Medication Refill Reminder Mobile App

Taylor A Gilg **Felicity M Lester** **Tyler Conwell**

vzu3@txstate.edu ***fml21@txstate.edu*** ***wgc14@txstate.edu***

Introduction

Managing multiple medications is complicated. When medicines can require extra steps to fulfill refills and prescriptions come in all different supplies and dosages, it can quickly become very challenging to stay on top of. Missing refills can mean treatment disruptions and possibly detrimental health effects. With medication adherence being so essential to maintaining proper health, it is critical to have an efficient, simple, and seamless solution to help manage all these factors. Many of the existing products on the market fall short in accessibility with subscription-based models and too many features that quickly overwhelm the user experience. RefillRadar solves this by offering a straightforward, reliable app that enables users to set timely refill reminders, helping users avoid missed doses, last-minute pharmacy trips, and interruptions in their treatment.

Background

The most popular products on the market currently include: Medisafe, Dosecast, and Mango Health. Each mobile app's features are most fleshed out in their own specific target areas. Medisafe is a free app and mostly focuses on medication regimen tracking. Although it does offer refill reminders, users may become overwhelmed by the many other secondary features alongside it including medication interaction warnings and syncing of other health data (ex. blood pressure, blood glucose, etc.). Dosecast is similar to Medisafe but most features, including refill reminders, are hidden behind a subscription paywall. Mango Health is a free solution but is a general health diary and used to track various different health markers (ex. blood glucose, water intake, etc.). Taking these products into account, we determined we wanted to create a solution that is free and comprehensive to specifically refill tracking, not wanting to overwhelm the user with various other extra features. In wanting to fill the accessibility gaps of the current market, we determined our app's target audience:

- Individuals managing multiple medications with different refill schedules
- Patients taking government-regulated medications that require extra steps to refill (e.g., Adderall, opioid pain medications, etc.)
- Elderly individuals or caregivers who need an organized way to track medication refills
- Busy professionals or students who might forget to refill prescriptions on time
- Individuals who rely on non-automatic refills and need reminders to initiate the process

With these demographics in mind, we planned to design an intuitive and user-friendly product and interface that minimizes complexity and the cognitive load of the user. Our goal was to ensure users could efficiently manage their refill schedules with minimal effort and ensure a smooth, frustration-free experience. Focusing on usability and essential functionalities, we wanted notifications that were effective but not disruptive, intuitive UI with clear interaction points and straightforward, few step navigation that could be accessible to a wide demographic.

Project Methodology and Implementation

Our team followed an iterative, user-centered development process to ensure the app met both technical requirements and real-world usability expectations. The process was broken into structured phases with the guidance of our Gannt chart:

Development Methodology: Agile-Inspired Iteration

- We worked in collaborative weekly sprints, with consistent check-ins and communications on personal progress.
- Each member focused on a core section of the app (e.g., UI layout, reminders logic, database integration).
- Feedback from teammates and self-testing drove iterative improvements, ensuring flexibility and continuous enhancement in the app’s development process.

Code Base Folder Structure:

```
refillradarapp
|_.env // Stores key for database encryption in environment variables
|_.gitignore // To hide certain files from and directories from git (ex. .env)
|_README.md // Explains set up process and contributions
|_requirements.txt // Lists all dependencies and needed libraries/tools
|_venv // Virtual environment to keep dependencies and tools consistent
|_backend
|   |_main.py // Running this will create the database and a test user and medication
|   |_notifications.py // Houses implementation for phone notification functionality
|   |_test_notifications.py //Testing environment for prototype notification system
|   |_database
|      |_calc_reminders.py // Calculates medication end date and dates for reminders
|      |_db_instance.py // Shared instance of database functions and security measures
|      |_db_manager.py // Houses functions for secure database manipulation
|      |_models.py // Holds database schema for user and medication fields
|      |_security.py // Stores hashing, encryption/decryption, and password verification
|_frontend
   |_main.py // Running this will launch app UI at login screen
   |_kv
   |   |_add_prescription_screen.kv // UI widget tree for add prescription screen
   |   |_home_screen.kv // UI widget tree for home screen
   |   |_login_screens.kv // UI widget tree for login screen and sign up screen
   |   |_edit_prescription_screen.kv // UI widget tree for screen to edit prescription details
   |   |_prescription_screen.kv // UI widget tree to display specific medication info
   |   |_settings_screen.kv // UI widget tree for logout button and delete data button
   |   |_calendar_screen.kv // UI widget tree for calendar screen to visualize reminders
   |_screens
      |_add_prescription_screen.py // Integrates adding prescription function to UI
      |_home_screen.py // Integrates displaying medication list with UI visuals
      |_login_screens.py // Integrates user authentication and account creation w/ UI
      |_edit_prescription_screen.py // Integrates editing of medication info w/ UI
      |_prescription_screen.py // Integrates prescription info display w/ UI
      |_settings_screen.py // Integrates logout function with UI
      |_calendar_screen.py // Demos visual reminder functionality w/ mock example setup
```

Screen Functionality:

Login & Create Account Screens (login_screens.py & login_screens.kv):

- These screens enable users to enter account credentials such as username and password.
- Pressing the login button calls authenticate_user() in db_manager.py which calls for checks to validate if the username exists in the database and calls upon verify_password() in security.py which verifies the password on its stored hash.
- Creating an account with the sign up screen calls upon register_user() in db_manager.py. The method checks for any existing matching username or email and proceeds to hash the password and encrypts the email via encrypt_sensitive_data() in security.py before storing all the entered values in the database.
- We wanted the login and account creation process to be straightforward and require as few steps as possible considering our audience of users could be older and less tech savvy. We wanted to balance security and convenience and want to involve email in the future for two factor authentication and account recovery.

Home Screen (home_screen.py & home_screen.kv):

- This screen displays a user's list of stored medications via clickable cards. Entering the screen calls get_user_medications() in db_manager() (uses user_id passed along by the login screen after verification to search for the user's medicines). This returns all the user's medication objects. These are then cycled through and attributes such as day supply, the date of the medication's first reminder, and icon and color selection are used to display on the cards.
- It was important to us to give users the option to visually define medications by icons and color coding to make checking in on the status of medications more efficient and convenient, lessening cognitive load.
- The "refill" button on the medication cards will turn to "refilled" when clicked. It is not implemented on the backend but in the future we want this to reset the start date of the medication for the day it was clicked to enable convenient overturn of scheduling for long term prescriptions.
- We also wanted to provide a few of the medication's details like supply and first reminder date to give an "at a glance" view for users on the go.
- To provide a more comprehensive look at all of a medication's details, a user just needs to click the medication card and it will bring them to the prescription info screen.

Prescription Screen (prescription_screen.py & prescription_screen.kv):

- Upon entering this screen calls get_medication_by_id() in db_manager (using med_id passed from selected card on home page) which extracts and decrypts all the details of a medication so that load_medication_data() can apply them to the UI widgets to display them for the user in a read only fashion.
- Pressing the enter button at the bottom of the screen sends the user to the edit prescription screen.
- Pressing the back arrow in the top left or the cancel button will bring you back to the home page.

Edit Prescription Screen (edit_prescription_screen.py & edit_prescription_screen.kv):

- The loaded values of the prescription info screen still show for reference but can now be changed via dropdowns and button selection. The save button calls edit_prescription() which extracts all the values in the widgets, default and edited, and calls update_medication() in db_manager to edit the variables of a specific, existing medication in the database.
- Pressing save, cancel, or the back arrow button in the top left will bring you back to the prescription info page where you came from. We wanted to make sure screen traversal was simple to visualize and intuitive. Also, when you save, when back on the prescription screen, you will already see the changes you made, giving users a responsive and satisfying experience.

Settings Screen (settings_screen.py & settings_screen.kv):

- This screen enables the user to logout and return to the login screen. Another button present is the Delete User Data button that we would like to implement in the future. This would allow a user to delete their user and medication data for security purposes (ex. Getting a new device, in case of account compromisation, etc.)

Add Prescription Screen (add_prescription_screen.py & add_prescription_screen.kv):

- Accessible via the plus button at the bottom of the home screen, a user can create a new medication by entering medication information and reminder preferences to be stored in the database. Each data field is highlighted in the user handbook but when the add button is clicked, add_prescription() is called to extract all the values from the widgets and send this medication object to add_prescription() in db_manager. This method takes in the object and uses calc_reminder.py ‘s process_medication_reminders() to calculate the end date of

the medication supply and the dates for push notifications to alert users in advance of their need for a refill coming up.
- We wanted to make the widgets intuitive and clear in what they were asking for. In these efforts, we included tooltips with popups with extra information to elaborate on what kind of information the system is asking for and expecting (ex. Information icons next to the dosage and supply sections).

Calendar Screen (calendar_screen.py & calendar_screen.kv)

- This screen is accessible via the calendar icon at the bottom of the home page. It is not implemented yet with the backend functionalities but is a working demo of its intended functionality.
- On dates that mark reminders or the end of a medication supply, it will be highlighted blue. When clicked, a popup will appear displaying the reminders that are scheduled for the day or the possible last day of supply for a medication.
- We think it is very important to grant a way for users to easily and conveniently visualize their refills coming up and the reminders they have scheduled in advance of end of supply to aid in better management of multiple medications.

Notifications.py and Test_Notifications.py:

- Due to technical difficulties with trying to package our code to be able to be run on an Android emulator, we were unable to test our push notification system on a mobile device. Both files currently utilize windows notification functionality but can be utilized via Kivy's simple push notification functionalities if able to be packaged and run on an android device. Test_Notifications.py sets up a testing environment for Notifications.py with the windows based notifications to test notifications in a smaller timetable than would be realistically executed in the app's functional lifespan.

Human Factors and HCI Principles

Our app integrates several key Human-Computer Interaction (HCI) and Human Factors principles to support usability, accessibility, and a smooth user experience:

- Consistency: Button styles, icon usage, and layout spacing remain uniform across all screens, helping users build familiarity and confidence as they navigate.
- Visibility of system status: When users add a prescription, select a color or icon, or interact with dropdown menus, immediate visual feedback (such as highlighting, borders, or dialogs) confirms their actions.

- Error prevention and recovery: Text inputs use filters, dropdowns limit invalid choices, and confirmation dialogs help prevent unintended submissions.
- Flexibility and efficiency: Scrollable layouts and adaptive spacing make the app functional on both mobile and desktop environments, while allowing for quick access to frequently used options.
- Affordance and discoverability: Icons are clearly labeled with tooltips, input fields include hint text, and all interactive elements follow familiar design conventions (e.g., pill buttons for selection).
- Accessibility and simplicity: Large touch targets, readable fonts, and simplified flows make the app usable for users with varying levels of tech experience.

Together, these design choices support the cognitive and physical needs of users, making the app intuitive, predictable, and comfortable to use.

Tools and Technologies Used:

- Frontend / UI:
  - Kivy / KivyMD – Used to build a responsive and customizable mobile interface for the app.
- Programming Language:
  - Python 3.13 – Core language for implementing app logic, user interactions, and backend functions.
- Database & ORM:
  - SQLite – Lightweight embedded offline database for local prescription data storage.
  - SQLAlchemy – ORM (Object Relational Mapper) for managing database schema and performing queries using Python objects.
- Security & Encryption:
  - cryptography.fernet – Used to encrypt and decrypt sensitive information like passwords and prescription data.
  - hashlib / base64 / secrets – Utilities for hashing, encoding, and generating secure tokens.
- Environment Configuration:
  - dotenv (.env) – Stores sensitive variables (e.g., encryption keys) safely and outside the main codebase.
- Dependency Management:
  - requirements.txt – Keeps track of all external libraries used, ensuring consistent environments across machines.

- Version Control & Collaboration:
    - Git & BitBucket – Used for code versioning, collaboration, and managing branches.
    - GitKraken – A visual Git client that simplifies staging, merging, and reviewing pull requests.
- Development Tools:
    - Visual Studio Code – Main code editor for writing, debugging, and testing.
    - Google Drive / Google Docs – Central hub for documentation, design drafts, and planning resources.
- Communication:
    - Discord – Daily team communication, issue tracking, and real-time development updates.

Design Process

- The UI was designed with mobile-first principles, ensuring accessibility across screen sizes.
- We used a hierarchical component-based layout approach in KivyMD to support dynamic content like dropdowns, scroll views, and effective separation of widgets.
- Design decisions prioritized clarity, touch responsiveness, and intuitive grouping of actions (e.g., medication customization, supply selection, reminders).

Testing & Debugging

- Each team member ran manual UI testing during development to identify layout issues and edge cases.
- Debugging was done using console logging, screenshot analysis, and trial-and-error iteration.
- Key visual and functional implementation bugs resolved including in widget functionality, user expectations of UI interaction, and frontend interaction with database.

Results and Findings

To test the usability and effectiveness of our app, we conducted user surveys with friends and family and observed them traverse the app on their own accord. The consensus among users was that the purpose of the app was clear, all the features they expected were present, and pathways were intuitive and easily accessible. Compared to similar solutions on the market, our app did not overwhelm users with excess features and tasks did not take searching through multiple layers of menus to achieve. They enjoyed the straightforward nature of the app and the prioritization of functional visuals over ad space. A common critique though was that it should be indicated more clearly for the add prescription screen and edit prescription screen what fields are mandatory. In the future, we wish to implement either more visual indicators or a fielding method that shows a popup when the user tries to submit when they don't have all the necessary data filled in. With this, we want to ensure a more responsive and reliable experience overall.

Insights Gained

- Some of the unexpected patterns that we identified were that users skipped certain sections of the add prescription section inferring that they would be autofilled, which informed future design adjustments.
- Seeing the user's interactions with the screens of our application helped support some of our initial assumptions regarding the users adoption to using the predefined medication information like Dosages and Supply days.
- We learned that users preferred fewer steps to set up a medication so having a but plus symbol with only two major screens really screamlines that part of the process.
- When the calendar was first implemented the users did not know what day they were at currently, so to help we made the current day a different color that way they knew when to start counting or subtracting days quicker.
- Adjustments to the settings screen made the process streamlined with only needing to have a "delete all information" button and a "sign out" button since users instinctively used the settings icon when trying to find these features.
- The development process emphasizes the importance of consistent communication to avoid any integration conflicts like over writing someone's work on a specific part of the project to midgate stepping on someone else's toes.
- Testing the application also confirms that our approach to handling the adding and editing of medications was the best option to reduce the amount of frustration users were having as well as minimizing any crashes on the application part.

Conclusion

Our project successfully demonstrates a user-friendly and secure prescription management app that allows users to store, manage, and receive reminders for their medications. The core contributions include the design and implementation of an intuitive UI using KivyMD, the integration of a secure encrypted SQLite database and backend using SQLAlchemy, and the incorporation of customizable reminders and user validation. We also ensured the application runs fully offline, which increases security and accessibility for users.

Throughout development, we learned the importance of user experience in healthcare-related apps — features like simple pathways, clear UI interaction, and responsive design made a significant difference in usability. We also gained firsthand experience navigating version control challenges as a team, especially in managing merges, branches, and collaboration via GitKraken and Bitbucket.

In the future, this app could be extended to include notifications on mobile devices, enabling downloading of personal data, full visualization of reminders and ends of supply via calendar, analytics on medication habits, and possibly caregiver dashboards for dependent users. With further development, this could become a valuable tool not only for individuals but also for clinics or small practices needing lightweight medication tracking for their patients.

Overall, this project pushed our technical skills and communication as a team, and we're proud of the functional demonstration we were able to achieve.

References

Choudhry NK, Krumme AA, Ercole PM, et al. Effect of Reminder Devices on Medication Adherence: The REMIND Randomized Clinical Trial. *JAMA Intern Med.* 2017;177(5):624–631. doi:10.1001/jamainternmed.2016.9627

Lam, Wai Yin, Fresco, Paula, Medication Adherence Measures: An Overview, *BioMed Research International*, 2015, 217047, 12 pages, 2015. https://doi.org/10.1155/2015/217047

Kristin Andersson, Arne Melander, Carin Svensson, Owe Lind, J. Lars G. Nilsson, Repeat prescriptions: refill adherence in relation to patient and prescriber characteristics, reimbursement level and type of medication, *European Journal of Public Health*, Volume 15, Issue 6, December 2005, Pages 621–626, https://doi.org/10.1093/eurpub/cki053

Appendix

User Manual

RefillRadar is a mobile-friendly app designed to help users manage prescriptions and set custom medication reminders. Below is a quick guide to navigating and using the app:

Login & Sign Up

- Running backend/main.py will generate the application's database and generate a test user and medication. After next running frontend/main.py, you will be presented with the app's **Login Screen**.
- Here, you can login to the pre-generated user (username: test_user, password: secure_password) or you can click the "Create an Account" button to be brought to the **Create Account Screen**.
- On the sign up screen, you can enter credentials to create your own account. After clicking "Create Account" you can click "Back to Login" to enter your credentials and login.

Home & Navigation

- After logging in, you will land on the **Home Screen.** Here, you will see a list of your medications. If you logged in as the test user, a medication is already loaded in to see.
- Tapping the large plus button at the bottom of the screen will bring you to the **Add Prescription** screen.

Adding a Prescription (* Indicates necessary selection)

1. **Select a Medication Category**: Choose an icon that best represents your medication (e.g., pill, injection).
2. **Pick a Color Tag**: Assign a color to easily identify the prescription visually.
3. **Enter Medication Name***: Type in the medication name in the outlined text field.
4. **Indicate your Dosage*:** Enter the max dosage you take per regular interval (eg. taking 1 pill every 2 days).
5. **Indicate your Supply*:** Click the button that matches your prescription's duration (30-Day, 60-Day, 90-Day, 100-Day).
6. **Indicate Start of Medication*:** If you take your medication on an hourly basis, enter the time of your first dosage taken, otherwise enter the date of your first dosage (one or the other is required).

7. **Indicate your Reminder Preferences*:** The first line of input is to schedule the first notification that will remind you of your ending supply. The second line indicates how many secondary notifications you want and the interval of time between them.
8. *(Optional)* **Add Notes**: Include any special instructions, timing, or warnings under “Medication Details.”
9. The **Cancel** button (along with the arrow at the very top left) will return you to the home page. Press the **Add** button after you’ve made all necessary selections and you will be returned to the home screen where you can see your new medication.

Responsive Design

The window size of the interface is by default set to 360x640, similar to a mobile phone screen, but widgets have been designed to be able to size dynamically. We were also attentive to add visual indicators of user selection and results of user actions where we could (eg. selection bolding, adaptive dropdowns, updating display in home screen).

Database Schema (Technical Overview)

The RefillRadar app uses **SQLAlchemy** as its Object Relational Mapper (ORM) to define and manage its database models.

Core Models:

- **User**
  - id
  - created_at
  - username
  - email
  - password_hash

- **Medication**
  - id
  - created_at
  - user_id
  - medication_id
  - med_name
  - icon
  - color
  - max_dosage
  - dosage_interval

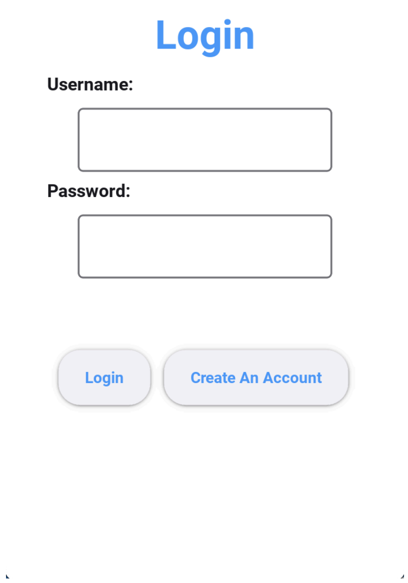
  - dosage_frequency
  - total_supply
  - start_date
  - start_time
  - end_date
  - duration_prior
  - reminder_unit
  - repeat_reminders
  - repeat_intervals
  - repeat_unit
  - reminder_dates
  - details

Relationships:

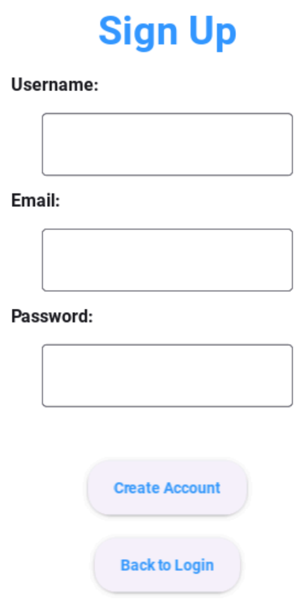
- One User can have many Medications.
- Each or all Medication can be found under a user.

This structure ensures that user data is separate but can aid in identifying a list of medications and present and edit a user's respective medications.

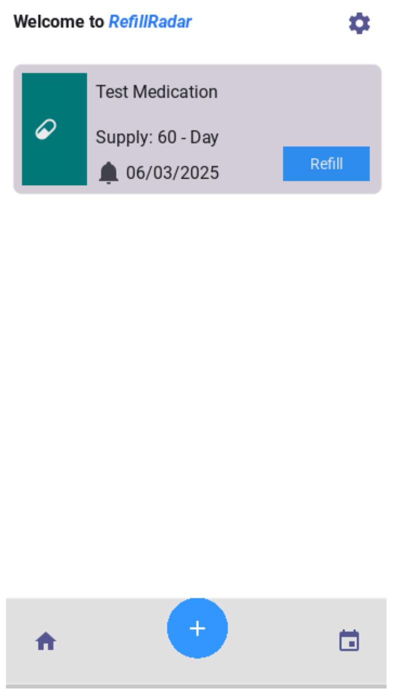
INITIAL LOGIN MOBILE (1)



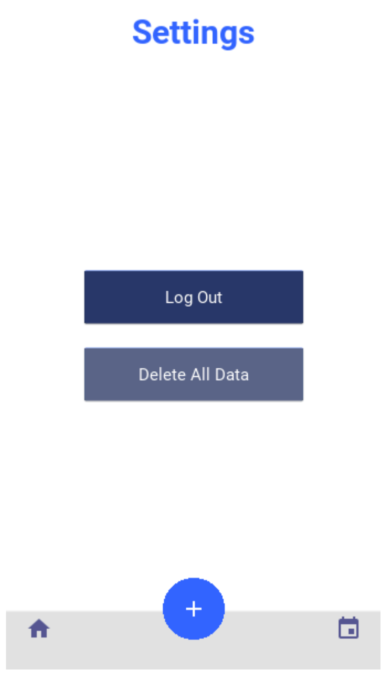
INITIAL SIGNUP MOBILE (2)



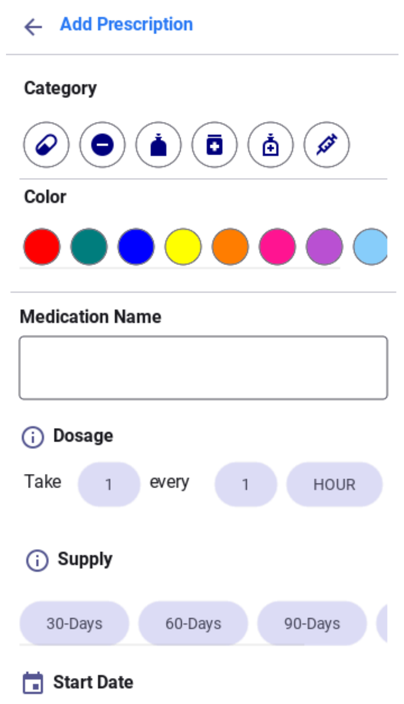
EXAMPLE HOME SCREEN MOBILE



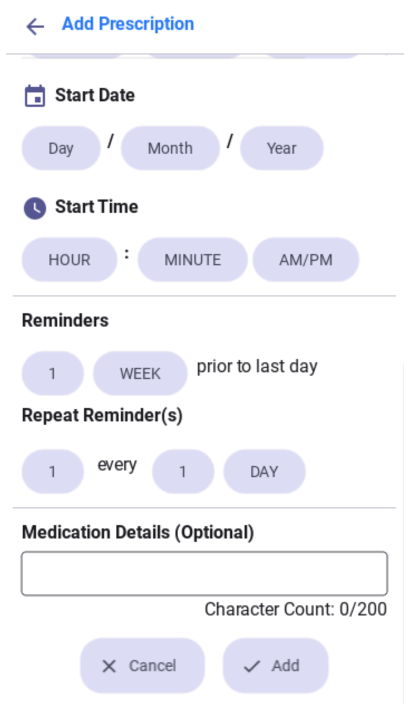
SETTINGS MOBILE



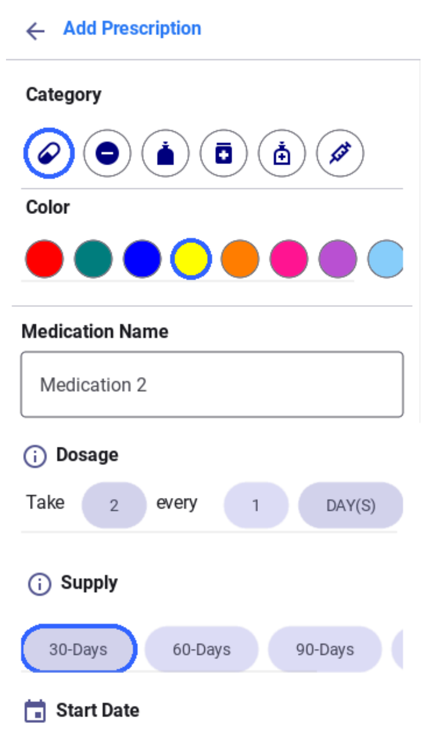
INITIAL PRESCRIPTION MOBILE (1)



INITIAL PRESCRIPTION MOBILE (2)



EXAMPLE INPUT MOBILE (1)



Add Prescription

Category

Color

Medication Name

Medication 2

Dosage

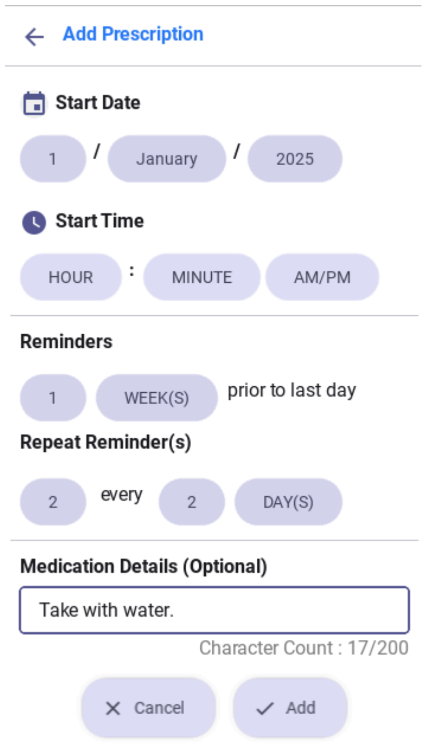
Take 2 every 1 DAY(S)

Supply

30-Days 60-Days 90-Days

Start Date

EXAMPLE INPUT MOBILE (2)



Add Prescription

Start Date

1 / January / 2025

Start Time

HOUR : MINUTE AM/PM

Reminders

1 WEEK(S) prior to last day

Repeat Reminder(s)

2 every 2 DAY(S)

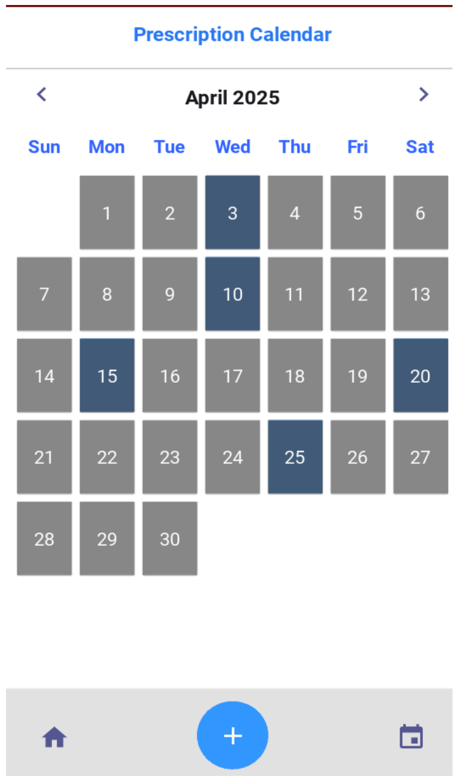
Medication Details (Optional)

Take with water.

Character Count : 17/200

Cancel Add

EXAMPLE CALENDAR MOBILE (1)

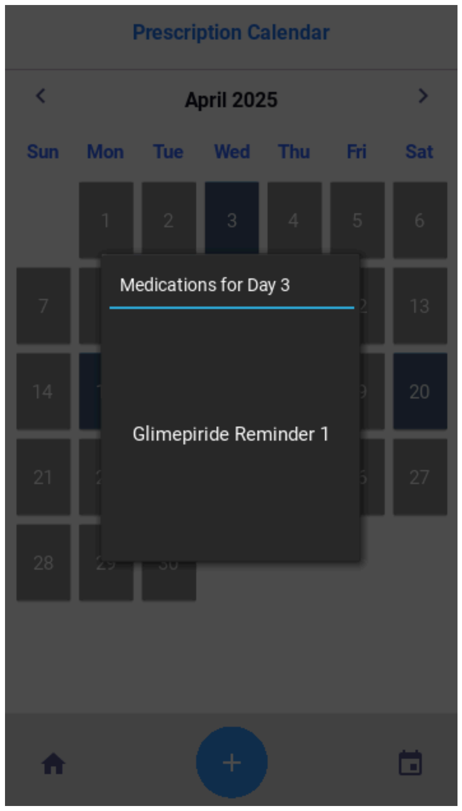


Prescription Calendar

April 2025

Sun	Mon	Tue	Wed	Thu	Fri	Sat
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30				

EXAMPLE CALENDAR MOBILE (2)



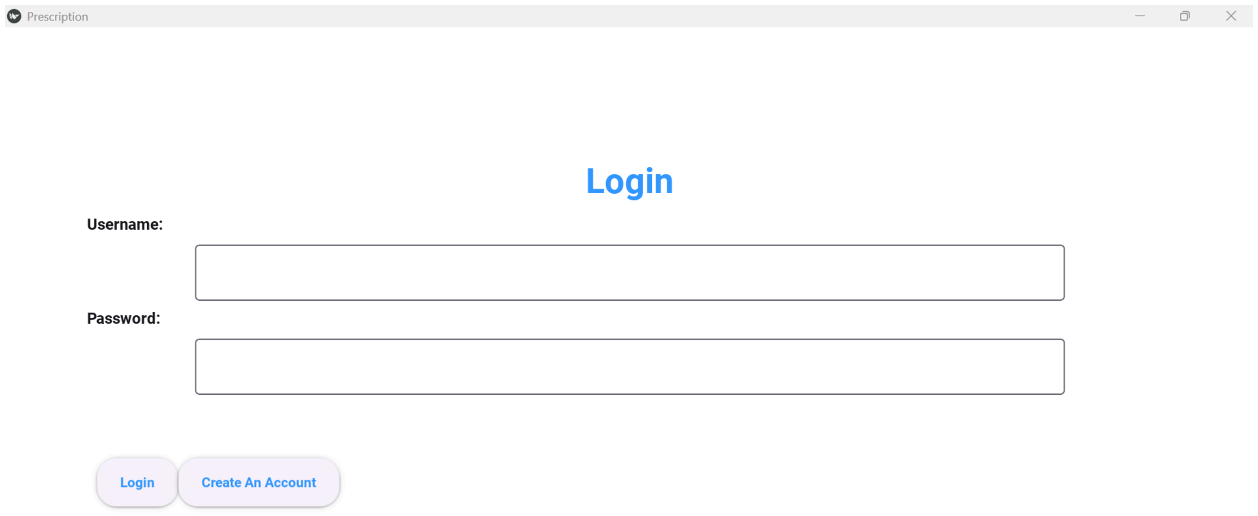
Prescription Calendar

April 2025

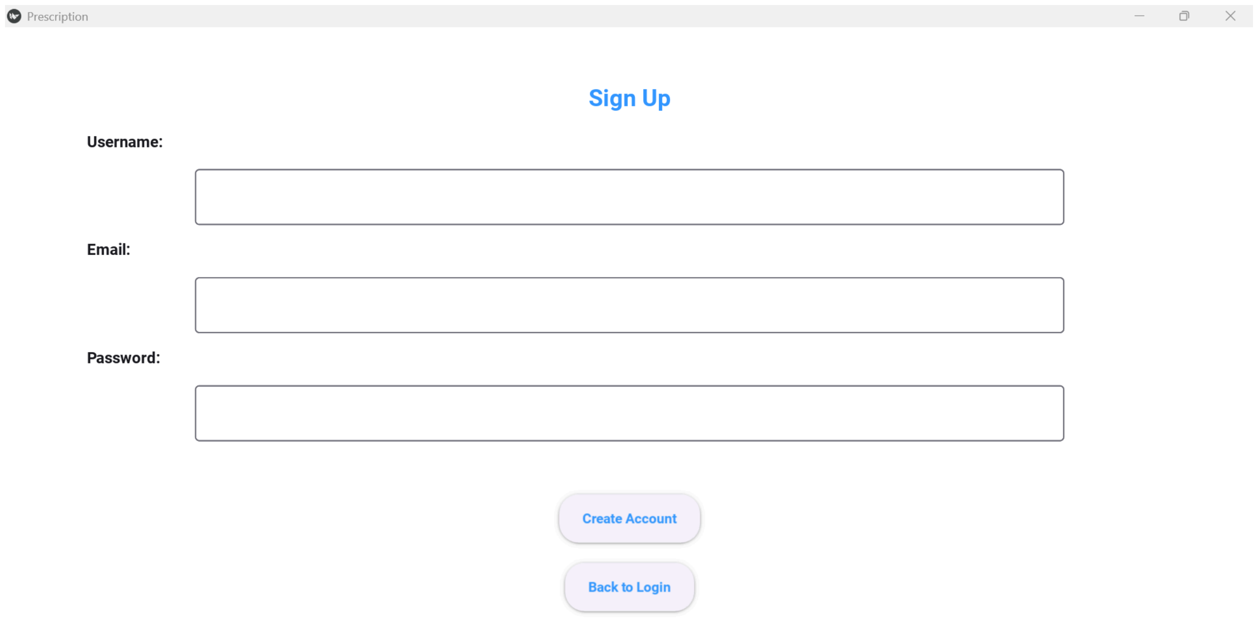
Medications for Day 3

Glimepiride Reminder 1

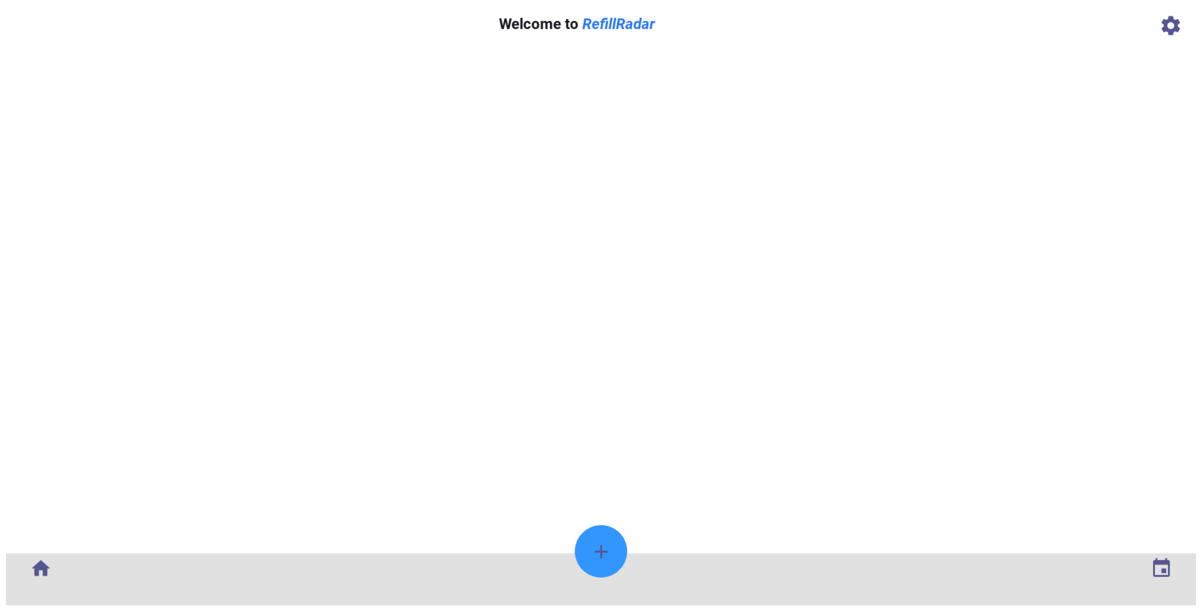
INITIAL LOGIN FULL SCREEN (1)



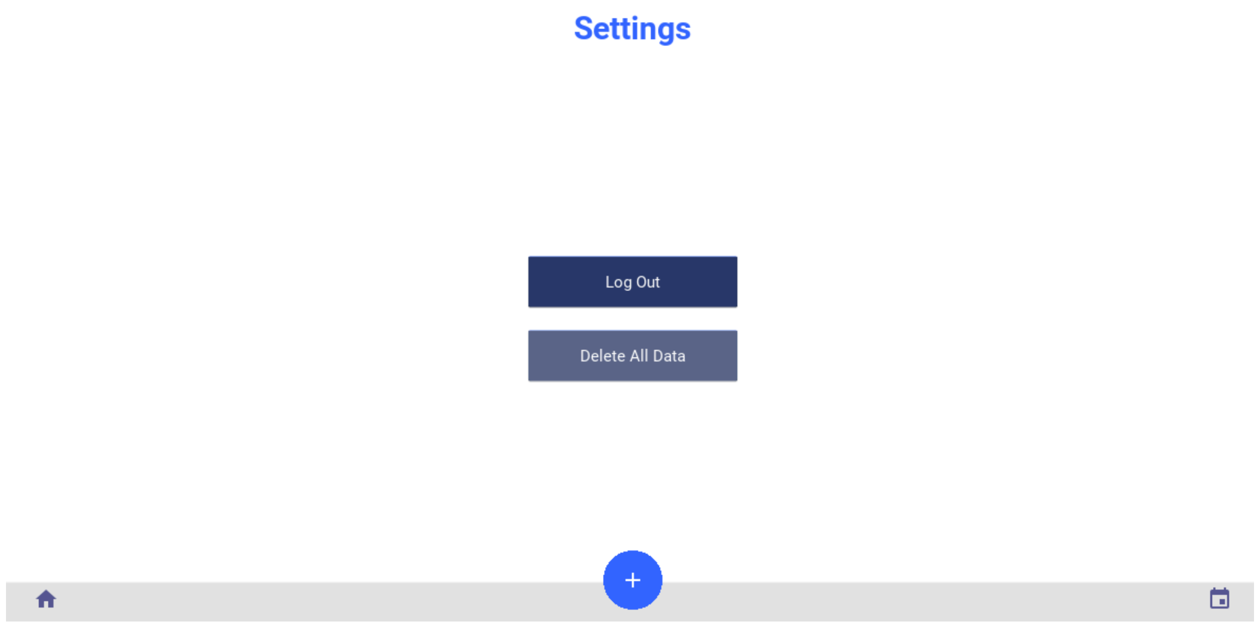
INITIAL LOGIN FULL SCREEN (2)



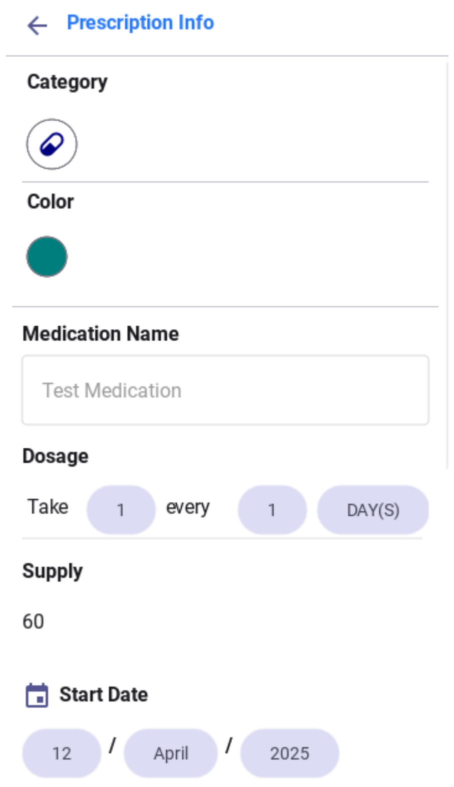
INITIAL HOME SCREEN FULL SCREEN (1)



INITIAL SETTINGS SCREEN FULL SCREEN (2)



INITIAL EDIT RX MOBILE (1)


Prescription Info

Category

Color

Medication Name

Test Medication

Dosage

Take 1 every 1 DAY(S)

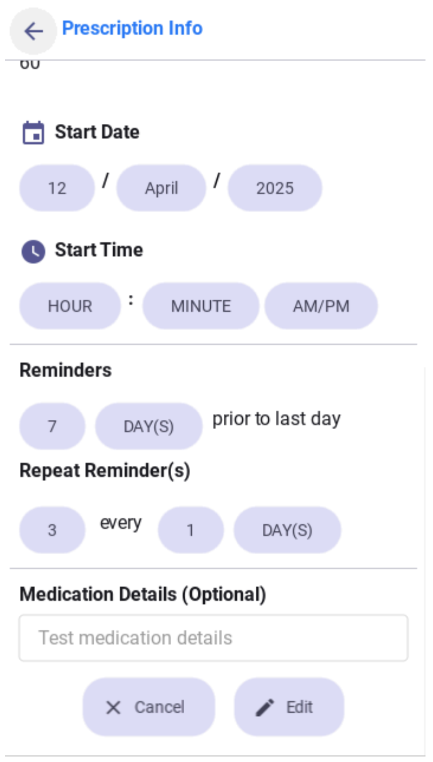
Supply

60

Start Date

12 / April / 2025

INITIAL EDIT RX MOBILE (2)


Prescription Info

60

Start Date

12 / April / 2025

Start Time

HOUR : MINUTE AM/PM

Reminders

7 DAY(S) prior to last day

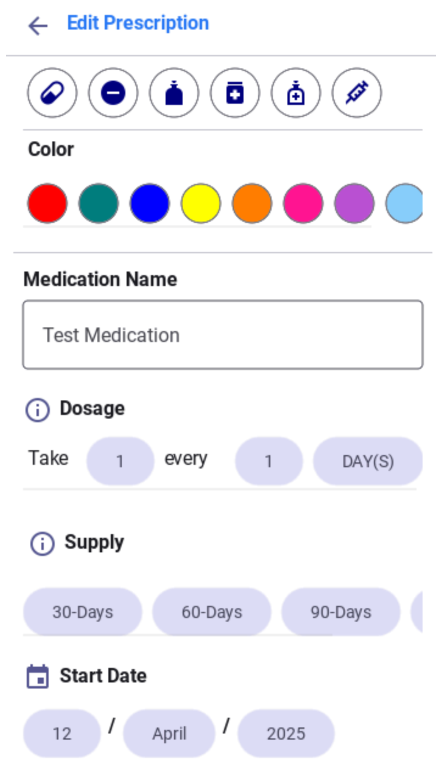
Repeat Reminder(s)

3 every 1 DAY(S)

Medication Details (Optional)

Test medication details

EDITED RX MOBILE (1)


Edit Prescription

Color

Medication Name

Test Medication

Dosage

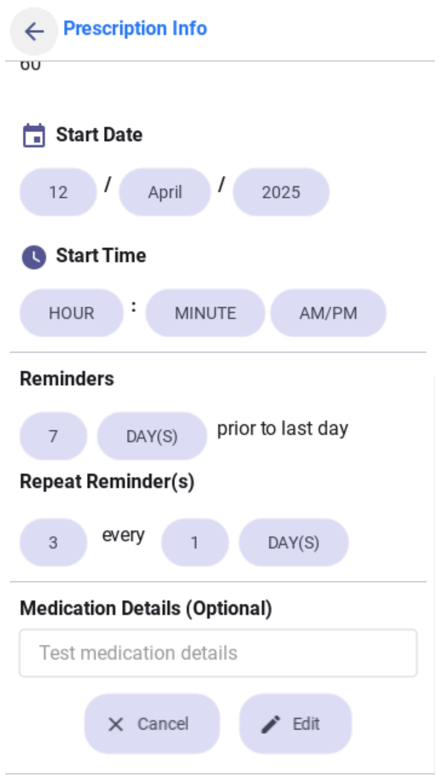
Take 1 every 1 DAY(S)

Supply

Start Date

12 / April / 2025

EDITED RX MOBILE (2)


Prescription Info

60

Start Date

12 / April / 2025

Start Time

HOUR : MINUTE AM/PM

Reminders

7 DAY(S) prior to last day

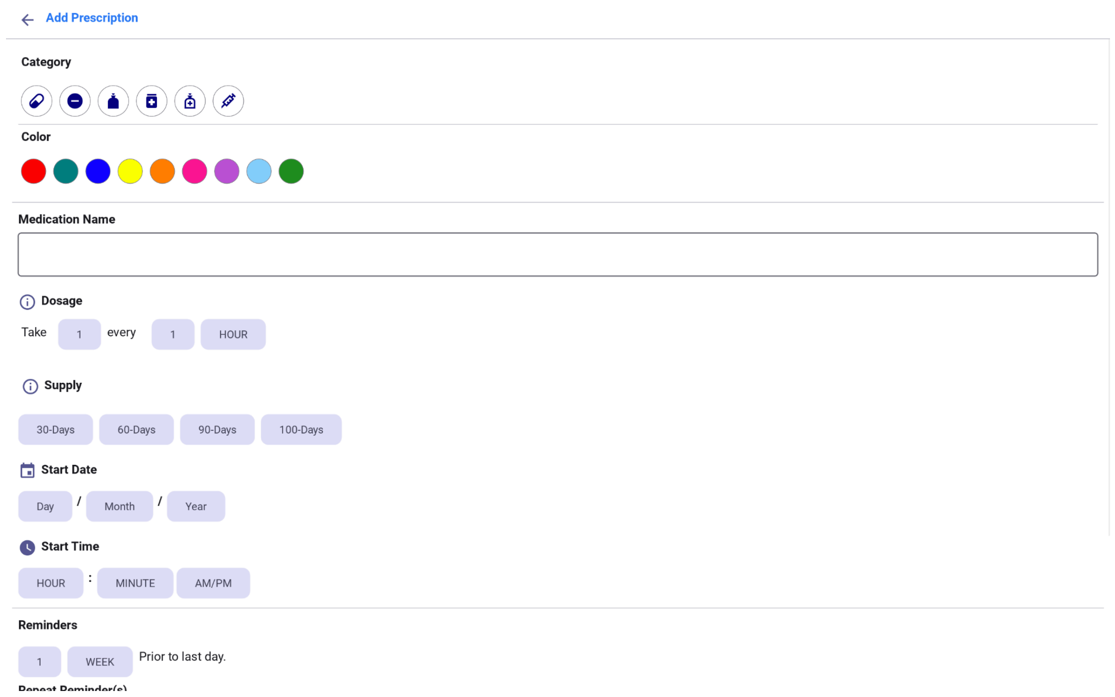
Repeat Reminder(s)

3 every 1 DAY(S)

Medication Details (Optional)

Test medication details

INITIAL FULL SCREEN (1)



Add Prescription

Category

Color

Medication Name

Dosage

Take 1 every 1 HOUR

Supply

30-Days 60-Days 90-Days 100-Days

Start Date

Day / Month / Year

Start Time

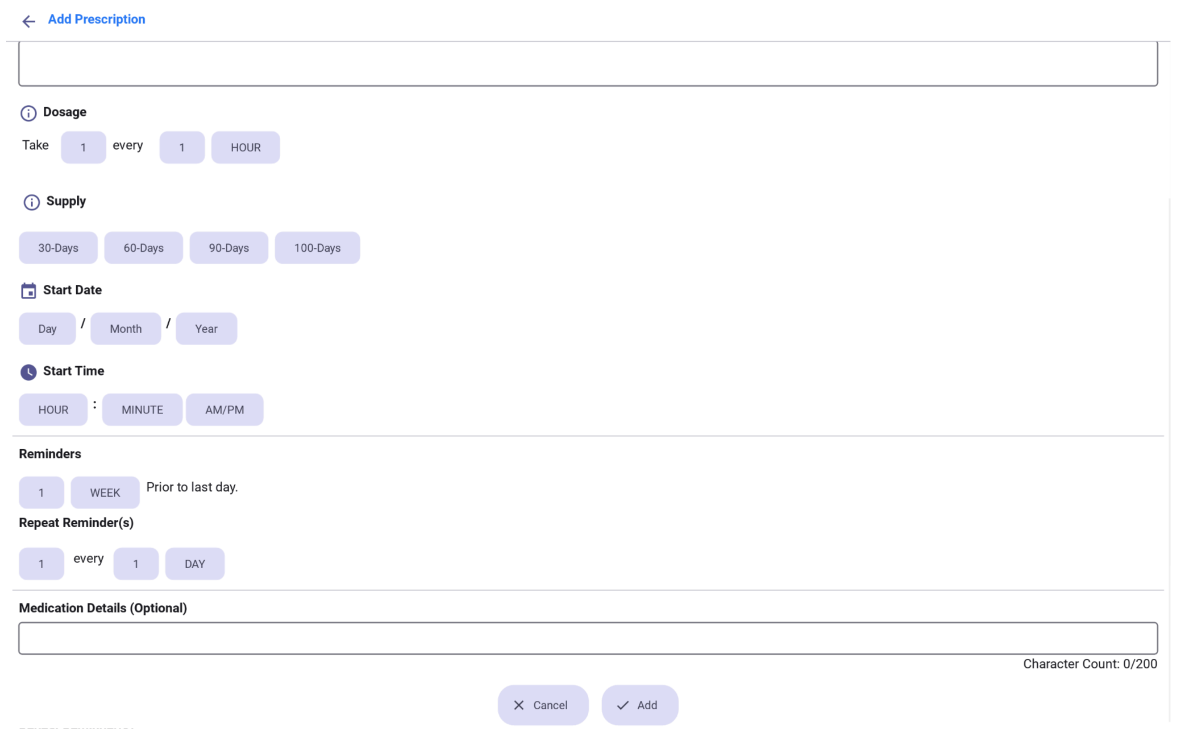
HOUR : MINUTE AM/PM

Reminders

1 WEEK Prior to last day.

Repeat Reminder(s)

INITIAL FULL SCREEN (2)



Add Prescription

Dosage

Take 1 every 1 HOUR

Supply

30-Days 60-Days 90-Days 100-Days

Start Date

Day / Month / Year

Start Time

HOUR : MINUTE AM/PM

Reminders

1 WEEK Prior to last day.

Repeat Reminder(s)

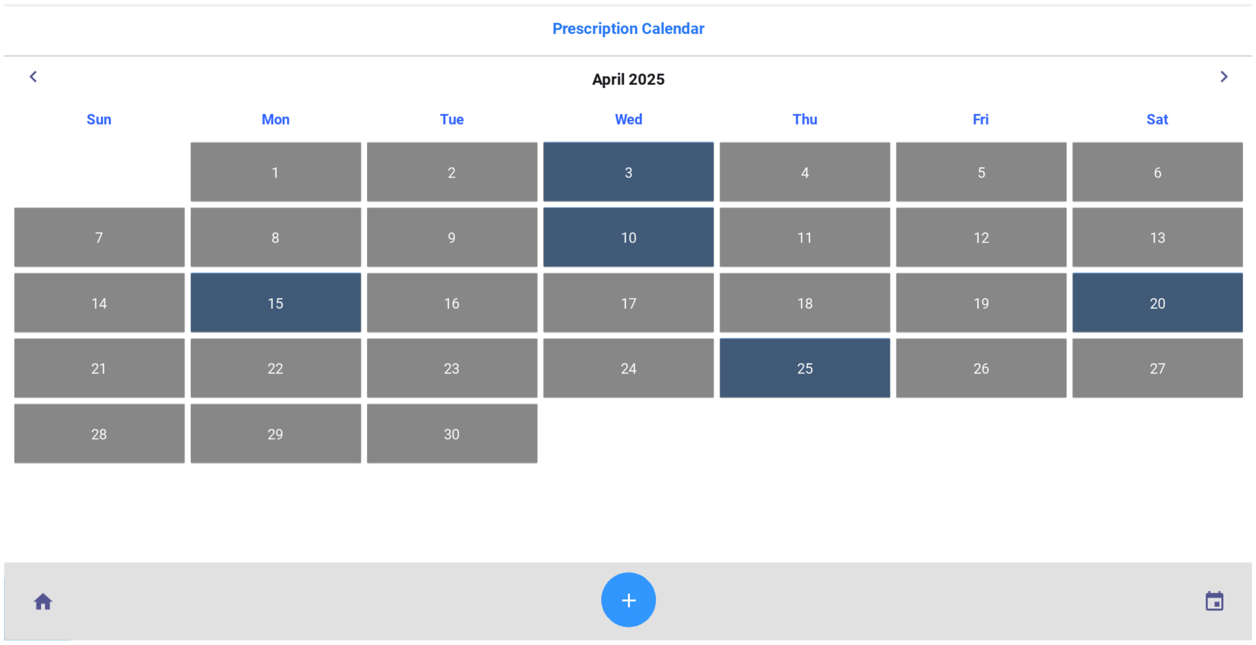
1 every 1 DAY

Medication Details (Optional)

Character Count: 0/200

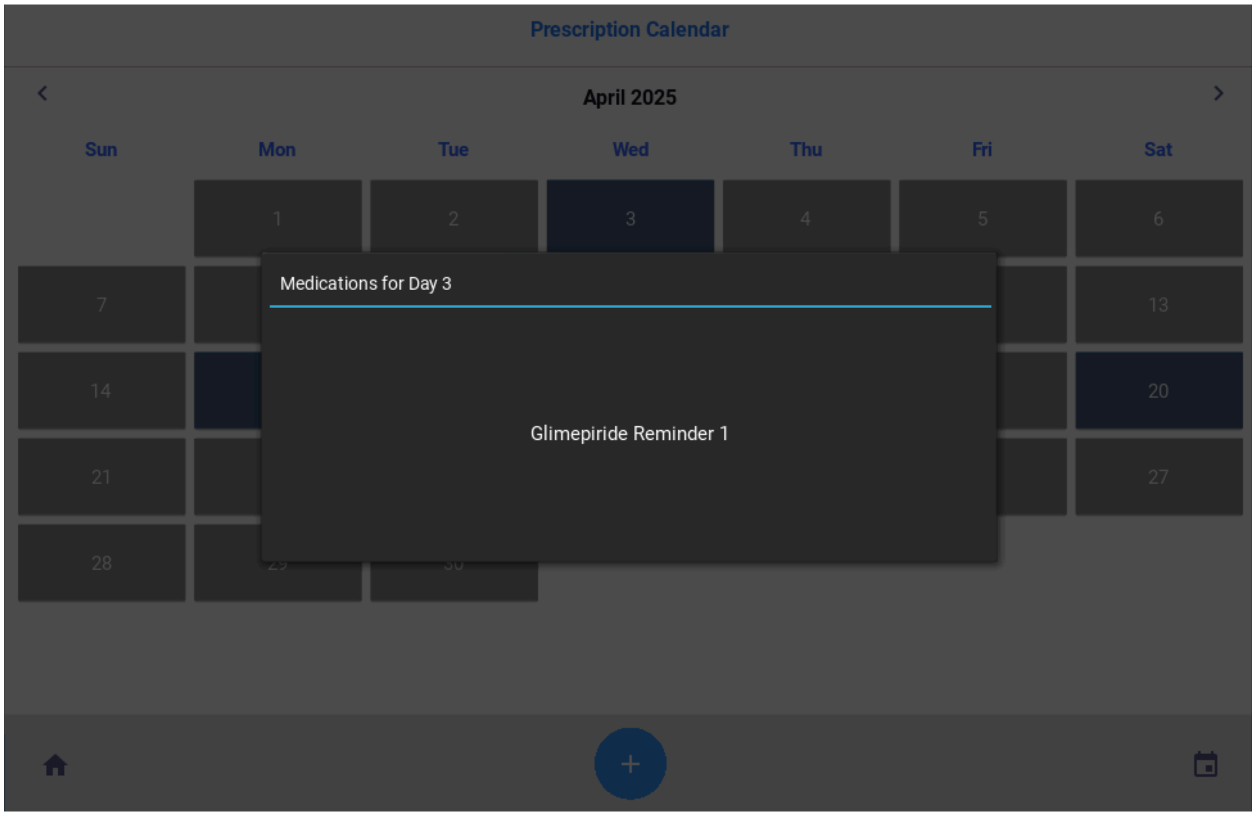
Cancel Add

CALENDAR FULL SCREEN (1)



	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30				

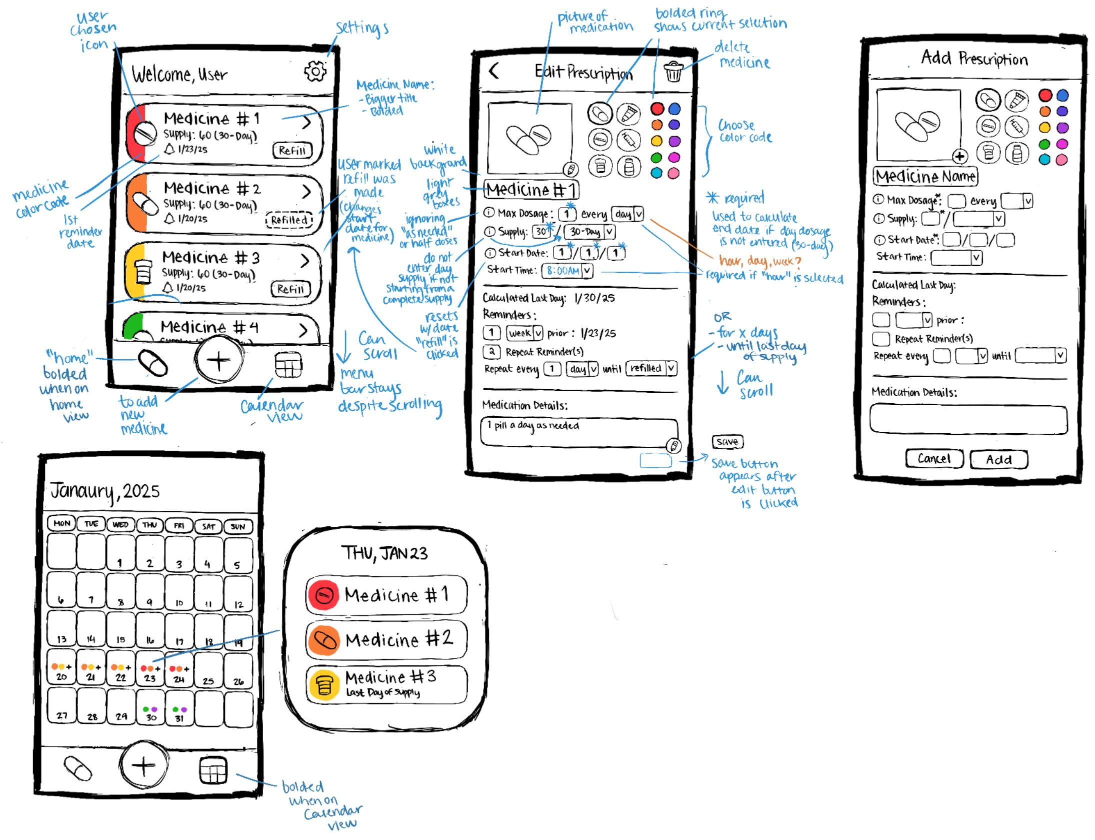
CALENDAR FULL SCREEN - DATE SELECTED (2)



Medications for Day 3

Glimepiride Reminder 1

UI Screens Developmental Sketch:



Team Members Contributions:

Felicity Lester: Lead UI/UX Developer & Logic Integration

- Created a mobile-responsive layout using KivyMD with consistent spacing, alignment, and adaptive design for desktop, tablet, and phone views.
- Designed and implemented the custom header, tonal buttons, and consistent scroll behavior across multiple sections.
- Built all interactive components for adding a prescription, including medication icon selectors, color tags, and supply duration buttons.
- Implemented scrollable horizontal button sections for medication categories and color selection.
- Designed and implemented the dynamic reminders layout with dropdown menus, repeat intervals, and user-friendly phrasing like "prior to last day."
- Ensured proper scroll behavior, sizing, and real-time label updates.
- Fixed critical layout and scroll bugs that caused crashes or distorted UI rendering.

- Adapted dozens of widgets for adaptive sizing, alignment consistency, and screen responsiveness.
- Connected interactive fields with interactive UI logic (e.g., dosage and supply selection adapt and display user choices).
- Added consistent spacing, icons, and character limits.
- Created a clean layout with visual hierarchy and instructional labels.

Tyler Conwell:

- Developed Kivy wireframes for home and calendar screens.
- Implemented a dynamic calendar layout that adapts to the current month, day, and year that the user needs, allowing navigation between each month.
- Populated the calendar with a dynamically scaling grid that has buttons for each day of the month. Including empty placeholders for the days that are before the start of the next month to replicate a natural calendar.
- Highlighted specific days with medication reminders, as well as visually differentiated the current day of the week with a blue background for better visible feedback.
- Added navigation buttons to switch between the months of the year which update the calendar dynamically.
- Created a popup for medication details that when pressed for that day display the medications scheduled. Ensuring that both the popup and the grid align for a responsive layout.
- Implemented the settings buttons “log out” and “delete all data” to allow the user to navigate back to the login screen as well as functionally allow mock data to be removed.
- Allow the user to functionally execute mock data to delete all user data in their account once they confirm with the popup buttons “cancel” or “Delete” which will serve as their warning message before continuing their actions.
- Ensure the proper alignment and spacing of the popup content, including the message and the buttons.

Taylor Gilg:

- Developed plans for UI screens and created UI development sketches.
- Implemented offline database schema, data encryption/decryption protocols, user authentication, and database interaction functionalities like adding medications and displaying a user’s medications (db_manager.py, models.py, security.py).
- Streamlined system interaction points with database with db_instance.py.

- Created a single point of database initialization for ease of use and start up with backend main.py (this file also creates a test user account with a pre-generated medication to help demo app functionalities).
- Implemented calc_reminders.py that calculates the last day of a medication's supply based on user dosage, supply, and start date/time inputs and calculates the dates to schedule refill reminders based on user preferences.
- Implemented notification functionality for windows (notifications.py) that can also be used for basic kivy notifications after the code base is packaged to be run on an android emulator. The file test_notifications.py serves as a testing environment for these notifications on windows.
- Created login and sign up screen UI and connected backend logic for user authentication and account creation with database (login_screen.py & login_screens.kv).
- Implemented card UI and population of user medications in home screen.
- Implemented traversal between screens from frontend main.
- Implemented prescription_screen and edit_prescription_screen UI and backend integration (displaying stored details of a respective medication and ability to edit variables of a medication and save them back into the specific medication object).