Med Manager 3.0

JIA 342

Members: David Burns, Samuel Gamage, Liam Oweida, Nathan Frazier, Peter Andersson

Client: Hampton Wingate, VivaLife

Table of Contents

0. List of Figures

1. Introduction

Background

Forgetting to take critical medications can lead to dangerous consequences. People with serious conditions may need to take upwards of 10 medications daily. Although physical pill bottle organizers help a patient separate their medications, they do not provide reminders or keep track of how often a patient forgets to take their medication. Pill dispensers provide an expensive solution to this important problem, but they can be difficult to use and maintain for an elderly individual.

Med Manager 3.0 is a cross-platform mobile application that helps people keep track of their prescriptions. It is designed to be as intuitive as possible, so it can easily be used by all ages. Our client, founder and CEO of Viva Life, has already created a physical Med Manager kit, and has turned to Georgia Tech's Junior Design program to create a mobile app to go along with this kit. Although interacting with the physical Med Manager kit is out of scope for our semester's team, we are implementing essential features that will be the backbone of the Med Manager Application.

Our main user features this semester are:

- registering/logging in a user
- adding/editing a medication
- sending confirmation notifications
- viewing a calendar

Document Summary

Our specific user stories for each of these features are divided into five sprints that are outlined in our client charter. The contents of this report will showcase our System Architecture, Data Storage Design, Component Detailed Design, and UI Design for these main use cases.

The System Architecture section outlines the frameworks and important architecture decisions that make up the core components of our project. We show the static and dynamic parts of the project through various UML diagrams.

The Data Storage Design section describes the thought process of the backend portion of our project, in which we use NodeJS and MongoDB to easily manipulate the data objects of our users.

The Component Design Detail describes the structural components of the application by showing the relationships between our objects and classes.

The UI Design section presents the four major user interfaces and our reasoning for our design choices.

2. System Architecture

Introduction

In this section, we outline the system architecture for Med Manager 3.0. To do so, we created two diagrams with the first being our app's Static System Architecture Diagram (Figure 2.1) and the second being our System Sequence Diagram (Figure 2.2); these diagrams together function to enhance the overall understanding of our app from an architectural standpoint. The former diagram includes the broader, static elements of our application and the high-level interactions between them, while the latter displays the dynamic elements/components of our app and the more detailed interactions between them.

Static System Architecture

The static architecture diagrammed below provides a functional view of our system, detailing three main layers: the Interface layer, the Application Logic layer, and the Data layer. Within each layer are sub-blocks for the main logical components and processes within our system as well as the interactions between them.

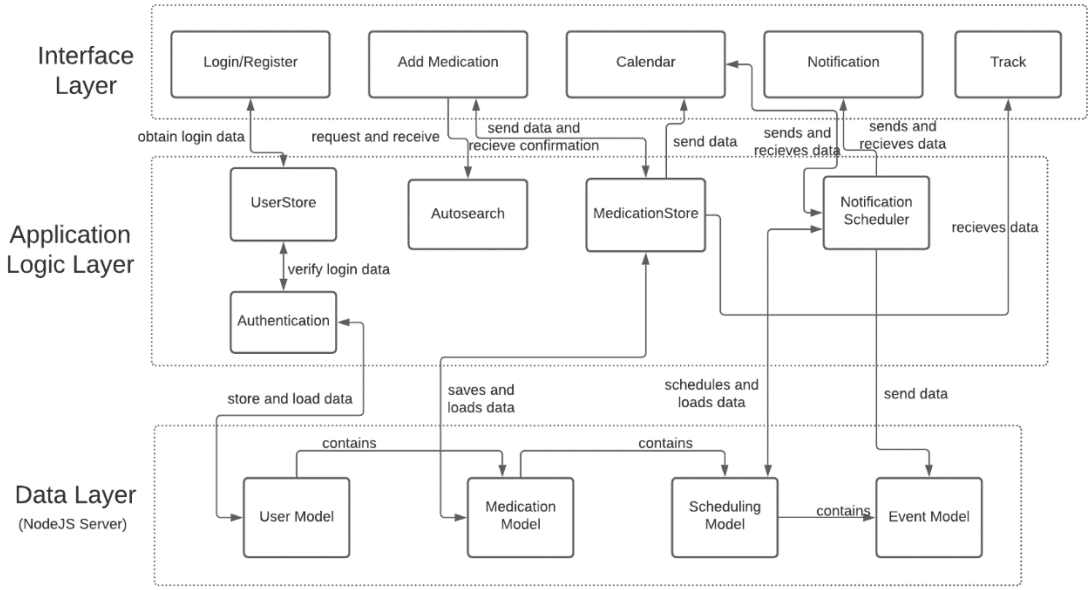

Figure 2.1 Static System Architecture Diagram

Interface Layer

The interface layer details the main ways the user interacts with the system. Though each interface is contained within the app, the user interacts with different parts of the system based on the interface they are using; the different interactions are detailed in the relationships between the interface layer and the application logic and data layers. Our application contains four main interfaces: Login/Registration, Add Medication, Calendar, Notifications, and Track.

The Login/Registration interface is responsible for establishing a secure user and storing the user's information in our database. Here, a user chooses an email and password to link to their account and provides their full name when registering. The interface alerts a user when certain credentials are required when registering, and alerts a user when their login credentials are invalid while logging into the app.

The Add Medication interface is accessed when a user wants to add a new medication to the app. Here, medication information is attached to the user's profile and stored within the app.

The Calendar interface contains a scheduled view of when a user is supposed to take their medications. The medication scheduling is done within the Add Medication interface but displayed in the Calendar interface.

The Notification interface exists both inside and outside of the application; the user receives notifications on their native phone OS as well as within the app itself, where notifications are viewed and interacted with (by a user confirming/denying if they have taken the medication as scheduled).

The Track interface is accessed via the tab bar and is an interface used to view your medications' tracking/compliance history.

Application Logic Layer

The application logic layer is where the system handles user actions, decisions, and any changes to the system. It functions as the bridge between the interface and data layers, processing different logic based off user interaction at the interface layer and storing any modifications in the data layer. The four main sub-levels of the application logic layer are the User Store, Authentication, Auto-search, Medication Store, and Notification Scheduler.

The User Store obtains the user's login/registration information from the login/registration interface layer. It also sends this data to be verified by the Authentication. Authentication handles authenticating a user's login/registration information which is stored in the User Model.

The Auto-search requests and receives information from the Add Medication interface. Auto-search processes the user inputted string from the Add Medication interface to be compared to common medication names acquired from a drug API (RxNav) of common medication names and dynamically displays the suggested medication names in the Add Medication interface based on the string the user has currently entered.

The Medication Store receives data from the Add Medication interface and processed the information to confirm if a medication was saved properly. It sends feedback to the Add Medication interface to denote which information is required when adding a medication. Also, the Track interface receives data from the Medication Store to correctly display the user's medication compliance for a certain drug (that is, the percentage of doses they took out of the total number of scheduled doses).

The Notification Scheduler handles processing the user's notifications settings and updating when a user has modified a notification (by confirming if they have taken the medication as scheduled).

Notification Scheduler also handles how the notifications are displayed in the Notification interface and when they are sent by receiving information from the Notification Model at the data layer. The notification scheduler uses the scheduled times for the medication and creates jobs for the program to send a notification through the respective device's notification scheme. iPhone users will have notifications confirmed through Apple Push Notifications (APN), and Android users will have notifications confirmed through Firebase; both systems require a development key from the appropriate company.

Data Layer

The data layer outlines all the data that is stored and moved within the system and its relationships to the application logic layer in terms of which processes need which specific kinds of data. The main sub-levels here are User Model, Medication Model, Scheduling Model, and Event Model.

The User Model contains all the user's credentials (name, email, password) as well as the information in the Medication Model. The Medication Model contains all the user's medication data (each medication they have entered in the app including the name, dosage, amount, notes, and reason for taking the medication) which also contains the Scheduling Model. The Scheduling Model contains all the scheduling information for a medication (when to take it, how frequently to take it, and what dosage to take at a given time). Finally, the Scheduling Model contains the Event Model which details when notifications should be sent to the user based on the schedule data defined in the Scheduling Model as well as which dosages should be displayed on the calendar for a given day with the proper time and time-of-day grouping (morning, afternoon, evening, or night).

Dynamic System Architecture

The dynamic architecture design (Figure 2.2) displays the flow within the app and the interactions between components of the system as information is exchanged amongst the three main layers and the user. The use case outlined in our SSD features a user opening the app, logging in, adding a medication, scheduling a dosage event, accessing the calendar, and closing the app. The SSD displays the specific information exchanges and operations included in the use case (e.g. checking the token and authenticating log in information as well as verification of the user data and displaying an error if necessary).

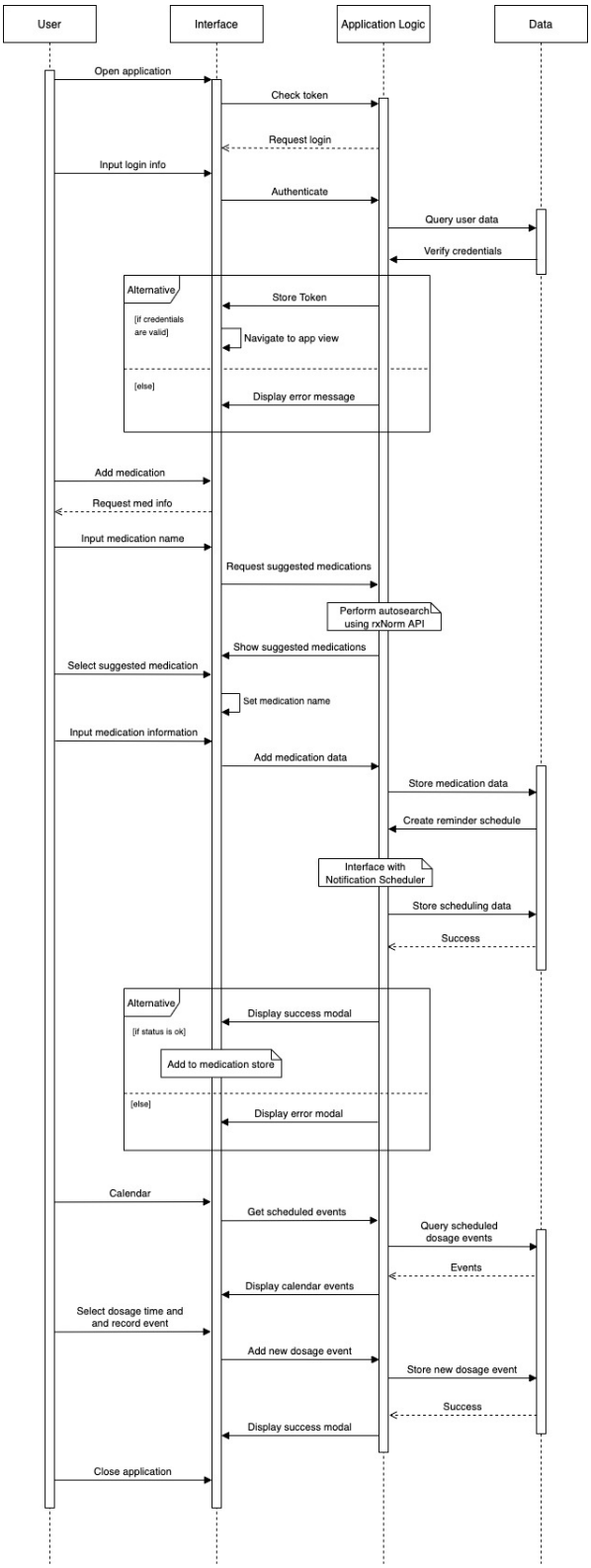

Figure 2.2 System Sequence Diagram

3. Data Storage Design

Introduction

Within this section, we define and showcase the functionality of the database used in the Med Manager 3.0 application. We provide information detailing the choice of database for our application as well as the underlying structure of the data within it by using diagrams and detailed explanations.

Database Description

The database that is employed within the Med Manager 3.0 application utilizes MongoDB, a non-relational document-based database. A diagram of the database shown below (Figure 3.1) depicts the different levels of the database and how the data fields within them are structured.

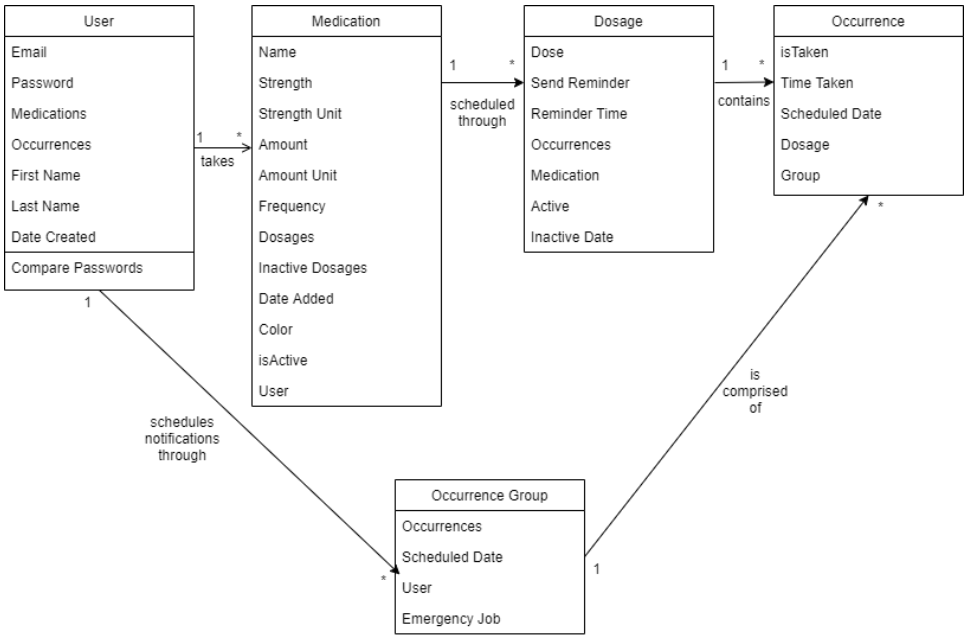

Figure 3.1 Database Diagram

Our database is composed of five different types of objects: Users, Medications, Dosages, Occurrences, and Occurrence Groups. Keeping track of Users to allow a simple registration and login service, while also keeping track of user-specific data, such as the medications they are taking. Med Manager 3.0's main functionality runs based on Medications, which store when a User is taking a particular brand, dosage, and schedule of medication. This allows for coherence tracking to the Medication's predetermined schedule and opens the possibility of reminders for the User to take the medication in their given time window. The Occurrence object represents one of these times, when the User is expected to take their Medication. A user's list of occurrences is ultimately used to create a report on coherence for a given User's Medication. To prevent notification fatigue, we created our final object, Occurrence Groups. This object is a grouping of different medication occurrences scheduled to occur around the same time and is attached to the user.

While our choice of database is sufficient, future teams working on this application should consider switching to a relational database, as a relational database would ultimately be a better option for our data based on our design. This is because the five types of data stored in the database, Users, Medications, Dosages, Occurrences, and Occurrence Groups, have relations between each other. These include Medications being stored within a User, and Occurrences being stored within a Medication. Using a relational database would allow for easier flow throughout the database and would require less backend implementation to work around the limits of a non-relational database.

4. Component Detailed Design

Introduction
This section offers a detailed look into the Component level of our system. We showcase both the static and dynamic details of the system through Figures 4.1 and 4.2, our Static Component Diagram and Dynamic Component Diagram, respectively. These diagrams offer a more in depth look at the system architecture presented in section 2 and expand on the many interactions between the different components of our system.

Static Elements
Figure 4.1 is our Static Component Diagram and displays the low-level components of our system and the different interactions between them. Each box represents a component and the arrows between them showcase the interactions amongst these components. Each component featured here is an interface that is reachable by the user themselves through navigation and viewing, and also details the kinds of operations made between certain components based on user input (e.g. updating profile information).

One specific route of navigation to follow as a case study is that of opening the app, seeing the welcome view, logging in, having your log in credentials authenticated, navigating to the calendar, viewing a dosage's details, navigating to the edit dosage interface, editing the dosage, and finally updating the tracking history of the medication associated with that dosage. The numerous other routes like this example offer a comprehensive depiction of all the static components of our system and the different operations that occur along the connections between the components.

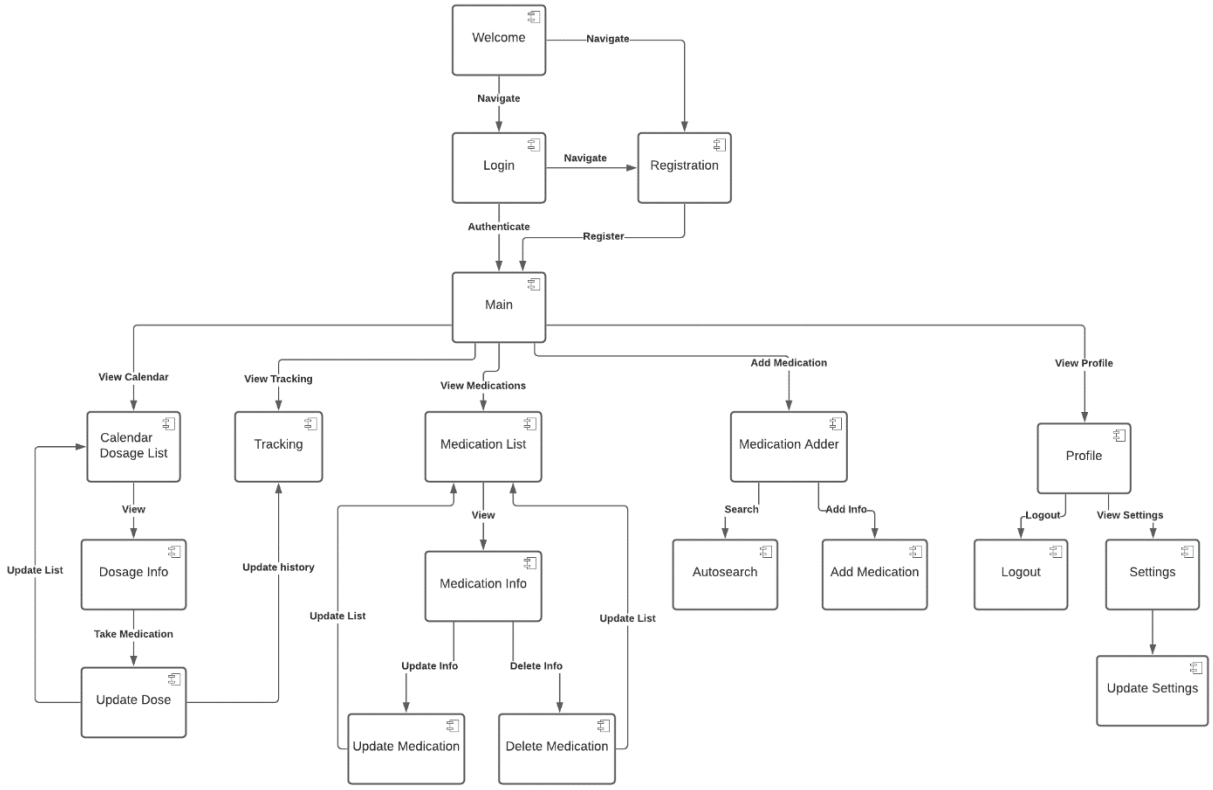

Figure 4.1 Static Component Diagram

Dynamic Elements

Figure 4.2 shows the Dynamic Component Diagram of our system. The static and dynamic diagrams are highly similar, but the key difference between the two is that the dynamic diagram displays the run-time behavior of our system and the dynamic interactions between the components (with special interest to user interaction and control flow). Again, each component is represented by a box on the diagram, however the diamonds represent conditional branches or user decisions, and the arrows showcase interactions between the component boxes.

To see the diagram in action, consider the use case of starting the application and arriving at the welcome screen, navigating to the login screen (by clicking the login button), entering incorrect credentials, having these credentials authenticated, failing the authentication (meaning incorrect credentials), being re-navigated to the login screen, entering the correct credentials, authenticating these credentials again, passing the authentication, then being directed through main and selecting to view your medication list (through a user selection of the medication list tab). Overall, these conditionals and places for user input to be evaluated (the diamonds) offer a more detailed look into the nature of our system by displaying the finer, dynamic qualities of the components in our system.

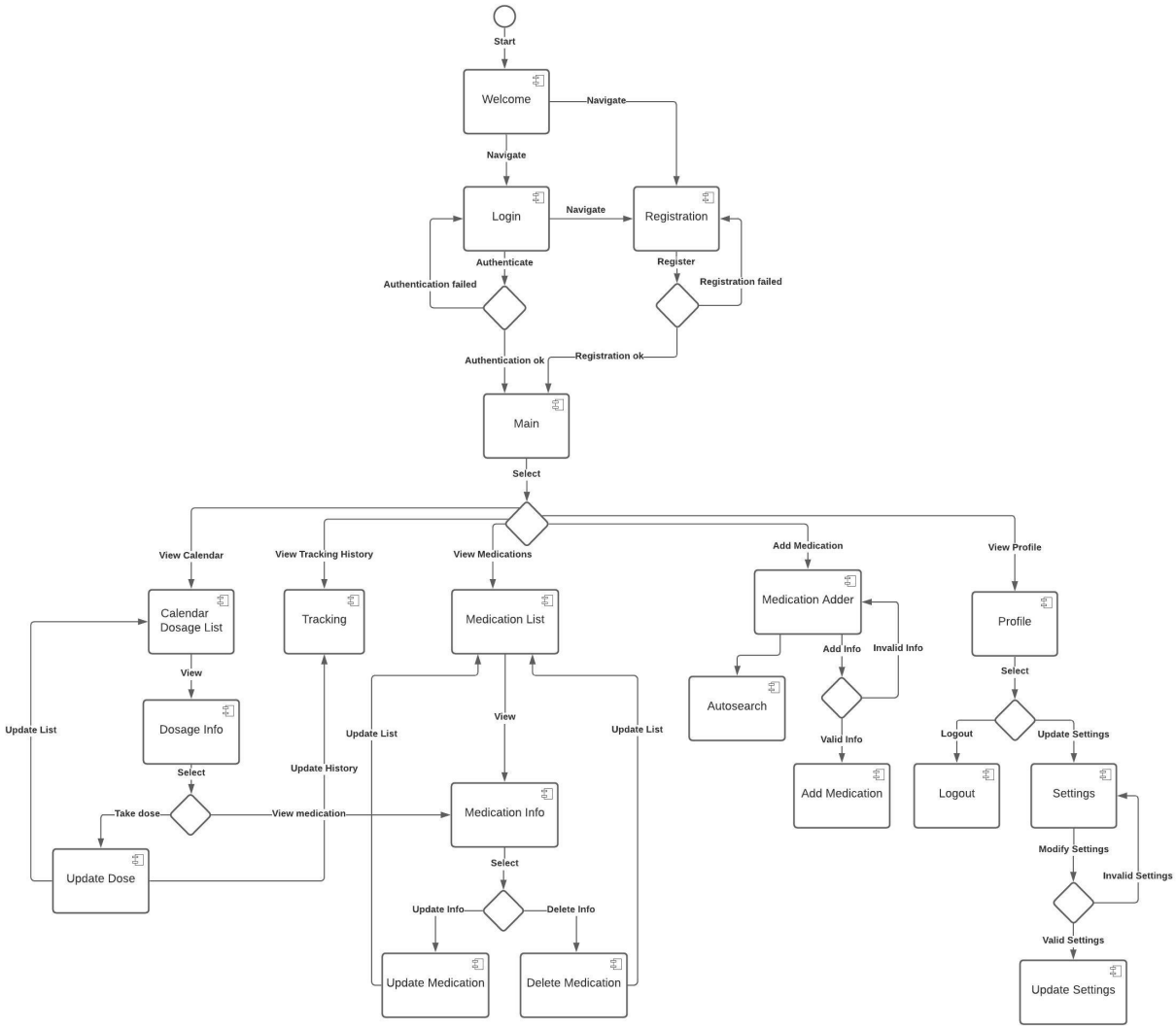

Figure 4.2 Dynamic Component Diagram

5. UI Design

This section contains images of the major screens of our app and functions as a walkthrough of the Med Manager 3.0 UI. Each screen will be presented both in terms of functionality and design coupled with explanations for the decisions made for each during the UI design process.

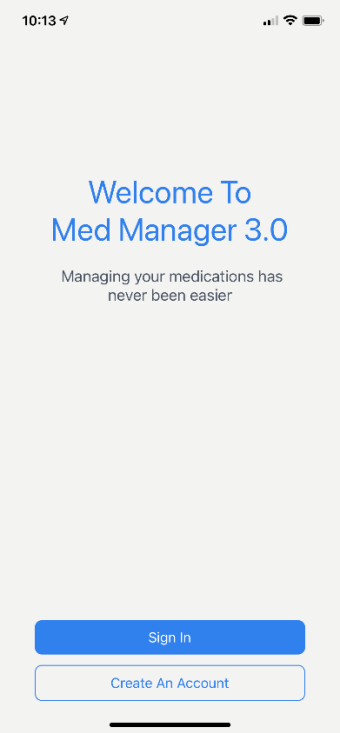

Figure 5.1 Welcome Screen

The first screen (Figure 5.1) the user sees once the app is loaded and running is the Welcome screen. Just as the name suggests, this screen functions to welcome the user to the app, give a brief explanation of the app's purpose, and finally offers the user two options to continue: login or sign up. The primary design goal for this screen was simplicity and cleanliness; we did not want to overwhelm the user on the very first screen, so we chose a simple, minimal, horizontally centered look. The login and sign-up buttons are flipped in color (blue text and white background vs white text and blue background) to create a clear distinction between the two. It's worth noting here too that blue is our app's primary color throughout the UI; we feel this shade of blue is very light and welcoming.

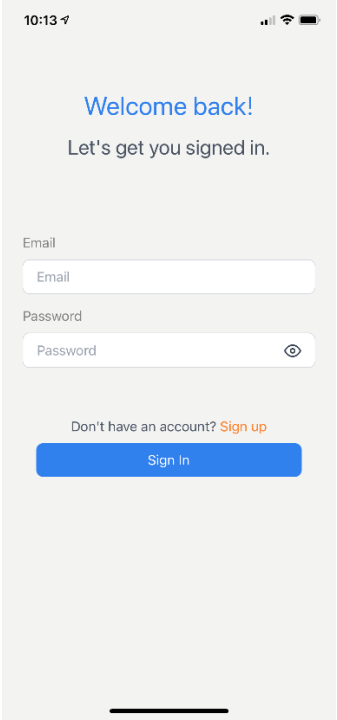

Figure 5.2 Sign In Screen

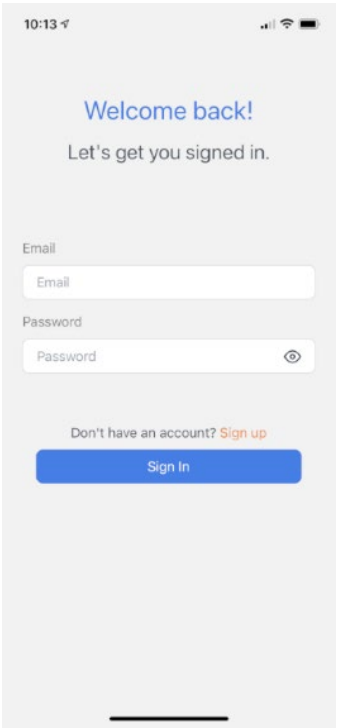

Figure 5.3 Sign Up Screen

The next two screens are of course the Sign In and Sign Up screens (Figures 5.2 and 5.3 respectively). They are very similar in that the Sign In screen is a simplified version of the Sign Up screen with less required fields (since the user has already registered if they are now logging in). For the Sign Up screen, the user is required to input their first and last name, their email, and their password; the two fields that carry over to the Sign In screen are just the email and password. Each screen features a similar string of text towards the bottom, "already have an account/don't have an account? sign in/sign up". The sign in and sign up parts of this string are clickable and are made to stand out by being in our main secondary color, orange; we felt orange was a great choice for our secondary color since it stands out well against blue, but doesn't do so as harshly as red for example. The action button at the bottom of the screen again features our primary blue color with white text to couple it.

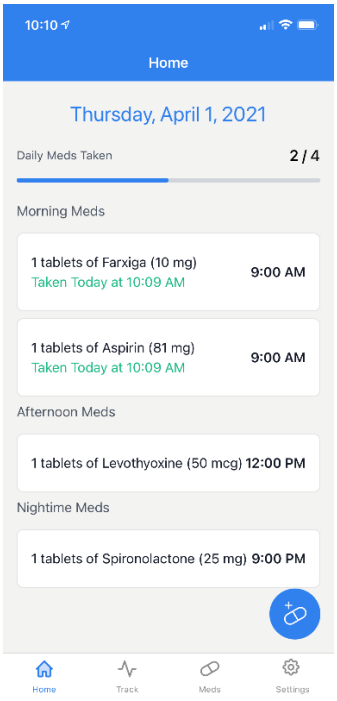

Figure 5.4 Home Screen

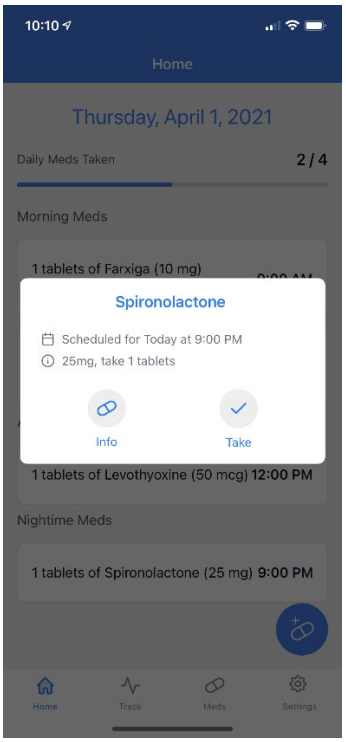

Figure 5.5 Dosage Modal

After completing the Sign In/Sign Up process, the user is brought to the Home screen of the app (Figure 5.4), which is the functionally the calendar screen. This screen displays all the scheduled dosages to be taken on the current day. These dosages are divided into four-time blocks: morning, afternoon, evening, and night (each defined the same as they are in the Add Medication screens which will be detailed in a later section). Within these groups are the specific dosages that are scheduled within these time blocks; they are represented in rounded rectangles that display how many doses to take, the type of dose (tablet, capsule, spray, etc.), the medication name, the medication strength, and the specific time the dosage is scheduled for. The information here is compact and straightforward as a design which was the main goal for representing these dosage events succinctly and effectively. Upon clicking on a dosage, a modal pops up with the option to "take" a medication or see more information about the medication. This modal also displays the scheduled time, dosage, and amounts, all underneath the name of the medication (Figure 5.5 Dosage Modal). The "info" button takes the user to the Medication Detail View which is detailed in a later section. Once a medication has been "taken" through this modal by pressing the take button, the corresponding dosage on the Calendar Screen is greyed out, to reflect that this dosage has been taken and the user should not take it again; to further reinforce this, if the user were to click on the greyed out medication, the same modal pops up, but instead of having a "take" option, it reads as "taken" and shows the time the user took the scheduled dosage.

Beneath the Home screen is a tab bar that is consistent throughout the four main screens of the app which are detailed in the tab bar itself: Home, Track, Meds, and Settings. This tab bar remains present and accessible across these four screens to make for quick and easy navigation between them. Furthermore, to reinforce to the user which screen they are currently on (aside from the header titles

on each screen), the selected tab bar icon turns blue to indicate it has been selected and you are on that selected screen. From a design perspective, we opted to combine the icons here (and throughout the rest of the app) with associated text labels to reinforce the icon's meaning and make everything straightforward and understandable for the user. Making the app simple and understandable from a UI perspective was a great emphasis in our design as our app will likely have a majority client base of elderly individuals who may need to manage many medications.

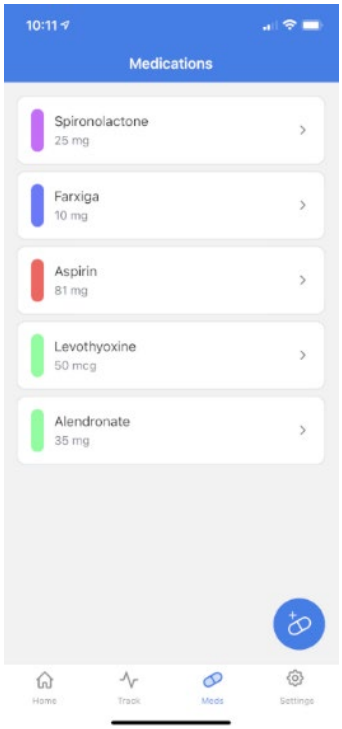

Figure 5.6 Medications Screen

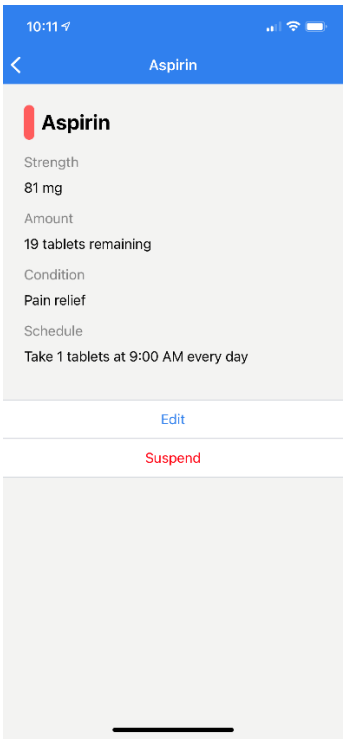

Figure 5.7 Medication Detail View

Next, we showcase the Medications screen (Figure 5.6). This screen functions as a list of all the medications the user has entered into the app. It displays the medication names and dosages alongside their associated color that the user has chosen in the Add Medication process. The same rounded rectangles that are used for the dosages in the Home screen are used again here, and on the right side of each medication is an arrow that prompts the user to click on the medication. Upon clicking, the user is shown the Medication Detail View (Figure 5.7). This screen shows every detail that the user has entered about the medication during the add medication process, along with two more action buttons: "suspend" and "edit". Clicking the "suspend" button stops the scheduling of all future dosages for a certain medication. Clicking the "edit" button brings the user to the Add Medication Screens with all the current info about the medication already entered and remembered in the app, and the user is free to edit these existing fields as they wish.

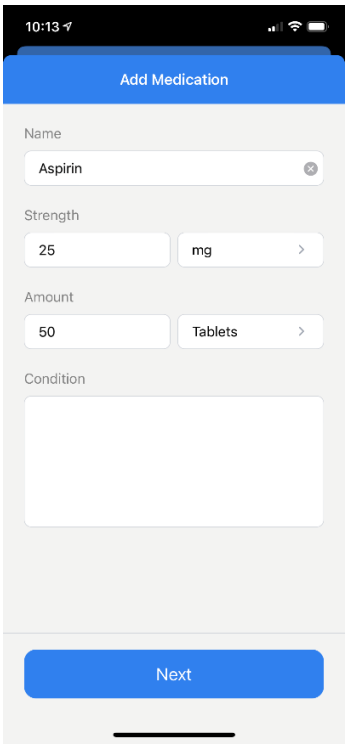

Figure 5.8 Add Medication 1

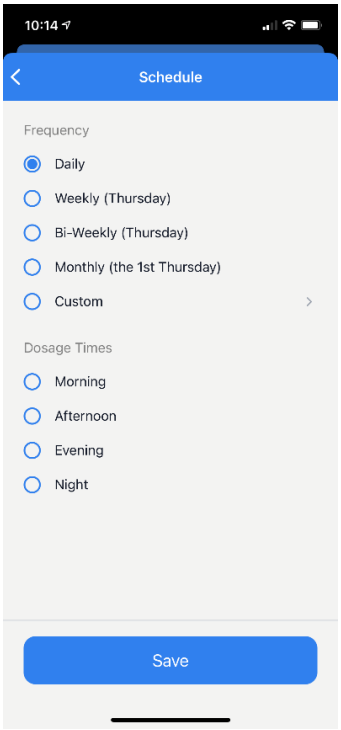

Figure 5.9 Add Medication 2

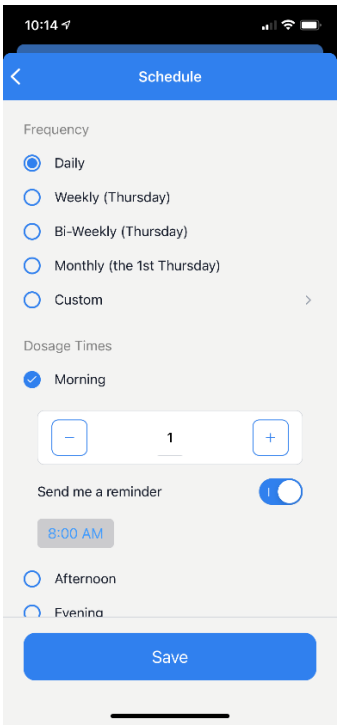

Figure 5.10 Add Medication 3

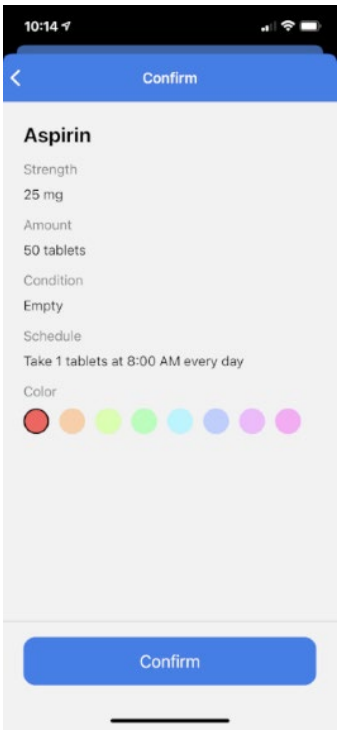

Figure 5.11 Confirmation Screen

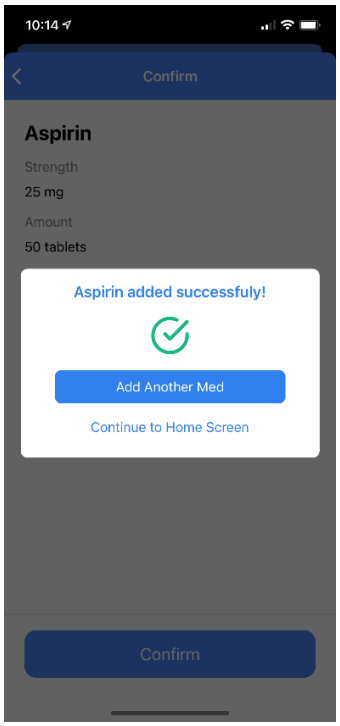

Figure 5.12 Success Modal

Next, we highlight the Add Medication screens (Figures 5.8, 5.9, 5.10, 5.11, and 5.12) and process. To access these screens, the user clicks the floating add medication button that is found on the four main screens throughout the app (defined in the tab bar) and is depicted as a blue button with a pill icon in the center and a plus sign on the top left. The Add Medication Screens are a series of modals that occupy nearly the entire screen, with the first being where the user enters the name of the medication, the dosage, the units of the dosage, the amount, the units of the amount, and the condition/reason for which they are taking the medication (Figure 5.8). The name is inputted as a typed string, but a dropdown of five suggested medication names appears based on the current entered string of text, and the user can choose to click on one of these names to autofill the medication name, so they don't have to type the entire name out. Both the dosage and amount are also text inputs (restricted to numerical values only), and for each of their corresponding units a drop-down menu appears with an exhaustive list of possible units for the user to click and choose. Finally, the condition field is a simple text input.

Upon clicking "next", the user is brought to the Frequency section of Add Medication (Figure 5.9). Here the user chooses between a few predefined frequencies or the custom option by clicking which frequency they wish to have (and once one is clicked, the circle to the left of its label fills in with blue). The default value here is "daily" and is preselected to speed up the Add Medication process as it is the most common frequency for medications to be taken. To select a specific time to take their medication, the user then selects a time block (morning, afternoon, evening, or night) and an amount input, reminder toggle, and time picker appear for the user to customize (Figure 5.10). The user can select as many time blocks as they wish for a medication to account for cases when a medication needs to be taken more than once a day. The amount input can be incremented up or down by

clicking the plus or minus to the right and left of the amount or can be entered as any numerical value (including decimals for cases of half dosages).

Once the user clicks "save" they are brought to the Confirmation screen (Figure 5.11). This screen functions as a summary of all the information the user has inputted during the add medication process and allows the user to select a color for their medication. The user can review their information to make sure it is correct before adding it to their list of medications. Upon clicking "confirm", a modal pops up indicating whether the medication was added correctly (Figure 5.12 depicts a successful addition) and gives the user the option to add another medication or continue to the Home screen.

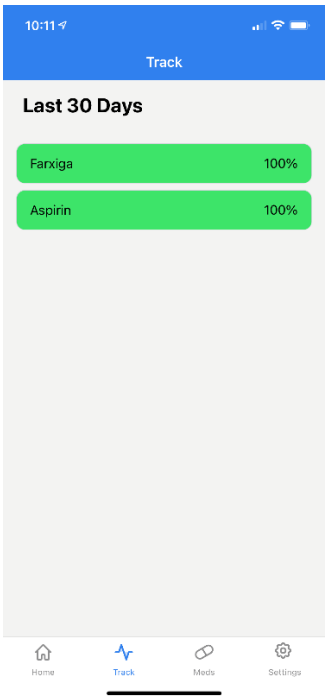

Figure 5.13 Track Screen

Next, we have the Track screen (Figure 5.13). The Track screen's purpose is to display the compliance percentage for each of the user's medications over the last 30 days, or in other words, the percentage of how many times a medication was taken vs not taken. If the compliance percentage is over 80% the bar's fill is green, and the bar's length is directly tied to the percentage (e.g. if the compliance rate was 85% the bar would occupy 85% of the rounded rectangle and would have a green fill). Compliance percentages between 60-79% are displayed in yellow, and percentages of 59 and lower are displayed in red. Each medication's tracking data contains the name of the medication and the exact compliance percentage (along with the visual bar representation). This screen's ultimate purpose is to be a quick look at a user's compliance history to easily see which medications they meet the mark for and which medications they need to work on taking more regularly. The colored percentage bar functions as the quick and easy heuristic to distinguish between these

categories of compliance while the actual percentage can be used to understand exactly how compliant a user has been with a certain medication.

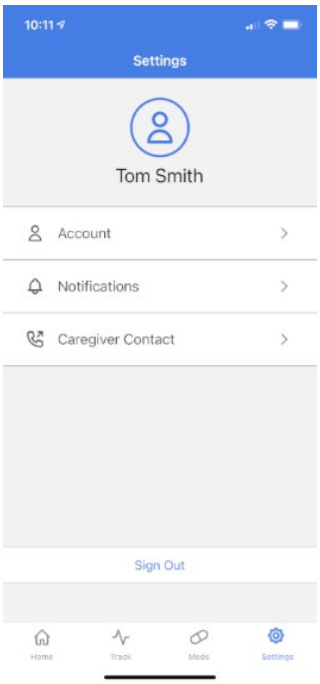

Figure 5.14 Settings Screen

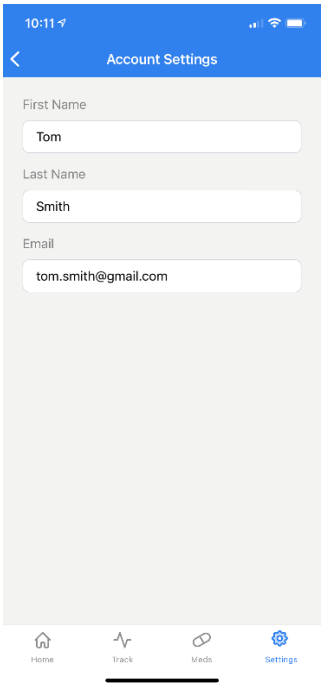

Figure 5.15 Account Settings

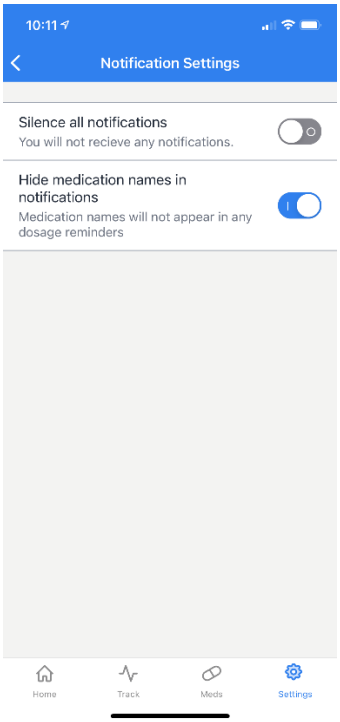

Figure 5.16 Notification Settings

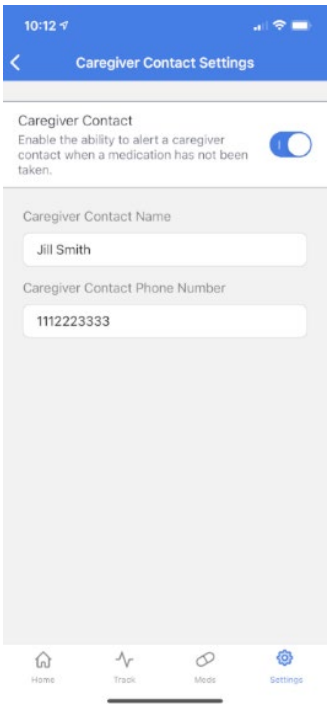

Figure 5.17 Caregiver Contact Settings

Finally, we have the Settings screens (Figures 5.14, 5.15, 5.16, and 5.17). The main Settings screen (Figure 5.14) displays the user's name underneath a generic profile icon; below this are labels for grouped settings, namely Account, Notification, and Caregiver Contact Settings with appropriate icons next to each label. These labels are reinforced as clickable by virtue of the arrows on the right of each block. Upon clicking the Account label, the user arrives at the Account Settings (Figure 5.15). This screen allows the user to view and edit their current account information such as first name, last name, and email. Clicking the Notifications label brings the user to the Notification Settings screen (Figure 5.16). Here, the user is presented with two toggles, one for silencing all notifications, and one for hiding the names of their medications in the notifications they receive (which defaults to "on" to protect user privacy). Finally, clicking the Caregiver Contact label brings the user to the Caregiver Contact Settings screen (Figure 5.17). Here, the user is presented with another toggle to enable the ability to alert a caregiver contact if the user has not taken a medication that was scheduled for that day. If this toggle is "on", the fields for Caregiver Contact Name and Phone Number appear for the user to fill out as a text entry, while un-toggling this setting hides these fields.

Overall, our UI is quite expansive but maintains simplicity and minimalism across all the screens and modals. We recognize that a large portion of our user base will likely be elderly and may not be the most tech-savvy, so understandability and clarity were of utmost importance and occupied our focus during our UI design process. The UI is designed to be straightforward and intuitive for this very reason, and we made sure to reinforce all icons with text to eliminate as much confusion as possible while interacting with our app.