Dry Weight Watchers

Team JIF-4300

Duy Nguyen, Nathan Fritchley, Jonathan Hopkins, Lara Bailen

Client: Dr. Martin Johnson

GitHub: https://github.com/DryWeightWatchers/JIF-4300-DryWeightWatchers

Table of Contents

Table of Figures

Terminology

Medical Terms

CHF (Congestive Heart Failure): Heart failure accompanied by swelling due to fluid accumulation in the body.

Dry weight: A person's weight as measured first thing in the morning before consuming any food or drink, used to reduce measurement fluctuations.

Heart Failure: A medical condition where the heart is unable to pump blood efficiently throughout the body.

HIPAA (Health Insurance Portability and Accountability Act): A regulation that healthcare entities must comply with regarding the privacy and security of consumers' personal health information.

Patient: In the context of this document, a person who has been diagnosed with Congestive Heart Failure and has been directed to use our mobile app by a doctor.

Provider: In the context of this document, a healthcare practitioner such as a doctor or nurse, or an administrator working with such practitioners, who uses our system to monitor patients.

Software Terms

API (Application Programming Interface): A set of functionalities provided by a software system for external use, along with a specification of how to use it.

AWS (Amazon Web Services): A suite of tools provided by Amazon for hosting, managing, and deploying computing resources on the cloud.

AWS RDS (AWS Relational Database Service): An Amazon cloud service for hosting, managing, and deploying relational databases.

Back-end: Any part of a software system that is not directly visible to users.

Django: A back-end development framework for creating web servers using the Python programming language.

Expo: A platform for building and deploying React Native applications.

Front-end: Any part of a software system pertaining to the user interface.

MySQL: A widely used platform for creating and managing relational databases.

ORM (Object-Relational Mapper): A type of software tool that translates SQL queries into program objects and vice versa, allowing developers to interact with a database in an abstracted form rather than writing SQL directly.

React: A front-end web development framework for creating user interfaces with the JavaScript programming language.

React Native: A platform that allows developers to write cross-platform mobile applications using an extension of the React framework.

Relational database: A database which stores data in tables whose records can be logically associated with records in other tables.

SQL (Structured Query Language): A language used to design, modify, and access data from relational databases.

SQL Injection: A type of cyberattack that involves injecting malicious SQL code into user input fields, tricking the system into performing unauthorized database operations.

UI: User Interface.

Web interface: A user interface that is accessed via a web browser.

Introduction

Background

We are developing a system that allows healthcare providers to monitor the dry weights of CHF patients. Patients will install a mobile app, generally at the direction of their doctor, and will use it daily to record their dry weight. Providers will be able to monitor patients' dry weights and set up notifications to alert them when a patient's dry weight has changed drastically in a short period, which often indicates worsening CHF symptoms and requires medical attention.

Our mobile app is developed using React Native and Expo and will support both Android and iOS. Providers will use a separate web interface developed using React and hosted with AWS. Data will be handled in a HIPAA-compliant manner, stored in a MySQL database hosted using AWS RDS, and accessed through a Django server.

Document Summary

The ***System Architecture*** section describes the major components of our system and how they interact to achieve the system's goals.

The ***Data Storage Design*** section details the structure of our MySQL database and what data is stored.

The ***Component Design*** section goes into more detail about specific software components of our system and how they are implemented.

The ***UI Design*** section shows the screens that patients and providers will interact with when using our mobile and web interfaces, respectively.

System Architecture

Introduction

Our system's main purpose is to collect and store dry weight data from CHF patients so that healthcare providers can monitor them and identify warning signs in a timely manner. A tiered client-server architecture is a natural choice for two reasons. First, it allows us to separate the user interface from our back-end logic, which is necessary because we need to develop two different UIs for two distinct groups of users – patients and providers. Second, it allows us to isolate our data in a centralized repository, minimizing exposure and thus potential for leaks.

Because our system handles personal health information, we are required by ethics and HIPAA regulations to ensure our data is handled securely. This includes encrypting data both at rest and in transit, ensuring data can only be accessed by authorized users, and implementing security best practices to protect against potential cyberattacks. Our architecture hides our database behind a server used for authentication, enforcing a single point of access. The Django server framework also gives us access to established security libraries that have been heavily audited by the open-source community.

Below is a detailed description of our architecture from a static and dynamic point of view. The static architecture section describes the major components of our system and their basic relationships to each other, using a UML Component Diagram for illustration. The dynamic architecture section describes how the components interact at runtime. There we combine two System Sequence Diagrams into one figure to give an integrated picture of how the system handles representative interactions from both patients and providers.

Static Architecture

Our system consists of two front-end interfaces, a server, and a database. All access to the database is mediated by the server, which exposes a Data API to meet the needs of front-end clients. Requests to the Data API are subject to authentication to ensure each client can only access data they have been explicitly authorized to access. The server, written in Django, also has an ORM that abstracts the SQL API provided by the database. The ORM functionality simplifies data handling for developers and protects against SQL injection attacks by automatically sanitizing input queries.

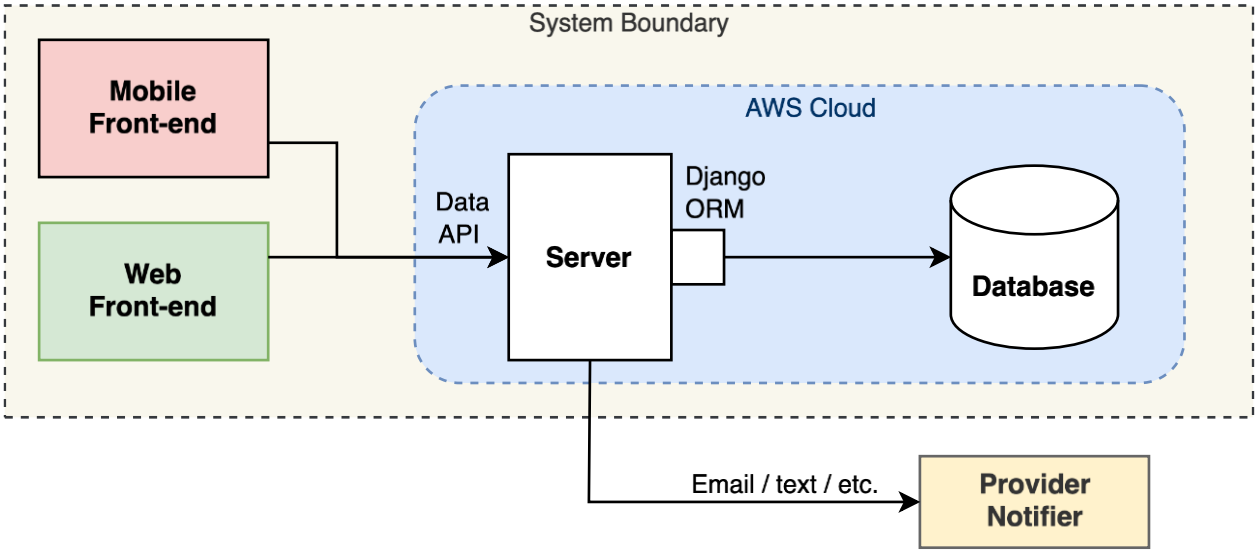

Figure 1: *Static System Diagram – Component Diagram*

Aside from mediating access to data, the server must also notify providers if any of their patients are showing alarming symptoms. The *Provider Notifier* component represents an external service, such as email or text, that the provider has registered with our system as a channel to receive notifications.

Dynamic Architecture

The System Sequence Diagrams (SSDs) below show the flow of interactions between the major system components. The top diagram shows what happens when a patient records their weight, which is the primary way in which the patient interacts with the system. The bottom diagram, separated by a break in the component lifelines, shows what happens when a provider views details about a particular patient. This is a common use case that the provider is likely to initiate after being notified of an anomalous dry weight record as shown in the top diagram.

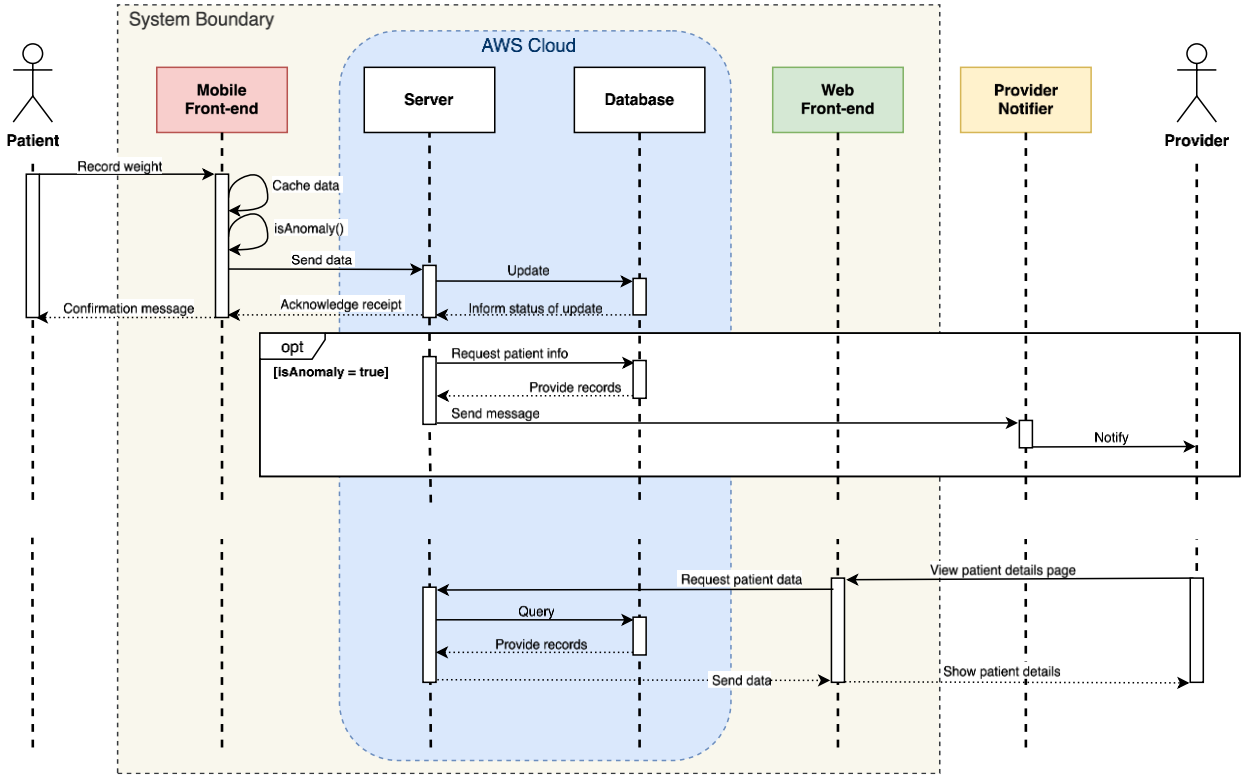

Figure 2: *Dynamic System Diagram – System Sequence Diagrams*

When the patient records their dry weight in the mobile app, the app checks to see if the weight represents a potential medical concern. Weights are considered anomalous based on how drastically they have increased in a short time period, so the mobile app needs to keep track of previously recorded weights locally. It therefore caches these so that it doesn't have to fetch them from the server each time a new weight is recorded. The app then sends the new record to the server, which passes it along to the database. Meanwhile, the server confirms receipt of the data so the app can show the patient a confirmation screen.

If the recorded weight was deemed anomalous by the mobile app, that information is also passed to the server, which alerts the provider through the *Provider Notifier*. In order to compile a useful message for the notifier, the server must first retrieve the relevant information from the database. The exact

information required is an implementation detail, but could include the patient's name, medications, and/or contact information, for example.

Note that the patient is not alerted if an anomalous dry weight is recorded. This is because (1) the patient is generally already aware of what is considered a potential concern via talking with their doctor, and (2) notifying the patient may cause stress that inhibits them from diligently recording their weight or encourages them to fudge their data. Our system places responsibility solely on providers for identifying and responding to anomalies; the patient is only responsible for recording their dry weight each day.

After a provider is notified, they may want to use the web interface to view more detailed information about a patient. When they access a patient's page, the web interface submits a request to the server to retrieve the data for that patient. The server handles querying the data from the database, then sends it back to the web front-end client.

Data Storage Design

Introduction

All of our application's data is stored in a MySQL database which is hosted using AWS RDS. The bulk of the stored data consists of daily dry weight measurements recorded by patients. Healthcare providers may also enter information about patients into the PatientInfo and PatientNote tables, and patients may also set up reminders which are stored in the Reminder table.

Patient and provider accounts are stored in the User table and distinguished by the *role* attribute. A many-to-many relationship table keeps track of which patient accounts are connected to which provider accounts. In order for two accounts to be connected, the patient must enter the provider's *shareable_id* into the app – this attribute in the User table is *null* for patients and has a unique value for each provider.

Database Use

All attributes named *patient_id* or *provider_id* are foreign keys pointing to the *id* attribute in the User table. Our actual database contains some tables not shown here that are used internally by Django, but these are the ones with relevant functionality for our application:

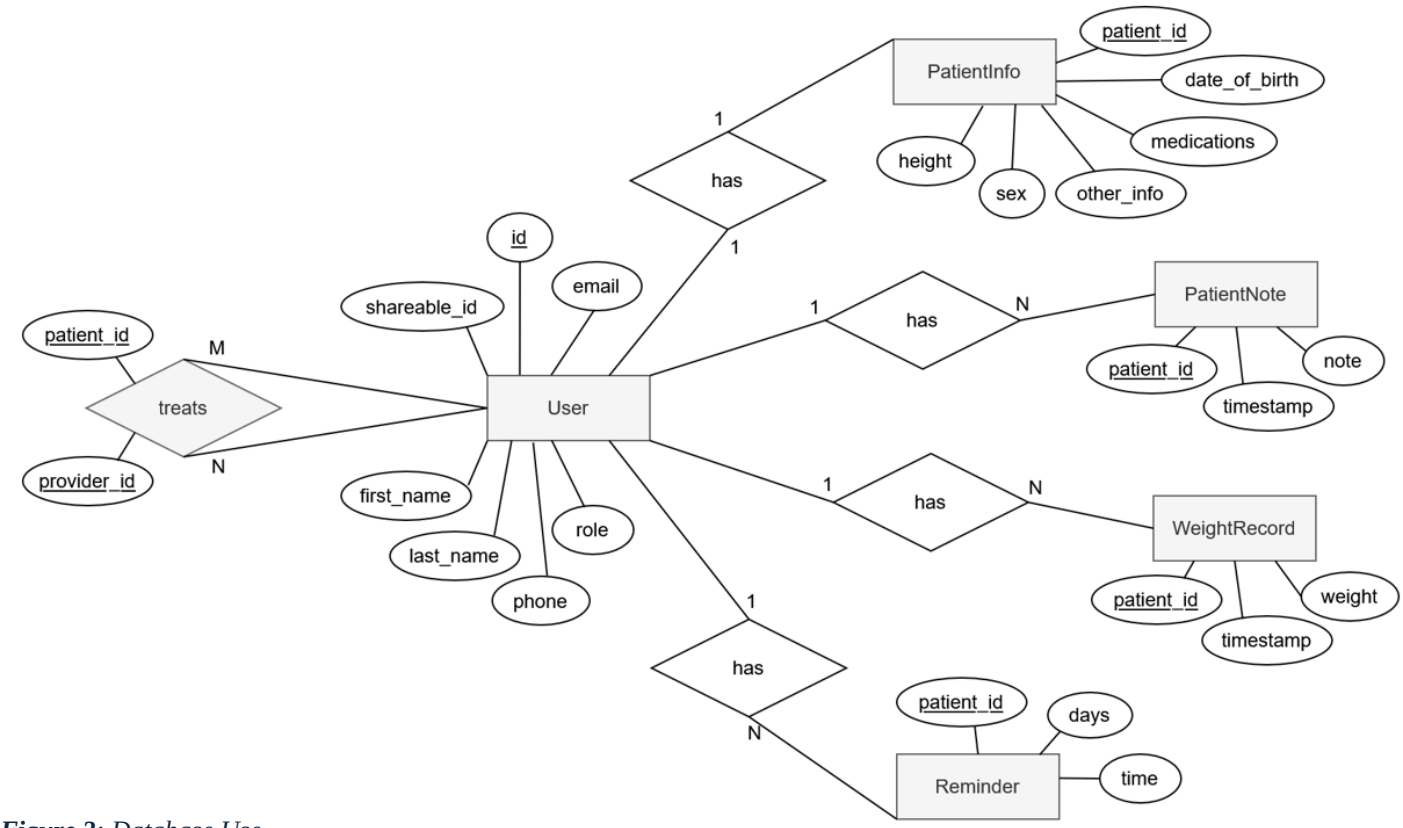

***Figure 3**: Database Use*

File Use

Our application will give healthcare providers the ability to export a patient's weight record in CSV format in case they need to port the data to another service. This is a simple text format where each record is contained on its own line in form `<timestamp>, <weight>`. We don't plan to support exporting any other patient information since there are only a few pieces of information that could be easily copied manually.

Aside from that, our application does not support reading or writing any files to the device's file system.

Data Exchange

Data is transferred between the two clients (patient mobile app and provider browser interface) and the server in JSON format.

Because we are dealing with personal health information, we are required to comply with HIPAA regulations regarding privacy and security. Our data is encrypted both in transit and at rest: HTTPS is used for all communication, and AWS RDS allows us to encrypt the database. No sensitive information is stored on the client devices.

Most functionality requires users to log in. Patients and providers each log in via an email and password. Core authentication functionality is built into our Django server framework, minimizing security risks that would be associated with implementing it ourselves. Passwords are stored securely in the database via hashing and salting.

Component Design

Introduction

The static architecture describes the fundamental building blocks of our system, including the front-end interfaces (provider website and patient application), backend logic (Django API), and the data storage layer (MySQL database). These components are represented in a UML Component Diagram (Figure 3), which highlights how they interact through API calls and database queries. Each component is designed to function independently while ensuring smooth data flow and security compliance.

The dynamic architecture captures how these components interact at runtime to handle user operations. We use UML Sequence Diagrams (Figure 4) to illustrate non-trivial interactions within the system. These diagrams depict the flow of data through the system by modeling API requests, data validation, storage operations, and response handling. Specifically, we examine a key interaction scenario where a patient records their weight via the mobile application, triggering a backend process that stores and validates the data while providing real-time feedback to the user.

Static Elements

The component diagram (Figure 3) illustrates the structural organization of the system, defining the main subsystems and their relationships. The architecture consists of:

1. Provider Website Subsystem
2. Patient Application Subsystem
3. Django Server (Backend API)
4. MySQL Database

The provider website subsystem handles the provider interactions through the web interface, including patient management, profile management, and alerts and notifications about their patients. The patient application subsystem supports patient account management, recording their weight, and setting reminders. The Django server acts as the middleware between the frontend and the database, handling authentication, data processing and API requests. And, finally, the database stores the patient and providers data and supports all key functionalities for the system.

The component diagram demonstrates how each module communicates via API calls, ensuring a well-structured, modular, and scalable system.

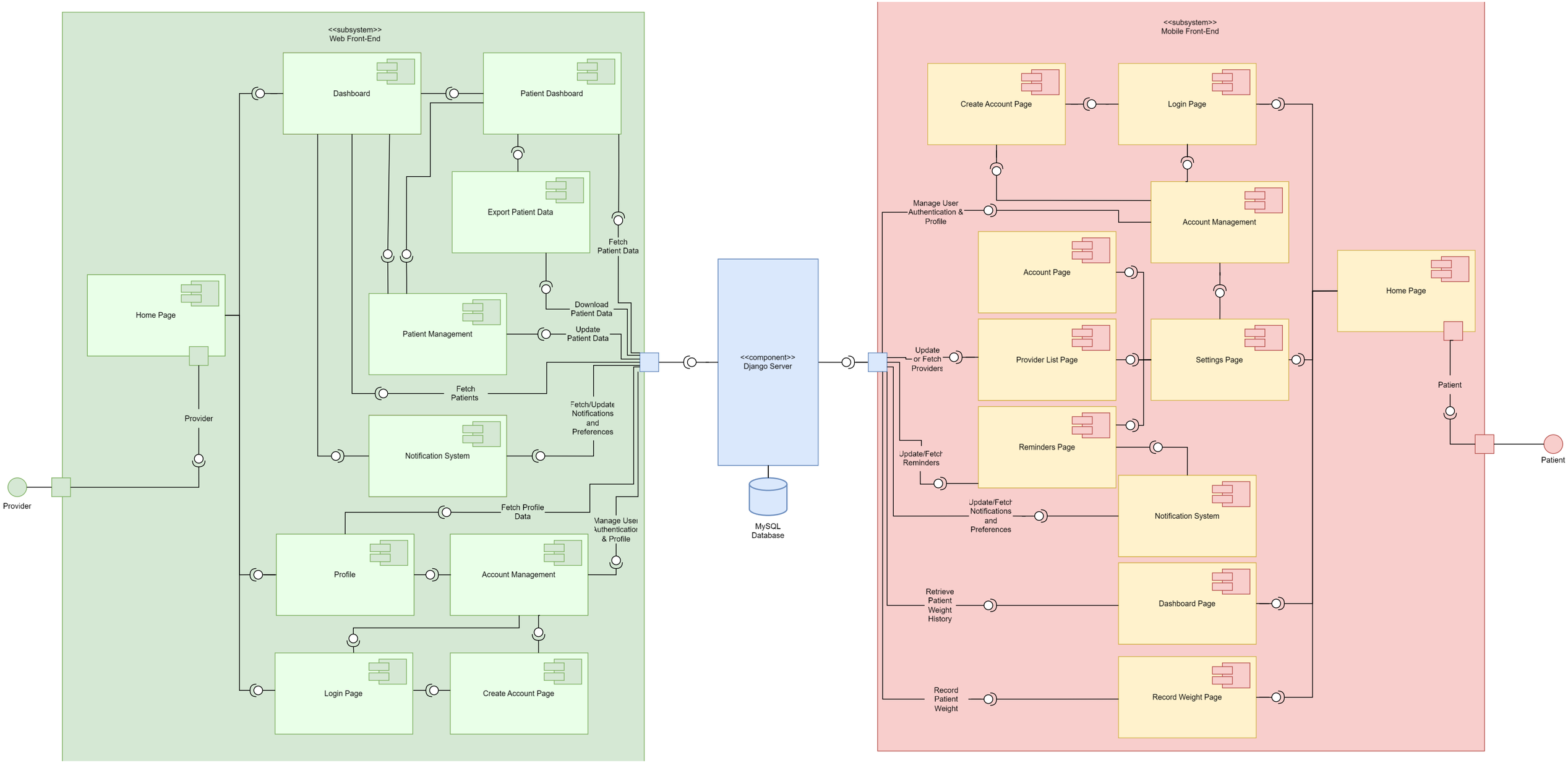

Figure 4: Static Component Diagram

Dynamic Elements

The sequence diagram (Figure 4) provides a detailed representation of runtime interactions focusing on a key system use case, a patient recording their weight.

The steps included in this use case are as follows: the patient records their weight using the mobile application, the application then sends the data to the Django server via an API call, the server validates and stores the data in the MySQL database, where if the data is flagged as an anomaly, the system sends a notification to their providers, who can retrieve the data through the website.

The sequence diagram highlights the flow of information between system components, ensuring accurate data handling and real-time responsiveness.

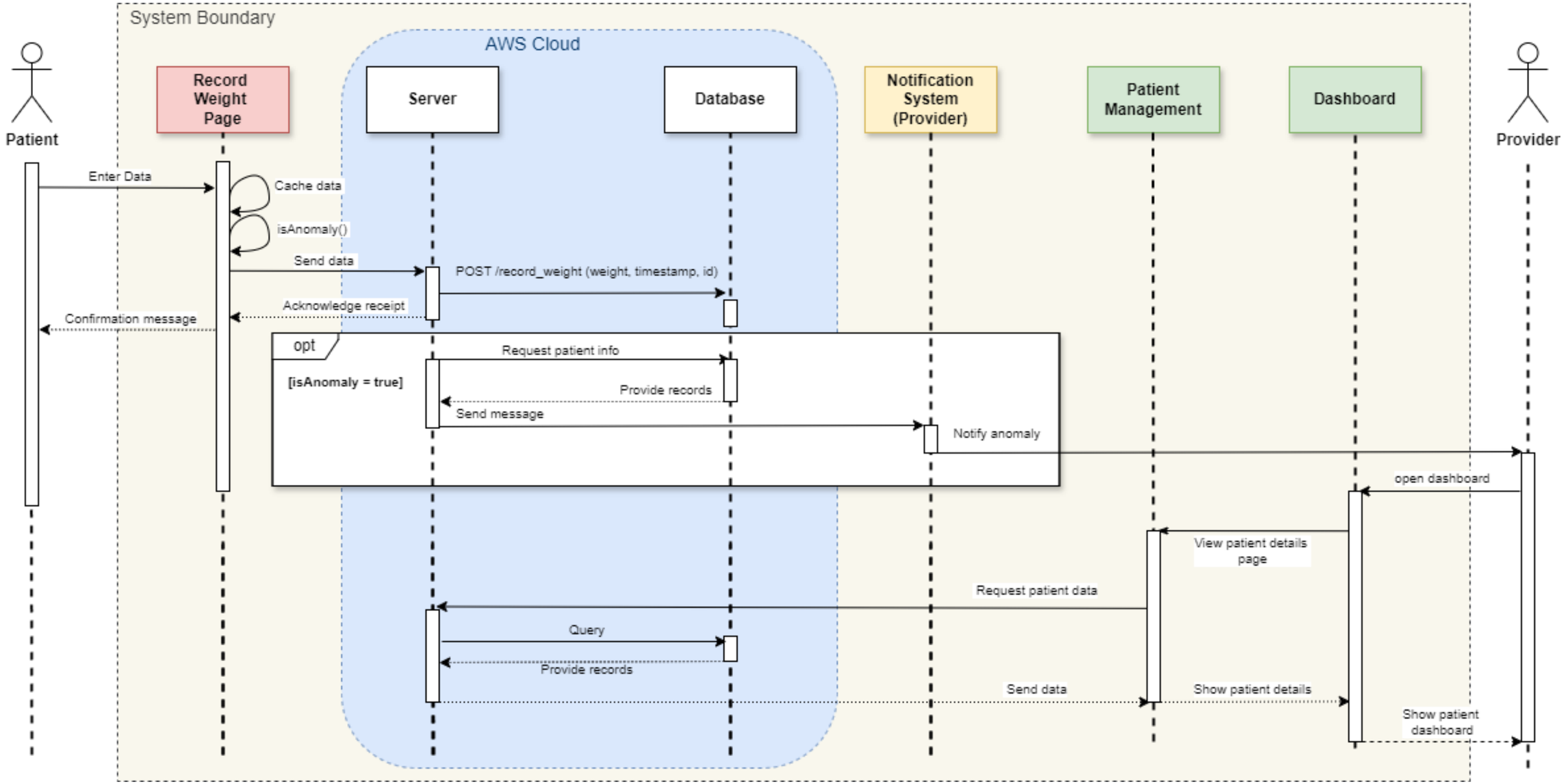

***Figure 5**: Sequence Diagram*

UI Design

Introduction

The UI Design section provides a detailed walkthrough of the design for the mobile app (for patients) and the web interface (for providers). The goal of the design is to present a clear and intuitive flow that facilitates smooth user experiences, ensuring that both patients and providers can easily record and monitor dry weights without confusion. This section will walk through key screens for both types of users, describing each screen's functionality, layout, and design considerations. Each screen is evaluated on three of the ten Jakob Nielsen heuristics:

1. **Visibility of System Status**: This heuristic emphasizes keeping users informed about what's happening in the system at any given moment. It's important that feedback is provided so users know their actions are being processed.
2. **Match Between the System and the Real World**: The system should use language, symbols, and concepts that are familiar to the user, based on their real-world experience. Information should be presented in a logical and natural way.
3. **User Control and Freedom**: Users should have the ability to undo or redo actions. Users should feel in control, with the freedom to backtrack or make corrections without being locked into an irreversible action.
4. **Consistency and Standards**: Consistency involves ensuring that similar elements function in the same way across the system, making the user experience predictable and intuitive.
5. **Error Prevention**: Rather than relying on error messages to fix mistakes after they happen, the system should be designed to prevent errors from occurring in the first place.
6. **Recognition Rather than Recall**: Users should be able to recognize information and options rather than having to recall them from memory. This means presenting all necessary options and information clearly on the interface.
7. **Flexibility and Efficiency of Use**: The design should accommodate both novice and expert users by offering features that allow faster interactions for experienced users, while still being simple and intuitive for newcomers.
8. **Aesthetic and Minimalist Design**: The user interface should focus on displaying only the essential information, avoiding unnecessary elements that might distract or confuse the user. A clean, visually appealing design with an organized layout helps users focus on the key tasks.
9. **Help Users Recognize, Diagnose, and Recover from Errors**: When errors do occur, the system should provide clear and constructive error messages. These messages should not only describe what went wrong but also offer a solution or guide users on how to fix the problem.
10. **Help and Documentation**: While the system should be designed to be intuitive enough that users don't always need help, sometimes guidance is necessary. In those cases, help or documentation should be easily accessible and provide concise instructions, troubleshooting tips, or step-by-step tutorials.

Patient Mobile App

Home Page

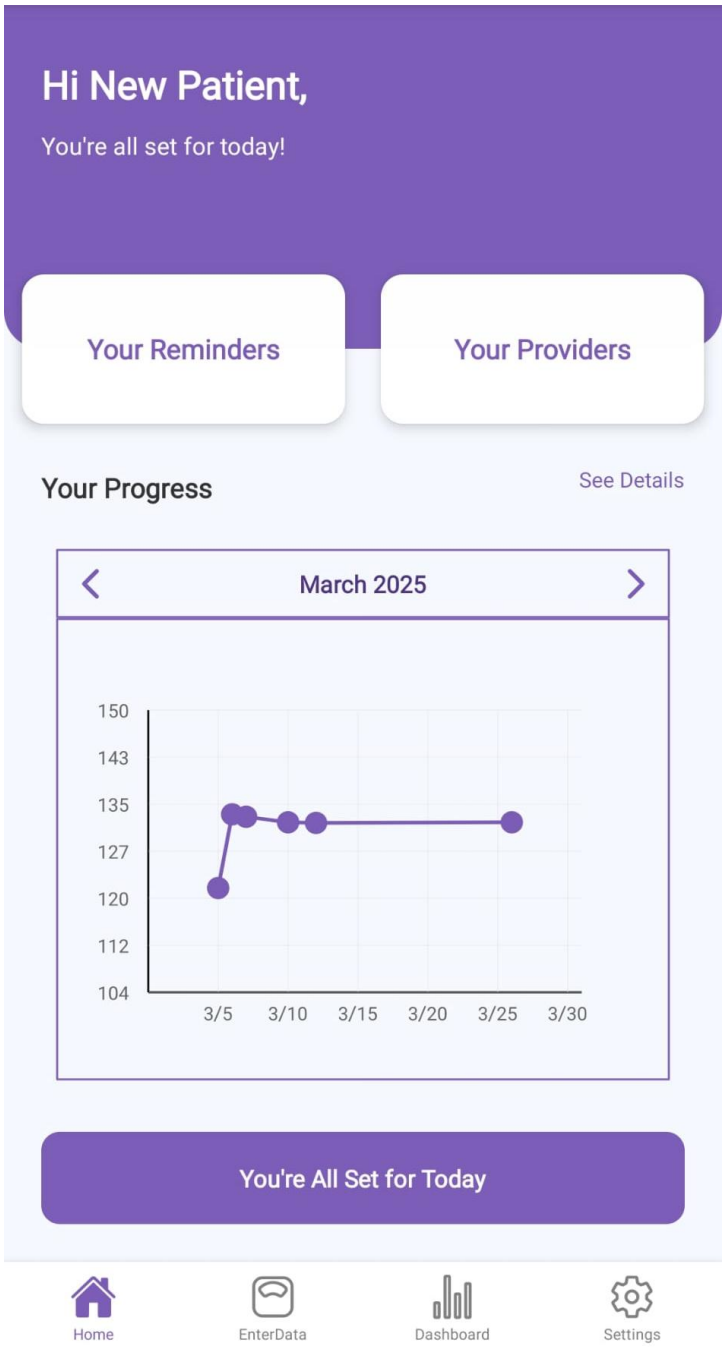

***Figure 6**: Patient Mobile App Home Page*

The home page of the patient mobile app. As the main hub of the app, it has easy access buttons to all main functions (reminders, providers, weight tracker and data input page). Additionally, it includes a greeting to the user and a status check. If they have updated their weight today it will inform the user that there is nothing else to do for the day, otherwise it will remind the user to update today's weight.

Walkthrough

After the patient logs in, this is the first page they will see, their name will be displayed at the top of the screen (where it says New Patient) and a status for the day will be displayed below it stating whether the patient has input their weight for the day or not.

If they haven't, the patient will see a button at the bottom that says "Enter Today's Data" which after pressing will lead them to the Enter Data page, which is explained right after this one in the document.

If they have already inputted their weight for today, then it shows a message saying, "You're all set for today". This doesn't mean the user cannot interact with the rest of the app - they can still access their progress by either using the nav bar or clicking the "See Details" button next to the "Your Progress" title. Additionally, they can access, add or modify their reminders and providers by using the two quick access buttons located above the progress section.

Heuristic Evaluation

The three heuristics that we will discuss for this page are: **Visibility of System Status**, **User Control and Freedom**, and **Consistency and Standards**.

Visibility of System Status

The home page provides clear feedback about the daily status of the user, such as confirming that the patient is "all set for today." The graph also visually displays the patient's progress over the month, allowing them to easily track their status.

User Control and Freedom

In the second place, The UI offers multiple options, such as viewing "Your Reminders," "Your Providers," and detailed progress information. This allows users to control their interactions with the system and navigate according to their preferences.

Consistency and Standards

Finally, the interface design also adheres to standard mobile UI conventions, which reduce cognitive load and make the app intuitive for new users. The familiar icons for navigation (Home, Enter Data, Dashboard, and Settings) create consistency with other apps. It also uses clear language like "You're All Set for Today," which is easy to understand.

Enter Data Page

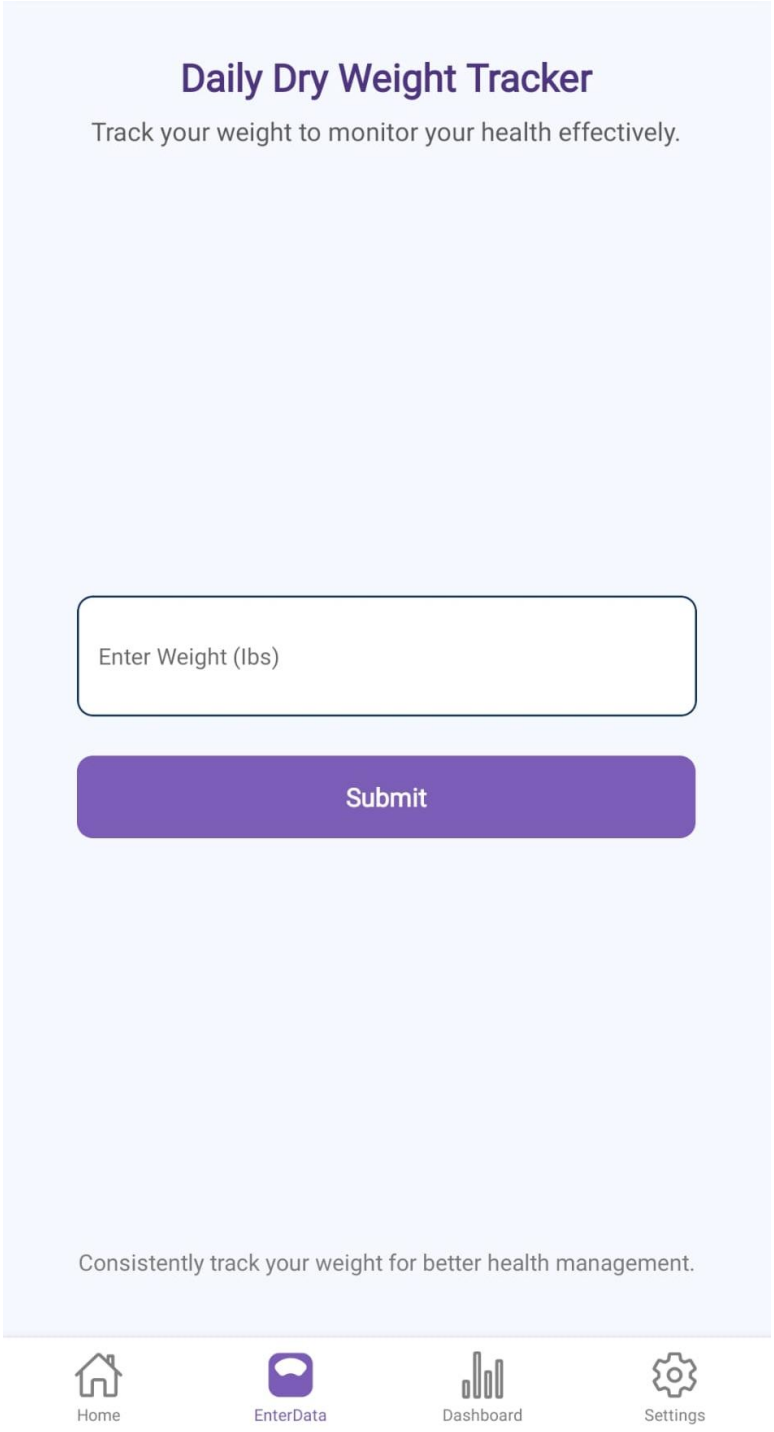

***Figure 7**: Patient Mobile App Enter Data Page*

The enter data page of the app is the simplest, yet one of the most important pages for our users to interact with. It simply includes a text box in which to add the current weight of the day.

Walkthrough

If the patient wishes to input a weight for the day they will do so on this page. Here, they will add their weight to the text box and press the submit button. After submission a pop-up will appear letting the user know that the weight was successfully inputted. If the weight change between the last input and the current input surpasses the provider's pre-determined threshold, then a notification is sent to the provider letting them know that their patient has had a drastic weight change and should act accordingly.

Heuristic Evaluation

The three main heuristics applied in this page are: **Match Between System and the Real World**, **Flexibility and Efficiency of Use**, and **Aesthetic and Minimalist Design**.

Match Between System and the Real World

The app uses clear, familiar language and terminology that users would understand in a health and wellness context. For example, the title "Daily Dry Weight Tracker" and the instruction "Enter Weight (lbs)" are both straightforward and aligned with real-world weight-tracking practices.

Flexibility and Efficiency of Use

The interface allows users to input their weight quickly using a single text field for weight entry, which is a simple and efficient way of collecting data. For frequent users, this simplicity makes it easy to enter and submit their weight without excessive steps.

Aesthetic and Minimalist Design

The design is clean, with a minimalist layout. It avoids any unnecessary information or visual clutter, keeping only the essential elements: the weight input field, the submit button, and supporting text. This simplicity helps the user focus on the task at hand.

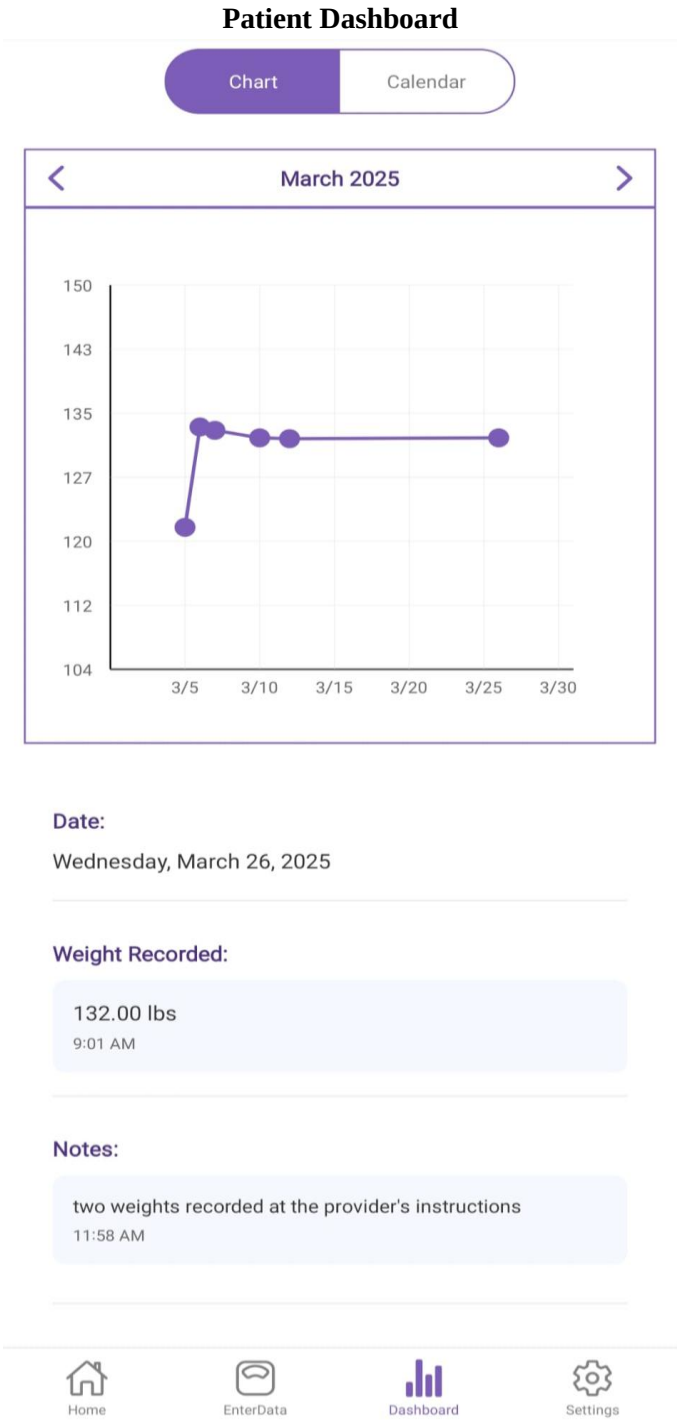

***Figure 8**: Patient Mobile App Patient Dashboard*

This is the dashboard page which helps the patient track their weight over time and see their providers' notes on their weights.

Walkthrough

If the patient wishes to view their progress, they can access their Dashboard page. Here they can view a graph of their weight through the months, or a calendar displaying which days they've logged their weight, and which ones haven't. Additionally, they can select the dots in the graph for each entry to see the exact date and weight recorded, and any notes added by their providers to that day's input.

Heuristic Evaluation

The three heuristics that are best applied in this page are: **Visibility of System Status**, **Match Between System and the Real World**, and **User Control and Freedom**.

Visibility of System Status

The chart clearly updates with the patient's weight over time, and the "Weight Recorded" section provides immediate feedback on the most recent entry, confirming to the user that the system is working as expected. The Notes section gives the patient their healthcare providers' notes on their recorded weights.

Match Between System and the Real World

The terminology and visual representation (e.g., "Weight Recorded" and the chart with dots) are aligned with everyday understanding, making it intuitive for patients to use without needing specialized knowledge.

User Control and Freedom

The design allows users to interact with the chart in a simple way using the "Chart" and "Calendar" toggle buttons at the top. This lets users' control whether they want to see the data in a graphical format or a calendar view. They can also look through different months and explore how their weight has changed over time.

Account Settings

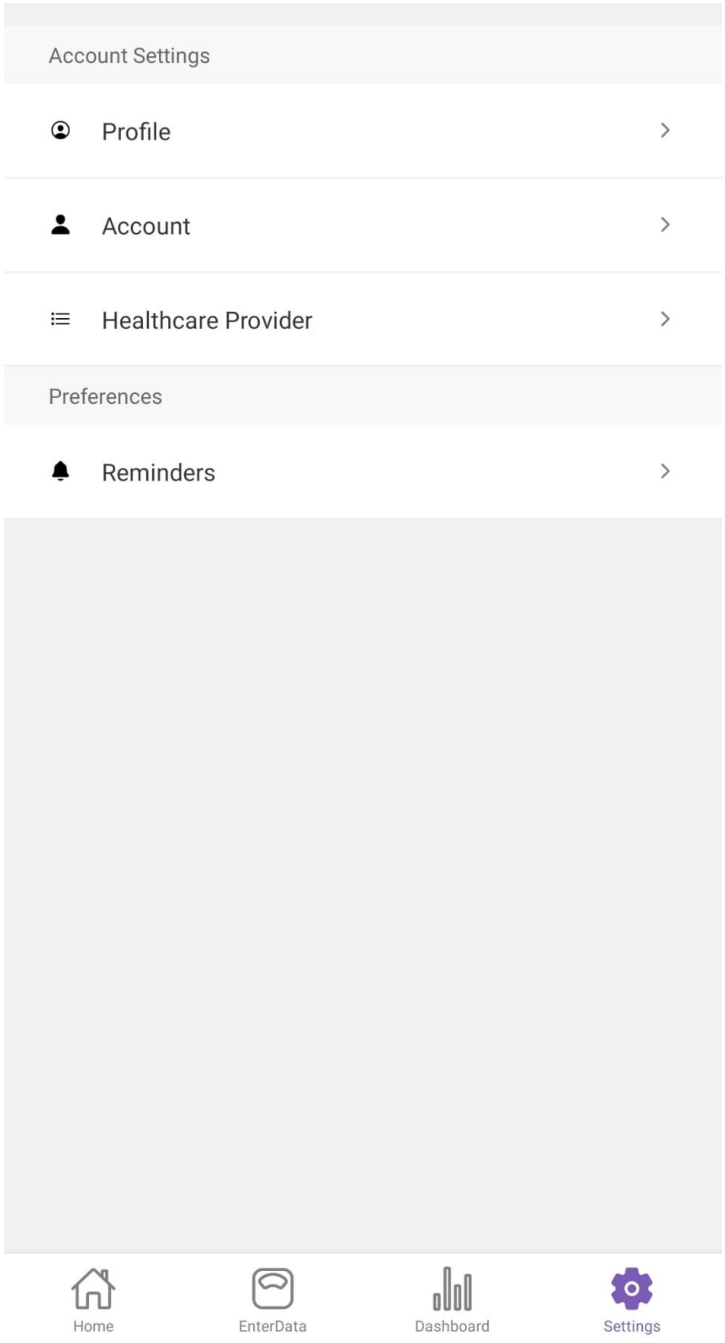

***Figure 9**: Patient Mobile App Account Settings*

This is the account management section of the app, where the user can access and edit their profile information, delete their accounts, access and edit their reminders, and healthcare providers.

Walkthrough

If the patient wishes to manage their account-related information, they can access the Settings Page, where they have several options, they can view/edit their profile information by going into the profile page, they can view their healthcare provider information by going into the corresponding page, and they can view or modify their reminders by going into the reminders page. Additionally, they can log out or delete their accounts by accessing the Account page.

Heuristic Evaluation

The three heuristics that are best demonstrated by this page are: **Consistency and Standards**, **Recognition Rather Than Recall**, and **Aesthetic and Minimalist Design**.

Consistency and Standards

The layout adheres to standard mobile interface patterns, with a clearly labeled settings section and easily recognizable icons for profile, account, healthcare provider, and reminders. The consistent use of icons and menu items across the app makes navigation straightforward.

Recognition Rather Than Recall

All options under "Account Settings" and "Preferences" are clearly labeled with icons that make it easy to understand their function at a glance. This reduces the cognitive load on users, allowing them to easily recognize options rather than having to recall their purpose.

Aesthetic and Minimalist Design

The screen is clean and minimalist, presenting only the necessary information. The use of white space and a simple layout helps the user focus on each option without distraction. The purple accent color for the selected tab ("Settings") is visually appealing and consistent with the overall theme.

Patient Profile Data

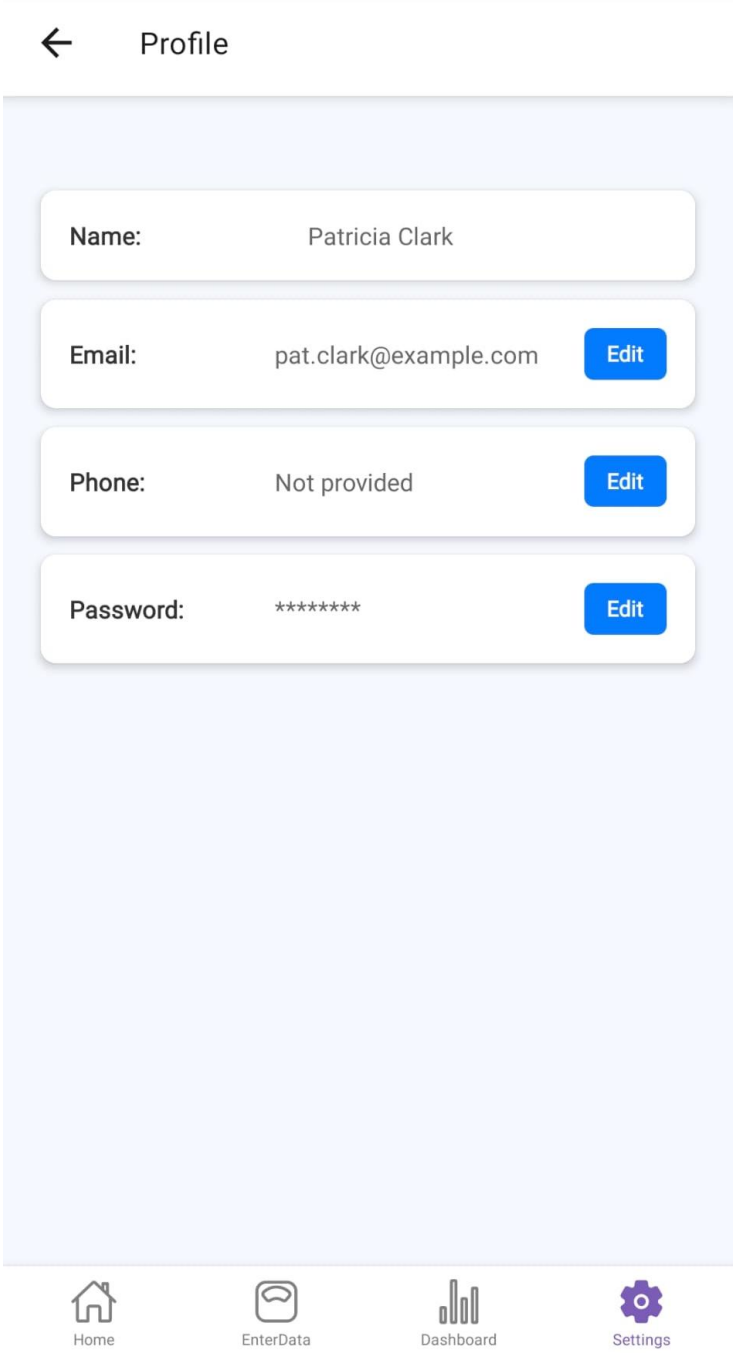

***Figure 10**: Patient Mobile App Profile Page*

This is the profile data page, where the patient can view and modify their contact details and password.

Walkthrough

If a patient wishes to modify their contact information or password, this is the page to do it, by clicking the update button they'll be able to modify said information. Checks are in place to ensure the correct formatting is being used in order to avoid invalid entries to the database.

Heuristic Evaluation

The three heuristics best demonstrated by this page are: **User Control and Freedom**, **Recognition Rather Than Recall**, and **Consistency and Standards**.

User Control and Freedom

The user has complete control over what information they want to edit. The "Edit" buttons allow the user to easily navigate and update each field without being locked into any action. Additionally, there is a clear back arrow for easy navigation to the previous screen.

Recognition Rather Than Recall

This screen uses a straightforward design, with the user's information (name, email, phone, and password) clearly displayed. Each field has a corresponding "Edit" button, which eliminates the need for the user to recall where to make updates.

Consistency and Standards

The design of this screen maintains consistency with the rest of the app, following a familiar mobile layout with clearly labeled sections and options. The use of the "Edit" button for each editable field is standard, making it easy for users to understand where they can make changes.

Provider List for Patients

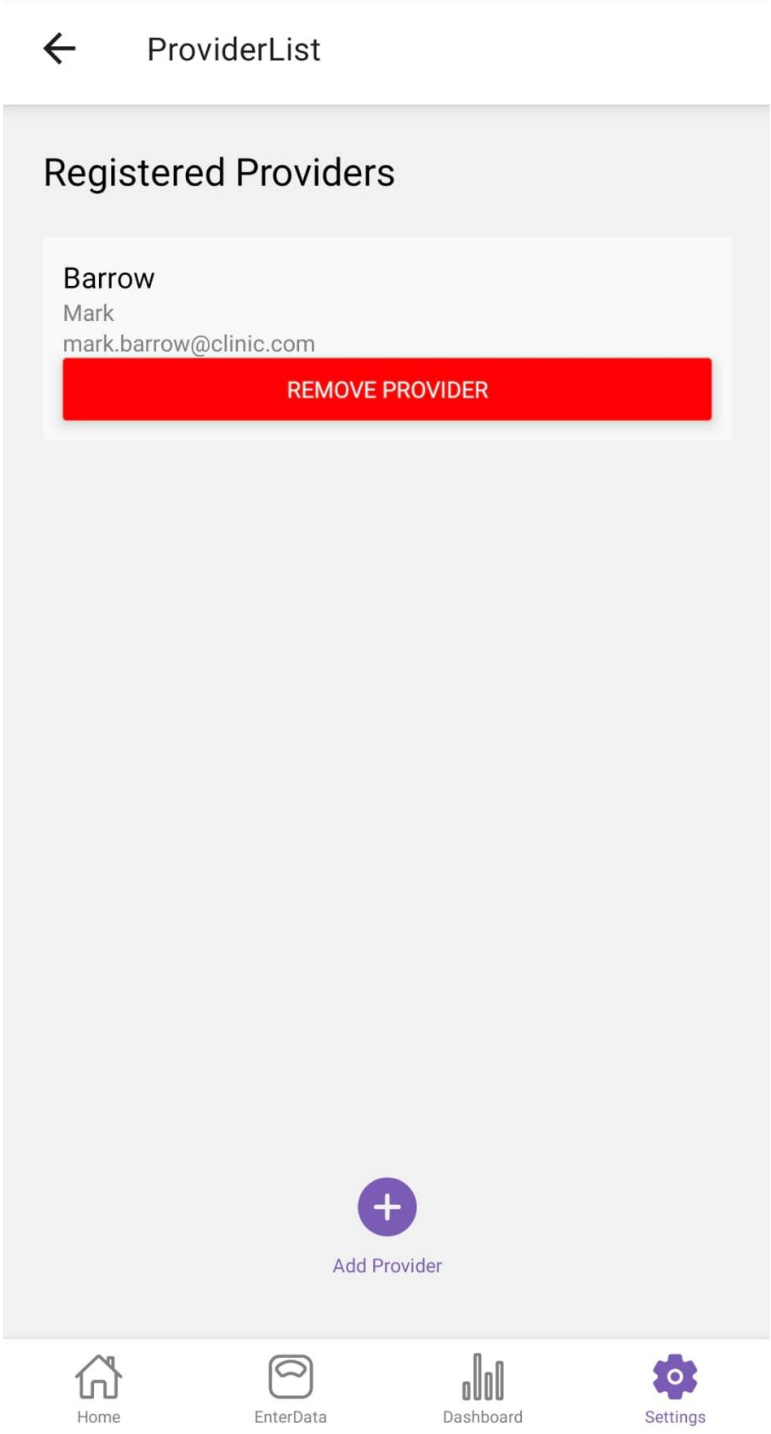

***Figure 11**: Patient Mobile App Healthcare Providers Page*

This is the page where the user can view all of their providers and remove them as necessary.

Walkthrough

If a patient wishes to add a provider, they will press the Add Provider button which would lead them to a specific page to complete the process (page is explained after this one). The patient can also remove a provider by clicking the remove provider under the corresponding healthcare provider information.

Heuristic Evaluation

The three best heuristics demonstrated by this page are: **Visibility of System Status**, **Match Between System and the Real World**, and **User Control and Freedom**.

Visibility of System Status

The screen clearly shows the user's registered provider(s) with visible contact information. The "Remove Provider" button is prominently displayed, allowing the user to immediately see their options. This gives clear feedback on the status of their provider's registration.

Match Between System and the Real World

The labels used for the provider's name, email, and the "Remove Provider" action align well with real-world expectations. The user's options (adding/removing a provider) are easy to understand, matching the familiar language and actions that users would expect in a healthcare-related app.

User Control and Freedom

The design allows the user to freely interact with their providers by removing or adding healthcare providers as needed. Additionally, there is a clear back arrow for easy navigation to the previous screen.

Add Provider

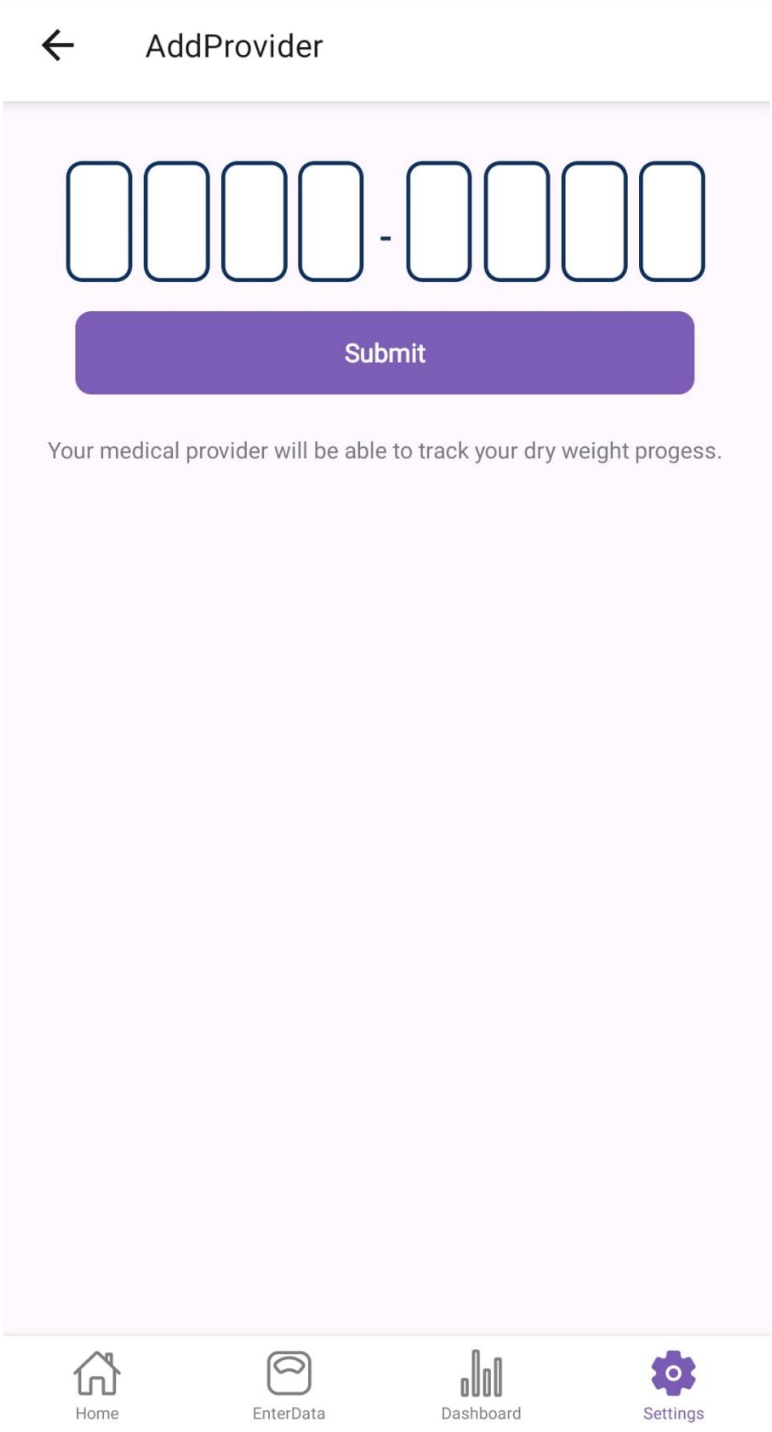

***Figure 12**: Patient Mobile App Add Provider Page*

This is the add provider page where the patient can add their provider by using their unique sharable id.

Walkthrough

When a patient wishes to add a new provider into their account they will be redirected to this page where they can input their provider’s unique sharable id. The text fields automatically move forward after one character has been inputted and it moves backward when deleting. When pressing the submit button, the backend will check whether the relationship already exists, or if the sharable id is incorrect and let the patient know which error has occurred. If nothing has gone wrong, there will be a pop-up informing the patient that the relationship was added successfully, and they will be able to see their new provider on the Provider List page described just before this.

Heuristic Evaluation

The three best heuristics demonstrated by this page are: **Visibility of System Status**, **Match Between System and the Real World**, and **Aesthetic and Minimalist Design**.

Visibility of System Status

The screen displays clear visual feedback with a form input area for entering the provider's shareable ID, and a "Submit" button. Additionally, the text beneath the input field informs the user that the provider will be able to track their dry weight progress once added.

Match Between System and the Real World

The language used ("Your medical provider will be able to track your dry weight progress") is straightforward and easily understood. The design of the input area mimics real-world forms, and the simple layout is intuitive.

Aesthetic and Minimalist Design

The design is clean and minimal, with the main focus on the provider ID input and the "Submit" button. The use of light colors, white space, and a clear button ensures a simple and non-cluttered interface.

Patient Reminders Management Page

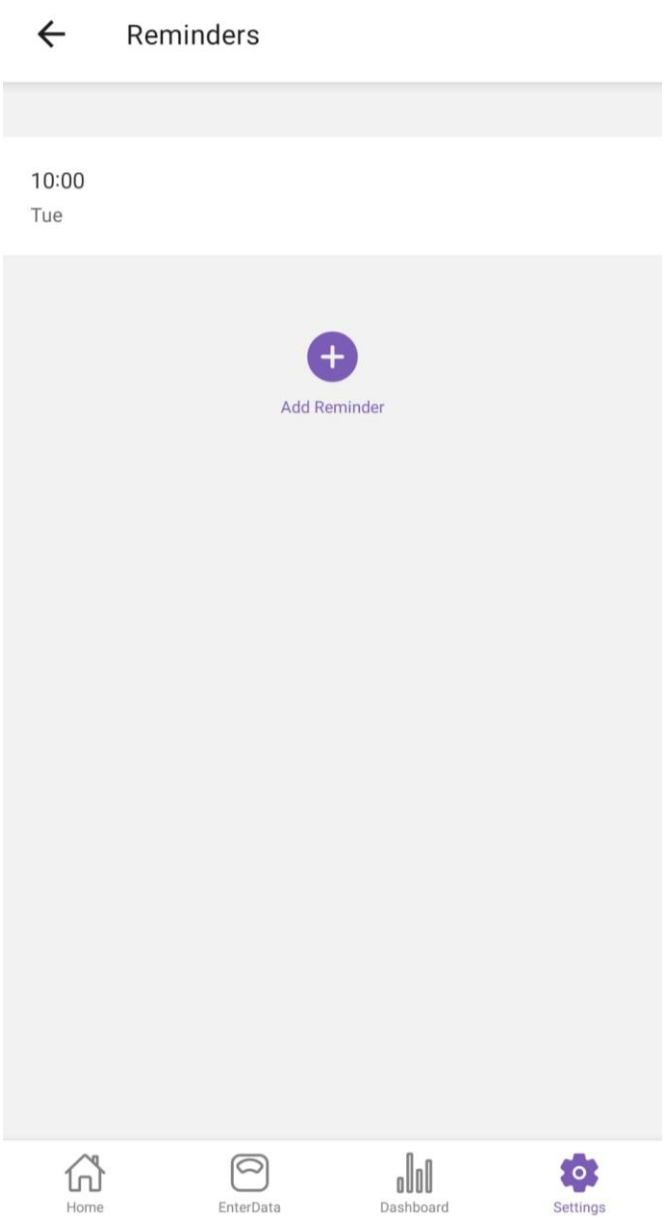

***Figure 13**: Patient Mobile App Reminders Page*

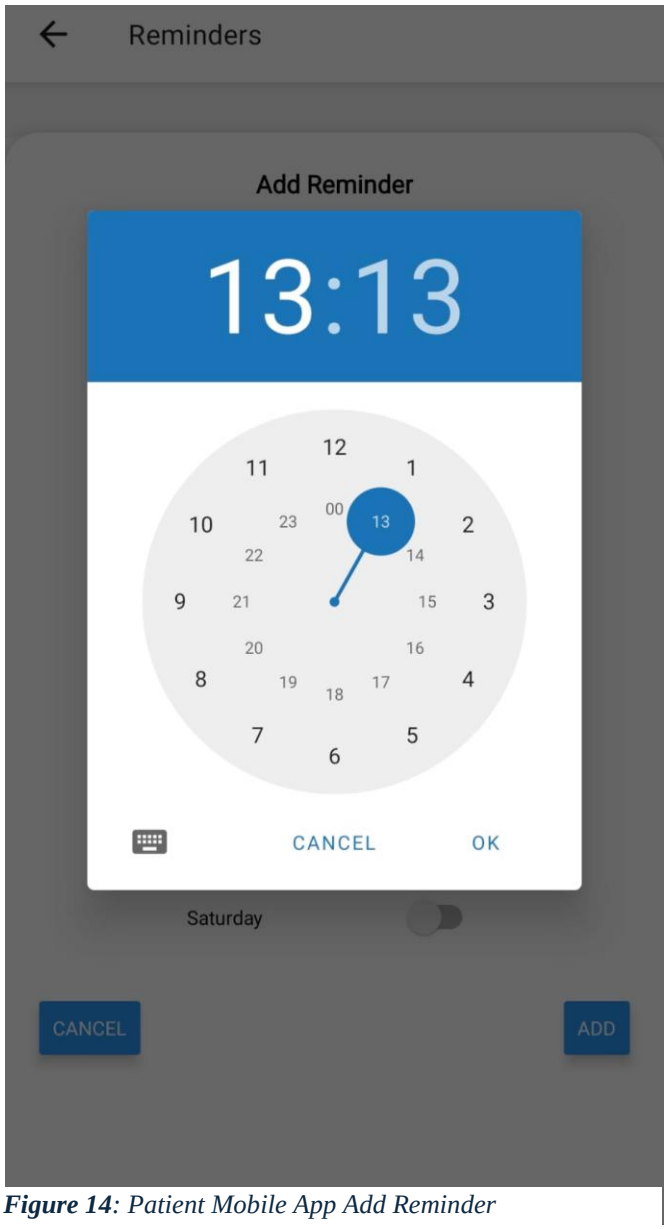

***Figure 14**: Patient Mobile App Add Reminder*

In this page users can manage their reminders, they can add, click on them and edit, or delete them as necessary.

Walkthrough

If a patient wishes to modify, view or create a reminder, they will do so on this page. To add a reminder, just press the add reminder button and the popup from Figure 14 will open allowing the patient to modify the time and days for this recurring reminder.

If the patient wishes to modify the reminder, they can press on the reminder and the same popup will appear, with an additional Delete button for the reminder to be permanently eliminated.

Heuristic Evaluation

The three best heuristics demonstrated by this page are: **Flexibility and Efficiency of Use**, **User Control and Freedom**, and **Error Prevention**.

Flexibility and Efficiency of Use

The design allows users to easily navigate and customize reminders. They can select the time and day(s) for reminders, enabling quick setup for both new and experienced users.

User Control and Freedom

The design allows the user to add, edit, or remove reminders as needed, providing full control over their reminder schedule. The interface also includes an easy way to navigate back to the previous screen.

Error Prevention

The reminder setup process is step-by-step, guiding users through selecting a time and day without the risk of selecting an invalid configuration. This prevents errors from occurring while creating reminders.

Provider Web App

Provider Home Page

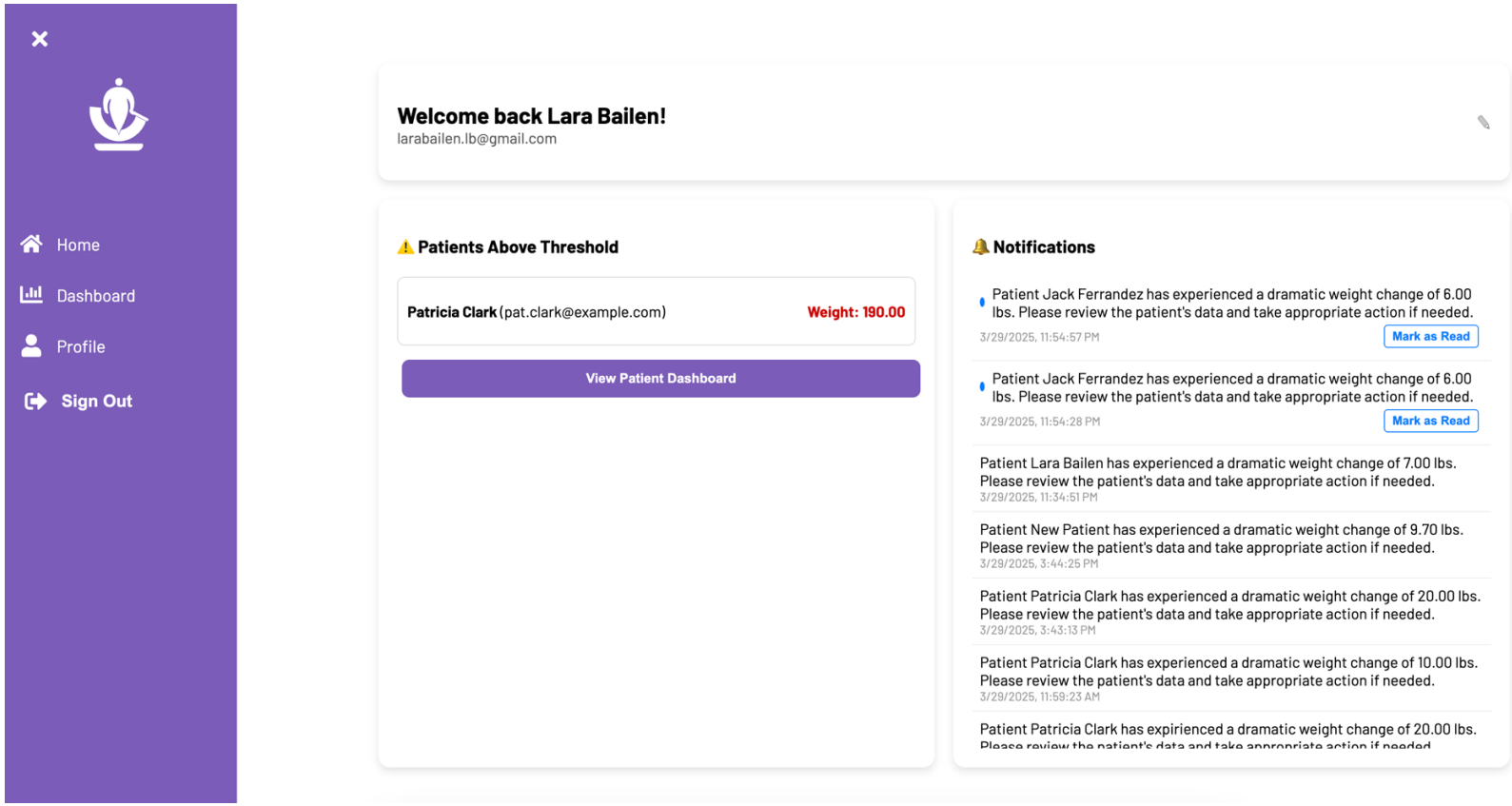

***Figure 15**: Provider Web App Home Page*

The Provider home page is a concise one-stop page for the provider to access all the most relevant information, like which patients have exceeded their threshold and their notifications.

Walkthrough

The provider can access this page to quickly see which patients are above the threshold. They can view all their patients by pressing the View Patient Dashboard or by using the navigation bar. And they can view the patient's details by pressing on the patient's card. The providers can view their notifications and mark them as read as necessary.

If the provider wishes to access or modify their profile they can click on the pencil icon to the right of the top container.

Heuristic Evaluation

The three best heuristics demonstrated by this page are: **Visibility of System Status**, **Match Between System and the Real World**, and **User Control and Freedom**.

Visibility of System Status

The provider home page provides clear feedback about the patient's status, particularly those who have surpassed a pre-specified threshold. This allows the provider to immediately see which patients require attention. Additionally, notifications about dramatic weight changes are prominently displayed, offering immediate system status updates. The interface also displays unread notifications with a blue dot and a button to mark them as read.

Match Between System and the Real World

The language used (e.g., "Patients Above Threshold" and "Notifications") is clear and aligns with healthcare terminology, making it easy for the provider to understand the system's feedback. The layout also mimics typical provider dashboards, with relevant patient details and weight information displayed intuitively.

User Control and Freedom

The "View Patient Dashboard" button provides an easy way for the provider to drill down into the specific patient details. Additionally, the "Mark as Read" button allows the provider to manage notifications and keep track of which alerts have been reviewed. This gives providers full control over interacting with patient data and notifications.

Patient Dashboard

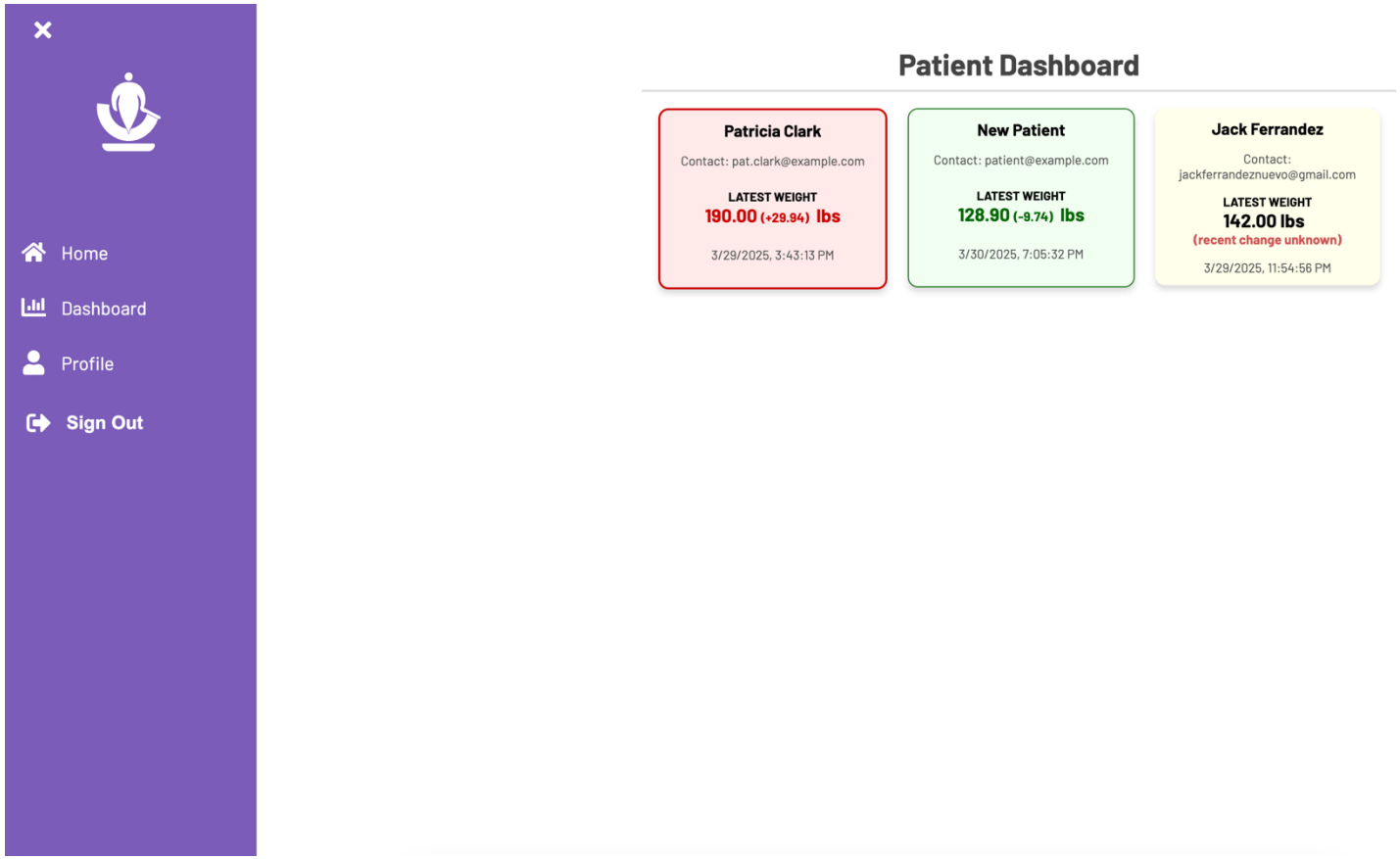

***Figure 16**: Provider Web App Patient Dashboard*

The patient dashboard is where the provider can view all their patients as quick cards with basic important information; colors notify the provider of whether the patient has exceeded their threshold, i.e. red if the patient's weight change has exceeded the threshold, green if it hasn't, and yellow if the information isn't available.

Walkthrough

The Patient Dashboard allows the provider to view all their patients, they can view whether they have had a drastic weight change or not by checking the colors of the latest weight (if red, drastic weight detected, green otherwise). If the provider wants a more detailed view of their patient's information, they can just press on the patient card and a detailed patient page will be opened (explained later in the document).

Heuristic Evaluation

The three best heuristics demonstrated by this page are: **Flexibility and Efficiency of Use**, **Recognition Rather Than Recall**, and **Aesthetic and Minimalist Design**.

Flexibility and Efficiency of Use

The design of the dashboard supports both novice and expert users. Both types of users can quickly navigate through the interface, accessing patient information with minimal effort. The layout is flexible, offering multiple ways to view patient data (e.g., viewing patient names and weights, or tapping to view individual patient dashboards). It's easy to identify the status of each patient at first glance.

Recognition Rather Than Recall

The patient dashboard displays key information such as patient names, latest weights, differences between current and previous weights, and timestamps in a clear and intuitive layout. This reduces cognitive load by presenting all essential details at a glance, which means providers don't have to remember or search for this information elsewhere.

Aesthetic and Minimalist Design

The dashboard has a clean and organized design that avoids unnecessary elements. The patient cards display the most critical information—patient name, weight, weight change, status, and contact details, without clutter. The overall layout focuses on providing just what is needed for efficient navigation and decision-making.

Provider Profile Page

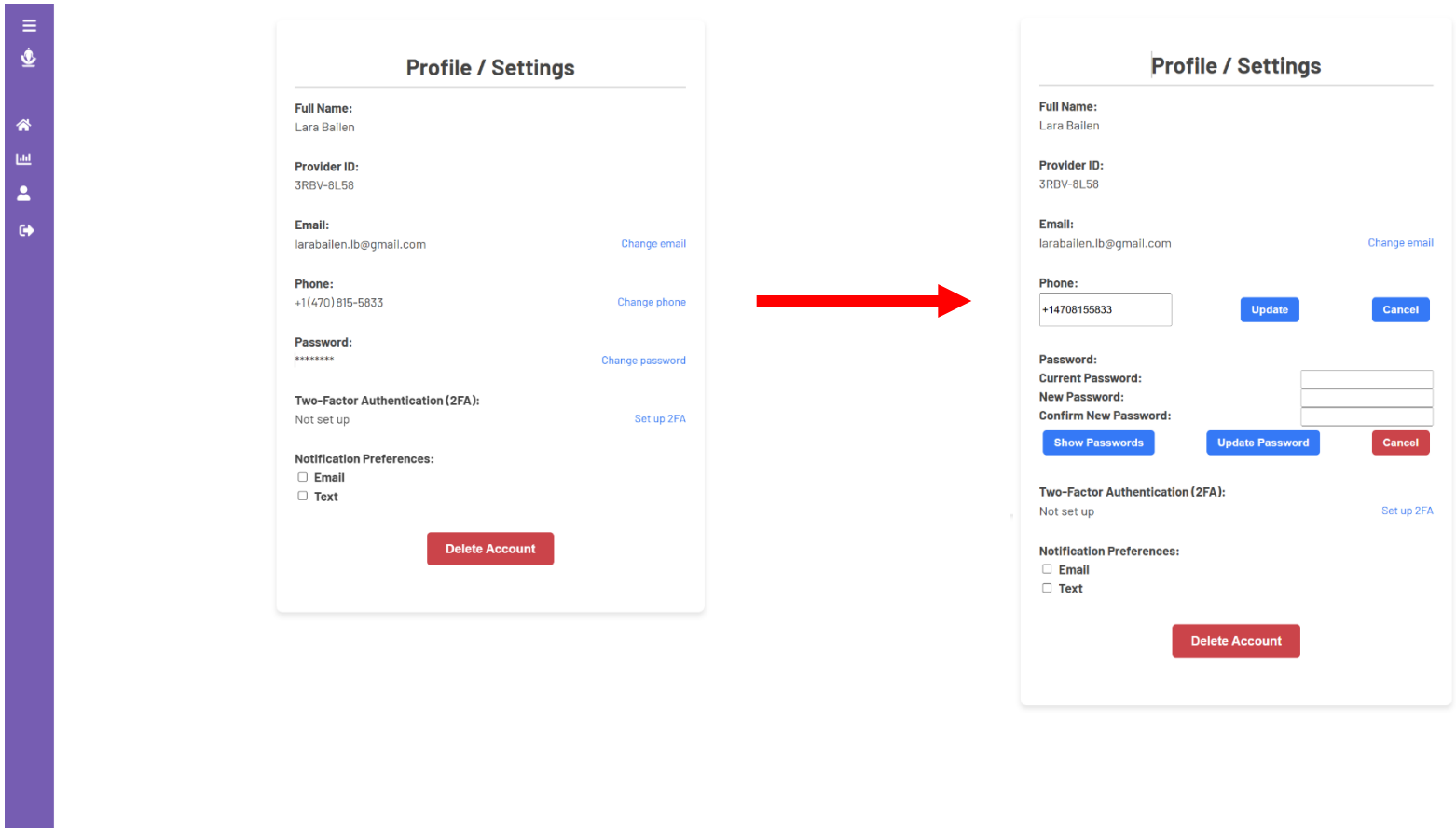

***Figure 17**: Provider Web App Profile Page*

The profile page is where the providers can modify their contact information and preferences, and change their password, along with other account-related operations such as deleting their account.

Walkthrough

If the provider wishes to modify their contact information and preferences, they would go to this page. Each modifiable field has a change button to the right that makes the fields modifiable. Before being able to submit these changes, there are checks in place to ensure that no invalid information is added to the database.

Here providers are also allowed to set up 2-factor Authentication or change their notification preferences.

Additionally, if the provider wishes to delete their account, they can press the delete account button to do so.

Heuristic Evaluation

The three best heuristics demonstrated by this page are: **Recognition Rather than Recall**, **Flexibility and Efficiency of Use**, and **Help Users Recognize, Diagnose, and Recover from Errors**.

Recognition Rather than Recall

The profile settings page displays all relevant details about the provider's account (e.g., full name, email, phone, provider ID) clearly labeled, with "Change" links next to editable fields. This allows the provider to easily recognize what information can be updated without needing to remember specific sections or actions.

Flexibility and Efficiency of Use

The page offers flexibility by allowing the provider to change essential account information, set up two-factor authentication (2FA), and manage notification preferences. The options to modify various aspects of the account, such as password, email, and notifications, provide an efficient workflow for maintaining account settings.

Help Users Recognize, Diagnose, and Recover from Errors

While user input errors are inevitable, we ensure that any attempt to update personal information is validated for correctness. When the user updates their email, phone number, or password, the system checks that the email and phone number are properly formatted and that the new passwords match. If an error occurs during this process, a clear and informative error message is displayed, guiding the user on what went wrong and how to correct it.

Patient Details Page

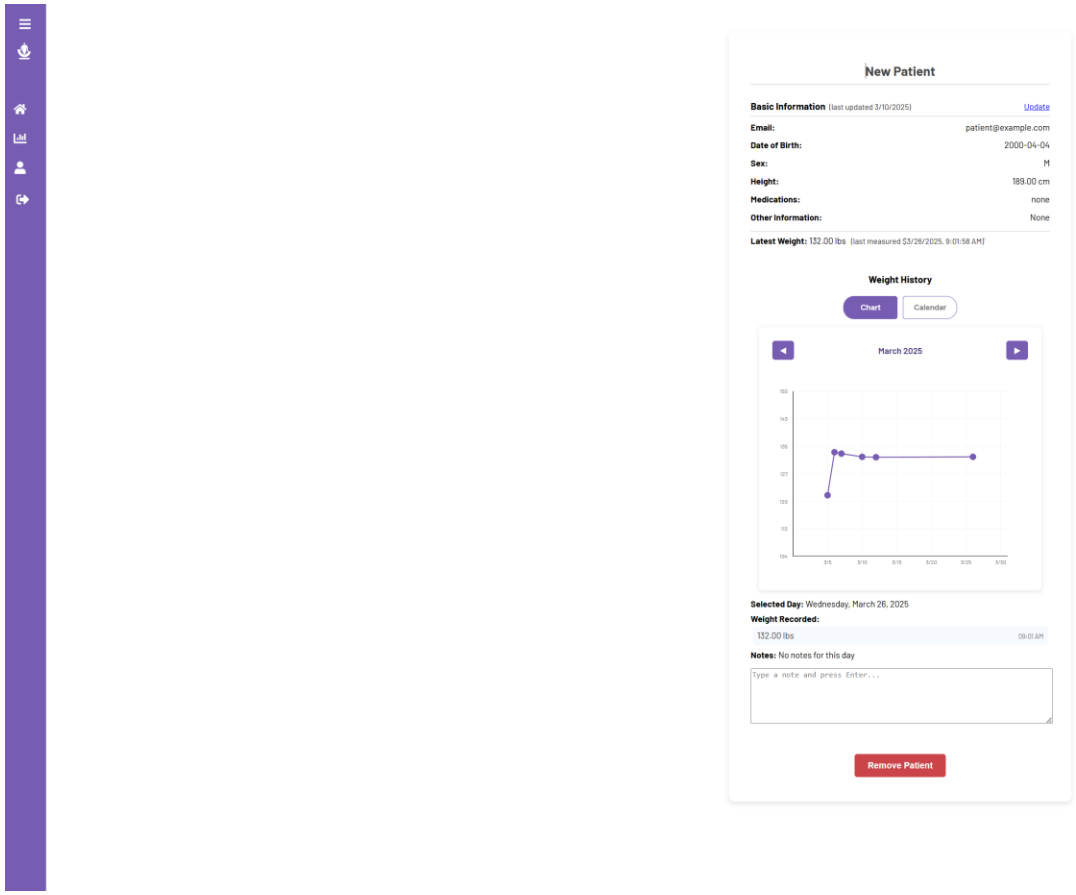

***Figure 18**: Provider Web App Patient Details Page*

The patient details page is where a provider can view more detailed information about their patients, such as weight progression through the months, notes added to weight records on certain days, and their patient's general information. Additionally, here is where providers can remove patients who are no longer under their care.

Walkthrough

This is the page where a provider can view all the details for one of their patients. Here they can update their information, like sex, date of birth, height, medications, etc. They can view their patients' progress through a graph or calendar (same as on the patients' side) and view specific details for selected entries. Additionally, they can add notes on the patient's weight records explaining some of the data as necessary. These notes can be edited or deleted as needed as well.

Additionally, if the patient will no longer be in the care of the provider, they can remove the patient from their tracking by pressing the Remove Patient button.

Heuristic Evaluation

The three best heuristics demonstrated by this page are: **Visibility of System Status**, **User Control and Freedom**, and **Aesthetic and Minimalist Design**.

Visibility of System Status

The page provides clear feedback to the user by displaying the selected patient's current status, weight, and related information. It also shows the weight history in the form of a chart, which updates as the user navigates through the graphs or calendar. The Weight Recorded section clearly shows the most recent data and the lack of new weight records for the day. The chart and calendar toggles give the user a visual representation of the patient's weight history, helping track progress. The page dynamically updates the status based on selected dates and notes

User Control and Freedom

The design of this page gives the user flexibility and control over their interaction, including the ability to add notes or remove a patient. Users can select different dates to view weight history and record notes, giving them control over the displayed information. There is a clear remove-patient option for providers, providing them full control over patient management. The design allows users to navigate through the calendar and chart views, ensuring they have the freedom to explore historical data.

Aesthetic and Minimalist Design

The interface maintains simplicity, presenting only relevant information without unnecessary elements that would distract the user. The use of white space and a clean layout enhances usability. The page avoids clutter by focusing on essential data: basic patient information, weight history, and notes. The minimalist design ensures that users can focus on their tasks (i.e., managing and viewing patient data) without being overwhelmed by unnecessary options or design elements. The color scheme (purple accents) is consistent, adding a subtle touch to the overall design without detracting from the functionality.

Appendix A: RESTful API Documentation

Possible Response Codes

Code	Text	Description
200	OK	Success
201	Created	A new resource was successfully created.
400	Bad Request	The request could not be understood due to invalid syntax or parameters
401	Unauthenticated	Authentication is required and has failed or has not yet been provided
403	Forbidden	Authentication credentials were provided, but you do not have permission
404	Not Found	The requested resource could not be found
405	Method Not Allowed	The HTTP method is not supported for the requested endpoint
500	Internal Server Error	A generic server error occurred on the server
503	Service Unavailable	The server is currently unavailable, often due to maintenance or overload

Endpoint Index

Shared Endpoints

POST /login/

Authenticates the user and logs them in. Returns a JWT token for patient users and a session-based authentication for provider users.

Resource URL: https://api.dryweightwatchers.com/login/

Request Information

- Request Format: JSON
- Requires Authentication: No (Establishes authentication)

Parameter	Description
email	Email of the user trying to login
password	Password of the user

Example Request

```
> POST https://api.dryweightwatchers.com/login/
```

JSON:

```
{
    "email": "user_email@gmail.com",
    "password": "user_password"
}
```

Example Response:

- Response Format: JSON

Example Response (Patient)

```
{
    "status": 200,
    "message": "Login successful",
    "role": "patient",
    "access_token": "access_token_value",
    "refresh_token": "refresh_token_value"
}
```

Example Response (Provider)

```
{
    "status": 200,
     "message": "Login successful",
     "role": "provider",
     "sessionid": "session_id_value"
 }
```

Response Fields:

Field	Description
status	Response code from request
message	Message indicating login success or failure
role	Role of the logged-in user ('patient' or 'provider')
access_token	JWT token for patient users
refresh_token	Refresh token for patient users
sessionId	Session ID for provider users

POST /logout/

Logs out the user by clearing their session

Resource URL: https://api.dryweightwatchers.com/logout/

Request Information

- Request Format: None (session-based logout)
- Requires Authentication: Yes (Session or JWT)

Example Request

```
> POST https://api.dryweightwatchers.com/logout/
```

Example Response

- Response Format: JSON

Example Response

```
{
    "status": 200,
    "message": "Logout successful"
}
```

Response Fields:

Field	Description
status	Response code from request
message	Message indicating logout success or failure

POST /change-email/

Changes the logged-in user's email address.

Resource URL: https://api.dryweightwatchers.com/change-email/

Request Information

- Request Format: JSON
- Requires Authentication: Yes (Session or JWT)

Parameter	Description
email	New email address to be set

Example Request

```
> POST https://api.dryweightwatchers.com/change-email/
```

JSON:

```
{
    "email": "new_email@example.com"
}
```

Example Response

- Response Format: JSON

Example Response (Patient)

```
{
    "status": 200,
    "message": "Email updated successfully",
    "email": "new_email@example.com"
}
```

Response Fields

Field	Description
status	Response code from request
message	Message indicating email update
email	The updated email address

POST /change-phone/

Changes the logged-in user's phone.

Resource URL: https://api.dryweightwatchers.com/change-phone/

Request Information

- Request Format: JSON
- Requires Authentication: Yes (Session or JWT)

Parameter	Description
phone	New phone number to be set

Example Request

```
> POST https://api.dryweightwatchers.com/change-phone/
```

JSON:

```
{
    "phone": "+1234567890"
}
```

Example Response

- Response Format: JSON

Example Response

```
{
    "status": 200,
    "message": "Phone number updated successfully",
    "email": "+1234567890"
}
```

Response Fields

Field	Description
status	Response code from request
message	Message indicating email update
phone	The updated phone number

POST /change-password/

Changes the logged-in user's password.

Resource URL: https://api.dryweightwatchers.com/change-password/

Request Information

- Request Format: JSON
- Requires Authentication: Yes (Session or JWT)

Parameter	Description
current_password	The current password of the user
new_password	The new password to set
confirm_password	The new password again for confirmation

Example Request

```
> POST https://api.dryweightwatchers.com/change-password/
```

JSON:

```
{
    "current_password": "old_password123",
    "new_password": "new_password456",
    "confirm_password": "new_password456"
}
```

Example Response

- Response Format: JSON

Example Response

```
{
    "status": 200,
    "message": " Password updated successfully",
}
```

Response Fields

Field	Description
status	Response code from request
message	Message indicating password update

DELETE /delete-account/

Deletes the logged-in user's account and creates a deactivated user entry.

Resource URL: https://api.dryweightwatchers.com/delete-account/

Request Information

- Request Format: JSON
- Requires Authentication: Yes (Session or JWT)

Example Request

```
> DELETE https://api.dryweightwatchers.com/delete-account/
```

Example Response

- Response Format: JSON

Example Response

```
{
    "status": 200,
    "message": " Successfully deactivated account",
}
```

Response Fields

Field	Description
status	Response code from request
message	Message indicating account deletion

DELETE /delete-relationship/

Deletes the relationship between the logged-in user (patient or provider) and their associated provider or patient.

Resource URL: https://api.dryweightwatchers.com/delete-relationship/

Request Information

- Request Format: JSON
- Requires Authentication: Yes (Session or JWT)

Parameter	Description
shareable_id (for patients only)	Shareable ID of the provider to delete
id (for providers only)	ID of the patient to delete the relationship with

Example Request

```
> DELETE https://api.dryweightwatchers.com/delete-relationship/
```

Example Request (Patient) - JSON:

```
{
    "shareable_id": "provider_shareable_id"
}
```

Example Request (Provider) - JSON:

```
{
    "id": 123
}
```

Example Response

- Response Format: JSON

Example Response (Success)

```
{
    "status": 200,
    "message": " Relationship deleted successfully",
}
```

Example Response (Failure - Relationship Not Found)

```
{
    "status": 404,
    "message": " Relationship not found",
}
```

Response Fields

Field	Description
status	Response code from request
message	Message indicating whether the relationship was deleted or not

GET /verify-email/

Verifies a newly registered user's email using a token sent via email. This endpoint is typically triggered when the user clicks the verification link.

Resource URL: https://api.dryweightwatchers.com/verify-email/

Request Information

- Request Format: URL Query String
- Requires Authentication: No
- Query Parameters: token (Unique verification token from email)

Example Request

```
> GET https://api.dryweightwatchers.com/verify-email?
                token=abc123XYZtoken987/
```

Example Response

- Response Format: JSON

Example Response (Success)

```
{
    "status": 200,
    "message": " Email verified successfully",
}
```

Example Response (Failure)

```
{
    "status": 404,
    "error": " Not found",
}
```

Response Fields

Field	Description
sttaus	Response code from request
message	Confirmation of successful verification
error	Error message if the token is invalid or expired

Patient-Related Endpoints

POST /refresh-jwt/

Refreshes the access token using the provided refresh token.

Resource URL: https://api.dryweightwatchers.com/refresh-jwt/

Request Information

- Request Format: JSON
- Requires Authentication: No

Parameter	Description
refresh_token	The refresh token to generate a new access token

Example Request

```
> POST https://api.dryweightwatchers.com/refresh-jwt/
```

JSON:

```
{
    "refresh_token": "your_refresh_token_here"
}
```

Example Response

- Response Format: JSON

Example Response (Success)

```
{
    "status": 200,
    "access_token": "new_access_token_value",
}
```

Example Response (Success)

```
{
    "status": 401,
    "error": "Invalid or expired refresh token",
}
```

Response Fields

Field	Description
access_token	The new access token generated from the refresh token
error	Error message if the refresh token is invalid or expired

POST /register/

Registers a new user (patient) by providing the necessary details.

Resource URL: https://api.dryweightwatchers.com/register/

Request Information

- Request Format: JSON
- Requires Authentication: No

Parameter	Description
first_name	First name of the user
last_name	Last name of the user
email	Email address of the user
phone	Phone number of the user (optional)
password	Password for the user
confirm_password	Confirmation of the password

Example Request

```
> POST https://api.dryweightwatchers.com/register/
```

JSON:

```
{
    "first_name": "John",
    "last_name": "Doe",
    "email": "john.doe@example.com",
    "phone": "+1234567890",
    "password": "user_password123",
    "confirm_password": "user_password123"
}
```

Example Response (JSON):

Example Response (Success)

```
{
    "status": 201,
    "message": "User registered successfully"
}
```

Example Response (Failure - Validation Error)

```
{
    "status": 400,
    "errors": {
        "email": ["This email address is already registered."]
    }
}
```

Response Fields

Field	Description
status	Response code from request
errors	Any errors or validation issues, if present (e.g., missing fields, invalid email)
message	Message indicating the success or failure of the registration process

POST /add-relationship/

Adds a new relationship between a patient and a provider.

Resource URL: https://api.dryweightwatchers.com/add-relationship/

Request Information

- Request Format: JSON
- Requires Authentication: Yes (JWT)

Parameter	Description
shareable_id	Shareable ID of the provider to establish the relationship

Example Request

```
> POST https://api.dryweightwatchers.com/add-relationship/
```

JSON:

```
{
    "shareable_id": "provider_shareable_id"
}
```

Example Response (JSON):

Example Response (Success)

```
{
    "status": 201,
    "message": "Relationship created successfully"
}
```

Example Response (Failure - Invalid Provider ID)

```
{
    "status": 202,
    "message": "Relationship already exists"
}
```

Example Response (Failure - Invalid Provider ID)

```
{
    "status": 404,
    "error": "Invalid provider ID"
}
```

Response Fields

Field	Description
status	Response code from request
message	Message indicating the result of the operation
error	Error message if any issues occur (e.g., provider not found)

POST /patient-change-password/

Allows an authenticated patient to change their password by providing the current password and confirming the new one.

Resource URL: https://api.dryweightwatchers.com/patient-change-password/

Request Information

- Request Format: JSON
- Requires Authentication: Yes (JWT)

Parameter	Description
current_password	The current password of the user
new_password	The new password to set
confirm_password	The new password again for confirmation

Example Request

```
> POST https://api.dryweightwatchers.com/patient-change-password/
```

JSON:

```
{
    "current_password": "old_password123",
    "new_password": "new_password456",
    "confirm_password": "new_password456"
}
```

Example Response

- Response Format: JSON

Example Response

```
{
    "status": 200,
    "message": " Password updated successfully",
}
```

Response Fields

Field	Description
status	Response code from request
message	Message indicating password update

POST /record_weight/

Records a new weight entry for the authenticated patient.

Resource URL: https://api.dryweightwatchers.com/record-weight/

Request Information

- Request Format: JSON
- Requires Authentication: Yes (JWT)

Parameter	Description
weight	New weight value (lbs)

Example Request

```
> POST https://api.dryweightwatchers.com/record-weight/
```

JSON:

```
{
    "weight": 67.5
}
```

Example Response

- Response Format: JSON

Example Response (Success)

```
{
    "status": 200,
    "message": " Weight recorded successfully",
}
```

Example Response (Failure)

```
{
    "status": 400,
    "error": {
        "weight": ["This field is required."]
    }
}
```

Response Fields

Field	Description
status	Response code from request
message	Message indicating weight recorded
error	Description of what went wrong

GET /user/providers/

Returns the list of providers associated with the authenticated patient.

Resource URL: https://api.dryweightwatchers.com/user/providers/

Request Information

- Request Format: None
- Requires Authentication: Yes (JWT)

Example Request

```
> GET https://api.dryweightwatchers.com/user/providers/
```

Example Response

- Response Format: JSON (array of providers)

Example Response

```
[
    {
        "id": 17,
        "first_name": "Dr. Jane",
        "last_name": "Smith",
        "email": "jane.smith@example.com"
    },
    {
        "id": 22,
        "first_name": "Dr. Alan",
        "last_name": "Brown",
        "email": "alan.brown@example.com"
    }
]
```

Response Fields

Field	Description
Array of providers containing the following data	
id	Provider's user ID
first_name	Provider's first name
last_name	Provider's last name
email	Provider's email address

GET /patient-profile/

Returns the basic profile information of the authenticated patient.

Resource URL: https://api.dryweightwatchers.com/patient-profile/

Request Information

- Request Format: None
- Requires Authentication: Yes (JWT)

Example Request

```
> GET https://api.dryweightwatchers.com/patient-profile/
```

Example Response

- Response Format: JSON

Example Response

```
{
    "firstname": "Alice",
    "lastname": "Johnson",
    "email": "alice.johnson@example.com",
    "phone": "+1234567890"
}
```

Response Fields

Field	Description
firstname	Patient's first name
lastname	Patient's last name
email	Patient's email
phone	Patient's phone number

GET /get-weight-record/

Retrieves the full weight history for the authenticated patient, ordered by timestamp.

Resource URL: https://api.dryweightwatchers.com/get-weight-record/

Request Information

- Request Format: None
- Requires Authentication: Yes (JWT)

Example Request

```
> GET https://api.dryweightwatchers.com/get-weight-record/
```

Example Response

- Response Format: JSON (array of weight records)

Example Response

```
[
    {
        "weight": 67.5,
        "timestamp": "2025-03-25T10:23:00Z"
    },
    {
        "weight": 69.2,
        "timestamp": "2025-03-15T09:10:00Z"
    }
]
```

Response Fields

Field	Description
Array of weight records containing the following data	
weight	Recorded weight (kg)
timestamp	Date and time of record

GET /get-patient-notes/

Retrieves all notes associated with the authenticated patient, ordered by timestamp (most recent first).

Resource URL: https://api.dryweightwatchers.com/get-patient-notes/

Request Information

- Request Format: None
- Requires Authentication: Yes (JWT)

Example Request

```
> GET https://api.dryweightwatchers.com/get-patient-notes/
```

Example Response

- Response Format: JSON (array of notes)

Example Response

```
[
      {
          "note": "Patient reported increased fluid retention.",
          "timestamp": "2025-03-25T14:45:00Z"
      },
      {
          "note": "Adjusted dry weight by 1.5kg based on recent
 trends.",
          "timestamp": "2025-03-20T08:30:00Z"
      }
 ]
```

Response Fields

Field	Description
Array of notes containing the following data	
note	Text content of the note
timestamp	Date and time of entry

GET /get-reminders/

Returns all active reminders set by the authenticated patient.

Resource URL: https://api.dryweightwatchers.com/get-reminders/

Request Information

- Request Format: None
- Requires Authentication: Yes (JWT)

Example Request

```
> GET https://api.dryweightwatchers.com/get-reminders/
```

Example Response

- Response Format: JSON (array of reminders)

Example Response

```
[
    {
        "id": 1,
        "time": "08:00",
        "days": ["Mon", "Wed", "Fri"]
    },
    {
        "id": 2,
        "time": "20:30",
        "days": ["Tue", "Thu", "Sat"]
    }
]
```

Response Fields

Field	Description
Array of weight records containing the following data	
id	Unique ID of the reminder
time	Time of day the reminder is scheduled (HH:MM)
days	List of days the reminder is active (e.g., ["Mon", "Wed", "Fri"])

POST /add-reminder/

Creates a new reminder for the authenticated patient.

Resource URL: https://api.dryweightwatchers.com/add-reminder/

Request Information

- Request Format: JSON
- Requires Authentication: Yes (JWT)

Parameter	Description
time	Time of day for the reminder (format: "HH:MM", 24-hour)
days	List of days when the reminder should be active (e.g., ["Mon", "Wed"])

Example Request

```
> POST https://api.dryweightwatchers.com/add-reminder/
```

JSON:

```
{
    "time": "07:30",
    "days": ["Mon", "Tue", "Thu"]
}
```

Example Response

- Response Format: JSON

Example Response (Success)

```
{
    "status": 200,
    "message": " Reminder added successfully",
}
```

Example Response (Failure)

```
{
    "status": 400,
    "error": {
        "time": ["This field is required."]
    }
}
```

Response Fields

Field	Description
status	Response code from request
message	Confirmation message
error	Error details if creation fails

PUT /save-reminder/

Updates an existing reminder for the authenticated patient.

Resource URL: https://api.dryweightwatchers.com/save-reminder/

Request Information

- Request Format: JSON
- Requires Authentication: Yes (JWT)

Parameter	Description
id	ID of the reminder to update
time	The new time of day for the reminder (format: "HH:MM", 24-hour)
days	The new list of days when the reminder should be active (e.g., ["Mon", "Wed"])

Example Request

```
> PUT https://api.dryweightwatchers.com/save-reminder/
```

JSON:

```
{
    "id": 3,
    "time": "09:00",
    "days": ["Mon", "Wed", "Fri"]
}
```

Example Response

- Response Format: JSON

Example Response (Success)

```
{
    "status": 201,
    "message": " Reminder saved successfully",
}
```

Example Response (Failure)

```
{
    "status": 404,
    "message": " Reminder not found",
}
```

Example Response (Failure)

```
{
    "status": 400,
    "error": {
        "time": ["Enter a valid time."]
    }
}
```

Response Fields

Field	Description
message	Confirmation of successful update
error	Error message if update fails

DELETE /delete-reminder/<int:id>/

Deletes a specific reminder by its ID for the authenticated patient.

Resource URL: https://api.dryweightwatchers.com/delete-reminder/<id>

Request Information

- Request Format: None
- Requires Authentication: Yes (JWT)

Parameter	Description
id	ID of the reminder to delete

Example Request

```
> DELETE https://api.dryweightwatchers.com/delete-reminder/5
```

Example Response

- Response Format: JSON

Example Response (Success)

```
{
    "status": 201,
    "message": " Reminder deleted successfully",
}
```

Example Response (Failure)

```
{
    "status": 404,
    "message": " Reminder not found",
}
```

Response Fields

Field	Description
message	Confirmation of successful deletion
error	Error message if deletion fails

Provider-Related Endpoints

GET /get-auth-status/

Used to check whether the current provider session is authenticated

Resource URL: https://api.dryweightwatchers.com/get-auth-status/

Request Information

- Request Format: None
- Requires Authentication: Yes (JWT)

Example Request

```
> GET https://api.dryweightwatchers.com/get-auth-status/
```

Example Response

- Response Format: JSON

Example Response (Authenticated)

```
{
    "status": 200,
}
```

Example Response (Unauthenticated)

```
{
    "status": 403/401,
}
```

Response Fields

Field	Description
status	Status code of authentication

GET /get-csrf-token/

Retrieves a CSRF token for use in provider session-based authentication. This token should be included in subsequent POST, PUT, or DELETE requests as part of CSRF protection.

Resource URL: https://api.dryweightwatchers.com/get-csrf-token/

Request Information

- Request Format: None
- Requires Authentication: No

Example Request

```
> GET https://api.dryweightwatchers.com/get-csrf-token/
```

Example Response

- Response Format: JSON

Example Response

```
{
    "csrfToken": "X7G8rT2kNqs0...abc"
}
```

Response Fields

Field	Description
csrfToken	Stores the CSRF token (HttpOnly, Secure)

Response Headers (example):

```
Set-Cookie: csrftoken=X7G8rT2kNqs0...abc; HttpOnly; Secure; SameSite=None
```

POST /register-provider/

Registers a new provider account by submitting the required personal and credential information.

Resource URL: https://api.dryweightwatchers.com/register-provider/

Request Information

- Request Format: JSON
- Requires Authentication: No

Parameter	Description
first_name	First name of the user
last_name	Last name of the user
email	Email address of the user
phone	Phone number of the user (optional)
password	Password for the user
confirm_password	Confirmation of the password

Example Request

```
> POST https://api.dryweightwatchers.com/register-provider/
```

JSON:

```
{
    "first_name": "Dr. Sarah",
    "last_name": "Lee",
    "email": "sarah.lee@example.com",
    "phone": "+123456789",
    "password": "strongPassword123",
    "confirm_password": "strongPassword123"
}
```

Example Response

- Response Format: JSON

Example Response

```
{
    "status": 201
}
```

Response Fields

Field	Description
status	Status code of authentication

GET /profile/

Retrieves the profile information of the currently authenticated provider.

Resource URL: https://api.dryweightwatchers.com/profile/

Request Information

- Request Format: None
- Requires Authentication: Yes (Session)

Example Request

```
> GET https://api.dryweightwatchers.com/profile/
```

Example Response

- Response Format: JSON

Example Response

```
{
    "firstname": "Sarah",
    "lastname": "Lee",
    "shareable_id": "ABC123XYZ",
    "email": "sarah.lee@example.com",
    "phone": "+123456789",
    "is_verified": true
}
```

Response Fields

Field	Description
first_name	Provider's first name
last_name	Provider's last name
email	Provider's email address
phone	Provider's phone number
shareable_id	Unique ID used by patients to connect
is_verified	Whether the email has been verified

GET /dashboard/

Returns a summary of all patients associated with the authenticated provider, including their latest weight info.

Resource URL: https://api.dryweightwatchers.com/dashboard/

Request Information

- Request Format: None
- Requires Authentication: Yes (Session)

Example Request

```
> GET https://api.dryweightwatchers.com/dashboard/
```

Example Response

- Response Format: JSON (patient array)

Example Response

```
{
    "patients": [
        {
            "id": 101,
            "first_name": "Alice",
            "last_name": "Johnson",
            "email": "alice@example.com",
            "latest_weight": 66.20,
            "latest_weight_timestamp": "2025-03-25T10:23:00Z",
            "prev_weight": 65.91,
            "prev_weight_timestamp": "2025-03-24T08:31:15Z",
            "alarm_threshold": 1.0
        },
        {
            "id": 102,
            "first_name": "Bob",
            "last_name": "Smith",
            "email": "bob@example.com",
            "latest_weight": null,
            "latest_weight_timestamp": null,
            "prev_weight": null,
            "prev_weight_timestamp": null,
            "alarm_threshold": null
        }
    ]
}
```

Response Fields

Field	Description
Array of patients containing the following data	
id	Patient's ID
first_name	Patient's first name
last_name	Patient's last name
email	Patient's email address
latest_weight	Most recent recorded weight (nullable)
latest_weight_timestamp	Timestamp of the most recent weight record
prev_weight	Most recent weight that was not recorded on same day as latest_weight (nullable)
prev_weight_timestamp	Timestamp of prev_weight
alarm_threshold	Weight change threshold for determining whether the patient is at risk

GET /get-patient-data/

Retrieves full medical and profile information for a specific patient associated with the authenticated provider.

Resource URL: https://api.dryweightwatchers.com/get-patient-data/

Request Information

- Request Format: Query String
- Requires Authentication: Yes (Session)
- Query Parameters: id (ID of the patient to retrieve)

Example Request

```
> GET https://api.dryweightwatchers.com/get-patient-data?id=101/
```

Example Response

- Response Format: JSON

Example Response

```
{
    "id": 101,
    "first_name": "Alice",
    "last_name": "Johnson",
    "email": "alice@example.com",
    "latest_weight": 66.2,
    "latest_weight_timestamp": "2025-03-25T10:23:00Z",
    "weight_history": [
        { "weight": 66.2, "timestamp": "2025-03-25T10:23:00Z" },
        { "weight": 65.0, "timestamp": "2025-03-15T09:00:00Z" }
    ],
    "notes": [
        { "id": 1, "note": "Started new diet", "timestamp": "2025-03-
15T10:00:00Z" }
    ],
    "patient_info": {
        "patient": 101,
        "height": 165,
        "date_of_birth": "1990-06-15",
        "sex": "F",
        "medications": "Lasix",
        "other_info": "No known allergies",
        "alarm_threshold": 1.00,
        "last_updated": "2025-03-10T14:00:00Z"
    }
```

```
}
```

Response Fields

Field	Description
id	Patient's ID
first_name	Patient's first name
last_name	Patient's last name
email	Patient's email address
latest_weight	Most recent recorded weight (nullable)
latest_weight_timestamp	Timestamp of the most recent weight record
weight_history	Array of past weight entries
notes	Array of patient notes
patient_info	Object with detailed patient info

weight_history: Array of:

Field	Description
weight	Recorded weight (lbs)
timestamp	Date and time recorded

notes: Array of:

Field	Description
id	Note ID
note	Text content
timestamp	Date and time recordedname

patient_info: Object:

Field	Description
patient	Patient's ID
height	Height in cm
date_of_birth	Date of birth
sex	Biological sex
medications	Current medications (free text)
other_info	Any other clinical notes
alarm_threshold	Weight change threshold for determining whether the patient is at risk
last_updated	Timestamp of last update

POST /add-patient-note/

Adds a clinical note to a specific patient by an authenticated provider.

Resource URL: https://api.dryweightwatchers.com/add-patient-note/

Request Information

- Request Format: JSON
- Requires Authentication: Yes (Session)

Parameter	Description
patient	ID of the patient
note	Text of the note to be added
timestamp	Timestamp of the note (ISO 8601 format)

Example Request

```
> POST https://api.dryweightwatchers.com/add-patient-note/
```

JSON:

```
{
    "patient": 101,
    "note": "Patient has responded well to treatment adjustments.",
    "timestamp": "2025-03-26T14:30:00Z"
}
```

Example Response

- Response Format: JSON

Example Response (Success)

```
{
    "status": 200
}
```

Example Response (Failure)

```
{
    "status": 404,
    "error": "Invalid patient ID",
}
```

Response Fields

Field	Description
status	Status code of authentication
error	Error message if failed

POST /delete-patient-note/

Deletes a clinical note from a specific patient by an authenticated provider.

Resource URL: https://api.dryweightwatchers.com/delete-patient-note/

Request Information

- Request Format: JSON
- Requires Authentication: Yes (Session)

Parameter	Description
note_id	ID of the note to be deleted

Example Request

```
> POST https://api.dryweightwatchers.com/delete-patient-note/
```

JSON:

```
{
    "note_id": 101,
}
```

Example Response

- Response Format: JSON

Example Response (Success)

```
{
    "status": 200,
    "message": "Note deleted successfully"
}
```

Example Response (Failure)

```
{
    "status": 404,
    "message": " Note not found"
}
```

Response Fields

Field	Description
status	Status code of authentication
error	Error message if failed
Message	Success message when note is deleted

POST /add-patient-info/

Adds or updates detailed medical information for a specific patient. Only accessible by authenticated providers.

Resource URL: https://api.dryweightwatchers.com/add-patient-info/

Request Information

- Request Format: JSON
- Requires Authentication: Yes (Session)

Parameter	Description
patient	ID of the patient
height	Patient's height (in cm) (optional)
date_of_birth	Date of birth (YYYY-MM-DD) (optional)
sex	Biological sex (e.g., "M", "F") (optional)
medications	Medications the patient is taking
other_info	Additional notes
alarm_threshold	Weight change threshold for determining whether the patient is at risk

Example Request

```
> POST https://api.dryweightwatchers.com/add-patient-info/
```

JSON:

```
{
    "patient": 101,
    "height": 165,
    "date_of_birth": "1990-06-15",
    "sex": "F",
    "medications": "Amlodipine",
    "other_info": "History of hypertension",
    "alarm_threshold": 1.5
}
```

Example Response

- Response Format: JSON

Example Response (Success)

```
{
    "status": 200
}
```

Example Response (Failure)

```
{
    "status": 404,
    "error": "Invalid patient ID",
}
```

Response Fields

Field	Description
status	Status code of authentication
error	Error message if failed

GET /get-provider-notifications/

Retrieves all notifications assigned to the authenticated provider, ordered by most recent. Only accessible by authenticated users with a provider role.

Resource URL: https://api.dryweightwatchers.com/get-provider-notifications/

Request Information

- Request Format: None
- Requires Authentication: Yes (Session)

Example Request

```
> GET https://api.dryweightwatchers.com/get-provider-notifications/
```

Example Response

- Response Format: JSON (array of notifications)

Example Response (Success)

```
{
    "status": 200,
    "notifications": [
        {
            "message": "Patient John Doe has exceeded weight
threshold",
            "created_at": "2025-03-28T15:40:18.000Z",
            "is_read": false,
            "id": 42
        },
        {
            "message": "New patient assigned",
            "created_at": "2025-03-27T13:21:00.000Z",
            "is_read": true,
            "id": 39
        }
    ]
}
```

Example Response (Failure)

```
{
    "status": 404,
    "error": " Only providers can access this",
}
```

Response Fields

Field	Description
status	Status code of authentication
message	Notification text
created_at	ISO 8601 timestamp when the notification was created
is_read	Boolean indicating if the notification has been read
id	Unique ID of the notification
error	Error message if unauthorized or invalid

POST /mark-notification-as-read/

Marks a specific notification as read by ID. Only accessible by authenticated providers.

Resource URL: https://api.dryweightwatchers.com/mark-notification-as-read/<id>/

Request Information

- Request Format: JSON or form-data (body not required)
- Requires Authentication: Yes (Session)
- Path Parameter: id: Integer ID of the notification to be marked as read

Example Request

```
> POST https://api.dryweightwatchers.com/mark-notification-as-read/42/
```

Example Response (Success)

```
{
    "status": 200,
    "message": "Marked as read"
}
```

Example Response (Failure)

```
{
    "status": 404,
    "error": " Notification not found",
}
```

Response Fields

Field	Description
status	Status code of authentication
message	Confirmation that the notification was marked
error	Error message if the notification was not found

Appendix B: Team Collaboration Roles

Jonathan Hopkins

Contact information:

- School email: jhopkins63@gatech.edu
- Personal email: j.miyagi.hopkins@gmail.com

Responsible for:

- Task breakdowns for sprints in Jira, project management
- AWS Account setup
- Design Document:
    - Introduction
    - Terminology
    - System Architecture
    - Data Storage Design
- Code-related:
    - Logic and page for providers to create an account
    - Implement CSRF tokens for authentication
    - Setting rate limits for login attempts for both patient and providers
    - Logic and design for providers to view their sharable id.
    - Design and logic for the provider dashboard to view all of their patients
    - Weight threshold setup, modification and visualization on the provider's website
    - Provider timestamped notes on patient's weight record (add, edit and delete functionality)
    - Patient information on provider's website (add, edit and delete functionality)

Duy Nguyen

Contact information:

- School email: dnguyen409@gatech.edu
- Personal email: dnguyen9074@gmail.com

Responsible for:

- Main point of contact with client
- Preparing meeting agendas
- Code-related:
    - Patient interface navigation structure
    - Logic and page for patients to create an account
    - Patient's page and logic to record their daily weight
    - Reminder's logic (add, delete, edit) and page design for patients
    - Patient weight visualization in the dashboard in the patient app

    - Patient weight visualization in the patient details page in the provider website
    - Allow patient email notifications
    - Change notification preferences for providers
    - Text notifications for providers

Nathan Fritchley

Contact information:

- School email: nfritchley3@gatech.edu
- Personal email: natefritchley@gmail.com

Responsible for:

- Taking meeting minutes
- Code-related:
    - Patient login logic and page design
    - Add provider on patient's account by using their unique sharable id
    - Automatically save patient's login logic and automatically log in when app is opened
    - Patient's page to view their providers
    - Patient's ability to remove one of their providers
    - The logic to delete an account for both providers and patients.
    - Allow for deleted information to be kept in database in a "deactivated" table
    - Change notification preferences for patients
    - Email notification for patients

Lara Bailen

Contact information:

- School email: lbailen3@gatech.edu
- Personal email: larabailen.lb@gmail.com

Responsible for:

- Submitting assignments once completed
- Logo Design
- Server setup in AWS (Hosting for Django Server and Hosting for provider React Frontend)
- Design Document:
    - Component Design
    - UI Design
    - Appendices
- Code-related:
    - Home pages for both provider website and patient app
    - Add Provider page for patients
    - Provider login logic and page design

- Patient and provider logout implementation
- Patient details page for providers
- Allow providers to remove patients from their account
- Email verification setup for providers
- Provider email notifications for drastic weight change
- UI improvements for provider’s website