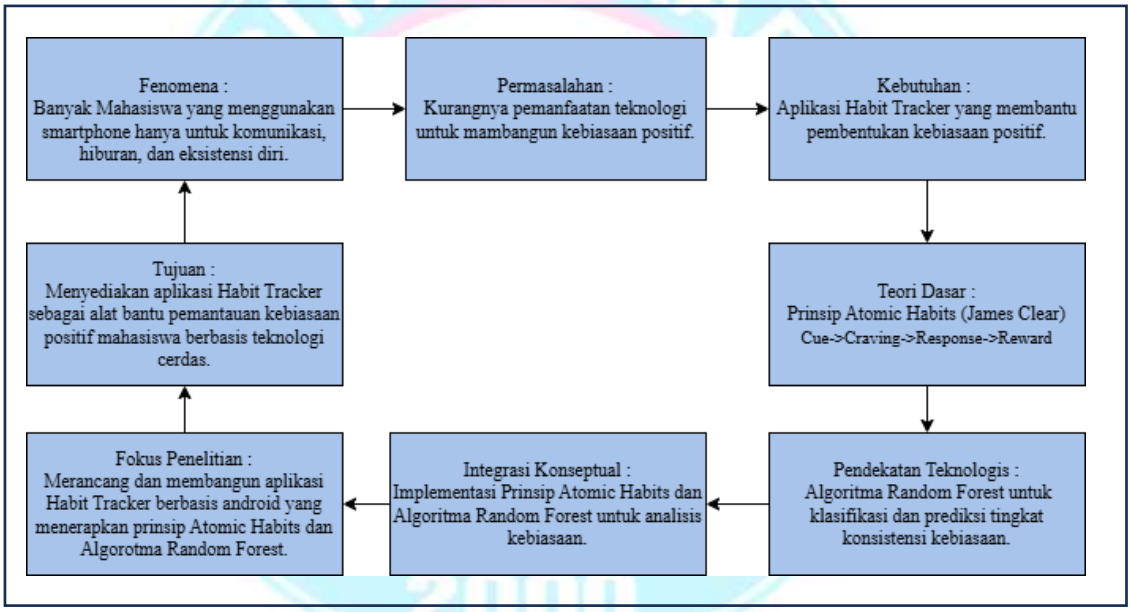
	Konten Kanal *YouTube*		1 dan jumlah pohon 100 menghasilkan akurasi klasifikasi sebesar 77,27%, yang menunjukkan performa baik dalam mengklasifikasikan kategori konten kanal *YouTube*. Variabel yang paling berpengaruh dalam pengkategorian adalah jumlah subscribers, diikuti oleh jumlah video, jumlah penayangan video, dan lama akun *YouTube*. Metode ini efektif untuk mengkategorikan konten kanal *YouTube* dengan tingkat akurasi yang tinggi.
4	Implementasi Metode *Random Forest* Untuk Memprediksi Lama Studi Siswa	(Akbar et al., 2023)	Penelitian ini menyimpulkan bahwa metode *Random Forest* mampu memprediksi lama studi mahasiswa dengan tingkat akurasi yang tinggi. Berdasarkan hasil pengujian menggunakan 10-fold cross-validation dan 100 pohon keputusan, model mencapai akurasi rata-rata 95,12%, dengan nilai spesifisitas dan sensitivitas yang juga tergolong baik. Variabel IPK semester 8 terbukti menjadi faktor paling berpengaruh dalam menentukan kelulusan tepat waktu. Hasil ini menunjukkan bahwa *Random Forest* efektif digunakan sebagai alat bantu bagi perguruan tinggi dalam memantau dan meningkatkan ketepatan waktu kelulusan mahasiswa.
5	Kebiasaan Belajar Mahasiswa Ditinjau Dari Self-Management dan Penyesuaian Diri	(Reba et al., 2021)	Penelitian ini menemukan bahwa self-management tidak berpengaruh signifikan terhadap kebiasaan belajar, sedangkan penyesuaian diri memiliki pengaruh yang cukup kuat. Secara bersama-sama, keduanya berkontribusi 35% terhadap kebiasaan belajar mahasiswa, dengan penekanan pada pentingnya penyesuaian diri akademik dalam membentuk kebiasaan belajar yang efektif.
6	Landasan dan Prinsip-Prinsip Perencanaan Pendidikan Islam	(Nuraeni & Mujahidin, 2021)	Pendidikan Islam bertujuan membentuk manusia beriman, bertakwa, dan berakhlak mulia dengan keseimbangan antara kehidupan dunia dan akhirat. Penelitian ini menggunakan studi pustaka dan menekankan bahwa perencanaan pendidikan harus didasari prinsip ikhlas, keseimbangan,

			musyawarah, integrasi ilmu agama dan umum, wajib belajar seumur hidup, profesionalisme, serta berbasis riset dan inovasi. Dengan landasan dan prinsip tersebut, pendidikan Islam diharapkan mampu melahirkan generasi berilmu, berkarakter, dan bertanggung jawab dunia akhirat.
7	Perancangan Sistem Prediksi Deteksi Alzheimer Berbasis Random Forest Menggunakan Metode *Scrum*	(Syahfitri et al., 2025)	Penelitian ini mengembangkan sistem web *MindCare* menggunakan algoritma *Random Forest* untuk memprediksi risiko penyakit *Alzheimer*. Pengembangan sistem menerapkan metode *Agile* Scrum, sedangkan hasil pengujian menunjukkan model memiliki akurasi 100% dan seluruh fitur sistem berfungsi dengan baik.
	AI-Based Habit Reminder For Mewing Exercise: Pengembangan Aplikasi Mobile Berbasis Kecerdasan Buatan untuk Meningkatkan Konsistensi Praktik Mewing	(Alfiansyah et al., 2026)	Penelitian ini mengembangkan aplikasi mobile *MewingAI* berbasis *Flutter* dengan algoritma Random Forest untuk memberikan pengingat kebiasaan yang dipersonalisasi. Hasil penelitian menunjukkan aplikasi berhasil meningkatkan konsistensi latihan mewing sebesar 40,6% dan memperoleh nilai *System Usability Scale* (SUS) sebesar 80,3 dengan kategori Good.

S*umber:Diolah oleh Penulis (2026)*

2.15 Kerangka Berpikir Penelitian

Kerangka berpikir penelitian adalah panduan yang digunakan peneliti untuk menjalankan penelitian secara terarah dan sistematis. Melalui kerangka ini, peneliti dapat menggambarkan hubungan antara teori, konsep, dan variabel yang diteliti, serta menjelaskan alur berpikir dalam mencapai tujuan penelitian. Kerangka penelitian juga memudahkan peneliti memahami langkah-langkah yang perlu dilakukan mulai dari identifikasi masalah, pengumpulan data, analisis, hingga penarikan kesimpulan. Dengan adanya kerangka penelitian, diharapkan hasil yang diperoleh menjadi lebih fokus, terukur, dan dapat memberi kontribusi dalam pengembangan ilmu pengetahuan.



Sumber: Diolah oleh Penulis (2026)

Gambar 2. 3 Kerangka Berfikir Penelitian

Gambar 2.3 menunjukkan kerangka berpikir penelitian yang menggambarkan alur logis dalam pengembangan aplikasi *Habit Tracker* berbasis Android dengan menerapkan prinsip *Atomic Habits* dan algoritma *Random Forest*. Fenomena yang melatarbelakangi penelitian ini adalah banyaknya mahasiswa yang menggunakan smartphone hanya untuk komunikasi, hiburan, dan eksistensi diri, sementara pemanfaatannya sebagai alat bantu untuk membangun kebiasaan positif belum dilakukan secara optimal. Kondisi tersebut menunjukkan bahwa teknologi mobile memiliki potensi untuk dimanfaatkan sebagai media yang lebih produktif, khususnya dalam membantu mahasiswa memantau pembentukan kebiasaan positif.

Permasalahan utama dalam penelitian ini adalah kurangnya pemanfaatan teknologi untuk membantu mahasiswa membangun kebiasaan positif secara terarah. Berdasarkan permasalahan tersebut, penelitian ini mengembangkan aplikasi *Habit Tracker* berbasis Android yang menerapkan prinsip *Atomic Habits* sebagai pendekatan dalam membangun kebiasaan positif. Aplikasi memungkinkan pengguna membuat target kebiasaan, mencatat progress harian, memantau perkembangan melalui halaman statistik, serta memperoleh penghargaan berupa *Diamond Badge* apabila berhasil mencapai atau melebihi target *progress* yang telah ditentukan. Selain itu, aplikasi memanfaatkan algoritma *Random Forest* untuk menganalisis data progress pengguna selama tujuh hari terakhir sehingga dapat memberikan prediksi tingkat konsistensi pengguna dalam kategori Konsisten atau Tidak Konsisten, disertai rekomendasi otomatis yang membantu pengguna meningkatkan kualitas kebiasaannya.Teori dasar yang digunakan dalam penelitian ini adalah prinsip *Atomic Habits* dari James Clear, yaitu *Cue*, *Craving*, *Response*, dan *Reward*. Prinsip *Cue* diterapkan melalui fitur notifikasi sebagai pengingat kebiasaan, *Craving* diterapkan melalui tampilan antarmuka yang menarik, *Response* diterapkan melalui tombol penanda atau *Checklist Habit*, sedangkan *Reward* diterapkan melalui sistem poin sebagai bentuk penghargaan atas kebiasaan yang berhasil diselesaikan. Keempat prinsip tersebut menjadi dasar dalam perancangan fitur aplikasi *Habit Tracker*.

Selain menggunakan prinsip Atomic *Habit*s, penelitian ini juga menerapkan pendekatan teknologi berupa algoritma *Random Forest*. Algoritma *Random Forest* digunakan untuk melakukan klasifikasi dan prediksi tingkat konsistensi kebiasaan berdasarkan data aktivitas pengguna, seperti target per minggu, jumlah *Habit* yang selesai, jumlah *Habit* yang terlewat, total log, usia *Habit*, dan total poin. Hasil dari proses ini digunakan untuk memberikan gambaran konsistensi kebiasaan kepada pengguna.

Integrasi konseptual dalam penelitian ini terletak pada penggabungan prinsip *Atomic Habits* dan algoritma *Random Forest*. Prinsip *Atomic Habits* digunakan sebagai dasar pembentukan fitur kebiasaan, sedangkan *Random Forest* digunakan sebagai alat analisis untuk memberikan prediksi atau penilaian terhadap tingkat konsistensi kebiasaan. Dengan integrasi tersebut, aplikasi *Habit Tracker*

tidak hanya berfungsi sebagai pencatat kebiasaan, tetapi juga sebagai alat bantu pemantauan dan evaluasi berbasis data.

Fokus penelitian ini adalah merancang dan membangun aplikasi *Habit Tracker* berbasis Android yang menerapkan prinsip *Atomic Habits* dan algoritma *Random Forest*. Tujuan dari penelitian ini adalah menyediakan aplikasi *Habit Tracker* sebagai alat bantu pemantauan kebiasaan positif mahasiswa berbasis teknologi cerdas. Dengan demikian, kerangka berpikir ini menunjukkan hubungan antara fenomena, permasalahan, kebutuhan aplikasi, teori dasar, pendekatan teknologi, integrasi konsep, fokus penelitian, dan tujuan akhir penelitian.

BAB III
METODE PENELITIAN

3.1 Objek dan Jadwal Penelitian

Bagian ini memaparkan objek serta jadwal pelaksanaan penelitian. Berikut penjelasan mengenai objek dan waktu pelaksanaan penelitian:

3.1.1 Objek Penelitian

Objek penelitian adalah sasaran yang dikaji secara sistematis untuk memperoleh data yang sesuai dengan tujuan penelitian. Dalam penelitian ini, objek utama yang dikaji adalah kebutuhan mahasiswa terhadap aplikasi *Habit Tracker* sebagai alat bantu pemantauan pembentukan kebiasaan positif. Penelitian ini melibatkan 75 mahasiswa Program Studi Informatika Universitas Prabumulih angkatan 2022 dan 2023 sebagai responden kuesioner untuk menggali permasalahan, kebutuhan, serta harapan pengguna terhadap aplikasi yang dikembangkan.

Mahasiswa dipilih sebagai responden karena memiliki aktivitas akademik yang cukup padat dan membutuhkan pengelolaan waktu serta pemantauan kebiasaan secara lebih terstruktur. Data yang diperoleh dari mahasiswa digunakan sebagai dasar dalam merancang kebutuhan sistem, menentukan fitur aplikasi, serta menyesuaikan penerapan prinsip *Atomic Habits* dalam aplikasi *Habit Tracker*. Dengan demikian, keterlibatan mahasiswa dalam penelitian ini difokuskan pada tahap pengumpulan data kebutuhan pengguna.

Aplikasi *Habit Tracker* yang dikembangkan dalam penelitian ini bertujuan sebagai alat bantu berbasis Android untuk mencatat, memantau, dan mengevaluasi pembentukan kebiasaan positif mahasiswa. Aplikasi ini menerapkan prinsip *Atomic Habits* melalui fitur notifikasi, tampilan antarmuka, tombol penanda kebiasaan, dan sistem poin. Selain itu, algoritma *Random Forest* digunakan untuk memberikan prediksi atau penilaian terhadap tingkat konsistensi kebiasaan berdasarkan data aktivitas yang tercatat pada aplikasi.

3.1.2 Waktu dan Jadwal Penelitian

Penelitian ini dijadwalkan berlangsung selama 6 bulan dengan rincian sebagai berikut.

Tabel 3. 1 Jadwal Penelitian

No	Kegiatan	Bulan Ke 1	Bulan Ke 2	Bulan Ke 3	Bulan Ke 4	Bulan Ke 5	Bulan Ke 6
1	Pengumpulan data dan studi literatur	✓	✓				
2	Analisis kebutuhan sistem		✓				
3	Perancangan sistem		✓	✓			
4	Implementasi aplikasi			✓	✓		
5	Integrasi algoritma *Random Forest*				✓	✓	
6	Pengujian sistem					✓	
7	Penyusunan laporan akhir					✓	✓

Sumber: Diolah oleh Penulis(2026)

Pada Tabel 3.1 di atas, dapat dijelaskan bahwa Penelitian ini dilaksanakan secara bertahap selama enam bulan. Tahapan penelitian meliputi pengumpulan data dan studi literatur, analisis kebutuhan sistem, perancangan sistem, implementasi aplikasi, integrasi algoritma *Random Forest*, pengujian sistem, serta penyusunan laporan akhir. Jadwal penelitian disusun untuk memastikan setiap tahapan penelitian dapat dilakukan secara terarah dan sistematis.

3.1.3 Sejarah, Visi Misi, dan Tujuan Universitas Prabumulih

a. Sejarah Universitas Prabumulih

Universitas Prabumulih, disingkat UNPRA, adalah perguruan tinggi swasta yang berlokasi di Prabumulih, Sumatera Selatan, Indonesia. Berdiri sejak tahun 2000 di bawah naungan Yayasan Pendidikan Prabumulih, universitas ini terbentuk dari penggabungan Sekolah Tinggi Ilmu Ekonomi Prabumulih (STIE Prabumulih), Sekolah Tinggi Ilmu Teknik Prabumulih (STIT Prabumulih), dan Sekolah Tinggi Manajemen Informatika dan Komputer Prabumulih (STMIK Prabumulih), berdasarkan surat keputusan menteri pendidikan, kebudayaan, riset, dan teknologi Republik Indonesia nomor 254/e/o/2022 tanggal 07 April 2022. Awalnya, Yayasan Pendidikan Prabumulih memiliki tiga sekolah tinggi:

1. Sekolah Tinggi Ilmu Ekonomi Prabumulih (STIE Prabumulih).
2. Sekolah Tinggi Manajemen Informatika dan Komputer Prabumulih (STMIK Prabumulih).
3. Sekolah Tinggi Ilmu Teknik Prabumulih (STIT Prabumulih).

Pada tahun 2022, ketiga sekolah tinggi ini digabungkan menjadi Universitas Prabumulih dengan tiga fakultas, yaitu: Fakultas Ekonomi dan Bisnis (FEB), Fakultas Teknik (FT), dan Fakultas Ilmu Komputer (FASILKOM). STIE Prabumulih menjadi FEB, STIT Prabumulih menjadi FT, dan STMIK Prabumulih menjadi FASILKOM).

b. Visi Universitas Prabumulih

Visi Universitas Prabumulih adalah menjadikan universitas ini sebagai perguruan tinggi terkemuka pada tahun 2029, yang mampu menghasilkan sumber daya manusia berkualitas dan bertanggung jawab dalam penguasaan serta pengembangan ilmu pengetahuan, teknologi, dan seni.

c. Misi Universitas Prabumulih

Berikut adalah misi dari Universitas Prabumulih:

1. Menyelenggarakan kegiatan pendidikan dan pengajaran berkualitas untuk menghasilkan sumber daya manusia yang kompeten;
2. Menghasilkan lulusan yang sesuai dengan kebutuhan dunia kerja;

3. Menciptakan lulusan yang kompetitif secara global dan profesional dalam ilmu pengetahuan, teknologi, dan seni;
4. Meningkatkan kualitas pelaksanaan tridharma perguruan tinggi;
5. Mengembangkan kemitraan dengan lembaga dan/atau instansi lain untuk menguatkan kegiatan tridharma;
6. Mengoptimalkan pemanfaatan sarana dan prasarana pendidikan secara dinamis, efektif, dan efisien produktiktif.

d. Tujuan Universitas Prabumulih

Adapun tujuan Universitas Prabumulih adalah sebagai berikut :

1. Membentuk dan menghasilkan lulusan yang mampu menjunjung tinggi almamaternya, berkepribadian Indonesia, bermoral tinggi. berbudi luhur, berwawasan kebangsaan serta tanggap terhadap keadaan tantangan dan perubahan yang timbul dan mampu menemukan solusi yang terbaik untuk mengatasi tantangan dan perubahan dalam kerangka turut bertanggung jawab atas kelangsungan hidup Bangsa dan Negara Kesatuan Republik Indonesia berdasarkan Panicasila dan UUD 1945;
2. Membentuk dan menghasilkan tenaga ahli yang mampu berpikir ilmiah, profesional, berbudi luhur, terampil, berjiwa wiraswasta serta berperan aktif dalam pembangunan nasional dengan berpegang teguh pada etika profesi;
3. Membentuk dan menghasilkan sarjana yang memiliki integritas tinggi, yang mampu memelihara, mengembangkan ilmu pengetahuan dan yang mampu berperan aktif dalam pengembangan budaya, ilmu pengetahuan, teknologi dan senı;
4. Mengembangkan dan menyebarluaskan ilmu pengetahuan teknologi dan seni serta mengupayakan penggunaannya untuk meningkatkan taraf hidup masyarakat dan memperkaya kebudayaan nasional;
5. Menghasilkan peneiitian yang memiliki karakteristik sebaga berikut:
   a. Bermanfaat bagi perkembangan iptek dan mendukung peningkatan daya saing bangsa;
   b. Bermanfaat bagi kesejahteraan masyarakat;

c. Bernilai publikasi, baik di tingkat nasional maupun internasional;

6. Melaksanakan kegiatan pengabdian kepada masyarakat dengan karakteristik sebagai berikut:
   a. Bermanfaat dalam memberdayakan potensi masyarakat;
   b. Mampu menumbuhkan partisipasi aktif masyarakat dalam upaya memecahkan masalah;
   c. Mampu meningkatkan kesejahteraan masyarakat.

3.2 Metode Penelitian

Metode penelitian merupakan cara ilmiah untuk mendapatkan data dengan tujuan dan kegunaan tertentu (Setiawan Iwan et al., 2020). Dalam penelitian ini, penulis menggunakan metode deskriptif melalui pendekatan kualitatif. Deskriptif dalam penelitian kualitatif berarti menggambarkan dan menjabarkan peristiwa, fenoma dan situasi sosial yang diteliti. Penelitian kualitatif merupakan suatu teknik penelitian yang menggunakan narasi atau kata-kata dalam menjelaskan dan menjabarkan makna dari setiap fenomena, gejala, dan situasi sosial tertentu (Waruwu, 2023).

Pendekatan deskriptif kualitatif dalam penelitian ini digunakan untuk mengeksplorasi dan memahami masalah serta kebutuhan mahasiswa terkait pembangunan kebiasaan belajar, berdasarkan data naratif dari responden. Hasil analisis ini menjadi dasar dalam merancang fitur aplikasi *Habit Tracker*, yang menggabungkan prinsip *Atomic Habits* dan algoritma *Random Forest*.

3.3 Sumber Data

Sumber data adalah asal informasi yang digunakan peneliti untuk memahami kebutuhan pengguna, pola kebiasaan mahasiswa, serta proses perancangan dan pengembangan aplikasi *Habit Tracker*. Dalam penelitian ini, peneliti memanfaatkan dua tipe sumber data, yaitu sumber data primer dan sekunder.

1. Sumber Data Primer

Data primer merupakan data yang diperoleh secara langsung dari sumber utama penelitian. Dalam penelitian ini, data primer diperoleh melalui kuesioner

kualitatif yang diberikan kepada mahasiswa Program Studi Informatika Universitas Prabumulih angkatan 2022 dan 2023. Kuesioner digunakan untuk menggali permasalahan mahasiswa dalam membentuk kebiasaan positif, mengetahui kebutuhan fitur aplikasi *Habit Tracker*, serta memperoleh gambaran mengenai pandangan mahasiswa terhadap penerapan prinsip *Atomic Habits* dan fitur prediksi konsistensi berbasis algoritma *Random Forest*.

Data primer dalam penelitian ini digunakan sebagai dasar analisis kebutuhan sistem dan perancangan fitur aplikasi. Data tersebut tidak digunakan untuk mengukur peningkatan konsistensi mahasiswa setelah menggunakan aplikasi, karena penelitian ini berfokus pada perancangan, pembangunan, dan pengujian teknis aplikasi *Habit Tracker*.

2. Sumber Data Sekunder

Sumber data sekunder diperoleh dari berbagai literatur yang relevan dengan topik penelitian. Sumber data sekunder bertujuan memperkuat landasan teori dan mendukung proses perancangan sistem. Sumber ini meliputi:

a. Jurnal ilmiah mengenai *Habit Tracker,* manajemen waktu, dan perilaku kebiasaan.
b. Jurnal dan buku mengenai prinsip *Atomic Habits* oleh James Clear.
c. Jurnal dan buku mengenai algoritma *Random Forest* serta penerapannya pada sistem rekomendasi.

3.4 Teknik Pengumpulan Data

Pengumpulan data merupakan suatu hal yang penting dalam penelitian, karena berbagai cara digunakan oleh peneliti untuk mengumpulkan data yang diperlukan dalam penelitiannya. Pengumpulan data dalam penelitian ini dimaksudkan untuk memperoleh bahan-bahan, keterangan, kenyataan-kenyataan dan informasi yang dapat dipercaya (Setiawan Iwan et al., 2020). Dalam penelitian ini, teknik pengumpulan data dipakai untuk mengidentifikasi permasalahan dan kebutuhan pengguna sebagai dasar pengembangan aplikasi *Habit Tracker*, serta untuk mendapatkan data pendukung terkait konteks penelitian. Teknik pengumpulan data yang digunakan meliputi:

1. Observasi

Observasi merupakan suatu cara pengumpulan data melalui pengamatan langsung terhadap objek yang diteliti guna memperoleh gambaran kondisi nyata di lapangan (Zanariyah Sri, 2024). Pada tahap ini, peneliti melakukan pengamatan langsung di lingkungan Universitas Prabumulih yang dikhususkan pada mahasiswa prodi Informatika angkatan 2022 dan angkatan 2023 untuk mengamati aktivitas perkuliahan dan perilaku umum mahasiswa. Observasi ini bertujuan sebagai data pendukung untuk memahami konteks penelitian dan lingkungan akademik tempat penelitian dilakukan.

2. Wawancara

Wawancara dilakukan dengan staf administrasi rektorat Universitas Prabumulih ibu Ranti Yuliningsih, A.Md pada tanggal 10 November 2025 sebagai sumber informasi. Tujuannya adalah untuk mendapatkan data umum dan institusional terkait objek penelitian, seperti sejarah singkat, visi, misi, dan tujuan Universitas Prabumulih. Wawancara ini tidak diarahkan untuk mengungkapkan masalah pengguna secara spesifik, tetapi berfungsi sebagai pelengkap data kontekstual dalam penelitian.

3. Studi Literatur

Pada tahap ini, peneliti mengumpulkan data dan informasi dari berbagai sumber diantaranya buku, artikel, jurnal, *ebook* serta penelitian-penelitian terdahulu sebagai referensi untuk penelitian ini.

4. Kuisioner

Kuisioner merupakan suatu cara pengumpulan data, di mana peneliti bertanya jawab dengan responden menggunakan angket (daftar pertanyaan) (Setiawan Iwan et al., 2020). Dalam penelitian ini, kuisioner yang digunakan bersifat kualitatif dan terbuka, serta dirancang sebagai pengganti wawancara pengguna. Kuisioner disusun dalam bentuk pertanyaan terbuka yang memungkinkan responden menyampaikan pengalaman, persepsi, dan kebutuhannya secara naratif tanpa dibatasi oleh pilihan jawaban tertentu.

Kuesioner ini digunakan untuk memperoleh informasi mengenai kebiasaan belajar mahasiswa, pengelolaan waktu, kendala dalam membentuk kebiasaan positif, kebutuhan fitur aplikasi *Habit Tracker*, pandangan terhadap penerapan prinsip Atomic *Habit*s, serta persepsi terhadap fitur prediksi konsistensi berbasis algoritma *Random Forest*. Data kuesioner digunakan sebagai dasar analisis kebutuhan sistem dan perancangan fitur aplikasi.

Kuisioner ini dibagikan kepada mahasiswa Informatika angkatan 2022 dan 2023 melalui *link Google Form* berikut*: https://forms.gle/HrLuAn7NrUv7YwnW9*. Karena bersifat kualitatif, kuisioner ini tidak menggunakan skala numerik, pilihan ganda, maupun analisis statistik, sehingga tetap selaras dengan pendekatan penelitian deskriptif kualitatif. Data yang diperoleh dari kuisioner digunakan untuk:

1. Memahami pola perilaku mahasiswa terkait pembentukan kebiasaan.
2. Mengidentifikasi kebutuhan dan ekspektasi pengguna terhadap aplikasi *Habit Tracker.*
3. Mengetahui persepsi mahasiswa mengenai penerapan prinsip *Atomic Habits* dalam aplikasi.
4. Menggali tanggapan mahasiswa terkait pemanfaatan algoritma *Random Forest* sebagai fitur rekomendasi kebiasaan.

3.4.1 Teknik Analisis Data Kuesioner

Data kuesioner dalam penelitian ini dianalisis menggunakan pendekatan deskriptif kualitatif. Analisis dilakukan untuk mengolah jawaban responden menjadi informasi yang dapat digunakan sebagai dasar perancangan aplikasi *Habit Tracker*. Tahapan analisis data kuesioner dalam penelitian ini adalah sebagai berikut:

1. Reduksi Data

   Reduksi data dilakukan dengan membaca seluruh jawaban responden, kemudian menyaring informasi yang relevan dengan kebutuhan penelitian. Jawaban yang tidak berhubungan langsung dengan kebiasaan mahasiswa, kebutuhan aplikasi, atau fitur *Habit Tracker* tidak dijadikan fokus utama analisis.

2. Pengelompokan Data

   Setelah data direduksi, jawaban responden dikelompokkan berdasarkan tema tertentu. Pengelompokan dilakukan untuk memudahkan peneliti memahami pola jawaban responden. Tema yang digunakan dalam penelitian ini meliputi kebiasaan belajar mahasiswa, pengelolaan waktu, kendala dalam membentuk kebiasaan, kebutuhan fitur aplikasi, pandangan terhadap Atomic *Habit*s, dan persepsi terhadap fitur prediksi konsistensi.

3. Pemberian Kode

   Pemberian kode dilakukan dengan memberikan tanda atau kategori pada jawaban responden yang memiliki makna serupa. Misalnya, jawaban yang berkaitan dengan kebutuhan pengingat dikelompokkan ke dalam kode "notifikasi", jawaban yang berkaitan dengan pemantauan perkembangan dikelompokkan ke dalam kode "statistik", dan jawaban yang berkaitan dengan motivasi dikelompokkan ke dalam kode "*reward*".

4. Perhitungan Jumlah Responden per Kategori

   Setelah jawaban dikelompokkan dan diberi kode, peneliti menghitung jumlah responden pada setiap kategori. Perhitungan ini digunakan untuk mengetahui kecenderungan kebutuhan pengguna terhadap fitur tertentu. Hasil perhitungan tidak digunakan sebagai analisis statistik inferensial, melainkan sebagai data pendukung dalam menentukan prioritas fitur aplikasi.

5. Penarikan Kesimpulan

   Tahap terakhir adalah penarikan kesimpulan berdasarkan hasil pengelompokan dan perhitungan kategori. Kesimpulan digunakan untuk menentukan kebutuhan utama aplikasi *Habit Tracker*, seperti fitur pencatatan kebiasaan, notifikasi pengingat, *Checklist* aktivitas, statistik perkembangan, sistem poin, dan prediksi konsistensi menggunakan algoritma *Random Forest*.

3.5 Metode Pengembangan Sistem

Metode pengembangan sistem yang digunakan pada penelitian ini adalah metode *Agile.* Metode *Agile* adalah pendekatan pengembangan yang

mengedepankan peningkatan bertahap, pelepasan perangkat lunak secara bertahap, pengurangan kerumitan proses, pembuatan kode berkualitas tinggi, serta keterlibatan aktif pelanggan dalam proses pengembangan (Hidayah & Asnadi, 2024). Metode *Agile* merupakan metode yang didasarkan pada prinsip-prinsip iteratif dan incremental, yang berarti bahwa pengembangan perangkat lunak dilakukan secara bertahap dan berulang-ulang sehingga setiap proses pengembangan dapat disesuaikan dengan kebutuhan sistem. Metode ini mendukung proses perencanaan, analisis kebutuhan, perancangan, pengembangan, pengujian, implementasi, dan pemeliharaan sistem secara fleksibel. Metode *Agile* dimulai dari tahap perencanaan, analisis kebutuhan, perancangan, uji coba, implementasi, dan pemeliharaan. Tahapan tersebut sesuai dengan kebutuhan penelitian ini karena aplikasi *Habit Tracker* dikembangkan secara bertahap, mulai dari pengumpulan kebutuhan pengguna, perancangan sistem, pembuatan aplikasi, integrasi algoritma *Random Forest*, pengujian sistem, hingga penyempurnaan aplikasi berdasarkan hasil pengujian (Vindua, 2024).

Metode *Agile* dipilih karena sesuai dengan penelitian pengembangan aplikasi yang membutuhkan proses iteratif dan fleksibel. Dalam penelitian ini, *Agile* digunakan untuk mengembangkan aplikasi *Habit Tracker* berbasis Android sebagai alat bantu pemantauan pembentukan kebiasaan positif mahasiswa. Data kebutuhan pengguna diperoleh melalui kuesioner kepada mahasiswa Program Studi Informatika Universitas Prabumulih angkatan 2022 dan 2023. Data tersebut digunakan sebagai dasar dalam menentukan fitur aplikasi, bukan sebagai pengujian efektivitas aplikasi kepada pengguna.

Pada penelitian ini, evaluasi dalam metode *Agile* tidak diarahkan pada uji coba aplikasi kepada mahasiswa, melainkan dilakukan melalui pemeriksaan mandiri, proses debugging, hasil pengujian sistem, dan arahan pembimbing. Hal ini dilakukan agar aplikasi yang dikembangkan dapat berjalan sesuai dengan rancangan dan memenuhi kebutuhan utama yang telah diperoleh dari hasil analisis kebutuhan pengguna.

Sumber: (Vindua, 2024)

Gambar 3.1 Tahapan Metode *Agile*

Dari gambar 3.1 di atas menjelaskan mengenai tahapan pada metode *Agile* yang digunakan dalam penelitian ini dan telah disesuaikan dengan proses pengembangan aplikasi *Habit Tracker,* adapun tahapannya sebagai berikut :

1. Perencanaan (*Plan*)

   Tahap perencanaan dilakukan untuk menentukan arah pengembangan aplikasi *Habit Tracker*. Pada tahap ini peneliti mengidentifikasi permasalahan yang berkaitan dengan kebutuhan mahasiswa terhadap alat bantu pemantauan kebiasaan positif. Peneliti juga menentukan tujuan pengembangan sistem, ruang lingkup aplikasi, kebutuhan data, fitur utama, serta teknologi yang digunakan dalam membangun aplikasi.

2. Analisis dan Perancangan (*Design*)

   Tahap Design pada penelitian ini difokuskan pada analisis kebutuhan dan perancangan sistem. Analisis kebutuhan dilakukan berdasarkan hasil observasi, wawancara, studi literatur, dan kuesioner yang diberikan kepada 75 mahasiswa Program Studi Informatika Universitas Prabumulih angkatan 2022 dan 2023. Data tersebut digunakan untuk mengetahui kebutuhan pengguna terhadap aplikasi *Habit Tracker*, seperti kebutuhan pencatatan kebiasaan, pengingat aktivitas, pemantauan perkembangan, dan evaluasi kebiasaan. Setelah kebutuhan sistem diperoleh, dilakukan perancangan sistem yang meliputi

pembuatan *Use Case Diagram*, *Activity Diagram*, dan *Class Diagram*, perancangan basis data *SQLite*, serta perancangan antarmuka pengguna menggunakan *Figma*. Tahap ini bertujuan agar proses pengembangan aplikasi dapat berjalan lebih terarah dan sesuai dengan kebutuhan sistem.

3. Pengembangan (*Develop*)

   Tahap Develop merupakan tahap pembangunan aplikasi *Habit Tracker*. Pada tahap ini, sistem mulai dikembangkan menggunakan *Flutter* dengan bahasa pemrograman *Dart*, serta *SQLite* sebagai media penyimpanan data. Fitur-fitur utama yang dikembangkan meliputi halaman registrasi, *login* , dashboard, tambah *Habit*, *Checklist Habit*, statistik perkembangan, sistem poin, serta fitur rekomendasi atau prediksi konsistensi kebiasaan.Pembuatan struktur aplikasi menggunakan *Framework Flutter*.

   Pada tahap ini juga dilakukan implementasi prinsip *Atomic Habits* ke dalam aplikasi, yaitu:

   a. *Cue* diimplementasikan dalam bentuk notifikasi pengingat,Integrasi *Database SQLite* untuk penyimpanan data kebiasaan pengguna.
   b. *Craving* diimplementasikan melalui tampilan antarmuka yang menarik.
   c. *Response* diimplementasikan melalui tombol *Checklist* atau penanda kebiasaan.
   d. *Reward* diimplementasikan melalui sistem poin.

   Selain itu, pada tahap develop juga dilakukan integrasi algoritma *Random Forest* untuk memberikan prediksi atau penilaian terhadap tingkat konsistensi kebiasaan berdasarkan data aktivitas pengguna.

4. Pengujian (*Test*)

   Tahap Test dilakukan untuk memastikan bahwa sistem yang telah dikembangkan dapat berjalan sesuai dengan rancangan. Pengujian dalam penelitian ini menggunakan *Statement*, yaitu pengujian yang berfokus pada logika program, jalur eksekusi kode, serta struktur kontrol pada sistem. Pada tahap ini dilakukan pengujian terhadap beberapa modul utama aplikasi, seperti *login* , *Register*, tambah *Habit*, *Checklist Habit*, statistik, dan fitur *Random Forest*. Pengujian dilakukan

melalui analisis *Flowchart* , *Flowgraph* , *Cyclomatic Complexity*, *Independent Path*, *test case*, dan *Statement Coverage*. Melalui tahap ini, peneliti dapat mengetahui apakah fungsi-fungsi utama pada aplikasi telah berjalan dengan baik.

5. Implementasi (*Relase*)

Tahap Release merupakan tahap implementasi hasil pengembangan sistem. Pada tahap ini, aplikasi *Habit Tracker* yang telah selesai dikembangkan dan diuji dijalankan pada perangkat Android. Tujuan tahap ini adalah untuk memastikan bahwa aplikasi dapat digunakan sesuai dengan fungsi yang telah dirancang. Pada penelitian ini, tahap release difokuskan pada penyajian hasil akhir aplikasi sebagai produk pengembangan sistem. Tahap ini tidak diarahkan pada uji efektivitas aplikasi kepada mahasiswa, melainkan pada implementasi teknis aplikasi sebagai alat bantu pemantauan pembentukan kebiasaan positif.

6. *Feedback* (Evaluasi dan Pemeliharaan)

Tahap *Feedback* dalam penelitian ini tidak diartikan sebagai pengumpulan umpan balik dari pengguna secara langsung, karena penelitian ini tidak melakukan uji coba aplikasi kepada mahasiswa sebagai pengguna akhir. Tahap *feedback* lebih difokuskan pada evaluasi teknis dan pemeliharaan sistem, yaitu melalui hasil pengujian, proses debugging, pemeriksaan mandiri, serta arahan dari dosen pembimbing. Pada tahap ini dilakukan perbaikan terhadap kesalahan sistem, penyempurnaan fitur aplikasi, penyesuaian tampilan antarmuka, serta penyusunan dokumentasi akhir penelitian. Dengan demikian, tahap *feedback* berfungsi untuk meningkatkan kualitas aplikasi agar lebih stabil dan sesuai dengan tujuan penelitian.

Tabel 3. 2 *Sprint* Pengembangan Sistem

Sprint	**Tahapan *Agile***	**Aktivitas**	**Deskripsi Kegiatan**	**Output**
***Sprint* 1**	Perencanaan (Plan)	Identifikasi masalah dan kebutuhan awal sistem	Peneliti mengidentifikasi permasalahan mahasiswa dalam memantau pembentukan kebiasaan positif, menentukan tujuan penelitian, ruang lingkup sistem, fitur utama, serta teknologi yang digunakan dalam pengembangan aplikasi *Habit Tracker*.	Rencana pengembangan aplikasi dan daftar kebutuhan awal sistem
***Sprint* 2**	Design (Analisis Kebutuhan dan Perancangan)	Analisis kebutuhan dan perancangan sistem	Peneliti menganalisis data hasil observasi, wawancara, studi literatur, dan kuesioner dari 75 mahasiswa untuk menentukan kebutuhan fitur aplikasi. Selanjutnya dilakukan perancangan *Use Case Diagram*, *Activity Diagram*, *Class Diagram*, *Database SQLite*, dan desain antarmuka menggunakan *Figma*.	Dokumen kebutuhan sistem, rancangan UML, rancangan *Database*, dan desain antarmuka
***Sprint* 3**	Pengembangan (Develop)	Pembuatan aplikasi *Habit Tracker*	Peneliti membangun aplikasi menggunakan *Flutter* dan *Dart* dengan *SQLite* sebagai media penyimpanan data. Fitur yang dikembangkan meliputi registrasi, *login* , dashboard, tambah *Habit*, *Checklist Habit*, statistik, sistem poin, serta penerapan prinsip Atomic *Habit*s.	Aplikasi *Habit Tracker* dengan fitur utama
***Sprint* 4**	Pengembangan (Develop)	Integrasi algoritma *Random Forest*	Peneliti menyiapkan dataset, melatih model *Random Forest*, serta menerapkan hasil prediksi atau aturan konsistensi ke dalam aplikasi. Algoritma *Random Forest* digunakan	Fitur prediksi konsistensi kebiasaan

			untuk memberikan prediksi atau penilaian tingkat konsistensi kebiasaan berdasarkan data aktivitas pengguna.	
***Sprint* 5**	Pengujian (Test)	Pengujian sistem	Peneliti melakukan pengujian sistem menggunakan *Statement* pada beberapa fitur utama, seperti *login* , *Register*, tambah *Habit*, *Checklist Habit*, statistik, dan *Random Forest*. Pengujian dilakukan melalui *Flowchart* , *Flowgraph* , *Cyclomatic Complexity*, *Independent Path*, *test case*, dan *Statement Coverage*.	Hasil pengujian sistem
***Sprint* 6**	Implementasi dan Pemeliharaan (Release dan *Feedback*)	Implementasi akhir, evaluasi teknis, dan pemeliharaan	Aplikasi dijalankan pada perangkat Android sebagai hasil akhir pengembangan sistem. Tahap *feedback* dalam penelitian ini difokuskan pada evaluasi teknis, debugging, perbaikan kesalahan sistem, penyempurnaan fitur, serta arahan pembimbing. Tahap ini tidak diarahkan pada uji efektivitas aplikasi kepada mahasiswa.	Aplikasi final dan dokumentasi penelitian

Berdasarkan Tabel di atas, pengembangan aplikasi *Habit Tracker* dilakukan melalui enam *Sprint* yang disesuaikan dengan tahapan metode *Agile*, yaitu *Plan*, *Design*, *Develop*, *Test*, *Release*, dan *Feedback*. *Sprint* 1 difokuskan pada perencanaan sistem, *Sprint* 2 pada analisis kebutuhan dan perancangan, *Sprint* 3 dan *Sprint* 4 pada pengembangan aplikasi serta integrasi algoritma *Random Forest*, *Sprint* 5 pada pengujian sistem, dan *Sprint* 6 pada implementasi akhir serta pemeliharaan.

Pada penelitian ini, tahap *Feedback* tidak diartikan sebagai pengumpulan umpan balik dari pengguna setelah menggunakan aplikasi, karena penelitian ini

tidak melakukan uji efektivitas aplikasi kepada mahasiswa. Tahap *Feedback* difokuskan pada evaluasi teknis, debugging, perbaikan kesalahan, penyempurnaan fitur, dan arahan pembimbing agar aplikasi yang dikembangkan dapat berjalan sesuai dengan rancangan.

3.6 Alat Bantu Perancangan

Pada proses pengembangan aplikasi *Habit Tracker* berbasis Android ini, peneliti menggunakan berbagai alat bantu untuk mendukung tahap perancangan, desain antarmuka, serta proses *Debugging* selama implementasi. Alat bantu ini dipilih agar setiap tahapan pengembangan dapat dilakukan secara terstruktur, efisien, dan sesuai dengan kebutuhan penelitian, termasuk integrasi prinsip *Atomic Habits* dan algoritma *Random Forest*. Berikut adalah alat bantu perancangan yang digunakan dalam penelitian ini.

3.6.1 Alat Bantu Utama

Alat bantu utama merupakan perangkat atau metode yang digunakan untuk merancang kebutuhan sistem, memodelkan alur proses aplikasi, serta menggambarkan struktur data dan hubungan antar komponen sistem. Penggunaan alat bantu ini bertujuan untuk memastikan bahwa rancangan sistem telah sesuai dengan kebutuhan pengguna dan tujuan penelitian sebelum memasuki tahap implementasi.

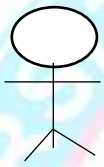
Dalam penelitian ini, alat bantu utama yang digunakan adalah *Unified Modeling Language (UML)* sebagai bahasa pemodelan sistem. *UML* digunakan untuk memvisualisasikan kebutuhan sistem melalui berbagai diagram, seperti *Use Case Diagram* untuk menggambarkan interaksi pengguna dengan sistem, *Activity Diagram* untuk menjelaskan alur proses aplikasi, serta *Class Diagram* untuk merancang struktur kelas dan relasi antar entitas dalam sistem.

Penggunaan *UML* membantu peneliti dalam mendokumentasikan rancangan sistem secara jelas dan sistematis, sehingga meminimalkan kesalahan pada tahap pengkodean serta mempermudah proses pengembangan aplikasi *Habit Tracker* berbasis Android. Berikut ini merupakan alat bantu utama perancangan sistem yang digunakan oleh penulis dalam penelitian ini:

1. *Use Case Diagram*

Use case diagram adalah abstraksi dari interaksi antara sistem dengan aktor (Ramdany et al., 2024). Diagram ini menggambarkan fungsi-fungsi apa saja yang dapat dilakukan oleh aktor terhadap sistem, serta batasan sistem dalam menyediakan layanan tertentu. Melalui *Use Case Diagram*, peneliti dapat mengidentifikasi kebutuhan fungsional aplikasi secara lebih jelas dan memastikan bahwa setiap fitur yang dirancang benar-benar sesuai dengan peran dan kebutuhan pengguna.

Tabel 3. 3 Simbol-simbol *Use Case Diagram*

Simbol	Keterangan
	Aktor: Mewakili peran orang, sistem lain, atau alat yang berinteraksi dengan sistem melalui use case.
	Use Case: Abstraksi dari fungsi atau interaksi antara sistem dan aktor.
	Association: Abstraksi dari penghubung atau relasi antara aktor dan *use case.*
	Generalization: Menunjukkan spesialisasi aktor agar dapat berpartisipasi dengan use case tertentu.
<<include>>	*Include*: Menunjukkan bahwa suatu *use case* seluruhnya merupakan bagian atau fungsi dari *use case* lainnya.
<<extend>>	*Extend*: Menunjukkan bahwa suatu *use case* merupakan tambahan fungsi dari *use case* lain jika kondisi tertentu terpenuhi.

Sumber :(Ramdany et al., 2024)

Tabel 3.3 di atas menjelaskan berbagai simbol yang digunakan dalam *Use Case Diagram* beserta keterangannya. Setiap simbol memiliki fungsi dan makna tertentu yang membantu dalam menggambarkan interaksi antara pengguna (aktor) dan sistem secara visual. Simbol-simbol tersebut digunakan untuk mempermudah pemahaman mengenai alur kebutuhan sistem, fungsi-fungsi yang disediakan, serta hubungan antar elemen dalam *Use Case*.

Simbol pertama adalah aktor, yang digambarkan dengan bentuk manusia stik. Aktor mewakili peran atau entitas luar yang berinteraksi langsung dengan sistem, baik itu pengguna manusia, perangkat lain, maupun sistem eksternal. Simbol berikutnya adalah *use case*, yang berbentuk oval. *Use case* merupakan abstraksi dari fungsi atau layanan yang disediakan sistem untuk aktor. Setiap u*se case* mendeskripsikan satu perilaku atau layanan khusus yang dapat dilakukan sistem sebagai *Respons* terhadap interaksi dari aktor.

Selanjutnya terdapat simbol *association*, yaitu garis penghubung antara aktor dan *use case*. *Association* menggambarkan adanya interaksi atau hubungan komunikasi antara keduanya. Simbol *generalization* digunakan untuk menunjukkan hubungan pewarisan antar aktor atau antar *use case*. Melalui *generalization*, terlihat bahwa suatu aktor atau *use case* memiliki atau mewarisi karakteristik dari aktor/*use case* lain yang lebih umum.

Kemudian terdapat simbol *include*, yang menunjukkan bahwa suatu *use case* memanggil atau menyertakan fungsi dari *use case* lain sebagai bagian dari prosesnya. Ini digunakan ketika ada fungsi yang selalu dibutuhkan oleh beberapa *use case* lain. Terakhir, simbol *extend* menunjukkan bahwa suatu *use case* dapat diperluas oleh *use case* lain dalam kondisi tertentu. *Extend* digunakan jika fungsi tambahan hanya dijalankan ketika syarat tertentu terpenuhi.

Dengan adanya simbol-simbol tersebut, *Use Case Diagram* dapat disusun secara lebih terstruktur dan mudah dipahami. Tabel ini juga menjadi acuan penting dalam proses analisis kebutuhan sistem sebelum masuk ke tahap perancangan lebih lanjut.

2. *Activity Diagram*

Activity Diagram adalah jenis diagram dalam *Unified Modeling Language (UML)* yang digunakan untuk menggambarkan aliran kerja atau aktivitas dalam suatu sistem atau proses. Diagram ini menyajikan serangkaian kegiatan, tindakan, dan keputusan yang terjadi sepanjang waktu (Rasiban et al., 2024).

Tabel 3. 4 Simbol *Activity Diagram*

Gambar	Nama	Simbol
	Activity	Memperlihatkan bagaimana masing-masing kelas atau entitas dalam sistem saling berinteraksi melalui suatu aktivitas.
	Action	State dari sistem yang mencerminkan eksekusi dari suatu aksi.
	Intial Node	Menunjukkan bagaimana suatu proses dimulai.
	Activity Final Node	Menunjukkan bagaimana suatu aktivitas berakhir.
	Decision	Digunakan untuk menggambarkan suatu keputusan atau percabangan tindakan berdasarkan kondisi tertentu.
	Line Connector	Digunakan untuk menghubungkan satu simbol dengan simbol lainnya dalam alur aktivitas.

Sumber: (Rasiban et al., 2024)

Tabel 3.4 di atas menjelaskan berbagai simbol yang digunakan dalam *Activity Diagram* beserta nama dan fungsinya. *Activity Diagram* merupakan diagram yang digunakan untuk menggambarkan alur aktivitas atau proses yang terjadi dalam sistem secara terstruktur. Setiap simbol memiliki fungsi dan makna tertentu yang menggambarkan tahapan, keputusan, ataupun aliran kerja dalam sebuah proses.

Simbol pertama adalah *Activity*, yang digunakan untuk memperlihatkan bagaimana suatu entitas atau bagian di dalam sistem melakukan interaksi melalui serangkaian aktivitas tertentu. *Activity* biasanya digambarkan sebagai langkah atau proses utama yang terjadi dalam alur sistem. Selanjutnya terdapat simbol *Action*, yang menunjukkan state atau keadaan dari sistem pada saat suatu aksi dieksekusi. *Action* menggambarkan tahapan yang lebih spesifik dari sebuah aktivitas dan merupakan bagian yang lebih rinci dalam proses.

Simbol berikutnya adalah *Initial Node,* berupa titik hitam solid yang menunjukkan titik awal dari suatu proses. Semua alur aktivitas dimulai dari simbol ini sebelum berpindah ke langkah-langkah berikutnya. Sementara itu, *Activity Final Node* digambarkan dengan lingkaran konsentris (lingkaran dalam dan luar), yang menunjukkan bahwa suatu aktivitas atau alur proses telah mencapai titik akhir. Simbol ini menjadi tanda bahwa rangkaian proses telah selesai dijalankan.

Simbol *Decision* memiliki bentuk belah ketupat dan digunakan untuk menggambarkan percabangan keputusan dalam alur sistem. Simbol ini menunjukkan bahwa terdapat kondisi tertentu yang menentukan jalur mana yang harus diambil dalam proses, sehingga alur dapat bercabang sesuai keputusan yang terjadi. Terakhir, simbol *Line Connector* digunakan untuk menghubungkan simbol-simbol dalam *Activity Diagram*. *Connector* ini menggambarkan arah aliran proses dari satu langkah ke langkah berikutnya, termasuk alur maju, kembali, maupun percabangan. Secara keseluruhan, tabel ini memberikan pemahaman yang jelas mengenai simbol-simbol dasar dalam *Activity Diagram*, sehingga memudahkan dalam proses perancangan model alur kerja sistem yang terstruktur dan mudah dipahami.

3. *Class Diagram*

Class Diagram menggambarkan serta deskripsi dari *class*, atribut dan objek serta hubungan satu sama lain. *Class Diagram* dapat memberikan pandangan global atas sebuah *system*. Hal tersebut tercermin dari *class* yang ada dan relasinya satu dengan yang lainnya (Suharni et al., 2023). Melalui *Class Diagram*, perancang sistem dapat memahami komponen dan tanggung jawab masing-masing *class* secara jelas, sehingga proses perancangan, pengembangan, dan pemeliharaan

sistem dapat dilakukan secara lebih terstruktur. Selain itu, *Class Diagram* membantu meminimalkan risiko kesalahan pada tahap implementasi serta berperan sebagai dokumentasi sistem yang memudahkan pengembangan lanjutan.

Tabel 3. 5 Simbol *Class Diagram*

Simbol	Nama	Deskripsi
Nama *Class* +atribut +atribut +atribut +method +method	*Class*	Himpunan objek-objek dari berbagai atribut yang memiliki operasi yang sama.
	Association	Relasi antar kelas dengan makna umum dan biasanya disertai multiplicity.
	Directed Association	Relasi antar kelas dengan makna kelas yang satu digunakan oleh kelas lain.
	Aggregation	Mengindikasikan keseluruhan bagian relationship disebut sebagai relasi.
	Composition	Relasi Composition terhadap *class* tempat dia bergantung.
	Dependency	Menunjukkan operasi pada suatu *class* yang menggunakan *class* yang lain.

Sumber: (Suharni et al., 2023)

Tabel 3.5 di atas menjelaskan berbagai simbol yang digunakan dalam *Class Diagram* beserta nama dan fungsinya. *Class Diagram* merupakan salah satu diagram dalam *Unified Modeling Language (UML)* yang digunakan untuk menggambarkan struktur statis dari suatu sistem. Diagram ini menampilkan kelas-kelas yang ada dalam sistem, atribut dan operasi yang dimiliki, serta hubungan antar kelas secara terstruktur. *Class Diagram* berperan penting dalam tahap perancangan

sistem karena memberikan gambaran menyeluruh mengenai struktur dan relasi data dalam sistem.

Simbol pertama adalah *Class*, yang digunakan untuk merepresentasikan himpunan objek-objek yang memiliki atribut dan operasi yang sama. *Class* menggambarkan entitas utama dalam sistem dan umumnya terdiri dari tiga bagian, yaitu nama *class*, atribut, dan *method*. *Class* menjadi dasar dalam proses implementasi sistem karena akan diterjemahkan ke dalam bentuk kelas pada tahap pengembangan aplikasi.

Simbol berikutnya adalah *Association*, yang digunakan untuk menggambarkan relasi umum antar kelas. *Association* menunjukkan bahwa terdapat hubungan atau keterkaitan antara satu kelas dengan kelas lainnya. Relasi ini biasanya disertai dengan *multiplicity* untuk menunjukkan jumlah objek yang terlibat dalam hubungan tersebut.

Selanjutnya terdapat simbol *Directed Association*, yang merupakan relasi *association* dengan arah tertentu. Simbol ini menunjukkan bahwa suatu kelas menggunakan atau bergantung pada kelas lain, namun tidak berlaku sebaliknya. *Directed association* digunakan untuk memperjelas arah interaksi atau penggunaan antar kelas dalam sistem.

Simbol *Aggregation* digunakan untuk menunjukkan hubungan keseluruhan dan bagian (*whole-part relationship*) antar kelas. Pada relasi ini, suatu kelas terdiri dari beberapa bagian kelas lain, namun bagian tersebut masih dapat berdiri sendiri tanpa sepenuhnya bergantung pada kelas induknya. *Aggregation* menggambarkan hubungan yang bersifat lebih longgar dibandingkan *composition*.

Simbol *Composition* juga menggambarkan hubungan keseluruhan dan bagian, tetapi dengan tingkat ketergantungan yang lebih kuat. Pada relasi composition, keberadaan suatu kelas sangat bergantung pada kelas induknya. Jika kelas induk dihapus, maka kelas bagian yang terkait juga akan ikut terhapus, sehingga menunjukkan keterikatan yang sangat erat.

Simbol terakhir adalah *Dependency*, yang digunakan untuk menunjukkan ketergantungan antar kelas. *Dependency* menggambarkan bahwa suatu kelas menggunakan operasi atau layanan dari kelas lain. Relasi ini bersifat sementara dan

menunjukkan bahwa perubahan pada kelas yang digunakan dapat memengaruhi kelas yang bergantung padanya.

Secara keseluruhan, tabel ini memberikan pemahaman yang jelas mengenai simbol-simbol dasar dalam *Class Diagram* beserta fungsi dan maknanya, sehingga memudahkan dalam proses perancangan struktur sistem yang terorganisir, konsisten, dan mudah dipahami.

3.6.2 Alat Bantu Desain

Alat bantu desain digunakan untuk merancang tampilan antarmuka (*UI*) aplikasi sebelum diimplementasikan dalam *Flutter*. Alat ini membantu peneliti memastikan tampilan aplikasi mudah digunakan dan sesuai prinsip *Atomic Habits* yang menekankan kemudahan dalam membentuk kebiasaan. Pada penelitian ini peneliti menggunakan *Figma* sebagai alat bantu desain. *Figma* adalah aplikasi desain grafis berbasis *web* yang memungkinkan pengguna untuk membuat desain antarmuka pengguna (*UI*) yang interaktif (Surianto et al., 2023).

3.6.3 Alat Bantu *Debugging* dan Pengembangan

Alat bantu *Debugging* dan pengembangan digunakan untuk mendukung proses implementasi aplikasi, mendeteksi kesalahan selama proses *coding*, serta memastikan integrasi sistem berjalan dengan baik. Alat bantu ini membantu peneliti dalam mengembangkan fitur *Habit Tracker*, mengelola basis data *SQLite*, serta mengintegrasikan model *Random Forest* yang telah dilatih sebelumnya. Adapun alat bantu yang digunakan meliputi:

1. *Visual Studio Code*

Visual Studio Code (VS Code) digunakan sebagai editor utama dalam proses pengembangan aplikasi menggunakan bahasa pemrograman *Dart* dan *Framework Flutter*. *VS Code* menyediakan berbagai ekstensi pendukung seperti *Flutter*, *Dart*, dan *Git* yang memudahkan peneliti dalam menulis kode, melakukan *debugging*, serta memantau struktur proyek aplikasi.

2. *Flutter Debug Console*

Flutter Debug Console digunakan untuk memantau *log* sistem, menampilkan pesan error, dan mendeteksi *bug* yang muncul selama aplikasi dijalankan di emulator atau perangkat fisik. Alat ini membantu peneliti menemukan

kesalahan logika, *error rendering UI*, kesalahan koneksi data, dan masalah lainnya yang terkait dengan proses eksekusi aplikasi.

3. *DB Browser for SQLite*

DB Browser for SQLite digunakan untuk mengelola basis data lokal yang digunakan pada aplikasi *Habit Tracker.* Melalui alat ini, peneliti dapat membuat tabel, memeriksa data yang tersimpan, memvalidasi struktur *Database*, serta memastikan bahwa proses penyimpanan dan pengambilan data pada aplikasi berjalan dengan benar. *DB Browser* juga membantu dalam *Debugging* ketika terjadi kesalahan pada proses *CRUD*.

4. *Python (Scikit-Learn)*

Python, khususnya *library Scikit-Learn*, digunakan dalam proses pelatihan model algoritma *Random Forest*. Data kebiasaan pengguna diekstraksi dan diproses menggunakan *Python* sebelum model dilatih. Setelah model *Random Forest* selesai dibuat, hasil model tersebut diintegrasikan ke dalam aplikasi untuk menghasilkan rekomendasi kebiasaan yang sesuai dengan pola aktivitas pengguna.

3.7 Pengujian Sistem

Pengujian sistem dilaksanakan untuk memastikan bahwa aplikasi *Habit Tracker* yang didasarkan pada *Atomic Habits* dan algoritma *Random Forest* berjalan sesuai dengan rencana serta dapat memproses data kebiasaan pengguna dengan tepat. Teknik pengujian yang diterapkan dalam penelitian ini adalah *Statement* , yang merupakan metode pengujian perangkat lunak yang berfokus pada analisis struktur internal kode (A. Fauzi & Yunial, 2025). Pilihan metode *Statement* didasarkan pada kenyataan bahwa aplikasi yang dikembangkan memiliki elemen logika internal yang cukup rumit, terutama terkait dengan proses pencatatan kebiasaan, pembaruan status kebiasaan, pengelolaan informasi aktivitas, dan pemanggilan model *Random Forest* untuk memprediksi tingkat konsistensi pengguna. Dengan pendekatan ini, peneliti dapat mengikuti dan memverifikasi apakah setiap bagian kode mulai dari fungsi penyimpanan data ke dalam *Database*, fungsi perhitungan kemajuan kebiasaan, hingga pelaksanaan model prediksi berjalan sesuai dengan desain tanpa menyebabkan kesalahan logika.

Proses pengujian ini dilakukan dengan menjalankan setiap modul dari aplikasi dan melacak jalur eksekusinya, mengecek kondisi percabangan seperti validasi masukan, status kebiasaan, hingga pemanggilan model prediksi. Setiap hasil diuji dengan berbagai kondisi masukan untuk memastikan sistem memberikan *Respons* yang sesuai, seperti penanganan data yang kosong, kesalahan format, atau keadaan ketika model *Machine Learning* tidak dapat diakses.

Hasil dari proses pengujian ini digunakan untuk mengidentifikasi dan memperbaiki kesalahan yang terdapat pada bagian internal aplikasi sebelum digunakan oleh pengguna. Dengan demikian, *Statement* menjamin bahwa aplikasi *Habit Tracker* yang dikembangkan memiliki stabilitas, keandalan, dan akurasi yang tinggi dalam mendukung pembentukan kebiasaan positif bagi mahasiswa Universitas Prabumulih.

BAB IV
ANALISA DAN PERANCANGAN

4.1 Analisa Sistem

Analisa sistem bertujuan untuk memahami kondisi sistem yang digunakan mahasiswa dalam membangun kebiasaan, mengidentifikasi kekurangan sistem yang berjalan, dan menentukan kebutuhan untuk sistem baru yang lebih efektif. Analisa ini menjadi dasar bagi penulis dalam pengembangan aplikasi *Habit Tracker* berbasis Android yang menggabungkan pendekatan perilaku dan teknologi kecerdasan buatan.

4.1.1 Analisa dan Evaluasi Sistem Berjalan

Analisa dan evaluasi sistem berjalan dilakukan untuk mengetahui kondisi nyata yang saat ini dialami mahasiswa dalam membangun dan mempertahankan kebiasaan positif, khususnya kebiasaan belajar. Tahap ini bertujuan untuk mengidentifikasi bagaimana sistem yang digunakan saat ini bekerja, serta mengevaluasi kelemahan dan keterbatasannya sebagai dasar perancangan sistem yang diusulkan.

a. Analisa Sistem Berjalan

Berdasarkan hasil observasi dan kuisioner kualitatif yang telah dilakukan pada mahasiswa Program Studi Informatika Universitas Prabumulih angkatan 2022 dan 2023 yang berjumlah 75 mahasiswa, diketahui bahwa pembentukan kebiasaan belajar masih dilakukan secara manual dan tidak terstruktur. Mahasiswa umumnya mengandalkan ingatan pribadi, catatan di buku tulis, atau fitur pengingat sederhana pada *smartphone* seperti *alarm* dan catatan harian. Sistem yang digunakan tersebut hanya berfungsi sebagai pengingat waktu tanpa adanya mekanisme pencatatan progres kebiasaan secara terintegrasi. Mahasiswa tidak memiliki media yang dapat merekam aktivitas kebiasaan harian secara konsisten, sehingga sulit untuk mengetahui pola perilaku dan tingkat kedisiplinan dalam menjalankan kebiasaan yang telah direncanakan. Selain itu, sistem berjalan belum menerapkan pendekatan psikologi perilaku dalam pembentukan kebiasaan. Empat prinsip dari buku *Atomic Habits* yaitu *Cue, Craving, Response, dan reward* belum diimplementasikan secara sistematis. Akibatnya, kebiasaan yang dibangun cenderung bersifat sementara dan

mudah ditinggalkan karena kurangnya dorongan dan penguatan perilaku secara berkelanjutan.

b. Evaluasi Sistem Berjalan

Berdasarkan hasil analisa sistem berjalan, dilakukan evaluasi untuk menilai efektivitas sistem yang digunakan mahasiswa saat ini. Evaluasi menunjukkan bahwa sistem berjalan memiliki beberapa kelemahan utama, yaitu tidak adanya pencatatan kebiasaan yang terpusat, tidak tersedia indikator konsistensi kebiasaan, serta tidak adanya umpan balik yang dapat memotivasi pengguna. Dari sisi teknologi, sistem yang digunakan masih bersifat pasif dan tidak adaptif. Tidak terdapat pemanfaatan algoritma analisis data atau *Machine Learning* untuk mengolah riwayat aktivitas pengguna. Hal ini menyebabkan mahasiswa tidak memperoleh informasi mengenai tingkat konsistensi kebiasaan, baik dalam bentuk evaluasi maupun prediksi.

Selain itu, sistem berjalan belum mampu memberikan rekomendasi atau dukungan berbasis data untuk membantu mahasiswa memperbaiki pola kebiasaan. Kondisi ini sejalan dengan permasalahan yang telah diuraikan pada Bab I, yaitu rendahnya konsistensi kebiasaan belajar mahasiswa meskipun telah memiliki akses terhadap teknologi digital. Berdasarkan evaluasi tersebut, dapat disimpulkan bahwa sistem berjalan belum efektif dalam mendukung pembentukan kebiasaan positif secara konsisten. Oleh karena itu, diperlukan sistem baru berupa aplikasi *Habit Tracker* berbasis Android yang mengimplementasikan prinsip *Atomic Habits* dan didukung algoritma *Random Forest*, sehingga mampu memberikan pemantauan, evaluasi, dan prediksi konsistensi kebiasaan secara lebih terstruktur dan adaptif.

4.1.2 Analisa Kebutuhan

Analisa kebutuhan dilakukan untuk menentukan spesifikasi sistem yang akan dikembangkan agar sesuai dengan permasalahan dan tujuan penelitian. Kebutuhan sistem dibagi menjadi kebutuhan pengguna, kebutuhan data, kebutuhan fungsional, kebutuhan non-fungsional, serta kebutuhan perangkat lunak dan perangkat keras.

a. Kebutuhan Pengguna

Pengguna utama aplikasi *Habit Tracker* adalah mahasiswa Program Studi Informatika Universitas Prabumulih angkatan 2022 dan 2023 berjumlah 75 mahasiswa. Pengguna membutuhkan aplikasi yang mudah digunakan untuk mencatat kebiasaan harian, memantau konsistensi, serta memperoleh motivasi dalam membangun kebiasaan positif. Selain itu, pengguna juga membutuhkan notifikasi sebagai pengingat, tampilan antarmuka yang menarik, serta sistem penghargaan yang mendorong keberlanjutan kebiasaan.

b. Kebutuhan Data

Data yang dibutuhkan dalam sistem meliputi data pengguna, data kebiasaan, data aktivitas harian, serta data hasil prediksi konsistensi. Data kebiasaan mencakup nama kebiasaan, target pelaksanaan, dan status penyelesaian. Data aktivitas digunakan sebagai input algoritma *Random Forest* untuk memprediksi tingkat konsistensi pengguna.

c. Kebutuhan Fungsional

Kebutuhan fungsional sistem meliputi kemampuan pengguna untuk menambah, mengubah, dan menghapus kebiasaan, menandai kebiasaan yang telah dilakukan, melihat riwayat aktivitas, serta memperoleh hasil prediksi konsistensi kebiasaan. Sistem juga harus mampu mengirimkan notifikasi pengingat dan menampilkan poin sebagai bentuk *reward*.

d. Kebutuhan Non Fungsional

Kebutuhan non-fungsional mencakup aspek kemudahan penggunaan, keamanan data, kinerja sistem, dan kompatibilitas perangkat. Aplikasi harus memiliki antarmuka yang sederhana dan *Respons*if, waktu respon yang cepat, serta mampu berjalan dengan baik pada perangkat Android .

e. Analisa *Software* dan *Hardware*

Analisis perangkat lunak dan perangkat keras dilakukan untuk mengetahui kebutuhan minimum dalam pengembangan dan implementasi aplikasi *Habit Tracker* berbasis Android . Kebutuhan ini meliputi perangkat lunak (*Software*) dan

perangkat keras (*hardware*) yang digunakan oleh pengembang maupun pengguna agar sistem dapat berjalan dengan optimal.

1. Kebutuhan Perangkat Lunak (*Software*)

Perangkat lunak komputer (*Software*) merupakan sekumpulan data elektronik yang disimpan dan diatur oleh komputer. Data elektronik yang disimpan oleh komputer itu dapat berupa program atau instruksi yang akan menjalankan suatu perintah. Perangkat lunak disebut juga sebagai penerjemah perintah-perintah yang dijalankan pengguna komputer untuk diteruskan atau diproses oleh perangkat keras(Jamas et al., 2023). Tabel berikut menunjukkan perangkat lunak yang digunakan dalam pengembangan sistem:

Tabel 4. 1 Kebutuhan Perangkat Lunak (*Software*)

No	Perangkat Lunak	Keterangan
1	*Flutter*	*Framework* untuk membangun aplikasi berbasis Android .
2	*Dart*	Bahasa pemrograman yang digunakan dalam *Flutter.*
3	*SQLite*	Basis data lokal untuk menyimpan data pengguna dan kebiasaan.
4	*Visual Studio Code*	Editor kode untuk pengembangan aplikasi.
5	Sistem operasi *windows*	Digunakan sebagai sistem operasi pada perangkat pengembangan.
6	*Bowser Chrome/Edge*	Digunakan untuk mencari referensi dan dokumentasi.

Berdasarkan tabel di atas, perangkat lunak yang digunakan dalam penelitian ini terdiri dari perangkat utama dan perangkat pendukung. *Flutter* digunakan sebagai *framework* utama dalam pengembangan aplikasi karena mendukung pembuatan aplikasi berbasis Android dengan efisien. Bahasa pemrograman *Dart* digunakan untuk mengimplementasikan logika sistem, sedangkan *SQLite* berfungsi sebagai basis data lokal untuk menyimpan data pengguna, data kebiasaan, serta data progres kebiasaan. *Visual Studio Code* digunakan sebagai editor kode yang ringan dan fleksibel, sementara sistem operasi *Windows* dan *browser* digunakan sebagai pendukung dalam proses pengembangan serta pencarian referensi.

2. Kebutuhan Perangkat Keras (*Hardware*)

Hardware atau perangkat keras komputer adalah sebuah komponen fisik pada komputer yang digunakan oleh sistem untuk menjalankan perintah yang telah diprogramkan atau dalam arti singkatnya sebuah komponen pada komputer yang bisa disentuh, dilihat dan diraba (H. Setiawan, 2022). Tabel berikut menunjukkan spesifikasi perangkat keras yang digunakan:

a. Perangkat Pengembangan (*Developer*)

Adapun minimum perangkat yang digunakan dalam pengembangan aplikasi *Habit Tracker*, meliputi:

Tabel 4. 2 *Perangkat Pengembangan (Developer)*

No	Komponen	Spesifikasi Minimum
1	*Prosesor*	*Intel Core i3* atau setara
2	*RAM*	4 GB
3	Penyimpanan	20 GB ruang kosong
4	Layar	Resolusi 1366 x 768

Berdasarkan tabel di atas, perangkat keras untuk pengembangan aplikasi membutuhkan spesifikasi yang cukup untuk mendukung proses *coding*, *debugging*, serta pengujian aplikasi. Prosesor minimal *Intel Core i3* dan RAM minimal 4 GB diperlukan agar aplikasi pengembangan seperti editor kode dan emulator dapat berjalan dengan baik. Penyimpanan yang memadai juga dibutuhkan untuk menyimpan file proyek dan *Database*, sedangkan resolusi layar yang cukup akan membantu pengembang dalam mendesain tampilan antarmuka aplikasi secara optimal.

b. Perangkat Pengguna (*User*)

Adapun minimum perangkat pengguna (*user*) yang dibutuhkan, meliputi:

Tabel 4. 3 Perangkat Pengguna (*User*)

No	Komponen	Spesifikasi Minimum
1	Perangkat	*Smartphone* Android
2	Sistem Operasi	Android 7.0 (Nougat)
3	RAM	2 GB
4	Penyimpanan	100 MB ruang kosong

Berdasarkan tabel di atas, perangkat pengguna yang dibutuhkan untuk menjalankan aplikasi adalah *smartphone* berbasis Android dengan spesifikasi minimum yang relatif ringan. Hal ini bertujuan agar aplikasi dapat digunakan oleh berbagai kalangan pengguna tanpa memerlukan perangkat dengan spesifikasi tinggi. Dengan RAM minimal 2 GB dan sistem operasi Android versi 7.0 ke atas, aplikasi diharapkan dapat berjalan dengan lancar, termasuk dalam melakukan pencatatan kebiasaan, pengelolaan data, serta menampilkan hasil prediksi konsistensi kebiasaan menggunakan algoritma *Random Forest.*

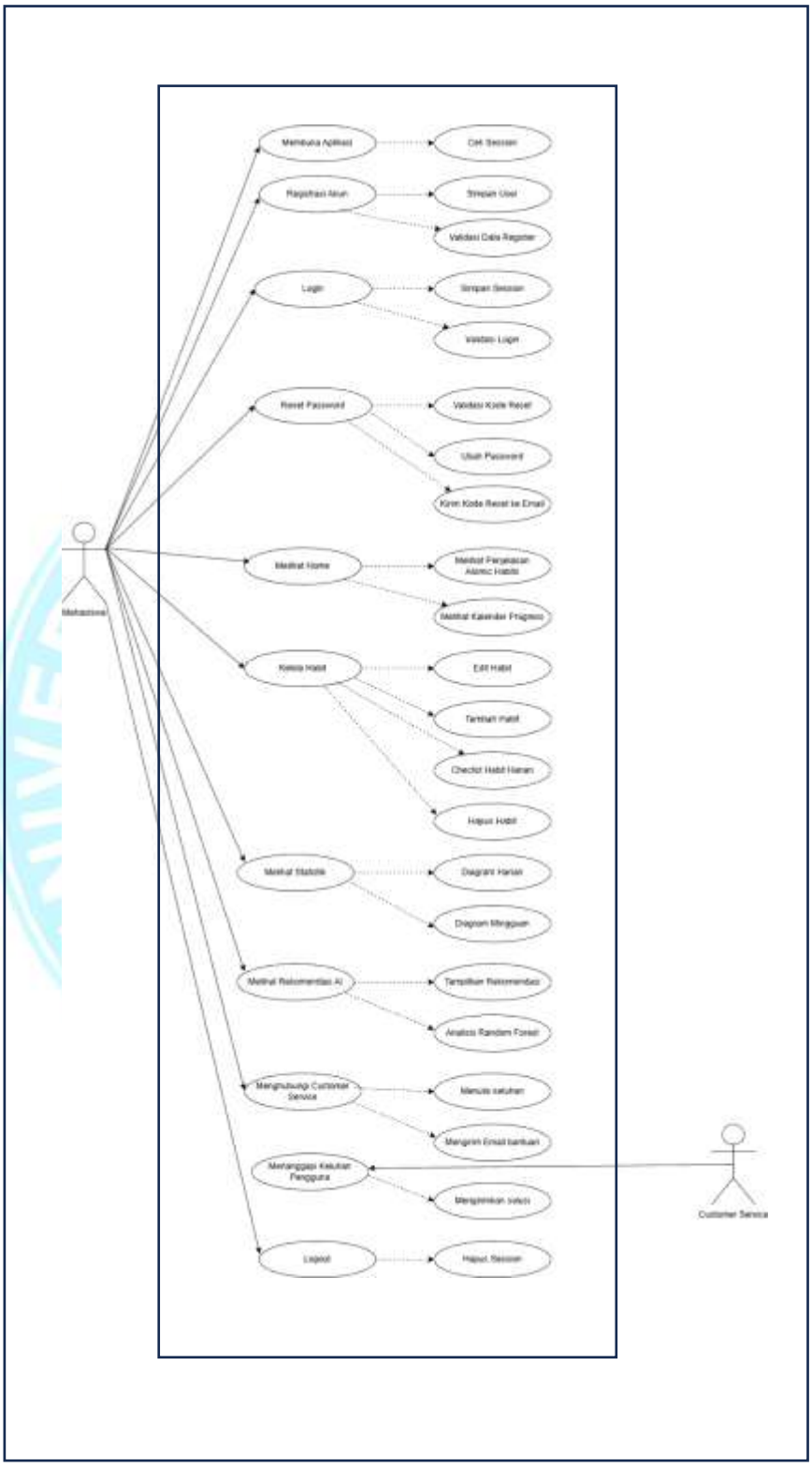
4.2 Perancangan Sistem (Desain) yang diusulkan

Perancangan sistem dilakukan berdasarkan hasil analisa kebutuhan untuk menghasilkan sistem yang terstruktur dan sesuai tujuan penelitian. Perancangan ini mencakup perancangan logis menggunakan *UML*.

4.2.1 Perancangan Sistem (Logis)

a. *Use Case Diagram*

Use Case Diagram digunakan untuk menggambarkan interaksi antara pengguna dan sistem. Diagram ini menunjukkan fungsi-fungsi utama yang dapat dilakukan pengguna, seperti mengelola kebiasaan, menandai aktivitas, melihat hasil prediksi, dan menerima notifikasi. *Use Case Diagram* membantu memastikan bahwa seluruh kebutuhan fungsional sistem telah terakomodasi.



Gambar 4. 1 *Use Case Diagram*

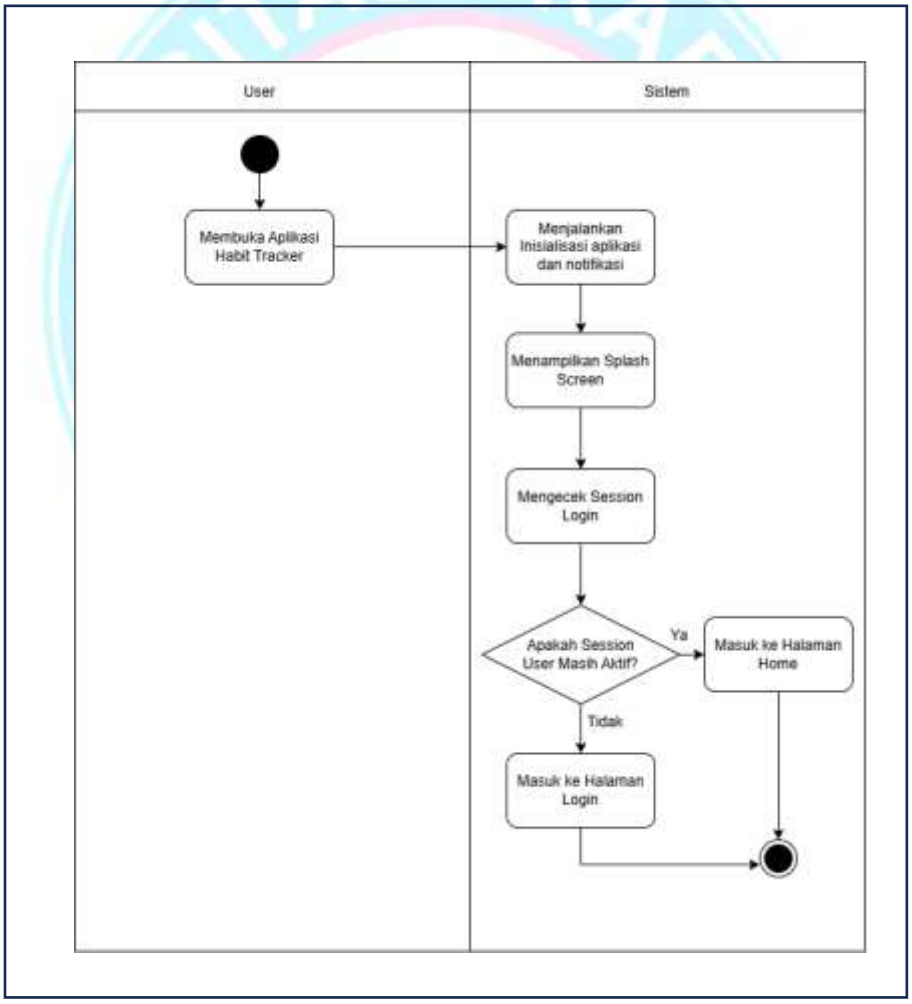
Gambar di atas menjelaskan hubungan antara pengguna dengan fitur-fitur utama pada aplikasi *Habit Tracker*. Pengguna dapat membuka aplikasi, melakukan

registrasi, *login* , *Reset Password*, mengelola *Habit*, melihat statistik, melihat rekomendasi *AI*, serta melakukan *Logout*. Diagram ini menunjukkan batasan fungsi sistem dan layanan yang dapat digunakan oleh pengguna.

b. *Activity Diagram*

Activity Diagram menggambarkan alur aktivitas pengguna dalam menggunakan aplikasi *Habit Tracker*, mulai dari membuka aplikasi, menambahkan kebiasaan, menandai kebiasaan selesai, hingga melihat hasil evaluasi konsistensi. Diagram ini memberikan gambaran proses sistem secara runtut dan logis.

1. Activity Splash dan Cek Session



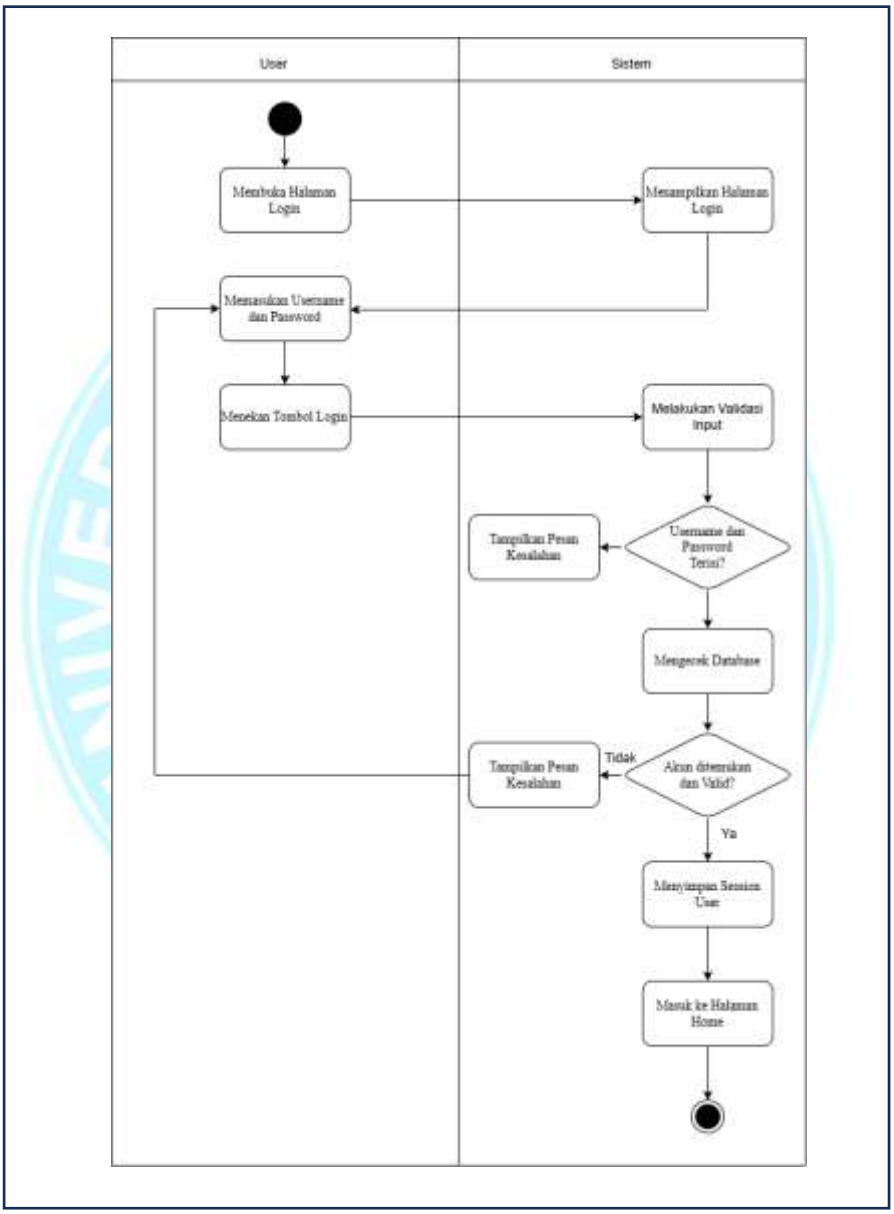
Sumber: data diolah oleh penulis (2026)

Gambar 4. 2 *Activity Splash* dan Cek *Session*

Gambar di atas menjelaskan alur awal ketika pengguna membuka aplikasi. Sistem akan menjalankan inisialisasi, menampilkan splash screen, lalu mengecek

session *login* . Jika session masih aktif, pengguna diarahkan ke halaman *Home*, sedangkan jika tidak aktif maka pengguna diarahkan ke halaman *Login* .

2. Activity *Login* Pengguna



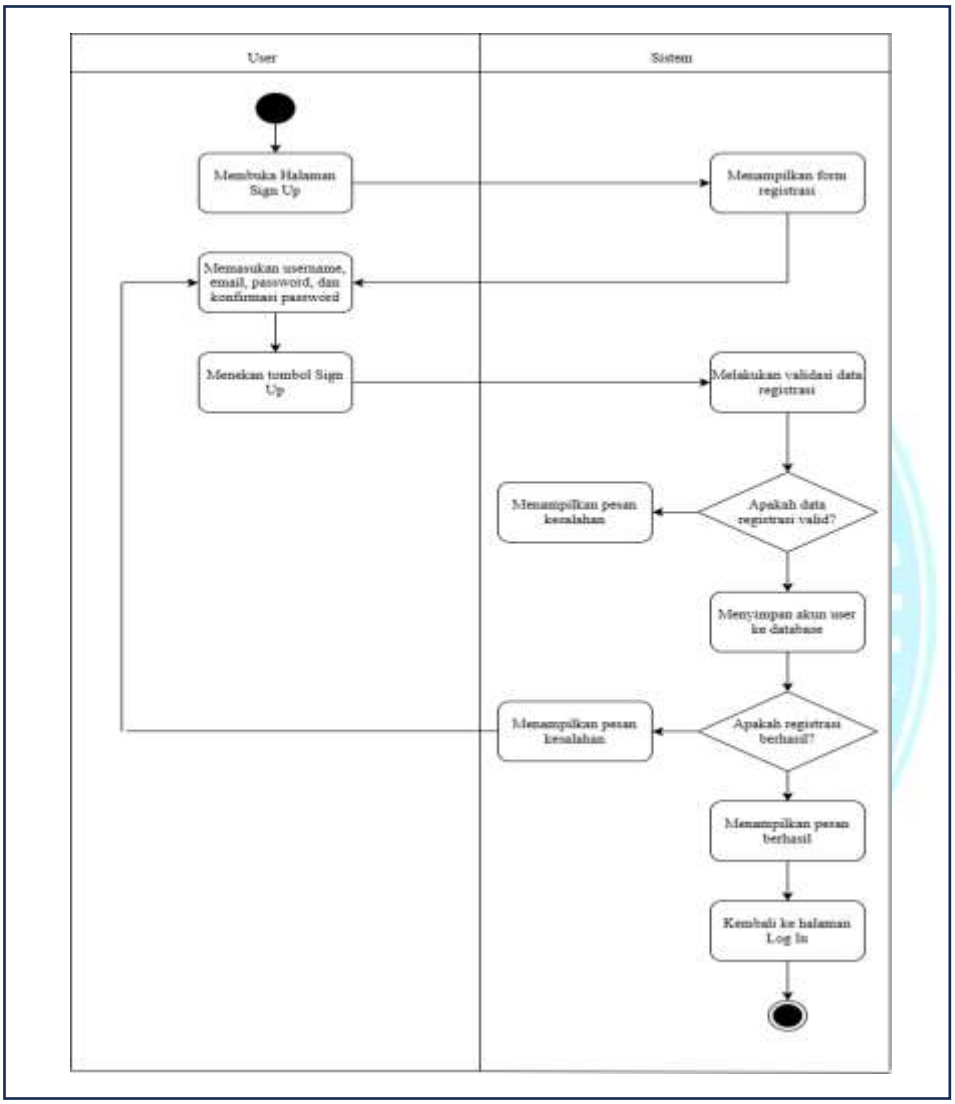
Sumber: data diolah oleh penulis (2026)

Gambar 4. 3 *Activity Login* Pengguna

Gambar ini menjelaskan proses *login* pengguna ke dalam aplikasi. Pengguna memasukkan *Username* dan *Password*, kemudian sistem melakukan validasi input serta mencocokkan data dengan *Database*. Jika data benar, sistem

menyimpan session dan mengarahkan pengguna ke halaman *Home*. Jika data salah, sistem menampilkan pesan kesalahan.

3. Activity *Register*

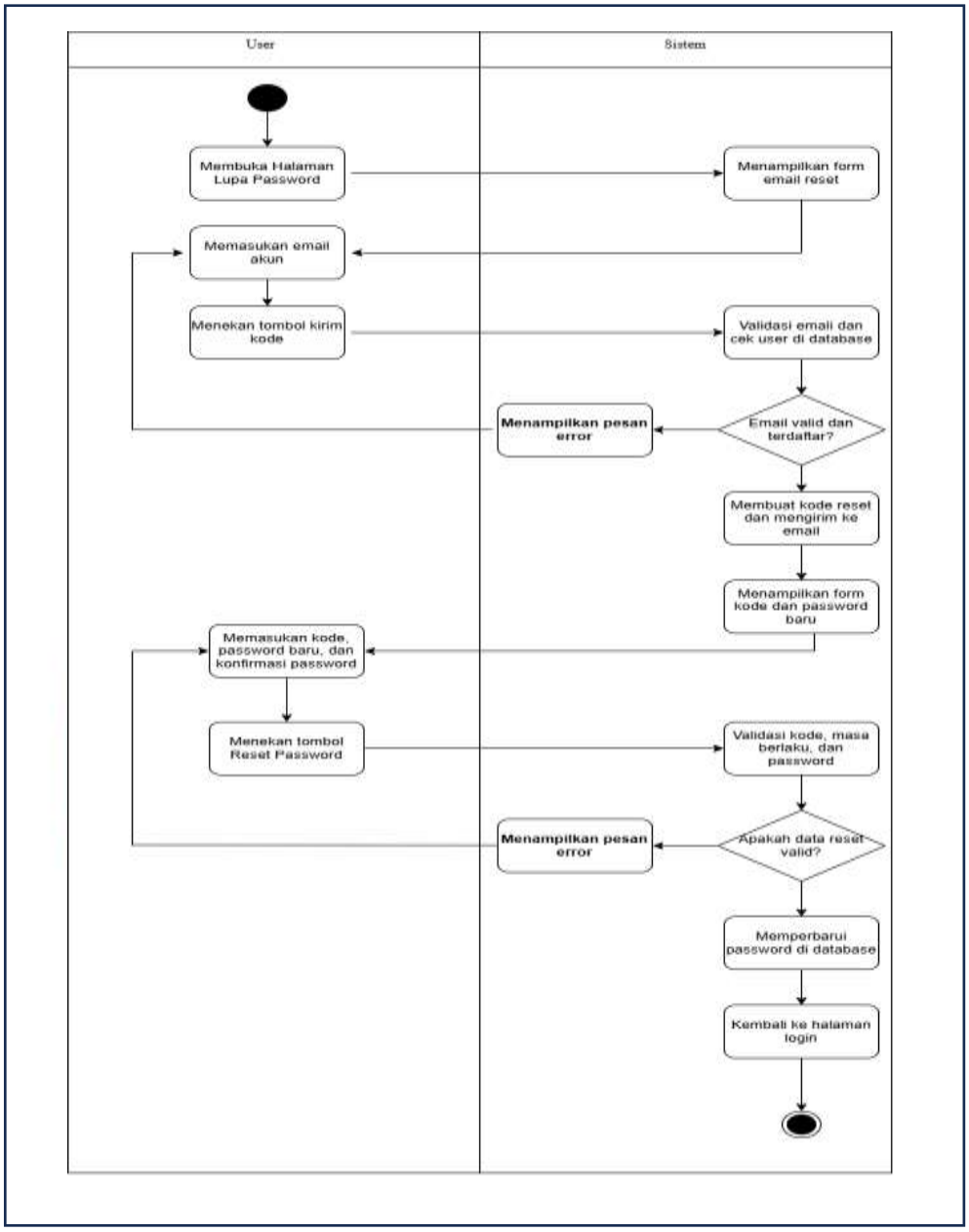


Sumber: data diolah oleh penulis (2026)

Gambar 4. 4 *Activity Register* Pengguna

Gambar di atas menjelaskan proses pendaftaran akun baru. Pengguna mengisi data registrasi berupa *Username*, email, *Password*, dan konfirmasi *Password*. Sistem kemudian memvalidasi data tersebut. Jika valid, akun disimpan ke *database* dan pengguna diarahkan kembali ke halaman *Login* .

4. *Activity Diagram Reset Password*

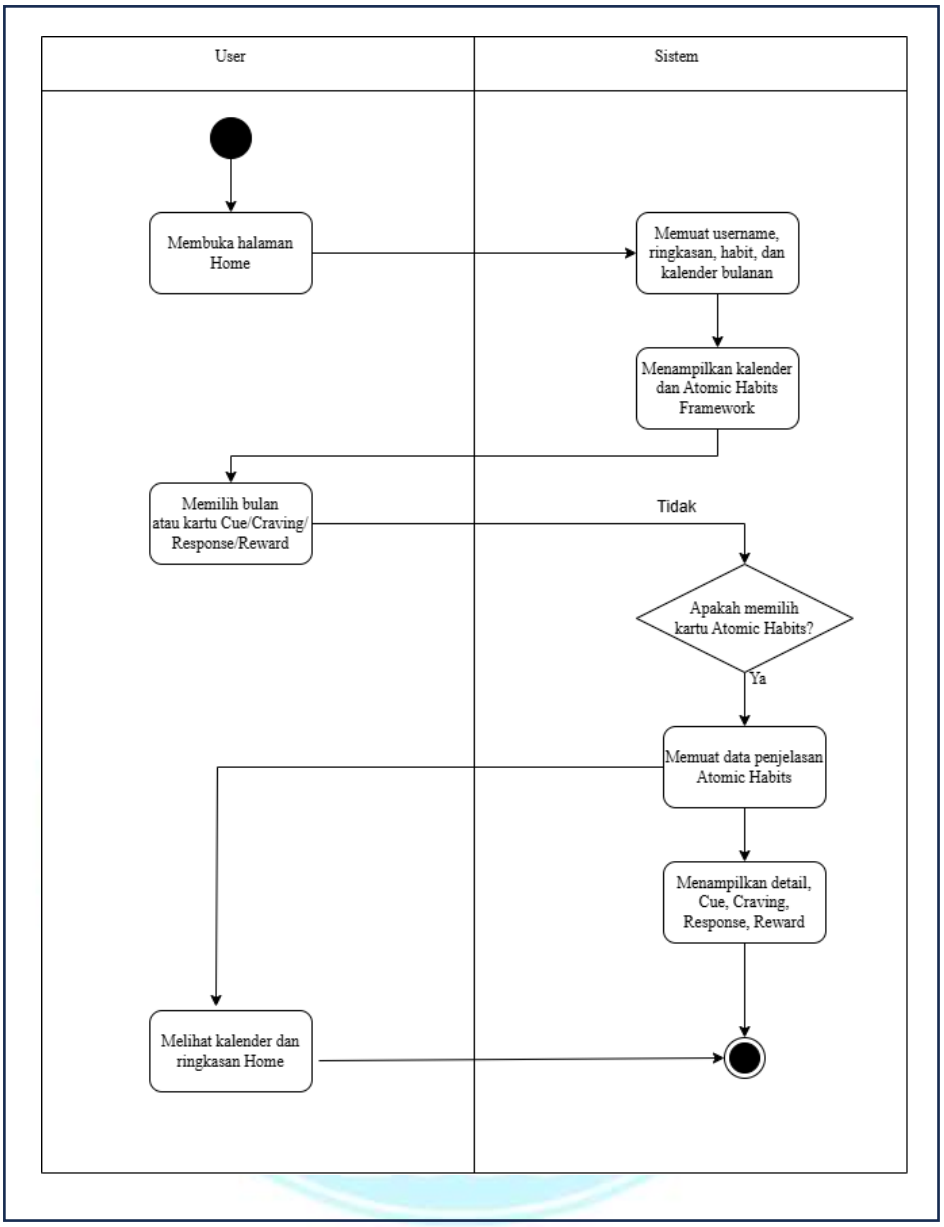


Sumber: data diolah oleh penulis (2026)

Gambar 4. 5 Activity *Reset Password*

Gambar di atas menjelaskan proses pengaturan ulang *Password*. Pengguna memasukkan email akun, lalu sistem memvalidasi email dan mengirimkan kode reset. Setelah kode dan *Password* baru dimasukkan, sistem melakukan validasi ulang. Jika data valid, *Password* diperbarui dan pengguna kembali ke halaman *Login*.

5. *Activity Diagram Home*

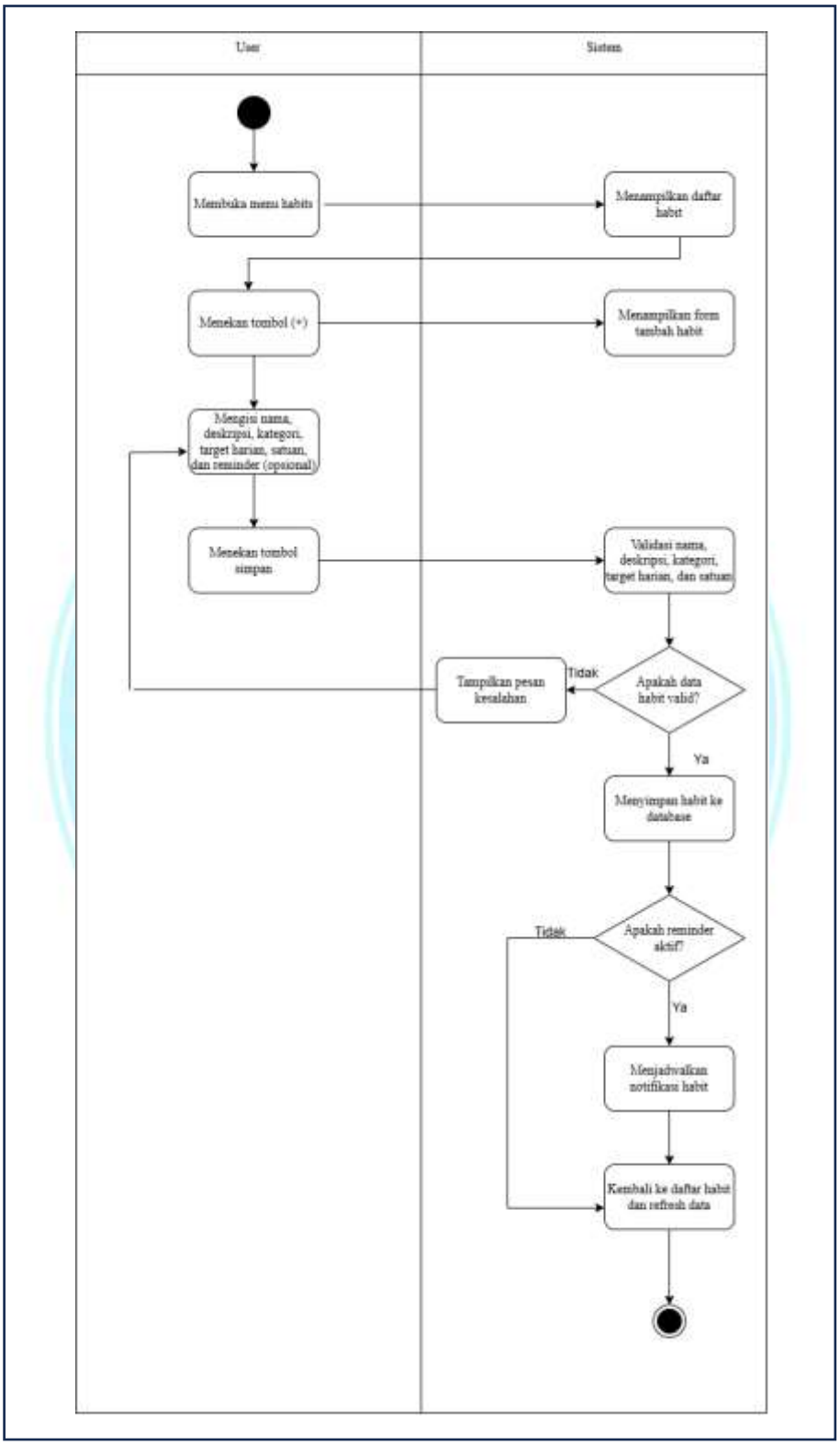


Sumber: data diolah oleh penulis (2026)

Gambar 4. 6 *Activity Home*

Gambar di atas menjelaskan alur pada halaman utama aplikasi. Sistem menampilkan ringkasan pengguna, daftar *Habit*, kalender bulanan, dan kartu penjelasan prinsip Atomic *Habit*s. Jika pengguna memilih kartu *Cue*, *Craving*, *Response*, atau *Reward*, sistem akan menampilkan detail penjelasan dari prinsip tersebut.

6. *Activity Diagram Add Habit*

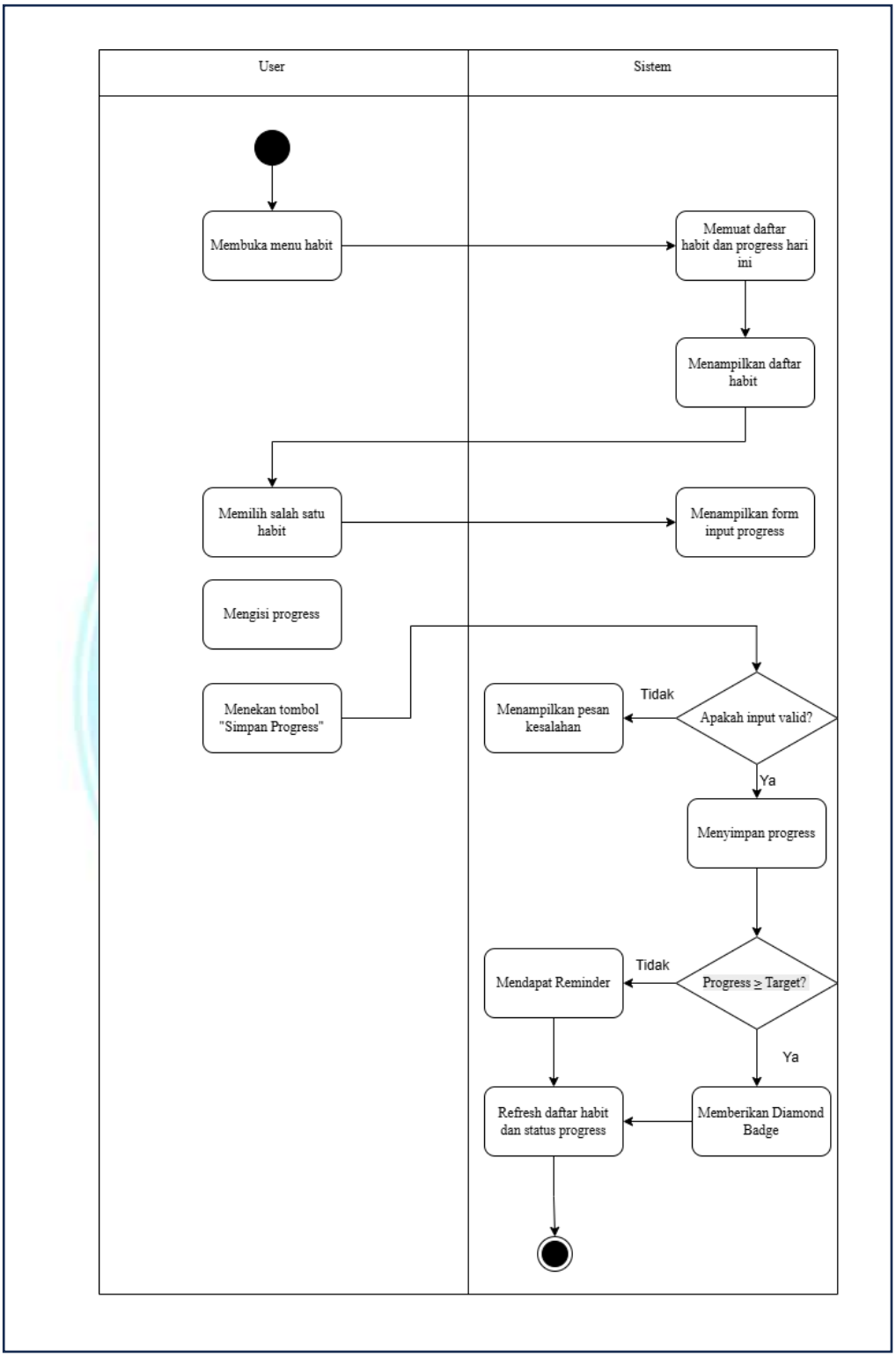


Sumber: data diolah oleh penulis (2026)

Gambar 4. 7 Activity Add Habit

Gambar di atas menjelaskan proses penambahan habit baru pada aplikasi Habit Tracker. Proses dimulai ketika pengguna membuka menu Habit sehingga sistem menampilkan daftar habit yang telah dibuat. Selanjutnya pengguna menekan tombol tambah (+) untuk membuka formulir penambahan habit. Pengguna kemudian mengisi data habit yang meliputi nama habit, deskripsi, kategori, target harian, satuan, serta pengaturan reminder apabila diperlukan. Setelah tombol simpan ditekan, sistem melakukan validasi terhadap seluruh data yang telah diinput. Apabila data belum valid, sistem akan menampilkan pesan kesalahan sehingga pengguna dapat memperbaiki data yang diisikan. Sebaliknya, apabila data telah valid, sistem akan menyimpan data habit ke dalam database. Selanjutnya sistem memeriksa apakah fitur reminder diaktifkan. Jika reminder aktif, sistem akan menjadwalkan notifikasi sesuai waktu yang telah ditentukan. Setelah seluruh proses selesai, sistem kembali menampilkan daftar habit yang telah diperbarui beserta data terbaru.

7. *Activity Input Progress Habit*

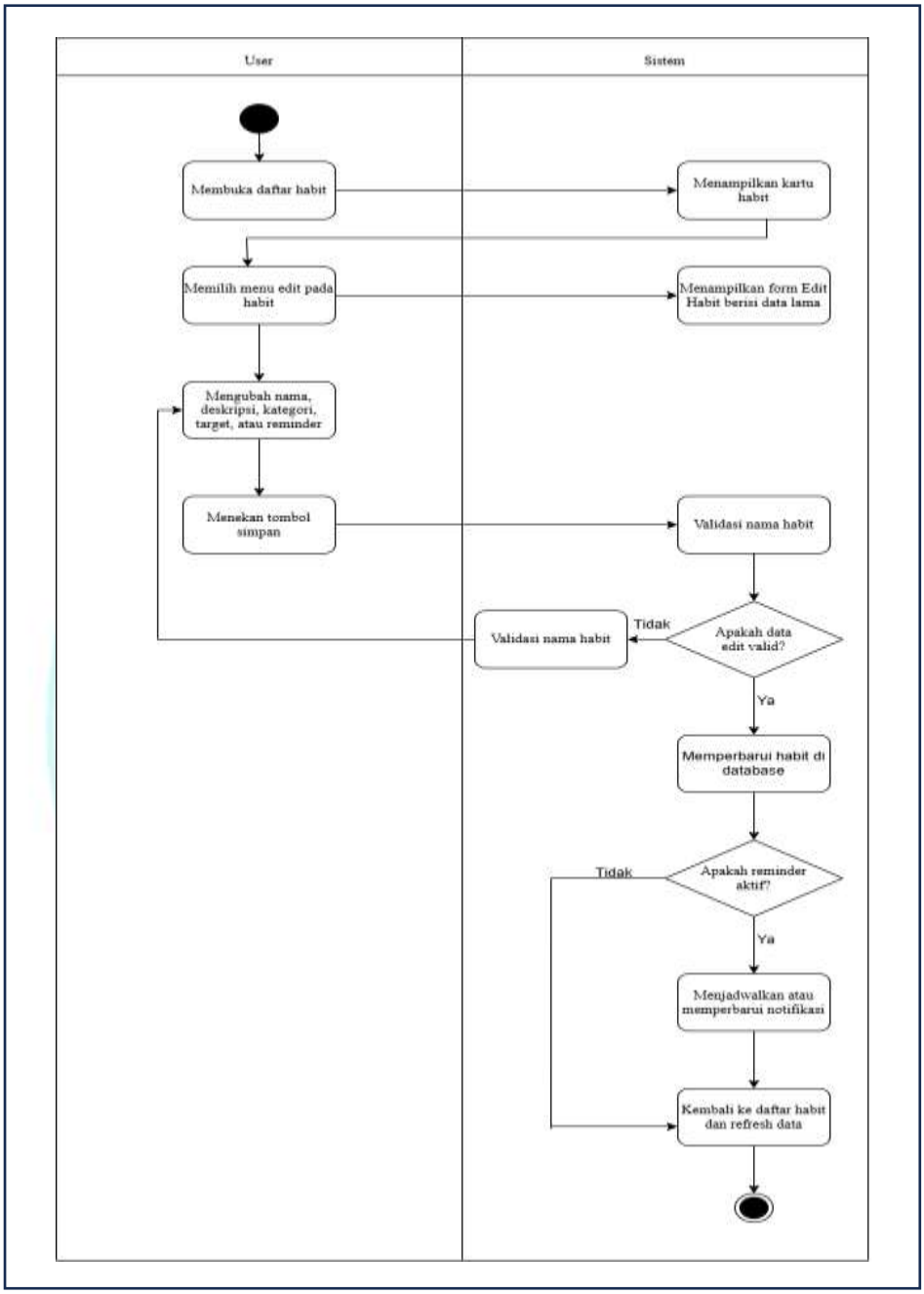


Sumber: data diolah oleh penulis (2026)

Gambar 4. 8 *Activity Input Progress Habit*

Gambar di atas menjelaskan proses pengguna dalam menginput progress habit pada aplikasi Habit Tracker. Proses dimulai ketika pengguna membuka menu Habit sehingga sistem memuat daftar habit beserta progress yang telah dilakukan pada hari tersebut. Selanjutnya pengguna memilih salah satu habit yang ingin diperbarui, kemudian sistem menampilkan formulir input progress. Pengguna memasukkan nilai progress sesuai aktivitas yang telah dilakukan dan menekan tombol Simpan Progress. Sistem kemudian melakukan validasi terhadap data yang diinput. Apabila data tidak valid, sistem akan menampilkan pesan kesalahan sehingga pengguna dapat memperbaiki data yang dimasukkan. Sebaliknya, apabila data valid, sistem akan menyimpan progress ke dalam database dan membandingkan nilai progress dengan target yang telah ditentukan. Jika progress telah mencapai atau melebihi target, sistem memberikan Diamond Badge sebagai bentuk penghargaan (Reward). Namun, apabila progress belum mencapai target, data progress tetap disimpan tanpa memberikan Diamond Badge. Setelah proses selesai, sistem memperbarui daftar habit beserta status progress pengguna sehingga informasi yang ditampilkan selalu sesuai dengan data terbaru.

8. Activity Edit *Habit*

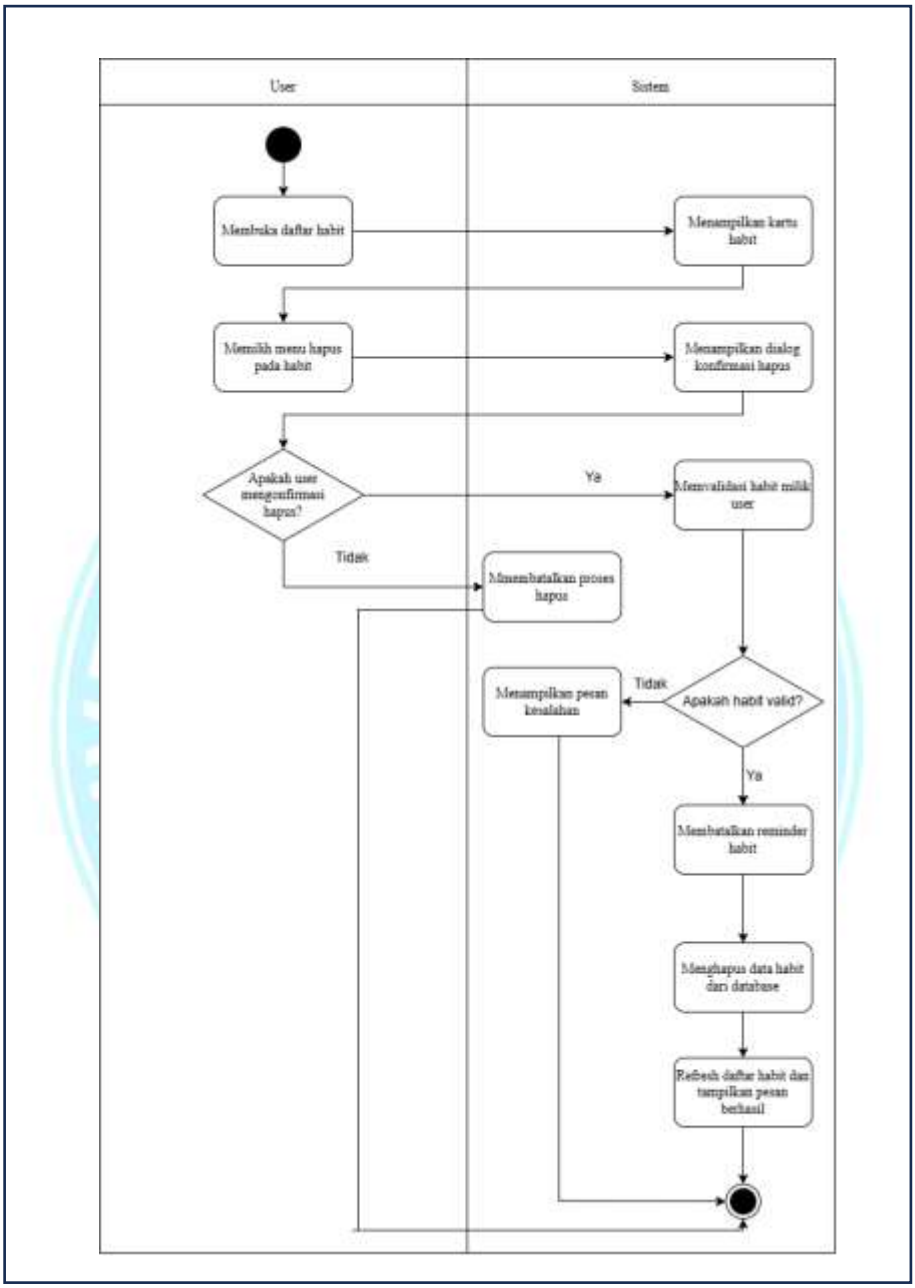


Sumber: data diolah oleh penulis (2026)

Gambar 4. 9 *Activity Edit Habit*

Gambar di atas menjelaskan proses mengubah data *Habit*. Pengguna memilih menu edit pada *Habit* yang ingin diubah, kemudian sistem menampilkan form berisi data lama. Setelah pengguna mengubah data dan menekan simpan, sistem melakukan validasi, memperbarui data di *Database*, serta memperbarui *Reminder* jika diaktifkan.

9. Activity Hapus *Habit*

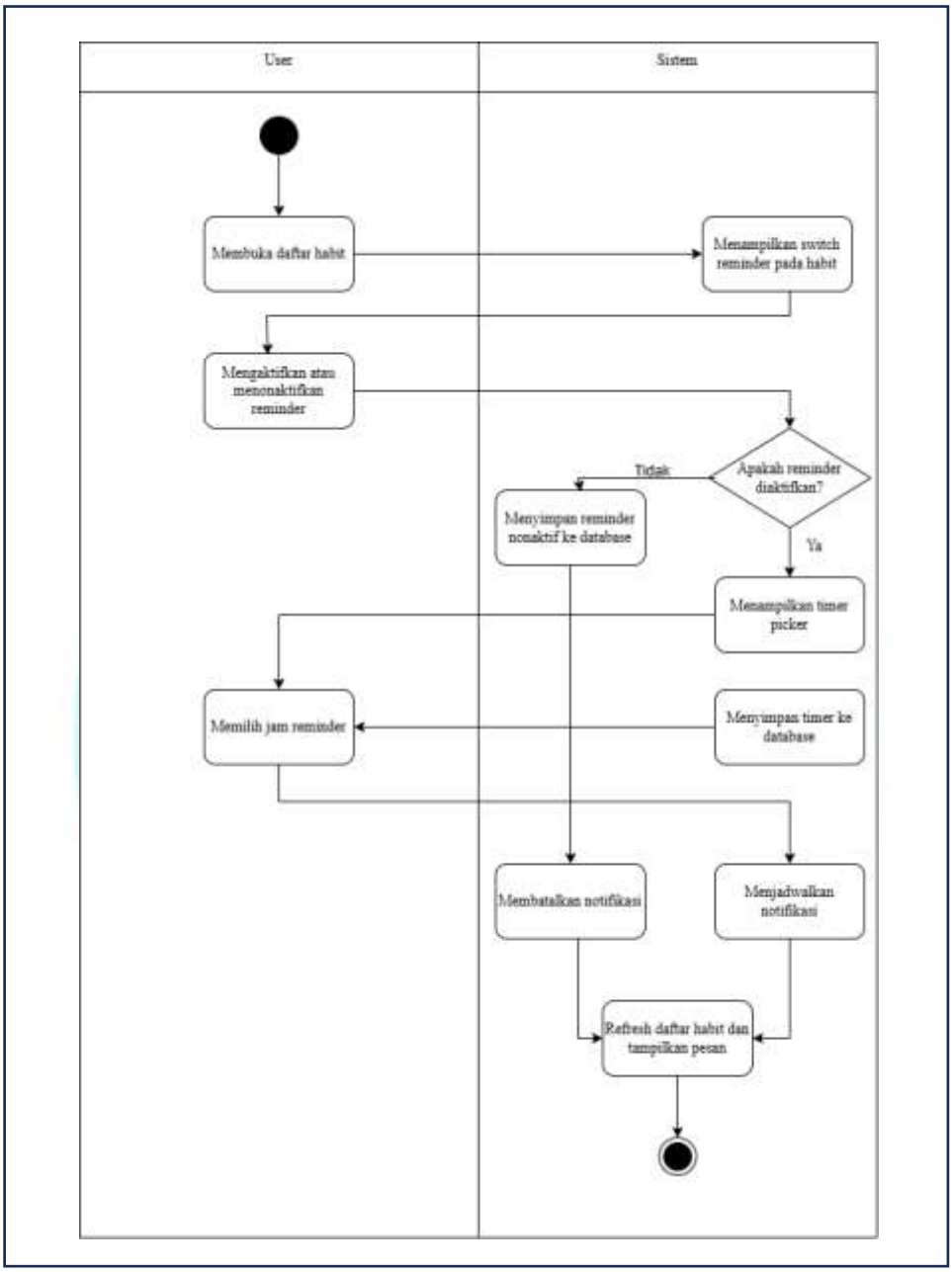


Sumber: data diolah oleh penulis (2026)

Gambar 4. 10 *Activity* Hapus *Habit*

Gambar di atas menjelaskan proses penghapusan *Habit*. Pengguna memilih menu hapus, kemudian sistem menampilkan dialog konfirmasi. Jika pengguna menyetujui, sistem memvalidasi data, membatalkan *Reminder Habit*, menghapus data dari *Database*, lalu memperbarui daftar *Habit*.

10. Activity Kelola *Reminder Habit*

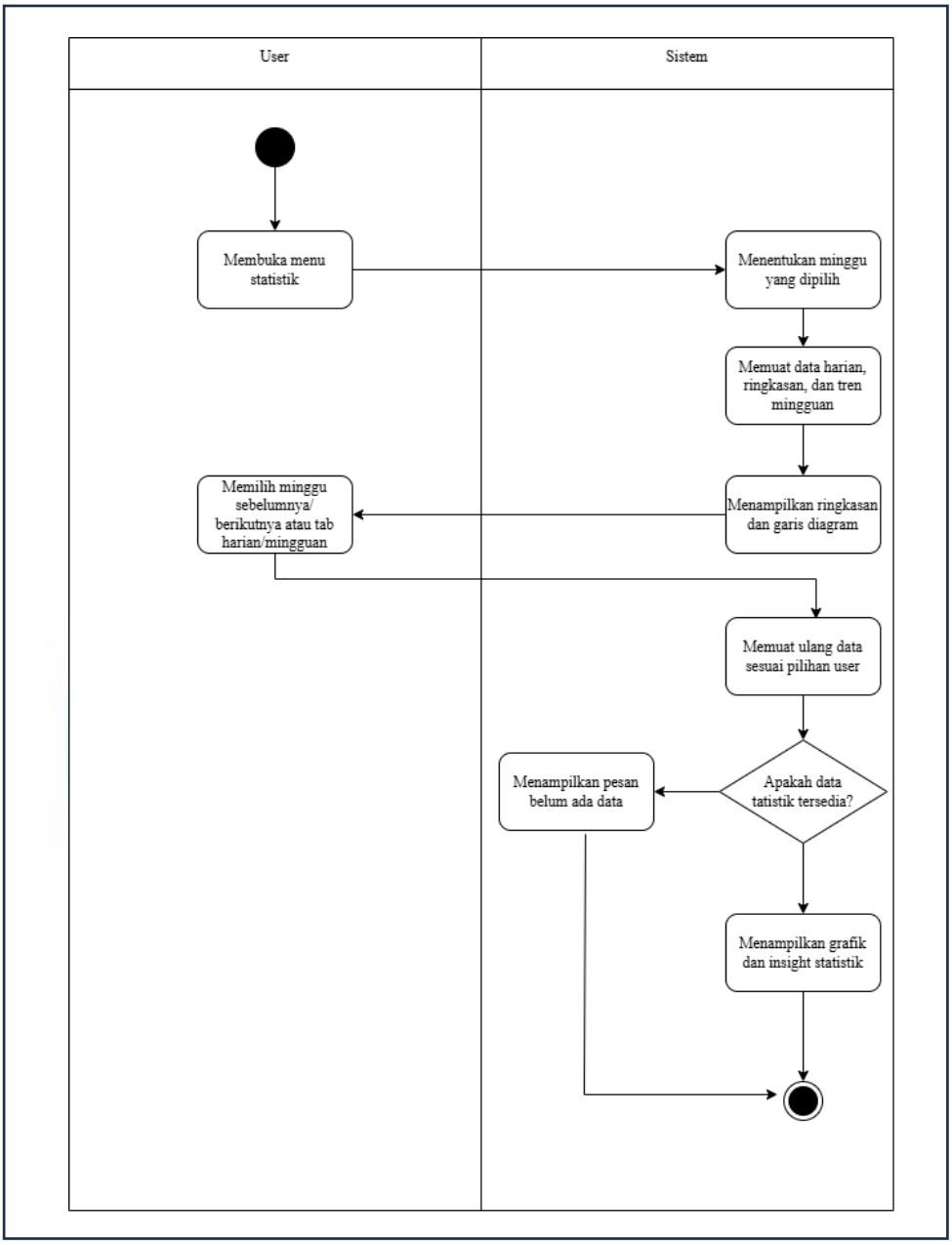


Sumber: data diolah oleh penulis (2026)

Gambar 4. 11 *Activity* Kelola *Reminder Habit*

Gambar di atas menjelaskan proses mengatur pengingat *Habit*. Pengguna dapat mengaktifkan atau menonaktifkan *Reminder* . Jika *Reminder* diaktifkan, pengguna memilih jam pengingat dan sistem akan menyimpan waktu tersebut serta menjadwalkan notifikasi. Jika *Reminder* dimatikan, sistem membatalkan notifikasi yang tersimpan.

11. Activity Statistik

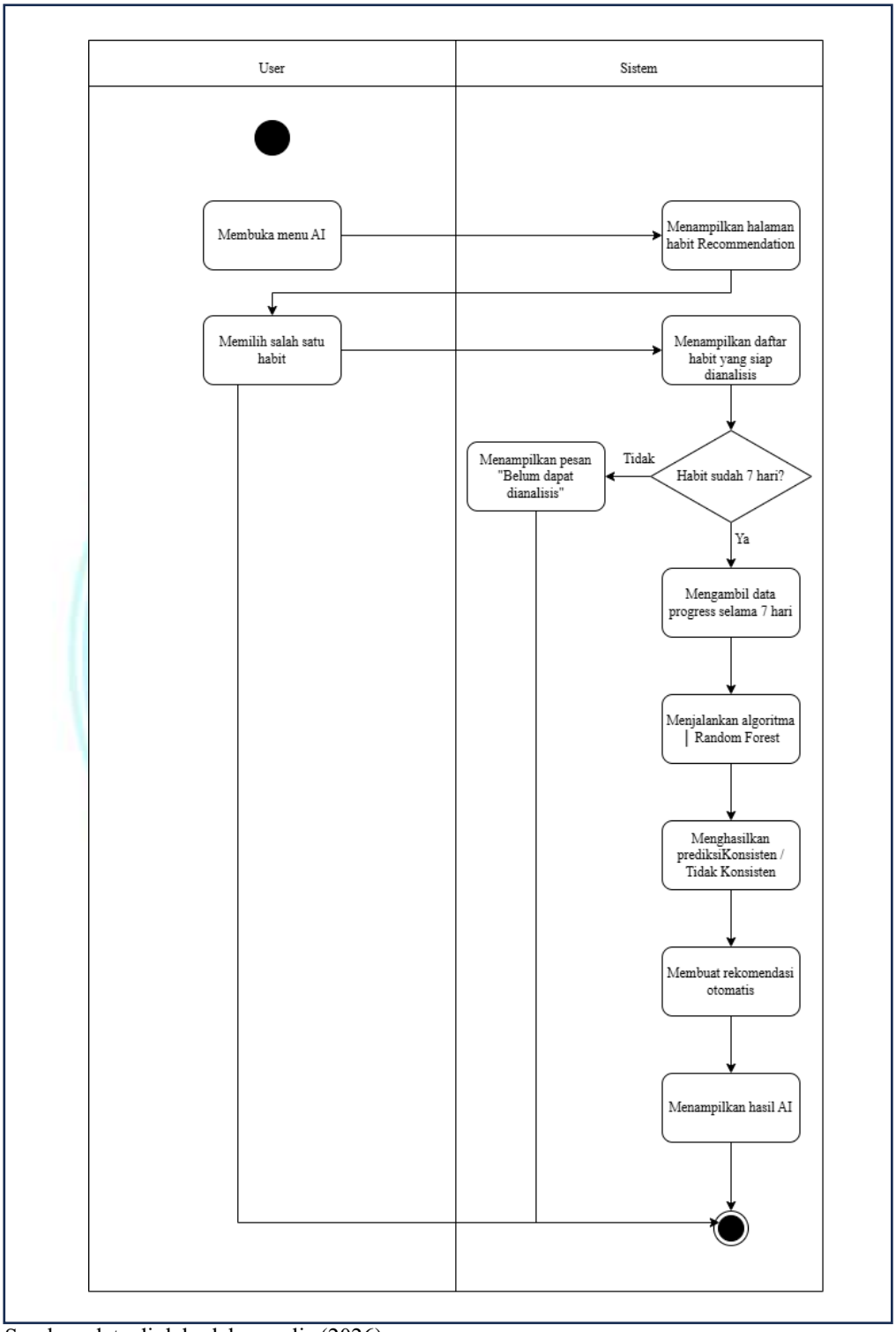


Sumber: data diolah oleh penulis (2026)

Gambar 4. 12 *Activity* Statistik *Habit*

Gambar di atas menjelaskan alur pengguna saat melihat statistik *Habit*. Sistem memuat data harian, ringkasan, dan tren mingguan. Pengguna dapat memilih minggu sebelumnya atau memilih tab statistik harian dan mingguan. Jika data tersedia, sistem menampilkan grafik dan insight statistik, sedangkan jika belum ada data maka sistem menampilkan pesan belum tersedia.

12. Activity Rekomendasi *AI Random Forest*



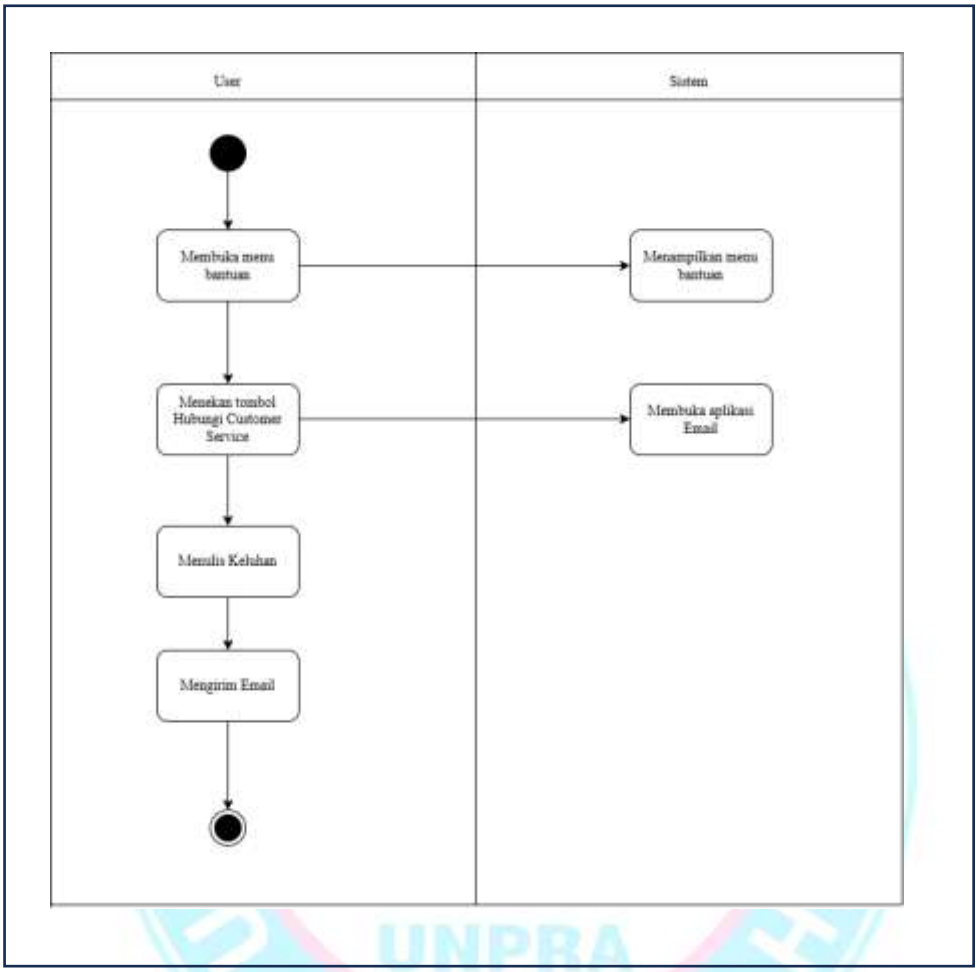
Sumber: data diolah oleh penulis (2026)

Gambar 4. 13 *Activity* Rekomendasi *AI*

Gambar di atas menjelaskan proses analisis kebiasaan menggunakan algoritma Random Forest pada aplikasi Habit Tracker. Proses dimulai ketika

pengguna membuka menu AI sehingga sistem menampilkan daftar habit yang dapat dianalisis. Selanjutnya pengguna memilih salah satu habit, kemudian sistem memeriksa apakah habit tersebut telah memiliki riwayat aktivitas minimal selama tujuh hari. Apabila syarat tersebut belum terpenuhi, sistem akan menampilkan informasi bahwa habit belum dapat dianalisis. Sebaliknya, apabila riwayat aktivitas telah memenuhi ketentuan, sistem mengambil data progress pengguna selama tujuh hari terakhir sebagai data masukan untuk algoritma Random Forest. Selanjutnya algoritma melakukan proses klasifikasi untuk menghasilkan prediksi tingkat konsistensi pengguna dengan kategori Konsisten atau Tidak Konsisten. Berdasarkan hasil prediksi tersebut, sistem secara otomatis menghasilkan rekomendasi yang sesuai untuk membantu pengguna meningkatkan konsistensi dalam menjalankan kebiasaan. Hasil prediksi beserta rekomendasi kemudian ditampilkan pada halaman rekomendasi AI kepada pengguna.

13. Activity Diagram

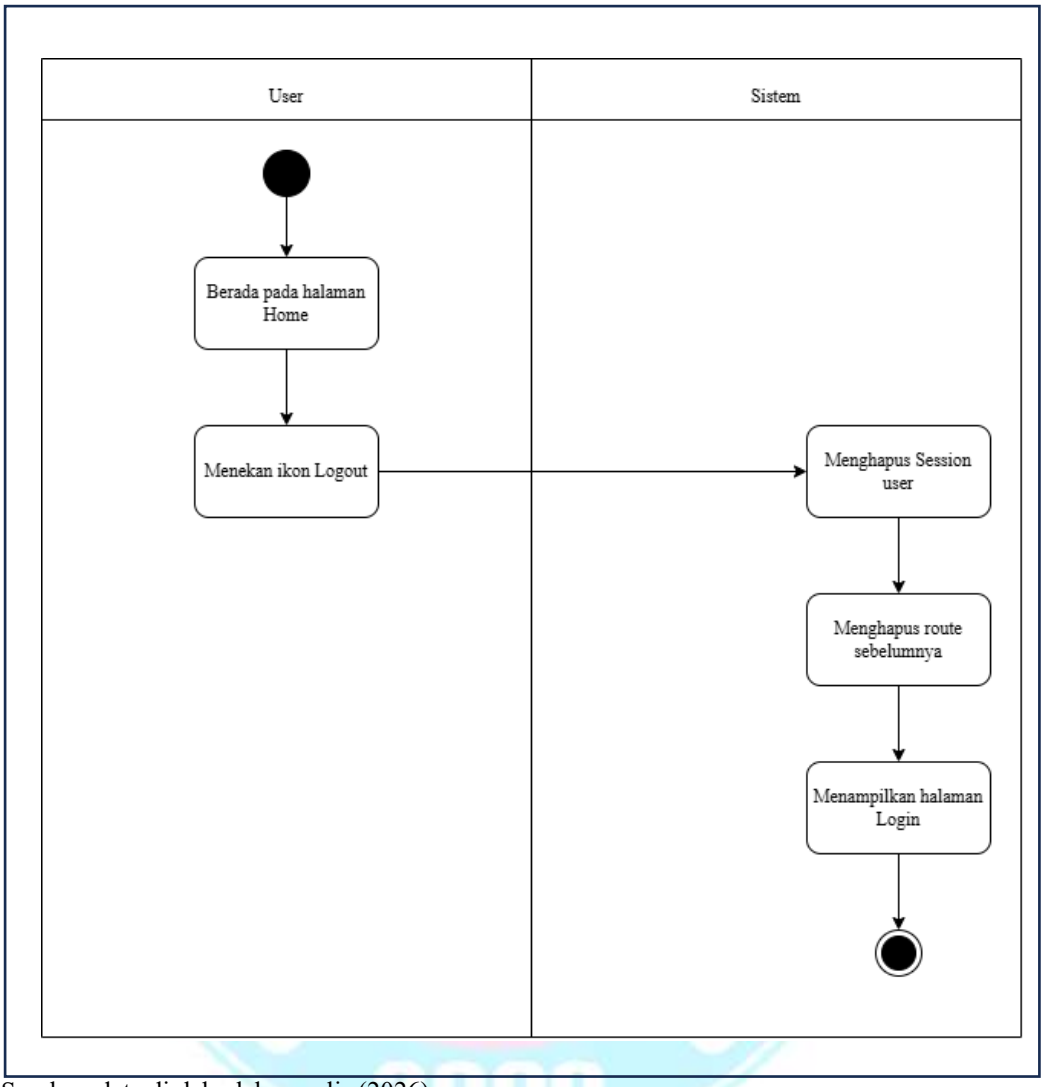


Sumber: data diolah oleh penulis (2026)

Gambar 4. 14 Activity Diagram Hubungi *Customer Service*

Gambar di atas menjelaskan proses pengguna menghubungi *Customer Service* melalui aplikasi *Habit Tracker*. Proses dimulai ketika pengguna membuka menu Bantuan atau *Customer Service*, kemudian sistem menampilkan halaman bantuan yang berisi informasi layanan. Selanjutnya pengguna menekan tombol Hubungi *Customer Service*, sehingga sistem membuka media komunikasi yang telah disediakan, seperti *WhatsApp* atau *email*. Pengguna kemudian menuliskan keluhan, pertanyaan, atau kendala yang dialami selama menggunakan aplikasi dan mengirimkannya kepada *Customer Service*. Setelah pesan berhasil dikirim, pengguna menunggu balasan dari *Customer Service* sebagai tindak lanjut terhadap permasalahan yang disampaikan.

14. Activity *Logout*



Sumber: data diolah oleh penulis (2026)

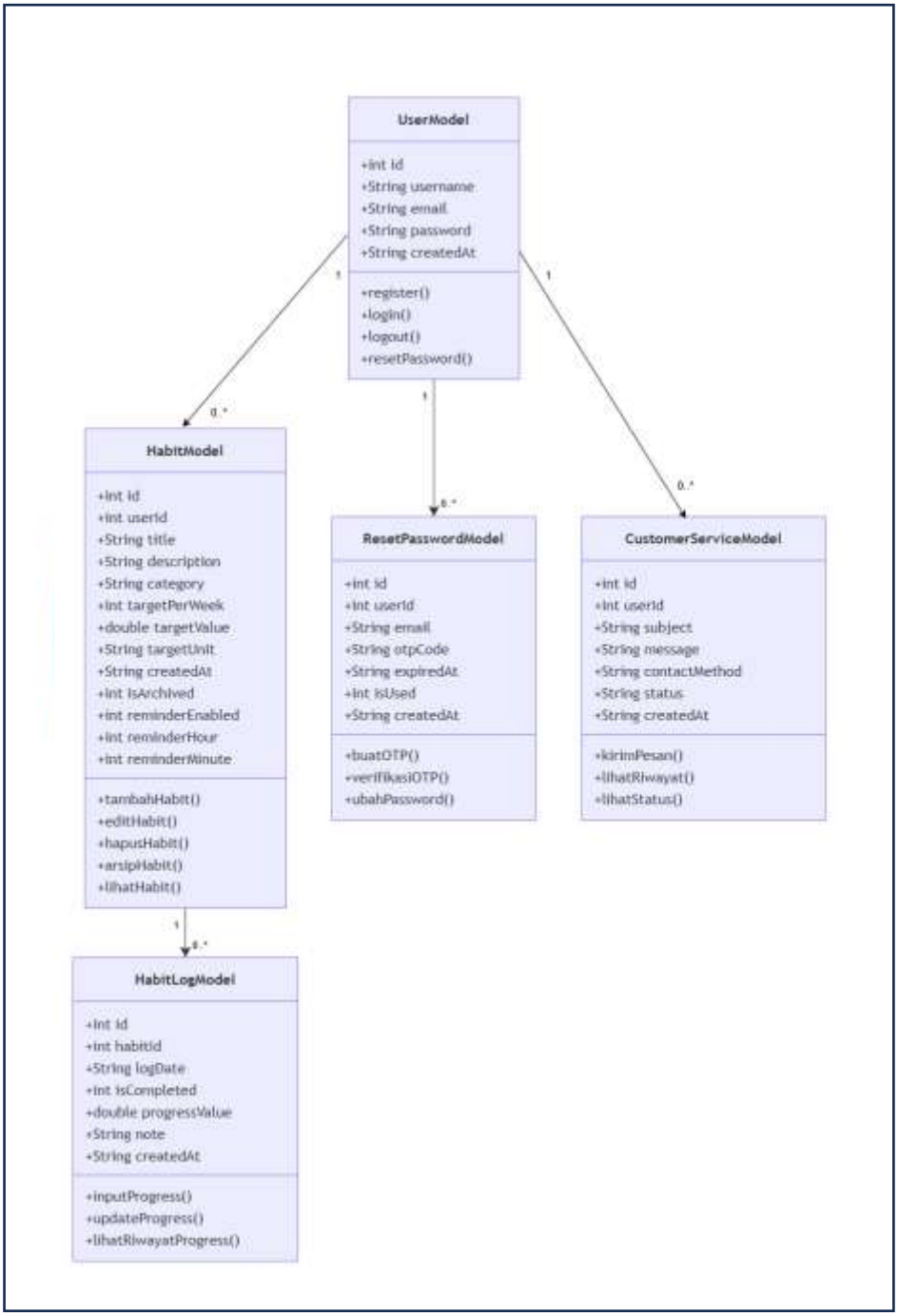
Gambar 4. 15 *Activity Logout*

Gambar di atas menjelaskan proses keluar dari akun pengguna. Saat pengguna menekan ikon *Logout*, sistem akan menghapus session *login* , membersihkan route sebelumnya, lalu menampilkan kembali halaman *Login* . Proses ini memastikan pengguna harus *login* ulang untuk mengakses aplikasi.

c. *Class Diagram*

Class Diagram digunakan untuk menggambarkan struktur class yang terdapat pada aplikasi *Habit Tracker*. Diagram ini menampilkan class utama, atribut, method, serta hubungan antar class yang digunakan dalam sistem. *Class*

Diagram berperan penting dalam perancangan karena menjadi dasar dalam pembuatan struktur data dan proses implementasi kode program.



Sumber: data diolah oleh penulis (2026)

Gambar 4. 16 *Class Diagram*

Gambar di atas menunjukkan *Class Diagram* pada aplikasi *Habit Tracker* yang menggambarkan struktur kelas beserta hubungan antar kelas yang digunakan

dalam sistem. *Class diagram* terdiri atas lima kelas utama, yaitu *UserModel*, *HabitModel*, *HabitLogModel*, *ResetPasswordModel*, dan *CustomerServiceModel*. Masing-masing kelas memiliki atribut dan metode yang mendukung proses utama pada aplikasi.

UserModel merupakan kelas utama yang digunakan untuk mengelola data pengguna. Kelas ini memiliki atribut berupa *id*, *username*, *email*, *password*, dan *createdAt*, serta menyediakan fungsi *register*(), *login*(), *logout*(), dan *resetPassword*(). Satu pengguna dapat memiliki banyak data habit, permintaan reset password, dan riwayat komunikasi dengan *Customer Service*. *HabitModel* digunakan untuk mengelola data *habit* yang dibuat oleh pengguna. Kelas ini menyimpan informasi berupa *title*, *description*, *category*, *targetPerWeek*, *targetValue*, *targetUnit*, *createdAt*, *isArchived*, serta pengaturan *reminder* yang terdiri atas *reminderEnabled*, *reminderHour*, dan *reminderMinute*. Metode yang tersedia meliputi *tambahHabit*(), edit*Habit*(), hapus*Habit*(), arsip*Habit*(), dan lihat*Habit*() untuk mendukung proses pengelolaan *habit* pada aplikasi.

HabitLogModel digunakan untuk menyimpan riwayat *progress* harian dari setiap *habit*. Kelas ini memiliki atribut berupa *logDate*, *isCompleted*, *progressValue*, *note*, dan *createdAt*, serta menyediakan fungsi *inputProgress*(), *updateProgress*(), dan *lihatRiwayatProgress*(). Data yang tersimpan pada kelas ini digunakan sebagai dasar perhitungan pencapaian target *habit*, pemberian *Diamond Badge*, serta sebagai data masukan bagi algoritma *Random Forest* dalam melakukan analisis tingkat konsistensi pengguna berdasarkan aktivitas selama tujuh hari terakhir. *ResetPasswordModel* digunakan untuk mendukung proses pemulihan akun melalui mekanisme *One Time Password* (*OTP*). Kelas ini menyimpan informasi berupa *email*, *otpCode*, *expiredAt*, *isUsed*, dan *createdAt*, serta menyediakan fungsi buat*OTP*(), verifikasi*OTP*(), dan ubah*Password*().

CustomerServiceModel digunakan untuk mengelola data komunikasi antara pengguna dan layanan *Customer Service*. Kelas ini menyimpan informasi berupa *subject*, *message*, *contactMethod*, status, dan *createdAt*, serta menyediakan fungsi kirimPesan(), lihatRiwayat(), dan lihatStatus() untuk mendukung proses penyampaian dan pemantauan keluhan pengguna. Hubungan antar kelas pada sistem menggunakan relasi one-to-many (1..*), yaitu satu *UserModel* dapat

memiliki banyak *HabitModel*, *ResetPasswordModel*, dan *CustomerServiceModel*, sedangkan satu *HabitModel* dapat memiliki banyak *HabitLogModel*. Struktur kelas tersebut mendukung seluruh proses utama pada aplikasi, mulai dari pengelolaan data pengguna, pengelolaan habit, pencatatan progress harian, pemulihan akun, hingga layanan *Customer Service.*

4.2.2 Perancangan *Database*

Perancangan *Database* pada aplikasi *Habit Tracker* digunakan untuk menyimpan dan mengelola data pengguna, data kebiasaan, riwayat aktivitas kebiasaan, serta data pendukung proses *Reset Password* secara terstruktur. *Database* yang digunakan adalah *SQLite* karena dapat berjalan secara lokal pada aplikasi Android tanpa membutuhkan server eksternal. Pada rancangan ini terdapat empat tabel utama, yaitu tabel *Users*, tabel *Habit*s, tabel *Habit*_logs, tabel hubungi customer service, dan tabel reset_*Password*.

Tabel 4. 4 *Users*

Nama Field	Tipe Data	Size	Keterangan
id	int	-	Primary key (*)
Username	varchar	-	Unique
email	varchar	-	Unique
Password	varchar	-	*Password* pengguna yang disimpan dalam bentuk hash
created_at	datetime	-	

Sumber: data diolah oleh penulis (2026)

Tabel *Users* digunakan untuk menyimpan data akun pengguna yang melakukan registrasi pada aplikasi. Data yang disimpan meliputi id, *Username* , email, *Password*, dan created_at. Field id berfungsi sebagai primary key untuk membedakan setiap pengguna, sedangkan *Username* dan email bersifat unique agar tidak terjadi penggunaan akun yang sama lebih dari satu kali. Field *Password* digunakan untuk menyimpan *Password* pengguna dalam bentuk hash, sehingga *Password* asli tidak tersimpan secara langsung di dalam *Database*. Tabel ini menjadi dasar untuk proses *Register*, *login*, *Reset Password*, dan pengelolaan session pengguna.

Tabel 4. 5 *Habits*

Nama Field	Tipe Data	Size	Keterangan
id	Integer	-	Primary Key (*)
user_id	Integer	-	Foreign Key (**) yang mengacu pada tabel Users
title	Text	-	Nama habit
description	Text	-	Deskripsi habit
category	Text	-	Kategori habit
target_per_week	Integer	-	Target pelaksanaan habit dalam satu minggu
target_value	Real	-	Nilai target yang harus dicapai pengguna pada setiap periode pencatatan progress
target_unit	Text	-	Satuan target, misalnya *Kali*, *Menit*, *Jam*, atau *Halaman*
created_at	Text	-	Tanggal dan waktu habit dibuat
is_archived	Integer	-	Status habit (0 = aktif, 1 = diarsipkan)
reminder_enabled	Integer	-	Status reminder (0 = tidak aktif, 1 = aktif)
reminder_hour	Integer	-	Jam pelaksanaan reminder
reminder_minute	Integer	-	Menit pelaksanaan reminder

Sumber: data diolah oleh penulis (2026)

Tabel *Habits* digunakan untuk menyimpan seluruh informasi mengenai *habit* yang dibuat oleh pengguna pada aplikasi *Habit Tracker*. Setiap data habit memiliki *id* sebagai primary *key* dan *user_id* sebagai foreign *key* yang menghubungkan *habit* dengan pengguna yang membuatnya. Tabel ini menyimpan informasi berupa nama *habit* (*title*), deskripsi (*description*), kategori (*category*), target pelaksanaan dalam satu minggu (*target_per_week*), nilai target (*target_value*), serta satuan target (*target_unit*) yang digunakan sebagai acuan dalam proses pencatatan progress harian. Selain itu, tabel ini juga menyimpan tanggal pembuatan *habit* (*created_at*), status arsip (*is_archived*), serta pengaturan reminder yang meliputi status reminder (*reminder_enabled*), jam (*reminder_hour*), dan menit (*reminder_minute*). Data yang tersimpan pada tabel *Habits* menjadi dasar dalam proses pengelolaan *habit*, pencatatan progress harian, penjadwalan notifikasi, serta analisis tingkat konsistensi pengguna menggunakan algoritma *Random Forest*.

Tabel 4. 6 *Habit*s Log

Nama Field	Tipe Data	Size	Keterangan
id	Integer	-	Primary Key (*)
habit_id	Integer	-	Foreign Key (**) yang mengacu pada tabel Habits
log_date	Text	-	Tanggal pencatatan progress habit
is_completed	Integer	-	Status penyelesaian habit (0 = belum selesai, 1 = selesai)
progress_value	Real	-	Nilai progress yang diinput pengguna
note	Text	-	Catatan tambahan mengenai progress habit
created_at	Text	-	Tanggal dan waktu data progress disimpan

Sumber: data diolah oleh penulis (2026)

Tabel *Habit_Logs* digunakan untuk menyimpan riwayat *progress* harian dari setiap *habit* yang dimiliki pengguna. Setiap data progress memiliki *id* sebagai *primary key* dan *habit_id* sebagai *foreign key* yang menghubungkan data *progress* dengan tabel *Habits*. Tabel ini menyimpan informasi berupa tanggal pencatatan *progress* (*log_date*), status penyelesaian *habit* (*is_completed*), nilai *progress* yang diinput pengguna (*progress_value*), catatan tambahan (*note*), serta waktu penyimpanan data (*created_at*). Seluruh data pada tabel *Habit_Logs* menjadi sumber data utama dalam proses analisis menggunakan algoritma *Random Forest*. Sistem memanfaatkan riwayat *progress* pengguna selama tujuh hari terakhir untuk menghasilkan prediksi tingkat konsistensi dalam kategori Konsisten atau Tidak Konsisten, kemudian memberikan rekomendasi otomatis sesuai hasil analisis.

Tabel 4. 7 *Reset Password*

Nama Field	Tipe Data	Size	Keterangan
id	Integer	-	Primary Key (*)
user_id	Integer	-	Foreign Key (**) yang mengacu pada tabel Users
email	Text	-	Alamat email pengguna yang melakukan reset password
otp_code	Text	-	Kode OTP yang telah dienkripsi (hash SHA-256)
expired_at	Text	-	Waktu berakhirnya masa berlaku kode OTP
is_used	Integer	-	Status penggunaan OTP (0 = belum digunakan, 1 = sudah digunakan)

created_at	Text	-	Waktu permintaan reset password dibuat

Sumber: data diolah oleh penulis (2026)

Tabel *Reset_Password* digunakan untuk menyimpan data permintaan *reset password* yang dilakukan oleh pengguna. Setiap data memiliki id sebagai *primary key* dan *user_id* sebagai *foreign key* yang menghubungkan data dengan tabel *Users*. Tabel ini menyimpan alamat *email* pengguna, kode *OTP* yang telah dienkripsi menggunakan algoritma SHA-256 (*otp_code*), waktu berakhirnya masa berlaku OTP (*expired_at*), status penggunaan kode *OTP* (*is_used*), serta waktu pembuatan permintaan *reset password* (*created_at*). Struktur tabel ini dirancang untuk meningkatkan keamanan proses pemulihan akun dengan memastikan bahwa setiap kode *OTP* hanya dapat digunakan satu kali dan hanya berlaku dalam batas waktu tertentu.

Tabel 4.8 Hubungi *Customer Service*

Nama Field	Tipe Data	Size	Keterangan
Id	Integer	-	Primary Key (*)
User_Id	Integer	-	Foreign Key (**) Yang Mengacu Pada Tabel Users
Subject	Text	-	Judul Pertanyaan Atau Keluhan
Message	Text	-	Isi Pertanyaan Atau Keluhan Pengguna
Contact_Method	Text	-	Media Komunikasi Yang Dipilih (Whatsapp Atau Email)
Status	Text	-	Status Penanganan Keluhan (Pending, Diproses, Atau Selesai)
Created_At	Text	-	Tanggal Dan Waktu Pengiriman Keluhan

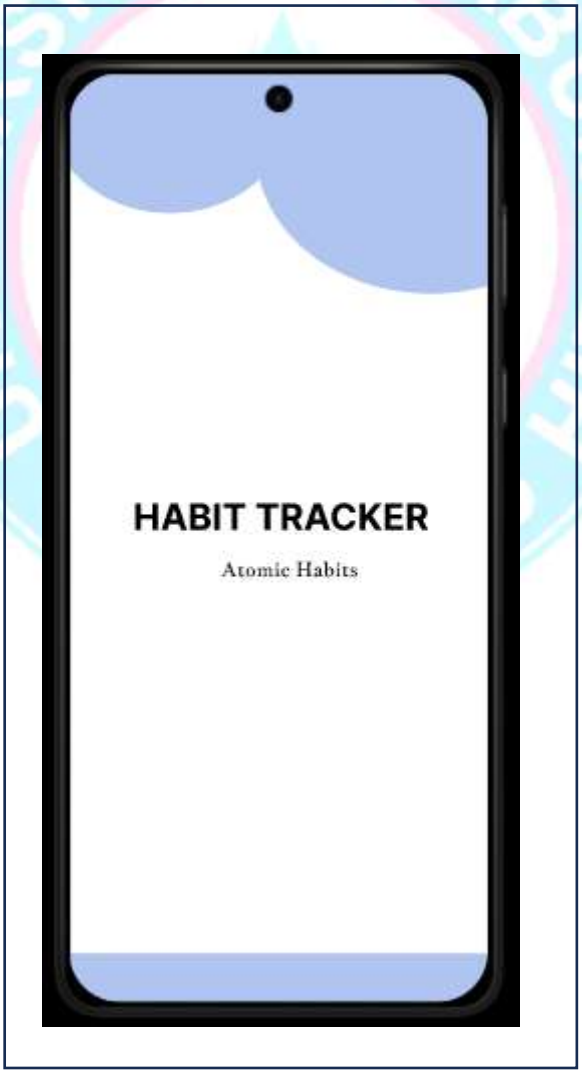
Sumber: data diolah oleh penulis (2026)

Tabel *Customer_Service* digunakan untuk menyimpan data pertanyaan, saran, maupun keluhan yang dikirimkan oleh pengguna kepada *Customer Service* melalui aplikasi *Habit Tracker*. Setiap data memiliki *id* sebagai *primary key* dan *user_id* sebagai *foreign key* yang menghubungkan data dengan tabel *Users*. Tabel ini menyimpan informasi berupa judul keluhan (*subject*), isi pesan (*message*), media komunikasi yang dipilih (*contact_method*), status penanganan keluhan (status), serta waktu pengiriman pesan (created_at). Data pada tabel ini digunakan

untuk mendokumentasikan riwayat komunikasi antara pengguna dan Customer *Service* sehingga setiap keluhan dapat dipantau proses penyelesaiannya.

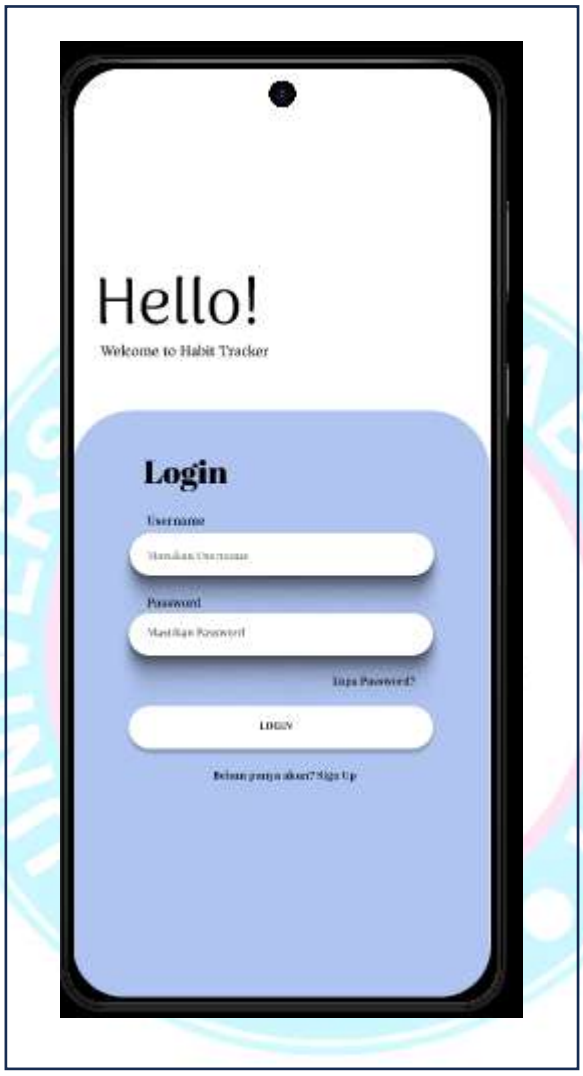
4.2.3 Perancangan Antarmuka (*UI*)

Perancangan antarmuka dilakukan untuk menghasilkan tampilan aplikasi yang menarik, mudah digunakan, dan mendukung prinsip *Atomic Habits*. Antarmuka dirancang menggunakan *Figma* dengan memperhatikan kemudahan navigasi, konsistensi warna, serta kejelasan informasi. *UI* yang baik diharapkan dapat meningkatkan keterlibatan pengguna dan mendorong konsistensi dalam membangun kebiasaan positif. Berikut rancangan *UI* aplikasi *Habit Tracker.*



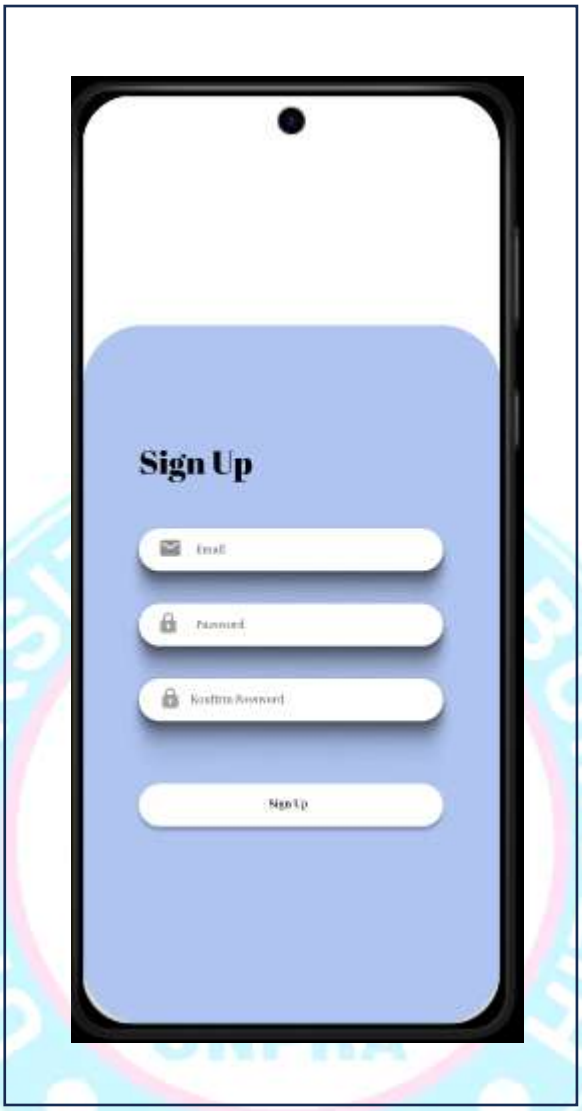
Gambar 4. 17 Halaman *SplashScreen*

Gambar di atas menampilkan halaman *SplashScreen* sebagai tampilan awal ketika aplikasi pertama kali dibuka. Halaman ini berfungsi sebagai identitas awal aplikasi *Habit Tracker* sebelum pengguna diarahkan ke halaman *login* .



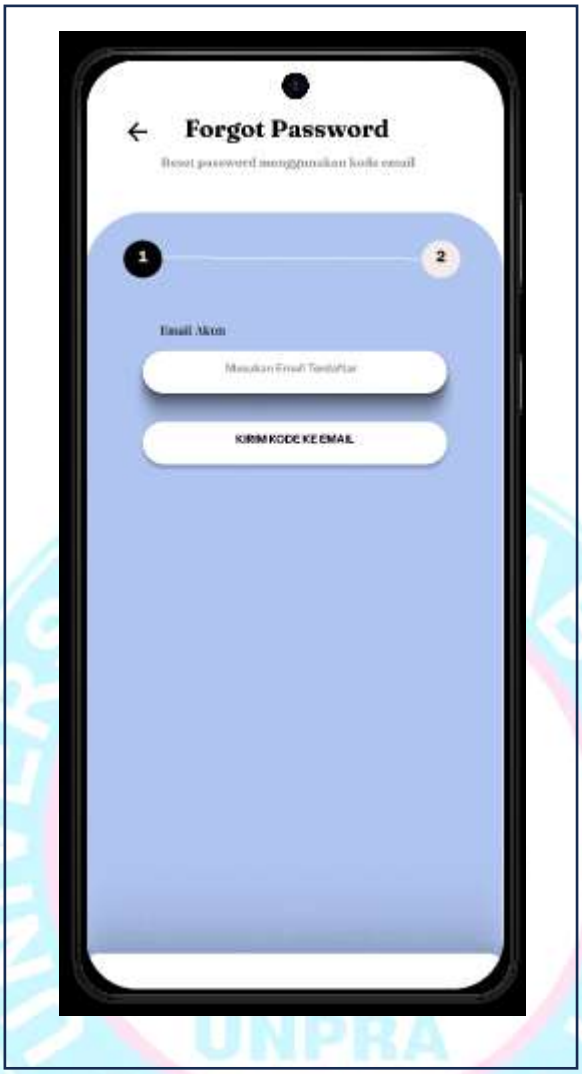
Gambar 4. 18 Halaman *Log In*

Gambar di atas menampilkan halaman *Log In* yang digunakan pengguna untuk masuk ke dalam aplikasi. Pengguna perlu mengisi *Username* dan *Password* yang telah terdaftar agar dapat mengakses fitur utama aplikasi.



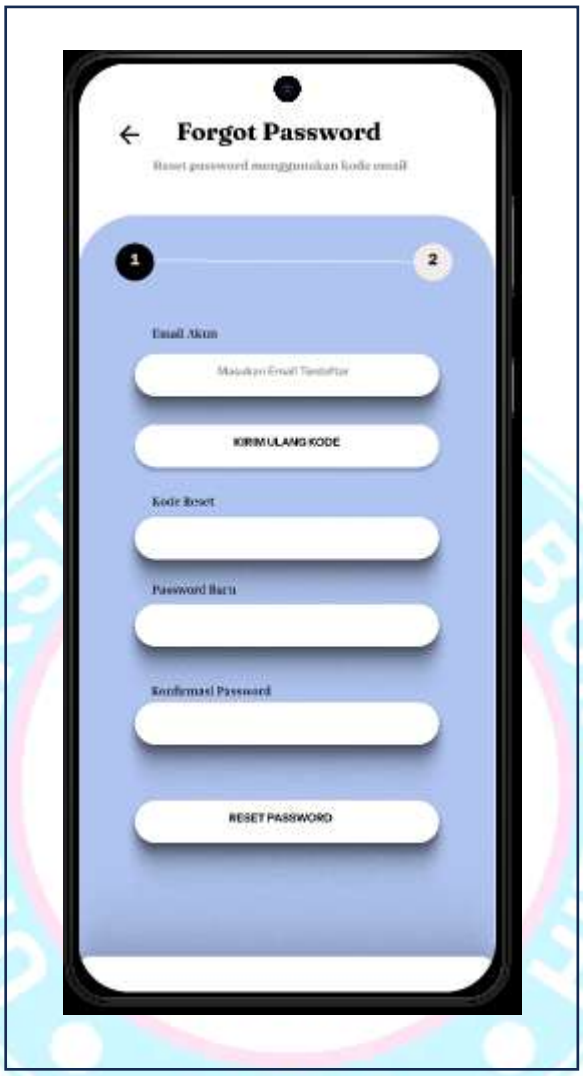
Gambar 4. 19 Halaman *Register*

Gambar di atas menampilkan halaman *register* yang digunakan oleh pengguna baru untuk membuat akun. Pada halaman ini pengguna mengisi username, email, *Password*, dan konfirmasi *Password* sebelum dapat menggunakan aplikasi *Habit Tracker*.



Gambar 4. 20 Halaman Lupa *Password*

Gambar di atas menampilkan halaman Lupa *Password* yang digunakan ketika pengguna mengalami kendala saat masuk ke akun. Pengguna dapat memasukkan email terdaftar untuk menerima kode *Reset Password*.



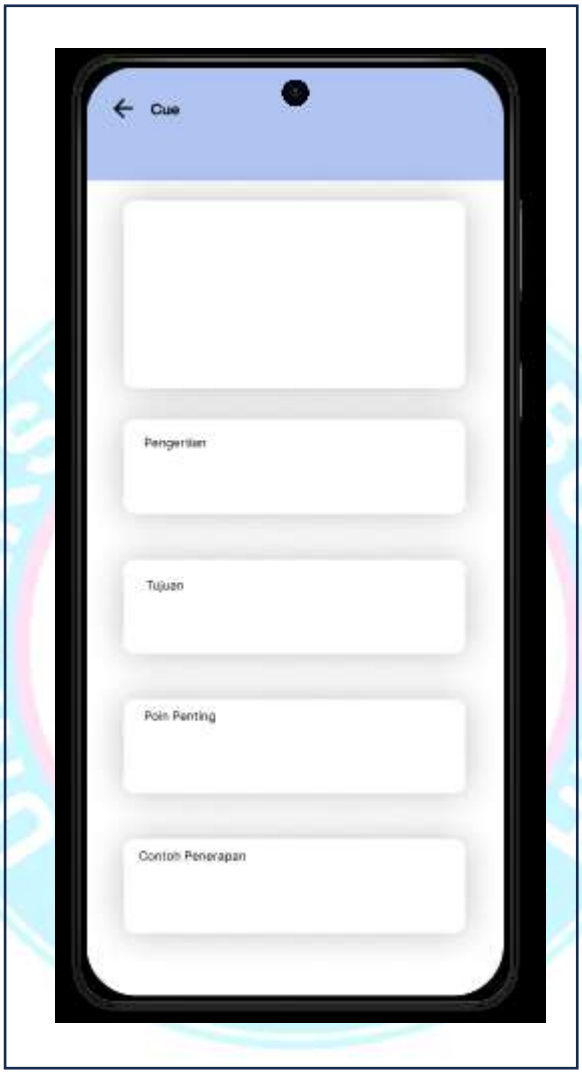
Gambar 4. 21 Halaman *Reset Password*

Gambar di atas menampilkan halaman *Reset Password* yang digunakan untuk membuat *Password* baru. Pengguna perlu memasukkan kode reset, *Password* baru, dan konfirmasi *Password* agar akun dapat digunakan kembali.

Gambar 4. 22 Halaman Home

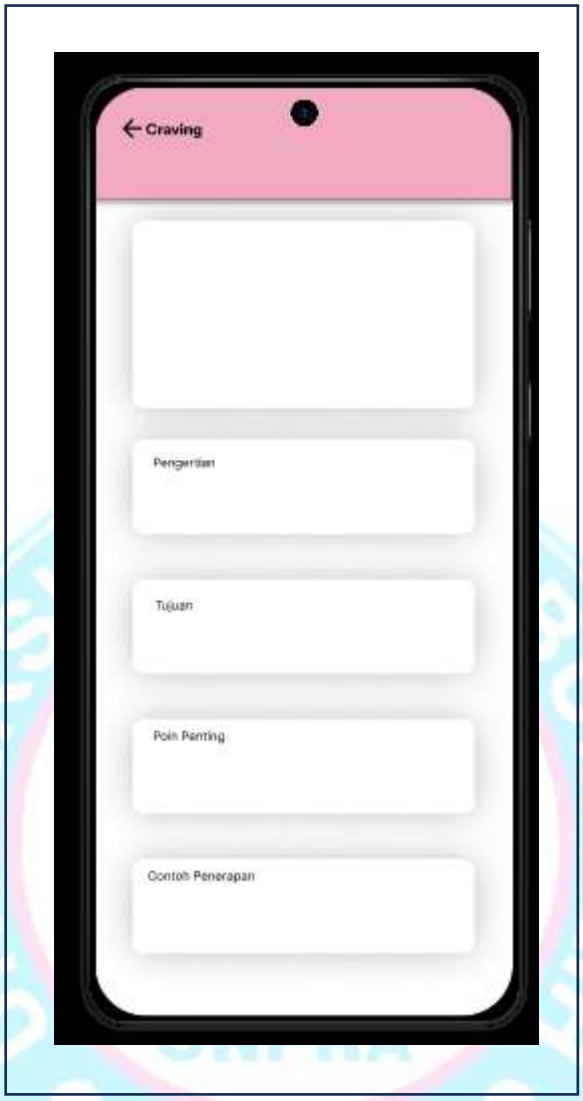
Gambar di atas menampilkan halaman Home sebagai halaman utama aplikasi setelah pengguna berhasil melakukan login. Pada bagian atas halaman ditampilkan sapaan kepada pengguna beserta kalimat motivasi untuk mendorong konsistensi dalam membangun kebiasaan positif. Di sebelah kanan terdapat ikon Customer Service yang dapat digunakan pengguna untuk memperoleh bantuan apabila mengalami kendala saat menggunakan aplikasi, serta ikon Logout untuk keluar dari akun. Pada bagian tengah halaman ditampilkan kalender aktivitas yang berfungsi untuk membantu pengguna melihat tanggal dan memantau aktivitas kebiasaan. Di bawah kalender terdapat menu Atomic Habits Framework yang terdiri dari Cue, Craving, Response, dan Reward sebagai media edukasi mengenai prinsip pembentukan kebiasaan. Sementara itu, pada bagian bawah tersedia

navigation bar yang memberikan akses menuju halaman Home, Habit, Statistik, dan Prediksi, sehingga pengguna dapat berpindah antarfitur dengan mudah.



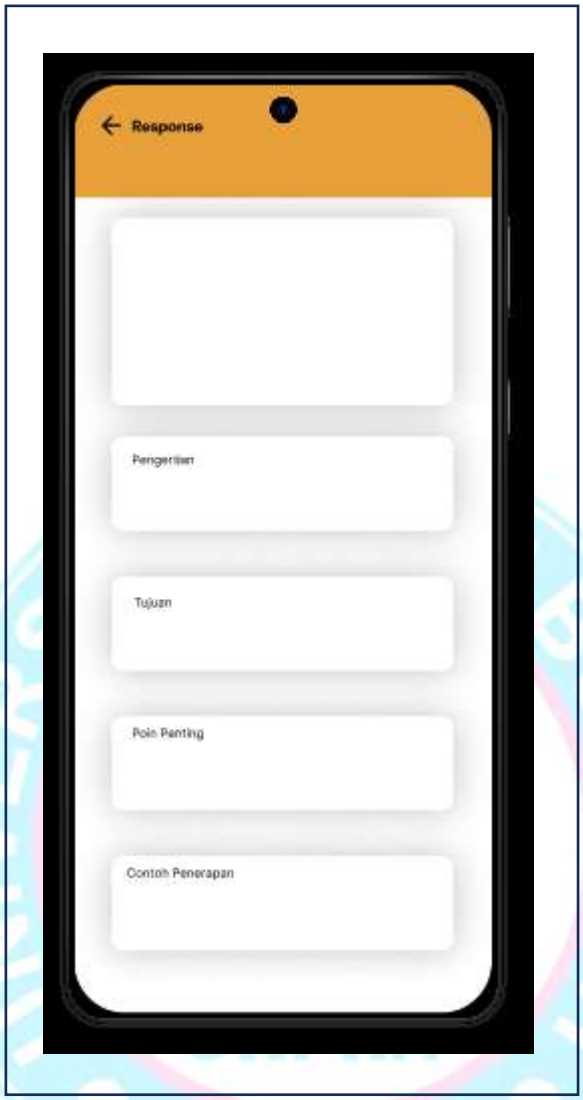
Gambar 4. 23 Halaman *Cue*

Gambar di atas menampilkan Halaman Cue yang berisi informasi mengenai konsep Cue dalam prinsip *Atomic Habits*. Pada halaman ini ditampilkan materi yang terdiri dari pengertian, tujuan, poin-poin penting, serta contoh penerapan Cue dalam membangun kebiasaan positif. Halaman ini dirancang untuk membantu pengguna memahami pentingnya isyarat atau pemicu sebagai langkah awal dalam membentuk suatu kebiasaan.



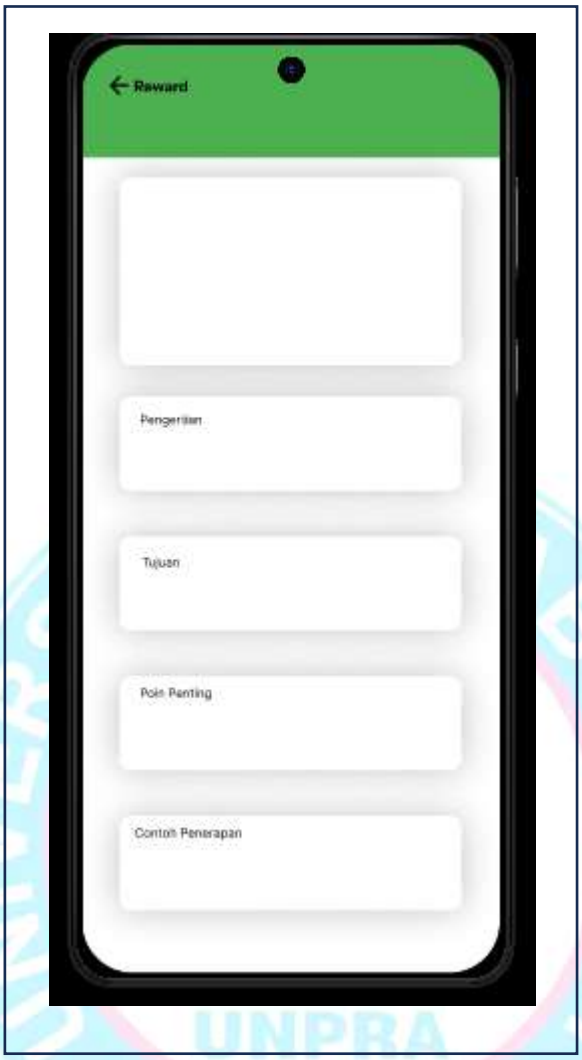
Gambar 4. 24 Halaman *Craving*

Gambar di atas menampilkan halaman *Craving* yang menjelaskan tentang keinginan dalam membangun kebiasaan. Halaman ini membantu pengguna memahami pentingnya dorongan atau motivasi agar kebiasaan lebih mudah dilakukan.



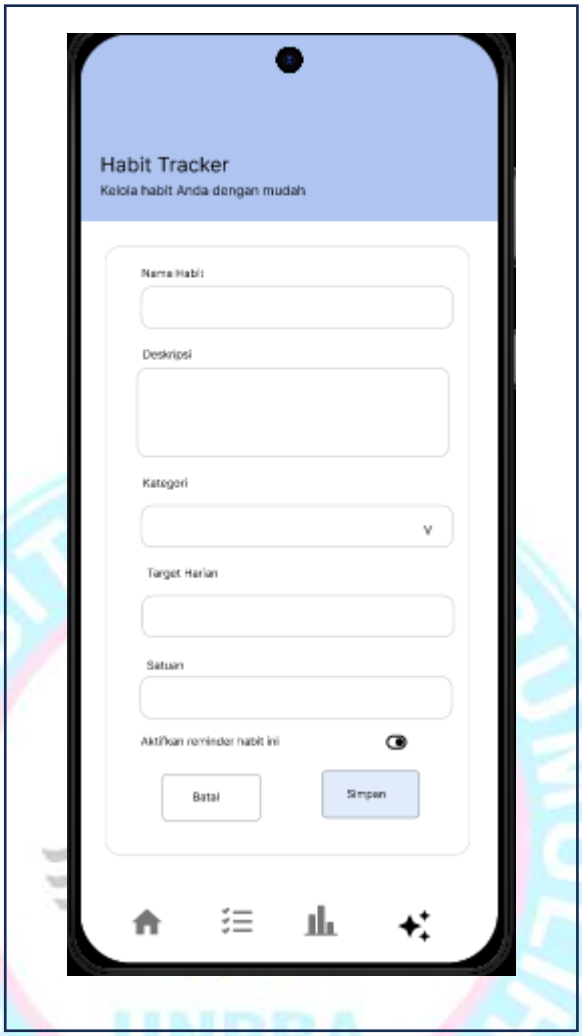
Gambar 4. 25 Halaman *Response*

Gambar di atas menampilkan halaman *Response* yang menjelaskan tindakan nyata dalam menjalankan kebiasaan. Halaman ini menekankan bahwa kebiasaan terbentuk ketika pengguna benar-benar melakukan aktivitas yang telah direncanakan.



Gambar 4. 26 Halaman *Reward*

Gambar di atas menampilkan halaman *Reward* yang menjelaskan bentuk penghargaan setelah pengguna menyelesaikan kebiasaan. Halaman ini bertujuan memberi pemahaman bahwa *reward* dapat meningkatkan motivasi pengguna untuk tetap konsisten.



Gambar 4. 27 Halaman Tambah *Habit*

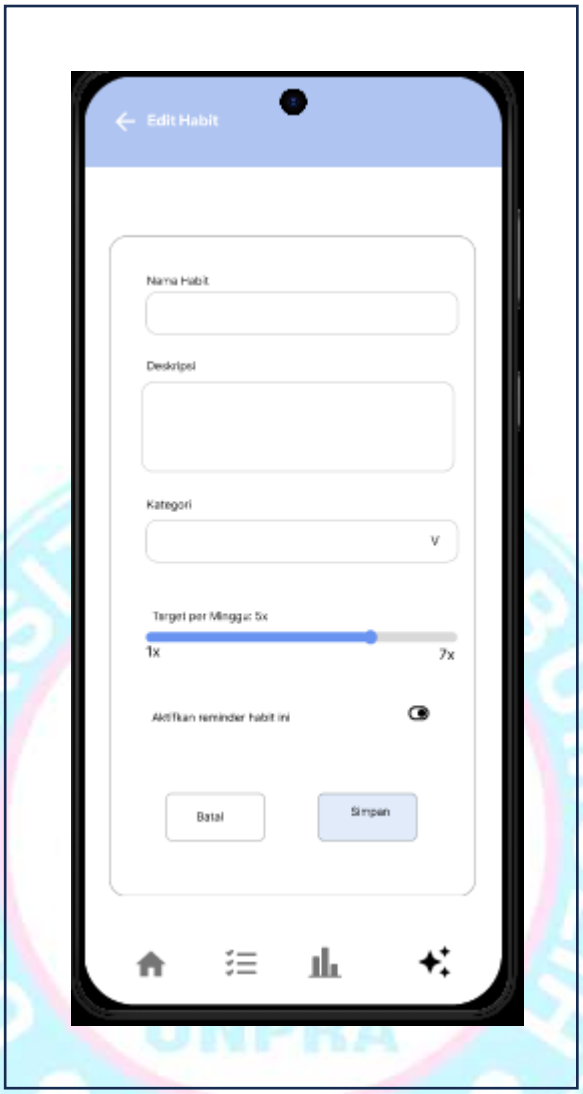
Gambar di atas menampilkan rancangan Halaman Tambah Habit yang digunakan untuk menambahkan kebiasaan baru ke dalam aplikasi. Pada halaman ini pengguna dapat mengisi beberapa informasi, yaitu nama habit, deskripsi, kategori, target harian, dan satuan target sesuai dengan kebiasaan yang ingin dibangun. Selain itu, tersedia fitur reminder yang dapat diaktifkan untuk memberikan pengingat kepada pengguna agar melaksanakan kebiasaan sesuai jadwal yang telah ditentukan. Pada bagian bawah halaman terdapat tombol Batal untuk membatalkan proses penambahan data serta tombol Simpan untuk menyimpan data habit ke dalam sistem. Di bagian paling bawah juga tersedia navigation bar yang memudahkan pengguna berpindah ke halaman Home, Habit, Statistik, dan Prediksi. Halaman ini dirancang agar proses penambahan kebiasaan dapat dilakukan dengan mudah, terstruktur, dan sesuai dengan kebutuhan pengguna.

Gambar 4. 28 Halaman Input Progress Habit

Gambar di atas menampilkan rancangan Halaman Input Progress Habit yang digunakan untuk membantu pengguna memantau dan mencatat perkembangan kebiasaan yang telah dibuat. Pada bagian atas halaman ditampilkan informasi jumlah habit yang telah diselesaikan serta total poin yang diperoleh pengguna sebagai bentuk penghargaan atas kebiasaan yang berhasil dilakukan. Di bagian tengah terdapat daftar habit yang menampilkan nama habit, target harian, serta kolom input progress yang digunakan untuk mencatat perkembangan aktivitas yang telah dilakukan. Setelah mengisi progress, pengguna dapat menekan tombol Simpan Progress untuk menyimpan data ke dalam sistem. Selain itu, tersedia fitur Set Reminder untuk mengatur pengingat pelaksanaan habit serta menu Ubah Jam untuk mengubah waktu pengingat sesuai kebutuhan pengguna. Pada setiap habit juga tersedia menu opsi untuk melakukan pengelolaan data habit. Di bagian kanan

bawah halaman terdapat tombol Tambah Habit yang digunakan untuk menambahkan kebiasaan baru, sedangkan pada bagian paling bawah tersedia navigation bar yang memudahkan pengguna berpindah ke halaman Home, Habit, Statistik, dan Prediksi. Halaman ini dirancang agar pengguna dapat memantau, mencatat, dan mengelola perkembangan kebiasaan secara mudah dan terstruktur.

UNIVERSITAS PRABUMULIH
UNPRA
2000



Gambar 4. 29 Halaman Edit *Habit*

Gambar di atas menampilkan halaman Edit *Habit* yang digunakan untuk memperbarui data kebiasaan. Pengguna dapat mengubah nama *Habit*, deskripsi, kategori, target per minggu, dan pengaturan *Reminder* sesuai kebutuhan.

Gambar 4. 30 Halaman Hapus *Habit*

Gambar di atas menampilkan tampilan konfirmasi Hapus *Habit*. Halaman ini digunakan untuk memastikan apakah pengguna benar-benar ingin menghapus *Habit* agar data tidak terhapus secara tidak sengaja.

Gambar 4. 31 Halaman Statistik Harian

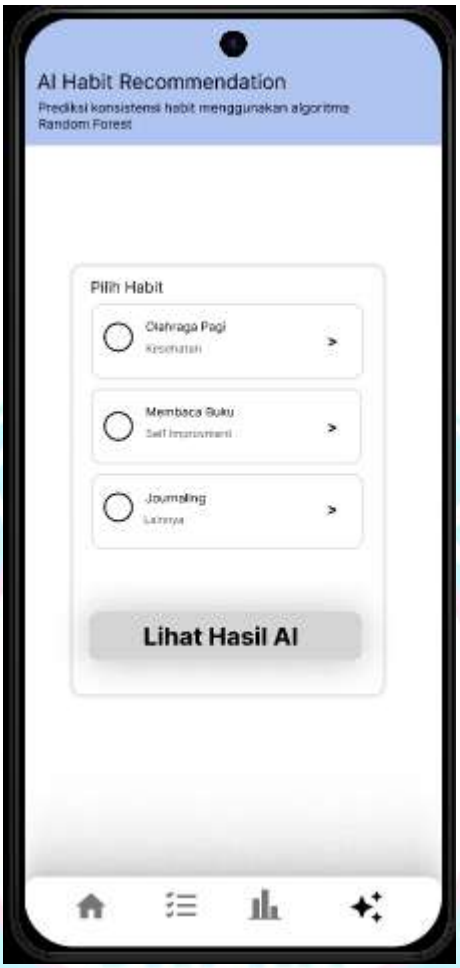
Gambar di atas menampilkan halaman Statistik Harian yang digunakan untuk melihat perkembangan kebiasaan pengguna dalam bentuk diagram harian. Halaman ini membantu pengguna mengevaluasi aktivitas yang telah dilakukan setiap hari.

Gambar 4. 32 Halaman Statistik Mingguan

Gambar di atas menampilkan halaman Statistik Mingguan yang digunakan untuk melihat perkembangan *Habit* dalam rentang waktu mingguan. Melalui halaman ini, pengguna dapat mengetahui pola konsistensi kebiasaan dari minggu ke minggu.

Gambar 4. Tampilan Habit yang akan dianalisis

Gambar di atas menampilkan halaman habit yang akan dianalisis menggunakan algoritma Random Forest. Pada halaman ini pengguna diberikan informasi mengenai proses analisis habit serta tersedia tombol Mulai Analisis AI untuk memulai proses prediksi tingkat konsistensi berdasarkan data aktivitas yang telah tercatat pada aplikasi.

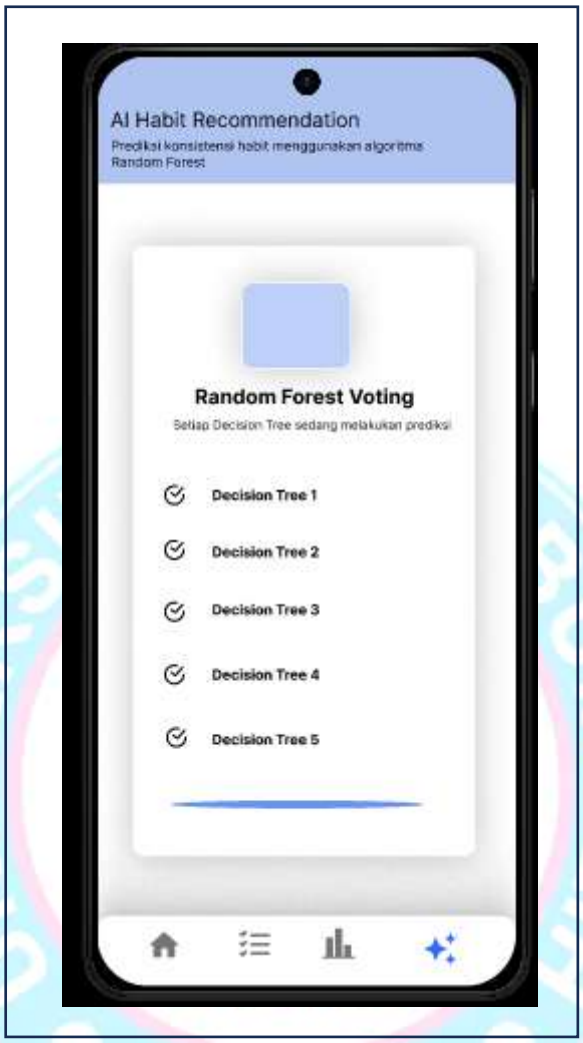


Gambar 4. 33 Halaman *List* Analisis *Habit*

Gambar di atas menampilkan halaman List Analisis *Habit* yang berisi daftar *Habit* yang dapat dianalisis. Pengguna dapat memilih salah satu *Habit* untuk melihat hasil analisis dan rekomendasi berdasarkan tingkat konsistensinya.

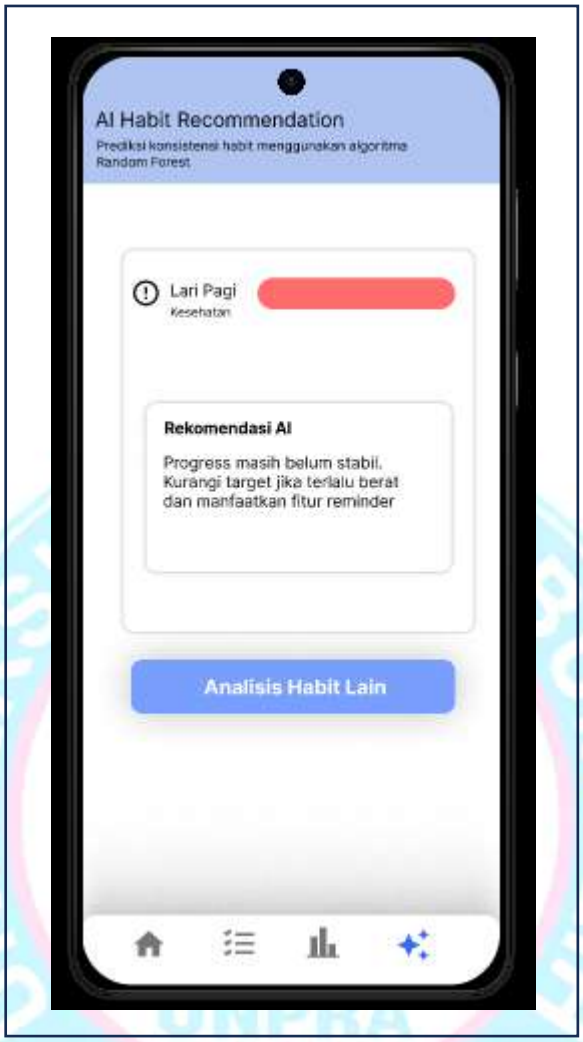
Gambar 4. 34 Proses Analisis AI

Gambar di atas menampilkan rancangan Halaman Proses Analisis Habit yang digunakan untuk menampilkan proses analisis data kebiasaan pengguna menggunakan algoritma Random Forest. Pada halaman ini sistem menampilkan tahapan analisis, seperti memuat data habit, menganalisis target, menghitung progress, menghitung achievement, dan menjalankan algoritma Random Forest. Setelah proses analisis selesai, sistem akan menampilkan hasil prediksi dan rekomendasi kepada pengguna.



Gambar 4. 35 Proses Random Forest Voting

Gambar di atas menampilkan rancangan Halaman Random Forest Voting yang digunakan untuk menampilkan proses pengambilan keputusan oleh algoritma Random Forest. Pada halaman ini ditampilkan proses prediksi yang dilakukan oleh beberapa Decision Tree sebelum sistem menentukan hasil prediksi akhir. Setelah seluruh proses voting selesai, sistem akan menampilkan hasil prediksi dan rekomendasi kepada pengguna.



Gambar 4. 36 Hasil Prediksi *AI*

Gambar di atas menampilkan rancangan Halaman Hasil Analisis AI yang digunakan untuk menampilkan hasil prediksi tingkat konsistensi habit menggunakan algoritma Random Forest. Pada halaman ini ditampilkan informasi habit yang dianalisis, hasil prediksi dalam bentuk indikator, serta rekomendasi yang diberikan oleh sistem untuk membantu pengguna meningkatkan konsistensi dalam menjalankan habit. Selain itu, tersedia tombol Analisis Habit Lain untuk melakukan analisis pada habit yang berbeda.

BAB V
HASIL DAN PEMBAHASAN

5.1 Spesifikasi Perangkat

Pada tahap implementasi sistem, perangkat yang digunakan mengacu pada kebutuhan pengembangan aplikasi berbasis Android . Spesifikasi perangkat ini digunakan untuk memastikan bahwa aplikasi dapat dikembangkan dan dijalankan secara optimal.

5.1.1 Perangkat Keras yang digunakan

Perangkat keras yang digunakan dalam proses pengembangan dan pengujian aplikasi *Habit Tracker* adalah sebagai berikut:

Tabel 5. 1 Perangkat Keras yang digunakan

No	Nama Perangkat Keras	Spesifikasi
1	Laptop *Acer*	1) Processor AMD Ryzen 3 3250U with Radeon Graphics (2.60 GHz) 2) Sistem Operasi *Windows 11 Home Single Language*1 3) RAM 8GB 4) Storage 466 GB
2	*Smartphone* Android	1) RAM 8GB 2) Penyimpanan 256GB 3) Sistem Operasi Android 16

Perangkat keras yang digunakan dalam proses pengembangan dan pengujian aplikasi *Habit Tracker* terdiri dari laptop sebagai perangkat utama pengembangan serta *smartphone* Android sebagai media pengujian aplikasi. Spesifikasi perangkat keras yang digunakan dapat dilihat pada Tabel 5.1. Laptop digunakan sebagai media utama dalam proses *coding*, perancangan sistem, serta *Debugging* aplikasi menggunakan *framework Flutter*. Selain itu, laptop juga digunakan untuk menjalankan emulator maupun *tools* pendukung lainnya selama proses pengembangan.

Sementara itu, *smartphone* Android digunakan untuk melakukan pengujian aplikasi secara langsung (*real device testing*). Pengujian ini bertujuan untuk

memastikan bahwa aplikasi dapat berjalan dengan baik pada perangkat *mobile*, baik dari segi tampilan antarmuka, performa, maupun fungsionalitas sistem secara keseluruhan. Dengan kombinasi penggunaan laptop dan *smartphone*, proses pengembangan dan pengujian aplikasi dapat dilakukan secara optimal sehingga aplikasi yang dihasilkan lebih stabil dan sesuai dengan kebutuhan pengguna.

5.1.2 Perangkat Lunak yang digunakan

Perangkat lunak yang digunakan dalam pengembangan aplikasi adalah sebagai berikut.

Tabel 5. 2 Perangkat Lunak yang digunakan

No	Nama Perangkat Lunak	Spesifikasi
1	*Flutter*	*Flutter* 3.24.5
2	*Dart*	*Dart* SDK version: 3.11.4
3	Android Studio	Runtime version: 17.0.11+0--11852314 amd64
4	*Visual Studio Code*	Version 1.118.1
5	DB Browser for *SQLite*	Version 3.13.1
6	Sistem Operasi	Windows 11 *Home* Single Language

Perangkat lunak yang digunakan dalam pengembangan aplikasi *Habit Tracker* meliputi berbagai *tools* dan teknologi yang mendukung proses pembuatan, pengujian, serta implementasi sistem. Daftar perangkat lunak yang digunakan dapat dilihat pada tabel yang telah disajikan. *Flutter* digunakan sebagai *framework* utama dalam pengembangan aplikasi karena mampu membangun aplikasi berbasis Android dengan satu basis kode (*single codebase*). Bahasa pemrograman *Dart* digunakan untuk menulis logika program serta mengatur tampilan antarmuka aplikasi.

Android *Studio* digunakan sebagai salah satu *Integrated Development Environment* (*IDE*) untuk menjalankan emulator Android dan mendukung proses *build* aplikasi. Selain itu, *Visual Studio Code* digunakan sebagai editor utama dalam proses *coding* karena ringan dan memiliki banyak ekstensi yang mendukung pengembangan *Flutter*. *DB Browser for SQLite* digunakan untuk mengelola dan melihat data yang tersimpan dalam *Database SQLite*, sehingga memudahkan proses pengujian data serta validasi penyimpanan data aplikasi.

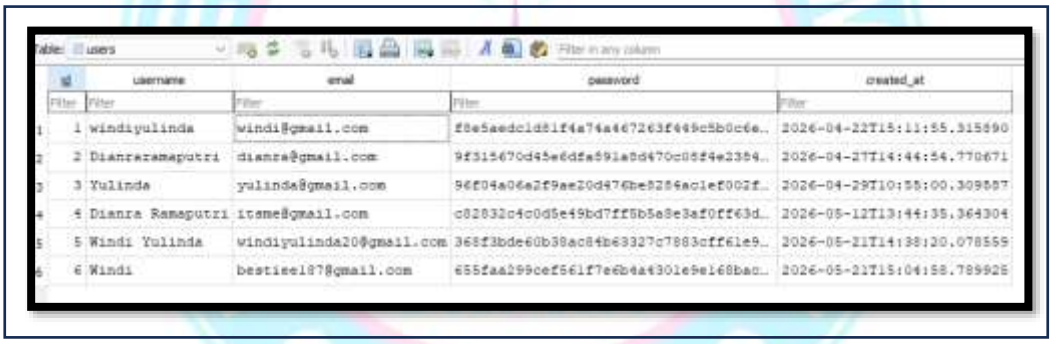
Sistem operasi *Windows* 11 digunakan sebagai platform utama dalam menjalankan seluruh perangkat lunak yang digunakan selama proses pengembangan aplikasi. Dengan penggunaan perangkat lunak tersebut, proses pengembangan aplikasi dapat dilakukan secara efektif, mulai dari tahap penulisan kode, pengujian sistem, hingga implementasi aplikasi pada perangkat *mobile*.

5.2 Implementasi *Database*

Implementasi *Database* dilakukan menggunakan *package sqflite* pada *Flutter*. *Database* dibuat melalui file *Database_helper. Dart* yang berfungsi sebagai penghubung antara aplikasi dan *Database SQLite*.

5.2.1 Struktur *Database*

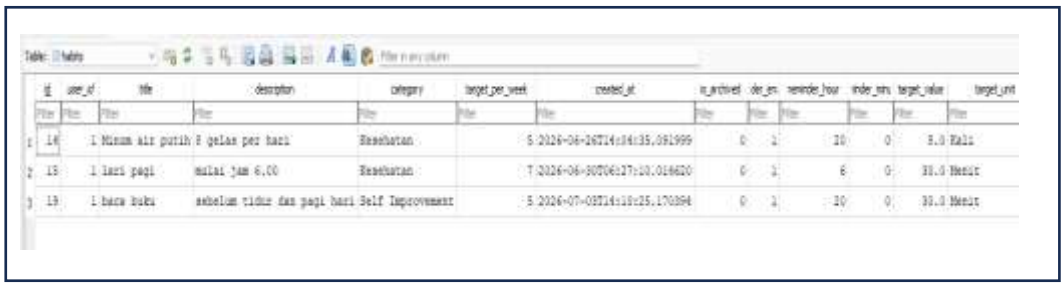
Struktur *Database* pada aplikasi *Habit Tracker* terdiri dari tiga tabel utama, yaitu tabel *user*, *Habits*, *Habit_logs,*dan *ResetPassword.* Berikut gambaran tabel *Database* yang berhasil diimplementasikan.



id	username	email	password	created_at
1	windiyulinda	windi@gmail.com	f8e5aedc1d81f4a74a467263f449c5b0c6e...	2026-04-22T15:11:55.315690
2	Dianreramaputri	dianra@gmail.com	9f31567cd45e6dfa591a8d470c05f4e2354...	2026-04-27T14:44:54.770671
3	Yulinda	yulinda@gmail.com	96f04a06a2f9ae20d476be8284ac1ef002f...	2026-04-29T10:55:00.309887
4	Dianra Ramaputri	itsme@gmail.com	c02832c4c0d5e49bd7ff5b5a8e3af0ff63d...	2026-05-12T13:44:35.364304
5	Windi Yulinda	windiyulinda20@gmail.com	368f3bde60b38ac84b63327c7883cff61e9...	2026-05-21T14:38:20.078559
6	Windi	bestieel87@gmail.com	655faa299cef561f7e6b4a4301e9e168bac...	2026-05-21T15:04:58.789926

Gambar 5. 1 *Tabel Users*

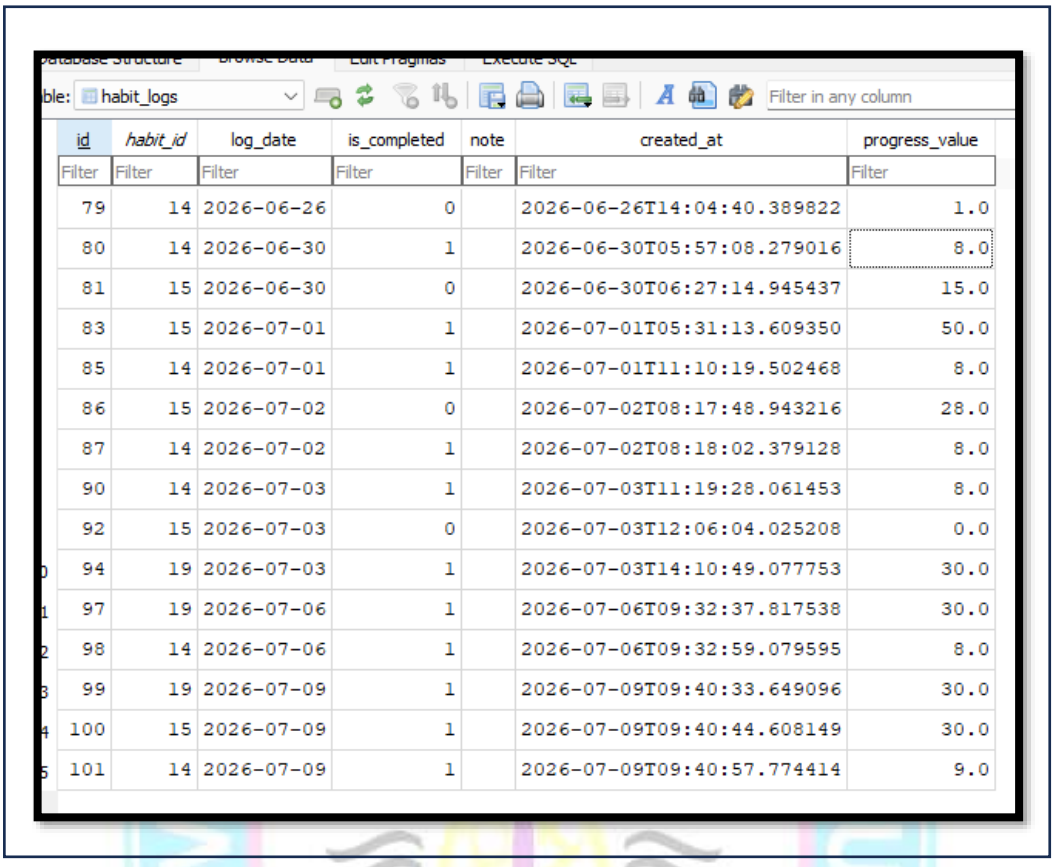
Gambar di atas merupakan gambar tabel *Users* pada aplikasi *Habit Tracker*. tabel tersebut merupakan hasil dari proses *Register* pengguna yang terdiri dari data *Username* , *email*, *Password*, dan tanggal akun dibuat/*Register*.



id	user_id	title	description	category	target_per_week	created_at	is_archived	del_en	reminder_hour	reminder_min	target_value	target_unit
14	1	Minum air putih	8 gelas per hari	Kesehatan	5	2026-06-26T14:34:35.091999	0	1	20	0	8.0	Kali
15	1	lari pagi	mulai jam 6.00	Kesehatan	7	2026-06-30T06:27:10.016620	0	1	6	0	30.0	Menit
19	1	baca buku	sebelum tidur dan pagi hari	Self Improvement	5	2026-07-03T14:18:25.170394	0	1	20	0	30.0	Menit

Gambar 5. 2 Tabel Habits

Gambar di atas menunjukkan implementasi tabel habits pada *database SQLite* yang digunakan dalam aplikasi *Habit Tracker*. Tabel ini berfungsi untuk menyimpan seluruh data *habit* yang dibuat oleh pengguna. Setiap data *habit* memiliki *id* sebagai *primary key* dan *user_id* sebagai *foreign key* yang menghubungkan data *habit* dengan pengguna yang membuatnya. Selain itu, tabel ini menyimpan informasi berupa *title* sebagai nama *habit*, *description* sebagai deskripsi *habit*, *category* sebagai kategori habit, *target_per_week* sebagai target pelaksanaan *habit* dalam satu minggu, *target_value* sebagai nilai target yang harus dicapai, serta target_unit sebagai satuan target, misalnya *Kali* atau *Menit*. Tabel ini juga menyimpan informasi *created_at* sebagai waktu pembuatan *habit*, *is_archived* sebagai status arsip *habit*, serta pengaturan *reminder* yang terdiri atas *reminder_enabled*, *reminder_hour*, dan *reminder_minute*. Data pada tabel *habits* digunakan sebagai dasar dalam proses pengelolaan *habit*, pencatatan *progress* harian, penjadwalan notifikasi, serta menjadi acuan dalam proses analisis tingkat konsistensi menggunakan algoritma *Random Forest* berdasarkan data *progress* yang tersimpan pada tabel *habit_logs*.



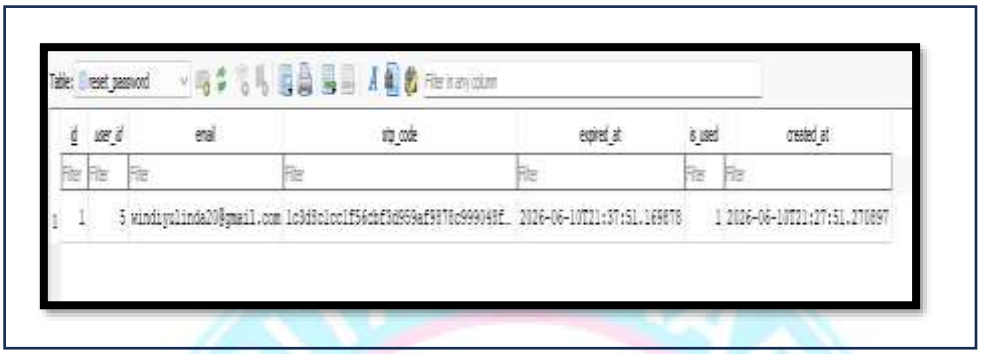
id	habit_id	log_date	is_completed	note	created_at	progress_value
79	14	2026-06-26	0		2026-06-26T14:04:40.389822	1.0
80	14	2026-06-30	1		2026-06-30T05:57:08.279016	8.0
81	15	2026-06-30	0		2026-06-30T06:27:14.945437	15.0
83	15	2026-07-01	1		2026-07-01T05:31:13.609350	50.0
85	14	2026-07-01	1		2026-07-01T11:10:19.502468	8.0
86	15	2026-07-02	0		2026-07-02T08:17:48.943216	28.0
87	14	2026-07-02	1		2026-07-02T08:18:02.379128	8.0
90	14	2026-07-03	1		2026-07-03T11:19:28.061453	8.0
92	15	2026-07-03	0		2026-07-03T12:06:04.025208	0.0
94	19	2026-07-03	1		2026-07-03T14:10:49.077753	30.0
97	19	2026-07-06	1		2026-07-06T09:32:37.817538	30.0
98	14	2026-07-06	1		2026-07-06T09:32:59.079595	8.0
99	19	2026-07-09	1		2026-07-09T09:40:33.649096	30.0
100	15	2026-07-09	1		2026-07-09T09:40:44.608149	30.0
101	14	2026-07-09	1		2026-07-09T09:40:57.774414	9.0

Gambar 5. 3 *Tabel Habit_logs*

Gambar di atas menunjukkan implementasi tabel *habit_logs* pada *database SQLite* yang digunakan dalam aplikasi *Habit Tracker*. Tabel ini berfungsi untuk menyimpan riwayat *progress* harian dari setiap habit yang dimiliki pengguna. Setiap data memiliki id sebagai *primary key* dan *habit_id* sebagai *foreign key* yang menghubungkan data *progress* dengan tabel *habits*. Selain itu, tabel ini menyimpan informasi berupa *log_date* sebagai tanggal pencatatan *progress*, *is_completed* sebagai status penyelesaian *habit*, *progress_value* sebagai nilai *progress* yang diinput oleh pengguna, *note* sebagai catatan tambahan apabila diperlukan, serta *created_at* sebagai waktu penyimpanan data.

Berdasarkan implementasi pada Gambar 5.3, setiap pengguna dapat melakukan pencatatan *progress* sesuai target yang telah ditentukan pada masing-masing *habit*. Nilai *progress* yang tersimpan pada kolom *progress_value* kemudian dibandingkan dengan target_value pada tabel habits untuk menentukan apakah target harian telah tercapai. Selain itu, data *progress* yang tersimpan selama tujuh hari terakhir digunakan sebagai data masukan bagi algoritma *Random Forest* untuk

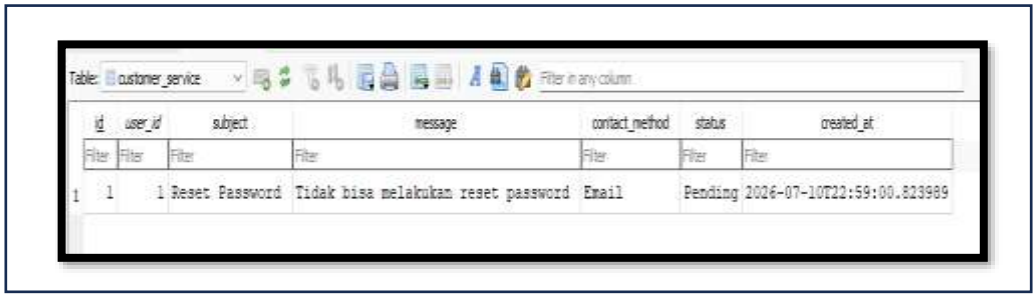
menghasilkan prediksi tingkat konsistensi pengguna dalam kategori Konsisten atau Tidak Konsisten, serta menghasilkan rekomendasi otomatis yang ditampilkan pada halaman rekomendasi *AI*.



	id	user_id	email	otp_code	expired_at	is_used	created_at
1	1	5	windipulinda20@gmail.com	1c3d8c1cc1f56cbf3d959af9878c999049f...	2026-06-10T21:37:51.169878	1	2026-06-10T21:27:51.270897

Gambar 5. 4 Tabel *Reset* Password

Gambar di atas menunjukkan tampilan tabel reset_password pada database aplikasi Habit Tracker. Tabel ini digunakan untuk menyimpan data proses reset kata sandi pengguna. Pada tabel tersebut terdapat beberapa field, yaitu id sebagai primary key, user_id sebagai penghubung dengan data pengguna, email untuk menyimpan alamat email pengguna, otp_code sebagai kode reset password, expired_at sebagai batas waktu berlakunya kode, is_used sebagai penanda apakah kode sudah digunakan atau belum, serta created_at sebagai waktu pembuatan data reset password. Dengan adanya tabel ini, proses lupa kata sandi dapat berjalan lebih terstruktur karena sistem dapat melakukan pengecekan kode reset password sebelum pengguna mengganti kata sandi baru.



	id	user_id	subject	message	contact_method	status	created_at
1	1	1	Reset Password	Tidak bisa melakukan reset password	Email	Pending	2026-07-10T22:59:00.823989

Gambar 5. 5 Tabel Hubungi Customer Service

Gambar 5.5 menunjukkan implementasi tabel customer_service pada *database SQLite* yang digunakan dalam aplikasi *Habit Tracker*. Tabel ini berfungsi

untuk menyimpan data komunikasi antara pengguna dengan layanan Customer Service. Setiap data memiliki id sebagai *primary key* dan user_id sebagai *foreign key* yang menghubungkan data keluhan dengan pengguna yang mengirimkan pesan. Selain itu, tabel ini menyimpan informasi berupa subject sebagai judul keluhan, message sebagai isi pesan yang disampaikan pengguna, contact_method sebagai media komunikasi yang digunakan, status sebagai status penanganan keluhan, serta created_at sebagai waktu pengiriman pesan.

Berdasarkan implementasi pada Gambar 5.xx, pengguna telah berhasil mengirimkan keluhan dengan subjek "Reset Password" melalui media komunikasi Email. Data tersebut tersimpan pada tabel customer_service dengan status awal Pending, yang menunjukkan bahwa keluhan telah berhasil direkam oleh sistem dan sedang menunggu proses penanganan. Implementasi ini memungkinkan setiap riwayat komunikasi antara pengguna dan Customer Service terdokumentasi dengan baik sehingga proses penanganan keluhan dapat dipantau secara lebih terstruktur.

5.3 Tampilan Antarmuka

Tampilan antarmuka (*user interface*) yang telah dirancang pada Bab IV berhasil diimplementasikan ke dalam aplikasi Android menggunakan *Flutter*. Implementasi ini menghasilkan tampilan yang interaktif dan mudah digunakan oleh pengguna. Beberapa halaman utama yang telah berhasil diimplementasikan antara lain sebagai berikut.

1. Halaman *SplashScreen*

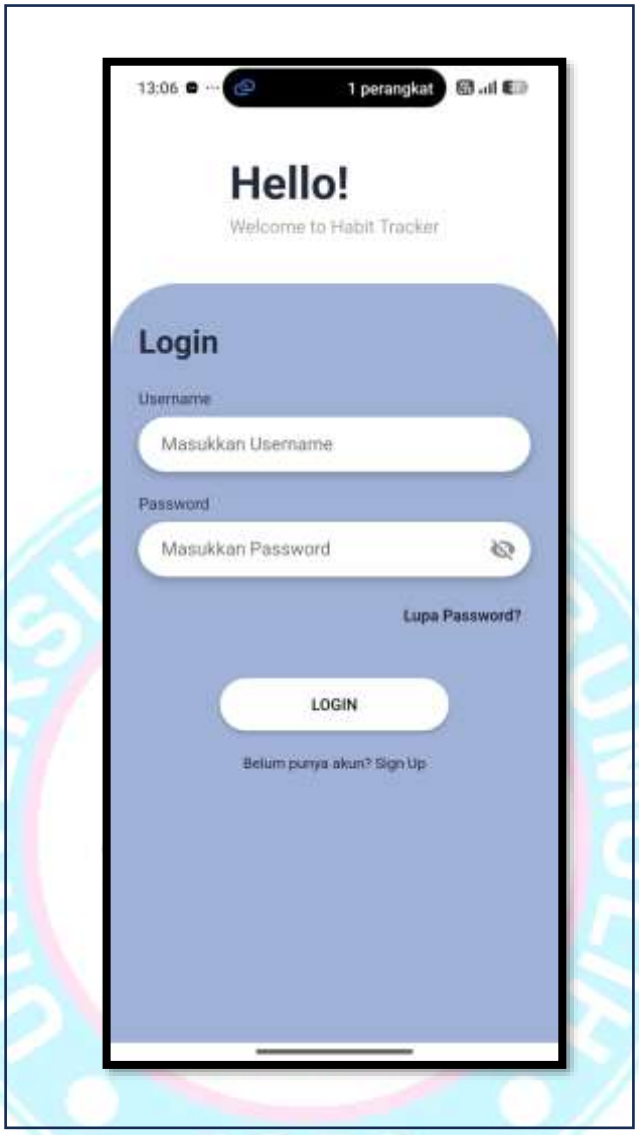
Berikut halaman *SplashScreen* hasil implementasi dari perancangan antarmuka yang penulis buat.

Gambar 5.6 Tampilan halaman *SplashScreen*

Halaman di atas menampilkan halaman *SplashScreen* yang muncul saat aplikasi pertama kali dibuka. Halaman ini berfungsi sebagai tampilan pembuka sekaligus identitas awal aplikasi *Habit Tracker* sebelum pengguna diarahkan ke halaman *login* .

2. Halaman *Login*

Berikut halaman *login* hasil implementasi dari perancangan antarmuka yang penulis buat.

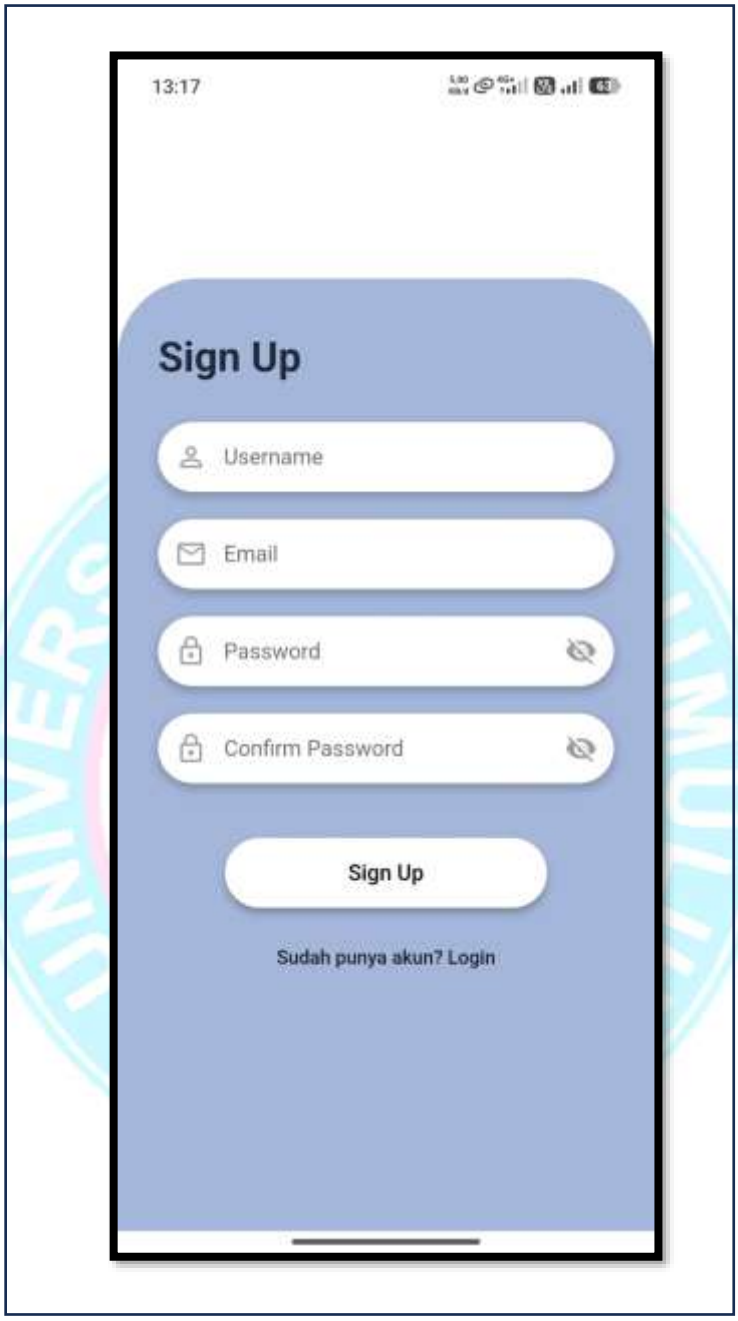


Gambar 5.7 Tampilan halaman *Login*

Gambar di atas menampilkan halaman *login* yang digunakan pengguna untuk masuk ke dalam aplikasi. Pada halaman ini pengguna memasukkan *Username* dan *Password* yang telah terdaftar agar dapat mengakses fitur utama aplikasi.

3. Halaman *Register*

Berikut halaman registrasi hasil implementasi dari perancangan antarmuka yang penulis buat.



Gambar 5.8 Tampilan halaman *Register*

Gambar di atas menampilkan halaman *Register* yang digunakan untuk membuat akun baru. Pengguna diminta mengisi data seperti *Username* , email, *Password*, dan konfirmasi *Password* sebelum dapat menggunakan aplikasi.

4. Halaman Lupa *Password*

Berikut halaman lupa *Password* hasil implementasi dari perancangan antarmuka yang penulis buat.

Gambar 5.9 Halaman Lupa *Password*

Gambar di atas menampilkan halaman lupa *Password* yang digunakan ketika pengguna mengalami kendala saat masuk ke akun. Pada halaman ini pengguna dapat memasukkan email terdaftar untuk menerima kode *Reset Password*.

5. Halaman *Reset Password*

Berikut halaman *Reset Password* hasil implementasi dari perancangan antarmuka yang penulis buat.

Gambar 5.10 Halaman *Reset Password*

Gambar di atas menampilkan halaman *Reset Password* setelah pengguna menerima kode *Reset* melalui *email*. Halaman ini digunakan untuk memasukkan kode reset, *Password* baru, dan konfirmasi *Password* agar akun dapat digunakan kembali.

6. Halaman *Home*

Berikut halaman *Home* hasil implementasi dari perancangan antarmuka yang penulis buat.

Gambar 5.11 Halaman *Home*

Gambar di atas menampilkan halaman *Home* yang berfungsi sebagai pusat informasi utama aplikasi. Pada halaman ini pengguna dapat melihat kalender aktivitas, menu *Atomic Habits* Framework, Fitur Hubungi Customer Service, dan navigasi menuju fitur lain seperti *Habit*, statistik, prediksi dan *logout*.

7. Halaman *Cue*

Berikut halaman *Cue* hasil implementasi dari perancangan antarmuka yang penulis buat.

Gambar 5.12 Halaman *Cue*

Gambar di atas menampilkan halaman penjelasan *Cue* atau isyarat dalam prinsip Atomic *Habit*s. Halaman ini membantu pengguna memahami bahwa *Cue* berperan sebagai pemicu awal untuk memulai kebiasaan.

8. Halaman *Craving*

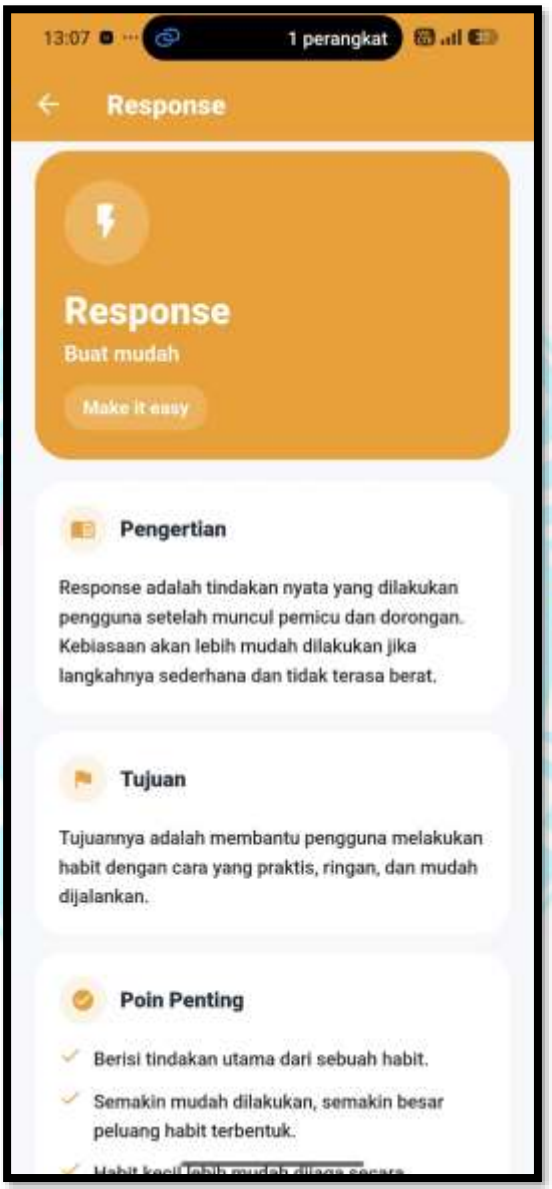
Berikut halaman *Craving* hasil implementasi dari perancangan antarmuka yang penulis buat.

Gambar 5.13 Halaman *Craving*

Gambar di atas menampilkan halaman penjelasan *Craving* atau keinginan. Halaman ini menjelaskan bahwa kebiasaan akan lebih mudah terbentuk apabila pengguna memiliki dorongan atau motivasi untuk melakukannya.

9. Halaman *Response*

Berikut halaman *Response* hasil implementasi dari perancangan antarmuka yang penulis bua



Gambar 5.14 Halaman *Response*

Gambar di atas menampilkan halaman penjelasan *Response* atau tanggapan. Halaman ini menjelaskan bahwa pengguna perlu melakukan tindakan nyata terhadap kebiasaan yang telah direncanakan agar kebiasaan tersebut dapat terbentuk.

10. Halaman *Reward*

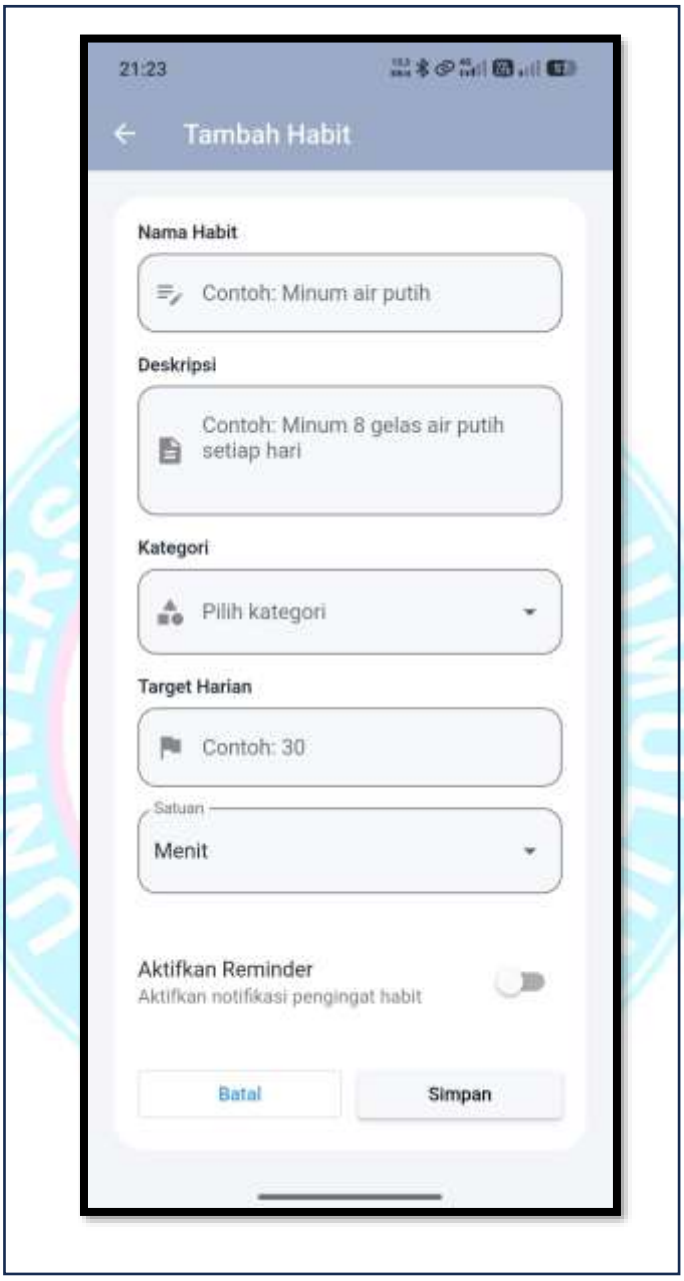
Berikut halaman *reward* hasil implementasi dari perancangan antarmuka yang penulis buat.

Gambar 5.15 Halaman *Reward*

Gambar di atas menampilkan halaman penjelasan *Reward* atau hadiah. Halaman ini menjelaskan bahwa pemberian penghargaan seperti poin dapat meningkatkan motivasi pengguna dalam menyelesaikan kebiasaan.

11. Halaman Tambah *Habit*

Berikut halaman Tambah *Habit* hasil implementasi dari perancangan antarmuka yang penulis buat.



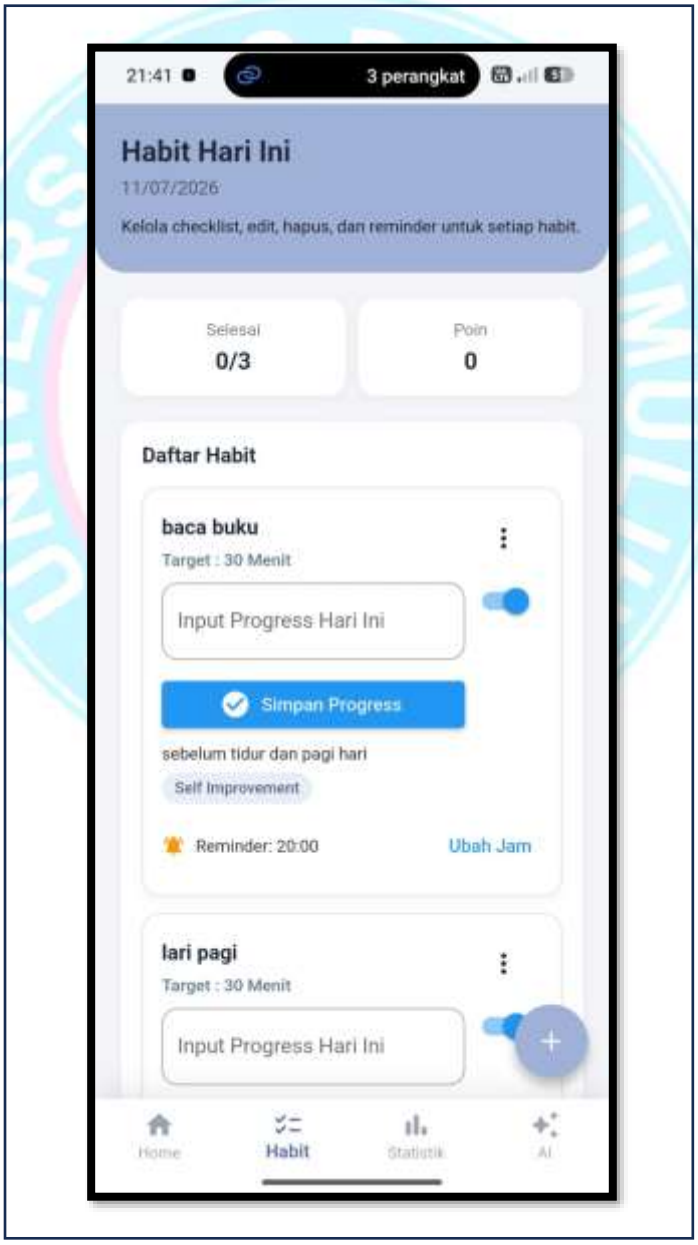
Gambar 5.16 Tampilan halaman Tambah *Habit*

Gambar di atas menampilkan halaman Tambah *Habit* yang digunakan untuk membuat data habit baru pada aplikasi *Habit Tracker*. Pada halaman ini, pengguna dapat mengisi informasi berupa nama *habit*, deskripsi, kategori, target harian, serta memilih satuan target sesuai dengan jenis habit yang dibuat, seperti Menit, Kali, atau satuan lainnya. Selain itu, pengguna dapat mengaktifkan fitur *Reminder* sebagai pengingat untuk melaksanakan habit pada waktu yang telah ditentukan.

Setelah seluruh data diisi dengan benar, pengguna dapat menekan tombol Simpan untuk menyimpan *habit* ke dalam *database*, sedangkan tombol Batal digunakan untuk membatalkan proses penambahan *habit*.

12. Halaman *Input progress Habit*

Berikut halaman *Input progress Habit* hasil implementasi dari perancangan antarmuka yang penulis buat.

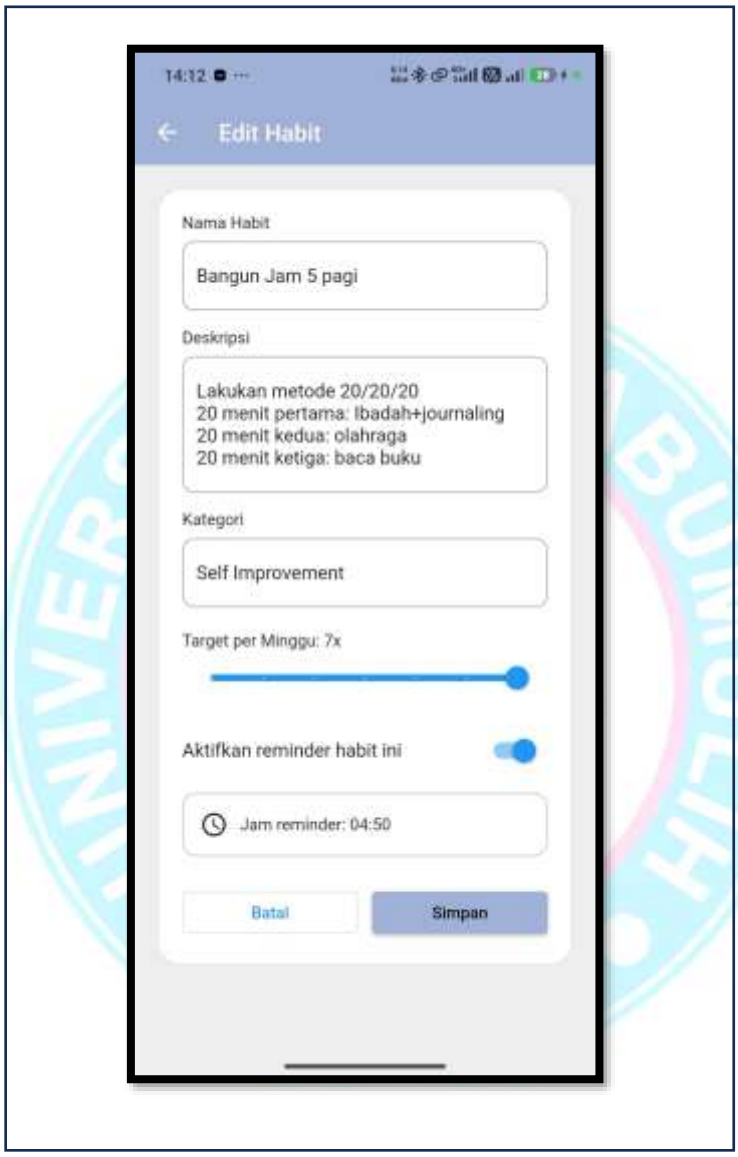


Gambar 5.17 Tampilan halaman Input progress Habit

Gambar di atas menampilkan halaman Halaman Input Progress Habit yang digunakan untuk mengelola seluruh *habit* yang telah dibuat oleh pengguna. Pada bagian atas halaman ditampilkan informasi jumlah habit yang telah diselesaikan dan total poin yang diperoleh pengguna pada hari tersebut. Selanjutnya, sistem menampilkan daftar *habit* yang berisi nama *habit*, target harian beserta satuannya, kolom *Input Progress* Hari Ini untuk mencatat perkembangan aktivitas, serta tombol Simpan *Progress* untuk menyimpan data ke dalam *database*. Setiap *habit* juga menampilkan deskripsi, kategori, serta pengaturan *Reminder* yang dapat diaktifkan atau dinonaktifkan, termasuk fitur untuk mengubah waktu pengingat. Selain itu, pengguna dapat mengelola data habit melalui menu yang tersedia untuk melakukan perubahan atau penghapusan habit sesuai kebutuhan.

13. Halaman Edit *Habit*

Berikut halaman edit *Habit* hasil implementasi dari perancangan antarmuka yang penulis buat.

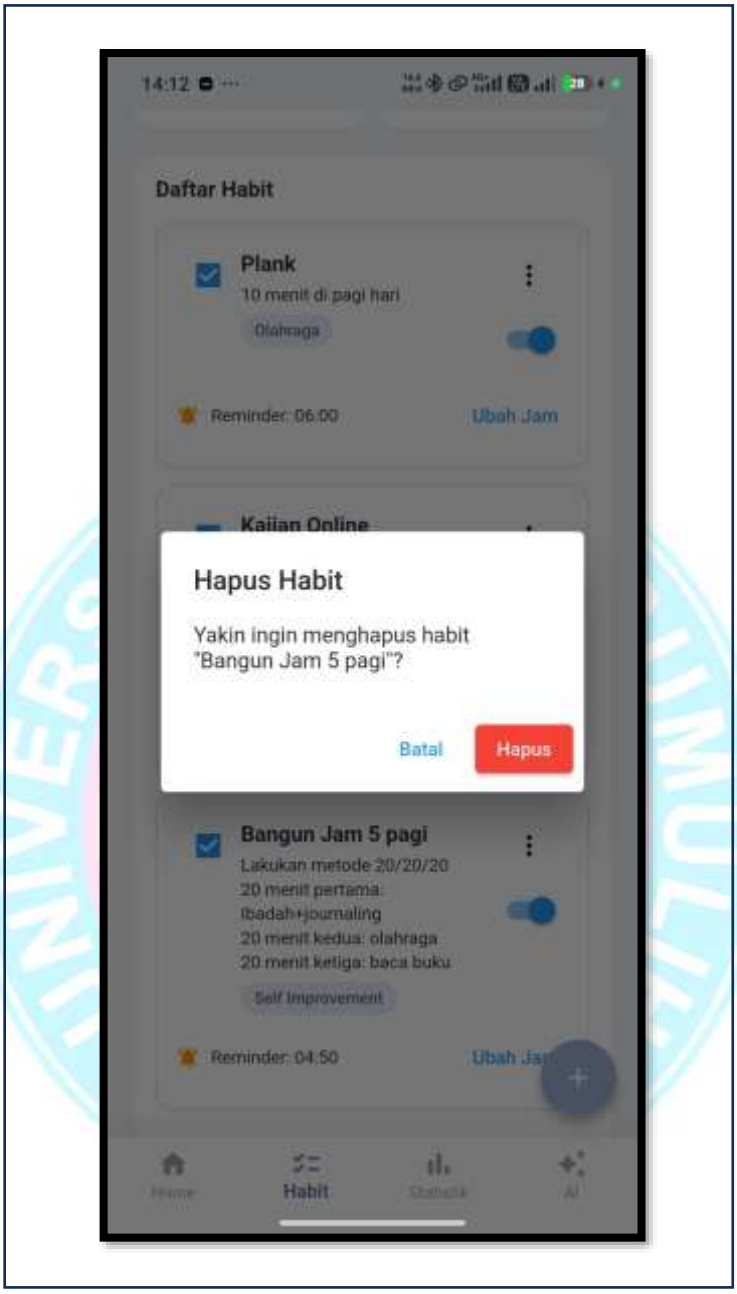


Gambar 5.18 Halaman Edit *Habit*

Gambar di atas menampilkan halaman edit *Habit* yang digunakan untuk memperbarui data kebiasaan. Melalui halaman ini pengguna dapat mengubah nama *Habit*, deskripsi, kategori, target, dan pengaturan *Reminder* .

14. Halaman Hapus *Habit*

Berikut halaman Statistik hasil implementasi dari perancangan antarmuka yang penulis buat.



Gambar 5.19 Halaman Hapus *Habit*

Gambar di atas menampilkan dialog konfirmasi hapus *Habit*. Fitur ini berfungsi untuk memastikan pengguna benar-benar ingin menghapus data kebiasaan agar tidak terjadi penghapusan secara tidak sengaja.

15. Halaman Statistik

Berikut halaman Statistik hasil implementasi dari perancangan antarmuka yang penulis buat.

Gambar 5. 20 Tampilan halaman Statistik Harian

Gambar di atas menampilkan statistik harian dalam bentuk visualisasi grafik. Halaman ini membantu pengguna melihat perkembangan kebiasaan berdasarkan aktivitas harian yang telah dilakukan.

16. Statistik Mingguan

Gambar 5. 21 Tampilan Statistik Mingguan

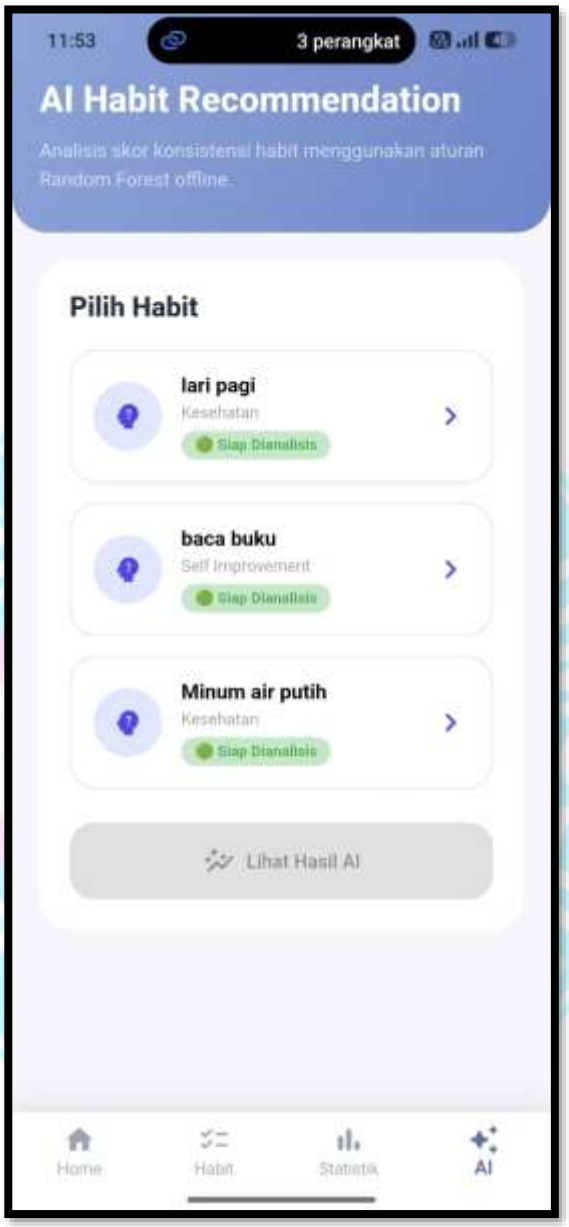
Gambar di atas menampilkan statistik mingguan yang berisi perkembangan *Habit* dalam satu minggu. Tampilan ini memudahkan pengguna mengevaluasi konsistensi kebiasaan dari waktu ke waktu.

17. Halaman Prediksi

Berikut halaman Prediksi hasil implementasi dari perancangan antarmuka yang penulis buat.

Gambar 5. 22 Tampilan Analisis Habit

Gambar di atas menampilkan halaman awal fitur analisis *Habit* menggunakan *AI*. Halaman ini menjadi pintu masuk bagi pengguna untuk menganalisis kebiasaan berdasarkan data aktivitas yang tersimpan.



Gambar 5. 23 Tampilan Habit yang akan dianalisis

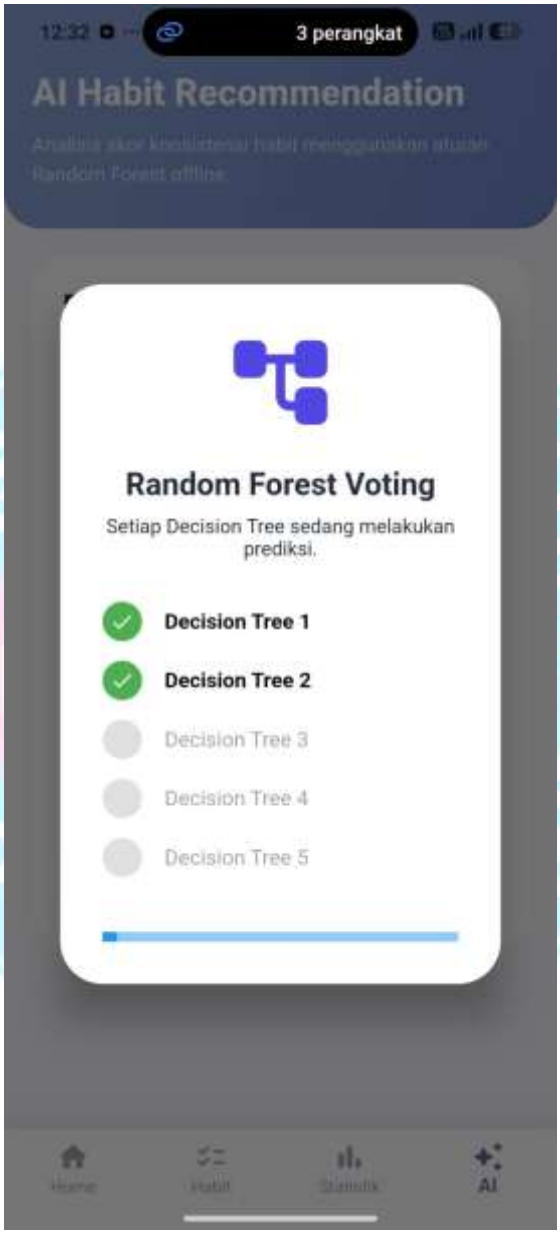
Gambar di atas menampilkan halaman Pemilihan *Habit* pada fitur *AI Habit Recommendation*. Halaman ini digunakan untuk menampilkan daftar *habit* yang telah memenuhi syarat untuk dianalisis. Pengguna dapat memilih salah satu habit yang berstatus Siap Dianalisis, kemudian sistem akan menjalankan proses analisis

menggunakan algoritma *Random Forest* untuk menghasilkan prediksi tingkat konsistensi beserta rekomendasinya.

Gambar 5.24 Proses Analisis *AI*

Gambar di atas menampilkan halaman Proses Analisis *AI* yang muncul setelah pengguna memilih salah satu *habit* untuk dianalisis. Pada halaman ini, sistem menampilkan tahapan analisis secara bertahap, mulai dari memuat data habit, menganalisis target, menghitung *progress*, menghitung tingkat pencapaian (*achievement*), hingga menjalankan algoritma *Random Forest*. Halaman ini

memberikan informasi kepada pengguna bahwa proses analisis sedang berlangsung sebelum hasil prediksi dan rekomendasi ditampilkan.



Gambar 5. 25 Proses *Random Forest Voting*

Gambar di atas menampilkan halaman *Random Forest Voting* yang menunjukkan proses pengambilan keputusan oleh algoritma *Random Forest*. Pada tahap ini, setiap *Decision Tree* melakukan prediksi secara independen berdasarkan data progress *habit* pengguna selama tujuh hari terakhir. Hasil prediksi dari seluruh *Decision Tree* kemudian digabungkan melalui mekanisme *voting* untuk

menghasilkan prediksi akhir mengenai tingkat konsistensi pengguna. Proses ini berlangsung secara otomatis sebelum sistem menampilkan hasil prediksi dan rekomendasi pada halaman berikutnya.

Gambar 5.26 Hasil Prediksi *Random Forest*

Gambar di atas menampilkan halaman Hasil Prediksi *AI* setelah proses analisis menggunakan algoritma *Random Forest* selesai dilakukan. Pada halaman ini, sistem menampilkan informasi mengenai *habit* yang dipilih beserta hasil

prediksi tingkat konsistensinya, yaitu Konsisten atau Tidak Konsisten. Selain itu, sistem juga memberikan rekomendasi otomatis berdasarkan hasil analisis data progress pengguna selama tujuh hari terakhir. Rekomendasi tersebut bertujuan membantu pengguna memperbaiki kebiasaan yang belum konsisten, misalnya dengan menyesuaikan target *habit* atau memanfaatkan fitur *Reminder* agar pelaksanaan *habit* menjadi lebih teratur. Pengguna juga dapat menekan tombol Analisis *Habit* Lain untuk kembali ke halaman pemilihan habit dan melakukan analisis pada *habit* yang berbeda.

5.4 Hasil Pengujian Sistem

Pengujian sistem dilakukan untuk memastikan bahwa aplikasi berjalan sesuai dengan fungsi yang telah dirancang. Metode pengujian yang digunakan adalah *Statement* , yaitu pengujian yang berfokus pada logika internal program.

5.4.1 Pengujian Modul *Login*

A. Potongan *Source Code*

```
Future<void> _login () async {
  FocusScope.of(context).unfocus();

  final Username  = Username Controller.text.trim();
  final Password = PasswordController.text.trim();

  if (Username .isEmpty) {
    return _showMessage('Username  wajib diisi');
  }

  if (Password.isEmpty) {
    return _showMessage('Password wajib diisi');
  }

  setState(() => _isLoading = true);

  try {
    final user = await DatabaseHelper.instance.login (Username , Password);

    if (user == null) {
      return _showMessage('Username  atau Password salah');
    }
```

```
    if (user.id == null) {
      return _showMessage('Data user tidak valid');
    }

    await SessionHelper.saveUsersession(user.id!);

    final prefs = await SharedPreferences.getInstance();
    await prefs.setString('Username ', user.Username );

    if (!mounted) return;

    ScaffoldMessenger.of(context).showSnackBar(
      SnackBar(
        content: Text('Selamat datang, ${user.Username }'),
      ),
    );

    Navigator.pushReplacement(
      context,
      MaterialPageRoute(builder: (_) => HomePage()),
    );
  } catch (e) {
    _showMessage('Login  gagal: $e');
  } finally {
    if (mounted) {
      setState(() => _isLoading = false);
    }
  }
}
```

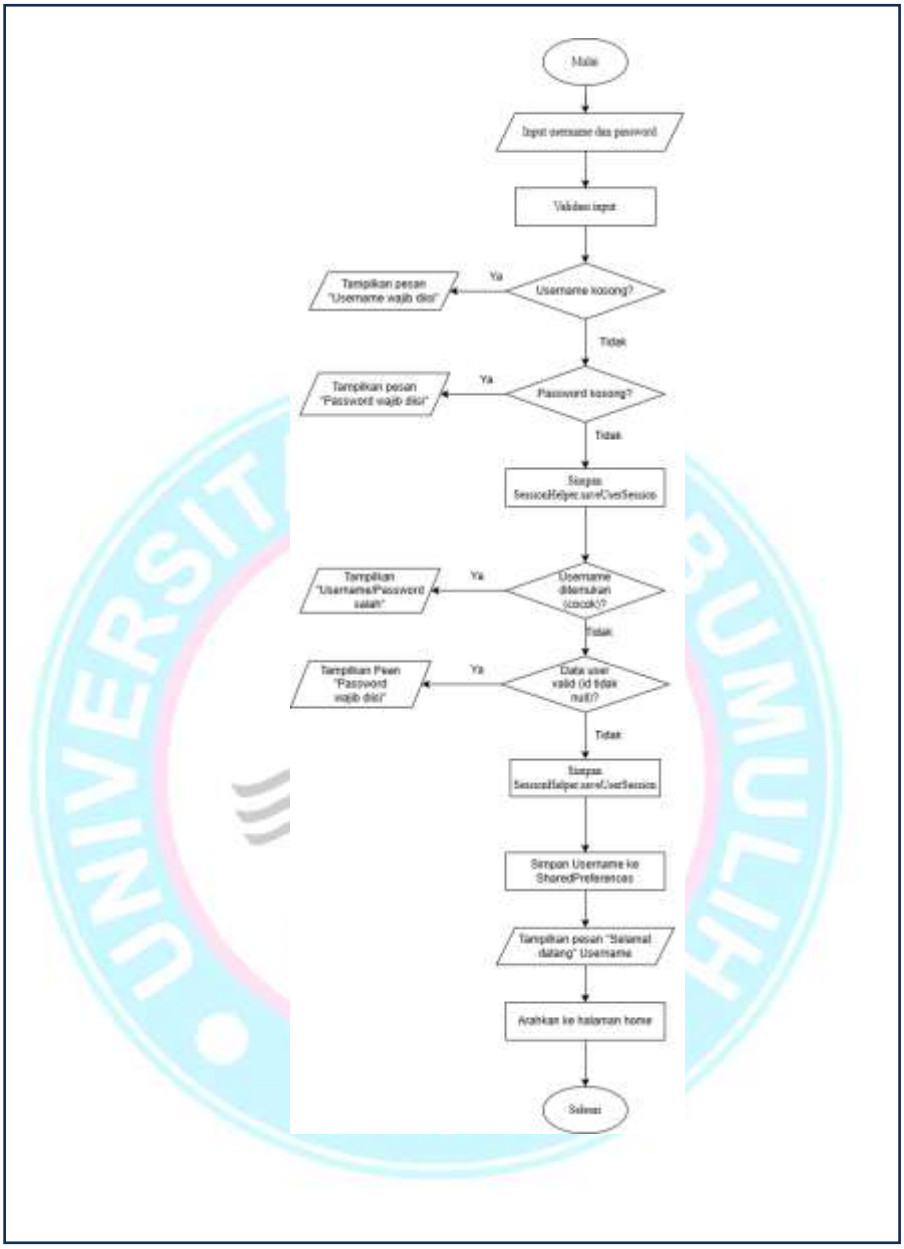
B. Identifikasi *Statement*

Tabel 5. 3 Identifikasi *Statement* Fungsi *Login*

Kode	*Statement*	Keterangan
S1	final *Username* = *Username* Controller.text.trim();	Mengambil input *Username*
S2	final *Password* = *Password*Controller.text.trim();	Mengambil input *Password*
S3	if (*Username* .isEmpty)	Mengecek apakah *Username* kosong
S4	_showMessage('*Username* wajib diisi')	Menampilkan pesan *Username* wajib diisi
S5	if (*Password*.isEmpty)	Mengecek apakah *Password* kosong

S6	_showMessage('*Password* wajib diisi')	Menampilkan pesan *Password* wajib diisi
S7	*Database*Helper.instance.*login* (*Username* , *Password*)	Mengecek data *login* ke *Database*
S8	if (user == null)	Mengecek apakah user tidak ditemukan
S9	showMessage('*Username* atau *Password* salah')	Menampilkan pesan *login* salah
S10	if (user.id == null)	Mengecek apakah id user valid
S11	_showMessage('Data user tidak valid')	Menampilkan pesan data user tidak valid
S12	SessionHelper.save*Users*session(user.id!)	Menyimpan session *login*
S13	prefs.setString('*Username* ', user.*Username*)	Menyimpan *Username* ke SharedPreferences
S14	showSnackBar('Selamat datang')	Menampilkan pesan *login* berhasil
S15	Navigator.pushReplacement(...)	Mengarahkan pengguna ke halaman *Home*

C. *Flowchart*



Gambar 5. 27 *Flowchart* fungsi *Login*

Gambar di atas menjelaskan *Flowchart* dari fungsi *login* . *Flowchart* tersebut menggambarkan alur proses ketika pengguna memasukkan *Username* dan *Password* sampai sistem menentukan apakah pengguna berhasil masuk ke halaman utama atau gagal *login* . Adapun penjelasannya sebagai berikut:

1. S1 dan S2 (Input dan validasi awal)

   Pada tahap ini, pengguna memasukkan *Username* dan *Password*. Setelah itu sistem mengambil data yang dimasukkan oleh pengguna dan melakukan validasi

awal, seperti menghapus spasi kosong di awal atau akhir input menggunakan proses trim().

2. S3 (Mengecek *Username* kosong)

   Pada tahap ini, sistem mengecek apakah *Username* yang dimasukkan pengguna masih kosong. Jika Ya, maka alur akan menuju ke S4. Jika Tidak, maka proses dilanjutkan ke S5.

3. S4 (Menampilkan pesan *Username* wajib diisi)

   Jika *Username* kosong, sistem akan menampilkan pesan "*Username* wajib diisi". Setelah pesan ditampilkan, proses *login* tidak dilanjutkan dan langsung selesai.

4. S5 (Mengecek *Password* kosong)

   Setelah *Username* terisi, sistem akan mengecek apakah *Password* masih kosong. Jika Ya, maka alur menuju ke S6. Jika Tidak, maka proses dilanjutkan ke S7.

5. S6 (Menampilkan pesan *Password* wajib diisi)

   Jika *Password* kosong, sistem akan menampilkan pesan "*Password* wajib diisi". Setelah itu proses *login* dihentikan dan selesai.

6. S7 (Mengecek data *login* ke *Database*)

   Jika *Username* dan *Password* sudah terisi, sistem akan melakukan pengecekan data ke *Database*. Pada tahap ini, *Username* dan *Password* yang dimasukkan pengguna dicocokkan dengan data akun yang tersimpan.

7. S8 (Mengecek apakah user tidak ditemukan)

   Pada tahap ini, sistem mengecek apakah data user ditemukan atau tidak. Jika Ya, artinya user tidak ditemukan atau *Username* dan *Password* salah, maka alur menuju ke S9. Jika Tidak, artinya user ditemukan, maka proses dilanjutkan ke S10.

8. S9 (Menampilkan pesan *Username* atau *Password* salah)

   Jika data user tidak ditemukan di *Database*, sistem akan menampilkan pesan "*Username* atau *Password* salah". Setelah pesan ditampilkan, proses *login* selesai.

9. S10 (Mengecek data user valid)

Setelah user ditemukan, sistem mengecek apakah data user valid, terutama bagian id user. Jika Ya, artinya data user tidak valid atau id user bernilai kosong, maka alur menuju ke S11. Jika Tidak, artinya data user valid, maka proses dilanjutkan ke S12.

10. S11 (Menampilkan pesan data user tidak valid)

    Jika id user tidak valid atau bernilai kosong, sistem akan menampilkan pesan "Data user tidak valid". Setelah itu proses *login* dihentikan dan selesai.

11. S12 (Menyimpan session *login*)

    Jika data user valid, sistem akan menyimpan session *login* pengguna. Session ini digunakan agar aplikasi mengetahui bahwa pengguna sudah berhasil *login* .

12. S13 (Menyimpan *Username* ke SharedPreferences)

    Setelah session *login* tersimpan, sistem menyimpan *Username* pengguna ke penyimpanan lokal menggunakan SharedPreferences. Data ini dapat digunakan kembali oleh aplikasi, misalnya untuk menampilkan nama pengguna.

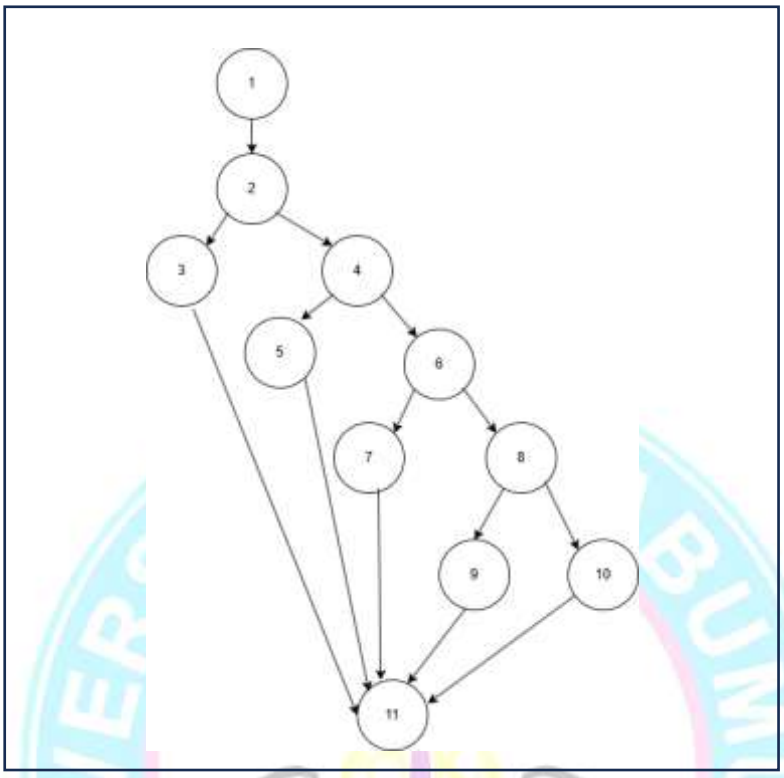
13. S14 (Menampilkan pesan selamat datang)

    Setelah data berhasil disimpan, sistem menampilkan pesan "Selamat datang" kepada pengguna sebagai tanda bahwa proses *login* berhasil.

14. S15 (Mengarahkan pengguna ke halaman *Home*)

    Setelah *login* berhasil, sistem mengarahkan pengguna ke halaman utama aplikasi atau *Home*Page menggunakan proses navigasi.

D. *Flowgraph*



Gambar 5. 28 *Flowgraph* fungsi *Login*

Gambar di atas menjelaskan *Flowgraph* dari fungsi *login* . *Flowgraph* tersebut mempresentasikan alur eksekusi program berdasarkan hasil identifikasi *Statement* yang telah dilakukan sebelumnya. *Flowgraph* yang dibentuk terdiri dari 11 node dan 14 edge. Adapun penjelasan dari setiap node adalah sebagai berikut.

1. Node 1 terdiri dari S1 dan S2

   Node ini merupakan proses awal pada fungsi *login* , yaitu pengguna memasukkan *Username* dan *Password*, kemudian sistem mengambil input tersebut untuk diproses. Pada tahap ini data input disiapkan sebelum masuk ke proses pengecekan.

2. Node 2 terdiri dari S3

   Node ini merupakan percabangan untuk mengecek apakah *Username* kosong. Jika *Username* kosong, maka alur akan menuju ke node 3. Jika *Username* tidak kosong, maka alur akan dilanjutkan ke node 4.

3. Node 3 terdiri dari S4

Node ini berisi proses menampilkan pesan "*Username* wajib diisi". Node ini dijalankan apabila pengguna tidak mengisi *Username*. Setelah pesan ditampilkan, proses langsung menuju node 11 atau selesai.

4. Node 4 terdiri dari S5

   Node ini merupakan percabangan untuk mengecek apakah *Password* kosong. Jika *Password* kosong, maka alur menuju node 5. Jika *Password* tidak kosong, maka alur dilanjutkan ke node 6.

5. Node 5 terdiri dari S6

   Node ini berisi proses menampilkan pesan "*Password* wajib diisi". Node ini dijalankan apabila pengguna belum mengisi *Password*. Setelah pesan ditampilkan, proses menuju node 11 atau selesai.

6. Node 6 terdiri dari S7 dan S8

   Node ini merupakan proses pengecekan data *login* ke *Database*. Sistem mencocokkan *Username* dan *Password* yang dimasukkan pengguna dengan data akun yang tersimpan. Pada node ini juga dilakukan pengecekan apakah user ditemukan atau tidak.

7. Node 7 terdiri dari S9

   Node ini berisi proses menampilkan pesan "*Username* atau *Password* salah". Node ini dijalankan apabila data user tidak ditemukan di *Database* atau *Username* dan *Password* tidak sesuai. Setelah pesan ditampilkan, proses menuju node 11 atau selesai.

8. Node 8 terdiri dari S10

   Node ini merupakan percabangan untuk mengecek apakah data user valid, yaitu memastikan id user tidak bernilai kosong atau null. Jika data user tidak valid, maka alur menuju node 9. Jika data user valid, maka alur menuju node 10.

9. Node 9 terdiri dari S11

   Node ini berisi proses menampilkan pesan "Data user tidak valid". Node ini dijalankan apabila user ditemukan, tetapi id user tidak valid. Setelah pesan ditampilkan, proses menuju node 11 atau selesai.

10. Node 10 terdiri dari S12, S13, S14, dan S15

    Node ini merupakan proses ketika *login* berhasil. Sistem menyimpan session *login*, menyimpan *Username* ke SharedPreferences, menampilkan pesan "Selamat datang", dan mengarahkan pengguna ke halaman utama atau *Home*.

11. Node 11 merupakan akhir proses

    Node ini menunjukkan bahwa proses *login* telah selesai. Semua jalur, baik yang gagal maupun berhasil, akan berakhir pada node ini.

E. *Cyclomatic Complexity*

Pada modul *login* terdapat 4 predicate node utama, yaitu:

Tabel 5. 4 *Cyclomatic Complexity* Fungsi *Login*

No	Predicate Node	Kondisi
1	if (*Username* .isEmpty)	Mengecek apakah *Username* kosong
2	if (*Password*.isEmpty)	Mengecek apakah *Password* kosong
3	if (user == null)	Mengecek apakah data user tidak ditemukan
4	if (user.id == null)	Mengecek apakah id user tidak valid

Maka perhitungan *Cyclomatic Complexity* adalah:

V(G) = P + 1

V(G) = 4 + 1

V(G) = 5

Berdasarkan hasil perhitungan tersebut, nilai *Cyclomatic Complexity* pada modul *login* adalah 5. Artinya, terdapat 5 jalur independen yang harus diuji untuk memastikan seluruh alur logika utama pada modul *login* berjalan dengan baik.

F. *Independent Path*

Berdasarkan hasil perhitungan *Cyclomatic Complexity*, diperoleh 5 *Independent Path* sebagai berikut.

1. Path 1:

   1 → 2→ 3→ 11

   (*Username* kosong)

2. Path 2

   1→ 2→ 4 → 5→ 11

(*Username* terisi, tetapi *Password* kosong)

3. Path 3

   1→ 2 → 4→ 6 → 7 → 11

   (*Username* dan *Password* terisi, tetapi user tidak ditemukan)

4. Path 4

   1→ 2→ 4→ 6→ 8→ 9→ 11

   (User ditemukan, tetapi data user tidak valid)

5. Path 5

   1→ 2→ 4→ 6→ 8→ 10→ 11

   (*Username* dan *Password* benar, *login* berhasil)

G. *Test case*

Berikut hasil *Test case* pada fungsi *login* berdasarkan hasil dari *Independent Path*.

Tabel 5. 5 *Test case* Fungsi *Login*

Test case	***Username***	***Password***	**Output yang Diharapkan**	**Hasil**
TC1	(Kosong)	Windi123	*Username* wajib diisi	Valid
TC2	windiyulinda	(Kosong)	*Password* wajib diisi	Valid
TC3	windi	123	*Username* atau *Password* salah	Valid
TC4	windiyulinda	Windi123	Data user tidak valid	Valid
TC5	windiyulinda	Windi123	Selamat datang, windiyulinda dan masuk ke halaman *Home*	Valid

H. *Statement Coverage*

Statement Coverage digunakan untuk mengetahui apakah seluruh *Statement* pada kode sudah dijalankan oleh *test case*.

Rumus:

Statement Coverage = (Jumlah *Statement* yang dieksekusi / Jumlah seluruh *Statement*) × 100% (sumber)

Tabel 5. 6 *Statement Coverege* Fungsi *Login*

Test case	*Statement* yang Dieksekusi	Keterangan
TC1	S1, S2, S3, S4	Menguji kondisi *Username* kosong
TC2	S1, S2, S3, S5, S6	Menguji kondisi *Password* kosong
TC3	S1, S2, S3, S5, S7, S8, S9	Menguji kondisi user tidak ditemukan
TC4	S1, S2, S3, S5, S7, S8, S10, S11	Menguji kondisi data user tidak valid
TC5	S1, S2, S3, S5, S7, S8, S10, S12, S13, S14, S15	Menguji kondisi *login* berhasil

Berdasarkan tabel di atas, seluruh *Statement* dari S1 sampai S15 telah dieksekusi melalui *test case* yang dibuat. Dengan demikian, jumlah *Statement* yang dieksekusi adalah 15 dari total 15 *Statement*.

Maka perhitungan *Statement Coverage* adalah:

Statement Coverage = (Jumlah *Statement* yang dieksekusi / Jumlah seluruh *Statement*) × 100%

Statement Coverage = (15 / 15) × 100%

Statement Coverage = 100%

Berdasarkan hasil perhitungan tersebut, nilai *Statement Coverage* pada modul *login* adalah 100%. Artinya, seluruh *Statement* utama pada modul *login* telah diuji melalui *test case* yang dibuat.

5.4.2 Pengujian Modul *Register*

A. Potongan *Source Code*

```
Future<void> _Register() async {
  FocusScope.of(context).unfocus();

  final Username  = Username Controller.text.trim();
  final email = emailController.text.trim();
  final Password = PasswordController.text.trim();
  final confirmPassword = confirmController.text.trim();

  if (Username .isEmpty) {
    _showMessage('Username  wajib diisi');
   return;
  }
```

```
if (Username .length < 3) {
  _showMessage('Username  minimal 3 karakter');
  return;
}

if (email.isEmpty) {
  _showMessage('EmaiIl wajib diisi');
  return;
}

final emailRegex = RegExp(r'^[^@\s]+@[^@\s]+\.[^@\s]+$');
if (!emailRegex.hasMatch(email)) {
  _showMessage('Format email tidak valid');
  return;
}

if (Password.isEmpty) {
  _showMessage('Password wajib diisi');
  return;
}

if (Password.length < 6) {
  _showMessage('Password minimal 6 karakter');
  return;
}

if (confirmPassword.isEmpty) {
  _showMessage('Konfirmasi Password wajib diisi');
  return;
}

if (Password != confirmPassword) {
  _showMessage('Konfirmasi Password tidak cocok');
  return;
}

final newUser = UserModel(
  Username : Username ,
  email: email,
  Password: Password,
  createdAt: DateTime.now().toIso8601String(),
```

```
    );

    final errorMessage = await DatabaseHelper.instance.RegisterUser(newUser);

    if (errorMessage != null) {
      _showMessage(errorMessage);
      return;
    }

    ScaffoldMessenger.of(context).showSnackBar(
      const SnackBar(
        content: Text('Registrasi berhasil, silakan login '),
      ),
    );

    Navigator.pushReplacement(
      context,
      MaterialPageRoute(builder: (_) => const Login Page()),
    );
  }
```

B. Identifikasi *Statement*

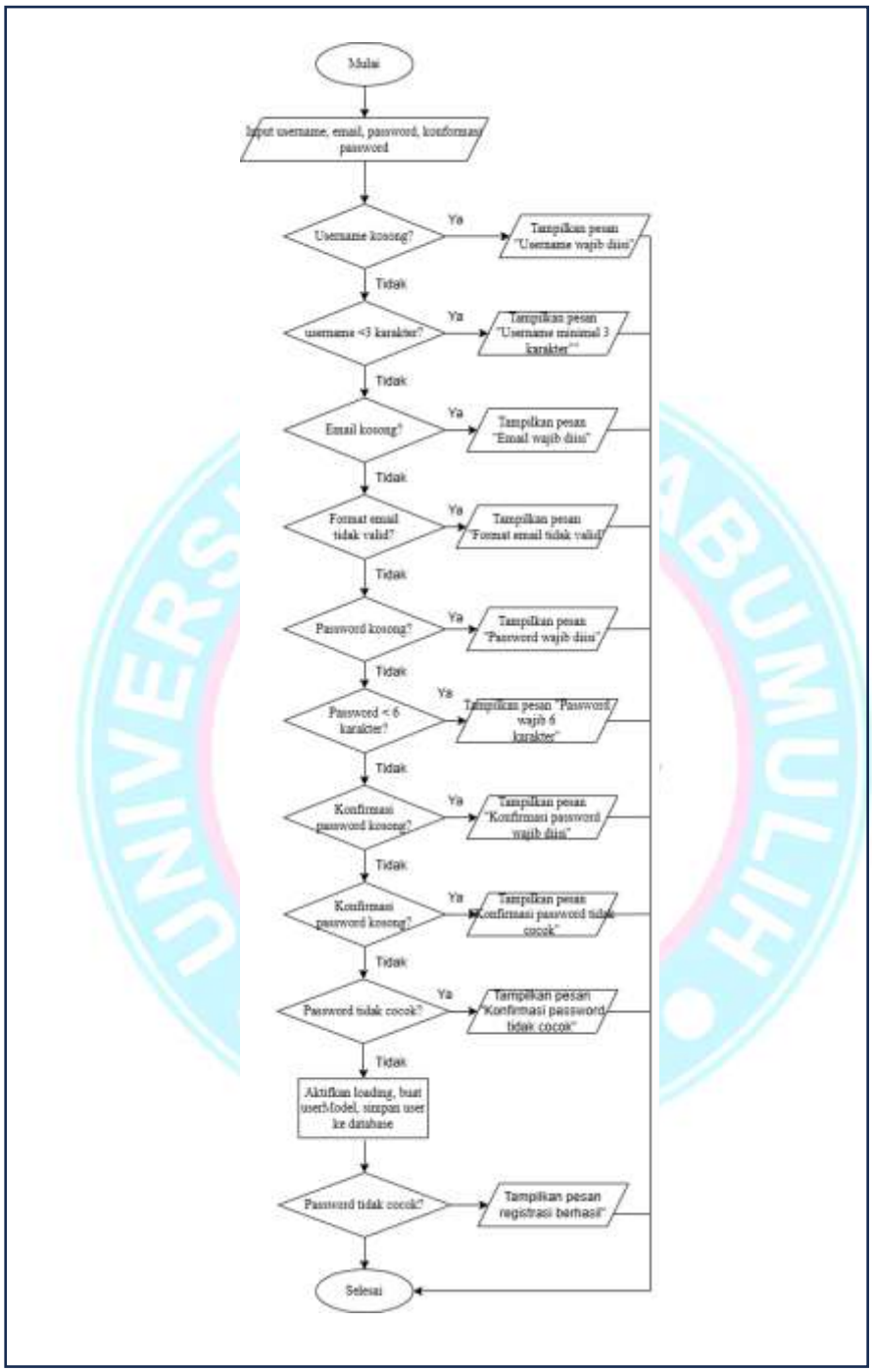
Tabel 5. 7 Identifikasi *Statement* Fungsi *Register*

Kode	*Statement*	Keterangan
S1	final *Username* = *Username* Controller.text.trim();final email = emailController.text.trim();final *Password* = *Password*Controller.text.trim();final confirm*Password* = confirmController.text.trim();	Mengambil input *Username* , email, *Password*, dan konfirmasi *Password*
S2	if (*Username* .isEmpty)	Mengecek apakah *Username* kosong
S3	_showMessage('*Username* wajib diisi')	Menampilkan pesan *Username* wajib diisi
S4	if (*Username* .length < 3)	Mengecek apakah *Username*

		kurang dari 3 karakter
S5	_showMessage('*Username* minimal 3 karakter')	Menampilkan pesan *Username* minimal 3 karakter
S6	if (email.isEmpty)	Mengecek apakah email kosong
S7	_showMessage('Email wajib diisi')	Menampilkan pesan email wajib diisi
S8	final emailRegex = RegExp(r'^[^@\s]+@[^@\s]+\.[^@\s]+$');if (!emailRegex.hasMatch(email))	Mengecek apakah format email tidak valid
S9	_showMessage('Format email tidak valid')	Menampilkan pesan format email tidak valid
S10	if (*Password*.isEmpty)	Mengecek apakah *Password* kosong
S11	_showMessage('*Password* wajib diisi')	Menampilkan pesan *Password* wajib diisi
S12	if (*Password*.length < 6)	Mengecek apakah *Password* kurang dari 6 karakter
S13	_showMessage('*Password* minimal 6 karakter')	Menampilkan pesan *Password* minimal 6 karakter
S14	if (confirm*Password*.isEmpty)	Mengecek apakah konfirmasi

		Password kosong
S15	_showMessage('Konfirmasi *Password* wajib diisi')	Menampilkan pesan konfirmasi *Password* wajib diisi
S16	if (*Password* != confirm*Password*)	Mengecek apakah *Password* dan konfirmasi *Password* tidak cocok
S17	_showMessage('Konfirmasi *Password* tidak cocok')	Menampilkan pesan konfirmasi *Password* tidak cocok
S18	setState(() { _isLoading = true; });final newUser = UserModel(...);final errorMessage = await *Database*Helper.instance.*Register*User(newUser);	Mengaktifkan loading, membuat data user baru, dan menyimpan user ke *Database*
S19	if (errorMessage != null)	Mengecek apakah terdapat error dari *Database*
S20	_showMessage(errorMessage)	Menampilkan pesan error dari *Database*
S21	ScaffoldMessenger.of(context).showSnackBar(...)Navigator.pushReplacement(...)	Menampilkan pesan *Register* berhasil dan mengarahkan pengguna ke halaman *login*

C. *Flowchart*



Gambar 5. 29 *Flowchart* fungsi *Register*

Gambar di atas menjelaskan *Flowchart* dari fungsi *Register*. *Flowchart* tersebut menggambarkan alur proses ketika pengguna mengisi data pendaftaran berupa *Username* , email, *Password*, dan konfirmasi *Password*, kemudian sistem melakukan serangkaian validasi hingga menentukan apakah registrasi berhasil atau gagal. Adapun penjelasannya sebagai berikut.

1. S1 (Input data registrasi)

   Pada tahap ini, pengguna memasukkan data registrasi berupa *Username* , email, *Password*, dan konfirmasi *Password*. Setelah itu sistem mengambil seluruh data input tersebut dan melakukan proses awal menggunakan trim() agar spasi kosong di awal atau akhir input tidak ikut terbaca.
2. S2 (Mengecek *Username* kosong)

   Pada tahap ini, sistem mengecek apakah *Username* yang dimasukkan pengguna masih kosong. Jika Ya, maka alur akan menuju ke S3. Jika Tidak, maka proses dilanjutkan ke S4.
3. S3 (Menampilkan pesan *Username* wajib diisi)

   Jika *Username* kosong, sistem akan menampilkan pesan "*Username* wajib diisi". Setelah pesan ditampilkan, proses registrasi tidak dilanjutkan dan langsung selesai.
4. S4 (Mengecek *Username* kurang dari 3 karakter)

   Setelah *Username* terisi, sistem akan mengecek apakah panjang *Username* kurang dari 3 karakter. Jika Ya, maka alur menuju ke S5. Jika Tidak, maka proses dilanjutkan ke S6.
5. S5 (Menampilkan pesan *Username* minimal 3 karakter)

   Jika *Username* yang dimasukkan terlalu pendek, sistem akan menampilkan pesan "*Username* minimal 3 karakter". Setelah itu proses registrasi dihentikan dan selesai.
6. S6 (Mengecek email kosong)

   Pada tahap ini, sistem mengecek apakah email yang dimasukkan pengguna masih kosong. Jika Ya, maka alur menuju ke S7. Jika Tidak, maka proses dilanjutkan ke S8.
7. S7 (Menampilkan pesan email wajib diisi)

   Jika email kosong, sistem akan menampilkan pesan "Email wajib diisi". Setelah pesan ditampilkan, proses registrasi selesai.
8. S8 (Mengecek format email tidak valid)

   Setelah email terisi, sistem akan mengecek apakah format email sesuai dengan pola email yang benar menggunakan regular expression (RegExp). Jika Ya,

artinya format email tidak valid, maka alur menuju ke S9. Jika Tidak, maka proses dilanjutkan ke S10.

9. S9 (Menampilkan pesan format email tidak valid)

   Jika format email tidak sesuai, sistem akan menampilkan pesan "Format email tidak valid". Setelah itu proses registrasi dihentikan dan selesai.

10. S10 (Mengecek *Password* kosong)

    Pada tahap ini, sistem mengecek apakah *Password* yang dimasukkan pengguna masih kosong. Jika Ya, maka alur menuju ke S11. Jika Tidak, maka proses dilanjutkan ke S12.

11. S11 (Menampilkan pesan *Password* wajib diisi)

    Jika *Password* kosong, sistem akan menampilkan pesan "*Password* wajib diisi". Setelah itu proses registrasi selesai.

12. S12 (Mengecek *Password* kurang dari 6 karakter)

    Setelah *Password* terisi, sistem mengecek apakah panjang *Password* kurang dari 6 karakter. Jika Ya, maka alur menuju ke S13. Jika Tidak, maka proses dilanjutkan ke S14.

13. S13 (Menampilkan pesan *Password* minimal 6 karakter)

    Jika *Password* terlalu pendek, sistem akan menampilkan pesan "*Password* minimal 6 karakter". Setelah pesan ditampilkan, proses registrasi dihentikan.

14. S14 (Mengecek konfirmasi *Password* kosong)

    Pada tahap ini, sistem mengecek apakah kolom konfirmasi *Password* masih kosong. Jika Ya, maka alur menuju ke S15. Jika Tidak, maka proses dilanjutkan ke S16.

15. S15 (Menampilkan pesan konfirmasi *Password* wajib diisi)

    Jika konfirmasi *Password* belum diisi, sistem akan menampilkan pesan "Konfirmasi *Password* wajib diisi". Setelah itu proses registrasi selesai.

16. S16 (Mengecek *Password* dan konfirmasi *Password* tidak cocok)

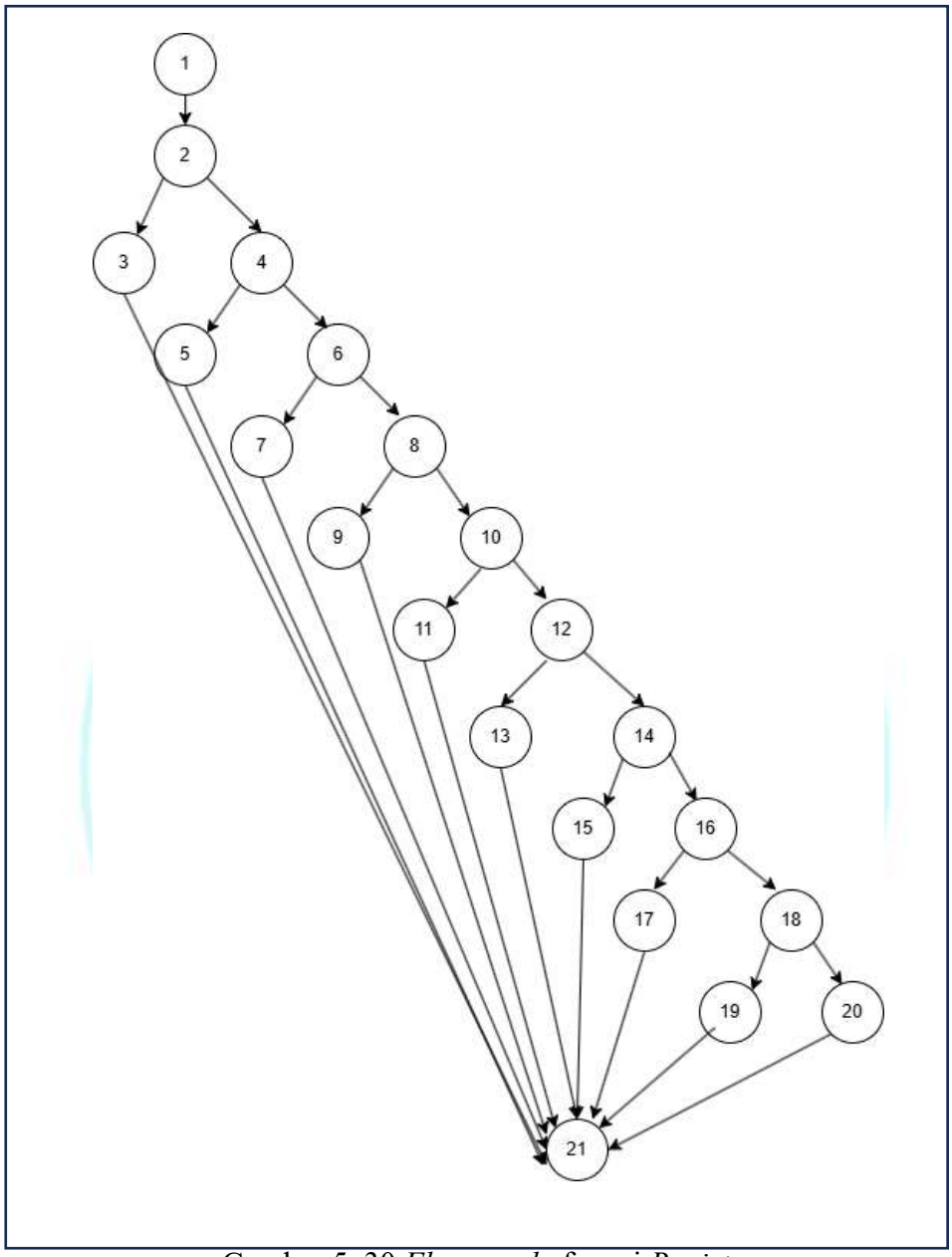
    Pada tahap ini, sistem membandingkan nilai *Password* dan konfirmasi *Password*. Jika Ya, artinya keduanya tidak cocok, maka alur menuju ke S17. Jika Tidak, artinya kedua input tersebut sama, maka proses dilanjutkan ke S18.

17. S17 (Menampilkan pesan konfirmasi *Password* tidak cocok)

Jika *Password* dan konfirmasi *Password* berbeda, sistem akan menampilkan pesan "Konfirmasi *Password* tidak cocok". Setelah itu proses registrasi dihentikan dan selesai.

18. S18 (Mengaktifkan loading, membuat data user, dan menyimpan ke *Database*)
    Jika seluruh input sudah valid, sistem akan mengaktifkan status loading, kemudian membuat objek UserModel berdasarkan data yang dimasukkan pengguna. Setelah itu sistem mengirimkan data tersebut ke *Database* melalui fungsi *Register*User() untuk proses penyimpanan akun baru.
19. S19 (Mengecek apakah terdapat error dari *Database*)
    Setelah proses penyimpanan dilakukan, sistem akan mengecek apakah terdapat pesan error dari *Database*. Jika Ya, maka alur menuju ke S20. Jika Tidak, maka proses dilanjutkan ke S21.
20. S20 (Menampilkan pesan error dari *Database*)
    Jika terjadi kesalahan saat menyimpan data, misalnya *Username* atau email sudah digunakan, sistem akan menampilkan pesan error sesuai hasil dari *Database*. Setelah itu proses registrasi selesai.
21. S21 (Menampilkan pesan registrasi berhasil dan mengarahkan ke halaman *login*)
    Jika tidak ada kesalahan dari *Database*, sistem akan menampilkan pesan registrasi berhasil menggunakan *SnackBar*. Setelah itu pengguna diarahkan ke halaman *login* menggunakan proses navigasi, sehingga akun yang baru dibuat dapat langsung digunakan untuk masuk ke aplikasi.

D. *Flowgraph*



Gambar 5. 30 *Flowgraph* fungsi *Register*

Gambar di atas menjelaskan *Flowgraph* dari fungsi *Register*. *Flowgraph* tersebut mempresentasikan alur eksekusi program berdasarkan hasil identifikasi *Statement* yang telah dilakukan sebelumnya. *Flowgraph* yang dibentuk terdiri dari 21 node dan 29 edge. Adapun penjelasan dari setiap node adalah sebagai berikut.

1. Node 1 terdiri dari S1

   Node ini merupakan proses awal pada fungsi *Register*, yaitu pengguna memasukkan *Username* , email, *Password*, dan konfirmasi *Password*, kemudian sistem mengambil seluruh input tersebut untuk diproses lebih lanjut.
2. Node 2 terdiri dari S2

   Node ini merupakan percabangan untuk mengecek apakah *Username* kosong. Jika *Username* kosong, maka alur akan menuju ke node 3. Jika *Username* terisi, maka alur dilanjutkan ke node 4.
3. Node 3 terdiri dari S3

   Node ini berisi proses menampilkan pesan "*Username* wajib diisi". Node ini dijalankan apabila pengguna belum mengisi *Username* .
4. Node 4 terdiri dari S4

   Node ini merupakan percabangan untuk mengecek apakah *Username* kurang dari 3 karakter. Jika Ya, maka alur menuju node 5. Jika Tidak, maka alur menuju node 6.
5. Node 5 terdiri dari S5

   Node ini berisi proses menampilkan pesan "*Username* minimal 3 karakter". Node ini dijalankan apabila *Username* yang dimasukkan terlalu pendek.
6. Node 6 terdiri dari S6

   Node ini merupakan percabangan untuk mengecek apakah email kosong. Jika email kosong, maka alur menuju node 7. Jika email terisi, maka alur dilanjutkan ke node 8.
7. Node 7 terdiri dari S7

   Node ini berisi proses menampilkan pesan "Email wajib diisi". Node ini dijalankan apabila pengguna belum mengisi email.
8. Node 8 terdiri dari S8

   Node ini merupakan percabangan untuk mengecek apakah format email valid atau tidak. Jika format email tidak valid, maka alur menuju node 9. Jika valid, maka alur dilanjutkan ke node 10.
9. Node 9 terdiri dari S9

   Node ini berisi proses menampilkan pesan "Format email tidak valid". Node ini dijalankan apabila email yang dimasukkan tidak sesuai format.

10. Node 10 terdiri dari S10

    Node ini berisi proses menampilkan pesan "*Password* wajib diisi". Node ini dijalankan apabila pengguna belum mengisi *Password*.

11. Node 11 terdiri dari S11

    Node ini berisi proses menampilkan pesan "*Password* wajib diisi". Node ini dijalankan apabila pengguna belum mengisi *Password*.

12. Node 12 terdiri dari S12

    Node ini merupakan percabangan untuk mengecek apakah *Password* kurang dari 6 karakter. Jika Ya, maka alur menuju node 13. Jika Tidak, maka alur dilanjutkan ke node 14.

13. Node 13 terdiri dari S13

    Node ini berisi proses menampilkan pesan "*Password* minimal 6 karakter". Node ini dijalankan apabila *Password* yang dimasukkan terlalu pendek.

14. Node 14 terdiri dari S14

    Node ini merupakan percabangan untuk mengecek apakah konfirmasi *Password* kosong. Jika Ya, maka alur menuju node 15. Jika Tidak, maka alur dilanjutkan ke node 16.

15. Node 15 terdiri dari S15

    Node ini berisi proses menampilkan pesan "Konfirmasi *Password* wajib diisi". Node ini dijalankan apabila kolom konfirmasi *Password* belum diisi.

16. Node 16 terdiri dari S16

    Node ini merupakan percabangan untuk mengecek apakah *Password* dan konfirmasi *Password* cocok. Jika tidak cocok, maka alur menuju node 17. Jika cocok, maka alur dilanjutkan ke node 18.

17. Node 17 terdiri dari S17

    Node ini berisi proses menampilkan pesan "Konfirmasi *Password* tidak cocok". Node ini dijalankan apabila nilai *Password* dan konfirmasi *Password* berbeda.

18. Node 18 terdiri dari S18

    Node ini merupakan proses utama setelah seluruh validasi input berhasil dilalui. Sistem mengaktifkan status loading, membuat objek UserModel, lalu mengirimkan data ke *Database* melalui proses *Register*User().

19. Node 19 terdiri dari S19

Node ini merupakan percabangan untuk mengecek apakah terdapat errorMessage dari proses penyimpanan *Database*. Jika terdapat error, maka alur menuju node 20. Jika tidak terdapat error, maka alur menuju node 21.

20. Node 20 terdiri dari S20

    Node ini berisi proses menampilkan pesan error dari *Database*. Node ini dijalankan apabila proses penyimpanan data pengguna gagal, misalnya karena *Username* atau email sudah terdaftar.

21. Node 21 terdiri dari S21

    Node ini merupakan proses akhir ketika registrasi berhasil. Sistem menampilkan pesan registrasi berhasil kepada pengguna, kemudian mengarahkan pengguna ke halaman *login* agar dapat masuk menggunakan akun yang baru dibuat.

E. *Cyclomatic Complexity*

Pada modul *Register* terdapat 9 predicate node utama, yaitu:

Tabel 5. 8 *Cyclomatic Complexity* Fungsi *Register*

No	Predicate Node	Kondisi
1	if (userame.isEmpty)	Mengecek apakah *Username* kosong
2	if (*Username* .length < 3)	Mengecek apakah *Username* kurang dari 3 karakter
3	if (email.isEmpty)	Mengecek apakah emai lkosong
4	if (!emailRegex.hasMatch(email))	Mengecek apakah format email tidak valid
5	if (*Password*.isEmpty)	Mengecek apakah *Password* kosong
6	if (*Password*.length < 6)	Mengecek apakah *Password* kurang dari 6 karakter
7	if (confirm*Password*.isEmpty)	Mengecek apakah konfirmasi *Password* kosong
8	if (*Password* != confirm*Password*)	Mengecek apakah *Password* dan konfirmasi *Password* tidak cocok
9	if (errorMessage != null)	Mengecek apakah terdapat error dari *Database*

Maka perhitungan *Cyclomatic Complexity* adalah:

V(G) = P + 1

V(G) = 9 + 1

V(G) = 10

Berdasarkan hasil perhitungan tersebut, nilai *Cyclomatic Complexity* pada modul *Register* adalah 10. Artinya, terdapat 10 jalur independen yang harus diuji untuk memastikan seluruh alur logika utama pada modul *Register* berjalan dengan baik.

F. *Independent Path*

Berdasarkan hasil perhitungan *Cyclomatic Complexity*, diperoleh 10 *Independent Path* sebagai berikut.

1. Path 1

   1 → 2 → 3

   (*Username* kosong)

2. Path 2

   1→2→4→5

   (*Username* terisi, tetapi kurang dari 3 karakter)

3. Path 3

   1→2→4→6→7

   (*Username* valid, tetapi email kosong)

4. Path 4

   1→2→4→6→8→9

   (Email diisi, tetapi format email tidak valid)

5. Path 5

   1→2→4→6→8→10→11

   (*Username* dan email valid, tetapi *Password* kosong)

6. Path 6

   1→2→4→6→8→10→12→13

   (*Password* diisi, tetapi kurang dari 6 karakter)

7. Path 7

   1→2→4→6→8→10→12→14→15

   (Konfirmasi *Password* kosong)

8. Path 8

   1→2→4→6→8→10→12→14→16→17

   (*Password* dan konfirmasi *Password* tidak cocok)

9. Path 9

   1→2→4→6→8→10→12→14→16→18→19→20

   (Semua input valid, tetapi terdapat error dari *Database*)

10. Path 10

    1→2→4→6→8→10→12→14→16→18→19→21

    (Semua input valid dan registrasi berhasil)

G. *Test case*

Berikut hasil *Test case* pada fungsi *Register* berdasarkan hasil dari *Independent Path*.

Tabel 5. 9 *Test case* Fungsi *Register*

Test case	***Username***	**Email**	***Password***	**Konfirmasi *Password***	**Output yang Diharapkan**	**Hasil**
TC1	(Kosong)	windi@gmail.com	Windi123	Windi123	*Username* wajib diisi	Valid
TC2	wi	windi@gmail.com	Windi123	Windi123	*Username* minimal 3 karakter	Valid
TC3	windiyulinda	(Kosong)	Windi123	Windi123	Email wajib diisi	Valid
TC4	windiyulinda	windi@gmaill.com	Windi123	Windi123	Format email tidak valid	Valid
TC5	windiyulinda	windi@gmail.com	(Kosong)	Windi123	*Password* wajib diisi	Valid
TC6	windiyulinda	windi@gmail.com	123	123	*Password* minimal 6 karakter	Valid
TC7	windiyulinda	windi@gmail.com	Windi123	(Kosong)	Konfirmasi *Password* wajib diisi	Valid
TC8	windiyulinda	windi@gmail.com	Windi123	Windi124	Konfirmasi *Password* tidak cocok	Valid
TC9	windiyulinda	windi@gmail.com	Windi123	Windi123	Pesan error dari *Database* tampil	Valid
C10	windiyulinda2	windi2@gmail.com	Wind123	Windi123	Registrasi berhasil, silakan *login* dan masuk ke halaman *Login*	Valid

H. *Statement* Covarege

Statement Coverage digunakan untuk mengetahui apakah seluruh *Statement* pada kode sudah dijalankan oleh *test case*.

Rumus:

Statement Coverage = (Jumlah *Statement* yang dieksekusi / Jumlah seluruh *Statement*) × 100%

Tabel 5. 10 *Statement Coverage* Fungsi *Register*

Test case	*Statement* yang Dieksekusi	Keterangan
TC1	S1, S2, S3	Menguji kondisi *Username* kosong
TC2	S1, S2, S4, S5	Menguji kondisi *Username* kurang dari 3 karakter
TC3	S1, S2, S4, S6, S7	Menguji kondisi email kosong
TC4	S1, S2, S4, S6, S8, S9	Menguji kondisi format email tidak valid
TC5	S1, S2, S4, S6, S8, S10, S11	Menguji kondisi *Password* kosong
TC6	S1, S2, S4, S6, S8, S10, S12, S13	Menguji kondisi *Password* kurang dari 6 karakter
TC7	S1, S2, S4, S6, S8, S10, S12, S14, S15	Menguji kondisi konfirmasi *Password* kosong
TC8	S1, S2, S4, S6, S8, S10, S12, S14, S16, S17	Menguji kondisi *Password* dan konfirmasi *Password* tidak cocok
TC9	S1, S2, S4, S6, S8, S10, S12, S14, S16, S18, S19, S20	Menguji kondisi error dari *Database*
TC10	S1, S2, S4, S6, S8, S10, S12, S14, S16, S18, S19, S21	Menguji kondisi registrasi berhasil

Berdasarkan tabel di atas, seluruh *Statement* dari S1 sampai S21 telah dieksekusi melalui *test case* yang dibuat. Dengan demikian, jumlah *Statement* yang dieksekusi adalah 21 dari total 21 *Statement*.

Maka perhitungan *Statement Coverage* adalah:

Statement Coverage = (Jumlah *Statement* yang dieksekusi / Jumlah seluruh *Statement*) × 100%

Statement Coverage = (21 / 21) × 100%

Statement Coverage = 100%

Berdasarkan hasil perhitungan tersebut, nilai *Statement Coverage* pada modul *Register* adalah 100%. Artinya, seluruh *Statement* utama pada modul *Register* telah diuji melalui *test case* yang dibuat.

5.4.3 Pengujian Modul Tambah *Habit*

A. Potongan *Source Code*

```
Future<void> _saveHabit() async {
  final title = nameController.text.trim();
  if (title.isEmpty) {
    ScaffoldMessenger.of(context).showSnackBar(
      const SnackBar(content: Text('Nama habit wajib diisi')),
    );
    return;
  }
  if (descController.text.trim().isEmpty) {
    ScaffoldMessenger.of(context).showSnackBar(
      const SnackBar(content: Text('Deskripsi wajib diisi')),
    );
    return;
  }
  if (selectedCategory == null || selectedCategory!.isEmpty) {
    ScaffoldMessenger.of(context).showSnackBar(
      const SnackBar(content: Text('Kategori wajib dipilih')),
    );
    return;
  }
  setState(() {
    _isSaving = true;
```

```
});
try {
  await _habitService.addHabit(
    title: title,
    description: descController.text.trim(),
    category: selectedCategory!,
    targetPerWeek: target.toInt(),
    targetValue: double.parse(targetValueController.text),
    targetUnit: selectedTargetUnit,
    reminderEnabled: _reminderEnabled,
    reminderHour:
        _reminderEnabled ? _reminderTime.hour : null,
    reminderMinute:
        _reminderEnabled ? _reminderTime.minute : null,
  );
  if (!mounted) return;
  ScaffoldMessenger.of(context).showSnackBar(
    const SnackBar(
      content: Text('Habit berhasil ditambahkan'),
    ),
  );
  Navigator.pop(context, true);
} catch (e) {
  if (!mounted) return;
  ScaffoldMessenger.of(context).showSnackBar(
    SnackBar(
      content: Text(
        'Gagal menambah habit: $e',
```

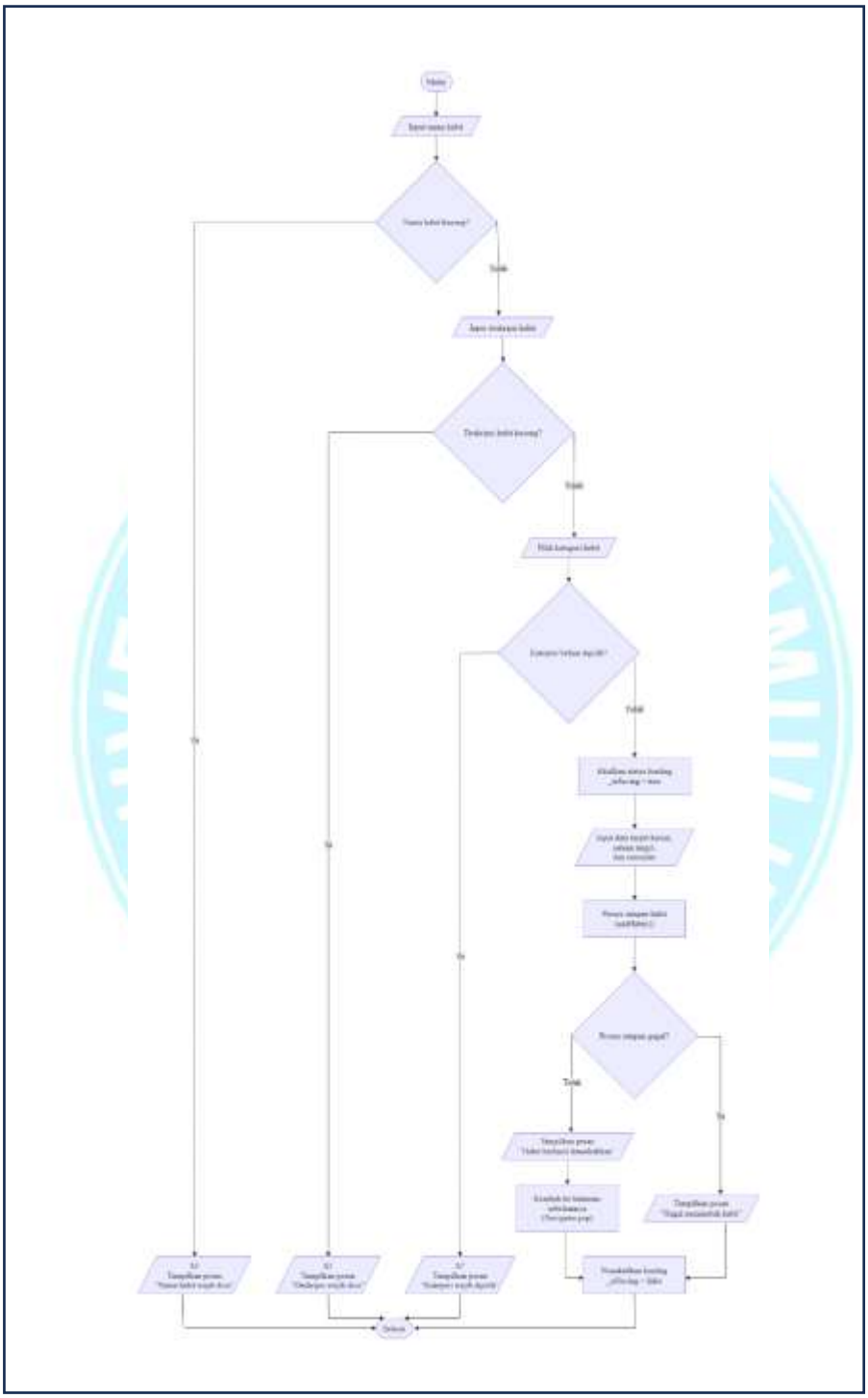
```
        ),
      ),
    );
  } finally {
    if (mounted) {
      setState(() {
        _isSaving = false;
      });
    }
  }
}
```

B. Identifikasi *Statement*

Tabel 5. 11 *Identifikasi Statement* Fungsi Tambah *Habit*

Statement	Keterangan
S1	Mengambil nilai nama habit dari nameController.
S2	Memeriksa apakah nama habit kosong (title.isEmpty).
S3	Menampilkan pesan "Nama habit wajib diisi" kemudian proses dihentikan (return).
S4	Memeriksa apakah deskripsi habit kosong (descController.text.trim().isEmpty).
S5	Menampilkan pesan "Deskripsi wajib diisi" kemudian proses dihentikan (return).
S6	Memeriksa apakah kategori belum dipilih (`selectedCategory == null
S7	Menampilkan pesan "Kategori wajib dipilih" kemudian proses dihentikan (return).
S8	Mengaktifkan indikator loading dengan mengubah _isSaving = true.
S9	Menjalankan proses penyimpanan habit menggunakan addHabit() dengan parameter nama habit, deskripsi, kategori, target mingguan, target harian, satuan target, dan reminder.
S10	Memeriksa kondisi mounted sebelum memperbarui tampilan.
S11	Menampilkan pesan "Habit berhasil ditambahkan".
S12	Kembali ke halaman sebelumnya menggunakan Navigator.pop(context, true).
S13	Menangani exception apabila proses penyimpanan gagal (catch).
S14	Menampilkan pesan "Gagal menambah habit".
S15	Menjalankan blok finally untuk memastikan proses pembersihan tetap dilakukan.
S16	Memeriksa kembali kondisi mounted.
S17	Menonaktifkan indikator loading dengan mengubah _isSaving = false.

C. *Flowchart*



Gambar 5. 31 *Flowchart* fungsi Tambah *Habit*

Gambar di atas menjelaskan *flowchart* fungsi Tambah *Habit*. *Flowchart* tersebut menggambarkan alur proses ketika pengguna menambahkan *habit* baru, mulai dari mengisi data *habit*, melakukan validasi data yang diinput, menyimpan

data ke dalam *database*, hingga sistem memberikan informasi apakah proses penyimpanan berhasil atau gagal. Adapun penjelasannya sebagai berikut.

1. S1 (Input nama *Habit*)

   Pada tahap ini, pengguna memasukkan nama habit ke dalam *form* tambah *habit*. Sistem kemudian mengambil nilai nama *habit* menggunakan fungsi trim() agar spasi kosong di awal maupun di akhir input tidak ikut diproses.
2. S2 (Mengecek nama *Habit* kosong)

   Setelah nama *habit* diinput, sistem memeriksa apakah nilai tersebut kosong. Apabila nama *habit* masih kosong, maka proses akan menuju ke S3. Sebaliknya, jika nama *habit* telah diisi, proses dilanjutkan ke tahap input deskripsi *habit*.
3. S3 (Menampilkan pesan nama *Habit* wajib diisi)

   Jika nama *habit* belum diisi, sistem akan menampilkan pesan "Nama *habit* wajib diisi" menggunakan *SnackBar*. Setelah pesan ditampilkan, proses tambah *habit* dihentikan dan berakhir.
4. S4 (Input Deskripsi Habit)

   Pada tahap ini, pengguna mengisi deskripsi *habit* yang akan dibuat. Selanjutnya sistem melakukan proses validasi terhadap data tersebut.
5. S5 (Mengecek deskripsi *Habit* kosong)

   Sistem memeriksa apakah deskripsi *habit* masih kosong. Jika deskripsi belum diisi, maka proses akan menuju ke S6. Sebaliknya, apabila deskripsi telah diisi, proses dilanjutkan ke tahap pemilihan kategori *habit*.
6. S6 (Menampilkan Pesan Deskripsi Wajib Diisi)

   Jika deskripsi *habit* kosong, sistem menampilkan pesan "Deskripsi wajib diisi" menggunakan *SnackBar*. Setelah itu proses tambah habit dihentikan.
7. S7 (Pilih Kategori Habit)

   Pada tahap ini, pengguna memilih kategori *habit* yang sesuai, seperti Kesehatan, Belajar, Produktivitas, dan kategori lainnya. Setelah kategori dipilih, sistem melakukan validasi kategori.
8. S8 (Mengecek Kategori Belum Dipilih)

   Sistem memeriksa apakah pengguna sudah memilih kategori *habit*. Jika kategori belum dipilih, maka proses menuju ke S9. Sebaliknya, jika kategori telah dipilih, proses dilanjutkan ke tahap berikutnya.

9. S9 (Menampilkan Pesan Kategori Wajib Dipilih)

   Jika kategori belum dipilih, sistem menampilkan pesan "Kategori wajib dipilih" menggunakan *SnackBar*. Setelah pesan ditampilkan, proses tambah habit dihentikan.

10. S10 (Mengaktifkan Status *Loading*)

    Setelah seluruh data dinyatakan valid, sistem mengaktifkan indikator *loading* dengan mengubah nilai *_isSaving = true*. Tujuannya untuk memberi informasi kepada pengguna bahwa proses penyimpanan data sedang berlangsung.

11. S11 (Mengambil Data Target Harian, Satuan Target, dan *Reminder*)

    Pada tahap ini sistem mengambil seluruh data pendukung yang telah diinput pengguna, yaitu target harian, satuan target, status reminder, serta waktu *reminder* apabila *reminder* diaktifkan. Data tersebut akan digunakan sebagai parameter saat proses penyimpanan *habit*.

12. S12 (Proses Simpan *Habit*)

    Sistem memanggil fungsi *addHabit*() untuk menyimpan seluruh data *habit* ke dalam *database*, meliputi nama *habit*, deskripsi, kategori, target mingguan, target harian, satuan target, status *reminder*, serta waktu *reminder*.

13. S13 (Mengecek Apakah Proses Simpan Gagal)

    Setelah proses penyimpanan dilakukan, sistem memeriksa apakah terjadi kesalahan (*exception*) selama proses penyimpanan data. Jika terjadi kegagalan, proses menuju ke S14. Sebaliknya, apabila penyimpanan berhasil, proses dilanjutkan ke S15.

14. S14 (Menampilkan Pesan Gagal Menambah *Habit*)

    Apabila terjadi kesalahan ketika menyimpan data, sistem akan menampilkan pesan "Gagal menambah *habit*" beserta informasi kesalahan yang terjadi. Selanjutnya proses dilanjutkan ke tahap menonaktifkan status *loading*.

15. S15 (Menampilkan Pesan Habit Berhasil Ditambahkan)

    Jika data habit berhasil disimpan ke database, sistem akan menampilkan pesan "*Habit* berhasil ditambahkan" sebagai tanda bahwa proses penyimpanan telah berhasil dilakukan.

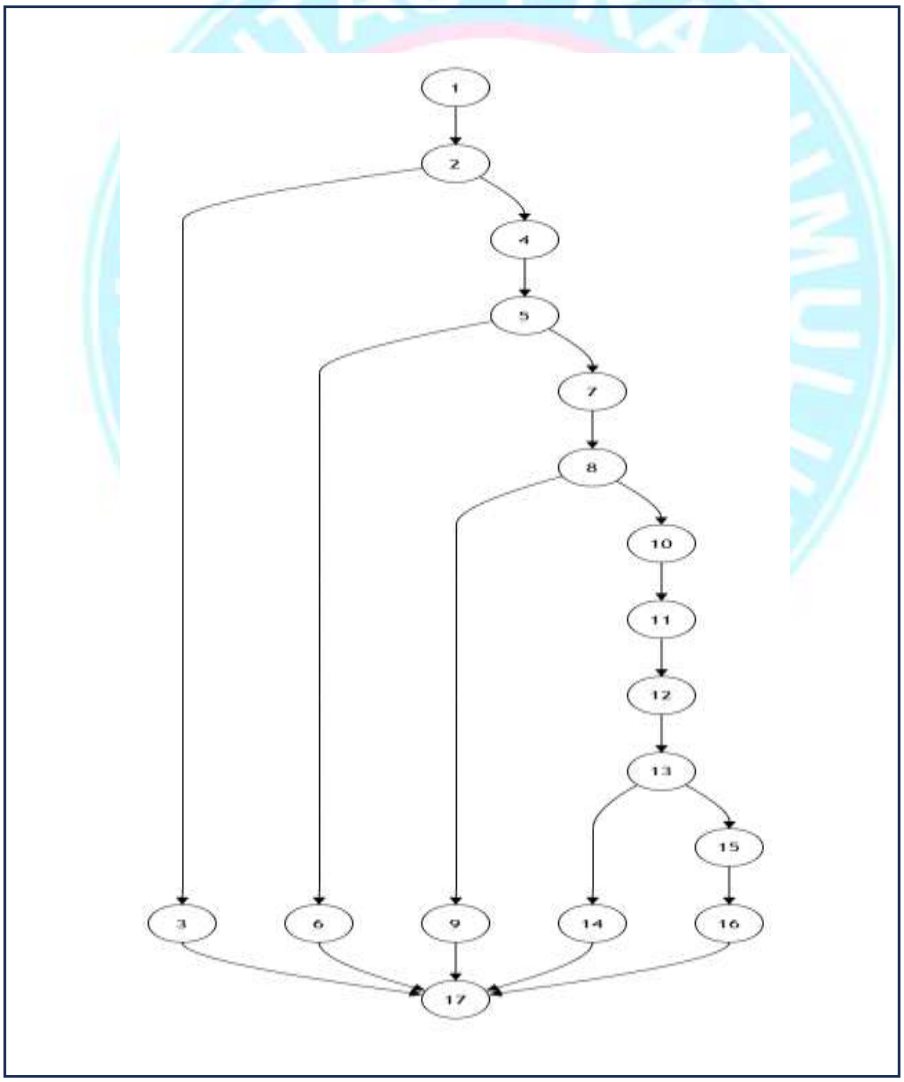
16. S16 (Kembali ke Halaman Sebelumnya)

Setelah pesan berhasil ditampilkan, sistem mengarahkan pengguna kembali ke halaman sebelumnya menggunakan *Navigator.pop*(*context, true*) sehingga halaman daftar *habit* dapat diperbarui.

17. S17 (Menonaktifkan Status *Loading*)

    Pada tahap terakhir, sistem menonaktifkan indikator loading dengan mengubah nilai *_isSaving = false*. Setelah status loading dinonaktifkan, proses tambah *habit* dinyatakan selesai.

D. *Flowgraph*



Gambar 5. 32 *Flowgraph* fungsi Tambah *Habit*

Gambar di atas menjelaskan *Flowgraph* dari fungsi tambah *Habit*. *Flowgraph* tersebut mempresentasikan alur eksekusi program berdasarkan hasil

identifikasi *Statement* yang telah dilakukan sebelumnya. *Flowgraph* yang dibentuk terdiri dari 16 node dan 19 edge. Adapun penjelasan dari setiap node adalah sebagai berikut.

1. Node 1 terdiri dari S1

   Node ini merupakan proses awal pada fungsi tambah *habit*, yaitu pengguna memasukkan nama *habit* ke dalam *form*. Sistem kemudian mengambil nilai *input* nama *habit* menggunakan fungsi trim() untuk menghilangkan spasi di awal maupun di akhir *input*.
2. Node 2 terdiri dari S2

   Node ini merupakan percabangan untuk memeriksa apakah nama *habit* yang dimasukkan kosong. Jika nama *habit* kosong, maka alur menuju node 3. Sebaliknya, apabila nama *habit* telah diisi, proses dilanjutkan ke node 4.
3. Node 3 terdiri dari S3

   Node ini berisi proses menampilkan pesan "Nama *habit* wajib diisi" menggunakan *SnackBar*. Setelah pesan ditampilkan, proses langsung menuju node akhir.
4. Node 4 terdiri dari S4

   Node ini merupakan proses pengguna memasukkan deskripsi *habit* sebelum dilakukan validasi terhadap data tersebut.
5. Node 5 terdiri dari S5

   Node ini merupakan percabangan untuk memeriksa apakah deskripsi *habit* masih kosong. Jika deskripsi kosong, maka alur menuju node 6. Sebaliknya, apabila deskripsi telah diisi, proses dilanjutkan ke node 7.
6. Node 6 terdiri dari S6

   Node ini berisi proses menampilkan pesan "Deskripsi wajib diisi" menggunakan *SnackBar*. Setelah itu proses langsung menuju node akhir.
7. Node 7 terdiri dari S7

   Node ini merupakan proses pengguna memilih kategori *habit* yang akan digunakan sebelum dilakukan validasi kategori.

8. Node 8 terdiri dari S8

   Node ini merupakan percabangan untuk memeriksa apakah kategori *habit* belum dipilih. Jika kategori belum dipilih, maka alur menuju node 9. Sebaliknya, apabila kategori telah dipilih, proses dilanjutkan ke node 10.

9. Node 9 terdiri dari S9

   Node ini berisi proses menampilkan pesan "Kategori wajib dipilih" menggunakan *SnackBar*. Setelah pesan ditampilkan, proses langsung menuju node akhir.

10. Node 10 terdiri dari S10

    Node ini merupakan proses mengaktifkan indikator loading dengan mengubah nilai *_isSaving* = *true* sebagai tanda bahwa proses penyimpanan data sedang berlangsung.

11. Node 11 terdiri dari S11

    Node ini berisi proses mengambil data target harian, satuan target, status reminder, serta waktu reminder yang akan digunakan sebagai parameter pada saat proses penyimpanan *habit*.

12. Node 12 terdiri dari S12

    Node ini merupakan proses memanggil fungsi *addHabit*() untuk menyimpan seluruh data habit ke dalam database.

13. Node 13 terdiri dari S13

    Node ini merupakan percabangan untuk memeriksa apakah proses penyimpanan habit mengalami kegagalan (*exception*). Jika terjadi kegagalan, alur menuju node 14. Sebaliknya, apabila penyimpanan berhasil, proses dilanjutkan ke node 15.

14. Node 14 terdiri dari S14

    Node ini berisi proses menampilkan pesan "Gagal menambah *habit*" apabila terjadi kesalahan saat penyimpanan data. Setelah itu proses menuju node 17.

15. Node 15 terdiri dari S15

    Node ini berisi proses menampilkan pesan "*Habit* berhasil ditambahkan" sebagai tanda bahwa data berhasil disimpan ke dalam *database*.

16. Node 16 terdiri dari S16

    Node ini merupakan proses mengembalikan pengguna ke halaman sebelumnya menggunakan *Navigator.pop*(context, true) setelah proses penyimpanan berhasil dilakukan.

E. *Cyclomatic Complexity*

Pada modul tambah *Habit* terdapat 4 predicate node utama, yaitu:

Tabel 5. 12 *Cyclomatic Complexity* Fungsi Tambah *Habit*

No	Predicate Node	Kondisi
1	if (title.isEmpty)	Mengecek apakah nama habit kosong.
2	if (descController.text.trim().isEmpty)	Mengecek apakah deskripsi habit kosong.
3	if (selectedCategory == null || selectedCategory!.isEmpty)	Mengecek apakah kategori habit belum dipilih.
4	catch (e)	Menangani kondisi apabila terjadi kegagalan saat proses penyimpanan habit.

Maka perhitungan *Cyclomatic Complexity* adalah:

V(G) = P + 1

V(G) = 4 + 1

V(G) = 5

Berdasarkan hasil perhitungan tersebut, nilai *Cyclomatic Complexity* pada modul tambah *Habit* adalah 5. Artinya, terdapat 5 jalur independen yang harus diuji untuk memastikan seluruh alur logika utama pada modul tambah *Habit* berjalan dengan baik.

F. *Independent Path*

Berdasarkan hasil perhitungan *Cyclomatic Complexity*, diperoleh 5 *Independent Path* sebagai berikut.

1. Path 1

   1 → 2 → 3 → 17

   (Pengguna tidak mengisi nama habit sehingga sistem menampilkan pesan *"Nama habit wajib diisi"* kemudian proses selesai.)

2. Path 2

   1 → 2 → 4 → 5 →6→ 17

   (Nama habit telah diisi, tetapi deskripsi habit kosong sehingga sistem menampilkan pesan *"Deskripsi wajib diisi"* kemudian proses selesai.)
3. Path 3

   1 → 2 → 4 → 5 → 7 → 8 → 9 → 17

   (Nama dan deskripsi telah diisi, tetapi kategori belum dipilih sehingga sistem menampilkan pesan *"Kategori wajib dipilih"* kemudian proses selesai.)
4. Path 4

   1 → 2 → 4 → 5 → 7 → 8 → 10 → 11 → 12 → 13 → 14 → 17

   (Seluruh data valid, tetapi terjadi kesalahan saat proses penyimpanan sehingga sistem menampilkan pesan *"Gagal menambah habit"* kemudian menonaktifkan loading.)
5. Path 5

   1 → 2 → 4 → 5 → 7 → 8 → 10 → 11 → 12 → 13 → 15 → 16 → 17

   (Seluruh data valid dan proses penyimpanan berhasil sehingga sistem menampilkan pesan *"Habit berhasil ditambahkan"*, kembali ke halaman sebelumnya, kemudian menonaktifkan loading.)

G. *Test case*

Berikut hasil *test case* pada fungsi tambah *Habit* berdasarkan *Independent Path*.

Tabel 5. 13 *Test case* Fungsi *Tambah Habit*

No	Independent Path	Data Uji	Hasil yang Diharapkan	Hasil Pengujian
TC1	1 → 2 → 3 → 17	Nama habit dikosongkan	Sistem menampilkan pesan "Nama habit wajib diisi", proses penyimpanan dibatalkan.	Sesuai
TC2	1 → 2 → 4 → 5 → 6 → 17	Nama habit diisi, deskripsi dikosongkan	Sistem menampilkan pesan "Deskripsi wajib diisi", proses penyimpanan dibatalkan.	Sesuai
TC3	1 → 2 → 4 → 5 → 7 → 8 → 9 → 17	Nama dan deskripsi diisi, kategori tidak dipilih	Sistem menampilkan pesan "Kategori wajib dipilih",	Sesuai

			proses penyimpanan dibatalkan.	
TC4	1 → 2 → 4 → 5 → 7 → 8 → 10 → 11 → 12 → 13 → 14 → 17	Seluruh data valid, namun terjadi kesalahan saat proses penyimpanan (exception)	Sistem menampilkan pesan "Gagal menambah habit" kemudian menonaktifkan status loading.	Sesuai
TC5	1 → 2 → 4 → 5 → 7 → 8 → 10 → 11 → 12 → 13 → 15 → 16 → 17	Seluruh data diisi dengan benar	Sistem berhasil menyimpan data habit, menampilkan pesan "Habit berhasil ditambahkan", kembali ke halaman sebelumnya, kemudian menonaktifkan status loading.	Sesuai

H. *Statement Coverage*

Statement Coverage digunakan untuk mengetahui apakah seluruh *Statement* pada kode sudah dijalankan oleh *test case*.

Rumus:

Statement Coverage = (Jumlah *Statement* yang dieksekusi / Jumlah seluruh *Statement*) × 100%

Tabel 5. 14 *Statement Covarege* Fungsi Tambah *Habit*

Test Case	**Statement yang Dieksekusi**	**Keterangan**
TC1	S1, S2, S3	Menguji kondisi nama habit kosong.
TC2	S1, S2, S4, S5, S6	Menguji kondisi deskripsi habit kosong.
TC3	S1, S2, S4, S5, S7, S8, S9	Menguji kondisi kategori habit belum dipilih.
TC4	S1, S2, S4, S5, S7, S8, S10, S11, S12, S13, S14, S17	Menguji kondisi seluruh data valid tetapi proses penyimpanan gagal.
TC5	S1, S2, S4, S5, S7, S8, S10, S11, S12, S13, S15, S16, S17	Menguji kondisi seluruh data valid dan proses penyimpanan berhasil.

Berdasarkan tabel di atas, seluruh statement mulai dari S1 sampai S17 telah dieksekusi melalui lima test case yang telah disusun. Dengan demikian, jumlah statement yang dieksekusi adalah 17 statement dari total 17 statement yang terdapat pada fungsi Tambah Habit.

Maka perhitungan *Statement Coverage* adalah:

Statement Coverage = (Jumlah *Statement* yang dieksekusi / Jumlah seluruh *Statement*) × 100%

Statement Coverage = (17 / 17) × 100%
Statement Coverage = 100%

Berdasarkan hasil perhitungan tersebut, nilai *Statement Coverage* pada modul tambah *Habit* adalah 100%. Artinya, seluruh *Statement* utama pada modul tambah *Habit* telah diuji melalui *test case* yang dibuat.

5.4.4 Pengujian *Input Progress Habit*

A. Potongan *Source Code*

```
onPressed: () async {

  if (habit.id == null) return;

  final alreadyInput =
     await _habitService.hasInputProgressToday(
    habit.id!,
  );

  if (alreadyInput) {

    if (!mounted) return;

    ScaffoldMessenger.of(context).showSnackBar(
      const SnackBar(
        content: Text(
          "Progress untuk habit ini sudah diinput hari ini. Silakan lanjutkan besok.",
        ),
        backgroundColor: Colors.orange,
      ),
    );

    return;
  }
```

```
final progress =
    double.tryParse(controller.text) ?? 0;

await _habitService.saveHabitProgress(
  habitId: habit.id!,
  date: selectedDate,
  progressValue: progress,
);

final bool targetTercapai =
    progress >= habit.targetValue;

if (!mounted) return;

if (targetTercapai) {

  setState(() {
    diamondBadgeHabitIds.add(habit.id!);
  });

  await showDialog(
    context: context,
    barrierDismissible: false,
    builder: (_) => FlyingBadgeAnimation(
      badge: "Diamond",
      badgeIcon: " 💎 ",
      onFinished: () {
        Navigator.pop(context);
      },
    ),
  );

  ScaffoldMessenger.of(context).showSnackBar(
    const SnackBar(
      backgroundColor: Colors.green,
      content: Text(
        " 🎉 Selamat! Target hari ini berhasil dicapai.",
      ),
    ),
  );
```

```
} else {

  ScaffoldMessenger.of(context).showSnackBar(
    const SnackBar(
      backgroundColor: Colors.orange,
      content: Text(
        "Progress berhasil disimpan. Target hari ini belum tercapai.",
      ),
    ),
  );

}

 await _loadHabits();

}
```

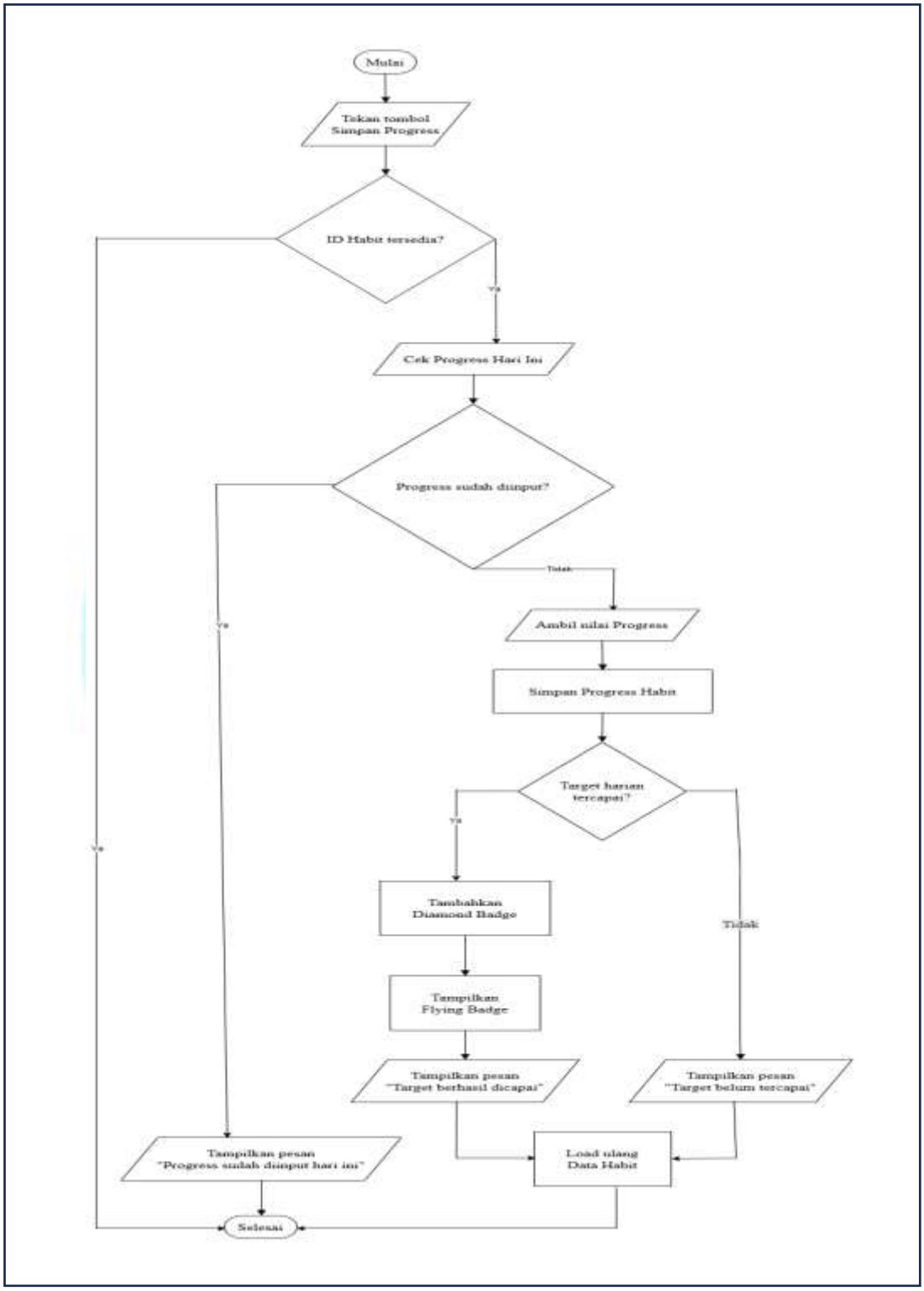
B. Identifikasi *Statement*

Tabel 5. 15 *Identifikasi Statement* Fungsi *Input Progress Habit*

Kode	Statement	Keterangan
S1	onPressed() async	Menjalankan proses ketika pengguna menekan tombol Simpan Progress.
S2	if (habit.id == null) return	Mengecek apakah ID habit tersedia. Jika tidak tersedia, proses dihentikan.
S3	hasInputProgressToday(habit.id!)	Mengecek apakah progress habit telah diinput pada hari yang sama.
S4	if (alreadyInput)	Mengecek apakah pengguna sudah menginput progress hari ini.
S5	showSnackBar("Progress untuk habit ini sudah diinput hari ini...")	Menampilkan pesan bahwa progress sudah diinput sehingga proses dihentikan.
S6	double.tryParse(controller.text) ?? 0	Mengambil dan mengubah nilai progress yang diinput pengguna menjadi tipe double.
S7	saveHabitProgress(...)	Menyimpan data progress habit ke dalam database.

S8	progress >= habit.targetValue	Membandingkan progress dengan target harian untuk menentukan apakah target telah tercapai.
S9	setState(() { diamondBadgeHabitIds.add(...) })	Menambahkan Diamond Badge apabila target berhasil dicapai.
S10	showDialog(FlyingBadgeAnimation(...))	Menampilkan animasi Diamond Badge kepada pengguna.
S11	showSnackBar("Selamat! Target hari ini berhasil dicapai.")	Menampilkan pesan bahwa target harian berhasil dicapai.
S12	showSnackBar("Progress berhasil disimpan. Target hari ini belum tercapai.")	Menampilkan pesan bahwa progress berhasil disimpan tetapi target belum tercapai.
S13	_loadHabits()	Memuat kembali data habit agar perubahan progress langsung ditampilkan pada halaman.

C. *Flowchart*



Gambar 5. 33 *Flowchart* fungsi *Checklist Habit*

Gambar di atas menjelaskan *Flowchart* dari fungsi *Input Progress Habit*. *Flowchart* tersebut menggambarkan alur proses ketika pengguna menekan tombol Simpan *Progress*, kemudian sistem melakukan validasi data *habit*, memeriksa apakah *progress* sudah pernah diinput pada hari yang sama, menyimpan *progress* ke dalam *database*, mengevaluasi pencapaian target harian, memberikan penghargaan apabila target tercapai, serta memuat ulang data habit agar perubahan dapat langsung ditampilkan pada aplikasi. Adapun penjelasannya sebagai berikut.

1. S1 (Pengguna menekan tombol Simpan Progress)

   Pada tahap ini, pengguna menekan tombol Simpan *Progress* pada *habit* yang dipilih. Proses ini menjadi awal eksekusi fungsi Input *Progress Habit*.
2. S2 (Mengecek apakah ID Habit tersedia)

   Setelah tombol ditekan, sistem memeriksa apakah ID *Habit* tersedia atau bernilai null. Jika *ID Habit* tidak tersedia, maka proses langsung dihentikan dan selesai. Jika *ID Habit* tersedia, maka proses dilanjutkan ke tahap berikutnya.
3. S3 (Mengecek progress hari ini)

   Pada tahap ini, sistem memanggil fungsi *hasInputProgressToday*() untuk memeriksa apakah pengguna telah memasukkan *progress* pada *habit* tersebut pada hari yang sama.
4. S4 (Mengecek apakah progress sudah diinput)

   Setelah proses pengecekan selesai, sistem menentukan apakah *progress* untuk *habit* tersebut sudah pernah diinput pada hari itu. Jika Ya, maka alur menuju ke S5. Jika Tidak, maka proses dilanjutkan ke S6.
5. S5 (Menampilkan pesan progress sudah diinput)

   Apabila pengguna telah menginput progress pada hari yang sama, sistem akan menampilkan pesan "*Progress* sudah diinput hari ini. Silakan lanjutkan besok." menggunakan *SnackBar*. Setelah pesan ditampilkan, proses selesai.
6. S6 (Mengambil nilai *progress*)

   Jika progress belum pernah diinput, sistem mengambil nilai progress yang dimasukkan oleh pengguna melalui TextField kemudian mengubahnya ke dalam tipe data double menggunakan fungsi double.tryParse().

7. S7 (Menyimpan progress habit)

   Setelah nilai progress diperoleh, sistem memanggil fungsi *saveHabitProgress*() untuk menyimpan data progress habit ke dalam *database* sesuai dengan tanggal yang dipilih.
8. S8 (Mengecek apakah target harian tercapai)

   Setelah proses penyimpanan selesai, sistem membandingkan nilai *progress* yang diinput dengan target harian habit. Jika *progress* telah mencapai atau melebihi target, maka alur menuju ke S9. Sebaliknya, apabila target belum tercapai, maka alur menuju ke S12.
9. S9 (Menambahkan *Diamond Badge*)

   Apabila target harian berhasil dicapai, sistem akan menambahkan *Diamond Badge* sebagai bentuk penghargaan kepada pengguna atas keberhasilan menyelesaikan target hari tersebut.
10. S10 (Menampilkan *Flying Badge*)

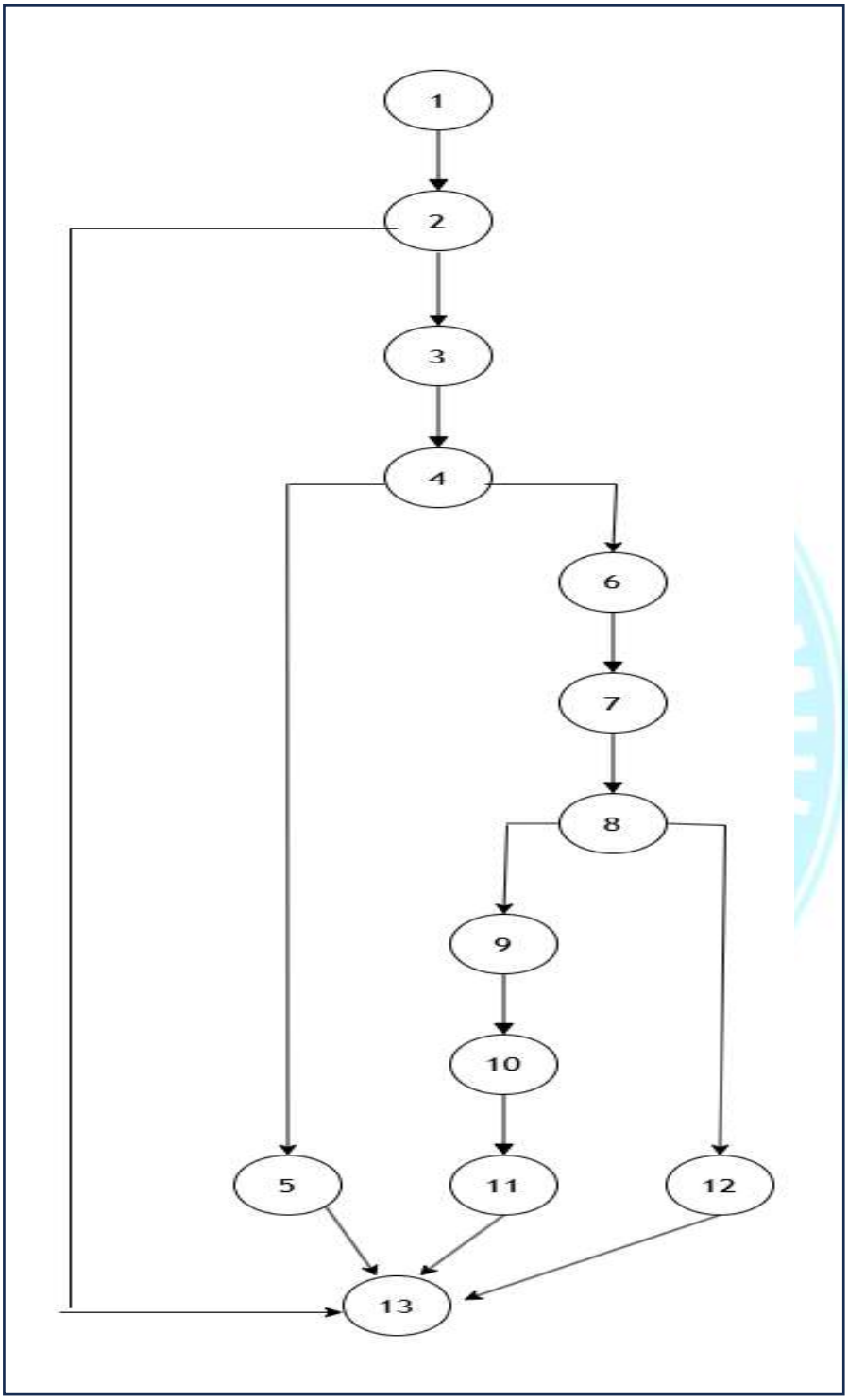
    Setelah *Diamond Badge* berhasil ditambahkan, sistem menampilkan animasi *Flying Badge* sebagai bentuk apresiasi kepada pengguna.
11. S11 (Menampilkan pesan target berhasil dicapai)

    Setelah animasi selesai ditampilkan, sistem menampilkan pesan "Selamat! Target hari ini berhasil dicapai." menggunakan *SnackBar* sebagai konfirmasi bahwa target harian telah berhasil diselesaikan.
12. S12 (Menampilkan pesan target belum tercapai)

    Apabila progress yang diinput masih berada di bawah target harian, sistem akan menampilkan pesan "Progress berhasil disimpan. Target hari ini belum tercapai." menggunakan *SnackBar* sebagai informasi kepada pengguna.
13. S13 (Memuat ulang data habit)

    Pada tahap terakhir, baik pada kondisi target berhasil dicapai maupun belum tercapai, sistem memanggil fungsi _loadHabits() untuk memuat kembali daftar habit sehingga progress terbaru dapat langsung ditampilkan pada halaman aplikasi. Setelah proses tersebut selesai, fungsi Input Progress Habit berakhir.

D. *Flowgraph*



Gambar 5. 34 *Flowgraph* fungsi *Input Progress habit Habit*

Gambar di atas menjelaskan *Flowgraph* dari fungsi *Input Progress Habit*. *Flowgraph* tersebut mempresentasikan alur eksekusi program berdasarkan hasil identifikasi *Statement* yang telah dilakukan sebelumnya. *Flowgraph* ini

menunjukkan jalur logika program mulai dari pengguna menekan *Checklist* hingga data log berhasil disimpan, diperbarui, atau terjadi error. Adapun penjelasan dari setiap node adalah sebagai berikut.

1. Node 1 terdiri dari S1

   Node ini merupakan proses awal pada fungsi Input *Progress Habit*, yaitu ketika pengguna menekan tombol Simpan *Progress* pada *habit* yang dipilih.

2. Node 2 terdiri dari S2

   Node ini merupakan percabangan untuk memeriksa apakah *ID Habit* tersedia. Jika *ID Habit* tidak tersedia, maka proses langsung menuju Node 13 dan selesai. Sebaliknya, apabila *ID Habit* tersedia, proses dilanjutkan ke Node 3.

3. Node 3 terdiri dari S3

   Node ini berisi proses pengecekan apakah pengguna telah menginput *progress* pada *habit* tersebut pada hari yang sama melalui fungsi *hasInputProgressToday*().

4. Node 4 terdiri dari S4

   Node ini merupakan percabangan untuk menentukan apakah *progress* pada hari tersebut sudah pernah diinput. Jika progress sudah pernah diinput, maka alur menuju Node 5. Sebaliknya, apabila belum pernah diinput, proses dilanjutkan ke Node 6.

5. Node 5 terdiri dari S5

   Node ini berisi proses menampilkan pesan "*Progress* sudah diinput hari ini. Silakan lanjutkan besok." menggunakan *SnackBar*. Setelah pesan ditampilkan, proses langsung menuju Node 13.

6. Node 6 terdiri dari S6

   Node ini merupakan proses mengambil nilai progress yang dimasukkan oleh pengguna melalui *TextField* kemudian mengubahnya menjadi tipe data *double* menggunakan fungsi *double.tryParse*().

7. Node 7 terdiri dari S7

   Node ini berisi proses menyimpan data progress habit ke dalam *database* menggunakan fungsi *saveHabitProgress***()**.

8. Node 8 terdiri dari S8

   Node ini merupakan percabangan untuk memeriksa apakah progress yang diinput telah mencapai target harian. Jika target tercapai, maka alur menuju Node 9. Sebaliknya, apabila target belum tercapai, maka alur menuju Node 12.
9. Node 9 terdiri dari S9

   Node ini berisi proses menambahkan *Diamond Badge* sebagai bentuk penghargaan kepada pengguna karena telah mencapai target harian.
10. Node 10 terdiri dari S10

    Node ini berisi proses menampilkan animasi *Flying Badge* sebagai bentuk apresiasi setelah pengguna berhasil memperoleh *Diamond Badge*.
11. Node 11 terdiri dari S11

    Node ini berisi proses menampilkan pesan "Selamat! Target hari ini berhasil dicapai." menggunakan *SnackBar* sebagai informasi bahwa target harian telah berhasil diselesaikan.
12. Node 12 terdiri dari S12

    Node ini berisi proses menampilkan pesan "*Progress* berhasil disimpan. Target hari ini belum tercapai." menggunakan *SnackBar* sebagai informasi bahwa data progress telah berhasil disimpan meskipun target harian belum tercapai.
13. Node 13 terdiri dari S13

    Node ini merupakan proses memanggil fungsi *_loadHabits*() untuk memuat ulang daftar habit sehingga perubahan progress yang telah dilakukan dapat langsung ditampilkan pada halaman aplikasi. Setelah proses tersebut selesai, fungsi *Input Progress Habit* berakhir.

E. *Cyclomatic Complexity*

Pada modul *Input Progress Habit* terdapat 4 predicate node utama, yaitu:

Tabel 5. 16 *Cyclomatic Complexity* Fungsi *Input Progress Habit*

No	Predicate Node	Kondisi
1	if (habit.id == null)	Mengecek apakah ID habit tersedia.
2	if (alreadyInput)	Mengecek apakah progress habit sudah diinput pada hari yang sama.
3	if (targetTercapai)	Mengecek apakah progress telah mencapai target harian.

Maka perhitungan *Cyclomatic Complexity* adalah:

V(G) = P + 1

V(G) = 3 + 1

V(G) = 4

Berdasarkan hasil perhitungan tersebut, nilai *Cyclomatic Complexity* pada modul *Input Progress Habit* adalah 4. Artinya, terdapat 4 jalur independen yang harus diuji untuk memastikan seluruh alur logika utama pada modul *Input Progress Habit* berjalan dengan baik.

F. *Independent Path*

Berdasarkan hasil perhitungan *Cyclomatic Complexity*, diperoleh 5 *Independent Path* sebagai berikut.

1. Path 1

   1 → 2 → 13

   (Menguji kondisi ketika ID *Habit* tidak tersedia, sehingga proses langsung dihentikan.)

2. Path 2

   1 → 2 → 3 → 4 → 5 → 13

   (Menguji kondisi ketika progress sudah pernah diinput pada hari yang sama, sehingga sistem menampilkan pesan bahwa progress telah diinput dan proses selesai.)

3. Path 3

   1 → 2 → 3 → 4 → 6 → 7 → 8 → 9 → 10 → 11 → 13

   (Menguji kondisi ketika progress belum pernah diinput dan target harian berhasil dicapai, sehingga sistem menambahkan *Diamond Badge*, menampilkan animasi *Flying Badge*)

4. Path 4

   1 → 2 → 3 → 4 → 6 → 7 → 8 → 12 → 13

   (Menguji kondisi ketika progress belum pernah diinput, namun target harian belum tercapai, sehingga sistem menampilkan pesan bahwa target belum tercapai kemudian memuat ulang data *habit*.)

G. *Test case*

Berikut hasil *test case* pada fungsi *Input Progress Habit* berdasarkan *Independent Path*.

Tabel 5. 17 *Test case* Fungsi *Input Progress Habit*

Test Case	Kondisi Pengujian	Input/Aksi	Output yang Diharapkan	Hasil
TC1	ID Habit tidak tersedia	Pengguna menekan tombol Simpan Progress pada data habit yang tidak memiliki ID Habit	Proses penyimpanan dihentikan dan tidak terjadi perubahan data.	Valid
TC2	Progress sudah pernah diinput pada hari yang sama	Pengguna kembali menginput progress pada habit yang sudah memiliki progress pada tanggal yang sama	Sistem menampilkan pesan "Progress untuk habit ini sudah diinput hari ini. Silakan lanjutkan besok." dan proses penyimpanan dibatalkan.	Valid
TC3	Progress berhasil disimpan dan target harian tercapai	Pengguna menginput nilai progress yang sama atau lebih besar dari target harian kemudian menekan tombol Simpan Progress	Progress berhasil disimpan, sistem menambahkan Diamond Badge, menampilkan animasi Flying Badge, menampilkan pesan "Selamat! Target hari ini berhasil dicapai.", kemudian memuat ulang data habit.	Valid
TC4	Progress berhasil disimpan tetapi target harian belum tercapai	Pengguna menginput nilai progress yang lebih kecil dari target harian kemudian menekan tombol Simpan Progress	Progress berhasil disimpan, sistem menampilkan pesan "Progress berhasil disimpan. Target hari ini belum tercapai.", kemudian memuat ulang data habit.	Valid

H. *Statement Coverage*

Statement Coverage digunakan untuk mengetahui apakah seluruh *Statement* pada kode sudah dijalankan oleh *test case*.

Rumus:

Statement Coverage = (Jumlah *Statement* yang dieksekusi / Jumlah seluruh *Statement*) × 100%

Tabel 5. 18 *Statement Covarege* Fungsi *Input Progress Habit*

Test Case	Statement yang Dieksekusi	Keterangan
TC1	S1, S2	Menguji kondisi ID Habit tidak tersedia, sehingga proses langsung dihentikan.
TC2	S1, S2, S3, S4, S5	Menguji kondisi progress sudah pernah diinput pada hari yang sama.
TC3	S1, S2, S3, S4, S6, S7, S8, S9, S10, S11, S13	Menguji kondisi progress berhasil disimpan dan target harian tercapai.

TC4	S1, S2, S3, S4, S6, S7, S8, S12, S13	Menguji kondisi progress berhasil disimpan tetapi target harian belum tercapai.

Berdasarkan tabel di atas, seluruh *Statement* dari S1 sampai S13 telah dieksekusi melalui *test case* yang dibuat. Dengan demikian, jumlah *Statement* yang dieksekusi adalah 13 dari total 13 *Statement*.

Maka perhitungan *Statement Coverage* adalah:

Statement Coverage = (Jumlah *Statement* yang dieksekusi / Jumlah seluruh *Statement*) × 100%

Statement Coverage = (13 / 13) × 100%

Statement Coverage = 100%

Berdasarkan hasil perhitungan tersebut, nilai *Statement Coverage* pada modul *Input Progress Habit* adalah 100%. Artinya, seluruh *Statement* utama pada modul *Input Progress Habit* telah diuji melalui *test case* yang dibuat.

5.4.5 Pengujian Modul Statistik *Habit*

A. Potongan *Source Code*

```
Future<void> loadData() async {
  setState(() {
    isLoading = true;
  });

  try {
    final data = await _service.getWeeklyCompletionCounts(
      weekStart: selectedWeekStart,
    );

    final sum = await _service.getStatisticsSummaryByWeek(
      weekStart: selectedWeekStart,
    );

    final trend = await _service.getWeeklyProgressTrend(
      numberOfWeeks: 6,
    );

    if (!mounted) return;
```

```
      setState(() {
        weeklyData = data;
        summary = sum;
        weeklyTrendData = trend;
        isLoading = false;
      });
    } catch (e) {
      if (!mounted) return;

      setState(() {
        isLoading = false;
      });

      ScaffoldMessenger.of(context).showSnackBar(
        SnackBar(content: Text('Gagal memuat statistik: $e')),
      );
    }
  }

  if (weeklyTrendData.isEmpty || isWeeklyTrendAllZero)
    Expanded(
      child: _emptyChartMessage(
        'Belum cukup data mingguan.',
      ),
    )
  else
    Expanded(
      child: CustomPaint(
        painter: WeeklyLineDiagramPainter(weeklyTrendData),
      ),
    )
```

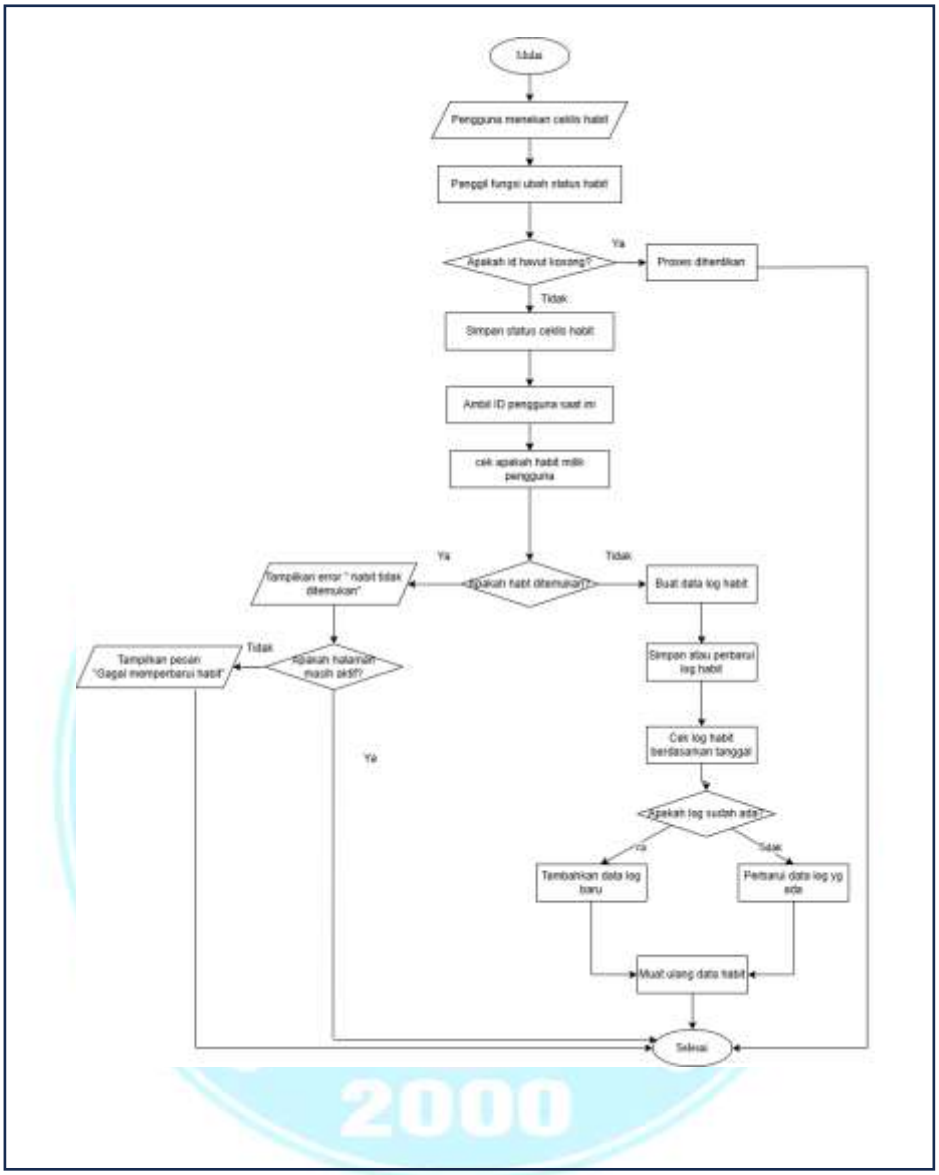
B. Identifikasi *Statement*

Tabel 5. 19 Identifikasi *Statement* Fungsi Statistik

Kode	*Statement*	Keterangan
S1	selectedWeekStart = ...	Menentukan awal minggu yang dipilih
S2	loadData()	Memanggil fungsi untuk memuat data statistik
S3	setState(() { isLoading = true; })	Menampilkan proses loading
S4	getWeeklyCompletionCounts(...)	Mengambil data jumlah *Checklist* harian

S5	getStatisticsSummaryByWeek(...)	Mengambil data ringkasan statistik mingguan
S6	getWeeklyProgressTrend(...)	Mengambil data tren progress mingguan
S7	if (!mounted) return	Mengecek apakah halaman masih aktif setelah data berhasil dimuat
S8	setState(() { weeklyData = data; ... })	Menyimpan data statistik ke state
S9	selectedTab == 0	Mengecek tab yang sedang dipilih
S10	if (weeklyData.isEmpty)	Mengecek apakah data harian kosong
S11	_emptyChartMessage('Belum ada data harian')	Menampilkan pesan data harian kosong
S12	DailyLineDiagramPainter(...)	Menampilkan diagram garis harian
S13	`if (weeklyTrendData.isEmpty	
S14	_emptyChartMessage('Belum cukup data mingguan')	Menampilkan pesan data mingguan kosong
S15	WeeklyLineDiagramPainter(...)	Menampilkan diagram garis mingguan
S16	catch (e)	Menangkap error saat data statistik gagal dimuat
S17	if (!mounted) return	Mengecek apakah halaman masih aktif saat terjadi error
S18	showSnackBar('Gagal memuat statistik')	Menampilkan pesan gagal memuat statistik

C. *Flowchart*



Gambar 5. 35 *Flowchart* fungsi Statistik *Habit*

Gambar di atas menjelaskan *Flowchart* dari fungsi Statistik *Habit*. *Flowchart* tersebut menggambarkan alur proses ketika pengguna membuka halaman statistik, kemudian sistem memuat data *Checklist* harian, ringkasan statistik mingguan, dan tren progress mingguan. Setelah data berhasil dimuat, sistem akan menentukan tampilan statistik berdasarkan tab yang dipilih, yaitu statistik harian atau statistik mingguan. Apabila data belum tersedia, sistem akan menampilkan pesan kosong, sedangkan jika terjadi kesalahan saat memuat data, sistem akan menampilkan pesan gagal memuat statistik. Adapun penjelasannya sebagai berikut.

1. S1 (Menentukan awal minggu yang dipilih)

   Pada tahap ini, sistem menentukan nilai selectedWeekStart, yaitu tanggal awal minggu yang sedang dipilih pengguna. Data ini digunakan sebagai dasar untuk mengambil statistik *Habit* pada minggu tertentu.
2. S2 (Memanggil fungsi loadData)

   Setelah awal minggu ditentukan, sistem memanggil fungsi loadData() untuk memuat seluruh data statistik yang dibutuhkan pada halaman Statistik *Habit*.
3. S3 (Menampilkan proses loading)

   Pada tahap ini, sistem mengubah nilai isLoading = true menggunakan setState(). Tujuannya untuk menampilkan bahwa data statistik sedang diproses atau sedang dimuat.
4. S4 (Mengambil data jumlah *Checklist* harian)

   Sistem mengambil data jumlah *Checklist* harian melalui fungsi getWeeklyCompletionCounts(...). Data ini digunakan untuk menampilkan grafik atau diagram statistik harian.
5. S5 (Mengambil data ringkasan statistik mingguan)

   Pada tahap ini, sistem mengambil data ringkasan statistik dalam satu minggu melalui fungsi getStatisticsSummaryByWeek(...). Ringkasan ini dapat berupa jumlah *Habit* selesai, progress, atau informasi statistik lain yang ditampilkan pada halaman statistik.
6. S6 (Mengambil data tren progress mingguan)

   Sistem mengambil data tren progress mingguan melalui fungsi getWeeklyProgressTrend(...). Data ini digunakan untuk menampilkan perkembangan *Habit* pengguna dalam beberapa minggu terakhir.
7. S7 (Mengecek apakah halaman masih aktif setelah data berhasil dimuat)

   Setelah data berhasil diambil, sistem mengecek apakah halaman statistik masih aktif menggunakan if (!mounted) return. Jika halaman sudah tidak aktif, proses dihentikan agar tidak terjadi error saat memperbarui tampilan.
8. S8 (Menyimpan data statistik ke state)

   Jika halaman masih aktif, sistem menyimpan data statistik ke dalam state, seperti weeklyData, summary, dan weeklyTrendData. Setelah itu nilai isLoading diubah menjadi false agar proses loading berhenti.

9. S9 (Mengecek tab yang sedang dipilih)

   Pada tahap ini, sistem mengecek tab yang sedang dipilih oleh pengguna. Jika pengguna memilih tab harian, maka sistem akan memproses data statistik harian. Jika pengguna memilih tab mingguan, maka sistem akan memproses data statistik mingguan.

10. S10 (Mengecek apakah data harian kosong)

    Jika tab harian dipilih, sistem mengecek apakah data harian pada weeklyData masih kosong. Jika data kosong, maka alur menuju ke S11. Jika data tersedia, maka alur menuju ke S12.

11. S11 (Menampilkan pesan data harian kosong)

    Jika data harian belum tersedia, sistem menampilkan pesan "Belum ada data harian". Pesan ini muncul agar pengguna mengetahui bahwa belum ada *Checklist Habit* yang dapat ditampilkan pada grafik harian.

12. S12 (Menampilkan diagram garis harian)

    Jika data harian tersedia, sistem menampilkan diagram garis harian menggunakan DailyLineDiagramPainter(...). Diagram ini menunjukkan perkembangan *Checklist Habit* pengguna berdasarkan data harian.

13. S13 (Mengecek apakah data mingguan kosong)

    Jika tab mingguan dipilih, sistem mengecek apakah weeklyTrendData masih kosong atau seluruh progress mingguan masih bernilai nol. Jika data mingguan kosong, maka alur menuju ke **S14**. Jika data tersedia, maka alur menuju ke **S15**.

14. S14 (Menampilkan pesan data mingguan kosong)

    Jika data mingguan belum cukup, sistem menampilkan pesan "Belum cukup data mingguan". Pesan ini menunjukkan bahwa data progress mingguan belum dapat divisualisasikan dalam bentuk grafik.

15. S15 (Menampilkan diagram garis mingguan)

    Jika data mingguan tersedia, sistem menampilkan diagram garis mingguan menggunakan WeeklyLineDiagramPainter(...). Diagram ini digunakan untuk melihat tren perkembangan *Habit* pengguna dari minggu ke minggu.

16. S16 (Menangkap error saat data statistik gagal dimuat)

Apabila terjadi kesalahan saat proses pengambilan data statistik, sistem akan masuk ke bagian catch (e). Bagian ini digunakan untuk menangkap error yang muncul selama proses pemuatan data.

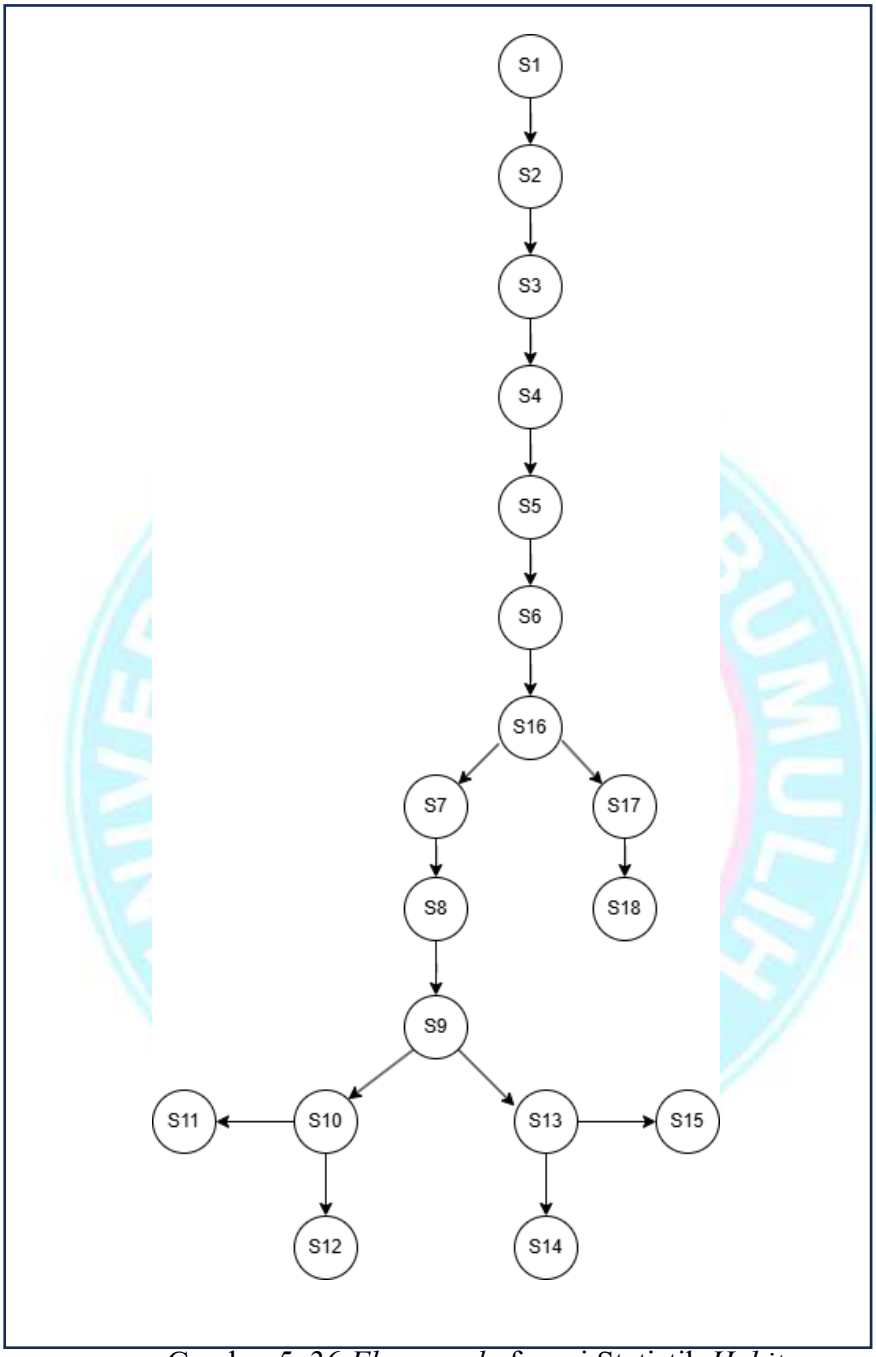
17. S17 (Mengecek apakah halaman masih aktif saat terjadi error)

    Saat terjadi error, sistem kembali mengecek apakah halaman masih aktif menggunakan if (!mounted) return. Jika halaman sudah tidak aktif, maka proses dihentikan tanpa menampilkan pesan error.

18. S18 (Menampilkan pesan gagal memuat statistik)

    Jika halaman masih aktif saat error terjadi, sistem menampilkan pesan “Gagal memuat statistik” menggunakan SnackBar. Pesan ini memberi informasi kepada pengguna bahwa data statistik tidak berhasil dimuat.

D. *Flowgraph*



Gambar 5. 36 *Flowgraph* fungsi Statistik *Habit*

Gambar di atas menjelaskan *Flowgraph* dari fungsi Statistik *Habit*. *Flowgraph* tersebut mempresentasikan alur eksekusi program berdasarkan hasil identifikasi *Statement* pada fungsi statistik. *Flowgraph* ini menunjukkan jalur program mulai dari pemilihan minggu, proses pemuatan data, pengecekan kondisi halaman, pemilihan tab statistik, hingga penanganan apabila data kosong atau terjadi error. Adapun penjelasan dari setiap node adalah sebagai berikut.

1. Node 1 terdiri dari S1 dan S2

   Node ini merupakan proses awal pada fungsi statistik, yaitu sistem menentukan awal minggu yang dipilih melalui selectedWeekStart, kemudian memanggil fungsi loadData() untuk memuat data statistik.
2. Node 2 terdiri dari S3

   Node ini berisi proses menampilkan loading dengan mengubah nilai isLoading = true. Node ini menunjukkan bahwa sistem sedang memuat data statistik.
3. Node 3 terdiri dari S4, S5, dan S6

   Node ini merupakan proses pengambilan data statistik dari service. Sistem mengambil data jumlah *Checklist* harian, ringkasan statistik mingguan, dan tren progress mingguan.
4. Node 4 terdiri dari S7

   Node ini merupakan percabangan untuk mengecek apakah halaman masih aktif setelah data berhasil dimuat. Jika halaman tidak aktif, maka proses dihentikan. Jika halaman masih aktif, maka proses dilanjutkan.
5. Node 5 terdiri dari S8

   Node ini berisi proses menyimpan hasil data statistik ke dalam state, yaitu weeklyData, summary, dan weeklyTrendData. Pada tahap ini loading juga dihentikan dengan mengubah isLoading = false.
6. Node 6 terdiri dari S9

   Node ini merupakan percabangan untuk mengecek tab yang sedang dipilih oleh pengguna. Jika pengguna memilih tab harian, maka sistem menuju alur statistik harian. Jika pengguna memilih tab mingguan, maka sistem menuju alur statistik mingguan.
7. Node 7 terdiri dari S10

   Node ini merupakan percabangan pada tab harian untuk mengecek apakah data harian kosong. Jika data harian kosong, maka alur menuju node yang menampilkan pesan kosong. Jika data tersedia, maka alur menuju node yang menampilkan diagram harian.
8. Node 8 terdiri dari S11

Node ini berisi proses menampilkan pesan “Belum ada data harian”. Node ini dijalankan apabila pengguna membuka tab harian tetapi belum ada data *Checklist Habit* yang tersedia.

9. Node 9 terdiri dari S12

   Node ini berisi proses menampilkan diagram garis harian menggunakan DailyLineDiagramPainter(...). Node ini dijalankan apabila data harian tersedia.

10. Node 10 terdiri dari S13

    Node ini merupakan percabangan pada tab mingguan untuk mengecek apakah data mingguan kosong atau seluruh progress masih nol. Jika data belum cukup, maka alur menuju node yang menampilkan pesan kosong. Jika data tersedia, maka alur menuju node yang menampilkan diagram mingguan.

11. Node 11 terdiri dari S14

    Node ini berisi proses menampilkan pesan “Belum cukup data mingguan”. Node ini dijalankan apabila data tren mingguan belum tersedia atau belum cukup untuk divisualisasikan.

12. Node 12 terdiri dari S15

    Node ini berisi proses menampilkan diagram garis mingguan menggunakan WeeklyLineDiagramPainter(...). Node ini dijalankan apabila data tren mingguan tersedia.

13. Node 13 terdiri dari S16

    Node ini merupakan jalur penanganan error. Jika proses pengambilan data statistik gagal, sistem akan masuk ke blok catch (e) untuk menangkap kesalahan yang terjadi.

14. Node 14 terdiri dari S17

    Node ini merupakan percabangan untuk mengecek apakah halaman masih aktif saat terjadi error. Jika halaman tidak aktif, maka proses dihentikan. Jika halaman masih aktif, maka alur dilanjutkan untuk menampilkan pesan error.

15. Node 15 terdiri dari S18

    Node ini berisi proses menampilkan pesan “Gagal memuat statistik” menggunakan SnackBar. Node ini dijalankan apabila terjadi error saat memuat data statistik dan halaman masih aktif.

E. *Cyclomatic Complexity*

Pada modul statistik *Habit* terdapat 6 predicate/branch node utama, yaitu:

Tabel 5. 20 *Cyclomatic Complexity* Fungsi Statistik

No	Predicate / Branch Node	Kondisi
1	Proses pengambilan data	Mengecek apakah data berhasil dimuat atau terjadi error
2	if (!mounted) return	Mengecek apakah halaman masih aktif setelah data berhasil dimuat
3	selectedTab == 0	Mengecek apakah pengguna memilih tab harian atau mingguan
4	if (weeklyData.isEmpty)	Mengecek apakah data harian kosong
5	if (weeklyTrendData.isEmpty	Mengecek apakah data harian kosong
6	if (!mounted) return	Mengecek apakah halaman masih aktif saat terjadi error

Maka perhitungan *Cyclomatic Complexity* adalah:

V(G) = P + 1

V(G) = 6 + 1

V(G) = 7

Berdasarkan hasil perhitungan tersebut, nilai *Cyclomatic Complexity* pada modul statistik *Habit* adalah 7. Artinya, terdapat 7 jalur independen yang harus diuji untuk memastikan seluruh alur logika utama pada modul statistik *Habit* berjalan dengan baik.

F. *Independent Path*

Berdasarkan hasil perhitungan *Cyclomatic Complexity*, diperoleh 7 *Independent Path* sebagai berikut.

1. Path 1

   1 → 2 → 3 → 15

   (Data berhasil dimuat, tetapi halaman sudah tidak aktif sehingga proses dihentikan)

2. Path 2

   1 → 2 → 4 → 5 → 6 → 7 → 14

   (Data berhasil dimuat, tab harian dipilih, tetapi data harian kosong)

3. Path 3

1 → 2 → 3 → 4 → 5 → 6 → 8 → 15

(Data berhasil dimuat, tab harian dipilih, dan data harian tersedia sehingga diagram harian ditampilkan)

4. Path 4

   1 → 2 → 3 → 4 → 5 → 9 → 10 → 15

   (Data berhasil dimuat, tab mingguan dipilih, tetapi data mingguan kosong atau seluruh progress masih nol)

5. Path 5

   1 → 2 → 3 → 4 → 5 → 9 → 11 → 15

   (Data berhasil dimuat, tab mingguan dipilih, dan data mingguan tersedia sehingga diagram mingguan ditampilkan)

6. Path 6

   1 → 2 → 12 → 13 → 15

   (Terjadi error saat memuat data statistik dan halaman sudah tidak aktif)

7. Path 7

   1 → 2 → 12 → 13 → 14 → 15

   (Terjadi error saat memuat data statistik, halaman masih aktif, lalu sistem menampilkan pesan gagal memuat statistik.

G. *Test case*

Berikut hasil *test case* pada modul statistik *Habit* berdasarkan *Independent Path*.

Tabel 5. 21 *Test case* Fungsi Statistik

Test case	Kondisi Pengujian	Input / Aksi	Output yang Diharapkan	Hasil
TC1	Halaman tidak aktif setelah data dimuat	Sistem berhasil mengambil data, tetapi halaman sudah tidak aktif	Proses dihentikan dan tidak memperbarui tampilan	Valid
TC2	Tab harian dipilih dan data harian kosong	Pengguna membuka statistik harian tanpa *Checklist Habit*	Muncul pesan "Belum ada data harian"	Valid
TC3	Tab harian dipilih dan data tersedia	Pengguna membuka statistik harian setelah melakukan *Checklist Habit*	Diagram garis harian tampil	Valid
TC4	Tab mingguan dipilih dan data mingguan kosong	Pengguna membuka statistik mingguan tanpa progress mingguan	Muncul pesan "Belum cukup data mingguan"	Valid

TC5	Tab mingguan dipilih dan data tersedia	Pengguna membuka statistik mingguan setelah beberapa data progress tersedia	Diagram garis mingguan tampil	Valid
TC6	Error saat memuat data dan halaman tidak aktif	Service gagal memuat data ketika halaman sudah tidak aktif	Proses dihentikan tanpa menampilkan pesan	Valid
TC7	Error saat memuat data dan halaman aktif	Service gagal memuat data ketika halaman masih aktif	Muncul pesan "Gagal memuat statistik"	Valid

H. *Statement Coverage*

Statement Coverage digunakan untuk mengetahui apakah seluruh *Statement* pada kode sudah dijalankan oleh *test case*.

Rumus:

Statement Coverage = (Jumlah *Statement* yang dieksekusi / Jumlah seluruh *Statement*) × 100%

Tabel 5. 22 *Statement Covarege* Fungsi Statistik

Test case	***Statement* yang Dieksekusi**	**Keterangan**
TC1	S1, S2, S3, S4, S5, S6, S7	Menguji kondisi data berhasil dimuat tetapi halaman tidak aktif
TC2	S1, S2, S3, S4, S5, S6, S7, S8, S9, S10, S11	Menguji kondisi tab harian dengan data kosong
TC3	S1, S2, S3, S4, S5, S6, S7, S8, S9, S10, S12	Menguji kondisi tab harian dengan data tersedia
TC4	S1, S2, S3, S4, S5, S6, S7, S8, S9, S13, S14	Menguji kondisi tab mingguan dengan data kosong
TC5	S1, S2, S3, S4, S5, S6, S7, S8, S9, S13, S15	Menguji kondisi tab mingguan dengan data tersedia
TC6	S1, S2, S3, S16, S17	Menguji kondisi error dan halaman tidak aktif
TC7	S1, S2, S3, S16, S17, S18	Menguji kondisi error dan pesan gagal ditampilkan

Berdasarkan tabel di atas, seluruh *Statement* dari S1 sampai S18 telah dieksekusi melalui *test case* yang dibuat. Dengan demikian, jumlah *Statement* yang dieksekusi adalah 18 dari total 18 *Statement*.

Maka perhitungan *Statement Coverage* adalah:

Statement Coverage = (Jumlah *Statement* yang dieksekusi / Jumlah seluruh *Statement*) × 100%

Statement Coverage = (18 / 18) × 100%

Statement Coverage = 100%

Berdasarkan hasil perhitungan tersebut, nilai *Statement Coverage* pada modul statistik *Habit* adalah 100%. Artinya, seluruh *Statement* utama pada modul statistik *Habit* telah diuji melalui *test case* yang dibuat.

5.4.6 Pengujian Modul *Random Forest AI*

A. Potongan *Source Code*

```
Future<void> loadRecommendations() async {

  setState(() {
    isProcessing = true;
    showResults = false;
  });

  final result =
      await _habitService.getHabitRecommendations();

  if (!mounted) return;

  predictions = result;

  setState(() {
    isProcessing = false;
    hasAnalyzed = true;
  });
}

...

onTap: () async {

  final canAnalyze =
      demoMode
          ? true
          : (pred["canAnalyze"] ?? false);

  if (!canAnalyze && !demoMode) {
```

```
    final age = pred["habitAgeDays"] ?? 0;

    final remaining = 7 - age;

    showDialog(
      context: context,
      builder: (_) => AlertDialog(
        title: const Text("AI Belum Tersedia"),
        content: Text(
          "Habit ini baru berjalan selama $age hari.\n"
          "Analisis AI akan tersedia dalam "
          "$remaining hari lagi.",
        ),
      ),
    );

    return;
  }

  await startPredictionAnimation(index);

}
```

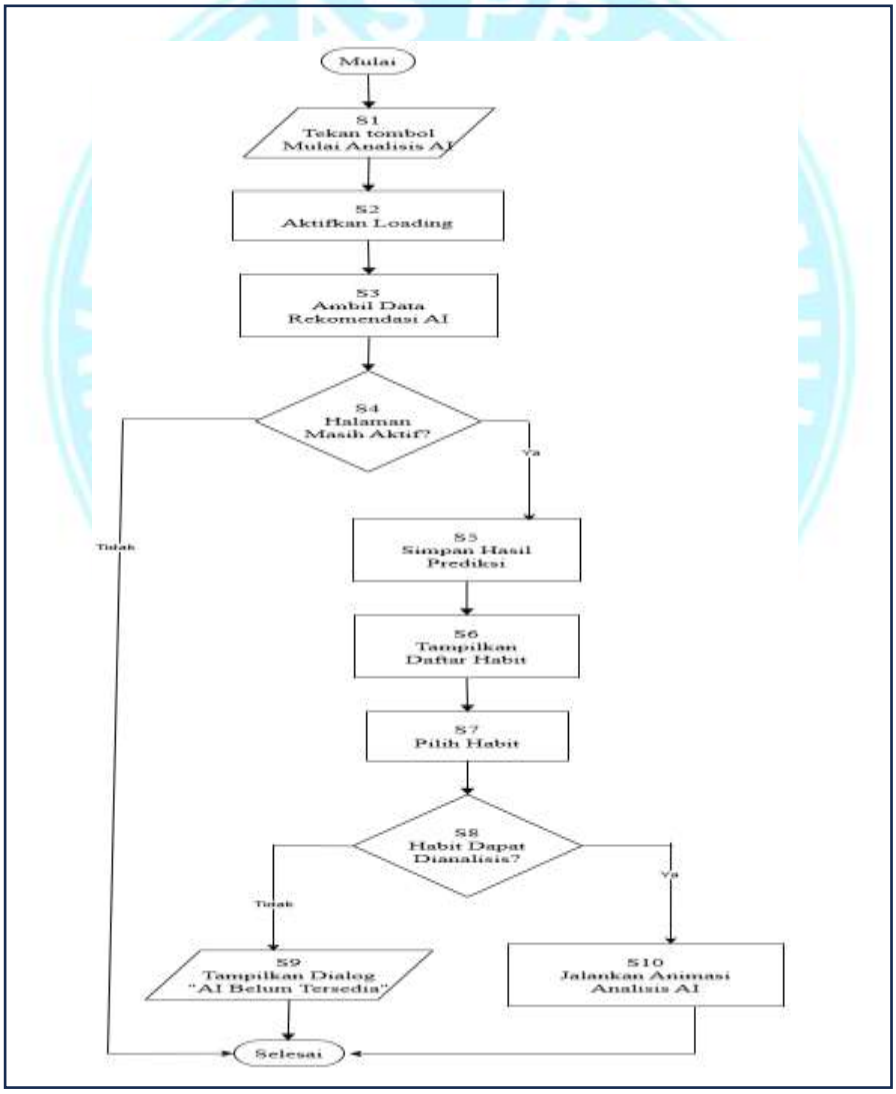
B. Identifikasi *Statement*

Tabel 5. 23 Identifikasi *Statement* Fungsi Rekomendasi *Random Forest*

Kode	Statement	Keterangan
S1	loadRecommendations()	Menjalankan proses analisis AI ketika pengguna menekan tombol Mulai Analisis AI.
S2	setState(() { isProcessing = true; showResults = false; })	Mengaktifkan status loading dan menyembunyikan hasil analisis sebelumnya.
S3	getHabitRecommendations()	Mengambil hasil rekomendasi habit menggunakan algoritma Random Forest.
S4	if (!mounted) return	Mengecek apakah halaman masih aktif sebelum memperbarui tampilan.
S5	predictions = result	Menyimpan hasil rekomendasi AI ke dalam daftar prediksi.
S6	setState(() { isProcessing = false; hasAnalyzed = true; })	Menonaktifkan loading dan menampilkan daftar habit yang siap dianalisis.

S7	final canAnalyze = demoMode ? true : (pred["canAnalyze"] ?? false)	Menentukan apakah habit sudah memenuhi syarat untuk dianalisis AI.
S8	if (!canAnalyze && !demoMode)	Mengecek apakah habit belum dapat dianalisis.
S9	showDialog(AlertDialog(...))	Menampilkan dialog "AI Belum Tersedia" beserta informasi sisa hari analisis.
S10	startPredictionAnimation(index)	Menjalankan animasi proses analisis AI sebelum hasil rekomendasi ditampilkan.

B. *Flowchart*



Gambar 5. 37 *Flowchart* fungsi Rekomendasi *Random Forest*

Gambar di atas menjelaskan *Flowchart* dari fungsi Rekomendasi *Random Forest AI. Flowchart* tersebut menggambarkan alur proses mulai dari pelatihan

model *Random Forest* menggunakan Python, proses ekspor beberapa decision *tree* ke dalam kode *Dart*, pengambilan data *Habit* pengguna, perhitungan fitur, proses voting *Random Forest offline*, perhitungan skor konsistensi, penentuan status rekomendasi, hingga hasil analisis ditampilkan kepada pengguna. Adapun penjelasannya sebagai berikut.

1. S1 (Menekan Tombol Mulai Analisis AI)

   Pada tahap ini, pengguna menekan tombol Mulai Analisis *AI* untuk memulai proses analisis habit menggunakan algoritma *Random Forest*.

2. S2 (Mengaktifkan Status *Loading*)

   Setelah tombol ditekan, sistem mengaktifkan indikator *loading* dengan mengubah nilai *isProcessing* = true dan menyembunyikan hasil analisis sebelumnya.

3. S3 (Mengambil Data Rekomendasi AI)

   Sistem memanggil fungsi getHabitRecommendations() untuk mengambil hasil rekomendasi habit berdasarkan algoritma Random Forest.

4. S4 (Mengecek Halaman Masih Aktif)

   Setelah proses pengambilan data selesai, sistem memeriksa apakah halaman masih aktif menggunakan if (!mounted). Jika halaman sudah tidak aktif, maka proses dihentikan. Sebaliknya, apabila halaman masih aktif, proses dilanjutkan ke tahap berikutnya.

5. S5 (Menyimpan Hasil Prediksi)

   Sistem menyimpan hasil rekomendasi yang diperoleh ke dalam variabel predictions untuk digunakan pada proses analisis berikutnya.

6. S6 (Menampilkan Daftar Habit)

   Sistem menonaktifkan indikator loading, mengubah status analisis menjadi selesai, kemudian menampilkan daftar habit yang dapat dipilih untuk dianalisis.

7. S7 (Mengecek Habit Dapat Dianalisis)

   Setelah pengguna memilih salah satu habit, sistem memeriksa apakah habit tersebut telah memenuhi syarat untuk dianalisis oleh *AI* melalui nilai *canAnalyze*.

8. S8 (Habit Sudah Memenuhi Syarat?)

   Sistem menentukan apakah habit telah memenuhi syarat analisis. Jika belum memenuhi syarat, maka proses menuju ke S9. Sebaliknya, apabila habit telah memenuhi syarat, proses dilanjutkan ke S10.

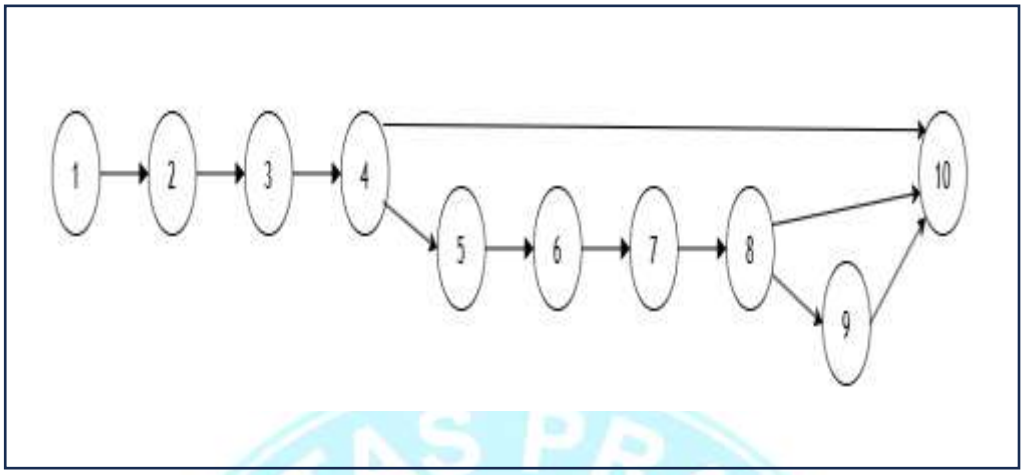
9. S9 (Menampilkan Dialog *AI* Belum Tersedia)

   Apabila habit belum memenuhi syarat analisis, sistem menampilkan dialog "*AI* Belum Tersedia" yang berisi informasi jumlah hari yang masih diperlukan sebelum analisis *AI* dapat dilakukan. Setelah dialog ditampilkan, proses selesai.

10. S10 (Menjalankan Animasi Analisis *AI*)

    Apabila habit telah memenuhi syarat analisis, sistem menjalankan fungsi startPredictionAnimation() yang menampilkan animasi proses analisis *AI* sebelum hasil rekomendasi ditampilkan kepada pengguna. Setelah animasi selesai, hasil rekomendasi *AI* ditampilkan dan proses berakhir.

C. *Flowgraph*



Gambar 5. 38 *Flowgraph* fungsi Rekomendasi *Random Forest*

Gambar di atas menjelaskan *Flowgraph* dari fungsi Rekomendasi *Random Forest AI. Flowgraph* tersebut mempresentasikan alur eksekusi program berdasarkan hasil identifikasi *Statement* yang telah dilakukan sebelumnya. *Flowgraph* ini menunjukkan jalur proses mulai dari *Training* model, ekspor aturan *Random Forest* ke *Dart*, pemanggilan proses analisis, pengambilan data *Habit*, perhitungan fitur, voting *tree* , penentuan skor konsistensi, hingga hasil rekomendasi ditampilkan kepada pengguna. Adapun penjelasan dari setiap node adalah sebagai berikut.

1. Node 1 terdiri dari S1
   Node ini merupakan proses awal ketika pengguna menekan tombol Mulai Analisis *AI* untuk memulai proses analisis habit menggunakan algoritma *Random Forest*.
2. Node 2 terdiri dari S2
   Node ini merupakan proses mengaktifkan status loading dengan mengubah nilai isProcessing menjadi true serta menyembunyikan hasil analisis sebelumnya agar sistem dapat memproses data rekomendasi.
3. Node 3 terdiri dari S3
   Node ini merupakan proses pemanggilan fungsi *getHabitRecommendations()* untuk mengambil data rekomendasi habit yang telah dihitung menggunakan algoritma *Random Forest*.

4. Node 4 terdiri dari S4
   Node ini merupakan *predicate node* yang digunakan untuk memeriksa apakah halaman masih aktif (*mounted*). Apabila halaman sudah tidak aktif, proses dihentikan. Sebaliknya, apabila halaman masih aktif, proses dilanjutkan ke tahap berikutnya.
5. Node 5 terdiri dari S5
   Node ini merupakan proses menyimpan hasil rekomendasi *AI* ke dalam variabel predictions sehingga hasil analisis dapat digunakan pada proses berikutnya.
6. Node 6 terdiri dari S6
   Node ini merupakan proses menonaktifkan status *loading* dan mengubah status analisis menjadi selesai sehingga daftar habit yang dapat dianalisis ditampilkan kepada pengguna.
7. Node 7 terdiri dari S7
   Node ini merupakan proses ketika pengguna memilih salah satu *habit*, kemudian sistem menentukan apakah habit tersebut telah memenuhi syarat untuk dilakukan analisis *AI*.
8. Node 8 terdiri dari S8
   Node ini merupakan *predicate node* yang digunakan untuk memeriksa apakah habit dapat dianalisis menggunakan *AI*. Apabila habit belum memenuhi syarat analisis, sistem akan menuju proses penampilan informasi bahwa *AI* belum tersedia. Sebaliknya, apabila habit telah memenuhi syarat, sistem akan melanjutkan proses analisis *AI*.
9. Node 9 terdiri dari S9
   Node ini merupakan proses menampilkan dialog "*AI* Belum Tersedia" yang berisi informasi mengenai usia habit serta jumlah hari yang masih diperlukan sebelum habit dapat dianalisis menggunakan *AI*.
10. Node 10 terdiri dari S10
    Node ini merupakan proses menjalankan fungsi *startPredictionAnimation*() untuk menampilkan animasi proses analisis *AI*, kemudian sistem menampilkan hasil rekomendasi berdasarkan analisis algoritma *Random Forest*. Node ini juga merupakan node akhir, yang menunjukkan bahwa proses pengujian pada modul Rekomendasi *AI* telah selesai, baik setelah analisis berhasil dilakukan maupun setelah sistem menampilkan informasi bahwa habit belum memenuhi syarat untuk dianalisis.

D. *Cyclomatic Complexity*

Pada modul Rekomendasi *Habit AI Random Forest* terdapat 7 predicate node utama, yaitu:

Tabel 5. 24 *Cyclomatic Complexity* Fungsi Rekomendasi *Random Forest*

No	Predicate Node	Kondisi
1	if (!mounted)	Mengecek apakah halaman masih aktif setelah proses pengambilan rekomendasi selesai. Jika halaman tidak aktif maka proses dihentikan.
2	if (!canAnalyze && !demoMode)	Mengecek apakah habit telah memenuhi syarat untuk dianalisis menggunakan AI. Jika belum memenuhi syarat, sistem menampilkan dialog AI Belum Tersedia.

Maka perhitungan *Cyclomatic Complexity* adalah:

V(G) = P + 1
V(G) = 2 + 1
V(G) = 3

Berdasarkan hasil perhitungan tersebut, nilai *Cyclomatic Complexity* pada modul Rekomendasi *AI* adalah 3. Artinya, terdapat 3 jalur independen yang perlu diuji untuk memastikan seluruh logika utama pada fitur Rekomendasi *AI* berjalan dengan baik.

E. *Independent Path*

Berdasarkan hasil perhitungan *Cyclomatic Complexity*, diperoleh 8 *Independent Path* sebagai berikut.

1. Path 1

   1 → 2 → 3 → 4 → 10

   (Jalur ini menggambarkan kondisi ketika halaman sudah tidak aktif (*!mounted*), sehingga proses analisis dihentikan dan sistem langsung mengakhiri proses tanpa menampilkan hasil rekomendasi.)

2. Path 2

   1 → 2 → 3 → 4 → 5 → 6 → 7 → 8 → 9 → 10

(Jalur ini menggambarkan kondisi ketika halaman masih aktif, namun habit belum memenuhi syarat untuk dianalisis menggunakan *AI*. Sistem menampilkan dialog "*AI* Belum Tersedia", kemudian proses berakhir.)

3. Path 3

   1 → 2 → 3 → 4 → 5 → 6 → 7 → 8 → 10

   (Jalur ini menggambarkan kondisi ketika halaman masih aktif dan habit telah memenuhi syarat untuk dianalisis. Sistem menjalankan proses analisis *AI* melalui fungsi *startPredictionAnimation*(), kemudian menampilkan hasil rekomendasi kepada pengguna dan proses selesai.)

F. *Test case*

Berikut hasil *test case* pada modul Rekomendasi *Habit AI Random Forest* berdasarkan *Independent Path*.

Tabel 5. 25 *Test case* Fungsi Rekomendasi *Random Forest*

Test case	**Kondisi Pengujian**	**Input / Aksi**	**Output yang Diharapkan**	**Hasil**
TC1	Halaman tidak aktif setelah proses memuat rekomendasi	Pengguna menekan tombol Mulai Analisis AI, kemudian halaman sudah tidak aktif (!mounted)	Proses dihentikan dan tampilan tidak diperbarui	Valid
TC2	Habit belum memenuhi syarat analisis AI	Pengguna memilih habit yang belum dapat dianalisis (canAnalyze = false)	Sistem menampilkan dialog "AI Belum Tersedia" beserta informasi sisa hari analisis	Valid
TC3	Habit memenuhi syarat analisis AI	Pengguna memilih habit yang dapat dianalisis (canAnalyze = true)	Sistem menjalankan animasi analisis AI dan menampilkan hasil rekomendasi	Valid
TC4	Data rekomendasi berhasil dimuat	Pengguna menekan tombol Mulai Analisis AI dan data rekomendasi berhasil diperoleh	Sistem menampilkan daftar habit yang dapat dipilih untuk dianalisis	Valid
TC5	Data rekomendasi tidak tersedia	Pengguna menekan tombol Mulai Analisis AI, namun tidak terdapat habit yang dapat dianalisis	Sistem menampilkan informasi "Belum ada habit untuk dianalisis"	Valid

G. *Statement Coverage*

Statement Coverage digunakan untuk mengetahui apakah seluruh *Statement* pada kode sudah dijalankan oleh *test case*.

Tabel 5. 26 *Statement Covarege* Fungsi Rekomendasi *Random Forest*

Test Case	**Statement yang Dieksekusi**	**Keterangan**
TC1	S1, S2, S3, S4, S10	Menguji kondisi halaman tidak aktif (!mounted) sehingga proses dihentikan.
TC2	S1, S2, S3, S4, S5, S6, S7, S8, S9, S10	Menguji kondisi habit belum memenuhi syarat analisis AI sehingga sistem menampilkan dialog AI Belum Tersedia.
TC3	S1, S2, S3, S4, S5, S6, S7, S8, S10	Menguji kondisi habit memenuhi syarat analisis AI sehingga sistem menjalankan proses analisis dan menampilkan hasil rekomendasi.
TC4	S1, S2, S3, S4, S5, S6	Menguji kondisi data rekomendasi berhasil dimuat sehingga daftar habit ditampilkan.
TC5	S1, S2, S3, S4, S5, S6	Menguji kondisi tidak terdapat habit yang dapat dianalisis sehingga sistem menampilkan informasi bahwa belum ada habit untuk dianalisis.

Berdasarkan tabel di atas, seluruh *Statement* dari S1 sampai S10 telah dieksekusi melalui *test case* yang dibuat. Dengan demikian, jumlah *Statement* yang dieksekusi adalah 46 dari total 46 *Statement*.

Maka perhitungan *Statement Coverage* adalah:

Statement Coverage = (Jumlah *Statement* yang dieksekusi / Jumlah seluruh *Statement*) × 100%

Statement Coverage = (10 / 10) × 100%

Statement Coverage = 100%

Berdasarkan hasil perhitungan tersebut, nilai *Statement Coverage* pada modul *Random Forest AI* adalah 100%. Artinya, seluruh *Statement* utama pada modul *Random Forest AI* telah diuji melalui *test case* yang dibuat. Pengujian ini menunjukkan bahwa fitur *Random Forest AI* dapat mengambil data *Habit*, menghitung fitur prediksi, menjalankan aturan *Random Forest offline*, menentukan

status rekomendasi, serta menampilkan hasil analisis sesuai dengan alur yang dirancang.

5.5 Pembahasan

Berdasarkan hasil implementasi yang telah dilakukan, aplikasi *Habit Tracker* berhasil dibangun sebagai aplikasi berbasis Android menggunakan *Flutter* dan bahasa pemrograman *Dart*. Aplikasi ini dikembangkan sebagai alat bantu bagi mahasiswa dalam mencatat, memantau, dan mengevaluasi pembentukan kebiasaan positif secara lebih terstruktur. Pembahasan pada bab ini difokuskan pada hasil implementasi sistem dan pengujian teknis aplikasi.

Implementasi aplikasi didukung oleh perangkat keras berupa laptop sebagai perangkat utama pengembangan dan smartphone Android sebagai perangkat pengujian langsung. Dari sisi perangkat lunak, aplikasi dikembangkan menggunakan *Flutter*, *Dart*, *Visual Studio Code*, Android Studio, *SQLite*, serta DB Browser for *SQLite*. Perangkat tersebut mendukung proses pengembangan mulai dari penulisan kode, *debugging*, pengujian tampilan, hingga validasi penyimpanan data pada *Database* lokal.

Implementasi *Database* dilakukan menggunakan *SQLite* melalui package sqflite agar aplikasi dapat berjalan secara *offline* tanpa bergantung pada server eksternal. Struktur *Database* terdiri dari tabel *Users*, *Habit*s, dan *Habit*_logs. Tabel *Users* digunakan untuk menyimpan data akun pengguna, tabel *Habit*s digunakan untuk menyimpan data kebiasaan, sedangkan tabel *Habit*_logs digunakan untuk menyimpan riwayat *Checklist* kebiasaan. Ketiga tabel tersebut saling mendukung dalam menjalankan fitur utama aplikasi, seperti registrasi, *login* , tambah *Habit*, *Checklist Habit*, statistik, dan prediksi konsistensi.

Penerapan prinsip *Atomic Habits* diwujudkan melalui empat konsep utama, yaitu *Cue, Craving, Response, dan reward*. Konsep *Cue* diterapkan melalui fitur *Reminder* atau notifikasi pengingat. Konsep *Craving* diterapkan melalui tampilan antarmuka yang menarik dan mudah digunakan. Konsep *Respons*e diterapkan melalui fitur *Checklist Habit* sebagai tindakan pengguna dalam menandai kebiasaan yang telah dilakukan. Sementara itu, konsep *reward* diterapkan melalui sistem poin, progress, statistik, dan rekomendasi. Dengan penerapan prinsip tersebut, aplikasi

tidak hanya berfungsi sebagai pencatat kebiasaan, tetapi juga sebagai alat bantu pemantauan pembentukan kebiasaan positif.

Fitur statistik pada aplikasi digunakan untuk membantu pengguna melihat perkembangan kebiasaan yang telah dilakukan. Statistik harian dan mingguan menampilkan data aktivitas dalam bentuk grafik sehingga pengguna dapat memahami pola kebiasaan, seperti jumlah kebiasaan yang selesai, kebiasaan yang terlewat, dan perkembangan aktivitas dalam periode tertentu. Fitur ini mendukung tujuan aplikasi sebagai media monitoring kebiasaan secara visual dan mudah dipahami.

Selain fitur pencatatan dan statistik, aplikasi juga menerapkan algoritma *Random Forest* untuk mendukung analisis konsistensi *Habit*. Proses pelatihan model dilakukan menggunakan beberapa fitur, yaitu target_per_week, completed_this_week, total_logs, completed_logs, missed_logs, dan *Habit*_age_days. Model *Random Forest* dilatih menggunakan Python, kemudian hasil pelatihan digunakan sebagai dasar penyusunan aturan prediksi *offline* di dalam aplikasi *Flutter*. Dengan pendekatan ini, aplikasi dapat menampilkan prediksi atau penilaian konsistensi tanpa harus terhubung ke server atau *API* eksternal.

Pengujian sistem dilakukan menggunakan *Statement* dengan fokus pada logika internal program. Modul yang diuji meliputi *Login* , *Register*, Tambah *Habit*, *Checklist Habit*, Statistik *Habit*, dan *Random Forest AI*. Pada setiap modul dilakukan identifikasi *Statement*, pembuatan *Flowchart* , *Flowgraph* , perhitungan *Cyclomatic Complexity*, *Independent Path*, *test case*, dan *Statement Coverage*. Hasil pengujian menunjukkan bahwa seluruh skenario *test case* memperoleh hasil valid, sehingga fungsi utama aplikasi telah berjalan sesuai dengan rancangan sistem.

Hasil *Statement Coverage* pada setiap modul menunjukkan nilai 100%, yang berarti seluruh *Statement* utama yang telah diidentifikasi berhasil dijalankan melalui *test case* yang dibuat. Namun, nilai *Statement Coverage* tidak dimaksudkan untuk menyatakan bahwa aplikasi sepenuhnya bebas dari kesalahan, melainkan menunjukkan bahwa jalur logika utama pada aplikasi telah diuji sesuai dengan *Independent Path* yang diperoleh dari perhitungan *Cyclomatic Complexity*.

Secara keseluruhan, hasil implementasi dan pengujian menunjukkan bahwa aplikasi *Habit Tracker* berhasil menggabungkan prinsip *Atomic Habits* dengan algoritma *Random Forest*. Prinsip *Atomic Habits* digunakan sebagai dasar dalam merancang fitur pembentukan kebiasaan, sedangkan *Random Forest* digunakan untuk memberikan prediksi atau penilaian konsistensi berdasarkan data aktivitas pengguna. Dengan demikian, aplikasi *Habit Tracker* dapat digunakan sebagai alat bantu bagi mahasiswa untuk mencatat, memantau, mengevaluasi, dan memperoleh gambaran mengenai konsistensi kebiasaan positif secara lebih terarah.

UNIVERSITAS PRABUMULIH
UNPRA
2000

BAB VI
KESIMPULAN DAN SARAN

6.1 Kesimpulan

Berdasarkan hasil analisis, perancangan, implementasi, dan pengujian yang telah dilakukan pada penelitian berjudul "Implementasi Prinsip *Atomic Habits* dan Algoritma *Random Forest* pada Aplikasi *Habit Tracker*", maka dapat disimpulkan bahwa penelitian ini berhasil menjawab rumusan masalah yang telah ditetapkan.

1. Aplikasi *Habit Tracker* berbasis Android berhasil dirancang dan dibangun sebagai media pendukung pembentukan kebiasaan positif mahasiswa. Aplikasi ini tidak hanya berfungsi sebagai pencatat kebiasaan, tetapi juga menyediakan fitur yang membantu pengguna dalam mengatur, memantau, dan mengevaluasi kebiasaan secara lebih terstruktur. Fitur utama yang diterapkan meliputi registrasi akun, *login* , tambah *Habit*, *Checklist Habit*, statistik *Habit*, dan rekomendasi berbasis *AI*.
2. Prinsip *Atomic Habits* berhasil diimplementasikan ke dalam fitur aplikasi melalui empat konsep utama, yaitu *Cue*, *Craving*, *Response*, dan *reward*. Konsep *Cue* diterapkan melalui fitur *Reminder* atau notifikasi sebagai pengingat kebiasaan. Konsep *Craving* diterapkan melalui tampilan antarmuka aplikasi yang menarik dan mudah digunakan. Konsep *Response* diterapkan melalui fitur *Checklist Habit* sebagai bentuk tindakan pengguna dalam menyelesaikan kebiasaan. Sedangkan konsep *reward* diterapkan melalui sistem poin sebagai bentuk penghargaan kepada pengguna, sedangkan progress, statistik, dan rekomendasi berperan sebagai umpan balik pendukung terhadap perkembangan kebiasaan pengguna.
3. Algoritma *Random Forest* berhasil dimanfaatkan untuk memprediksi tingkat konsistensi kebiasaan mahasiswa berdasarkan data aktivitas *Habit* pengguna. Model *Random Forest* dilatih menggunakan data *Training* dengan fitur seperti *target_per_week, completed_this_week, total_logs, completed_logs, missed_logs, dan Habit_age_days*. Hasil pelatihan tersebut kemudian digunakan sebagai dasar logika prediksi *offline* di dalam aplikasi *Flutter*, sehingga aplikasi dapat menampilkan hasil prediksi tanpa harus terhubung ke server atau API

eksternal. Hasil prediksi ditampilkan dalam bentuk status seperti success, warning, dan danger, sehingga pengguna dapat mengetahui kondisi konsistensi kebiasaan yang sedang dijalankan.

6.2 Saran

Berdasarkan hasil penelitian yang telah dilakukan, aplikasi *Habit Tracker* ini masih dapat dikembangkan pada penelitian selanjutnya agar menjadi lebih optimal. Pengembangan dapat difokuskan pada penyempurnaan fitur utama, seperti notifikasi yang lebih adaptif, tampilan statistik yang lebih informatif, serta penggunaan data aktivitas pengguna yang lebih banyak agar hasil prediksi konsistensi menggunakan algoritma *Random Forest* menjadi lebih akurat. Selain itu, pengujian aplikasi juga dapat diperluas dengan melibatkan pengguna secara langsung untuk mengetahui kemudahan penggunaan dan kesesuaian aplikasi dengan kebutuhan mahasiswa. Dengan demikian, aplikasi tidak hanya berjalan dengan baik secara teknis, tetapi juga dapat memberikan manfaat yang lebih sesuai dalam membantu pengguna memantau dan mengevaluasi kebiasaan positif secara terstruktur.

DAFTAR PUSTAKA

Akbar, A., Indra, Z., Andriyani, Y., & Melia, T. (2023). Implementation Of The Random Forest Method For Predicting Students' Length Of Study. *Journal Of Statistical Methods And Data Science*, *01*(2), 32–43. Https://Jurnalmipa.Unri.Ac.Id/Jsmds

Alfiansyah, D. M., Setiyani, L., Wati, D. F., Jomantara, M. J., Dedih, D., & Hilman, A. (2026). AI-Based Habit Reminder For Mewing Exercise: Pengembangan Aplikasi Mobile Berbasis Kecerdasan Buatan Untuk Meningkatkan Konsistensi Praktik Mewing. *Jurnal Pengabdian Masyarakat Dan Riset Pendidikan*, *4*(4), 25729–25736. Https://Doi.Org/10.31004/Jerkin.V4i4.6468

Annisa, Abduh, H., & Dani, A. A. H. (2022). Sistem Informasi Geografis Lokasi Dokter Praktek Berbasis Android Di Kota Palopo. *KOMPUTA : Jurnal Ilmiah Komputer Dan Informatika*, *Vol. 11, No. 2*(2). Https://Doi.Org/Https://Doi.Org/10.34010/Komputa.V11i2.7358

Arifah, Sari, A. Y., Balqis, A., Fahlevy, P. L., & Marhadi, H. (2024). *Analisis Prinsip Dan Prosedur Penilaian Di Kelas V SDN 136 Pekanbaru*. *7*(1), 22–31. Https://Doi.Org/Doi:Http://Dx.Doi.Org/10.33369/....

Asrul. (2025). *Machine Learning*. Yash Media.

Aulia, F. N., & Robiansyah, F. (2023). Implementasi Metode Atomic Habits Dalam Membina Karakter Religius Siswa Sekolah Dasar. *Jurnal Review Pendidikan Dasar*, *9*(3). Http://Journal.Unesa.Ac.Id/Index.Php/PD

Aulianti, W. D., Karim, S. A., & Riska, M. (2021). *Pengembangan Game Pendidikan Anti Korupsi Berbasis Android*. *4*(2). Https://Journal.Unm.Ac.Id/Index.Php/Mediatik/Article/View/3077

Clear, J. (2018). *Atomic Habits*.

Fahmi, R. A., & Hidayah, F. (2025). *Designing A Character Education Development Based On Atomic Habit By James Clear*. Https://Proceedingsiches.Com/Index.Php/Ojs/Article/View/324

Fauzi, A., & Yunial, A. H. (2025). Testing Dan Qa Perangkat Lunak.

Fauzi, R., Zainy, A., Nasution, H. N., Hastini, F., & Simanjutak, F. A. (2022). Perancangan Aplikasi Pariwisata Berbasis Android Di Kota Padang Sidempuan. *Jurnal Education And Development*, *11*(1), 437–442. Https://Doi.Org/10.37081/Ed.V11i1.2687

Firnando, J., Franko, B., Tanzil, S. P., Wilyanto, N., Tan, H. C., & Kom, H. M. (2023). *Pembuatan Website Menggunakan Visual Studio Code Di SMA Xaverius 3 Palembang* (Vol. 3, Number 1). Https://Jurnal.Mdp.Ac.Id/Index.Php/Fordicate/Article/View/5057

Harmita, D., & Aly, H. N. (2023). *Implementasi Pengembangan Dan Tujuan Kurikulum*. *3*(1). Https://Share.Google/H76p8cjfwqjmmmpdn

Hidayah, A. N., & Asnadi, N. M. (2024). Penerapan Metode Agile Dalam Manajemen Proyek: Systematic Literature Review. In *Februari* (Vol. 6, Number 1). Https://Www.Researchgate.Net/Publication/379379797_Penerapan_Metode_Agile_Dalam_Manajemen_Proyek_Systematic_Literature_Review

Jamas, A. R., Anggraeni, D., Mubarak, M. A. N., & Aribowo, D. (2023). Perangkat Lunak Komputer. *Jurnal Sains Dan Teknologi*, *2*(2), 94–105. Https://Doi.Org/10.58169/Saintek.V2i2.254

Krismon, A., Prasetya, D. Z. E., & Holidie, Y. A. (2022). *Aplikasi Wisata Kota Kediri Berbasis Android Menggunakan Android Studio*. Https://Files01.Core.Ac.Uk/Download/Pdf/660624815.Pdf

Kurnia, D. (2025). *Pengembangan Aplikasi Mobile Pengelola Data Sarana Berbasis Android Studio Dengan Fitur Crud Menggunakan Sqlite* (Vol. 3, Number 2). Https://Www.Journal.Hdgi.Org/Index.Php/Git/Index

Larasati, F. A., Ratnawati, D. E., & Hanggara, B. T. (2022). *Analisis Sentimen Ulasan Aplikasi Dana Dengan Metode Random Forest* (Vol. 6, Number 9). Http://J-Ptiik.Ub.Ac.Id

Mahmuda, S. (2024). Implementasi Metode Random Forest Pada Kategori Konten Kanal Youtube. *Jurnal Jendela Matematika*, *2*.

Muslimah, F. W., & Subando, J. (2025a). Habit Tracker Sebagai Media Cetak Untuk Membiasakan Aktivitas Produktif Santriwati Di Pondok Pesantren. *Jurnal Ilmiah Pengembangan Pendidikan*, *4*(1), 38–44. Https://Ejournal.Lpipb.Com/Index.Php/Inovasi

Muslimah, F. W., & Subando, J. (2025b). Habit Tracker Sebagai Media Cetak Untuk Membiasakan Aktivitas Produktif Santriwati Di Pondok Pesantren. *Jurnal Ilmiah Pengembangan Pendidikan*, *4*(1), 38–44. Https://Ejournal.Lpipb.Com/Index.Php/Inovasi

Nasution M. Adlan Al Hawari, Siswanto, & Suryana Eko. (2023). Rancangan Media Pembelajaran Berupa Aplikasi Augmented Reality Berbasis Android.

Nistrina, K., & Lestari, T. A. (2024). *Desain Inovatif Sistem Informasi Profil Hotel Damanaka Pangalengan Berbasis Website Menggunakan UML Dan Figma*. Https://Ejournal.Unibba.Ac.Id/Index.Php/J-Sika/Article/View/1457

Nugraha, F. A., & A. Susetyo, Y. (2023). Analisis Perbandingan Performa Database Duckdb Dan Sqlite Pada Pengolahan Big Data. *Jipi (Jurnal Ilmiah Penelitian Dan Pembelajaran Informatika)*, *8*(3), 1052–1060. Https://Doi.Org/10.29100/Jipi.V8i3.4032

Nuraeni, N., & Mujahidin, E. (2021). Landasan Dan Prinsip-Prinsip Perencanaan Pendidikan Islam. *Idarah Tarbawiyah: Journal Of Management In Islamic Education*, *2*(2), 104. Https://Doi.Org/10.32832/Itjmie.V2i2.4596

Puspasari, I., & Dafit, F. (2021). Implementasi Gerakan Literasi Sekolah Di Sekolah Dasar. *Jurnal Basicedu*, *5*(3), 1390–1400. Https://Doi.Org/10.31004/Basicedu.V5i3.939

Ramdany, S. W., Kaidar, S. A., & Aguchino, B. (2024). Penerapan UML Class Diagram Dalam Perancangan Sistem Informasi Perpustakaan Berbasis Web. In *Journal Of Industrial And Engineering System* (Vol. 5, Number 1).

Rasiban, Septiansyah, A., Hasanah, S., Nita Permatasari, V., & Yuliawati, A. (2024). *Sistem Informasi Otomatisasi Pelaporan Data Penjualan Toko Buku Nazwa Yang Masuk Dan Yang Keluar*. Https://Doi.Org/10.37817/Ikraith-Informatika.V8i1

Reba, Y. A., Horota, N., Smas, M. H., & Mokay, M. M. (2021). Kebiasaan Belajar Mahasiswa Ditinjau Dari Self-Management Dan Penyesuaian Diri. *JKI (Jurnal Konseling Indonesia)*, *7*(1), 7–15. Http://Ejournal.Unikama.Ac.Id/Index.Php/JKI

Safaat, N. (2021). *Membangun Aplikasi Dengan Android Flutter*. Informatika Bandung.

Setiawan, H. (2022). *Peran Software, Hardware Dan Brainware Dalam Sistem Informasi Manajemen Sekolah*. *1*. Https://Ejurnal.Kptk.Or.Id/Oase/Article/View/9

Setiawan Iwan, Ariansyah, & Purbasari Yuntari. (2020). *Buku Ajar Kuliah Metode Penelitian Kualitatif*. Deepublish.

Setiawan, P. R., Syefriani, & Wandri, R. (2025). Inovasi Teknologi Melalui Pembelajaran Flutter: Menyongsong Era Aplikasi Mobile. In *Jurnal Sains Dan Teknologi Dalam Pengabdian Masyarakat* (Vol. 2). Https://Jurnal.Pnj.Ac.Id/Index.Php/JAP/Article/View/8125

Sofi, N., & Dharmawan, R. (2022). Perancangan Aplikasi Bengkel Csm Berbasis Android Menggunakan Framework Flutter (Bahasa Dart). *JTS*, *1*(2). Https://Journal.Admi.Or.Id/Index.Php/JTS/Article/View/125

Suharni, Susilowati, E., & Pakusadewa, F. (2023). *Perancangan Website Rumah Makan Ninik Sebagai Media Promosi Menggunakan Unified Modelling Language*. *12*. Https://Journal.Istn.Ac.Id/Index.Php/Rekayasainformasi/Article/View/1527

Surianto, D. F., Wahid, M. S. N. W., Parenreng, J. M., Wahid, A., Zain, S. G., Edy, M. R., & Risal, A. A. N. (2023). *PKM Pelatihan Figma Untuk Desain*

Prototipe Sistem Informasi. *01*(02). Https://Journal.Diginus.Id/Index.Php/VOKATEK/Index

Suryana, T. (2021). *Belajar Bahasa Pemrograman Dart (1)*.

Syahfitri, D., Rahayuningtyas, D., Garcia, R., Rakhmah, S. N., Sariasih, F. A., & Sutoyo, I. (2025). *Perancangan Sistem Prediksi Deteksi Alzheimer Berbasis Random Forest Menggunakan Metode Scrum*. Https://Ejournal.Pppmitpa.Or.Id/Index.Php/Betrik/Article/View/298

Vindua, R. (2024). *Metode Pengembangan Sistem Secara Tradisional Dan Terbaru*. Eureka Media Askara.

Waruwu, M. (2023). *Pendekatan Penelitian Pendidikan: Metode Penelitian Kualitatif, Metode Penelitian Kuantitatif Dan Metode Penelitian Kombinasi (Mixed Method)*. (1). Https://Jptam.Org/Index.Php/Jptam/Article/View/6187

LAMPIRAN

Lampiran 1 Surat Permohonan Penelitian

FAKULTAS ILMU KOMPUTER
UNIVERSITAS PRABUMULIH

Prabumulih, 03 November 2025

Nomor : 052 /FASILKOM/UNPRA/Inf-S1/XI/2025
Lampiran : -
Perihal : Permohonan Penelitian dan Pengambilan Data

Kepada Yth,
Rektor Universitas Prabumulih
Di
Tempat

Dalam rangka pelaksanaan skripsi mahasiswa Program Studi Informatika Fakultas Ilmu Komputer Universitas Prabumulih, maka kami harapkan bantuan dari Bpk/Ibu kiranya mengizinkan mahasiswa:

1. Nama : Windi Yulinda
   Nim : 2022230015
   Program Studi : Informatika

Untuk melaksanakan penelitian dan pengambilan data ditempat Bpk/Ibu Pimpin guna menyelesaikan skripsi. Atas bantuan dan kerjasamanya kami ucapkan terima kasih.

Ketua Program Studi,

Iwan Setiawan, S.Kom., M.Kom.
NIDN.0209129101

Jalan Patra No 50 RT 01 RW 03 Kelurahan Sukaraja Kecamatan Prabumulih Selatan
Kota Prabumulih Sumatera Selatan Indonesia Telpon 0713-3224417 Fax 0713-322418

Dipindai dengan CamScanner

Lampiran 2 Surat Balasan Permohonan Penelitian

UNIVERSITAS PRABUMULIH

Prabumulih, 07 November 2025

Nomor : 382/UNPRA/XI/2025

Lampiran : -

Perihal : **Balasan Permohonan Penelitian dan Pengambilan Data**

Kepada Yth.

Ketua Program Studi Informatika

Fakultas Ilmu Komputer - Universitas Prabumulih

Di -

Tempat

Dengan hormat,

Menindaklanjuti surat dari Ketua Program Studi Informatika Fakultas Fakultas Ilmu Komputer Universitas Prabumulih pada tanggal 03 November 2025 nomor: 032/FASILKOM/UNPRA/Inf-S1/XI/2025 perihal: Permohonan Penelitian dan Pengambilan Data atas nama mahasiswa:

Nama : Windi Yulinda

NIM : 2022230015

Pada dasarnya kami tidak keberatan dan silahkan saja melakukan penelitian guna mendapatkan data pendukung yang dibutuhkan dalam penyusunan Skripsi tersebut dengan mengikuti peraturan yang ada di lingkungan Universitas Prabumulih.

Demikianlah, atas perhatian dan kerjasamanya diucapkan terima kasih.

Rektor,

Dr. Yunlar Pratiwi, S.Si., M.Si.

Jalan Patra No.50 RT.01 RW.03 Kelurahan Sukaraja Kecamatan Prabumulih Selatan
Kota Prabumulih Sumatera Selatan Indonesia Telpon 0713-3224417 Fax 0713-322418
e-mail : mail@unpra.ac.id

Dipindai dengan CamScanner

Lampiran 3 Dokumentasi Wawancara & Observasi

Lampiran 4 Daftar Pertanyaan Wawancara

Draf Wawancara

Nama Narasumber : Ranti Yuliningsih, A.Md.

Jabatan : Staf Administrasi Rektorat

Tempat Wawancara : Kantor Rektorat Universitas Prabumulih

Tanggal Wawancara : 10 November 2025

Pewawancara : Windi Yulinda

No.	Pertanyaan	Jawaban Narasumber
1	Dapatkah Ibu menjelaskan sejarah berdirinya Universitas Prabumulih?	Universitas Prabumulih (UNPRA) berdiri di bawah Yayasan Pendidikan Prabumulih melalui penggabungan tiga sekolah tinggi: STIE Prabumulih, STIT Prabumulih, dan STMIK Prabumulih. Penggabungan tersebut ditetapkan dalam Surat Keputusan Menteri Pendidikan, Kebudayaan, Riset, dan Teknologi Nomor 254/E/O/2022 pada 7 April 2022.
2	Bagaimana proses tiga sekolah tinggi tersebut digabungkan menjadi UNPRA?	Ketiga sekolah tinggi awalnya berjalan terpisah, namun tahun 2022 digabungkan menjadi universitas. Setelah penggabungan, STIE menjadi Fakultas Ekonomi dan Bisnis, STMIK menjadi Fakultas Ilmu Komputer, dan STIT menjadi Fakultas Teknik.
3	Apa visi resmi Universitas Prabumulih?	"Terwujudnya Universitas Prabumulih menjadi Perguruan Tinggi yang terkemuka dalam menghasilkan sumber daya manusia yang berkualitas dan bertanggung jawab dalam penguasaan dan pengembangan ilmu pengetahuan, teknologi dan seni pada tahun 2029."
4	Apa saja misi Universitas Prabumulih?	1) Menyelenggarakan pendidikan dan pengajaran yang bermutu. 2) Menghasilkan SDM yang kompeten sesuai kebutuhan dunia kerja. 3) Menghasilkan lulusan berdaya saing global. 4) Meningkatkan kualitas tridharma perguruan tinggi. 5) Mengembangkan kemitraan dengan berbagai instansi. 6) Mendayagunakan sarana dan prasarana secara efektif dan produktif.
5	Apa tujuan Universitas Prabumulih dalam penyelenggaraan pendidikan tinggi?	1) Menghasilkan lulusan berkepribadian Indonesia dan bermoral tinggi. 2) Profesional, terampil, dan beretika. 3) Mampu mengembangkan IPTEKS.

		bermanfaat. 6) Melaksanakan pengabdian kepada masyarakat secara berkelanjutan.
6	Apa saja fakultas dan program studi yang tersedia di Universitas Prabumulih?	Fakultas Ilmu Komputer: Sistem Informasi (S1), Informatika (S1), Komputerisasi Akuntansi (D3). Fakultas Ekonomi & Bisnis: Manajemen (S1), Akuntansi (S1), Bisnis Digital (S1). Fakultas Teknik: Teknik Lingkungan (S1), Teknik Pertambangan (D3).

Menyetujui,
Narasumber

Ranti Ruminingsih, Am.d.

Prabumulih, 07 November 2025
Pewawancara

Windi Yulinda

Dipindai dengan CamScanner

Lampiran 5 Pertanyaan Kuisioner

Data Responden	
Nama	:
Program Studi	:
Semester	:
Jenis Kelamin	:

Tema 1 : Kebiasaan Belajar Mahasiswa

1. Bagaimana kebiasaan belajar Anda selama ini dalam keseharian perkuliahan?
2. Seberapa sering Anda merasa kesulitan menjaga konsistensi belajar dari hari ke hari? Jelaskan.
3. Faktor apa yang paling sering menyebabkan kebiasaan belajar Anda tidak berjalan secara rutin?
4. Apakah Anda pernah membuat jadwal belajar tetapi tidak menjalankannya dengan konsisten? Mengapa?
5. Menurut Anda, apa dampak kebiasaan belajar yang tidak konsisten terhadap hasil belajar Anda?

Tema 2 : Penggunaan Smartphone dan Aplikasi Pendukung Belajar

6. Untuk apa saja Anda paling sering menggunakan *smartphone* dalam sehari?
7. Apakah penggunaan *smartphone* lebih sering membantu atau justru mengganggu waktu belajar Anda? Mengapa?
8. Dalam kondisi apa smartphone paling sering membuat Anda menunda atau lupa belajar?
9. Apakah Anda pernah mencoba memanfaatkan smartphone untuk membantu mengatur atau mendukung aktivitas belajar? Ceritakan.
10. Menurut Anda, apakah smartphone berpotensi menjadi alat pendukung kebiasaan belajar yang konsisten? Jelaskan alasannya.

Tema 3 : Kebutuhan Fitur Aplikasi *Habit Tracker*

11. Apakah sebelum ini Anda pernah mendengar atau menggunakan aplikasi pelacak kebiasaan (*Habit Tracker*)? Jelaskan pengalaman Anda.
12. Jika pernah menggunakan aplikasi sejenis, apa kendala yang membuat Anda jarang atau berhenti menggunakannya?Seberapa penting fitur notifikasi pengingat bagi anda?
13. Menurut Anda, fitur apa yang paling dibutuhkan agar aplikasi dapat membantu menjaga konsistensi belajar?Jika Anda bisa menambahkan satu fitur bebas ke aplikasi *Habit Tracker,* fitur apa itu?
14. Bagaimana tampilan aplikasi yang menurut Anda nyaman dan tidak membosankan untuk digunakan setiap hari?
15. Menurut Anda, apa kekurangan dari aplikasi belajar atau pengatur aktivitas yang pernah Anda gunakan sebelumnya?

Tema 4 : Persepsi terhadap Prinsip *Atomic Habits*

16. Menurut Anda, apakah perubahan kecil yang dilakukan setiap hari dapat berdampak pada kebiasaan belajar jangka panjang? Jelaskan. 17. Apakah pengingat atau notifikasi belajar dapat membantu Anda memulai belajar tepat waktu? Mengapa? 18. Apakah tampilan aplikasi yang menarik dapat meningkatkan keinginan Anda untuk belajar secara konsisten? Jelaskan. 19. Apakah adanya tanda selesai atau *Checklist* setelah belajar dapat memberi rasa puas atau pencapaian bagi Anda? Mengapa? 20. Menurut Anda, apakah pemberian *reward* kecil seperti poin atau badge dapat membantu menjaga motivasi belajar?
Tema 5 : Persepsi terhadap Fitur Prediksi Konsistensi (*Random Forest)*
21. Bagaimana pendapat Anda jika aplikasi dapat menunjukkan apakah kebiasaan belajar Anda sedang konsisten atau tidak? 22. Apakah informasi tentang tingkat konsistensi belajar dapat membantu Anda melakukan evaluasi diri? Mengapa? 23. Menurut Anda, apakah pesan otomatis seperti “kebiasaan belajar Anda cukup konsisten” dapat memengaruhi semangat belajar Anda? Jelaskan. 24. Jika aplikasi dapat memberikan saran belajar berdasarkan kebiasaan Anda, apakah Anda tertarik menggunakannya? Mengapa? 25. Menurut Anda, apa manfaat utama dari fitur yang dapat menilai atau memprediksi konsistensi belajar mahasiswa?

UNPRA 2000

Lampiran 6 Data koding jawaban responden Tema 1

Pertanyaan 1

Bagaimana kebiasaan belajar Anda selama ini dalam keseharian perkuliahan?		
Kode	**Isi**	**Jumlah**
1	Cukup Baik	20
2	Mengulangi materi	9
3	Menggunakan teknik pomodoro	1
4	Tidak teratur/Tidak efektif/Tidak konsisten	22
5	Menggunakan bantuan Internet	1
6	Cenderung lebih suka praktikum	1
7	Biasa saja	5
Kesimpulan : Berdasarkan hasil jawaban responden, dapat disimpulkan bahwa kebiasaan belajar mahasiswa dalam keseharian perkuliahan masih cenderung tidak teratur dan belum konsisten. Meskipun terdapat sebagian responden yang menyatakan memiliki kebiasaan belajar yang cukup baik atau menggunakan metode tertentu, jumlahnya relatif lebih sedikit dibandingkan responden yang mengalami kesulitan dalam mempertahankan pola belajar yang efektif. Temuan ini menunjukkan bahwa mahasiswa masih membutuhkan dukungan sistem yang dapat membantu membangun kebiasaan belajar yang lebih terstruktur dan konsisten.		

Pertanyaan 2

Seberapa sering Anda merasa kesulitan menjaga konsistensi belajar dari hari ke hari? Jelaskan.		
Kode	**Isi**	**Jumlah**
1	Cukup Sering, karena mood/lelah/kurang manajemen waktu/Motivasi menurun/distraksi sosmed dan teman nongkrong/lupa jadwal	63
2	Tidak terlalu sering/Bisa mengatur waktu/suka belajar	13
Kesimpulan: Berdasarkan hasil jawaban responden, sebagian besar mahasiswa cukup sering mengalami kesulitan dalam menjaga konsistensi belajar dari hari ke hari. Kesulitan tersebut disebabkan oleh faktor internal seperti mood, kelelahan, dan motivasi yang menurun, serta faktor eksternal seperti distraksi media sosial, lingkungan pertemanan, dan lupa jadwal belajar. Meskipun terdapat sebagian kecil mahasiswa yang mampu mengatur waktu belajar dengan baik, temuan ini menunjukkan bahwa secara umum mahasiswa masih membutuhkan dukungan sistem yang dapat membantu menjaga konsistensi belajar secara berkelanjutan.		

Pertanyaan 3

Faktor apa yang paling sering menyebabkan kebiasaan belajar Anda tidak berjalan secara rutin?		
Kode	**Isi**	**Jumlah**
1	Distraksi Smartphone/Sosmed/Game	24
2	Susah fokus	6
3	Kurangnya motivasi	9
4	Malas/Manajemen waktu buruk	26
5	Lelah/pekerjaan	7
Kesimpulan: Berdasarkan hasil jawaban responden, faktor yang paling sering menyebabkan kebiasaan belajar mahasiswa tidak berjalan secara rutin adalah distraksi penggunaan smartphone dan media sosial, diikuti oleh malas serta manajemen waktu yang kurang baik. Selain itu, faktor internal seperti kurangnya motivasi dan kesulitan fokus, serta faktor situasional seperti kelelahan atau aktivitas pekerjaan, juga turut memengaruhi konsistensi belajar. Temuan ini menunjukkan		

bahwa mahasiswa membutuhkan dukungan sistem yang mampu meminimalkan distraksi, membantu pengelolaan waktu, serta meningkatkan motivasi belajar.

Pertanyaan 4

Apakah Anda pernah membuat jadwal belajar tetapi tidak menjalankannya dengan konsisten? Mengapa?		
Kode	**Isi**	**Jumlah**
1	Konsisten	4
2	Ya, Karena Distraksi sosmed/pekerjaan	13
3	Ya, faktor kesehatan	2
4	Ya, susah fokus	5
5	Ya, tidak disiplin/menunda	17
6	Ya, malas/bosan	7
7	Ya, karena lupa/Jadwal tidak terlihat	4
8	Jadwal tidak realistis	4
9	Tidak pernah	10
Kesimpulan: Berdasarkan hasil jawaban responden, sebagian besar mahasiswa pernah membuat jadwal belajar namun tidak mampu menjalankannya secara konsisten. Ketidakkonsistenan tersebut disebabkan oleh faktor internal seperti kurangnya disiplin, kecenderungan menunda, rasa malas, dan bosan, serta faktor eksternal seperti distraksi media sosial, pekerjaan, dan lupa jadwal karena tidak terlihat. Selain itu, pembuatan jadwal yang tidak realistis juga menjadi salah satu penyebab kegagalan dalam menjalankan jadwal belajar. Temuan ini menunjukkan bahwa mahasiswa membutuhkan dukungan sistem yang mampu membantu pengingat jadwal, meningkatkan disiplin, serta memudahkan pemantauan dan penyesuaian jadwal belajar.		

Pertanyaan 5

Menurut Anda, apa dampak kebiasaan belajar yang tidak konsisten terhadap hasil belajar Anda?		
Kode	**Isi**	**Jumlah**
1	Lupa dengan materi yang dipelajari sebelumnya/hasil belajari tidak optimal	56
2	Tidak ada masalah	1
Kesimpulan: Berdasarkan hasil jawaban responden, kebiasaan belajar yang tidak konsisten berdampak pada menurunnya pemahaman materi dan tidak optimalnya hasil belajar, yang ditandai dengan mudah lupa terhadap materi yang telah dipelajari sebelumnya. Temuan ini menunjukkan bahwa konsistensi belajar merupakan faktor penting dalam menjaga kualitas dan keberhasilan proses belajar mahasiswa.		

Kesimpulan Tema 1

Berdasarkan hasil analisis jawaban responden pada Tema Kebiasaan Belajar Mahasiswa, dapat disimpulkan bahwa sebagian besar mahasiswa belum memiliki kebiasaan belajar yang teratur dan konsisten. Mahasiswa sering mengalami kesulitan dalam menjaga konsistensi belajar akibat berbagai faktor, seperti distraksi penggunaan smartphone, manajemen waktu yang kurang baik, menurunnya motivasi, serta kecenderungan menunda belajar.

Ketidakkonsistenan dalam belajar juga berdampak pada menurunnya pemahaman materi dan tidak optimalnya hasil belajar, yang ditandai dengan mudah lupa terhadap materi yang telah dipelajari sebelumnya. Temuan ini menunjukkan bahwa mahasiswa membutuhkan dukungan sistem yang dapat membantu membangun kebiasaan belajar yang lebih terstruktur, konsisten, dan berkelanjutan.

UNIVERSITAS PRABUMULIH
UNPRA
2000

Lampiran 7 Data koding jawaban responden Tema 2

Pertanyaan 1

Untuk apa saja Anda paling sering menggunakan *smartphone* dalam sehari?		
Kode	**Isi**	**Jumlah**
1	Sosmed	18
2	Apk hiburan (game/*YouTube*/tiktok/dll)	10
3	Komunikasi	25
4	*Searching* Informasi/Belajar	18
5	Buka E-commerce	2
Kesimpulan: Berdasarkan hasil jawaban responden, penggunaan smartphone oleh mahasiswa dalam keseharian didominasi oleh aktivitas non-akademik, terutama untuk media sosial dan aplikasi hiburan. Sementara itu, penggunaan smartphone untuk keperluan akademik seperti mencari informasi atau belajar masih tergolong lebih rendah. Temuan ini menunjukkan bahwa meskipun smartphone memiliki potensi sebagai media pendukung belajar, pemanfaatannya oleh mahasiswa belum optimal untuk menunjang aktivitas akademik.		

Pertanyaan 2

Apakah penggunaan *smartphone* lebih sering membantu atau justru mengganggu waktu belajar Anda? Mengapa?		
Kode	**Isi**	**Jumlah**
1	Membantu	37
2	Mengganggu/Distraksi	21
3	Kadang membantu kadang mengganggu	19
Kesimpulan: Berdasarkan hasil jawaban responden, penggunaan smartphone dalam aktivitas belajar lebih sering dirasakan membantu, terutama dalam mencari informasi dan mengakses materi pembelajaran. Namun demikian, smartphone juga berpotensi mengganggu waktu belajar apabila penggunaannya tidak terkontrol dan mengarah pada distraksi. Temuan ini menunjukkan bahwa smartphone memiliki peran ganda sebagai pendukung sekaligus penghambat belajar, sehingga diperlukan pengelolaan penggunaan smartphone yang lebih terarah dan terkontrol.		

Pertanyaan 3

Dalam kondisi apa smartphone paling sering membuat Anda menunda atau lupa belajar?		
Kode	**Isi**	**Jumlah**
1	Membuka sosmed/Apk hiburan	36
2	Mengerjakan proyek editing	1
3	Bermain game	23
4	Tanpa koneksi internet (tidak bisa searching informasi)	1
5	Tidak ada (smartphone bukan penyebab penundaan)	III
Kesimpulan: Berdasarkan hasil jawaban responden, kondisi yang paling sering menyebabkan mahasiswa menunda atau lupa belajar akibat penggunaan smartphone adalah membuka media sosial dan aplikasi hiburan, diikuti oleh aktivitas bermain game. Kondisi ini menunjukkan bahwa penggunaan smartphone yang tidak terkontrol, khususnya untuk hiburan, menjadi salah satu faktor utama terjadinya penundaan belajar. Temuan ini mengindikasikan perlunya dukungan sistem yang mampu mengendalikan distraksi dan membantu pengguna tetap fokus pada aktivitas belajar.		

Pertanyaan 4

Apakah Anda pernah mencoba memanfaatkan smartphone untuk membantu mengatur atau mendukung aktivitas belajar? Ceritakan.		
Kode	**Isi**	**Jumlah**
1	Ya, Menulis materi/Menonton tutorial/menggunakan *AI*	20
2	Ya, menggunakan timer/Apk pengingat	20
3	Ya, membuat jadwal/ TO DO LIST	10
4	Tidak pernah	7
Kesimpulan: Berdasarkan hasil jawaban responden, sebagian besar mahasiswa telah mencoba memanfaatkan smartphone untuk mendukung aktivitas belajar, baik sebagai sumber materi pembelajaran, media tutorial, maupun sebagai alat bantu pengelolaan waktu melalui timer dan aplikasi pengingat. Namun demikian, pemanfaatan smartphone untuk fungsi pengaturan belajar seperti pembuatan jadwal dan to-do list masih belum merata. Temuan ini menunjukkan bahwa smartphone memiliki potensi besar sebagai alat pendukung belajar, namun masih diperlukan sistem yang mampu mengintegrasikan fungsi belajar dan pengelolaan kebiasaan secara lebih terstruktur.		

Pertanyaan 5

Menurut Anda, apakah smartphone berpotensi menjadi alat pendukung kebiasaan belajar yang konsisten? Jelaskan alasannya.		
Kode	**Isi**	**Jumlah**
1	Ya, berpotensi jika digunakan dengan baik	73
2	Tidak	1
Kesimpulan: Berdasarkan hasil jawaban responden, sebagian besar mahasiswa berpendapat bahwa smartphone memiliki potensi besar sebagai alat pendukung kebiasaan belajar yang konsisten, selama penggunaannya dilakukan secara tepat dan terkontrol. Temuan ini menunjukkan bahwa smartphone tidak hanya dipandang sebagai sumber distraksi, tetapi juga sebagai media yang dapat dimanfaatkan untuk mendukung pembentukan kebiasaan belajar.		

Kesimpulan Tema 2

Berdasarkan hasil analisis jawaban responden pada Tema Penggunaan Smartphone dalam Aktivitas Belajar, dapat disimpulkan bahwa smartphone lebih sering digunakan untuk aktivitas non-akademik, seperti media sosial dan hiburan, sehingga berpotensi menjadi sumber distraksi belajar. Meskipun demikian, mahasiswa juga menyadari bahwa smartphone memiliki peran penting sebagai alat pendukung belajar, terutama untuk mencari informasi, mengakses materi, dan membantu pengelolaan waktu melalui fitur pengingat dan timer.

Penggunaan smartphone yang tidak terkontrol, khususnya untuk media sosial dan hiburan, menjadi faktor utama yang menyebabkan penundaan dan gangguan fokus belajar. Namun, mayoritas responden meyakini bahwa smartphone berpotensi menjadi alat pendukung kebiasaan belajar yang konsisten apabila difasilitasi oleh

sistem yang mampu mengarahkan penggunaan smartphone secara lebih terstruktur, produktif, dan terkontrol.

Lampiran 8 Data koding jawaban responden Tema 3

Pertanyaan 1

Apakah sebelum ini Anda pernah mendengar atau menggunakan aplikasi pelacak kebiasaan (*Habit Tracker*)? Jelaskan pengalaman Anda.		
Kode	**Isi**	**Jumlah**
1	Tidak pernah	43
2	Pernah mencoba namun penggunaannya kurang maksimal	25
3	Baru pertama kali mendengar tentang *Habit Tracker*	2
Kesimpulan: Berdasarkan hasil jawaban responden, sebagian besar mahasiswa belum pernah menggunakan aplikasi pelacak kebiasaan (*Habit Tracker*), bahkan sebagian di antaranya baru pertama kali mendengar konsep tersebut. Meskipun terdapat responden yang pernah mencoba menggunakan aplikasi *Habit Tracker*, penggunaannya masih belum maksimal dan tidak berkelanjutan. Temuan ini menunjukkan bahwa tingkat pemanfaatan aplikasi *Habit Tracker* di kalangan mahasiswa masih rendah, sehingga diperlukan perancangan aplikasi yang lebih mudah dipahami, menarik, dan relevan dengan kebutuhan pengguna.		

Pertanyaan 2

Jika pernah menggunakan aplikasi sejenis, apa kendala yang membuat Anda jarang atau berhenti menggunakannya?		
Kode	**Isi**	**Jumlah**
1	Tidak pernah menggunakan	33
2	Pernah, tapi terkendala fitur tanpa pengingat, tampilan kurang menarik	4
3	Pernah, namun aplikasinya membosankan(kurang termotivasi)	4
4	Tidak konsisten menginput data	25
5	Berbayar	1
6	Tidak ada fitur histori/progressbar	1
7	Aplikasinya tidak adaftif dengan kebutuhan pengguna	1
Kesimpulan: Berdasarkan hasil jawaban responden, kendala utama yang menyebabkan mahasiswa jarang atau berhenti menggunakan aplikasi *Habit Tracker* adalah rendahnya tingkat keterlibatan pengguna, yang ditandai dengan ketidakkonsistenan dalam menginput data. Selain itu, aplikasi sejenis dinilai kurang menarik, kurang memotivasi, serta belum menyediakan fitur pendukung seperti pengingat, tampilan yang menarik, dan visualisasi progres. Temuan ini menunjukkan bahwa aplikasi *Habit Tracker* perlu dirancang dengan pendekatan yang lebih interaktif, adaptif, dan mampu mendorong pengguna untuk tetap konsisten menggunakan aplikasi.		

Pertanyaan 3

Menurut Anda, fitur apa yang paling dibutuhkan agar aplikasi dapat membantu menjaga konsistensi belajar?		
Kode	**Isi**	**Jumlah**
1	Ceklis	3
2	pengingat	48
3	*Reward*	5
4	Laporan Progress	26

5	Kalender	3
6	Fitur Vidio	2
7	Fitur rekomendasi/Motivasi	8
8	*Login* & *Logout*	1
9	Tampilan Menarik	4

Kesimpulan: Berdasarkan hasil jawaban responden, fitur yang paling dibutuhkan untuk membantu menjaga konsistensi belajar adalah fitur pengingat, diikuti oleh laporan progres yang dapat menampilkan perkembangan kebiasaan belajar pengguna. Selain itu, fitur *reward* serta rekomendasi atau motivasi juga dinilai penting untuk meningkatkan keterlibatan dan semangat belajar. Temuan ini menunjukkan bahwa aplikasi *Habit Tracker*
perlu dirancang dengan fokus pada pengingat yang konsisten, pemantauan progres, serta mekanisme motivasi agar mampu mendukung pembentukan kebiasaan belajar secara berkelanjutan.

Pertanyaan 4

Bagaimana tampilan aplikasi yang menurut Anda nyaman dan tidak membosankan untuk digunakan setiap hari?

Kode	Isi	Jumlah
1	Menggunakan soft colors	27
2	Tampilan minimalis dan aestetics	25
3	Tampilan dengan fitur menarik	16
4	Colorfull	1
5	Tanpa iklan	1
6	Interaktif	3
7	Ada *AI* nya	1

Kesimpulan: Berdasarkan hasil jawaban responden, tampilan aplikasi yang diharapkan untuk mendukung penggunaan harian adalah tampilan yang sederhana, minimalis, dan menggunakan warna lembut, sehingga nyaman dan tidak membosankan. Selain itu, responden juga mengharapkan adanya elemen interaktif dan fitur visual yang menarik untuk meningkatkan pengalaman pengguna. Temuan ini menunjukkan bahwa desain antarmuka aplikasi *Habit Tracker* perlu dirancang secara sederhana namun tetap menarik, agar pengguna dapat menggunakan aplikasi secara konsisten.

Pertanyaan 5

Menurut Anda, apa kekurangan dari aplikasi belajar atau pengatur aktivitas yang pernah Anda gunakan sebelumnya?

Kode	Isi	Jumlah
1	Tidak ada notifikasi	3
2	Navigasi yang rumit	18
3	Adanya iklan yang mengganggu	2
4	Tampilan Kurang menarik	11
5	Tidak ada history/progressbar	3
6	Tidak ada fitur *AI*	1

Kesimpulan: Berdasarkan hasil jawaban responden, kekurangan utama dari aplikasi belajar atau pengatur aktivitas yang pernah digunakan adalah navigasi yang rumit dan tampilan aplikasi yang kurang menarik, sehingga menurunkan kenyamanan penggunaan. Selain itu, responden juga menilai bahwa aplikasi tersebut belum menyediakan fitur pendukung konsistensi, seperti notifikasi, histori progres, serta fitur cerdas yang adaptif. Temuan ini menunjukkan bahwa aplikasi *Habit Tracker* perlu dirancang dengan antarmuka yang sederhana, menarik, dan dilengkapi fitur pendukung kebiasaan belajar secara berkelanjutan.

Kesimpulan tema 3

Berdasarkan hasil analisis jawaban responden pada Tema Kebutuhan Fitur Aplikasi *Habit Tracker*, dapat disimpulkan bahwa sebagian besar mahasiswa belum familiar atau belum menggunakan aplikasi *Habit Tracker* secara optimal. Aplikasi sejenis yang pernah digunakan dinilai kurang menarik, kurang memotivasi, serta tidak mampu mendorong konsistensi penggunaan, terutama karena ketidakkonsistenan dalam menginput data dan minimnya fitur pendukung.

Responden mengharapkan aplikasi *Habit Tracker* yang dilengkapi fitur pengingat sebagai kebutuhan utama, disertai dengan laporan progres, *reward*, rekomendasi atau motivasi, serta tampilan aplikasi yang sederhana, minimalis, dan nyaman digunakan. Selain itu, aplikasi juga diharapkan memiliki navigasi yang mudah, bebas dari gangguan iklan, serta fitur adaptif yang sesuai dengan kebutuhan pengguna, sehingga mampu membantu membentuk kebiasaan belajar yang konsisten dan berkelanjutan.

Lampiran 9 Data koding jawaban responden Tema 4

Pertanyaan 1

Menurut Anda, apakah perubahan kecil yang dilakukan setiap hari dapat berdampak pada kebiasaan belajar jangka panjang? Jelaskan.		
Kode	**Isi**	**Jumlah**
1	Ya	67
2	Tidak	3
Kesimpulan: Berdasarkan hasil jawaban responden, sebagian besar mahasiswa meyakini bahwa perubahan kecil yang dilakukan secara konsisten setiap hari dapat memberikan dampak positif terhadap kebiasaan belajar dalam jangka panjang. Temuan ini menunjukkan bahwa mahasiswa memiliki persepsi yang sejalan dengan prinsip *Atomic Habit*s, yaitu bahwa pembentukan kebiasaan yang berkelanjutan dapat dimulai dari langkah-langkah kecil yang dilakukan secara rutin.		

Pertanyaan 2

Apakah pengingat atau notifikasi belajar dapat membantu Anda memulai belajar tepat waktu? Mengapa?		
Kode	**Isi**	**Jumlah**
1	Ya	56
2	Tidak	4
Kesimpulan: Berdasarkan hasil jawaban responden, sebagian besar mahasiswa menyatakan bahwa pengingat atau notifikasi belajar dapat membantu mereka memulai belajar tepat waktu. Hal ini menunjukkan bahwa pengingat berperan sebagai pemicu awal yang efektif dalam membangun kebiasaan belajar. Temuan ini sejalan dengan prinsip *Cue* dalam teori *Atomic Habit*s, di mana rangsangan awal dapat mendorong seseorang untuk melakukan suatu perilaku secara konsisten.		

Pertanyaan 3

Apakah tampilan aplikasi yang menarik dapat meningkatkan keinginan Anda untuk belajar secara konsisten? Jelaskan.		
Kode	**Isi**	**Jumlah**
1	Ya	70
2	Tidak	3
Kesimpulan: Berdasarkan hasil jawaban responden, sebagian besar mahasiswa berpendapat bahwa tampilan aplikasi yang menarik dapat meningkatkan keinginan mereka untuk belajar secara konsisten. Temuan ini menunjukkan bahwa desain visual aplikasi memiliki peran penting dalam menjaga minat dan keterlibatan pengguna. Hal ini sejalan dengan prinsip *make it attractive* dalam teori *Atomic Habit*s, yang menekankan bahwa suatu perilaku akan lebih mudah dilakukan jika terasa menarik bagi individu.		

Pertanyaan 4

Apakah adanya tanda selesai atau *Checklist* setelah belajar dapat memberi rasa puas atau pencapaian bagi Anda? Mengapa?		
Kode	**Isi**	**Jumlah**
1	Ya	68
2	Tidak	4
Kesimpulan: Berdasarkan hasil jawaban responden, sebagian besar mahasiswa merasakan bahwa adanya tanda selesai atau *Checklist* setelah belajar dapat memberikan rasa puas dan		

pencapaian. Hal ini menunjukkan bahwa pengakuan atas keberhasilan kecil mampu memperkuat motivasi belajar. Temuan ini sejalan dengan prinsip *reward* dalam teori *Atomic Habit*s, di mana pemberian umpan balik positif setelah melakukan suatu kebiasaan dapat mendorong individu untuk mengulang perilaku tersebut secara konsisten.

Pertanyaan 5

Menurut Anda, apakah pemberian *reward* kecil seperti poin atau badge dapat membantu menjaga motivasi belajar?		
Kode	**Isi**	**Jumlah**
1	Ya	70
2	Tidak	2
Kesimpulan: Berdasarkan hasil jawaban responden, sebagian besar mahasiswa berpendapat bahwa pemberian *reward* atau umpan balik positif setelah belajar dapat meningkatkan motivasi dan mendorong konsistensi belajar. Meskipun terdapat sebagian kecil responden yang menilai *reward* tidak terlalu berpengaruh, temuan ini menunjukkan bahwa penguatan positif tetap memiliki peran penting dalam pembentukan kebiasaan belajar.		

Kesimpulan Tema 4

Berdasarkan hasil analisis jawaban responden pada Tema Persepsi terhadap Prinsip *Atomic Habit*s, dapat disimpulkan bahwa sebagian besar mahasiswa memiliki persepsi positif terhadap penerapan prinsip *Atomic Habits* dalam membangun kebiasaan belajar. Mahasiswa meyakini bahwa perubahan kecil yang dilakukan secara konsisten dapat memberikan dampak jangka panjang terhadap kebiasaan belajar.

Selain itu, responden menilai bahwa pengingat atau notifikasi dapat membantu memulai belajar tepat waktu (*Cue*), tampilan aplikasi yang menarik dapat meningkatkan keinginan untuk belajar (*make it attractive*), serta *Checklist* dan *reward* dapat memberikan rasa puas dan pencapaian (*reward*). Temuan ini menunjukkan bahwa prinsip-prinsip *Atomic Habits* relevan dan dapat diterapkan secara efektif dalam perancangan fitur aplikasi *Habit Tracker* untuk mendukung konsistensi belajar mahasiswa.

Lampiran 10 Data koding jawaban responden Tema 5

Pertanyaan 1

Bagaimana pendapat Anda jika aplikasi dapat menunjukkan apakah kebiasaan belajar Anda sedang konsisten atau tidak?		
Kode	**Isi**	**Jumlah**
1	Sangat bagus/membantu sekali/keren	72
Kesimpulan: Berdasarkan hasil jawaban responden, mahasiswa memberikan respon yang sangat positif terhadap fitur aplikasi yang dapat menunjukkan tingkat konsistensi kebiasaan belajar. Fitur ini dinilai membantu, menarik, dan bermanfaat untuk meningkatkan kesadaran pengguna terhadap pola kebiasaan belajar yang sedang dijalani. Temuan ini menunjukkan bahwa fitur penilaian konsistensi memiliki potensi besar untuk diterapkan dalam aplikasi *Habit Tracker* .		

Pertanyaan 2

Apakah informasi tentang tingkat konsistensi belajar dapat membantu Anda melakukan evaluasi diri? Mengapa?		
Kode	**Isi**	**Jumlah**
1	Ya, sangat membantu	74
Kesimpulan: Berdasarkan hasil jawaban responden, mahasiswa menilai bahwa informasi mengenai tingkat konsistensi belajar sangat membantu dalam melakukan evaluasi diri. Informasi tersebut memungkinkan pengguna untuk memahami pola kebiasaan belajar yang telah dijalani, sehingga dapat menjadi dasar untuk memperbaiki atau mempertahankan kebiasaan belajar ke arah yang lebih konsisten.		

Pertanyaan 3

Menurut Anda, apakah pesan otomatis seperti "kebiasaan belajar Anda cukup konsisten" dapat memengaruhi semangat belajar Anda? Jelaskan.		
Kode	**Isi**	**Jumlah**
1	Ya	69
2	Tidak juga	5
Kesimpulan: Berdasarkan hasil jawaban responden, sebagian besar mahasiswa menyatakan bahwa pesan otomatis yang memberikan umpan balik positif mengenai konsistensi belajar dapat memengaruhi semangat belajar mereka. Pesan semacam ini berfungsi sebagai penguatan psikologis yang mendorong pengguna untuk mempertahankan kebiasaan belajar. Temuan ini menunjukkan bahwa pesan otomatis memiliki potensi sebagai fitur motivasional dalam aplikasi *Habit Tracker*.		

Pertanyaan 4

Jika aplikasi dapat memberikan saran belajar berdasarkan kebiasaan Anda, apakah Anda tertarik menggunakannya? Mengapa?		
Kode	**Isi**	**Jumlah**
1	Ya	73
2	Tidak	1
Kesimpulan: Berdasarkan hasil jawaban responden, sebagian besar mahasiswa menyatakan tertarik menggunakan aplikasi yang mampu memberikan saran belajar berdasarkan kebiasaan pengguna. Temuan ini menunjukkan bahwa fitur rekomendasi personal dinilai relevan dan bermanfaat untuk membantu pengguna meningkatkan kualitas dan konsistensi belajar. Hal ini menguatkan potensi penerapan sistem rekomendasi berbasis data dalam aplikasi *Habit Tracker* .		

Pertanyaan 5

Menurut Anda, apa manfaat utama dari fitur yang dapat menilai atau memprediksi konsistensi belajar mahasiswa?		
Kode	**Isi**	**Jumlah**
1	Sebagai evaluasi/*Reminder*	70
2	Tidak ada	2
Kesimpulan: Berdasarkan hasil jawaban responden, sebagian besar mahasiswa menilai bahwa fitur yang dapat menilai atau memprediksi konsistensi belajar bermanfaat sebagai alat evaluasi dan pengingat. Fitur ini membantu pengguna memahami kondisi kebiasaan belajarnya serta mendorong perbaikan perilaku belajar secara berkelanjutan. Temuan ini menunjukkan bahwa fitur prediksi konsistensi memiliki nilai praktis yang tinggi dalam mendukung proses belajar mahasiswa.		

Kesimpulan Tema 5

Berdasarkan hasil analisis jawaban responden pada Tema Persepsi terhadap Fitur Prediksi Konsistensi, dapat disimpulkan bahwa mahasiswa memberikan respon yang sangat positif terhadap fitur yang mampu menilai dan memprediksi konsistensi belajar. Fitur tersebut dinilai membantu pengguna memahami kondisi kebiasaan belajar, melakukan evaluasi diri, serta meningkatkan motivasi melalui pesan otomatis dan umpan balik yang bersifat personal.

Selain itu, responden juga menunjukkan ketertarikan tinggi terhadap fitur rekomendasi belajar berbasis kebiasaan, yang dinilai relevan untuk mendukung konsistensi belajar. Manfaat utama dari fitur prediksi konsistensi adalah sebagai alat evaluasi dan pengingat, sehingga membantu mahasiswa memperbaiki dan mempertahankan kebiasaan belajar secara berkelanjutan. Temuan ini menguatkan potensi penerapan algoritma *Random Forest* dalam aplikasi *Habit Tracker* sebagai sistem prediksi dan rekomendasi kebiasaan belajar.

FAKULTAS ILMU KOMPUTER
UNIVERSITAS PRABUMULIH

KESEDIAAN SEBAGAI PEMBIMBING SKRIPSI

PROGRAM STUDI INFORMATIKA

Yang bertanda tangan di bawah ini:

Nama	: Iwan Setiawan, S.Kom., M.Kom.
NUPTK	: 1541769670130303
Program Studi	: Informatika
Fakultas	: Ilmu Komputer

Bersedia menjadi pembimbing I Skripsi, dan

Nama	: Nur Aini H, S.Kom., M.Si., M.Kom.
NUPTK	: 2434764664300022
Program Studi	: Informatika
Fakultas	: Ilmu Komputer

Bersedia menjadi pembimbing II Skripsi

dari mahasiswa Program Studi Informatika Fakultas Ilmu Komputer Universitas Prabumulih di bawah ini:

Nama	: Windi Yulinda
NIM	: 2022230015
Judul Skripsi	: Implementasi Prinsip *Atomic Habits* dan Algoritma *Random Forest* pada Aplikasi *Habit Tracker*

Demikian surat kesediaan membimbing ini kami buat, atas perhatiannya kami ucapkan terima kasih.

Prabumulih, Juni 2026

Dosen Pembimbing I	Dosen Pembimbing II
(Iwan Setiawan, S.Kom., M.Kom.)	(Nur Aini H, S.Kom., M.Si., M.Kom.)
NUPTK.1541769670130303	NUPTK. 2434764664300022

Jalan Patra No.50 RT.01 RW.03 Kelurahan Sukaraja Kecamatan Prabumulih Selatan
Kota Prabumulih Sumatera Selatan Indonesia Telpon 0713-3224417 Fax 0713-322418
e-mail : mail@unpra.ac.id

Dipindai dengan CamScanner

FAKULTAS ILMU KOMPUTER
UNIVERSITAS PRABUMULIH

KARTU BIMBINGAN SKRIPSI

AMA : WINDI YULINDA
M : 2022230015
OGRAM STUDI : INFORMATIKA
AMA PEMBIMBING I : IWAN SETIAWAN, S.Kom., M.Kom
DUL : IMPLEMENTASI PRINSIP *ATOMIC HABITS* DAN ALGORITMA *RANDOM FOREST* PADA APLIKASI *HABIT TRACKER*

NO	TGL. KONSULTASI	TOPIK POKOK YANG DIBICARAKAN	TANDA TANGAN PEMB. I	TGL MENG-HADAP KEMBALI
1	10-2-2026	BAB I ACC		
		BAB II		
		font gambar		
		font tabel		
		BAB III		
		Tabel 3.1		
		font tabel & gambar		
2	10-2-2026	BAB II ACC		
		BAB III ACC		

lai Bimbingan : 10-2-2026
as Akhir Bimbingan :

Mengetahui Kaprodi Informatika

IWAN SETIAWAN, S.Kom.M.Kom.

" Menjadi fakultas yang berkompetensi teknologi informasi yang unggul di tingkat nasional "

Dipindai dengan CamScanner

FAKULTAS ILMU KOMPUTER
UNIVERSITAS PRABUMULIH

KARTU BIMBINGAN SKRIPSI

NAMA : WINDI YULINDA
NIM : 2022230015
PROGRAM STUDI : INFORMATIKA
NAMA PEMBIMBING I : IWAN SETIAWAN, S.Kom., M.Kom
JUDUL : IMPLEMENTASI PRINSIP *ATOMIC HABITS* DAN ALGORITMA *RANDOM FOREST* PADA APLIKASI *HABIT TRACKER*

NO	TGL. KONSULTASI	TOPIK POKOK YANG DIBICARAKAN	TANDA TANGAN PEMB. I	TGL MENG-HADAP KEMBALI
3	30-3-2026	BAB IV a. kebutuhan pengguna b. Soft & hard		
4	14-4-2026	BAB IV a. Tabel Prangkat Lunak dan teras pengguna		
5	14/4/2026	BAB IV ACC		
6.	20/5/2026	BAB V a. Bab IV harus ada b. prangkat keras & lunak c. lyn pesiward d. Integrasi e. penjelasan gambar f. perfijian		

Mulai Bimbingan :
Batas Akhir Bimbingan :

Mengetahui Kaprodi Informatika

IWAN SETIAWAN, S.Kom.M.Kom.

Dipindai dengan CamScanner

FAKULTAS ILMU KOMPUTER
UNIVERSITAS PRABUMULIH

KARTU BIMBINGAN SKRIPSI

NAMA : **WINDI YULINDA**
NIM : **2022230015**
PROGRAM STUDI : **INFORMATIKA**
NAMA PEMBIMBING I : **IWAN SETIAWAN, S.Kom., M.Kom**
JUDUL : **IMPLEMENTASI PRINSIP *ATOMIC HABITS* DAN ALGORITMA *RANDOM FOREST* PADA APLIKASI *HABIT TRACKER***

NO	TGL. KONSULTASI	TOPIK POKOK YANG DIBICARAKAN	TANDA TANGAN PEMB. I	TGL MENG-HADAP KEMBALI
7	25/5/2026	BAB V		
		a. perangkat lunak		
		b. selesaikan pengujian		
8.	3/6/2026	BAB V ACC		
		BAB VI		
		a. kesimpulan		
9	4/6/2026	BAB VI ACC		
10	9/6/2026	kelengkapan SIDANG		

Mulai Bimbingan :
Batas Akhir Bimbingan :

Mengetahui Kaprodi Informatika

IWAN SETIAWAN, S.Kom.M.Kom.

" Menjadi fakultas yang berkompetensi teknologi informasi yang unggul di tingkat nasional "

Dipindai dengan CamScanner

FAKULTAS ILMU KOMPUTER
UNIVERSITAS PRABUMULIH

KARTU BIMBINGAN SKRIPSI

NAMA : **WINDI YULINDA**
NIM : **2022230015**
PROGRAM STUDI : **INFORMATIKA**
NAMA PEMBIMBING II : **NUR AINI H, S.Kom., M.Kom**
JUDUL : **IMPLEMENTASI PRINSIP *ATOMIC HABITS* DAN ALGORITMA *RANDOM FOREST* PADA APLIKASI *HABIT TRACKER***

NO	TGL. KONSULTASI	TOPIK POKOK YANG DIBICARAKAN	TANDA TANGAN PEMB. II	TGL MENG-HADAP KEMBALI
1	02/02/2026	Bab 1		03/02/2026
		Revisi : - Latar Belakang		
		- Rumusan Masalah		
2	04/02/2026	Bab 1 ACC		05/02/2026
		Bab 2 : Cara Penyelesaian		
3	06/02/2026	ACC Bab II		09/02/2026
4	11/03/2026	Bab III		
		Revisi - Objek		12/02/2026
		- teori UML		
		- Lihat kuisioner		
5	12/02/2026	ACC bab III		
6	25/02/2026	Rancangan Tampilan Aplikasi		26/02/2026
		Revisi		

Mulai Bimbingan :
Batas Akhir Bimbingan :

Mengetahui Kaprodi Informatika

IWAN SETIAWAN, S.Kom.M.Kom.

" Menjadi fakultas yang berkompetensi teknologi informasi yang unggul di tingkat nasional "

Dipindai dengan CamScanner

FAKULTAS ILMU KOMPUTER
UNIVERSITAS PRABUMULIH

KARTU BIMBINGAN SKRIPSI

NAMA : WINDI YULINDA
NIM : 2022230015
PROGRAM STUDI : INFORMATIKA
NAMA PEMBIMBING II : NUR AINI H, S.Kom., M.Kom
JUDUL : IMPLEMENTASI PRINSIP *ATOMIC HABITS* DAN ALGORITMA *RANDOM FOREST* PADA APLIKASI *HABIT TRACKER*

NO	TGL. KONSULTASI	TOPIK POKOK YANG DIBICARAKAN	TANDA TANGAN PEMB. II	TGL MENG-HADAP KEMBALI
7	30/3/2026	Bab IV		
		Revisi : Desain tpilan aplikasi		
		Database		
8	20/4/2026	Revisi . Desain tpilan aplikasi		
9.	29/4/2026	Acc Bab IV		10 mei 2026
10.	8/5/2026	Bab V		
		Revisi : - Spesifikasi Hardware		
		- PrototYp		
		- perjelas fitur		
		- Output.		
11.	29/5/2026	Revisi : perjelas fitur output		

Mulai Bimbingan :
Batas Akhir Bimbingan :

Mengetahui Kaprodi Informatika

IWAN SETIAWAN, S.Kom.M.Kom.

" Menjadi fakultas yang berkompetensi teknologi informasi yang unggul di tingkat nasional "

Dipindai dengan CamScanner

FAKULTAS ILMU KOMPUTER
UNIVERSITAS PRABUMULIH

KARTU BIMBINGAN SKRIPSI

NAMA : **WINDI YULINDA**
NIM : **2022230015**
PROGRAM STUDI : **INFORMATIKA**
NAMA PEMBIMBING II : **NUR AINI H, S.Kom., M.Kom**
JUDUL : **IMPLEMENTASI PRINSIP *ATOMIC HABITS* DAN ALGORITMA *RANDOM FOREST* PADA APLIKASI *HABIT TRACKER***

NO	TGL. KONSULTASI	TOPIK POKOK YANG DIBICARAKAN	TANDA TANGAN PEMB. II	TGL MENG-HADAP KEMBALI
12	25/5/2026	Acc Bab V		
13	26/5/2026	Bab VI		
		Revisi : Kopi		
14.	3/6/2026	Acc Bab VI		
15.	9/6/2026	Kelengkapan		

Mulai Bimbingan :
Batas Akhir Bimbingan :

Mengetahui Kaprodi Informatika

IWAN SETIAWAN, S.Kom.M.Kom.

" Menjadi fakultas yang berkompetensi teknologi informasi yang unggul di tingkat nasional "

Dipindai dengan CamScanner

FAKULTAS ILMU KOMPUTER
UNIVERSITAS PRABUMULIH

PENILAIAN REVIEW SKRIPSI

Nama : Windi Yulinda

NIM : 2022230015

Judul Skripsi : Implementasi Prinsip *Atomic Habits* dan Algoritma *Random Forest* pada Aplikasi *Habit Tracker*

Hari/Tanggal : Senin, 22 - 6 - 2026

Saran dan masukan :

- Sistematika Penulisan - Metode Pengembangan ?
- Referensi
- Prototype : - Actor < Admin User
- Feed Back : Informasi kepada User (Gold, Perak, Perunggu)
- Sebagai contoh PP (Strava)

ACC Revisi GIAP Jilid

03/07/2026

Rekomendasi*:

a. Dilanjutkan tanpa perbaikan
b. Dilanjutkan dengan perbaikan
c. Ditolak

Prabumulih, 03 - 7 - 2026
Dosen Penguji

(Khana Wijaya, S.Kom., M.Kom)
NUPTK.7146768669130363

*pilih salah satu

CS Dipindai dengan CamScanner

FAKULTAS ILMU KOMPUTER
UNIVERSITAS PRABUMULIH

PENILAIAN REVIEW SKRIPSI

Nama : Windi Yulinda

NIM : 2022230015

Judul Skripsi : Implementasi Prinsip *Atomic Habits* dan Algoritma *Random Forest* pada Aplikasi *Habit Tracker*

Hari/Tanggal : SENIN / 22-6-2026

Saran dan masukan :

IKUTI ARAHAN PENGUJI DENGAN KONSULTASI KE PEMBIMBING DAHULU

ACC SIDANG ULANG

24/7 2026

Rekomendasi*:

a. Dilanjutkan tanpa perbaikan
b. Dilanjutkan dengan perbaikan
c. Ditolak

Prabumulih, 22-6-2026
Pembimbing I

(Iwan Setiawan, S.Kom., M.Kom)
NUPTK. 1541769670130303

pilih salah satu

Dipindai dengan CamScanner

FAKULTAS ILMU KOMPUTER
UNIVERSITAS PRABUMULIH

PENILAIAN REVIEW SKRIPSI

Nama : Windi Yulinda

NIM : 2022230015

Judul Skripsi : Implementasi Prinsip *Atomic Habits* dan Algoritma *Random Forest* pada Aplikasi *Habit Tracker*

Hari/Tanggal : SENIN / 22 JUNI 2026 .

Saran dan masukan :

Siap Jilid
16/6/2026.

Rekomendasi*:

a. Dilanjutkan tanpa perbaikan
b. Dilanjutkan dengan perbaikan
c. Ditolak

Prabumulih, 22 / 06 / 2026
Pembimbing II

(Nur Aini H, S.Kom., M.Si., M.Kom.)
NUPTK. 2434764664300022

**pilih salah satu*

atra No.50 RT.01 RW.03 Kelurahan Sukaraja Kecamatan Prabumulih Selatan
rabumulih Sumatera Selatan Indonesia Telpon 0713-3224417 Fax 0713-322418
mail@unpra.ac.id

Dipindai dengan CamScanner