LEBANESE INTERNATIONAL UNIVERSITY

LIFE PLANNER

All-in-one planner for everyday life.

Project submitted in the context of the course CSCI490-Information System Development

In Part of fulfillment of requirements for the degree of
BACHELOR OF SCIENCE IN COMPUTER SCIENCE

Presented by: Hiba Shaito 42230627
Presented to: Dr. Mohammad Jaafar Nehme
Department of Arts and Sciences
Lebanese International University

MAY 29, 2025
LEBANESE INTERNATIONAL UNIVERSITY

Dedication

This report is respectfully dedicated to my family, whose unwavering support, encouragement, and steadfast belief in my potential have been a continual source of inspiration. Their enduring patience and understanding have provided the foundation upon which this work has been built. It is through their guidance, sacrifices, and unwavering presence that I have been able to pursue and complete this endeavor.

Acknowledgment

I would like to express my sincere gratitude to all those who supported and guided me throughout the course of this project. I am especially thankful to my supervisor for their valuable insights, constructive feedback, and continuous encouragement. I also extend my appreciation to my friends and peers for their collaboration and moral support, as well as to my family for their unwavering patience and motivation. Their contributions have been truly instrumental in the successful completion of this work.

Abstract

This project presents the development of **Life Planner**, a comprehensive mobile application designed to help users efficiently manage their daily lives through integrated task planning, note-taking, wellness tracking, and financial management features.

The app offers a clean and intuitive interface that allows users to create and organize tasks, set reminders, track moods and water intake, manage personal notes, and monitor finances—all in one place. Built using the Flutter framework and integrated with Firebase services,

Life Planner ensures real-time data synchronization and offline functionality for a seamless user experience. The system also supports a Chabot feature for natural language interaction, enhancing usability.

While the app effectively meets core productivity needs, areas such as deeper analytics and advanced user personalization are identified as opportunities for future enhancement.

Table of Contents

Chapter 1: Introduction

1. Problem Statement / Motivation

In today's fast-paced world, individuals often struggle to manage various aspects of their personal and professional lives effectively. Many people rely on separate applications for tasks such as scheduling, note-taking, budgeting, and wellness tracking, resulting in fragmentation, inconsistency, and inefficiency. There is a growing need for an all-in-one solution that can centralize essential life management tools into a single, user-friendly platform.
The motivation behind the Life Planner app stems from this need to simplify and unify everyday productivity and wellness activities in one cohesive digital space.

2. Objectives

The primary objective of the Life Planner app is to provide a holistic platform that empowers users to take control of their daily lives through structured planning and organization. Specific goals include:

- Enabling users to create and manage tasks with due dates, reminders, and repeat schedules.
- Allowing users to take and organize personal notes.
- Integrating mood tracking and water intake logging to promote self-awareness and healthy habits.
- Providing a simple financial tracker to record income and expenses.
- Supporting Chabot interaction for any type of daily life questions.
- Ensuring offline support and real-time synchronization using Firebase.

3. Technologies and Programming Languages

The app was developed using the following technologies and languages:

- **Flutter** – for cross-platform UI development.
- **Dart** – the primary programming language used in Flutter.

- **Firebase** – used for backend services including Firestore (database), Authentication, offline persistence and Storage.

4. Tools and Frameworks

The following tools and environments were used during development:

- **Android Studio** – for emulator testing and debugging.
- **Firebase Console** – for managing backend services.
- **Git & GitHub** – for version control.
- **Figma** – for UI/UX design prototyping (if used).

5. Folder Structure and Explanation

The application follows a modular folder structure to ensure maintainability and scalability. Below is a simplified version:

```
lib/
├── pages/          # All UI pages grouped by feature (e.g., auth, home, settings)
│   ├── auth/       # Authentication-related pages (e.g., login, register)
│   ├── home/       # Home screen and related views
│   ├── onboarding/ # First-time user onboarding screens
│   └── settings/   # User settings and preferences
├── services/       # Business logic, Firestore, and backend services
├── utils/          # Reusable utility classes, constants, and helpers
├── widgets/        # Reusable UI components (e.g., custom buttons, tiles)
├── firebase_options.dart  # Firebase config generated by FlutterFire CLI
└── main.dart       # Application entry point
```

Additional directories and files:

```
test/                          # Unit and widget tests
├── widget_test.dart           # Default Flutter widget test

web/                           # Web-specific configuration and assets

Other project files:
├── pubspec.yaml               # Project dependencies and assets
├── firebase.json              # Firebase hosting configuration
├── flutter_native_splash.yaml  # Splash screen customization
├── analysis_options.yaml      # Linter rules
```

Summary:

- **pages/**: Structured by feature for better separation of concerns.
- **services/**: Encapsulates data handling and business logic.
- **utils/**: Centralized place for helpers.
- **widgets/**: Common UI components to reduce duplication.
- **test/**: Ensures app functionality through unit and widget tests.

Chapter 2: Screenshots with Explanation (User Manual)

This section provides a visual walkthrough of the Life Planner application. Each screenshot is accompanied by a caption and a brief explanation to help users understand how to interact with the app and make the most of its features.

The guide follows the typical user journey — starting from login and onboarding, through task management and settings. These visuals aim to enhance usability by illustrating the app's interface and user interactions step-by-step.

1. Startup & Authentication

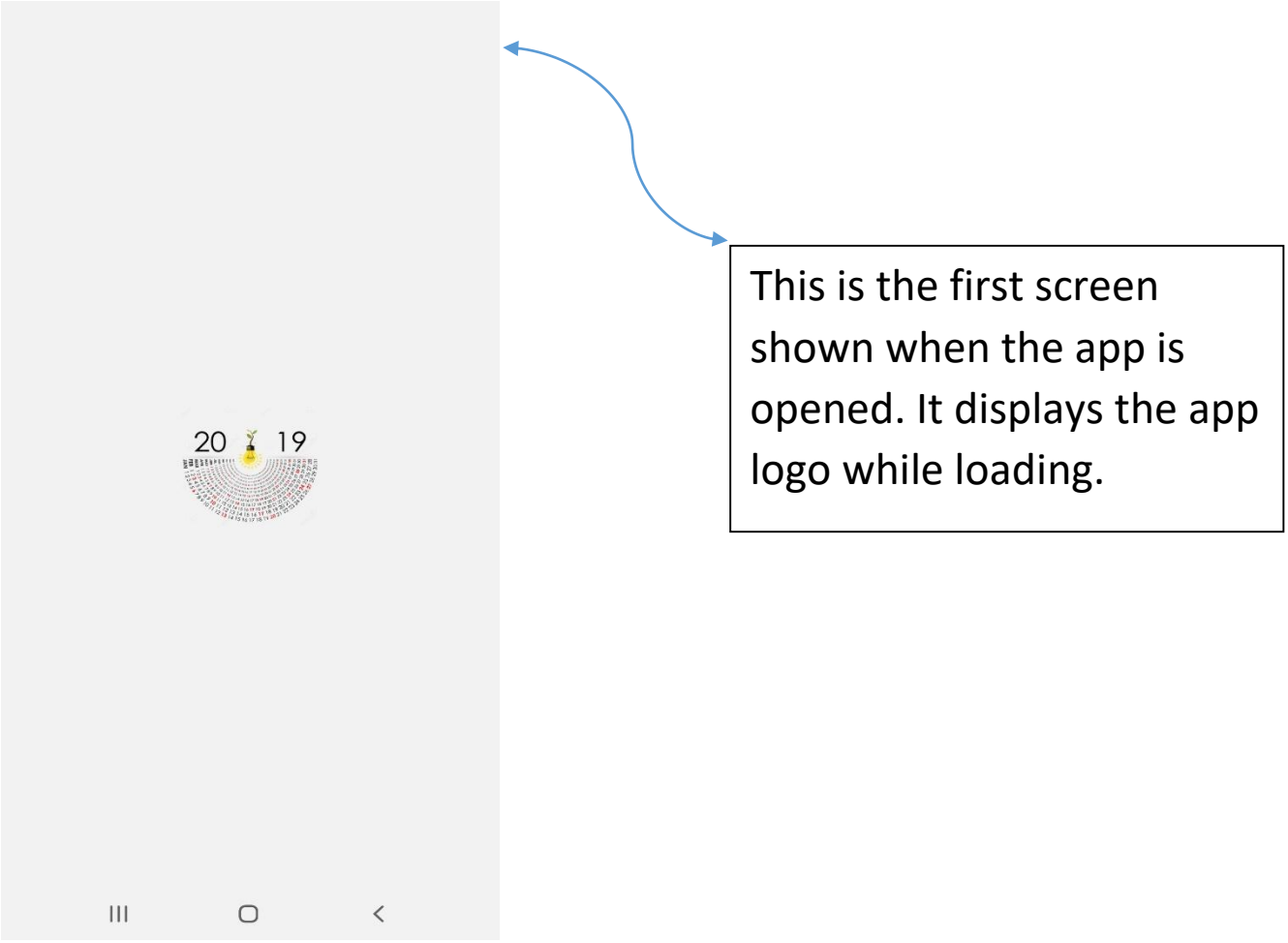


Figure 1-Splash Screen

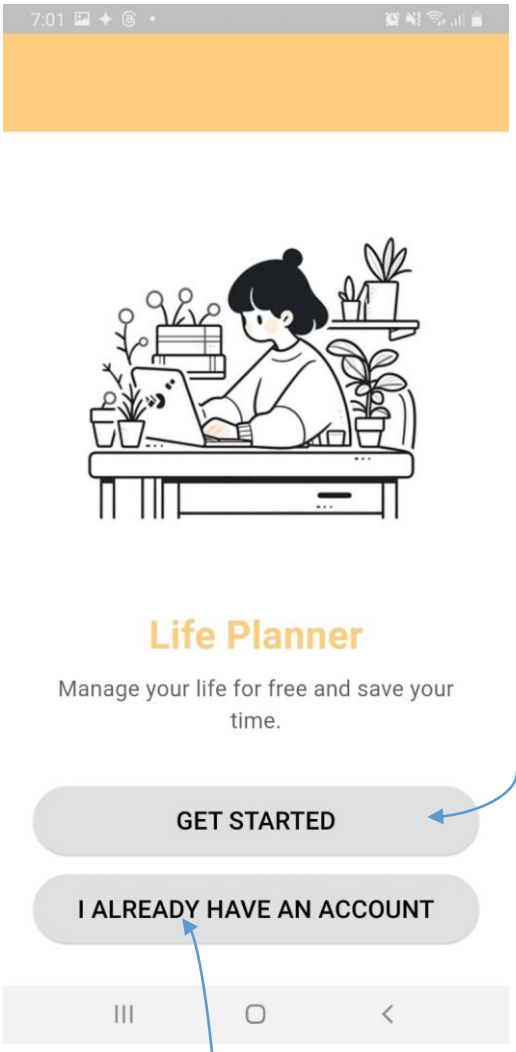


Figure 4-Welcome Page

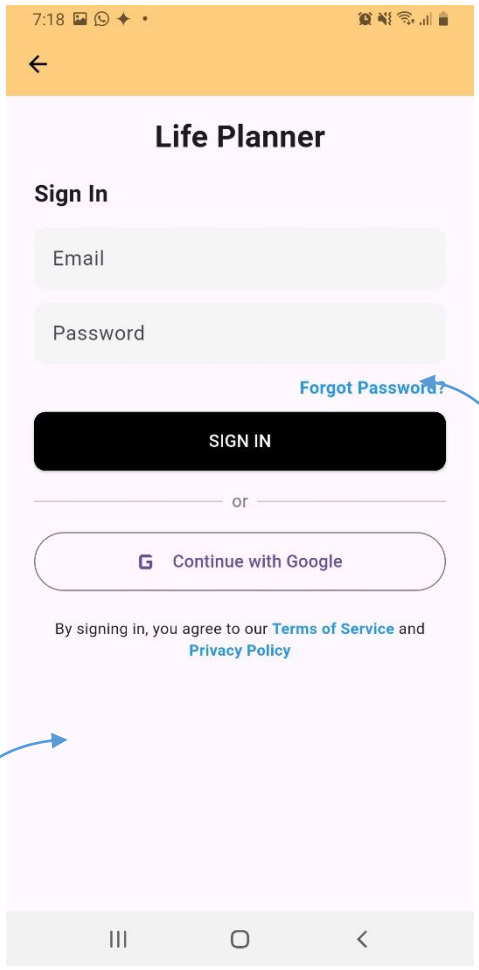


Figure 2-Sign in page

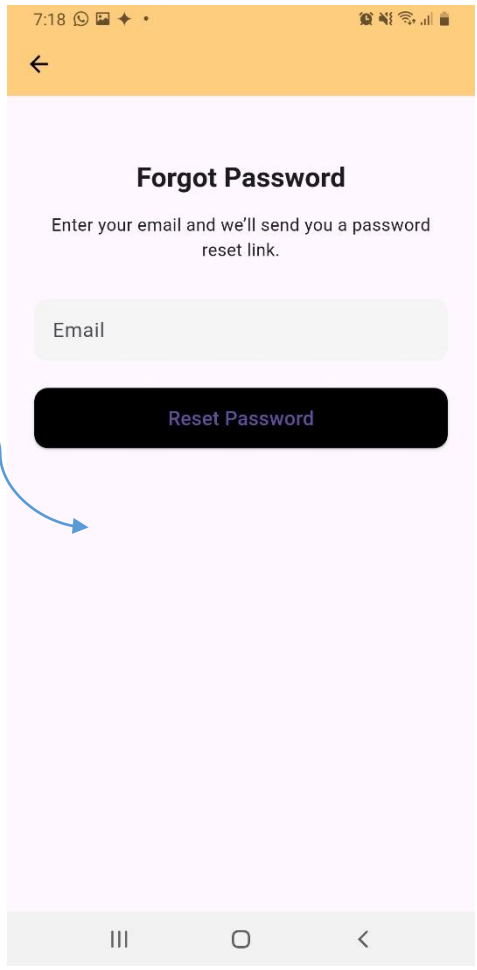
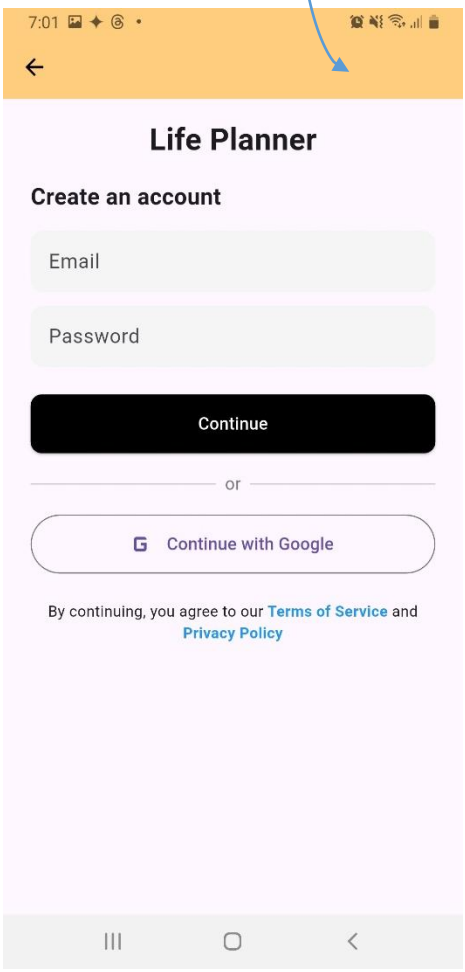


Figure 3-Forget Password Page

Welcome Page:

This is the entry point of the app for new or unauthenticated users. It contains buttons to either sign in or sign up.

- On "Get Started" button tap: Navigates to the Sign in page after a small tutorial.
- On "I already have an account" button tap: Navigates to the Sign Up page.



Forgot Password Page:
Allows users to reset their password by entering their email address.

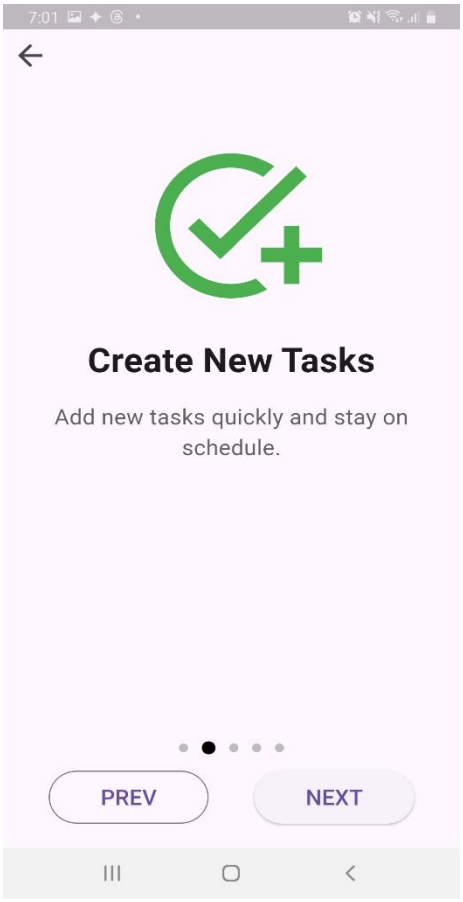
Sign In Page:
Allows existing users to log into their account using their email and password.

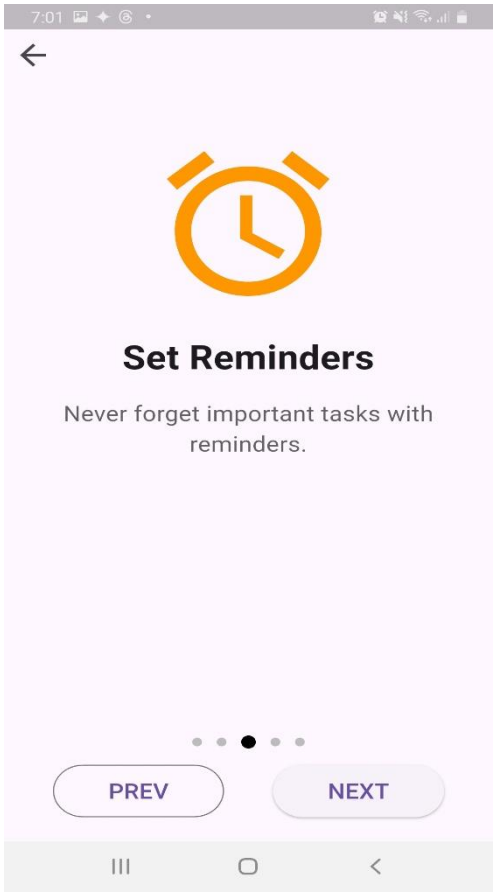
On successful login: Navigates to the Home page.

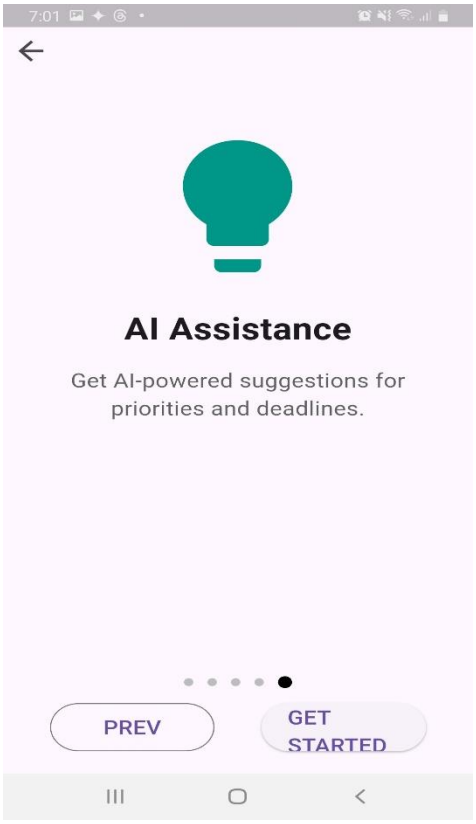
On "Forgot Password?" link tap: Navigates to the Forgot Password page.

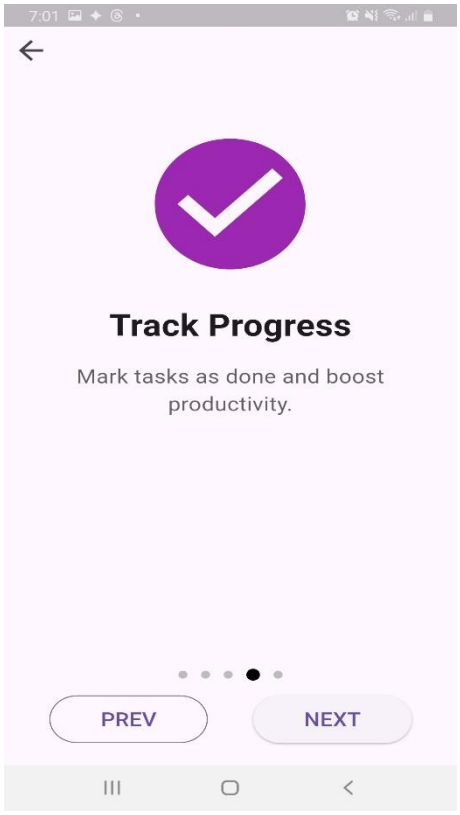
Sign Up Page:
Enables new users to register an account.

On successful sign-up: Navigates to the Sign in page.



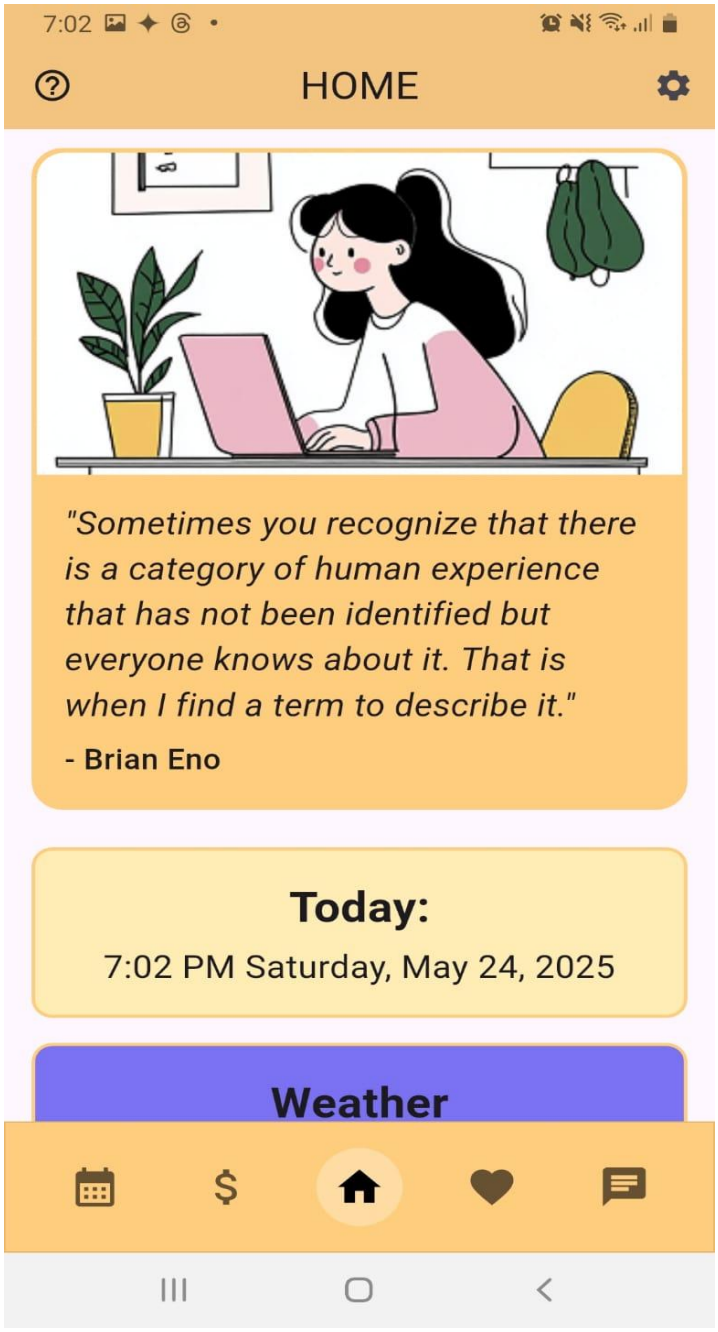


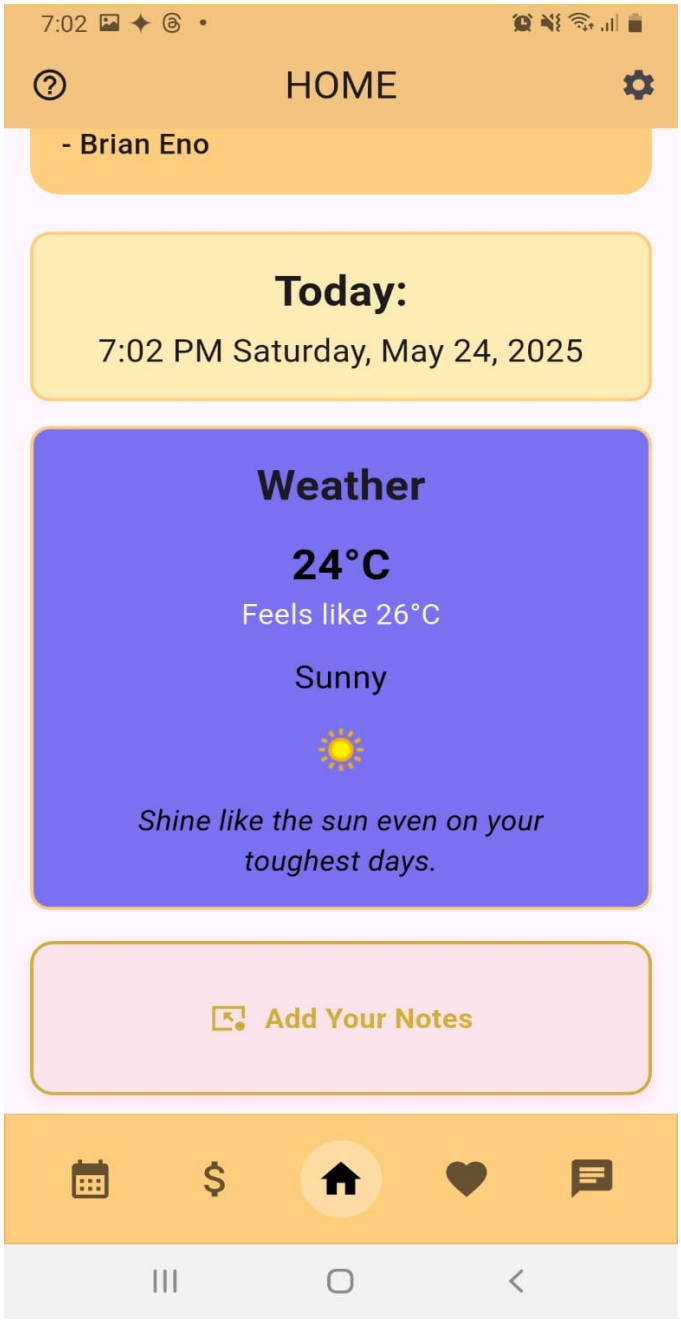




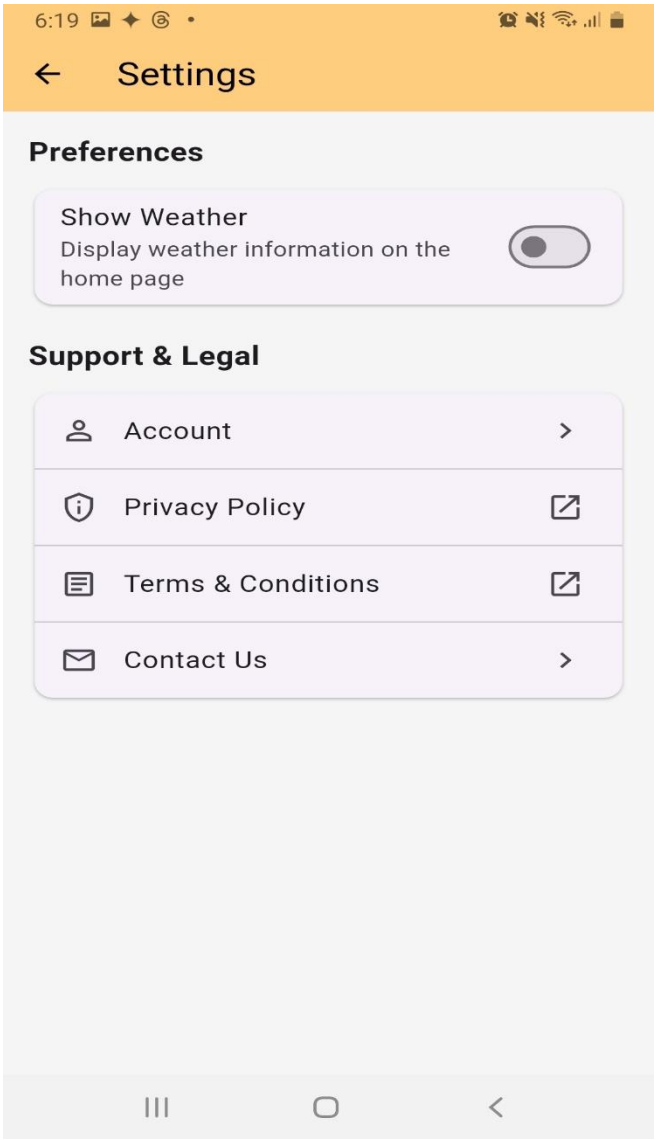
The **Tutorial Page** provides a brief walkthrough of the app's key features to guide new users before they start using the main interface.

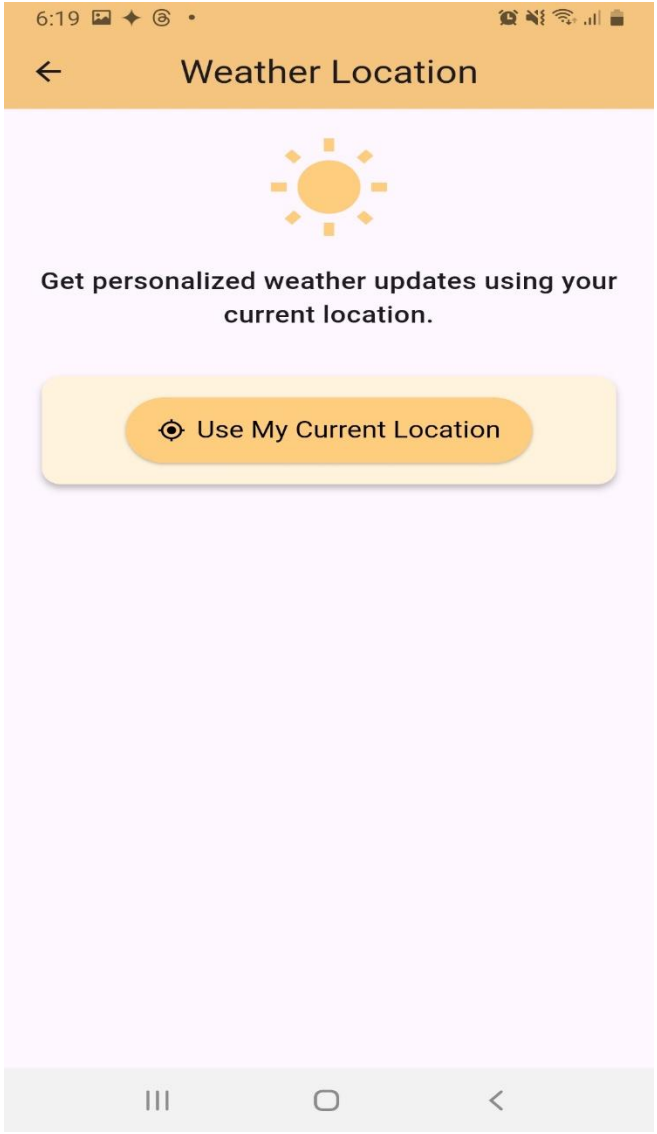
2. Main App features



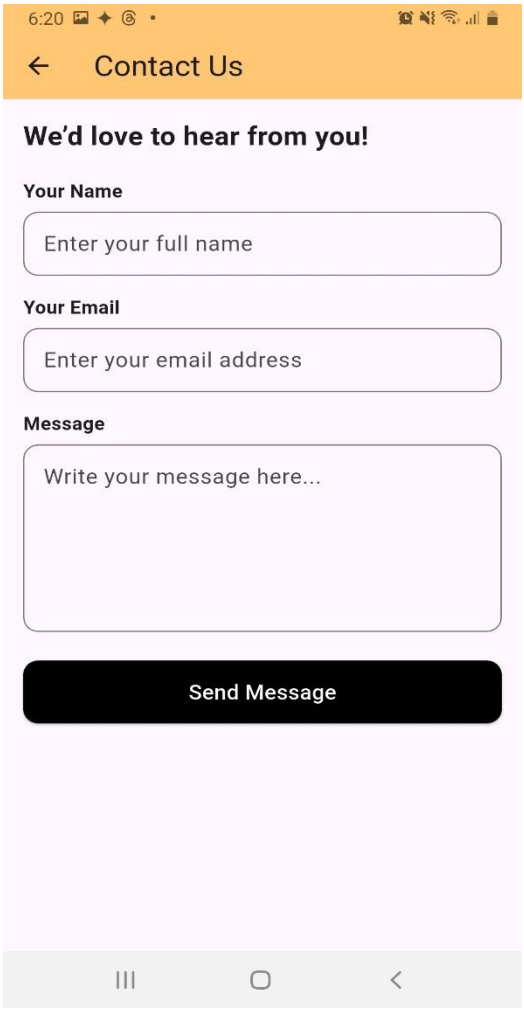


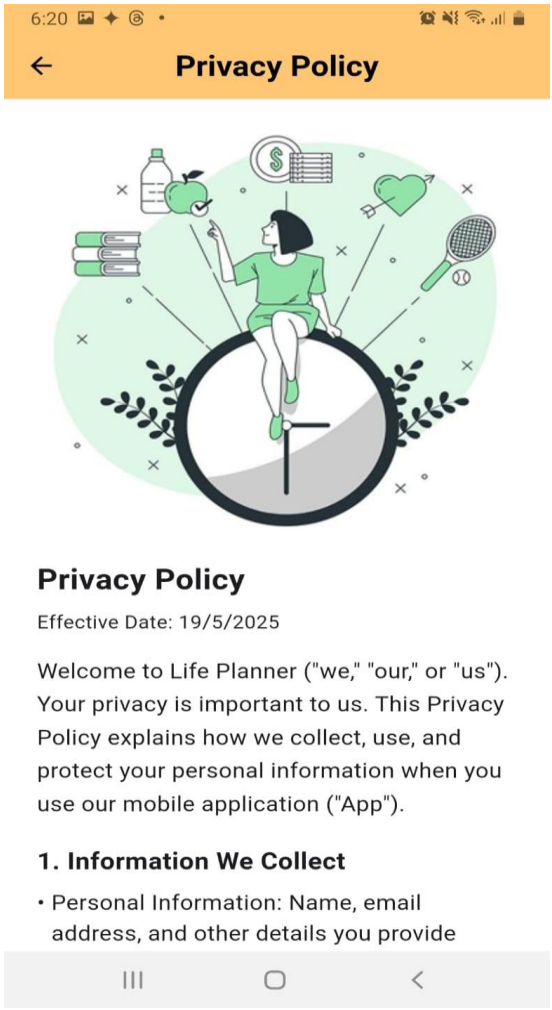
The **Home Page** serves as the main hub of the app, allowing users to navigate to Settings, Schedule, Finance, Health, Chat bot, and Notes pages, as well as view the current weather and a special daily quote for inspiration.

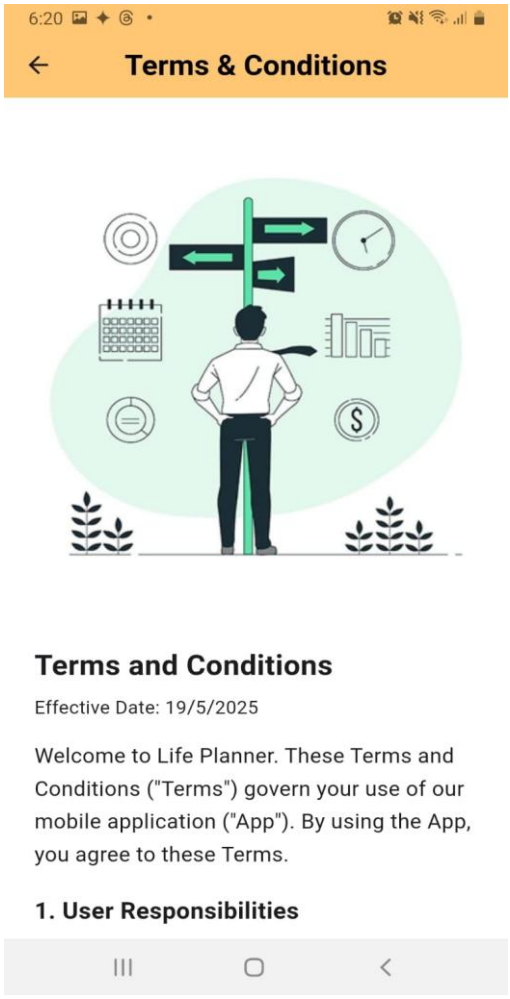




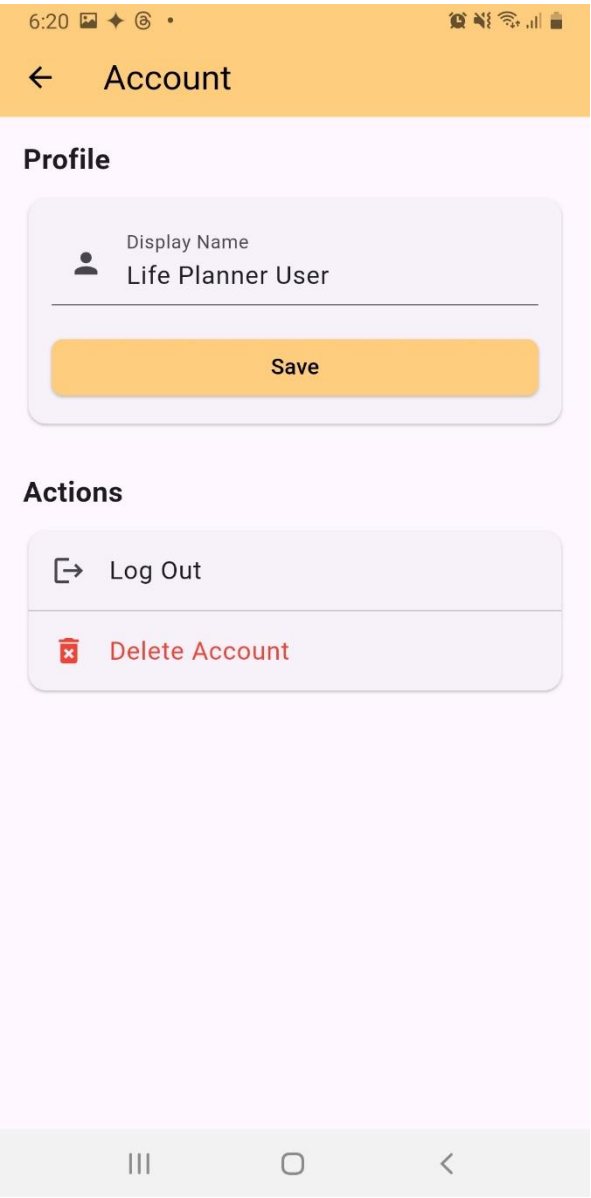
The **Settings Page** allows users to access the Weather Page to check or update their location, the Account Page to change their display name, log out, or delete their account, and view the Privacy Policy, Terms and Conditions, and Contact Us information.



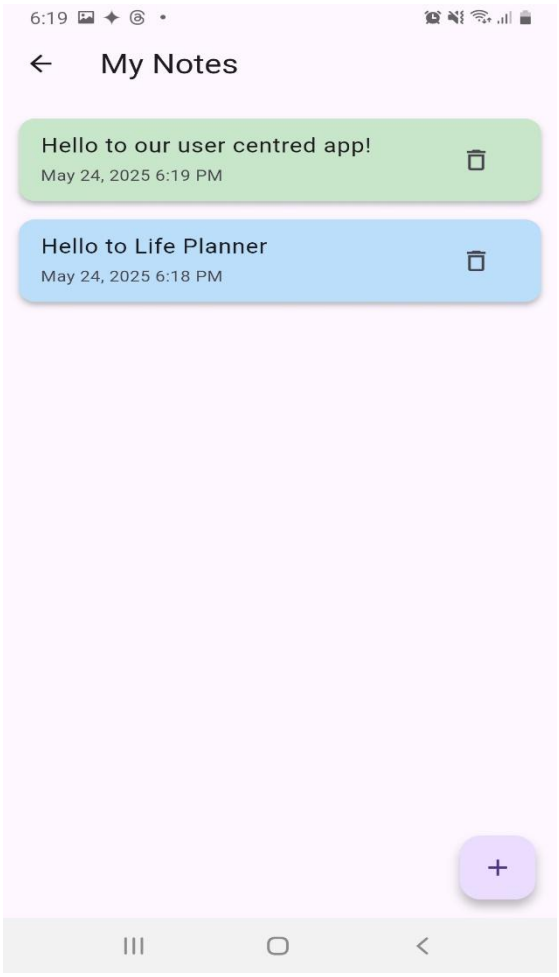


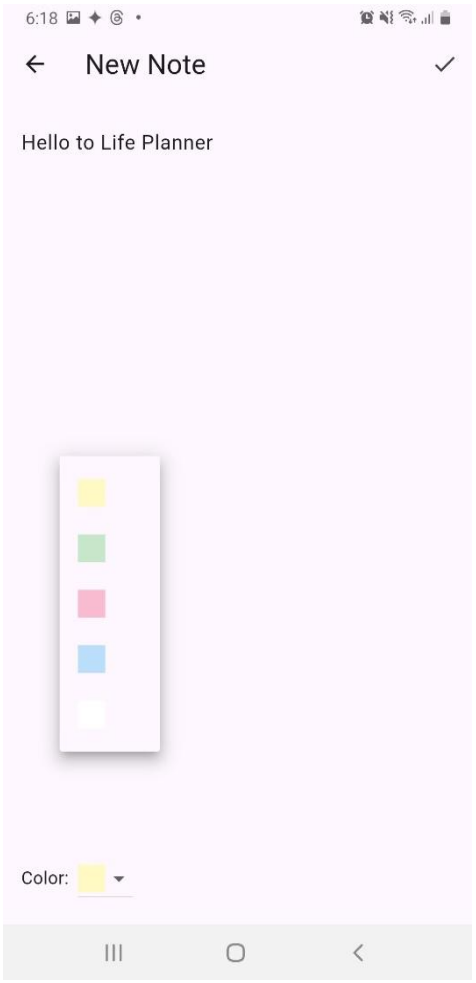


The **Contact Us Page** can be accessed from the Settings Page, as well as from the Privacy Policy and Terms & Conditions pages. Users can also navigate to the Privacy Policy and Terms & Conditions from the Settings Page and from the Sign in and Sign Up pages via a "Read Terms and Conditions" link.

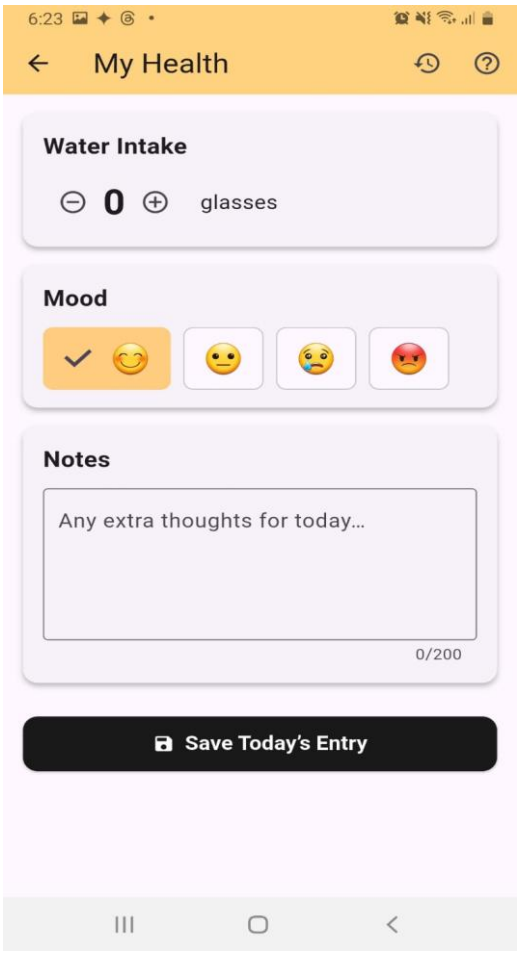


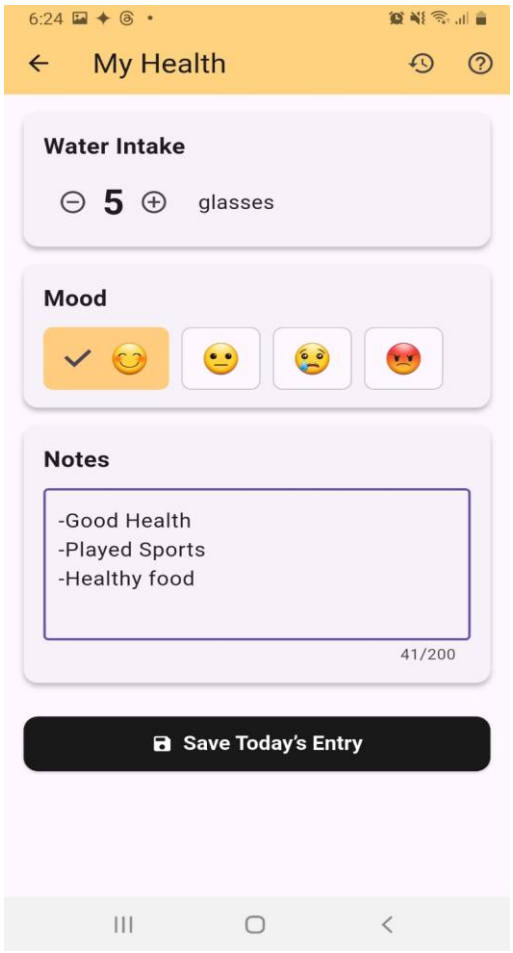
The **Account Page** allows users to change their display name, log out of their account, or permanently delete their account.

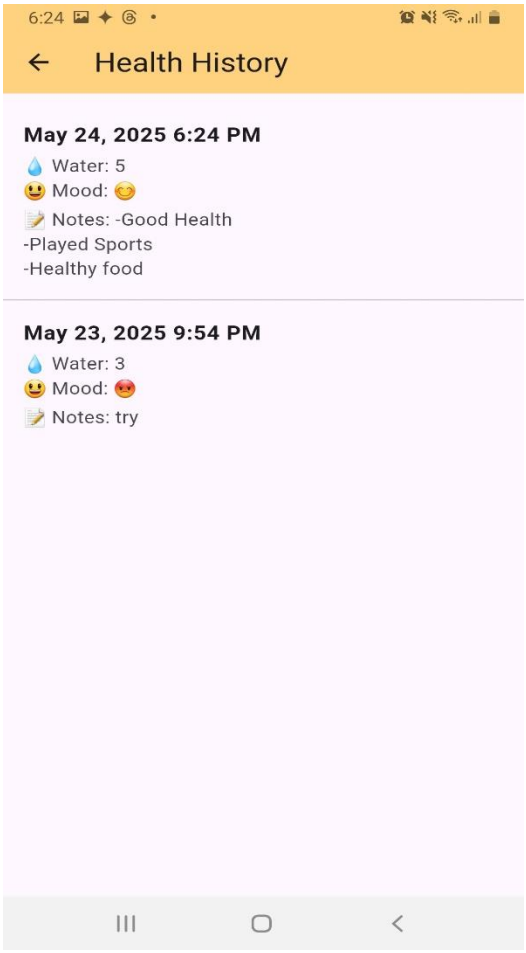




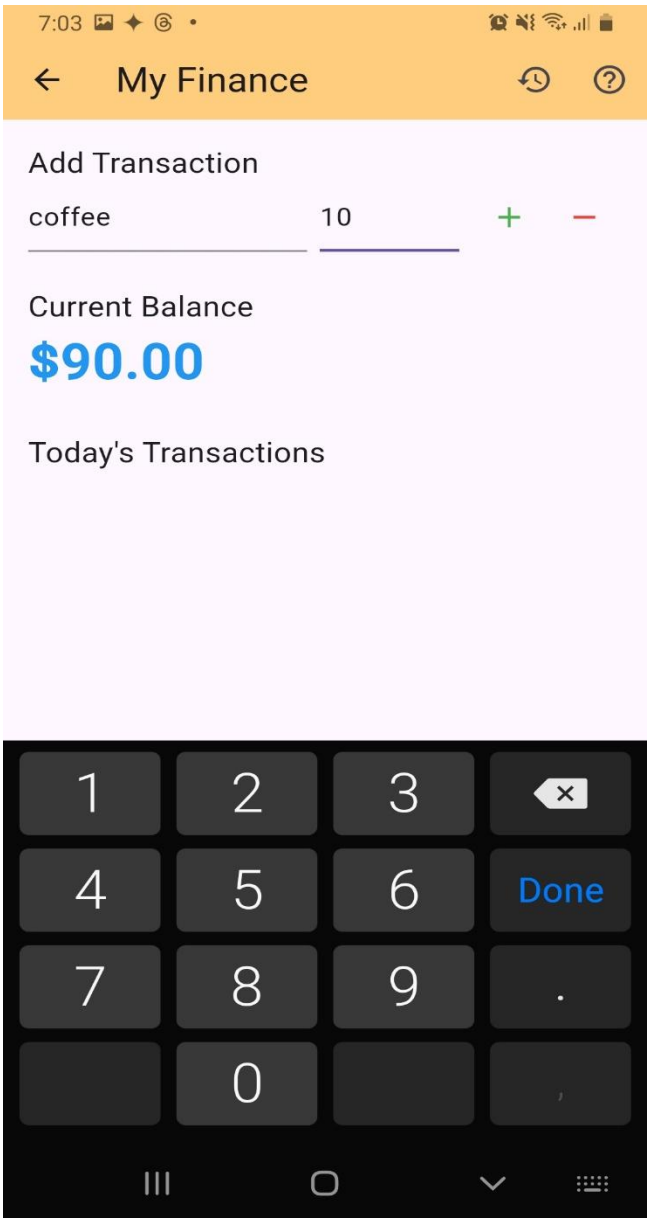
The **Notes Page** functions like a typical notes app, allowing users to add new notes, view, edit, delete them, and change their background color for better organization.

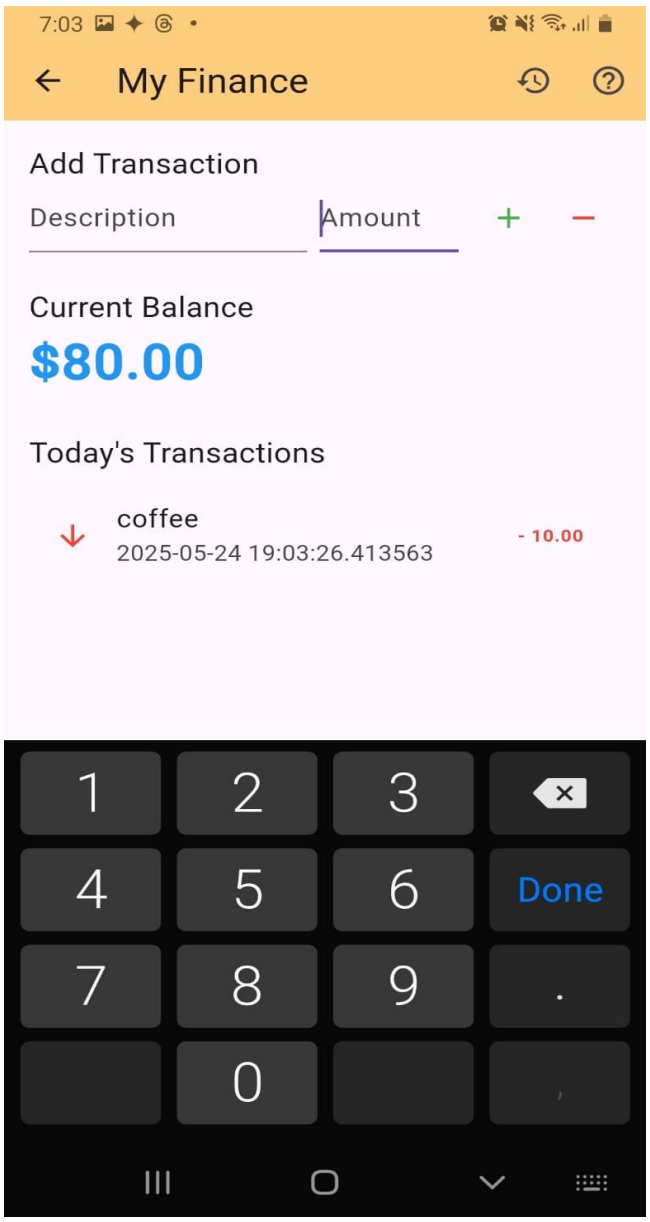




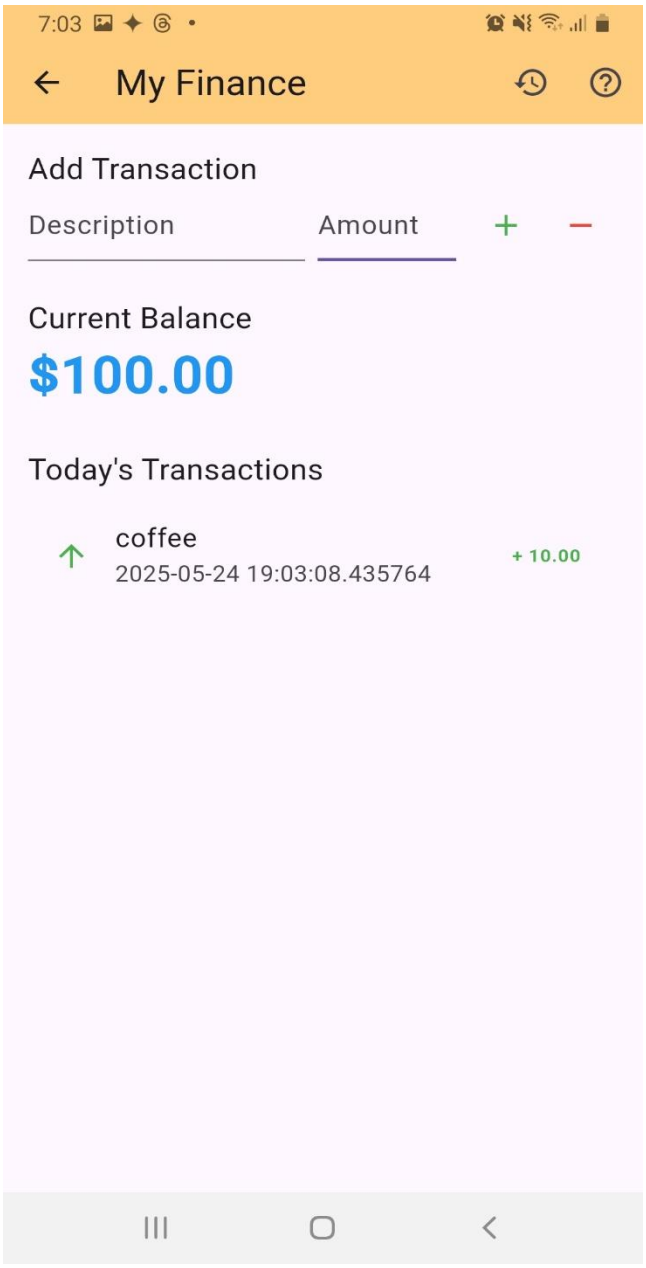


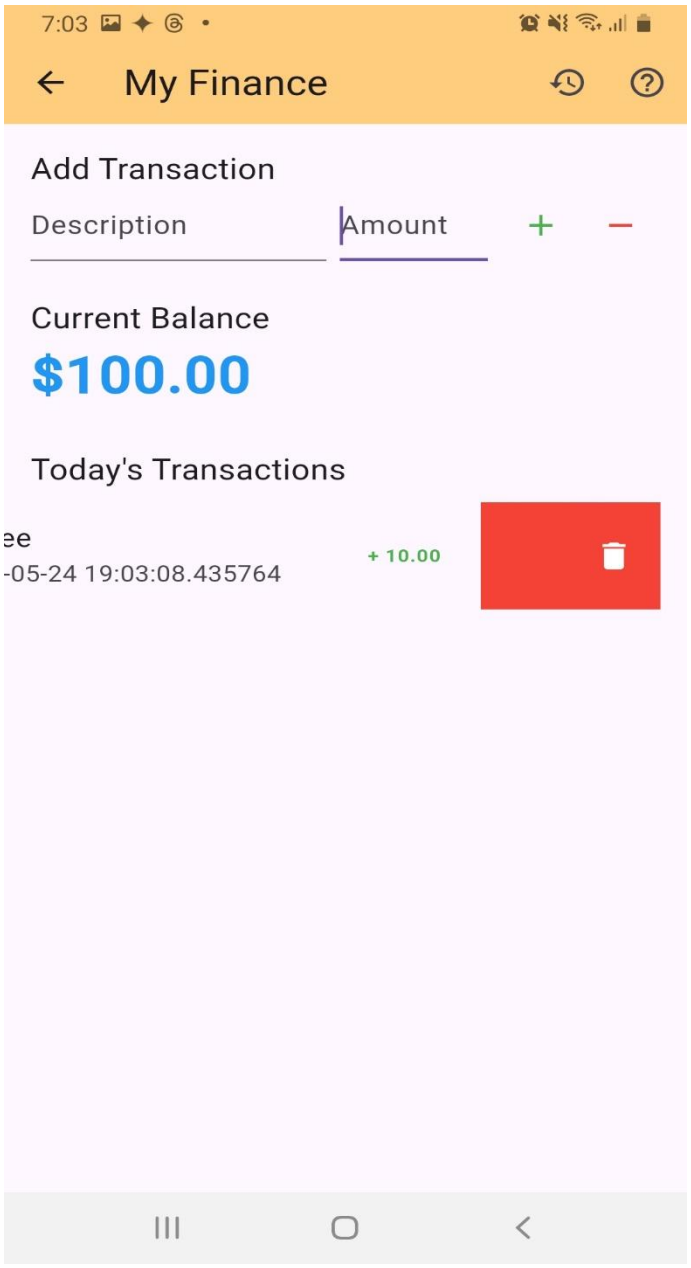
The **Health Page** allows users to log their daily water intake (in glasses), track their mood, add personal health notes, and view their health history over time.

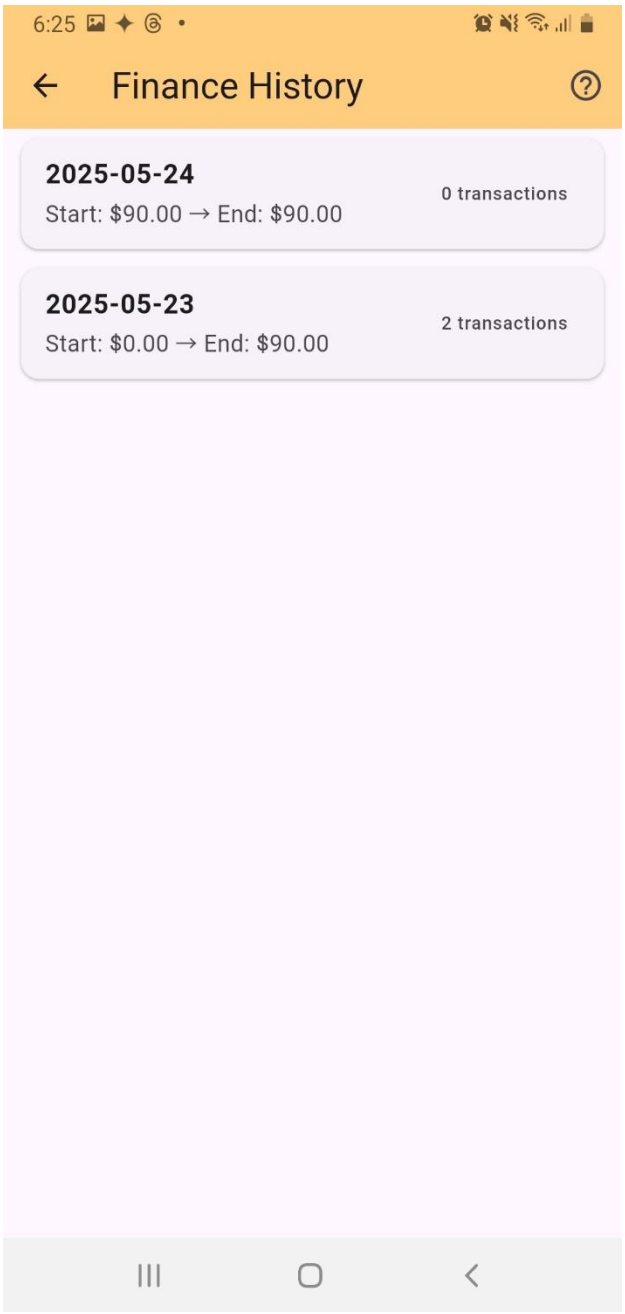


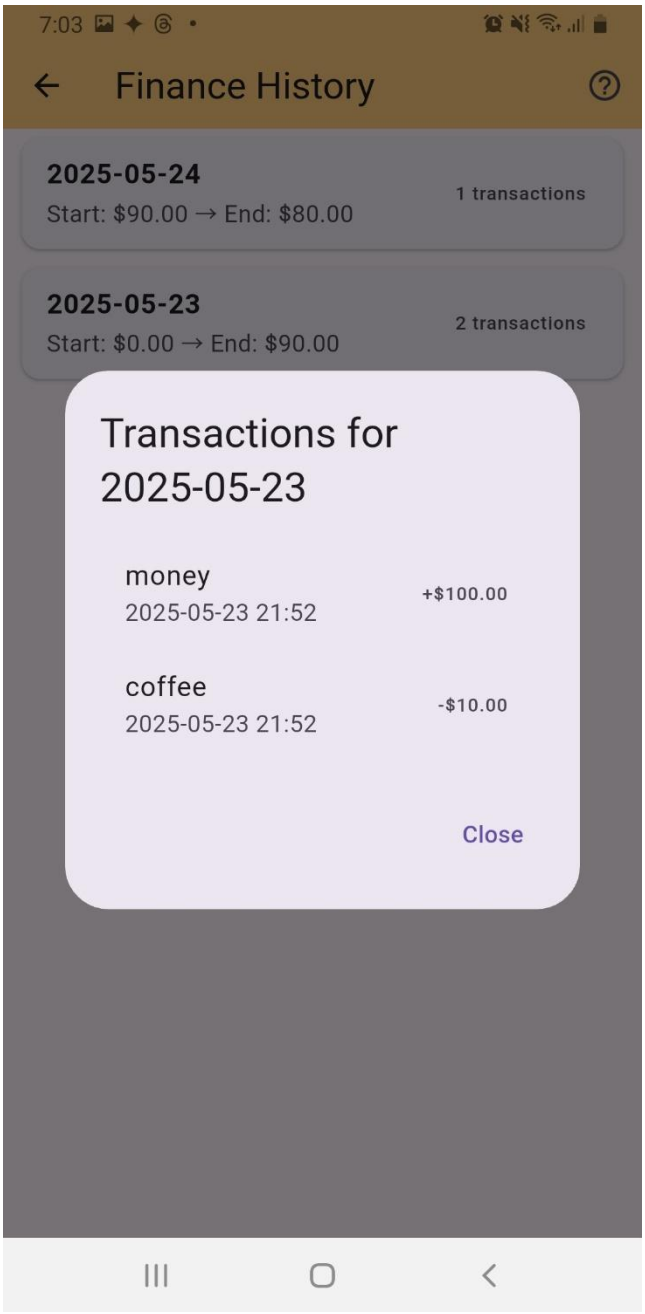


The **Finance Page** allows users to add or remove transactions, whether they are gains or losses, and provides navigation to the Finance History Page for a detailed view of past records.

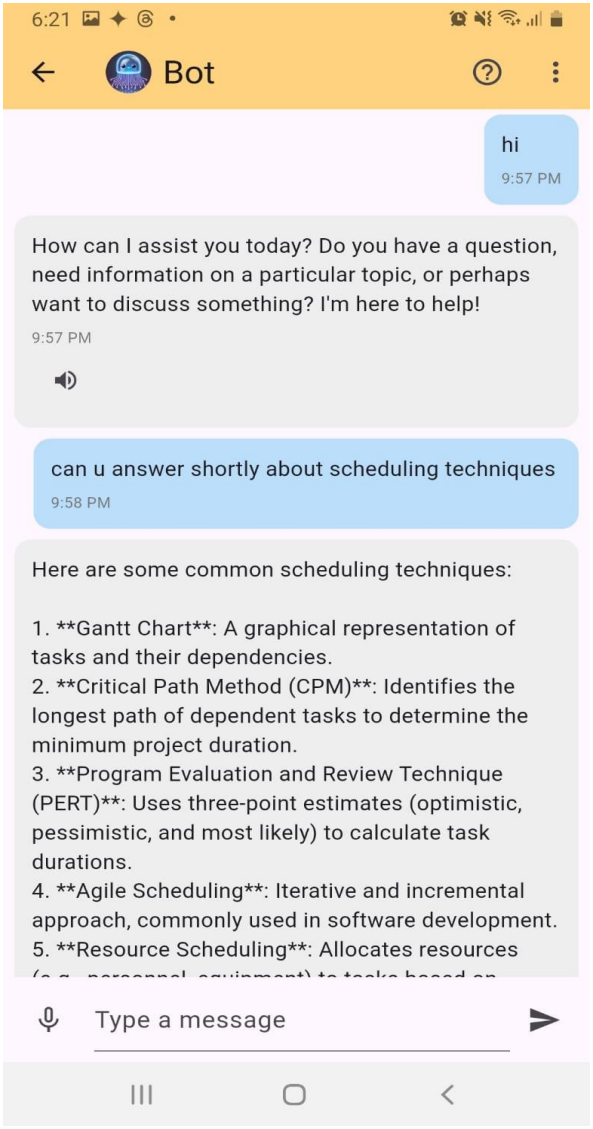

7:03
My Finance
Add Transaction
Description
Amount
Current Balance
$100.00
Today's Transactions
coffee
2025-05-24 19:03:08.435764
+ 10.00

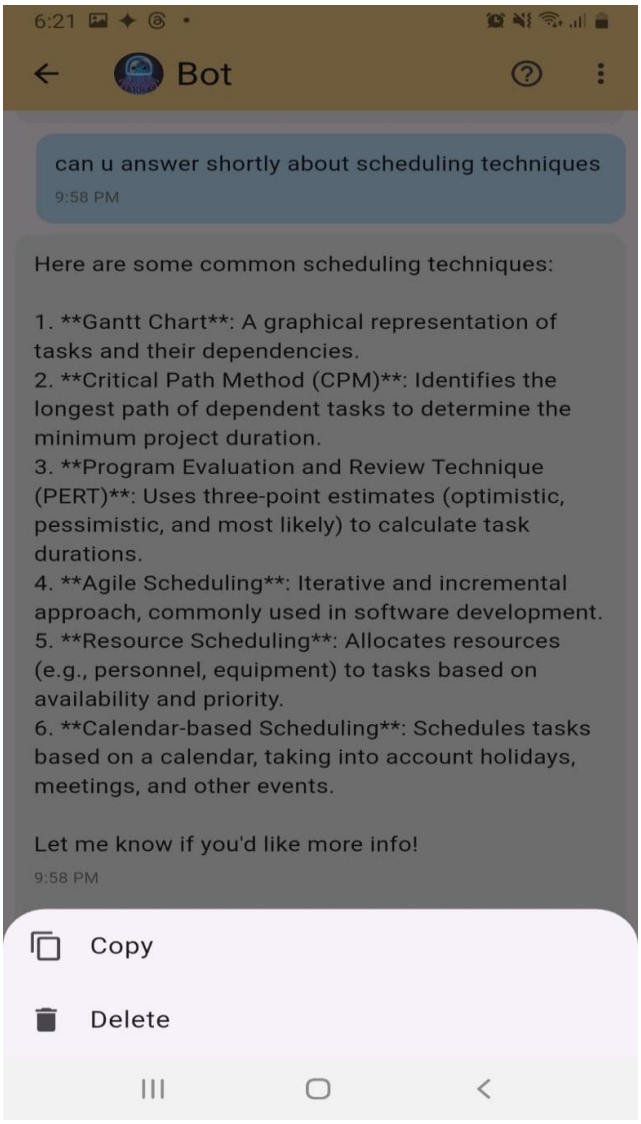

7:03
My Finance
Add Transaction
Description
Amount
Current Balance
$100.00
Today's Transactions
+ 10.00



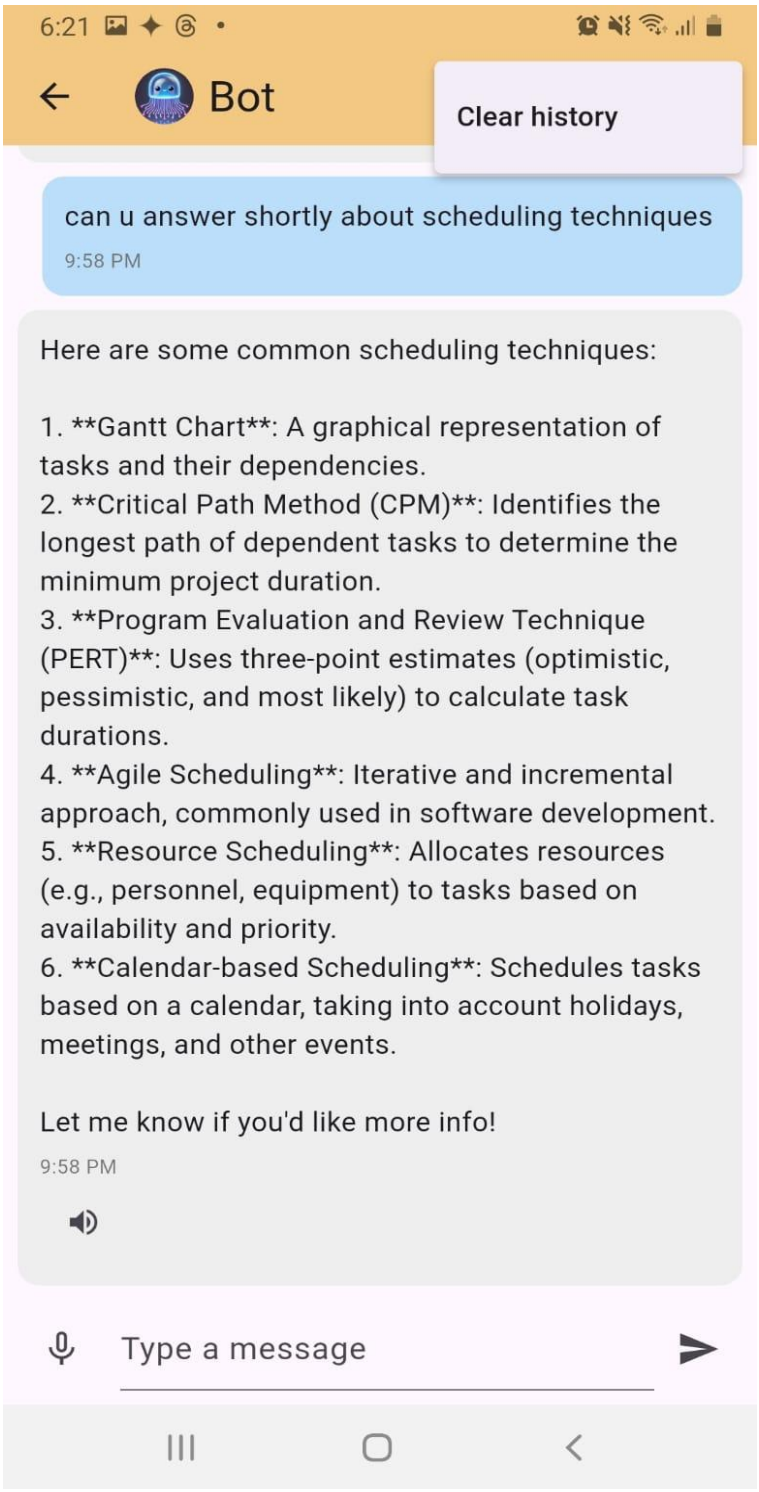


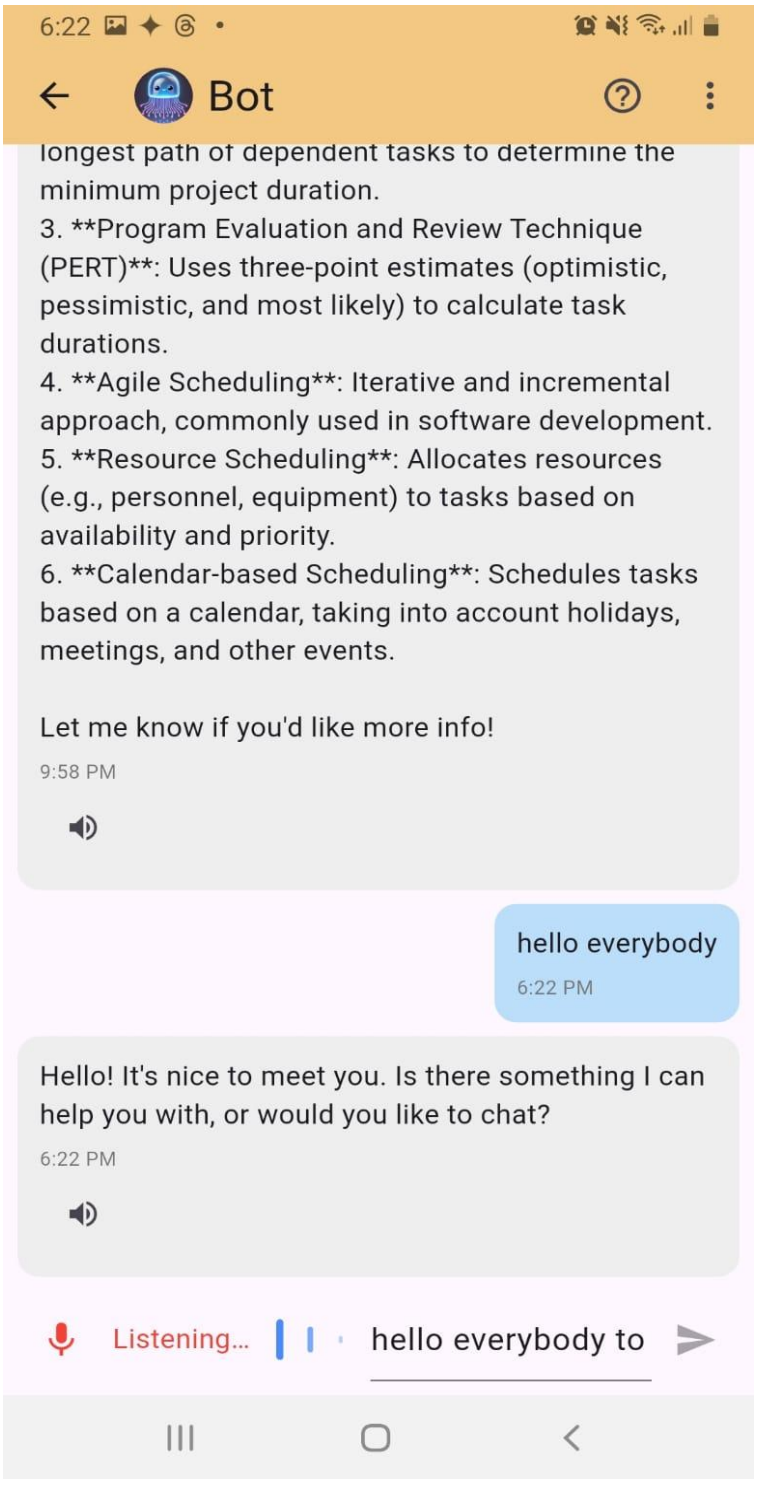
The **Finance History Page** allows users to view past days' income and transaction records for better financial tracking and analysis.

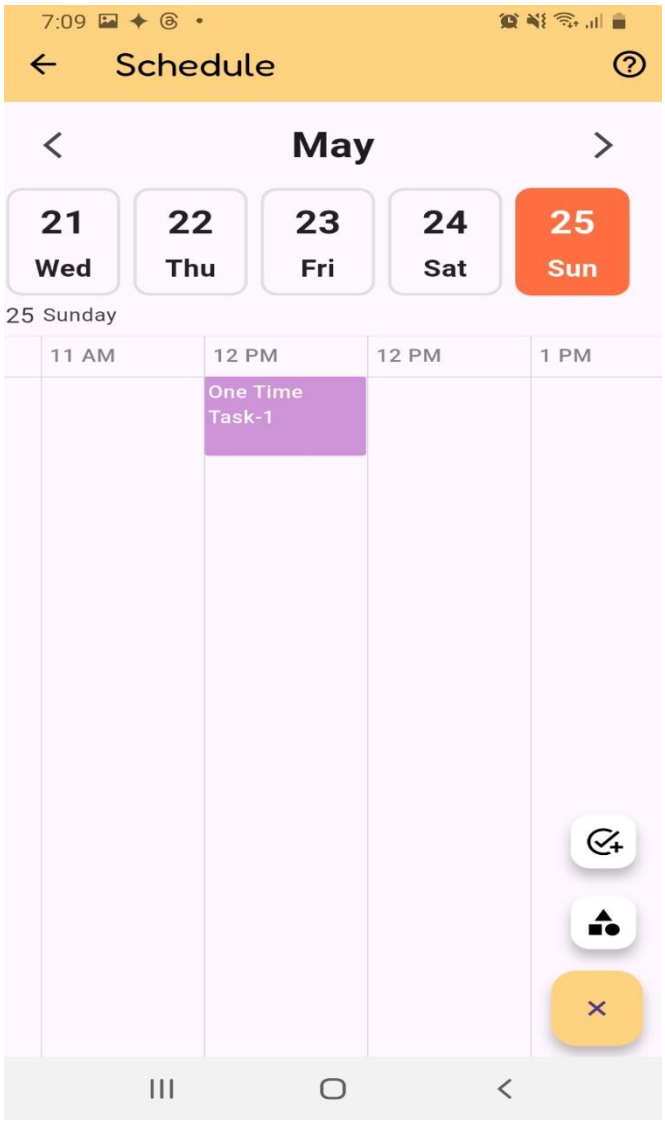


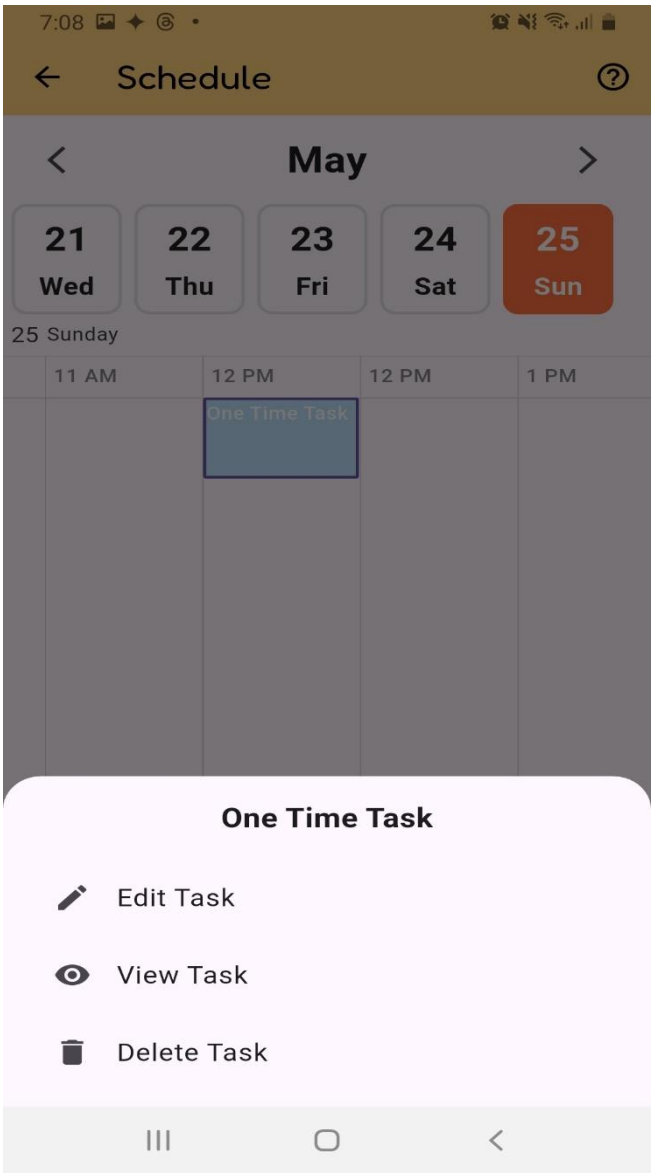


The **Chat Bot Page** lets users chat with the bot about general topics and daily life matters. Users can long-press a message to delete or copy it, listen to messages read aloud, speak directly to the bot, and clear the entire chat history when needed.

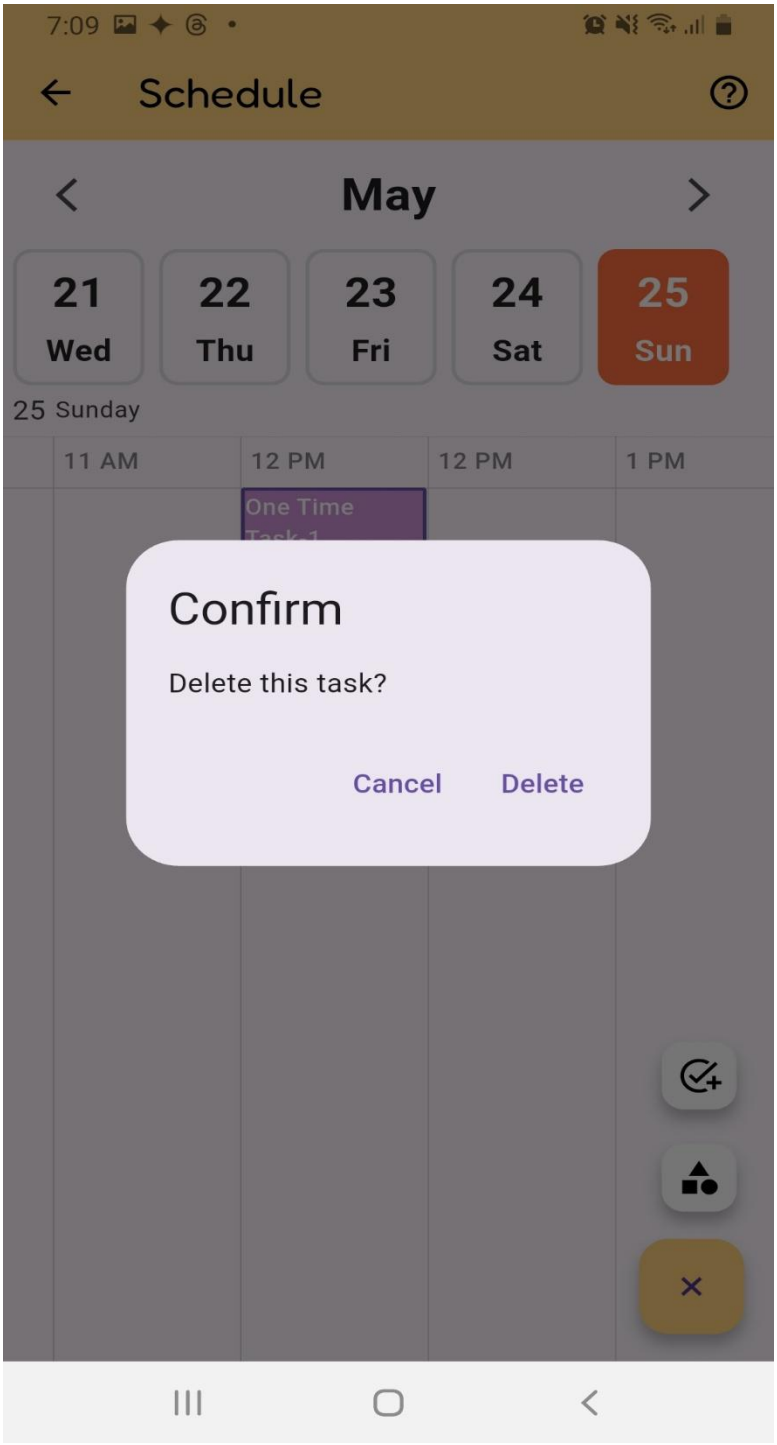

6:21
Bot
Clear history
can u answer shortly about scheduling techniques
9:58 PM
Here are some common scheduling techniques:
1. **Gantt Chart**: A graphical representation of tasks and their dependencies.
2. **Critical Path Method (CPM)**: Identifies the longest path of dependent tasks to determine the minimum project duration.
3. **Program Evaluation and Review Technique (PERT)**: Uses three-point estimates (optimistic, pessimistic, and most likely) to calculate task durations.
4. **Agile Scheduling**: Iterative and incremental approach, commonly used in software development.
5. **Resource Scheduling**: Allocates resources (e.g., personnel, equipment) to tasks based on availability and priority.
6. **Calendar-based Scheduling**: Schedules tasks based on a calendar, taking into account holidays, meetings, and other events.
Let me know if you'd like more info!
9:58 PM
Type a message

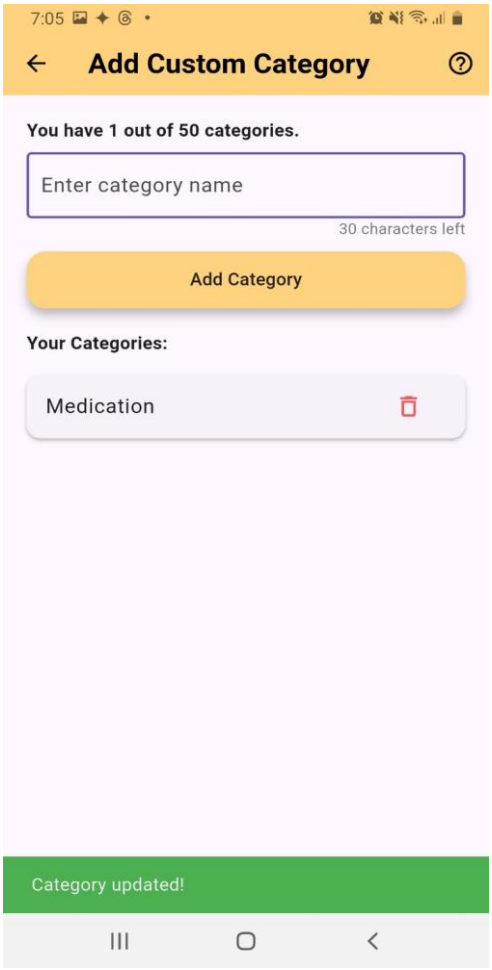

6:22
Bot
longest path of dependent tasks to determine the minimum project duration.
3. **Program Evaluation and Review Technique (PERT)**: Uses three-point estimates (optimistic, pessimistic, and most likely) to calculate task durations.
4. **Agile Scheduling**: Iterative and incremental approach, commonly used in software development.
5. **Resource Scheduling**: Allocates resources (e.g., personnel, equipment) to tasks based on availability and priority.
6. **Calendar-based Scheduling**: Schedules tasks based on a calendar, taking into account holidays, meetings, and other events.
Let me know if you'd like more info!
9:58 PM
hello everybody
6:22 PM
Hello! It's nice to meet you. Is there something I can help you with, or would you like to chat?
6:22 PM
Listening...
hello everybody to

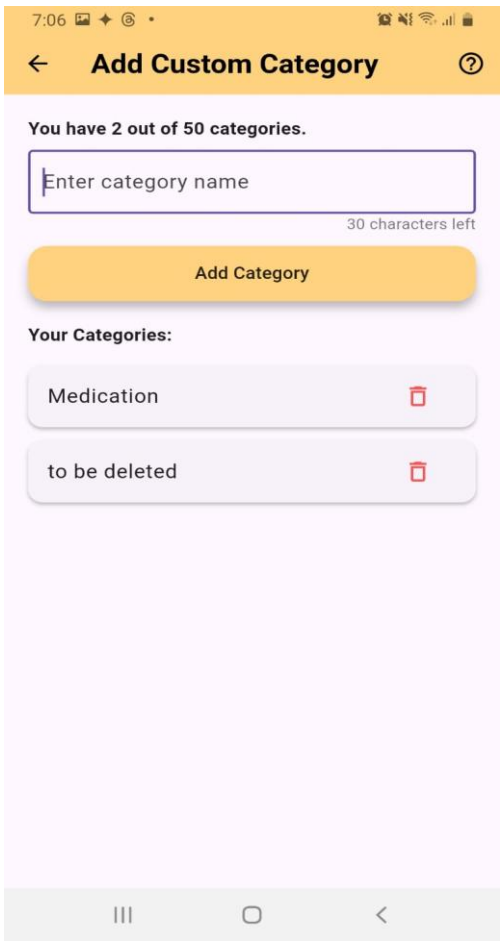


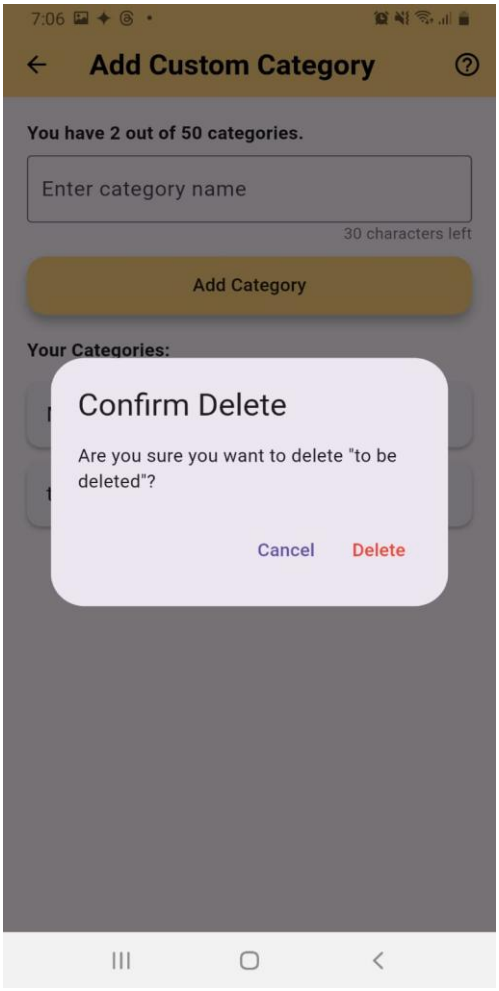


The **Schedule Page** presents user tasks on a calendar timeline, making it easy to visualize daily and upcoming activities. Tapping on a task can make users view full task information, edit the task, or delete it. From this page, users can also navigate to the **Add Task Page** to create new tasks with specific details, such as time, reminder, and recurrence. Additionally, users can go to the **Add Category Page** to create custom task categories for better organization and filtering.

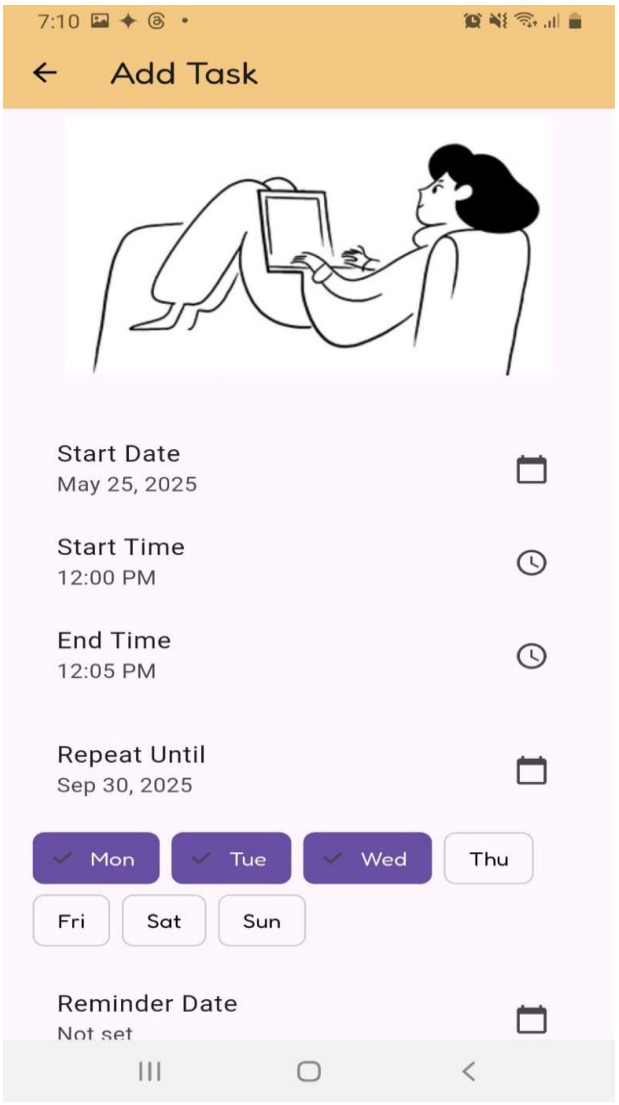

7:09
Schedule
May
21
Wed
22
Thu
23
Fri
24
Sat
25
Sun
25 Sunday
11 AM
12 PM
12 PM
1 PM
One Time
Confirm
Delete this task?
Cancel
Delete



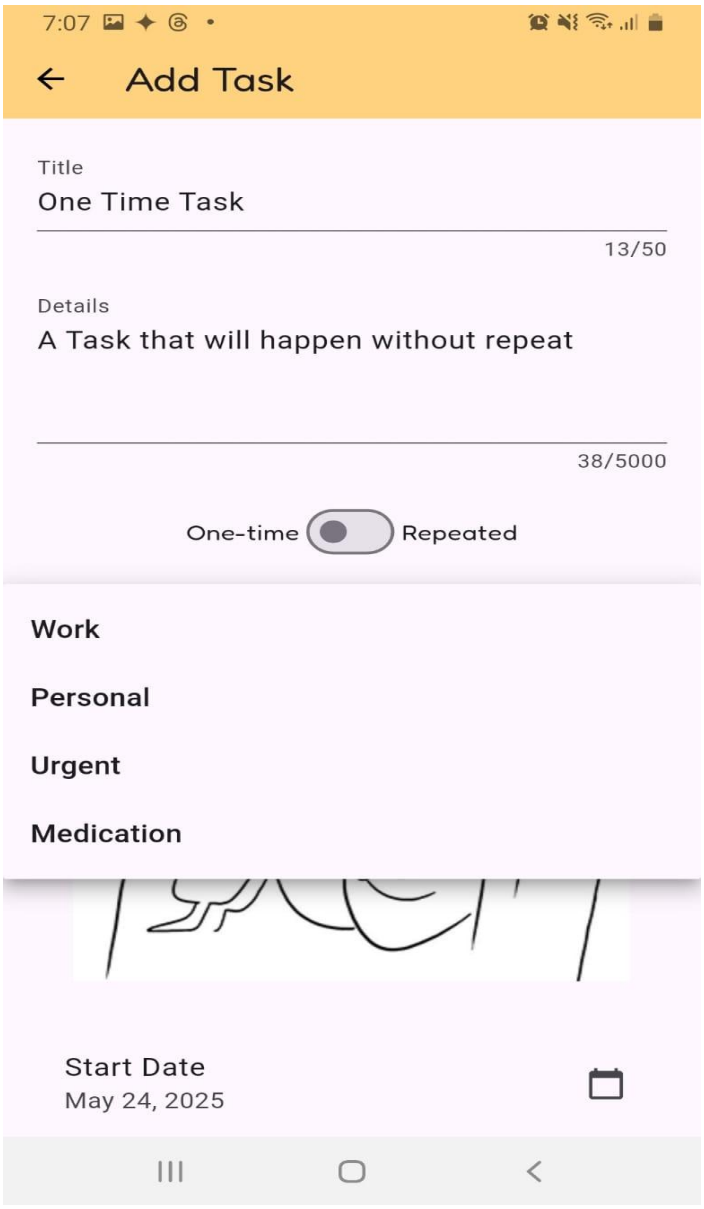


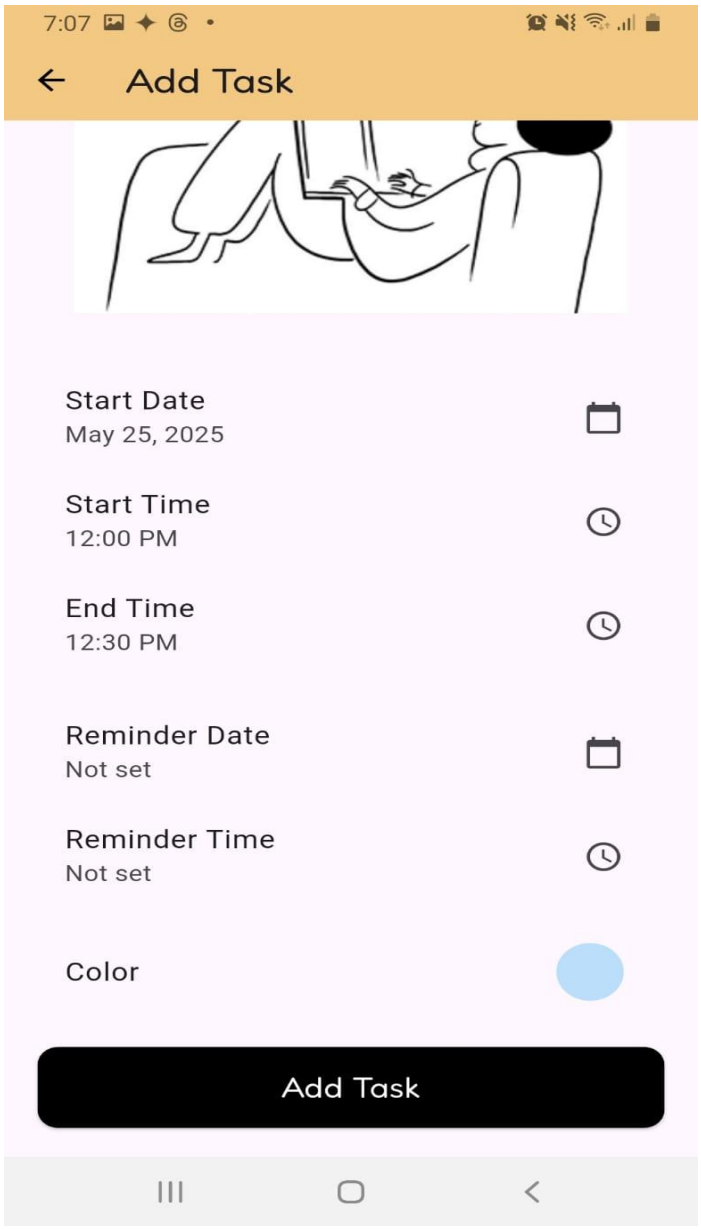


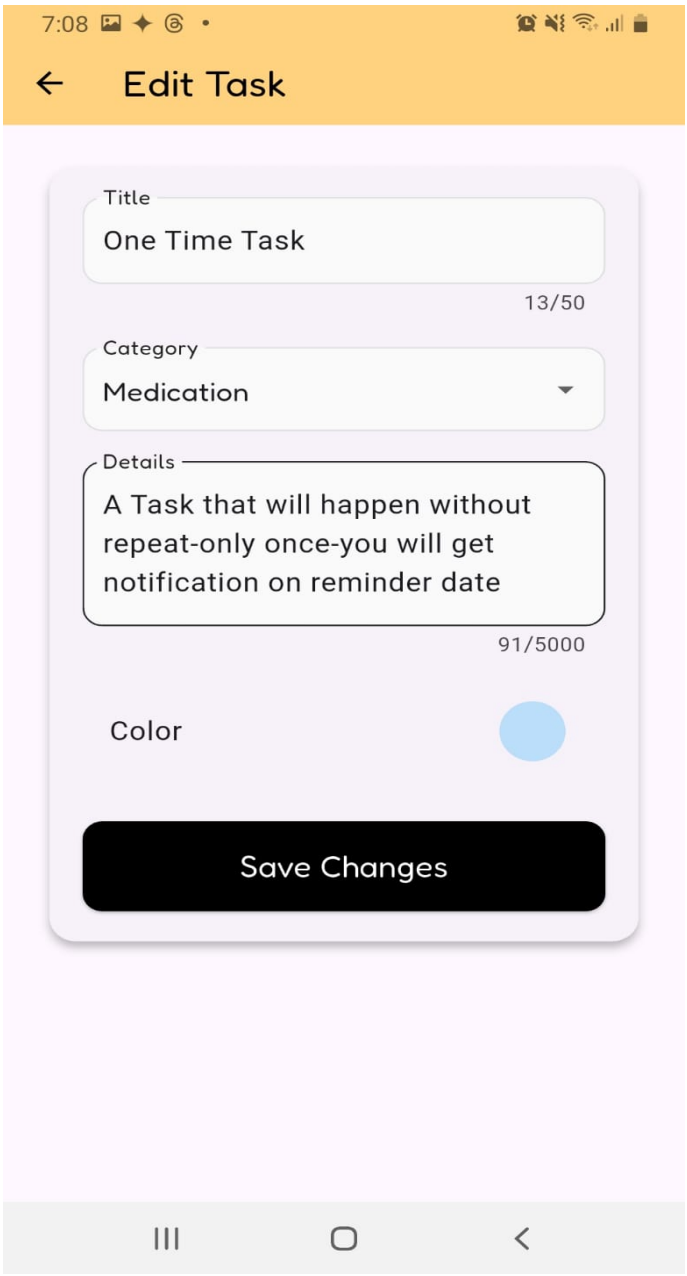
The **Add Category Page** allows users to create, edit, and delete custom task categories for better task organization.



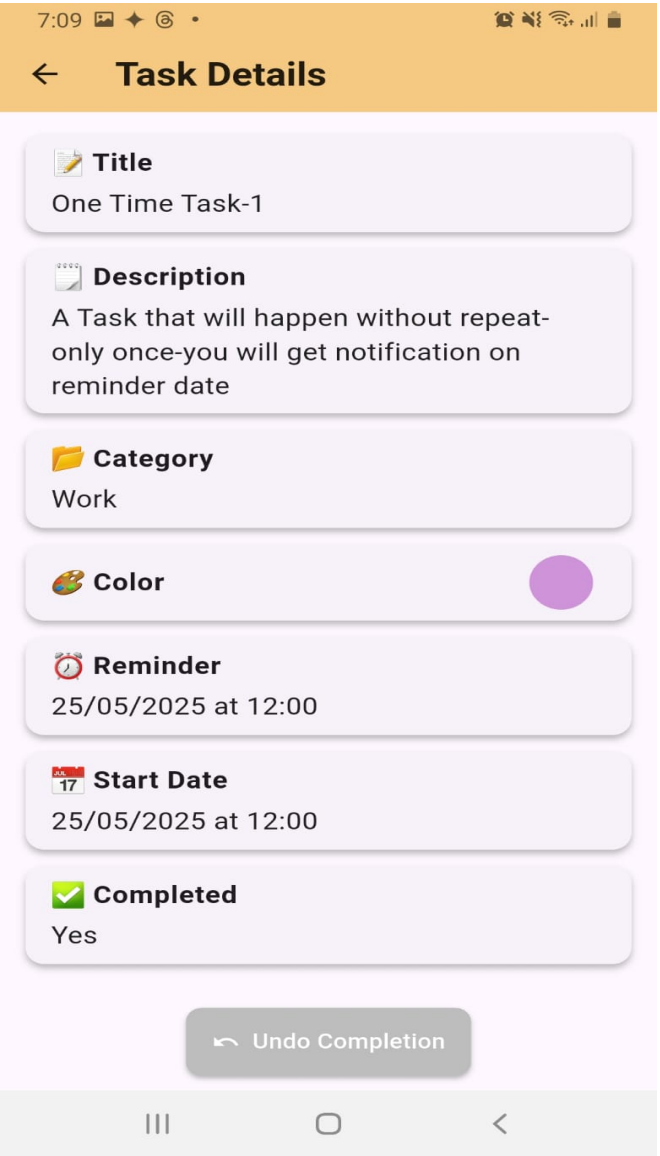
The **Add Task Page** allows users to create new tasks by selecting either a **one-time** or **repeated** task type. Users can assign the task to a category and optionally write a description or set a reminder. If no reminder date is selected, it defaults to the task's start date. This flexible form ensures quick task creation while allowing additional detail when needed.

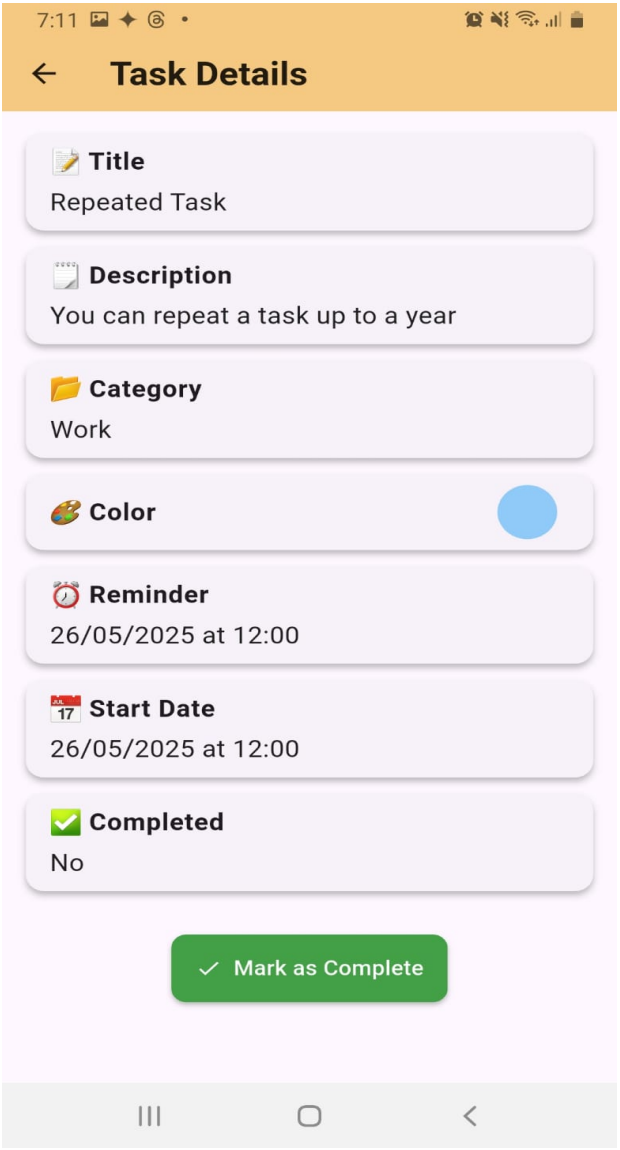

7:07
Add Task
Title
One Time Task
13/50
Details
A Task that will happen without repeat
38/5000
One-time
Repeated
Work
Personal
Urgent
Medication
Start Date
May 24, 2025


7:07
Add Task
Start Date
May 25, 2025
Start Time
12:00 PM
End Time
12:30 PM
Reminder Date
Not set
Reminder Time
Not set
Color
Add Task



The **Edit Task Page** allows users to modify an existing task's details, including the **title**, **description**, **category**, and **color**. This enables users to keep their tasks up to date and organized according to their preferences.





The **View Task Page** displays the complete details of a selected task, including its title, description, category, color, date, and reminder. Users can also **mark the task as completed**.

User Manual Summary

This manual provides an overview of the core features and navigation flow within the app. Starting from the authentication process, users are guided through onboarding and account management. The main interface offers seamless access to essential pages like Home, Notes, Health, Finance, Schedule, and Chat bot, each designed to enhance productivity and personal management. Settings and legal information are easily reachable to ensure transparency and customization. Together, these features create a cohesive and user-friendly experience that supports daily planning, health tracking, finance management, and interactive assistance.

Chapter 3: Project timeline

This chapter outlines the planned schedule and key milestones for the development of the app. It details the phases, tasks, and expected completion dates to ensure the project stays on track and delivers all features efficiently.

Milestone description	Category	Weeks	Progress	Start	Days
Documentation					
Project Introduction	On Track		**100%**	2/17/2025	2
Storyboard Presentation	On Track		**100%**	2/19/2025	26
Business Process Presentation	On Track		**100%**	3/17/2025	5
Use Case and ER Diagram Presentation	On Track		**100%**	3/22/2025	27
Final Documentation	On Track		**100%**	4/18/2025	42
Firebase Authentication Implementation					
Set Up Firebase Project	High Risk		**100%**	2/19/2025	2
Enable Authentication Methods (e.g., email/password, Google Sign-In)	High Risk		**100%**	4/16/2025	6
Implement Authentication UI	Low Risk		**100%**	4/16/2025	3
Test and Secure Authentication Flows	Med Risk		**100%**	4/22/2025	5

Firestore Database Design and Integration				
Set Up Firestore in Firebase Console:	High Risk	**100%**	2/19/2025	2
Develop Data Interaction Functions: Write functions to add, retrieve, update, and delete data within Firestore.	High Risk	**100%**	4/22/2025	22
Test Database Operations	On Track	**100%**	5/14/2025	5
Chatbot Development and Integration				
Develop Chatbot Conversational Flow	Med Risk	**100%**	5/15/2025	2
Integrate Chatbot with Firestore	Low Risk	**100%**	5/17/2025	1
Implement User Interaction Features	Med Risk	**100%**	5/18/2025	3
Test Chatbot Functionality	On Track	**100%**	5/21/2025	1
Application Design and Frontend Development				
Develop Frontend Components	Med Risk	**100%**	4/22/2025	10
Integrate Frontend with Backend	Med Risk	**100%**	5/14/2025	5
Optimize UI for Performance and Responsiveness	Med Risk	**100%**	5/20/2025	10
Additional Features Development				
Weather Forecast Integration	Low Risk	**100%**	5/21/2025	1

Daily Quote Display	Low Risk	**100%**	5/22/2025	1
Test and Refine Additional Features:	Med Risk	**100%**	5/23/2025	3
Apk creation and testing	Milestone	**100%**	5/26/2025	2

Link to the chart:

[Hiba Shaito 42230627-Life Planner\Agile Gantt chart.xlsx]

The attached Gantt chart outlines the full project timeline, covering key milestones such as documentation, Firebase integration, chat bot development, frontend design, and additional features. All tasks are marked 100% complete, with clear progress across high-risk and medium-risk areas, ensuring the project was delivered on schedule.

Chapter 4: Use Cases

This section presents the key use cases of the system, outlining how users interact with the application to achieve specific goals. Each use case describes a functional requirement from the user's perspective, helping to ensure that the system meets real-world needs and supports all intended user activities.

Use case narrative

In software development, a use case narrative serves as a detailed description of how users interact with a system to accomplish specific goals or tasks.

It outlines the sequence of steps involved, the actors or users participating in the process, and the system's response to each action.

Use case narratives are valuable tools for understanding and documenting system requirements, guiding development efforts, and ensuring that the software meets user needs effectively.

✓ Use Case: Login

Actors:
Life Planner User

Preconditions:
User is on the login screen.

Main Flow:

1. User enters registered email and password.
2. System verifies the credentials.
3. If authentication is successful, the user is granted access to the app.
4. System redirects the user to the main dashboard.

Alternative/Exceptional Flows:

- **Invalid Credentials:** If the email or password is incorrect, the system displays an error message prompting retry.
- **Forgot Password:** User clicks “Forgot Password.” → System sends reset instructions via email.
- **Login with Google:** User selects “Login with Google.” → System initiates OAuth → User grants permission → User is authenticated and redirected.

Post conditions:
User is authenticated and granted access to authorized features.

✓ Use Case: Manage Schedule

Actors:
Life Planner User

Preconditions:
User is logged in and has access to the scheduler.

Main Flow:

1. User opens the schedule management interface.
2. User can view existing tasks.
3. User selects an action: Add, Edit, or Remove a task.
4. System processes the chosen action accordingly.

Alternative/Exceptional Flows:

- **Add Task:**
  - User fills task details (e.g., title, date, time).
  - System validates input and saves the task.
- **Edit Task:**
  - User selects a task to modify.
  - System loads task details and saves updates after confirmation.
- **Remove Task:**
  - User confirms deletion.
  - System removes the task from the schedule.

Post conditions:
User's task list is updated as per the changes made.

✓ Use Case: Get Assistance

Actors:
Life Planner User

Preconditions:
User is on the main interface or dashboard of the app.
Chatbot service is available and functional.

Main Flow:

1. User opens a chatbot interface designed for general-purpose interaction.
2. User types a question or message on any topic (e.g., “Tell me a joke,” “What’s the capital of Japan?”).
3. Chatbot processes the input using NLP.
4. Chatbot responds with appropriate and relevant information, guidance, or entertainment.
5. User may continue the conversation in a free-form manner.

Alternative/Exceptional Flows:

- Unrecognized Query:
  - **If the chatbot doesn’t understand the query, it asks for clarification or provides general help prompts.**
- Offline or Service Error:
  - **If the chatbot service is unreachable, the system displays an error message and suggests trying again later.**

Post conditions:
User receives a natural-language response or assistance on a wide range of topics.

✓ Use Case: View Weather

Actors:
Life Planner User

Preconditions:
User is logged in.
Device has internet connection.
Location access is enabled .

Main Flow:

1. User selects “View Weather” from the menu/dashboard.
2. System retrieves the current location (or uses a default/cached one).
3. System fetches weather data from an external API.
4. Weather information (e.g., temperature, condition, forecast) is displayed.

Alternative/Exceptional Flows:

- **Location Denied:** If location access is denied, system prompts user to enable it.
- **API Failure:** If weather data cannot be fetched, system displays a friendly error message or fallback content.

Post conditions:
User sees the current weather and/or forecast for their location.

✓ Use Case: View Motivational Quote

Actors:
Life Planner User

Preconditions:
User is logged in.

Main Flow:

1. User is on home page.
2. System retrieves a quote from API and cache it.
3. Quote is displayed with author.

Alternative/Exceptional Flows:

- **Network Issue:** If quote service fails, a cached quote or fallback a default quote.

Post conditions:
User sees a motivational quote for the day.

✓ Use Case: Write Notes

Actors:
Life Planner User

Preconditions:
User is logged in.

Main Flow:

1. User selects “Write Notes.”
2. System opens a note editor screen.
3. User types in their note content and optionally selects a color.
4. User saves the note.
5. System stores the note in Firestore under the user's account.

Alternative/Exceptional Flows:

- **Sync Issue:** If offline, note is stored locally and synced later.

Post conditions:
A new note is saved and accessible from the user’s notes page.

✓ Use Case: Manage Health Data

Actors:
Life Planner User

Preconditions:
User is logged in.

Main Flow:

1. User opens the “Manage Health Data” screen.
2. User can add or edit information like mood, water intake, or personal notes.
3. System validates input and stores the data in Firestore.
4. User sees a history or summary of their health data.

Alternative/Exceptional Flows:

- **Offline Mode:** Data is stored locally and synced when online.
- **Invalid Input:** System notifies the user if required fields are missing or incorrectly formatted.

Post conditions:
Health data is recorded and viewable for future tracking.

✓ Use Case: Manage Finance Data

Actors:
Life Planner User

Preconditions:
User is logged in.

Main Flow:

1. User selects “Manage Finance Data.”
2. System displays existing financial entries.
3. User can add, edit, or delete income/expenses.
4. System updates records in Firestore and recalculates balance.

Alternative/Exceptional Flows:

- **Offline Mode:** System queues changes for syncing later.

Post conditions:
User’s financial records are updated and synced to the cloud.

Use Case Summary

The Life Planner app enables users to efficiently manage daily tasks, track health and finances, and take notes—all in one place. Key features include task scheduling with reminders, mood and water intake tracking, and a user-friendly notes system. This integrated approach supports better personal organization, productivity, and well-being.

Chapter 5: Database

Introduction to Firebase

Firebase is Google's mobile/back-end platform offering services like Authentication, Cloud Firestore (a real-time NoSQL database), Hosting, and Functions. It lets you focus on UI and business logic while Google handles scaling and security.

Cloud Firestore

Cloud Firestore is a document-store NoSQL database:

- **Documents** hold key/value fields (strings, timestamps, arrays, maps, etc.).
- **Collections** group documents; sub collections nest under documents.
- **Real-time & offline**: SDKs sync changes automatically and cache data when offline.

NoSQL Structure

Unlike SQL (tables & rows), Firestore is schema-less. You don't define tables or columns up front. Any collection or document path you write to is created on the fly:

Example: writing a task under user "UID123"

```
FirebaseFirestore.instance
  .collection('users')        // → creates 'users' if needed
  .doc('UID123')              // → creates document UID123 if needed
  .collection('Tasks')        // → creates 'Tasks' subcollection
  .add({ title: 'Buy milk', dueDate: Timestamp.now() });
```

Database Tree Map

```
(default)
├── Contact
│   └── QP1A.190711.020          ← Device ID document
│       └── messages
│           └── 8ZKdaogHCnab3ARS0pUP  ← Message document
│               ├── deviceId: "QP1A…"
│               ├── email: "…@students.edu.lb"
│               ├── name: "Aliac"
│               ├── message: "nice work …"
```

```
│               └── timestamp: 23 May 2025…
│
└── Users
    └── yTZRS1tErAc0FHwKxZcA0CD2sVc2   ← UID document
        ├── auth_type: "Email"
        ├── createdAt: 27 May 2025...
        ├── customCategories:
        │   ├── "Grocery Shopping"
        │   ├── "food"
        │   └── "Medication"
        ├── displayName: ""
        ├── email: "42230627@students.liu.edu.lb"
        ├── timestamp: 27 May 2025...
        ├── Chats                      ← Subcollection
        │   └── {auto-ID}
        │       ├── isUser: false
        │       ├── text: "Hello! I'm happy to help..."
        │       └── timestamp: 27 May 2025...
        ├── Notes                      ← Subcollection
        │   └── {note-ID}
        │       ├── color: 4290502395
        │       ├── text: "Dear Life Planner Users..."
        │       └── timestamp: 27 May 2025...
        ├── finance                    ← Subcollection
        │   └── 2025-05-27
        │       ├── date: 27 May 2025...
        │       ├── startBalance: 0
        │       ├── endBalance: 295
        │       ├── timestamp: 27 May 2025...
        │       └── transactions:
        │           ├── 0:
        │           │   ├── amount: 300
        │           │   ├── date: 27 May 2025...
        │           │   ├── title: "Finance"
        │           │   └── type: "income"
        │           └── 1:
        │               ├── amount: 5
        │               ├── date: 27 May 2025...
        │               ├── title: "coffee"
        │               └── type: "expense"
        ├── health data                ← Subcollection
        │   └── 2025-05-27
        │       ├── mood: "😊"
        │       ├── notes: "healthy"
        │       ├── waterIntake: 2
        │       └── timestamp: 27 May 2025...
```

```
    └── settings                    ← Subcollection
        └── preferences
            ├── showWeather: true
            └── weatherLocation:
                ├── city: "Laylaki"
                ├── country: "Lebanon"
                ├── lat: 33.8307124
                └── lon: 35.5119233
```

How Data Is Saved

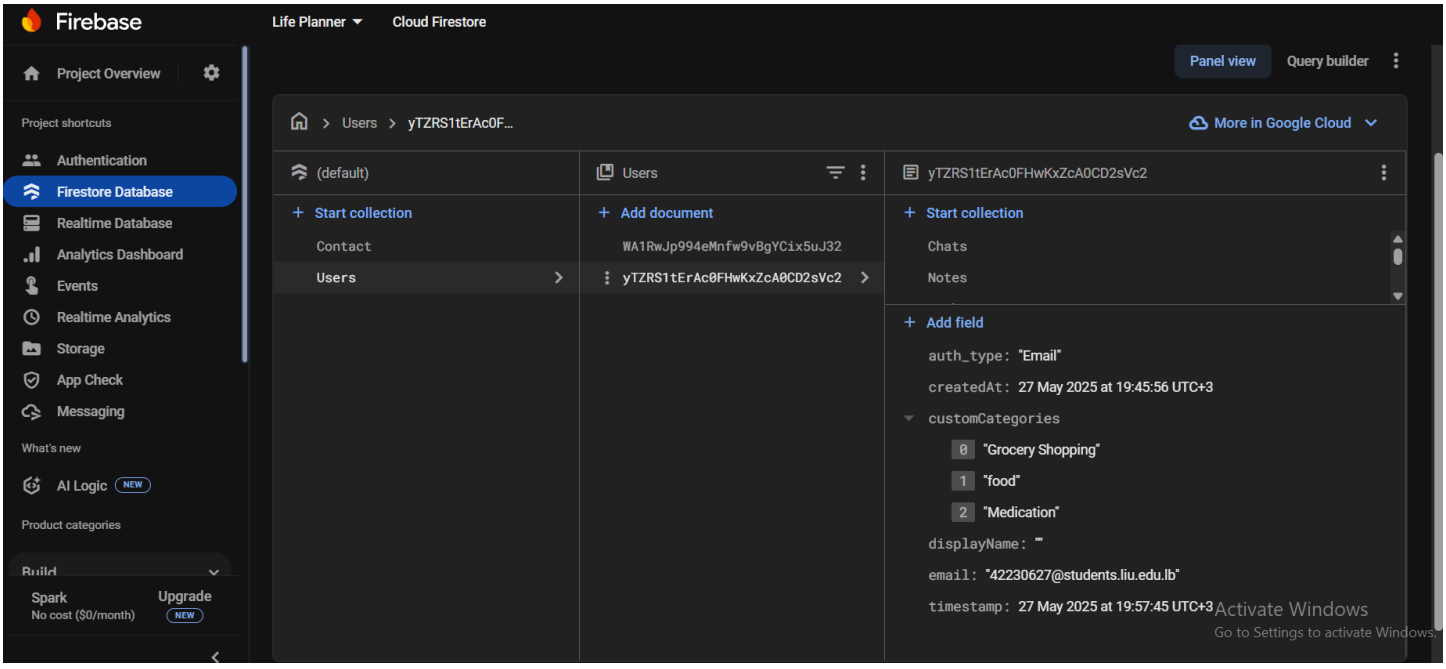
- **Auto-creation**: Writing to any collection/document path auto-generates it—no migrations needed.
- **Subcollections**: Nest related data (messages, tasks, settings) under each user or device.
- **Timestamps**: Use Timestamp.now() for consistent server-side date/time.

Authentication & Security

Firebase Auth manages sign-in (Email, Google, etc.) and issues a unique **UID** per user.

Firebase Console Snapshot (Visual Reference)

Below is a screenshot of the Firebase Console with the Firestore Database open. It visually represents how data is structured and stored in the Life Planner app.



Chapter 6: Conclusion

The Life Planner project successfully achieved its goal of creating a comprehensive personal management app that helps users stay organized, track their health, manage finances, and take notes—all in one place. Throughout the development process, I integrated various Firebase services such as Cloud Firestore, Authentication, and real-time syncing, while ensuring a clean and user-friendly interface in Flutter.

What I Learned

This project provided hands-on experience with:

- Full-stack mobile app development using Flutter and Firebase.
- Managing complex Firestore database structures and relationships.
- Implementing authentication and secure user data access.
- Handling offline capabilities, syncing logic, and data consistency.
- Designing UI/UX flows and improving usability.

It also helped improve my problem-solving skills, especially when handling nested collections, offline caching, and real-time updates.

Limitations

Despite the progress made, there were a few limitations:

- Limited testing on different device sizes and platforms (e.g., iOS).
- Some features, like advanced chatbot NLP or budgeting insights, were implemented in a basic form.
- No deep analytics was integrated due to time constraints.

Future Improvements

Given more time, the following improvements can be made:

- **Integrate NLP (e.g., Wit.ai)** for advanced task management
- **Improve App Optimization** to reduce resource usage and improve responsiveness
- **Use More Robust APIs and Services** once budget allows, for better AI, analytics, and storage
- **Add More Calendar Views** for better visualization of tasks and health logs
- **Advanced Notifications:** Add snooze, priority levels, repeat rules, and smart alerts
- **Theme Customization:** Offer light/dark modes and user-selected themes
- **Data Export and Backup Options** (e.g., PDF, cloud sync)
- **Expand Platform Support** to iOS and prepare for Play Store release
- **Cross-Device Sync:** Improve real-time sync across multiple devices
- **Multi-language Support:** Make the app accessible to non-English speakers
- **Unit & Widget Testing:** Improve reliability with automated test coverage
- **Premium Features for Monetization:** Advanced analytics & insights (e.g., spending trends, mood graphs), Exclusive themes and layout options, Ad-free experience, Priority customer support

References

The following resources were instrumental throughout the development of the Life Planner app:

- pub.dev – for finding and using Flutter packages.
- Flutter Official Documentation
- Firebase Documentation
- Stack Overflow – for resolving coding issues and debugging.
- GitHub repositories and community forums for sample code and best practices.

These resources provided valuable guidance on Flutter development, Firestore integration, and UI/UX design.