**Open-Source Coding (Intermediate)**

**OPSC6312**

**POE**

**Part 1**

**Rahil Sirkissoon**

**ST10388332**

[Research 1](#_Toc156582586)

[Introduction 1](#_Toc1242009365)

[Research on Existing Habit Tracker Apps 1](#_Toc253841681)

[HabitNow 1](#_Toc1114676020)

[Loop/Habit Rewards Habit Tracker 1](#_Toc1118395804)

[Grow/ Habit Forest Habit Tracker 1](#_Toc629338708)

[Comparison of all 3 apps 1](#_Toc636492960)

[Best Features to use in HabitMate 2](#_Toc951110504)

[Conclusion 2](#_Toc1091813983)

[References 2](#_Toc124208475)

[Planning and Design 2](#_Toc317699264)

[App Overview 2](#_Toc1606777852)

[Innovative Features 2](#_Toc1210624639)

[Requirements 2](#_Toc1405754420)

[Additional Features 2](#_Toc1819301908)

[User Interface Design 2](#_Toc602683402)

[Navigation Flow Diagram 2](#_Toc1789734948)

[API Design for Habit Tracker App 2](#_Toc645342015)

[UML Diagram 5](#_Toc2084881973)

[Data Model and Types 5](#_Toc530125229)

[Gantt Chart 6](#_Toc1399555084)

[Conclusion 6](#_Toc1534312915)

[References 6](#_Toc271438064)

[POE Part 2 6](#_Toc68273688)

[Use of AI Tools in the Development of the Habit Tracker App 6](#_Toc1740893710)

[GitHub Links 6](#_Toc1837128146)

# Research

## Introduction

We live in a fast-paced world with personal productivity and consistency are crucial to achieving goals. Habit tracking applications help users build healthy routines by recording their daily actions and providing feedback on progress. For this project I will be developing a mobile application called HabitMate, a habit tracker designed to help users build positive habits, break the bad ones and monitor their consistency over time.

To better understand the functionality, user experience and technologies required to build this app, I have analyzed three existing android-based habit tracker apps: Habit Now, Loop Habit Tracker and Grow – Habit Tracker. This report compares their features, strengths and weaknesses and evaluates how they may have been developed based on knowledge of android studio. These findings will guide the design and functionality of HabitMate.

## Research on Existing Habit Tracker Apps

### HabitNow

Overview

HabitNow is a well-structured productivity and habit tracking app that combines both task management and daily tracking. It allows users to organize tasks by category and track habits through detailed stats and reminders

Strengths

* Easy to use interface with an option to change to dark mode
* Categorization of habits
* Daily task style checklist format that is intuitive
* Offers reminders and notifications
* Habit performance stats and streaks are available

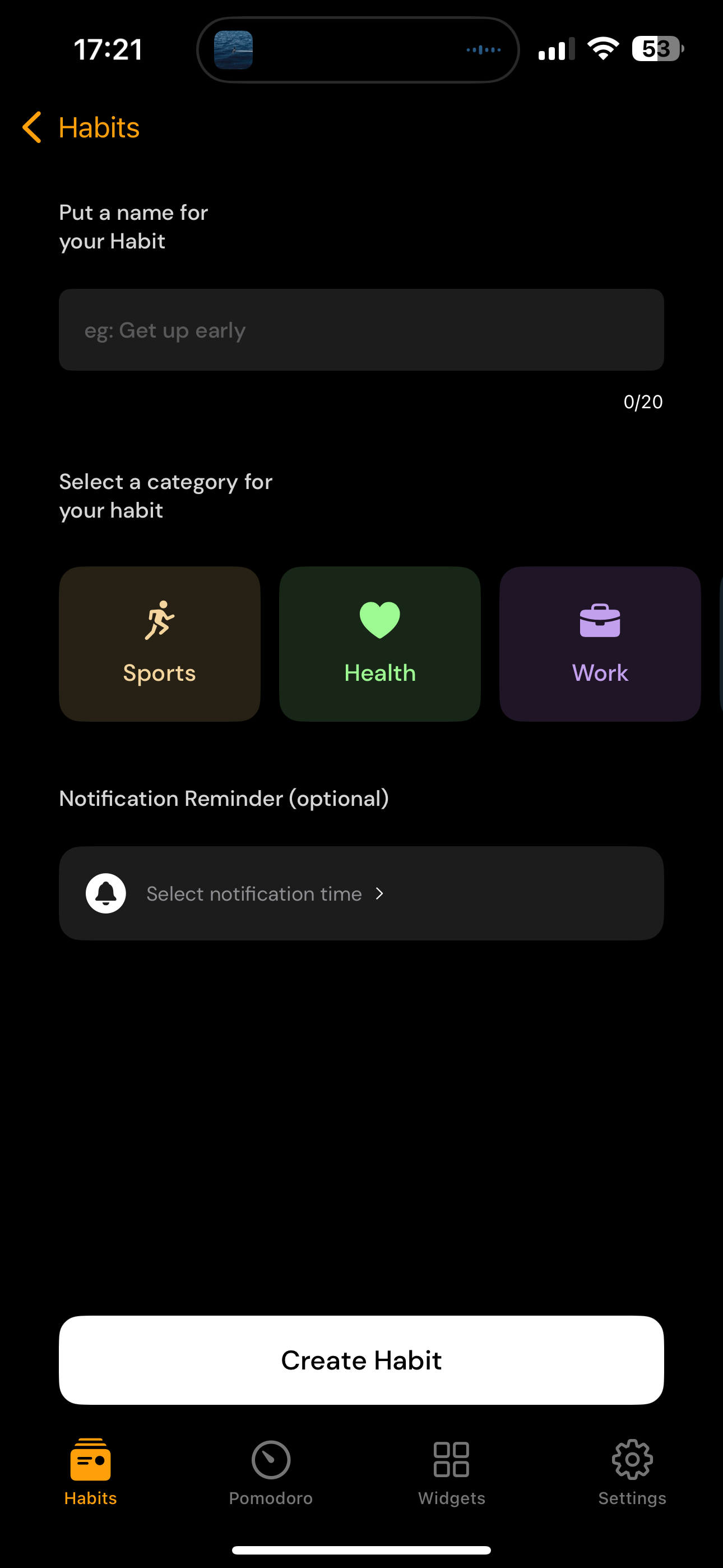
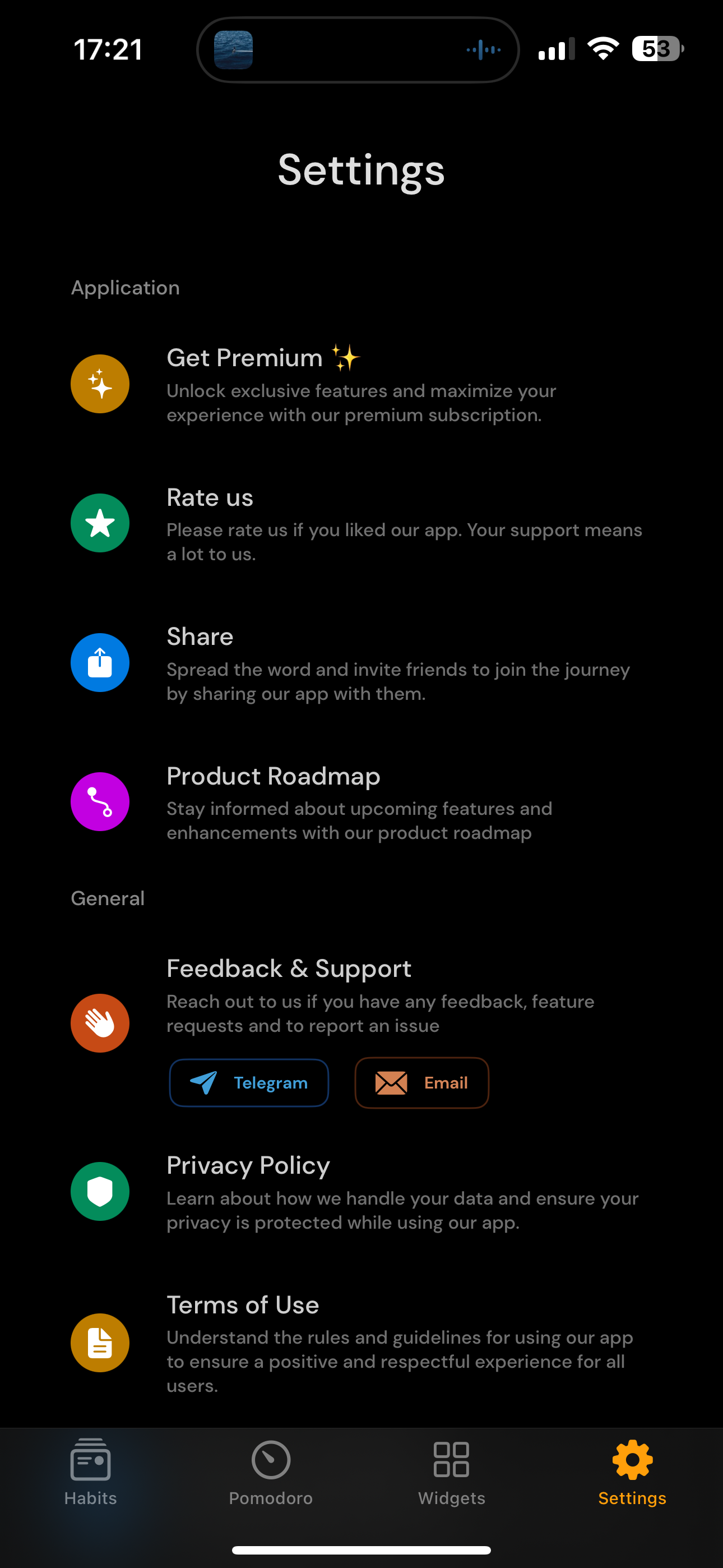
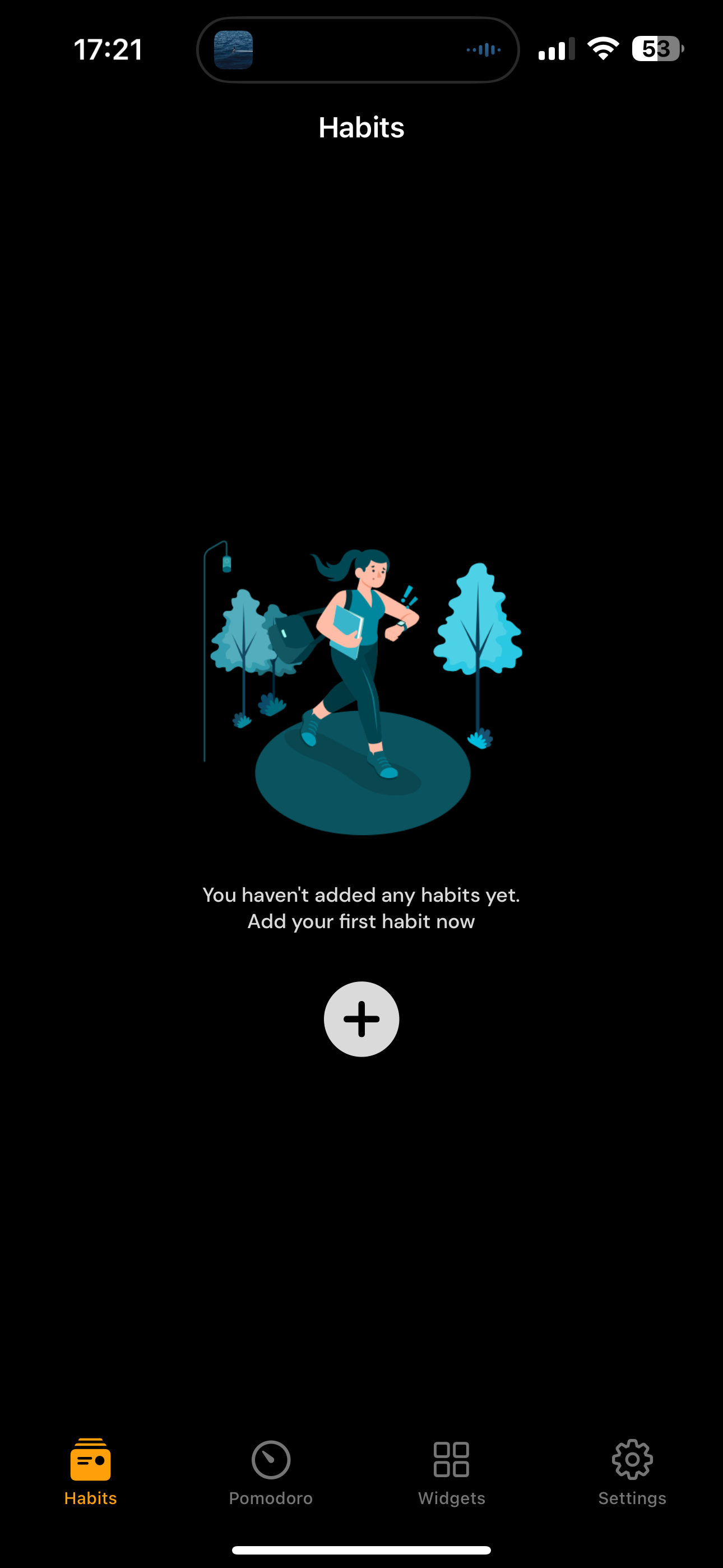
Weaknesses

* Many features are locked behind a paywall
* Ads in the free version can interrupt user experience
* No cloud sync for users on the free option

How I think the app was implemented

Based on my android development knowledge, HabitNow likely uses Room DB or SQLite for local data storage, Alarm manager or workmanager for scheduling reminders and RecyclerView for listing habits. It may use SharedPreferences for settings such as the theme. It likely adopts an MVVM architecture using ViewModel and LiveData to handle state changes

Screenshots of the App



### Loop/Habit Rewards Habit Tracker

Overview

Loop is a lightweight, open-source habit tracker that focuses on simplicity, privacy and performance. It offers data visualization and streak tracking while being ad free and fully offline

Strengths

* Completely free and open source
* No ads or internet required
* Visual habit analytics and flexible schedule options
* Export import habit data as a CSV
* High performance and low battery usage

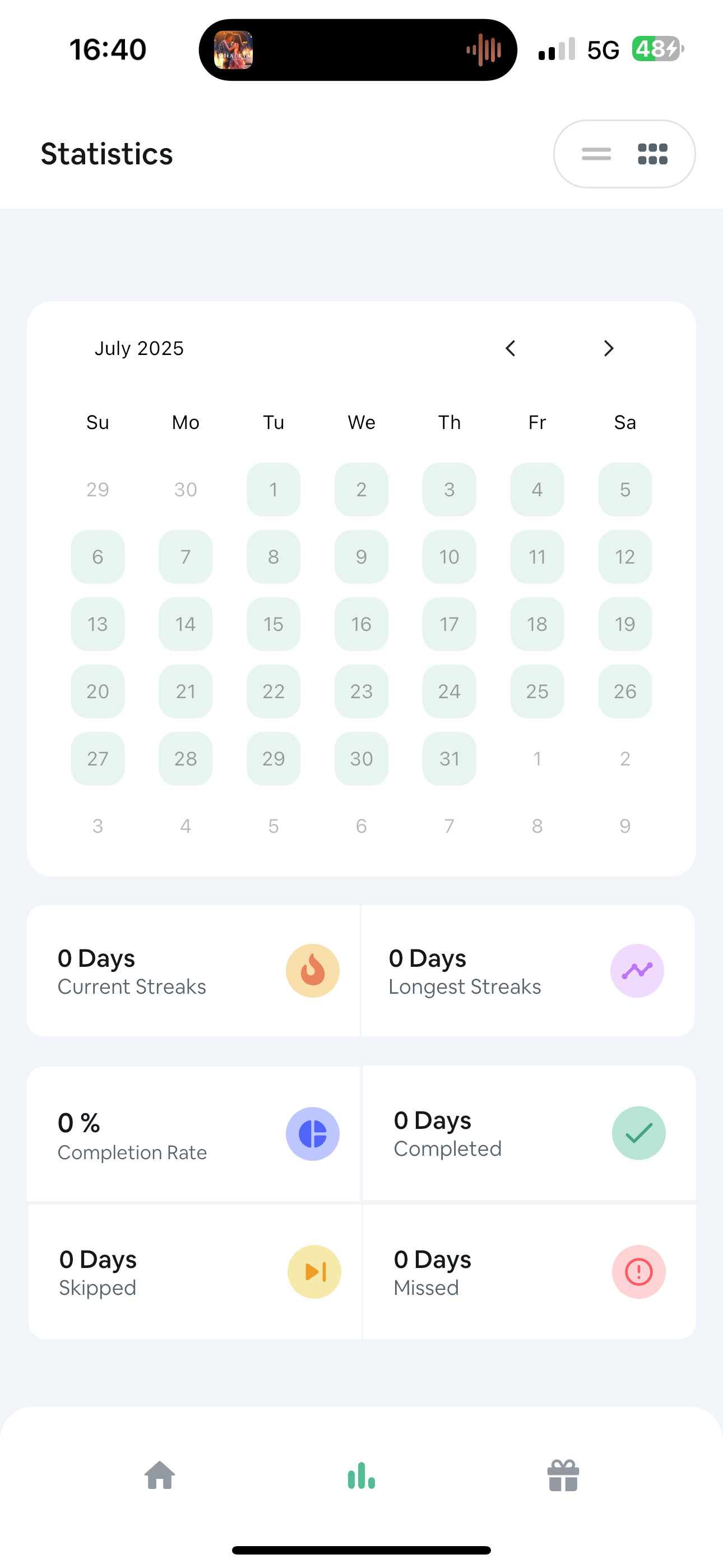
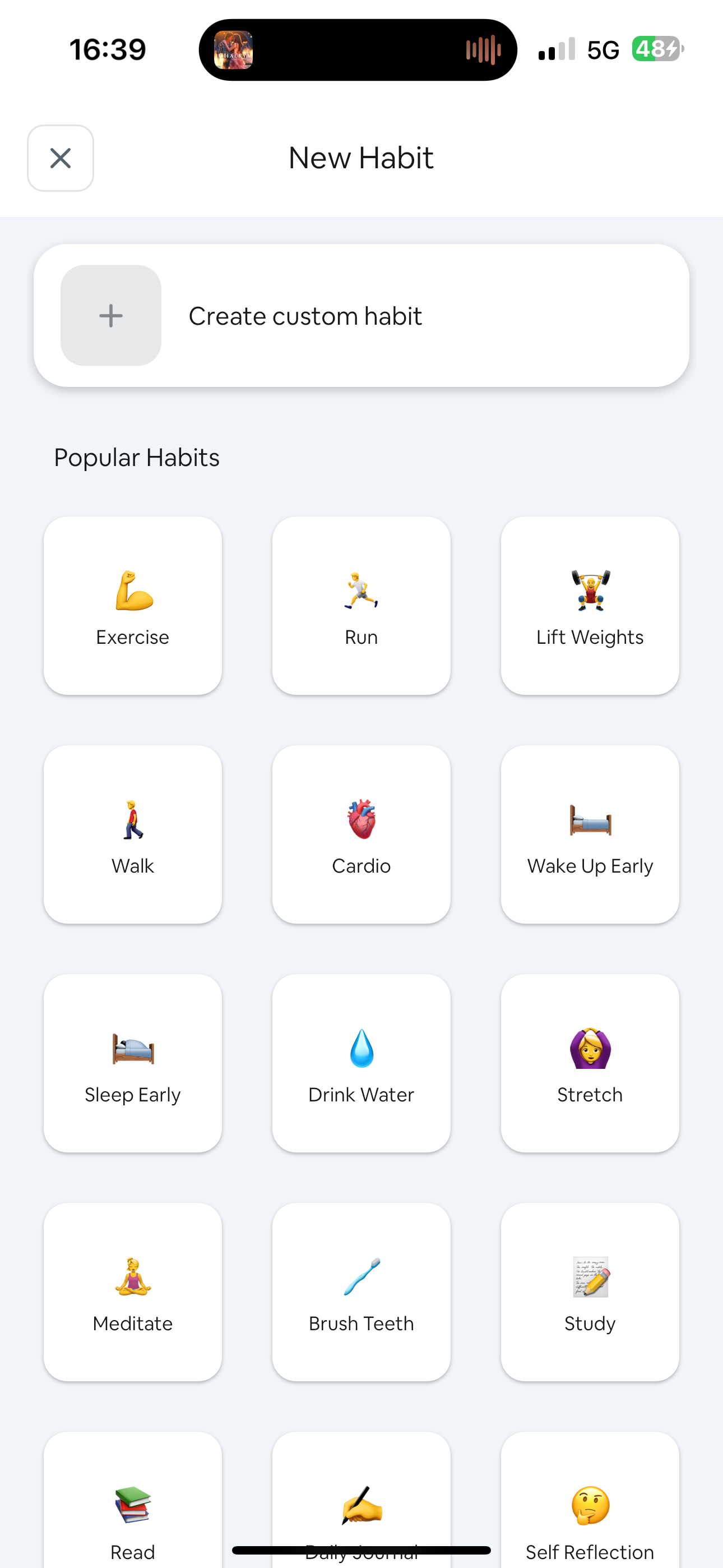
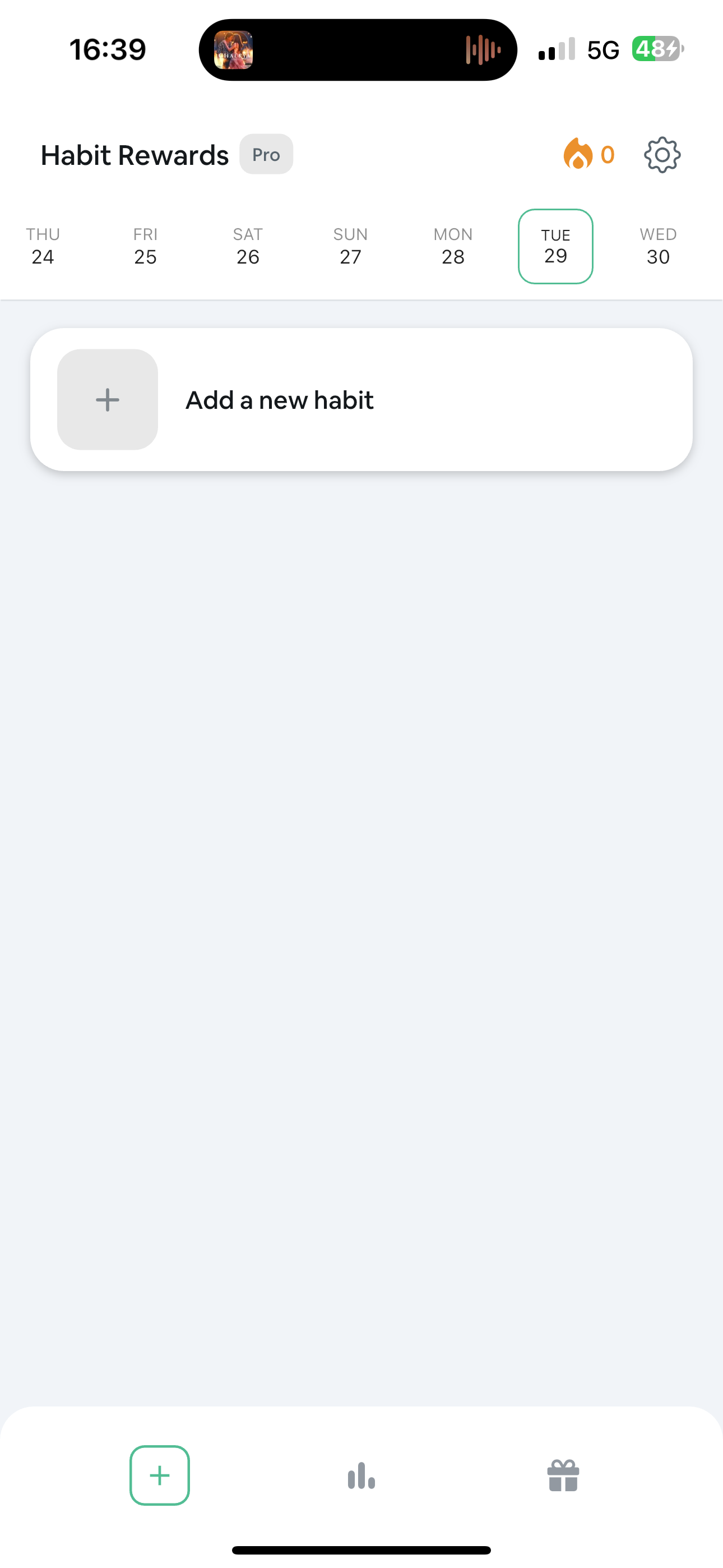
Weaknesses

* No login or cloud backup
* Very basic UI
* No notification customization beyond basic reminders

How I think the app was implemented

As an open-source project, Loop likely uses Room DB for storing habits locally. LiveData is used for real time updates and MPAndroidChart for generating habit graphs. The app uses ConstraintLayoiut and Material components for UI and follows MVVM architecture. The simplicity implies that background work is managed efficiently using WorkManager

Screenshots



### Grow/ Habit Forest Habit Tracker

Overview

Grow takes a gamified approach to habit tracking by using a virtual plant that grows as the user completes their habits. It aims to motivate users by turning daily habits into a fun challenge

Strengths

* Unique gamified design
* Appealing for younger audiences or casual users
* Offers cloud backup and login via firebase
* Includes motivational messages and animations

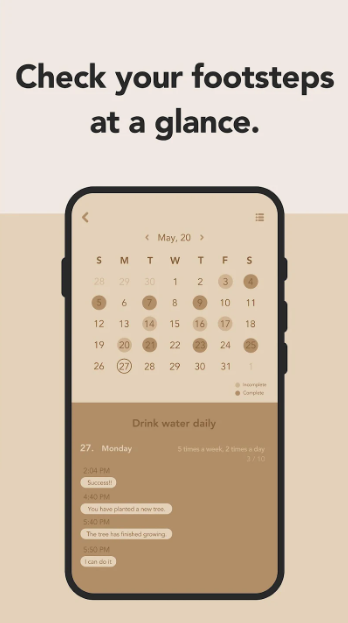
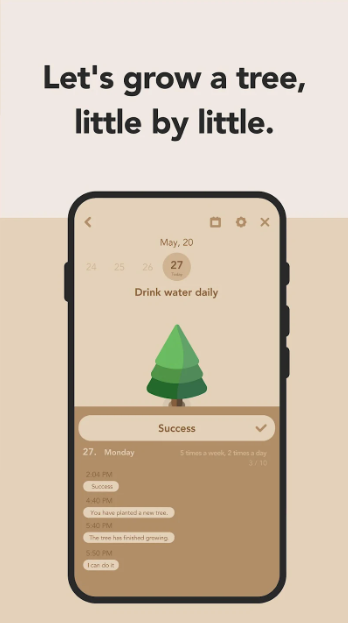
Weaknesses

* Less minimalist and can feel cluttered
* Limited control over plant design or customization, but still an innovative feature
* Some features are locked to a premium version
* No offline mode

How I think the app was implemented

Grow likely uses Firebase Authentication and Firebase Realtime Database to store user habits in the cloud. Gamification is likely achieved through custom animations, canvas drawing and Glide for loading animated visuals. Reminders are managed using AlarmManager and firebase Crashlytics is likely used for monitoring bugs. It may also use jetpack compose for newer UI elements,

Screenshots



## Comparison of all 3 apps

|  |  |  |  |
| --- | --- | --- | --- |
| **Feature** | **HabitNow** | **Loop Tracker** | **Grow – Habit Tracker** |
| Offline Use | Yes | Yes | No |
| Cloud Sync | Premium only | No | Yes |
| Notifications | Yes | Yes | Yes |
| Gamification | No | No | Yes |
| Analytics/Charts | Yes | Yes | No |
| UI | Task based and neat | Minimalistic | Visual and playful |
| Ads (in free version) | Yes | No | Yes |
| Export/Import Data | No | Yes | No |
| Firebase Integration | No | No | Yes |

## Best Features to use in HabitMate

1. Custom habit creation: users should be able to add and edit habits
2. Streak tracking: display progress visually to encourage consistency and enhance user experience
3. Daily checklist UI: like HabitNow a to do style habit list is intuitive
4. Reminders and notifications: to prompt user action daily
5. Data visualization: integrate a simple progress chart like Loop
6. Firebase Sync: for backing up user data
7. Simple and Clean UI: focus on usability

These features will strike a balance between practicality, engagement and usability. HabitMate will remain ad free and have a firebase optional login to appeal to both offline and cloud preferring users

## Conclusion

The analysis of HabitNow, Loop and Grow highlights the diverse approaches developers take when designing habit tracking apps. Each app has unique strengths: HabitNow excels in task management, Loop in minimalism and performance and Grow in gamified motivation

The key takeaway is that a successful habit tracker should be simple, reliable and visually motivating. For HabitMate, I will aim to blend the strengths and innovative features from the apps to create a balanced experience that empowers users to build and track their habits.

## References

* HabitNow. (2024). *HabitNow: Daily Routine Planner*. Google Play Store. <https://play.google.com/store/apps/details?id=com.habitnow>
* Loop. (2024). *Loop Habit Tracker*. Google Play Store. <https://play.google.com/store/apps/details?id=org.isoron.uhabits>
* Grow. (2024). *Grow – Habit Tracker*. Google Play Store. <https://play.google.com/store/apps/details?id=com.growapp>
* Android Developers. (2024). *Build your first app*. <https://developer.android.com/training/basics/firstapp>
* Firebase. (2024). *Firebase for Android.* <https://firebase.google.com/docs/android/setup>

# Planning and Design

App Name: HabitMate

App Icon:



## App Overview

HabitMate is a habit tracking mobile application designed to help users build and maintain healthy routines while breaking bad ones. The app provides a daily checklist interface, motivational reminders and visual progress tracking. It aims to be minimal, efficient and motivating without overwhelming the user with a cluttered interface.

## Innovative Features

* Gamification: Streak system and digital rewards for consistency
* Cloud Sync (Optional): firebase authentication for data sync
* Minimalist Offline Mode: Fully functional without requiring internet access
* Progress Analytics: weekly and monthly performance charts using MPAndroidChart
* Custom Scheduling: Habits can be repeated on specific days e.g. (Monday, Wednesday, Friday)

## Requirements

Core Functional Requirements

1. User Account System (Optional)

* Users can register/login with firebase authentication
* Guest access available for offline users

1. Habit Management

* Add/edit/delete habits
* Set habit frequency (daily, weekly, custom)
* Mark as complete for each day

1. Habit Progress Tracking

* Display current streak, best streak
* Calendar view of completion history
* Graphs to visualize performance

1. Reminders and Notifications

* Push notifications for daily habits
* Set custom reminders per habit

1. Offline Functionality

* All features work offline
* Data saved locally to Room DB

1. Backup and Sync

* Firebase Realtime database used for syncing across devices

## Additional Features

1. Gamification System

* Earn badges for milestones (1 week, 1-month etc.)
* Motivational messages

1. Themes

* Light/Dark mode toggle option

1. Export/Import Data

* Users can export their data to a CSV for backup analysis

## User Interface Design

1. Welcome Screen

* Purpose: first time user entry point. Buttons for sign up and login or continue as a guest
* Navigation: leads to dashboard or login flow

1. Dashboard Screen

* Purpose: displays todays habits in a checklist form
* Features: Mark as done, view current streak, edit and delete
* Navigation: Add habit – habit form – view stats – analytics screen

1. Add/Edit Habit Screen

* Purpose: allows users to create or modify habits
* Fields: Name, frequency, icon, color, reminder time
* Navigation: back to dashboard

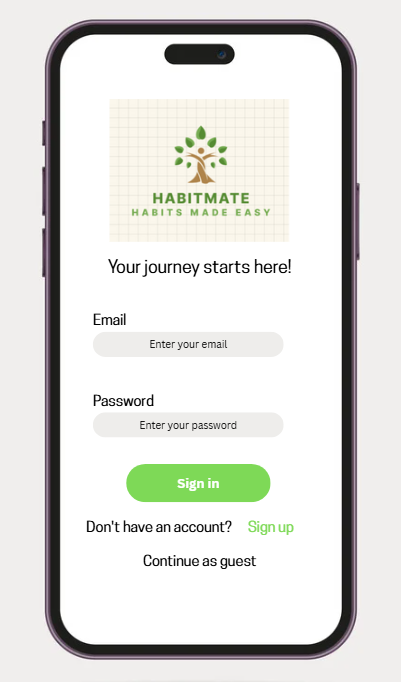
1. Analytics/Progress Screen

* Purpose: Graphs showing progress over time
* Navigation: back to dashboard

1. Settings/Profile Screen

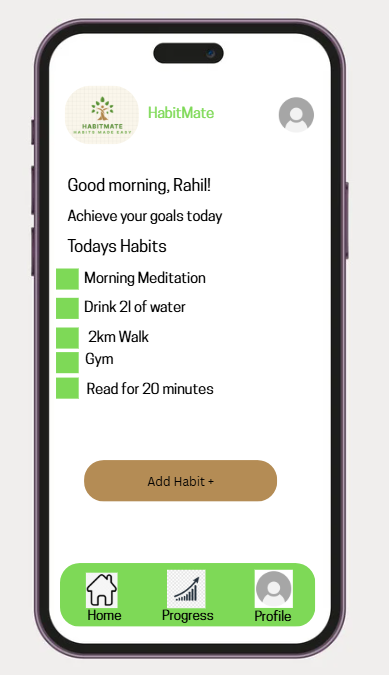
* Purpose: toggle dark mode, sync data, export habits, change language

Welcome/Login Page



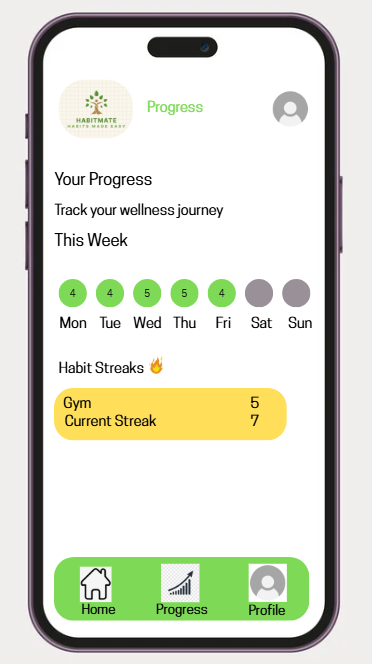
The login page welcomes the users and allows them to login with their email and password sign in if they don't have an account and continue as a guest if they want to use an offline mode, the page consists of a minimalist and welcoming design.

Main Dashboard



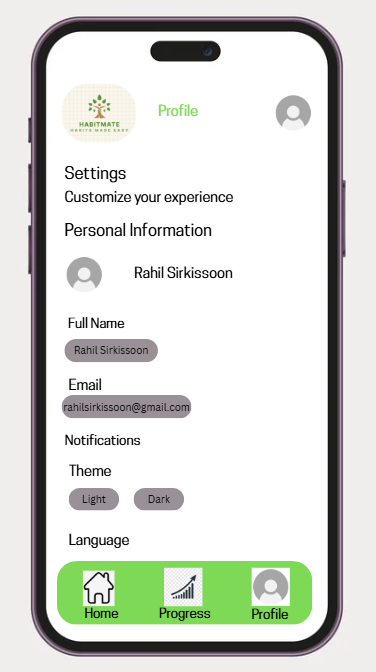
The dashboard is the main page; it allows users to manage their habitsas well as create new ones. Their daily habits are in checklist form so users can easily check which habits they have completed throughout the day. The user can also access all the other pages from the main dashboard such as the progress and profile page.

Progress Page



The progress page shows users their progress for the week and their habit streaks. It allows users to visually see their progress and keep uo with their habits.

Profile Page



The profile page allows users to customize their personal information as well as choose their persona preferences. The user is able to change the language, change the theme to light or dark mode as well as edit their name and email.

## Navigation Flow Diagram

## API Design for Habit Tracker App

1. Chosen API: Firebase Realtime Database

For this project, the habit tracker app uses the Firebase Realtime database RESTAPI. Firebase provides a cloud hosted NoSQL database that allows apps to read and write data in real time using JSON

The key reasons for this choice are:

* Ease of integration with android studio using firebase SDKs
* Real time syncing of data across multiple devices
* Authentication support through firebase authentication (email/password, google sign in etc.)
* Scalable and secure storage for structured habit and user data
* Free tier available, making it cost effective for development and testing

1. How the RESTAPI works

A REST API (representational state transfer) allows communication between the android app) the client\_ and the firebase database (the server)

* The app sends HTTP requests e.g. (GET, POST, PATCH, DELETE to firebase endpoints
* Firebase processes the request and responds with JSON data
* All habit and user data is stored in structured JSON nodes inside the database

Example firebase endpoint format:

https://<project-id>.firebaseio.com/users/<userId>/habits.json

* Each userID represents a unique user
* Inside each user node, multiple habits are stored
* The .json extension tells firebase we are using the REST API

1. API data flow in the habit tracker

Sending data

POST

Endpoint

POST https://<project-id>.firebaseio.com/users/<userId>/habits.json

Body

{

"habitName": "Drink Water",

"days": ["Monday", "Tuesday"],

"reminderTime": "08:00",

"color": "#3498db",

"icon": "water"

}

PATCH

Endpoint

PATCH https://<project-id>.firebaseio.com/users/<userId>/habits/<habitId>.json

Body

{

"progress": {

"2025-09-25": true

}

}

Recieving Data

GET

Endpoint

GET https://<project-id>.firebaseio.com/users/<userId>/habits.json

Response

{

"habit1": {

"habitName": "Drink Water",

"days": ["Monday", "Tuesday"],

"reminderTime": "08:00"

},

"habit2": {

"habitName": "Morning Run",

"days": ["Wednesday", "Friday"],

"reminderTime": "07:00"

}

}

1. Authentication and security

Firebase authentication ensures only authorized users access their own data

* Sign up/ login – handled by firebase authentication SDK
* Secure API calls – each request includes an auth token that verifies the users identify
* Rules – firebase security rules ensure one user cannot read or modify another user's habits

Example:

{

"rules": {

"users": {

"$userId": {

".read": "$userId === auth.uid",

".write": "$userId === auth.uid"

}

}

}

}

1. Hosting and future expansion

* Firebase hosting: if a web dashboard is added later, firebase hosting can serve a front end that uses the same REST API
* Scalability: as the app grows, more feattures such as push notifications, cloud functions and analytics can be integrated without changing the API structure

1. Benefits of using firebase REST API in this app

* Ensures real time updates: if a user marks a habit as complete on one device, all devices sync instantly
* Provides cross platform support: android, IOS and web apps can all use the same API
* Handles authentication and database together simplifying the development
* Scales easily from a single user to thousands

2. Chosen APIs

For this project HabitMate uses a custom Mock API for core habit tracking data and an external quotes API for motivational content

API Overview

|  |  |  |  |
| --- | --- | --- | --- |
| API Name | Purpose | Data Type | Hosting/Provider |
| Custom MOCKAPI | Core application/ habit data | JSON | MOCKAPI |
| External quotes API | Motivational messages | JSON | Zenquotes |

Custom Mock API for Habit Data

The mock API follows REST principles and is used to stimulate a production backend for all user specific habit operations (create,read,update,delete)

API Data flow

|  |  |  |  |
| --- | --- | --- | --- |
| Method | Endpoint | purpose | Example body request |
| POST | /users/{userId}/habits | Create a new habit | {"habitName": "Meditate", "days": ["Daily"], "reminderTime": "19:00", "color": "#2ecc71"} |
| GET | /users/{userId}/habits | Read all user habits | (No Body) |
| PATCH | /users/{userId}/habits/{habitId} | Update partial habit data | {"progress": {"2025-10-04": true}} |
| DELETE | /users/{userId}/habits/{habitId} | Delete a specific habit | (No Body) |

Example GET response (read all habits)

[{  
 "id": "h1",  
 "habitName": "Drink Water",  
 "days": ["Monday", "Tuesday"],  
 "reminderTime": "08:00",  
 "color": "#3498db"  
 },  
 {  
 "id": "h1",  
 "habitName": "Drink Water",  
 "days": ["Monday", "Tuesday"],  
 "reminderTime": "08:00",  
 "color": "#3498db"  
 },  
   
  
]

External quoted API

The external quotes API provides motivational content integrated into the application interface

‘API data flow for quotes

|  |  |  |  |
| --- | --- | --- | --- |
| Method | Endpoint | Purpose | Authentication |
| GET | /random | Fetch a single random quote | None |

Example GET response (random quote)

{

"\_id": "rJ8m",  
 "content": "The best way to predict the future is to invent it.",  
 "author": "Alan Kay",  
 "tags": ["inspirational", "technology"]  
 }

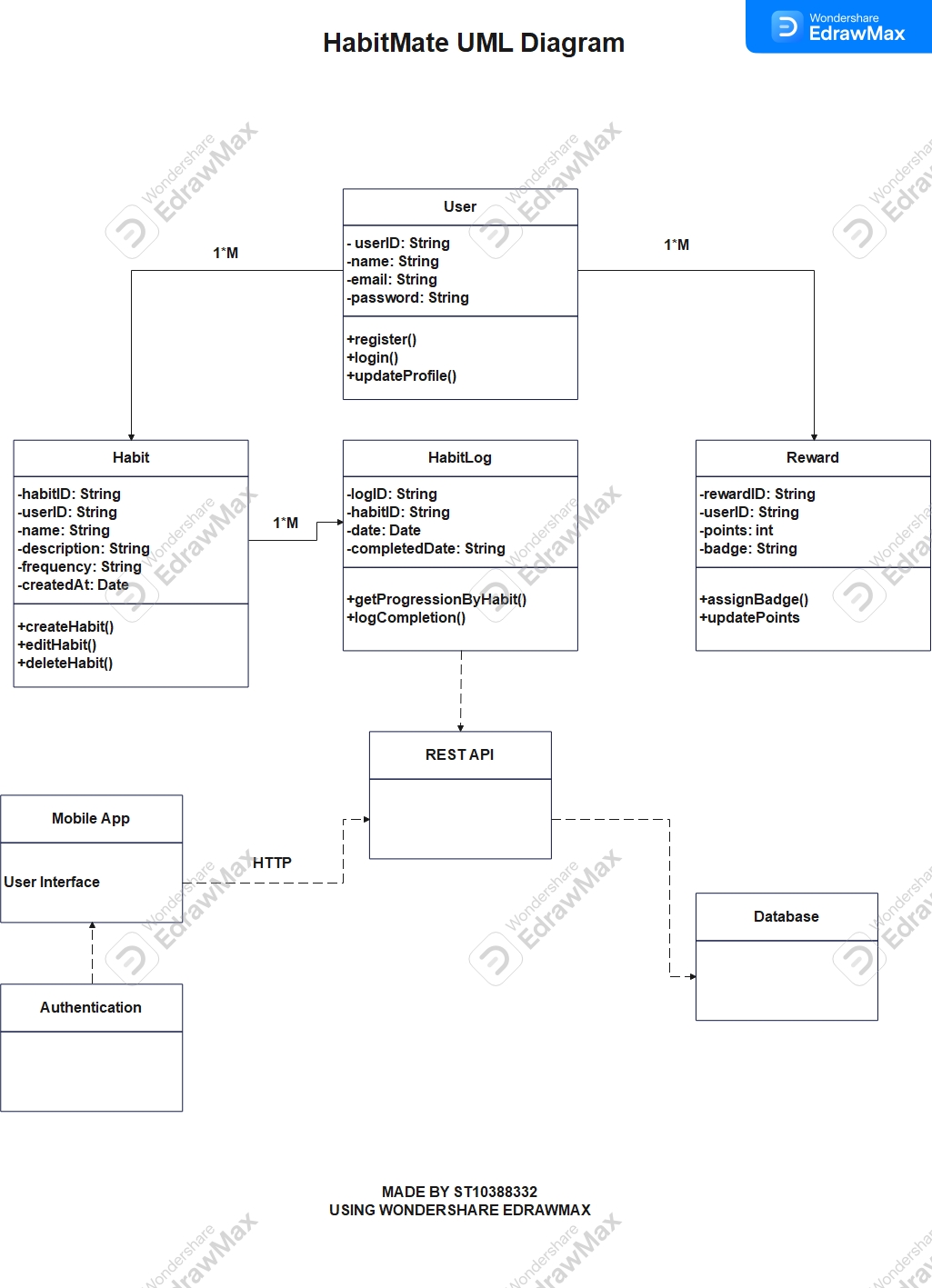
Authentication and Security

* Mock API: for this development phase, authentication is simplified or omitted. The main goal is to validaye the data contract. Im production, robust authentication such as OAuth would secure userID based endpoints
* External quotes API: no authentication is required, as the aPI provides public read only content

Hosting and Future expansion

* Mock API: currently hosted on a mock server
* Future plan: the mock API contract will serve as a blueprint got migration to a secure scalable production backend

## UML Diagram

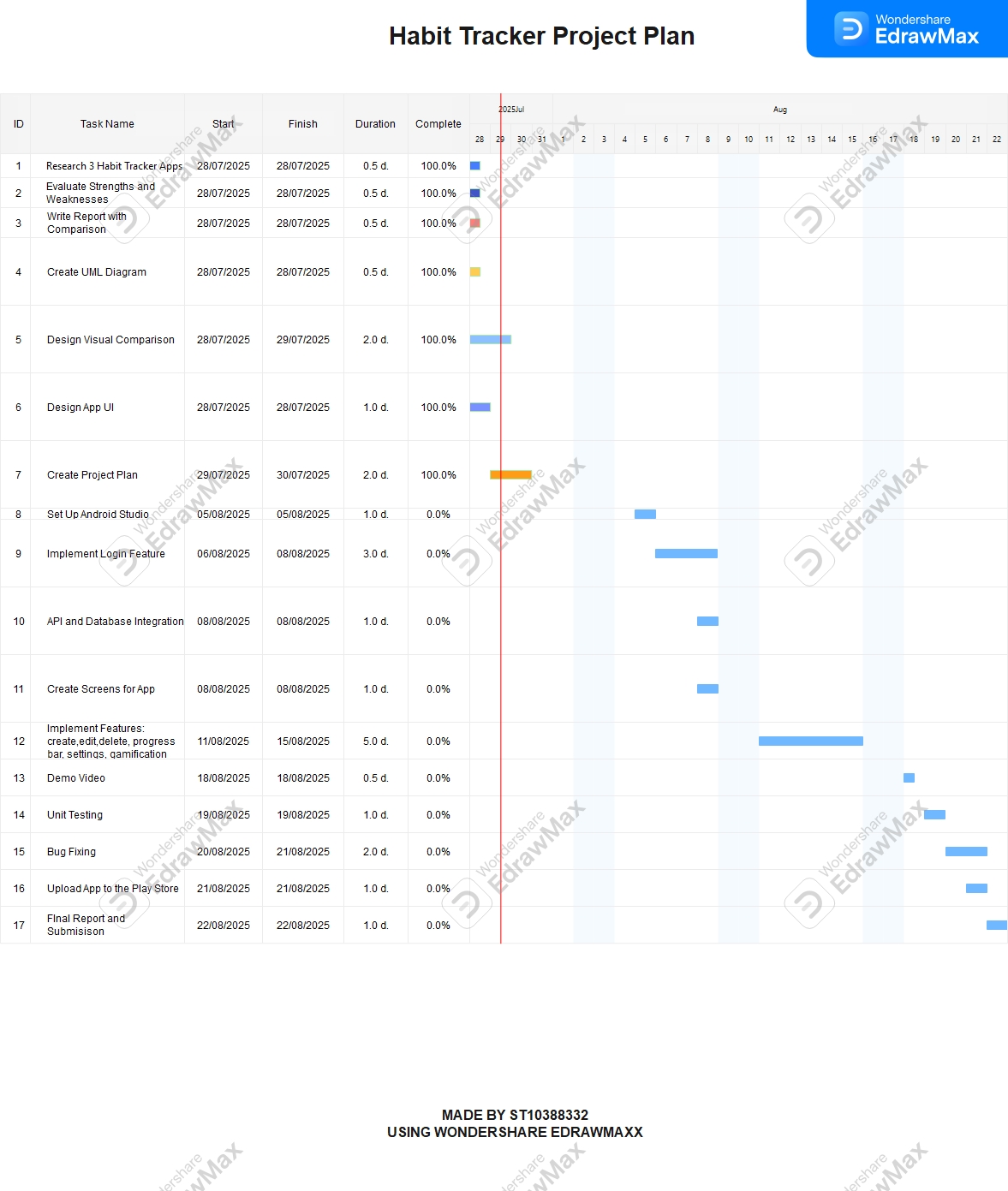


## Data Model and Types

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| habitID | String | Unique ID for each habit |
| userID | String | Firebase user ID |
| name | String | Habit name e.g. “gym” |
| icon | String | Icon assigned to habit |
| color | String | Color hex code for habit |
| frequency | List <String> | Days of week |
| reminderTime | String | Time string e.g. “14:22” |
| createdDate | DateTime | Timestamp of creation |
| completedDate | List<Date> | Dates the habits were completed |

## Gantt Chart

(I will upload this separately in case it isn't clear)



## Conclusion

This research allowed me to investigate 3 android based habit tracking apps – Grow, loop Habit tracker and HabitNow – to understand design, functionality and technical implementation. Each app has its own strengths and unique features

Through this comparison, I identified core features that enhance user engagement and usability, such as visual progress indicators, notifications, habit streaks and the ability to group and schedule tasks/ while each app addresses habit formation differently, they all aim to foster consistency and positive behavior

This analysis has laid the foundation for designing my own habit tracker, HabitMate. I plan to incorporate the best aspects from each app – the growth-based visualization from grow, the clean analytics from Loop and the organization features from HabitNow – to create a modern, motivating and easy to use experience. The research process has also deepened my understanding of android development principles and API integration which will guide the technical build in the next phases of the project.

## References

* <https://www.wondershare.com/>
* <https://www.canva.com/>
* HabitNow. (2024). *HabitNow: Daily Routine Planner*. Google Play Store. <https://play.google.com/store/apps/details?id=com.habitnow>
* Loop. (2024). *Loop Habit Tracker*. Google Play Store. <https://play.google.com/store/apps/details?id=org.isoron.uhabits>
* Grow. (2024). *Grow – Habit Tracker*. Google Play Store. <https://play.google.com/store/apps/details?id=com.growapp>
* Android Developers. (2024). *Build your first app*. <https://developer.android.com/training/basics/firstapp>
* Firebase. (2024). *Firebase for Android.* <https://firebase.google.com/docs/android/setup>

# POE Part 2

## Use of AI Tools in the Development of the Habit Tracker App

During the development of the Habit Tracker App, artificial intelligence tools—specifically OpenAI’s ChatGPT and Google Gemini—were used to support various stages of design, development, debugging, and documentation. The integration of AI assistance allowed for a smoother workflow, reduced coding errors, and provided opportunities for learning and applying best practices in Kotlin and Firebase integration.

**1. Code Generation and Structural Guidance**

ChatGPT was used to generate and refine Kotlin code for several parts of the application. This included the creation of the AddHabitActivity.kt file, which handles user input, data validation, and Firebase Realtime Database integration. AI assistance ensured that the activity followed Android development conventions, including the correct use of lifecycle methods (onCreate()), widget initialization via findViewById, and implementation of listeners for buttons and time pickers.

Additionally, ChatGPT guided the structuring of the Habit.kt data model. It helped ensure that the class followed the correct data class structure in Kotlin and matched Firebase’s JSON storage format. This avoided data type mismatches and made the integration with Firebase seamless.

**2. Debugging and Error Resolution**

AI assistance was instrumental in debugging issues that arose during the coding process. For example, ChatGPT helped interpret Kotlin compiler errors such as “unresolved reference” and “source must not be null”, providing targeted solutions like cleaning Gradle builds, verifying package paths, and using “Invalidate Caches and Restart” in Android Studio. It also clarified issues related to imports and helped ensure files such as Habit.kt were placed correctly under the com.example.habittracker.models package.

When Firebase-related issues occurred—such as data not appearing in the Realtime Database—ChatGPT provided step-by-step troubleshooting instructions, including verifying dependencies, checking the google-services.json placement, and validating Firebase database rules.

**3. Design and UI Recommendations**

AI was also used to recommend layout improvements in XML files to ensure proper spacing, alignment, and readability. It provided guidance on using components such as RelativeLayout, EditText, Button, and CheckBox effectively, as well as best practices for responsive design across different screen sizes.

**4. Learning Enhancement**

Throughout development, AI served as a learning companion, explaining how Firebase Realtime Database works, how to structure Kotlin data classes, and how to manage user interface interactions. Instead of merely copying code, the process involved understanding and adapting AI-suggested snippets to fit the app’s requirements.

**5. Ethical Use and Citation**

All AI-generated content was used ethically and responsibly. ChatGPT was cited as an AI-assisted development tool and not as a human collaborator. Final decisions regarding code structure, logic, and design were reviewed and implemented manually by the developer.

## GitHub Links

<https://github.com/RahilSir/HabitMate>