SANTA CLARA UNIVERSITY

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Date: May 31, 2024

I HEREBY RECOMMEND THAT THE THESIS PREPARED UNDER MY SUPERVISION BY

Madeleine Waldie

ENTITLED

The Journey to Desensitization: A Mobile App for Oral Immunotherapy Patients

BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

BACHELOR OF SCIENCE IN COMPUTER SCIENCE AND ENGINEERING

_____

Thesis Advisor

_____

Department Chair

The Journey to Desensitization: A Mobile App for Oral Immunotherapy Patients

by

Madeleine Waldie

Submitted in partial fulfillment of the requirements
for the degree of
Bachelor of Science in Computer Science and Engineering
School of Engineering
Santa Clara University

Santa Clara, California
May 31, 2024

The Journey to Desensitization: A Mobile App for Oral Immunotherapy Patients

Madeleine Waldie

Department of Computer Science and Engineering
Santa Clara University
May 31, 2024

ABSTRACT

Millions of individuals in the United States confront the daily challenges of food allergies, with the threat of potentially life-threatening reactions ever-present. Oral Immunotherapy (OIT) emerges as a beacon of hope in this landscape, gradually desensitizing patients to their allergens. This groundbreaking treatment, contrasting avoidance strategies, promises a life with fewer restrictions. However, it is a journey fraught with complexities, requiring rigorous adherence to protocols, emotional fortitude, and frequent medical supervision for patients. This thesis centers on the creation of an iPhone application tailored to OIT patients, leveraging Apple's privacy and security features. The app integrates seamlessly with the Apple Health app, providing a platform for dose tracking, symptom logging, and the visualization of long-term trends. As a comprehensive resource, it educates users about anaphylaxis and OIT. This project seeks to address the unmet needs of OIT patients, offering vital support, guidance, and resources, ultimately enhancing the quality of life for those grappling with food allergies.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Background

Every day, approximately 32 million Americans grapple with the challenges of food allergies [1]. A staggering 10 percent of the population finds itself on the precipice of a potentially life-threatening reaction with every meal.

Living with food allergies is a relentless battle – a constant state of vigilance. The mere act of eating transforms into a high-stakes game of Russian roulette, where the wrong choice can lead to dire consequences. Exclusion from social activities, bullying, and the looming specter of emergency room visits become haunting hallmarks of life for those afflicted [2]. Having lived with life-threatening, airborne food allergies my entire life, I know firsthand what this is like. In the first six months of my junior year of high school, I had over thirty-two allergic reactions, received seven epinephrine injections, was evacuated from school by ambulance four times, and was hospitalized once. The physical and emotional toll of living like this is immeasurable.

However, there is a glimmer of hope on the horizon – oral immunotherapy. Oral immunotherapy (OIT) is a medical treatment that gradually exposes a person to increasing amounts of an allergen to desensitize them to it. The goal of OIT is to help people with food allergies develop a tolerance to an allergen so that they can safely consume small amounts of it without experiencing an allergic reaction [3]. This groundbreaking approach represents a paradigm shift in the treatment of food allergies. Unlike the decades-old strategy of avoidance [4], oral immunotherapy offers the prospect of desensitizing patients to their allergens. By reaching a maintenance dose, a new level of protection allows them to be near, touch, or even consume foods they were previously forced to shun. This revolutionary approach has shown remarkable promise in clinical trials, raising the possibility of a life with fewer dietary restrictions and reduced fear of accidental allergen exposure.

Despite its immense potential, oral immunotherapy is not without its complexities and uncertainties – especially for the patient. It requires rigorous adherence to protocols, frequent medical supervision, and the emotional fortitude to face one's allergens head-on – not to mention, patients frequently experience side effects when updosing (i.e., increasing the intake of their allergen), such as itchiness or vomiting, and risk more severe complications, like anaphylaxis.

As with any medical treatment, patients need to log their doses and symptoms to keep track of their progress. But, for many, this is a difficult, impractical task. The magnitude of this challenge is evident in my three-inch binder filled with over 1,600 handwritten logs from my five-year OIT journey. And this binder will only grow, as I continue to take my maintenance dose of OIT for the rest of my life.

The lack of accessible tools to support patients throughout this journey is glaring, and it is within this context that the need for an innovative solution emerges. The overarching problem addressed by this thesis lies in the unmet needs of oral immunotherapy patients. These individuals require comprehensive support, guidance, and resources to navigate the intricate path toward desensitization safely and effectively.

1.2 Project Goal

This project primarily focuses on the design and development of an iPhone application for OIT patients. By opting for the iOS platform, I aim to leverage Apple's robust privacy and security features, ensuring the safe storage of sensitive information. A key feature of the app will be its integration with the Apple Health app, allowing users to seamlessly view their oral immunotherapy data with other health metrics such as exercise and heart rate, all of which can significantly influence treatment outcomes. Integration with the Apple Health app will allow patients to easily, and securely, share their OIT information with their doctors.

Not only will the app facilitate daily dose tracking and symptom logging, but also it will provide the visualization of long-term trends, providing users with valuable insights into their progress. It will also serve as an educational resource, offering essential information about anaphylaxis and oral immunotherapy. Ultimately, my goal for this app is for it to become an accessible, valuable tool for anyone navigating this stressful, yet life-changing process.

Chapter 2

Project Requirements

2.1 Functional and Nonfunctional Requirements

In the development of any software application, defining clear and comprehensive requirements is essential to guide the design and development process. These requirements are typically categorized into two main types: functional and non-functional requirements. The following sections will dive into what, exactly, these two types of requirements entail, and what requirements have been identified for my project.

2.1.1 Functional Requirements

Functional requirements describe the specific features and functionalities that an application must possess to meet the needs of its users. For my application, which focuses on OIT and aims to provide a comprehensive and user-friendly experience, several functional requirements have been identified.

First, I've identified the following as critical functional requirements. Critical requirements are the highest priority and must-have features or functionalities. These are non-negotiable elements that are essential for the core functionality and success of the application. Failure to implement critical requirements would severely impact the application's usability or safety.

- **Integration with Apple Health App:** The application will seamlessly integrate with the Apple Health app, allowing it to retrieve and display health metrics relevant to OIT, such as exercise and heart rate.
- **Data Logging and Tracking:** Users will be able to log their daily OIT doses and associated symptoms. The app will also enable users to input and track their treatment progress over time.

Next, I've identified the following as necessary functional requirements, or important features and attributes that are required for the software to function effectively and provide a satisfactory user experience. While not as vital as critical requirements, necessary requirements are still essential for the software to fulfill its intended purpose and meet

user expectations. They are the second-highest priority and are addressed immediately after critical requirements, and are crucial for the software's overall functionality and user satisfaction.

- **Data Visualization:** To assist users in understanding their progress, the application will provide graphical representations of long-term trends in OIT progress. These charts and graphs will be interactive and easy to interpret.
- **Data Sharing:** Users will have the capability to securely share their OIT data with healthcare providers through Apple Health, ensuring efficient communication between patients and professionals.

Finally, I've identified the following optional requirements. Also known as "nice-to-have" or "enhancement" features, these are functionalities or attributes that are not essential for the core functionality of the software and are the lowest priority to implement.

- **Educational Resources:** The application will offer educational content on essential topics like anaphylaxis and oral immunotherapy, including articles and other educational materials to empower users with knowledge.
- **Notifications:** The application will send reminders and alerts to users for taking their doses and logging symptoms, with customizable notification settings to accommodate individual preferences.

2.1.2 Nonfunctional Requirements

Nonfunctional requirements, on the other hand, focus on the qualities or attributes of the application that enhance its overall performance and user experience. These requirements ensure the application functions effectively and efficiently while adhering to certain standards and guidelines.

The following are critical nonfunctional requirements that have been identified for my application. These are the highest priority nonfunctional requirements to implement, and will thus be implemented first.

- **Compliance:** Adherence to Apple's App Store guidelines and healthcare data privacy regulations, such as HIPAA, is crucial to avoid legal and reputational risks.
- **Compatibility:** The application must be compatible with a wide range of iPhone models and iOS versions, optimizing it for different screen sizes and resolutions.
- **User Interface (UI) and User Experience (UX):** An intuitive, user-friendly interface with a consistent and aesthetically pleasing design is imperative to enhance user satisfaction.

Next, the necessary nonfunctional requirements are as follows.

- **Reliability:** The app should function reliably without frequent crashes or errors. Implementing error handling and reporting mechanisms helps ensure a seamless user experience.
- **Performance:** The application must be responsive, providing a smooth user experience with minimized load times for data and visual elements to ensure users are not kept waiting.

Finally, the optional nonfunctional requirements are listed below.

- **Accessibility:** Accessibility for users with disabilities is a non-negotiable requirement, necessitating compliance with accessibility guidelines and standards.
- **Scalability:** The application's architecture could be designed to accommodate potential growth in the user base, ensuring it can handle increased data and user loads gracefully.

2.2 Use Cases

After defining the nonfunctional and functional requirements, the next natural step is a comprehensive analysis and visualization of the system's functionalities. This section delves into the Use Case Tables and Unified Modeling Language (UML) diagrams that serve as the blueprint for my envisioned app. These diagrams are instrumental in illustrating various interactions between users and the application, offering a systematic representation of the app's behavior, structure, and relationships. The Use Case Tables and UML diagrams dive into the specific scenarios in which users engage with the application, outlining the goals and interactions within each use case.

UC-01	**Initial App Onboarding**	
Dependencies	• User has never opened the app before	
Description	The system will behave in the following use case when the user initially opens the app.	
Precondition	The user has downloaded the app from the app store, and goes to use it.	
Ordinary Sequence	**Step**	**Action**
	1	Application displays a screen, with a button to get started
	2	User taps the “Get Started” button

	3	Application displays a screen with basic information for the user to fill out:
		• Name
		• Birthdate
		• Allergens
		• Preferences
		  ○ Share Data with Apple Health
		  ○ Protect Application with FaceID

	4	User fills out the necessary information:
		• Name
		• Birthdate
		• Allergens
		• Preferences
		  ○ Share Data with Apple Health
		  ○ Protect Application with FaceID

	5	User taps “Get Started” button
	6	Application saves information, sets a flag that the user has opened the application before, and then displays the tabbed application.
Postcondition	The user will be able to use the application.	
Exceptions	**Step**	**Action**
	6	If the user hasn’t filled out all of the necessary information, then an alert will be displayed to complete the information, and the user won’t be able to proceed until everything’s filled out.
Comments	The information entered here will be displayed in the profile section of the app, and will be editable.	

Table 2.1: Use Case 01: Initial App Onboarding

UC-02	**Sharing Data with Apple Health**	
Dependencies	• User hasn't enabled sharing data with Apple health yet	
Description	The system will behave in the following use case when the user enables sharing data with Apple health	
Precondition	The user is going through the setup page.	
Ordinary Sequence	**Step**	**Action**

	1	Application displays a screen with basic information for the user to fill out:
		• Name
		• Birthdate
		• Allergens
		• Preferences
		  ○ Share Data with Apple Health
		  ○ Protect Application with FaceID

	2	User taps the "Share Data with Apple Health" toggle, toggling it on
	3	Application displays a pop up sheet, asking the user if they would like to share data with the Apple Health app. It will list out every symptom / thing that it will be reading from Apple Health / writing to Apple Health.
	4	User goes through all of the options and decides whether or not they would like to share that data, tapping on the toggles.
	5	User taps "Done"
Postcondition	The user will have all of the selected data shared with Apple Health.	
Exceptions	**Step**	**Action**
	0	If the user has already gone through the onboarding process, they can edit their permissions in the Profile section, and then follow the ordinary sequence.
Comments	The user doesn't have to select all of the categories, if they're uncomfortable sharing that information.	

Table 2.2: Use Case 02: Sharing Data with Apple Health

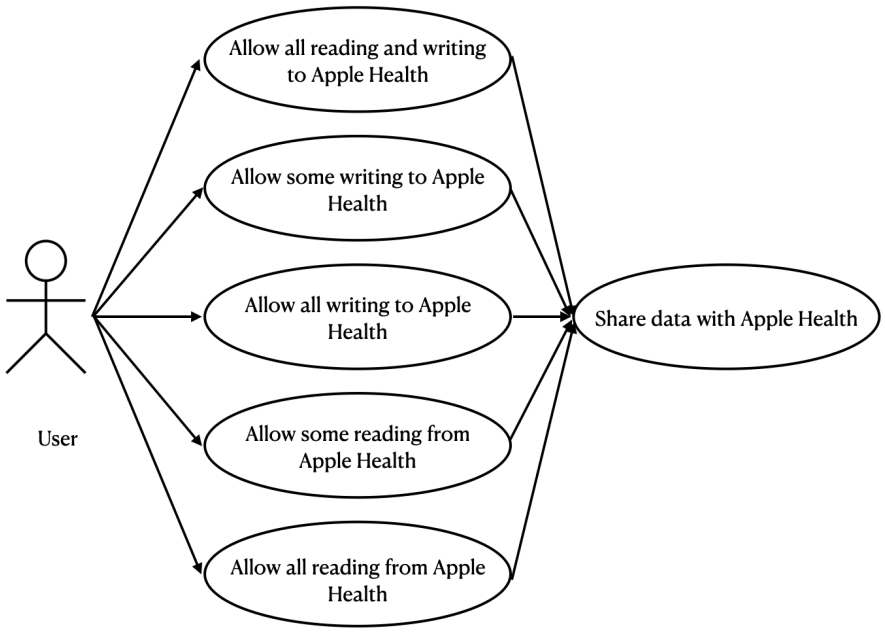


Figure 2.1: Use Case 02: Visual Diagram

UC-03	**Adding Allergen to Profile**	
Description	The system will behave in the following use case when the user adds an allergen to their profile.	
Precondition	The user is going through the setup page.	
Ordinary Sequence	**Step**	**Action**
	1	Application displays a screen with basic information for the user to fill out: ● Name ● Birthdate ● Allergens ● Preferences ○ Share Data with Apple Health ○ Protect Application with FaceID
	2	User taps the plus to add an allergen
	3	Application displays a pop up of possible allergens to add
	4	User taps an allergen
Postcondition	The user will have the allergen(s) they selected listed in their profile	
Comments	The user can add multiple allergens. There's also an other option, where the user can add an allergen that isn't in the list.	

Table 2.3: Use Case 03: Adding Allergen to Profile

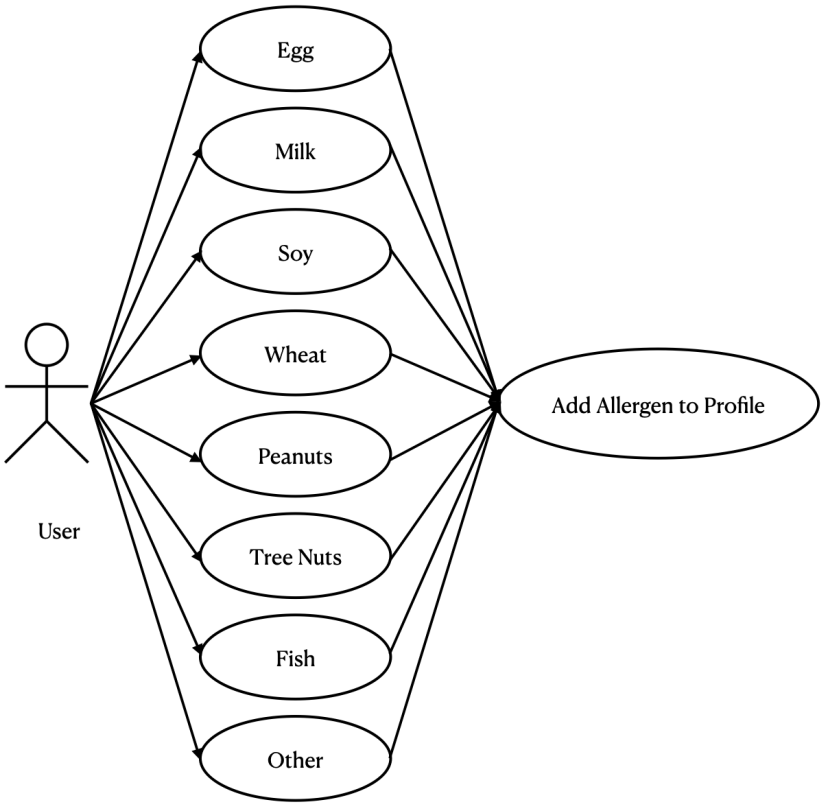


Figure 2.2: Use Case 03: Visual Diagram

UC-04	**Log a Dose Taken**	
Dependencies	• Allergens patient is doing OIT for • List of possible doses patient could take for each allergen • Date of dose	
Description	The app will behave in the following use case when the user goes to log a dose.	
Precondition	The patient has taken a dose and wants to log it, so they open their app, which opens to the "Home" tab.	
Ordinary Sequence	**Step**	**Action**
	1	User clicks the "Log Dose" button
	2	Application displays a sheet, with fields for the user to answer the following information: • Allergens in the dose • Time of dose • Any medications taken before the dose • Dosage of each allergen • Notes
	3	User inputs information for each of the following fields: • Time of dose • Any medications taken before the dose • Dosage of each allergen • Notes
	4	User taps "Done button
	5	Application saves dose information in its database, for the given date of the dose
	6	Application displays dose information that was logged
Postcondition	The user can see their dose was logged, and can quickly view the dose information for the day.	
Exceptions	**Step**	**Action**
	0	If the user wants to log a dose for a different day (not today), then they will select a different day from the calendar at the top of the Home screen. Then, they will follow the ordinary sequence.
	2	If the user has already logged a dose for that day, the sheet that the application displays will have the appropriate dose information filled out from the database. Then, the user can edit the information if needed and follow the rest of the ordinary sequence.
Comments	Only one dose can be logged per day.	

Table 2.4: Use Case 04: Log a Dose Taken

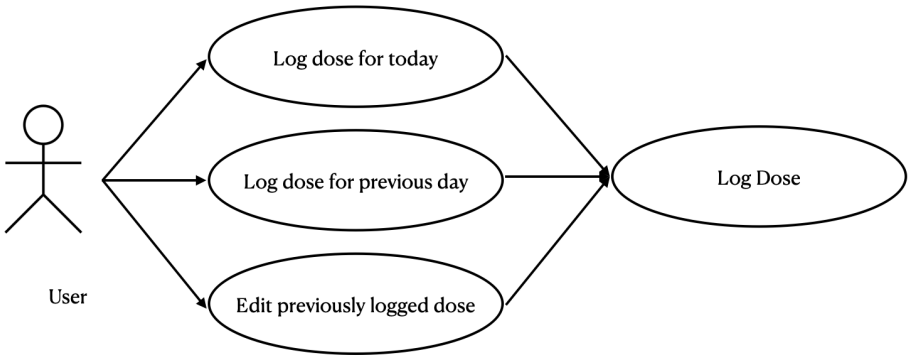


Figure 2.3: Use Case 04: Visual Diagram

UC-05	**Log Symptoms for Dose Taken**	
Dependencies	• Date of dose • List of possible symptoms	
Description	The app will behave in the following use case when the user goes to log symptoms they have experienced.	
Precondition	The patient wants to log symptoms they're experiencing after taking a dose, so they open the app.	
Ordinary Sequence	**Step**	**Action**
	1	User taps "Log your Symptoms" button
	2	Application displays a sheet, with a list of possible symptoms for the user to choose from
	3	User chooses symptoms they're experiencing from the list
	5	User taps "Done" button
	6	Application saves symptom information for the given day
	7	Application displays symptom information that was logged
Postcondition	The user can see their symptoms were logged, and can quickly view their symptom information for the day.	
Exceptions	**Step**	**Action**
	0	If the user wants to log symptoms for a different day (not today), then they will select a different day from the calendar at the top of the Home screen. Then, they will follow the ordinary sequence.
	2	If the user has already logged symptoms for that day, the sheet that the application displays will have the appropriate symptom information filled out from the database. Then, the user can edit the information if needed and follow the rest of the ordinary sequence.
Comments	The user doesn't need to log a dose in order to log symptoms, as patients can experience symptoms over time, and it is important to allow them to log symptoms at any time.	

Table 2.5: Use Case 05: Log Symptoms for Dose Taken

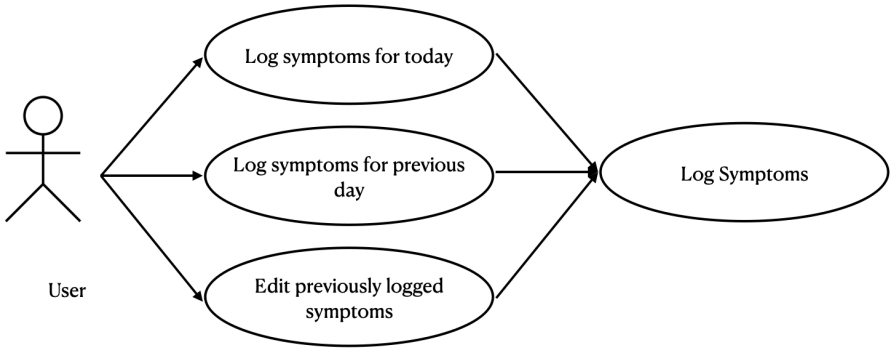


Figure 2.4: Use Case 05: Visual Diagram

UC-06	**Add a Dosage for an Allergen**	
Dependencies	• User has one or more allergens listed in their profile	
Description	The system will behave in the following use case when the user wants to create a new dosage for an allergen.	
Precondition	The patient has received a new dosage for their OIT from their doctor, and wants to add it to the app.	
Ordinary Sequence	**Step**	**Action**
	1	User taps the “Dosing” tab of the application
	2	Application switches from the “Home” tab to the “Dosing” tab, which displays a list of possible doses for each allergen
	3	User taps the plus button to add a new dosage
	4	Application displays a sheet, with the following input fields for a new dose: • Allergen • Name of Dose • Dosage • Notes
	5	User enters the necessary information: • Allergen • Name of Dose • Dosage • Notes
	6	User taps “Done” button
	7	Application saves the new dose for the given allergen, and then closes the sheet
	8	Application displays the new dosage in the “Dosing” tab, under the appropriate allergen. The application also displays a half dose for that dose.
Postcondition	The user can see their dose was added to the list of possible doses.	
Exceptions	**Step**	**Action**
	5	If the user enters a dose with a name that already exists, the application will display an alert asking the user to choose a name that doesn’t already exist.
Comments	Allergens can have multiple doses listed in the dosing tab, as users move from dose to dose while going through OIT and may have to go back doses if complications arise.	

Table 2.6: Use Case 06: Add a Dosage for an Allergen

UC-07	**View Trends**	
Dependencies	• Doses have been logged • User allows connection with the Apple Health app	
Description	The system will behave in the following use case when the user wants to view trends for their symptoms.	
Precondition	The user has opened the application.	
Ordinary Sequence	**Step**	**Action**
	1	User taps the “Insights” tab of the application
	2	Application gathers symptom and dose data and analyzes it
	3	Application displays graphs highlighting trends in symptoms and doses
Postcondition	The user can see the trends of their symptoms.	
Exceptions	**Step**	**Action**
	2	If the user has never logged any information, the Insights tab will have a message, telling the user insights graphs will show up once they have logged doses and symptoms.
Comments	The type of insights information displayed will depend on the information logged.	

Table 2.7: Use Case 07: View Trends

UC-08	**Open Article**	
Description	The system will behave in the following use case when the user wants to view one of the resource articles.	
Precondition	The user wants to learn more about the signs and symptoms of anaphylaxis.	
Ordinary Sequence	**Step**	**Action**
	1	User taps the “Education” tab of the application
	2	Application displays a list of rich link previews, linking to articles that the user can click on
	3	User taps on an article about anaphylaxis
	4	Application opens a sheet, with information about the signs and symptoms of anaphylaxis
Postcondition	User reads the article, learning new information	
Comments	Tapping on different articles will show different information.	

Table 2.8: Use Case 08: Open Article

UC-09	Navigating the Calendar View	
Description	The system will behave in the following use case when the user wants to see dose / symptom information for a day in the past.	
Precondition	The user has logged some doses over the past week, and wants to look at them again.	
Ordinary Sequence	**Step**	**Action**
	1	User taps the calendar button in the top right corner of the application
	2	Application shows a calendar view with the current month
	3	User taps a different day in the past month
	4	Application switches the date, and shows the dose and symptoms associated with that date
Postcondition	The user can see the dose and symptom data for the date they're interested in.	
Exceptions	**Step**	**Action**
	1	If the user doesn't want to use the monthly calendar view, they can also scroll through the week view until they find the day they're interested in, and then continue from step 3.
Comments	The user won't be allowed to select dates in the future, as they could not have taken a dose in the future.	

Table 2.9: Use Case 09: Navigating the Calendar View

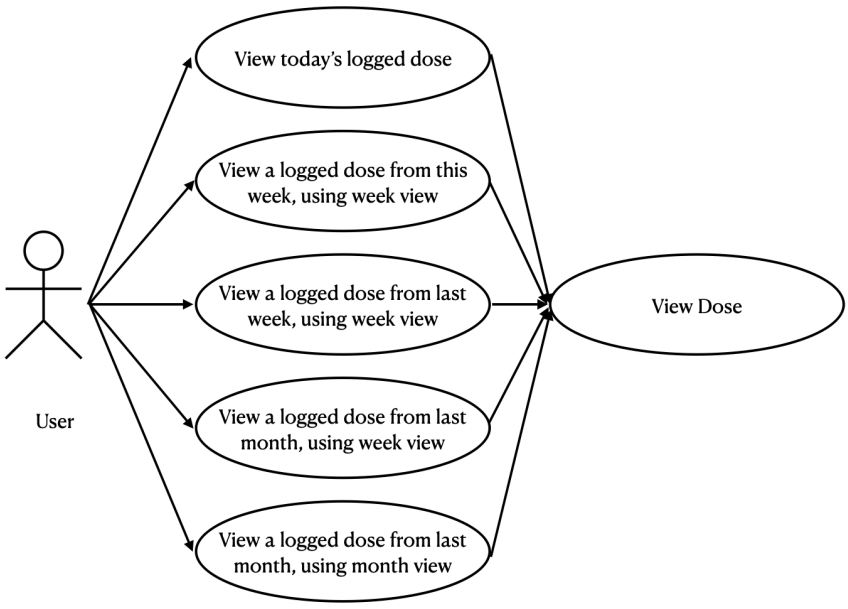


Figure 2.5: Use Case 09: Visual Diagram 1

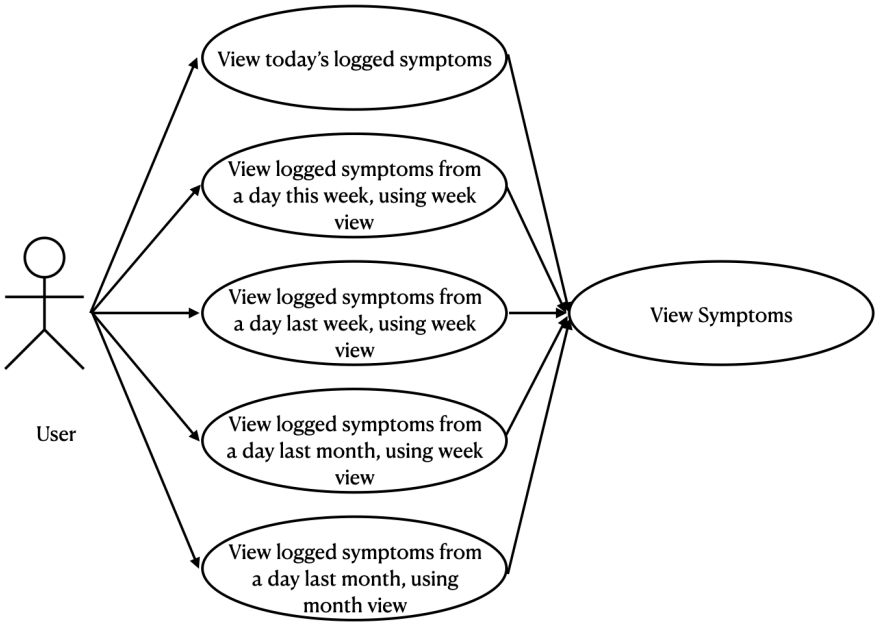


Figure 2.6: Use Case 09: Visual Diagram 2

Chapter 3

Research

While I know a lot about OIT from my own experiences, it was important that I thoroughly researched the topic before creating my initial design – especially because OIT is a new, emerging treatment and isn't standardized, so it's different from clinic to clinic. This chapter outlines my research on OIT, shedding light on its definition and mechanisms, as well as the initial advice I received from allergists and immunologists in the field.

3.1 Research

3.1.1 Background

Oral Immunotherapy has exhibited remarkable efficacy in addressing a spectrum of food allergies, including peanuts, tree nuts, milk, and eggs. The impact on patients' quality of life is profound, offering hope and tangible relief. As revealed by recent studies, the success of OIT is particularly notable in young patients, especially those under three years old. For example, a study found that young patients demonstrate the highest success rates in OIT, showcasing the treatment's potential for transformative outcomes in this age group [6].

The mechanistic underpinnings of OIT unfold within the intricate interactions of the immune system. Research indicates that carefully administered doses of allergens through OIT orchestrate a controlled immune response [8]. This calibrated exposure triggers the production of regulatory T cells and other immune modulators, facilitating a shift from hypersensitivity to a more tolerant state. Such immune modulation is crucial for the success of OIT, and understanding these processes informs the app design to ensure patients comprehend the science behind their treatment.

3.1.2 The Patient's Journey

The OIT patient journey is a nuanced process, beginning with a meticulous selection phase that takes into account factors such as age, allergy severity, and overall health. Recent research has highlighted the critical role of comprehensive assessments and diagnosis in guiding a tailored treatment plan. For instance, a study found that the success of OIT is intricately linked to understanding the specific allergens triggering reactions in individual patients [9]. This

research underscores the importance of incorporating detailed allergy testing and evaluations into the app to guide users through a personalized OIT journey.

The carefully designed OIT protocol involves a gradual introduction of allergens in controlled doses, with customization based on individual needs. Data from ongoing studies emphasizes that adapting the frequency and intensity of allergen doses to individual responses is essential for the efficacy of OIT. Incorporating this insight into the app ensures that users have a clear understanding of the gradual exposure process, fostering adherence and positive engagement.

3.1.3 Predosing Medications and Precautions

Predosing medications play a pivotal role in preparing patients for controlled allergen exposure during OIT. Recent findings, as seen in various clinical studies, underscore the importance of this precautionary step. For example, research suggests that administering medications, such as antihistamines, mitigates the risk of immediate allergic reactions, enhancing the overall safety profile of OIT [10]. This crucial insight informs the app's design to include features that remind users about predosing medications and educate them on their significance in ensuring a smooth OIT progression.

The close monitoring of patient responses to predosing medications, as supported by research, further emphasizes the adaptability of treatment plans to individual sensitivities. Integrating tools in the app that allow users to track and report their responses ensures a more personalized OIT experience, aligning with the variability observed in patient reactions.

3.1.4 Adherence to Strict Rules and Lifestyle Adjustments

The path to successful Oral Immunotherapy involves a commitment to adherence to strict rules and lifestyle adjustments. Patients partaking in OIT are often advised to refrain from vigorous exercise after each session and abstain from alcohol during the treatment period. Research has demonstrated that these seemingly restrictive measures are integral to the safety and efficacy of the treatment [9].

For instance, ongoing studies have shown that adherence to guidelines significantly influences the success of OIT, reducing the risk of adverse reactions. This underscores the personalized nature of OIT, emphasizing the need for patients to actively engage in their own care. The app design takes inspiration from this research, incorporating features that not only educate users on the importance of adherence but also provide tools for tracking and maintaining lifestyle adjustments.

3.1.5 Highly Personalized Treatment Plans

A key takeaway from the research is the highly personalized nature of OIT treatment plans. Recent studies have emphasized that the success of OIT lies in recognizing the uniqueness of each patient's journey. Treatment plans are meticulously crafted based on factors such as age, medical history, specific allergens, and overall health. This tailored approach allows for adjustments in allergen dose frequency and intensity, ensuring effective and well-tolerated progress.

For example, ongoing research has shown that individualized treatment plans significantly contribute to positive outcomes in OIT. Patients progress through therapy at a pace that is both effective and well-tolerated, reducing the risk of adverse reactions. The individualized nature of OIT reflects its commitment to providing bespoke solutions, acknowledging the diversity of allergic conditions and accommodating variations in the treatment process. The app, inspired by this research, prioritizes customization, offering users a personalized roadmap for their OIT journey.

3.2 Consultation with Immunologists

After thoroughly researching OIT, the natural next step in my research and design process was to bring my app idea to doctors, and garner their advice. I collaborated with top immunologists from The Children's Hospital of Philadelphia, Mt. Sinai Hospital, and Stanford Medicine to ensure that my application aligns with expert advice and industry best practices. Here are some key pieces of advice I received:

- **Pop Up to Seek Medical Care:** Immunologists recommended displaying a pop-up notification encouraging users to "Call your doctor" or "Call 911" if any aspect of their symptoms raises concerns.
- **Engagement through Notifications:** Immunologists emphasized the importance of subtle reminders, such as "Did you take your dose today?" These notifications not only promote adherence to treatment plans but also serve as a gentle nudge towards consistent health practices.
- **Emergency Preparedness:** Immunologists advised incorporating an innovative feature allowing users to upload images of their emergency action plans to the 'Resources' section. This practical addition ensures that critical information is readily accessible, fostering a sense of empowerment in navigating unforeseen health scenarios.
- **Educational Prompts:** In the realm of education, immunologists recommended maintaining a delicate balance to avoid offering explicit medical advice. Instead, the app should gently prompt users with thought-provoking questions like, "Did you ask your doctor about this? Or that?" This approach encourages informed discussions with healthcare providers, fostering a proactive and engaged patient community.
- **Autoinjector Management:** Recognizing the prevalence of autoinjectors in allergy management, immunologists suggested a dedicated section providing functionality to check the expiration status of these crucial devices.

This integration aligns with the app's commitment to supporting users in maintaining the efficacy of their medical tools.

- **Individualization:** Immunologists highlighted the importance of individualization, guiding the development process with an unwavering commitment to recognizing the distinct needs of every user. This approach acknowledges the nuanced nature of healthcare management.

- **Gamification Elements:** To foster user motivation, immunologists recommended embracing an innovative approach by incorporating gamification elements. This is something I hadn't thought of adding, but if I had time, would love to incorporate it into my application.

In essence, the collaborative efforts with renowned immunologists have shaped an app that transcends conventional health management tools. By embracing flexibility, individualization, and innovative features, this application aspires to be not just a companion in healthcare but a personalized guide, empowering users to navigate their health journeys with confidence and informed decision-making.

Chapter 4

Design Overview

4.1 General Design

When developing this application, the user experience was at the forefront of my mind, guiding my design decisions.

I decided to design my application as a tab-based application, ensuring straightforward navigation and user-friendly interaction. This approach allows users to easily access different sections of the app, keeping the user experience organized and intuitive. I also decided to place a strong emphasis on minimalistic design principles to create an uncluttered and efficient interface. By stripping away unnecessary elements and focusing on essential functionalities, the app becomes user-centric and easy to use.

For the color scheme, shown in Figure 4.1, I drew inspiration from the concept of cool colors. Cool colors, with their calming and relaxing properties, are chosen to evoke a sense of tranquility and ease within the app, ensuring users feel at ease and comfortable during their interaction with the platform [5].

Figure 4.1: Color Scheme

In addition, I aim to provide a design that is inclusive and accessible to all users. To achieve this, I carefully considered contrast and color choices, making sure they were tested against colorblind scales to ensure that the content was easily distinguishable and readable by a wide range of individuals.

To add a friendly and inviting touch to the app, I incorporated images and graphics that resonate with a warm and approachable feel. These images are thoughtfully chosen to create a connection with users and enhance their engagement with the app, contributing to a positive and enjoyable user experience.

In the following sections, you will see my initial prototype designs.

4.2 Onboarding View

The Onboarding Vew, shown in Figure 4.2, consists of screens designed to introduce users to the app's key features and benefits. This view is essential for guiding new users and helping them get acquainted with the platform. The screens are simplistic and visually appealing, setting a positive tone for the user's journey.

The Set Up form asks the user to input key information, such as their name, birthdate, and allergens, and asks the user if they will allow the application to share data with Apple Health. This is a crucial step in the onboarding process, as without allowing access, the user will not have access to key insights and a tailored app experience.

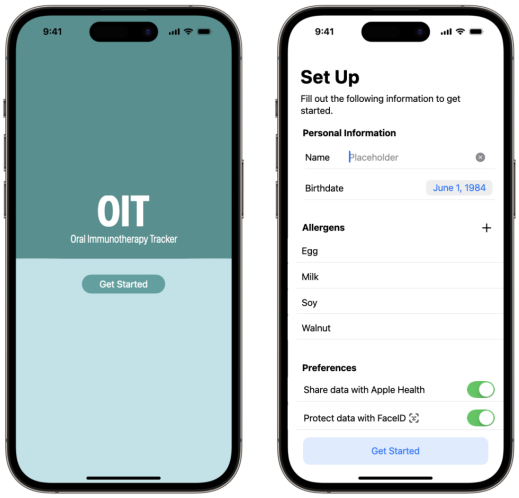


Figure 4.2: Onboarding Screens

4.3 Today Tab

The Today Tab serves, shown in Figure 4.3, as the core of the app, where users view and log dose and symptom information for the selected date. The design prioritizes clarity and easy navigation, ensuring users can quickly find the information they need. A sliding weekly calendar view allows the user to select a date, or the month button in the top right corner allows the user to switch to a monthly view.

Additionally, the user can navigate to their profile from the Today Tab, where they can easily update their information, such as their allergens.

Figure 4.3: Today Tab Screens

4.4 Dose and Symptom Pop-Ups

The Dose and Symptom Pop-Ups, shown in Figure 4.4, are opened when the user goes to log a dose or symptom in the Today tab. These pop-ups are designed to be user-friendly, with helpful graphics, and to make it take minimal time to log a dose or symptom.

In addition to allergens and dosages, the Dosing Pop-Up will collect other important information, such as time of dose and whether the patient has pre-dosed with any medications. And, the symptoms listed in the Symptoms Pop-Up will directly correlate with HealthKit symptom types to ensure ease of Health App reporting.

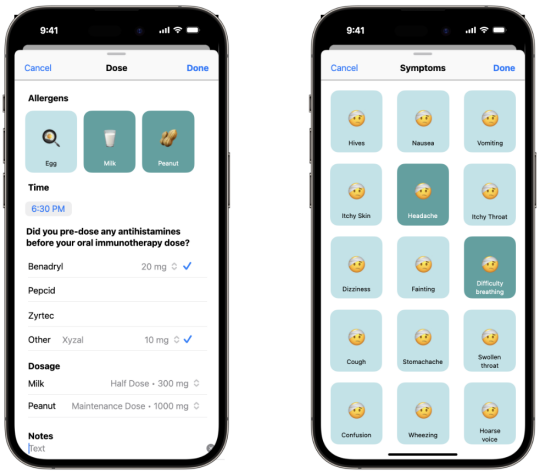


Figure 4.4: Dose and Symptoms Pop-Ups

4.5 Insights Tab

The Insights Tab, shown in Figure 4.5, serves as a place for the user to view any trends identified by their symptom and dose data, in conjunction with their other Apple Health data. Leveraging SwiftCharts and CareKit, this tab of the application will display these trends in easy-to-interpret ways.

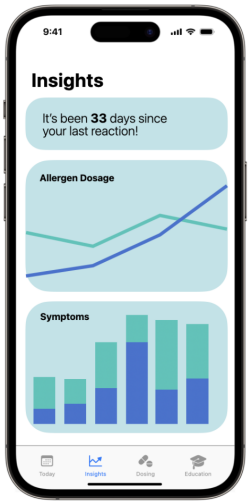


Figure 4.5: Insights Tab

4.6 Dosing Tab

The Dosing Tab, shown in Figure 4.6, is a place where users can log new dosages of their allergens. The application will store all of the dosages in this tab, as well as calculate half doses for the user, as OIT patients are frequently advised to take half doses when they are sick, traveling, stressed, or in other circumstances that may cause issues with a full dose.

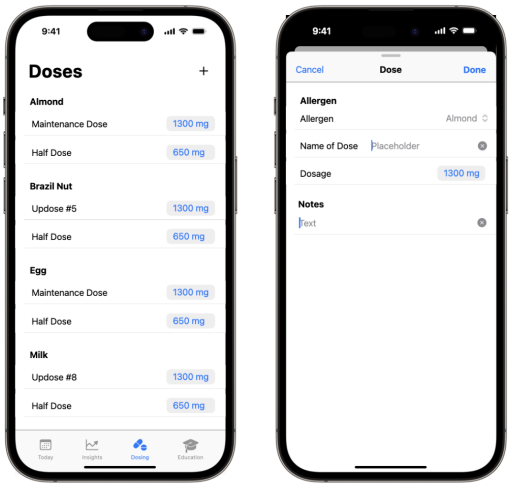


Figure 4.6: Dosing Tab Screens

4.7 Education Tab

The Education Tab, shown in Figure 4.7, is a dedicated space within the application, offering users a wealth of valuable knowledge and resources. This section is thoughtfully curated to provide a comprehensive educational experience, aiming to empower users with essential information related to oral immunotherapy, anaphylaxis, and other pertinent topics. Through a collection of articles and informative resources, the Education Tab serves as an invaluable repository for users seeking to enhance their understanding of the complexities associated with food allergies and the oral immunotherapy process, enabling users to make informed decisions about their health and well-being.

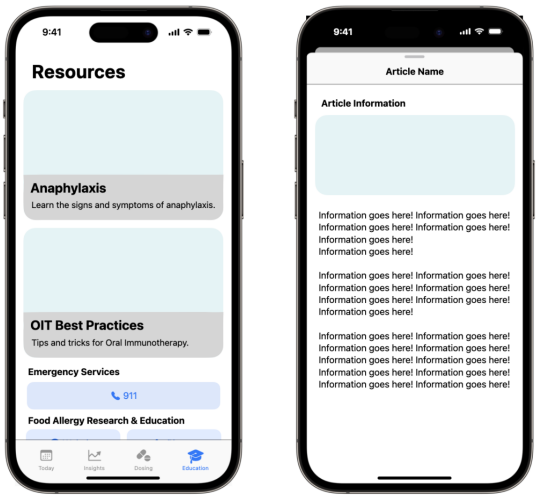


Figure 4.7: Education Tab Screens

Chapter 5

Software Design

5.1 General Architecture

My application will be written in Swift and SwiftUI. The file architecture diagram in Figure 5.1 serves as the foundational blueprint for my application. This visual representation offers a high-level overview of how data and functionalities are organized within the application. It outlines the structure of the app, showcasing the relationships between various components, data storage, and user interfaces. The teal files represent UI-level views, whereas the grey files represent classes that are at a lower level.

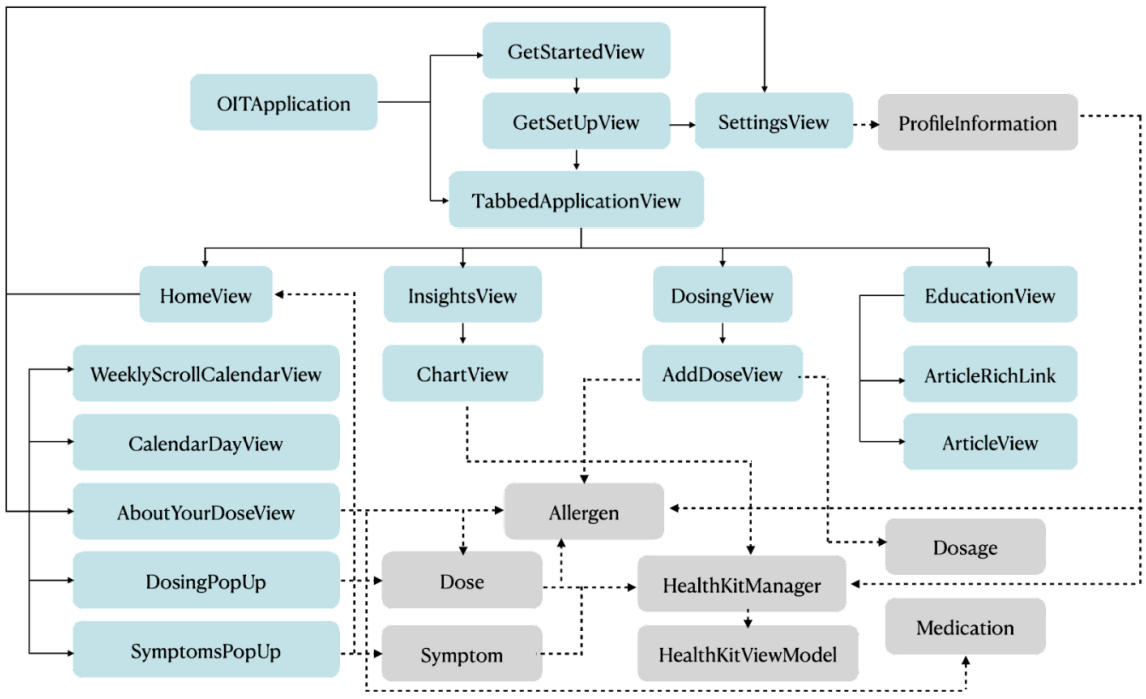


Figure 5.1: Overall File Architecture

5.2 APIs

Leveraging a combination of powerful APIs, my application aims to provide an enhanced and user-centric healthcare experience. I plan to utilize HealthKit, an integral component of the iOS ecosystem, which acts as a central repository for health and fitness data. With the user's explicit consent, my application will seamlessly integrate with HealthKit, facilitating the access and sharing of vital health information from the user's iPhone and Apple Watch [19]. This integration will not only enable the application to provide personalized insights but also enhance the accuracy and reliability of the data we collect.

In addition, CareKit stands as another essential pillar in my app's framework. CareKit empowers users to take control of their health, enabling us to offer functionalities that support trend analysis, goal setting, and incentives. With CareKit 2.0's availability for iOS 13 and iPadOS 13 and above, my app will be able to provide a comprehensive health management experience, fostering a proactive approach to Oral Immunotherapy [20].

To transform data into informative visualizations and offer a visually appealing user interface, I plan to implement the Swift Charts framework. Swift Charts offers a concise SwiftUI-based solution for creating a wide array of customizable charts [21]. This powerful tool provides a range of essential building blocks, such as marks, scales, axes, and legends, which I can creatively combine to develop interactive and data-driven charts. These visualizations will not only enhance the user's understanding of their OIT journey but also make the app more engaging and informative.

Lastly, Swift Data will play a pivotal role in ensuring data persistence and accessibility across app launches. This framework simplifies the management of custom data types, making it easier to store and retrieve information efficiently [22]. By integrating Swift Data with SwiftUI, I can seamlessly fetch and display key data, creating a seamless and reliable user experience. Whether it's preserving user progress or maintaining essential healthcare records, Swift Data will be instrumental in maintaining data integrity and availability.

By harnessing the capabilities of HealthKit, CareKit, Swift Charts, and Swift Data, I aim to deliver a comprehensive, informative, and user-friendly platform for individuals undergoing oral immunotherapy, transforming the way they manage their health and improve their overall well-being.

5.3 Class diagrams

The Class Diagrams below serve as a blueprint for the software development of the application. By visually representing classes and their attributes, this section delves into the heart of the application's design. Following UML standards, each diagram shows the private and public variables and methods for the class.

Table 5.1: Class Diagram: OITApplication

OITApplication
– hasAppBeenOpenedBefore: Bool

Table 5.2: Class Diagram: TabbedApplicationView

TabbedApplicationView
– homeViewTab: NavigationView
– insightsViewTab: NavigationView
– dosingViewTab: NavigationView
– educationViewTab: NavigationView

Table 5.3: Class Diagram: GetStartedView

GetStartedView
– getStartedButton: Button
– appTitle: Text
– backgroundColor: Color

Table 5.4: Class Diagram: GetSetUpView

GetSetUpView
– title: String
+ personalInformation: ProfileInformation
– settingsView: SettingsView
– getStartedButton: Button

Table 5.5: Class Diagram: ProfileInformation

ProfileInformation
+ name: String
+ birthdate: Date
+ allergens: [Allergen]
+ shareDataWithAppleHealth: Bool
+ protectDataWithFaceID: Bool

Table 5.6: Class Diagram: Allergen

Allergen
– AllergenType: enumeration(String)
+ allergen: AllergenType

Table 5.7: Class Diagram: Symptom

Symptom
– SymptomType: enumeration(HKCategoryType)
+ symptom: HKCategoryType
+ getSymptomString( ): String

Table 5.8: Class Diagram: Medication

Medication
– MedicationType: enumeration(HKClinicalTypeIdentifier)
+ medication: HKClinicalTypeIdentifier
+ getMedicationString( ): String

Table 5.9: Class Diagram: SymptomsPopUp

SymptomsPopUp
+ symptomsSelected: [Symptom]
- symptomsButtons: [Button]
– changeButtonAppearance( ): Void

Table 5.10: Class Diagram: HomeView

HomeView
– todayDate: Date
+ selectedDate: Date
– weekView: Bool
– logSymptoms: Bool
– logDose: Bool
– profileButton: Button
– calendarButton: Button
– weeklyScrollCalendarView: WeeklyScrollCalendarView
– logDoseButton: Button
– logSymptomsButton: Button
– aboutYourDoseView: AboutYourDoseView
– dose: Dose
– symptoms: Symptoms

Table 5.11: Class Diagram: WeeklyScrollCalendarView

WeeklyScrollCalendarView
– scrollView: ScrollView
– dayCircle: CalendarDayView
+ startOfWeek( ): Date
+ monthHeader( ): String

Table 5.12: Class Diagram: CalendarDayView

CalendarDayView
– date: Date
– isSelected: Bool
– dayText: Text
– dateText: Text
+ getDayText( ): String
+ getDateText( ): String

Table 5.13: Class Diagram: AboutYourDoseView

AboutYourDoseView
– titleText: Text
– allergenText: Text
– timeOfDose: Date
– dose: Dose
– doseIcon: Image

Table 5.14: Class Diagram: Dose

Dose
+ name: String
+ allergens: [Allergen]
+ dosages: [[Allergen, float]]
+ dateOfDose: Date
+ timeOfDose: Date
+ preDoseMedications: [Medication]
+ notes: String

Table 5.15: Class Diagram: DosingPopUp

DosingPopUp
+ dose: Dose – allergenButtons: [Button] – saveDoseButton: Button
– saveDose( ): Void

Table 5.16: Class Diagram: SettingsView

SettingsView
+ personalInformation: ProfileInformation - nameTextField: TextField - birthdateDatePicker: DatePicker - addAllergenButton: Button - shareDataWithAppleHealth: Toggle - enableFaceID: Toggle

Table 5.17: Class Diagram: HealthKitManager

HealthKitManager
+ symptoms: [HKCategoryType] – dataToRead: [HKCategoryType] – dataToWrite: [HategoryType]
+ setUpHealthRequest()

Table 5.18: Class Diagram: HealthKitViewModel

HealthKitViewModel
+ healthStore: HKHealthStore + healthKitManager: HKManager + isAuthorized: Bool
+ healthRequest() + changeAuthorizationStatus()

Table 5.19: Class Diagram: InsightsView

InsightsView
– title: Text - charts: [ChartView]

Table 5.20: Class Diagram: ChartView

ChartView
– chartType: Chart – chartData: [AnyObject]

Table 5.21: Class Diagram: DosingView

DosingView
- title: Text – addNewDoseButton: Button + allergensAndDoses: [Allergen, [Dosage]] - tableView: TableView

Table 5.22: Class Diagram: AddDoseView

AddDoseView
– allergen: Picker
– nameOfDose: TextField
– dosage: TextField
– notes: TextField
– saveDosageButton: Button
+ savedDosage: Dosage

Table 5.23: Class Diagram: Dosage

Dosage
+ allergen: Allergen
+ nameOfDose: String
+ dosage: Int
+ notes: String

Table 5.24: Class Diagram: EducationView

EducationView
- title: Text
– articles: [ArticleRichLink]
- resourceHeadings: [String]
- resourceButtons: [Button]

Table 5.25: Class Diagram: ArticleRichLink

ArticleRichLink
– title: Text
– description: Text
– image: Image
– article: ArticleView

Table 5.26: Class Diagram: ArticleView

ArticleView
– title: Text
– description: Text
– image: Image
– articleInformation: Text

Chapter 6

Constraints and Standards

Before diving into the development process, it is important to recognize and address the various constraints and standards that shape and govern the project, so that I can plan for them. This section delves into the constraints that have influenced the project's trajectory, ranging from time limitations imposed by academic schedules to legal considerations in ensuring the utmost user privacy. Simultaneously, the section explores the standards adhered to throughout the development process, aligning the project with recognized benchmarks in software engineering, accessibility, and interoperability. By understanding and adhering to these constraints and standards, my application will not only fit within the confines of practical realities but also conform to industry best practices, fostering a robust and ethically sound healthcare solution.

6.1 Constraints

Developing a software application as a solo endeavor brings its own set of challenges, and recognizing and navigating these constraints is paramount for success. While financial considerations are mitigated in this context, time and legal factors emerge as critical constraints that demand careful attention.

- **Time:** The timeline of the senior capstone experience and academic deadlines impose significant time constraints on all phases of the project. Structuring the project plan and development process within these allocated timeframes is essential for meeting deliverables and achieving project goals.
- **Legal:** Legal considerations, particularly healthcare privacy regulations such as HIPAA, stand as significant factors influencing the project. Ensuring strict compliance with these regulations is non-negotiable to safeguard patient data and uphold legal standards. This legal awareness adds complexity to the development process, demanding meticulous attention to detail.
- **Maintenance Considerations:** Planning for the long-term maintenance of the project post-deployment is a critical aspect of the development process. Designing the system with foresight for future updates and enhance-

ments ensures that ongoing maintenance can be executed seamlessly. This forward-thinking approach aims to maximize the application's lifespan and adaptability, addressing potential challenges before they arise.

In navigating these constraints, a strategic allocation of resources and a proactive approach to time management will be key to the success of the project. By addressing these challenges head-on, I hope to lay the foundation for a robust and sustainable software solution.

6.2 Standards

Ensuring that the developed application aligns with established industry standards is paramount for creating a robust, reliable, and inclusive healthcare solution. The adherence to specific standards across various facets of the project speaks to a commitment to excellence and user-centric design.

- **Unified Modeling Language (UML) Standards**: Unified Modeling Language (UML) Standards constitute a crucial framework for expressing and communicating the structural and behavioral aspects of a software system in a standardized visual notation. UML provides a common language for software developers, analysts, and designers to collaboratively conceptualize, design, and document complex systems. These standards encompass a variety of diagram types, including class diagrams, sequence diagrams, and use case diagrams, each serving a distinct purpose in capturing different facets of system architecture and functionality [12]. In creating my diagrams and plans for this project, I've utilized UML standards, ensuring ease of readability in the software engineering community.

- **IEEE Standards:** Adhering to a suite of IEEE standards has been integral to the development process. Particularly, in the realm of software engineering practices, these standards have acted as guiding principles [13]. I've outlined some of the IEEE standards I've followed below:

  - **IEEE 830 - Software Requirements Specification:** This standard provides guidelines for preparing a software requirements specification document, ensuring clarity and completeness in documenting the functional and non-functional requirements of the software [27].

  - **IEEE 1016 - Software Design Descriptions:** This standard outlines the structure and content of software design descriptions, aiding in the effective communication of design details among project stakeholders [28].

  - **IEEE 1471 - Architecture Description:** Focused on architectural descriptions, this standard offers guidance on documenting and presenting system architectures, helping to ensure consistency and comprehensibility [29].

  - **IEEE 830.1 - Recommended Practice for Software Requirements Specifications Extensions:** This standard offers additional guidance on extending and customizing IEEE 830 for specific project needs, potentially useful in tailoring requirements specifications [30].

- **ISO Standards:** The application also adheres to ISO standards, encompassing programming languages and design methodologies, some of which are listed below [14].

  - **ISO/IEC 9126 - Software Engineering - Product Quality:** This standard provides a framework for defining and evaluating software quality characteristics, including functionality, reliability, usability, efficiency, maintainability, and portability [31].
  - **ISO/IEC 25000 - Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE):** This series of standards provides guidance on the application of the ISO/IEC 9126 quality model and defines a set of quality requirements and evaluation standards for software products [32].
  - **ISO 9241-171:** This ISO standard provides guidelines for designing software that is accessible to people with disabilities. It covers aspects like visual, auditory, and cognitive accessibility [33].

- **Apple Standards**: Because I am creating an iPhone application, it is important to follow the Apple-specific standards listed below.

  - **Human Interface Guidelines (HIG):** Apple's HIG provides design principles and recommendations for creating a consistent and intuitive user experience across iOS, macOS, watchOS, and tvOS. When designing the UI of my application, I closely followed the HIG [15].
  - **App Store Review Guidelines:** These guidelines outline the criteria that apps must meet to be accepted on the App Store. It covers various aspects including app functionality, design, security, and more [16].
  - **Accessibility Guidelines:** Apple provides comprehensive accessibility guidelines for developers. This includes guidance on designing accessible user interfaces, providing alternative content for multimedia, and ensuring compatibility with VoiceOver, Apple's built-in screen reader [17].

- **Health Insurance Portability and Accountability Act (HIPAA) Compliance Standards:** HIPAA sets the standard for protecting sensitive patient data and defines how this information should be handled, stored, and transmitted [18].

  - **Secure Data Transmission:** Ensure that all patient data transmitted between the application and any backend servers or databases is encrypted. HIPAA requires the use of secure communication protocols, such as TLS, to protect patient information during transmission.

- **Data Storage and Encryption:** Implement robust encryption mechanisms for stored patient data. This includes data at rest on servers or devices. Encryption helps safeguard patient information in the event of unauthorized access.
- **Access Controls:** Enforce strict access controls to limit who can access patient data within the application. Implement user authentication and authorization mechanisms to ensure that only authorized personnel can view or modify sensitive information.

Chapter 7

Schedule

This project requires a meticulously planned roadmap to navigate the complexities and milestones that lie ahead. The Gantt Chart in Figure 7.1 serves as a visual compass, delineating the chronological sequence of tasks and deadlines essential for the successful completion of the project. Each bar, carefully plotted along the timeline, represents a crucial aspect of the thesis development process, offering a comprehensive overview of the project's progression. This dynamic tool not only aids in project management but also provides a clear visualization of interdependencies, helping to anticipate potential challenges and facilitating effective decision-making throughout the thesis endeavor.

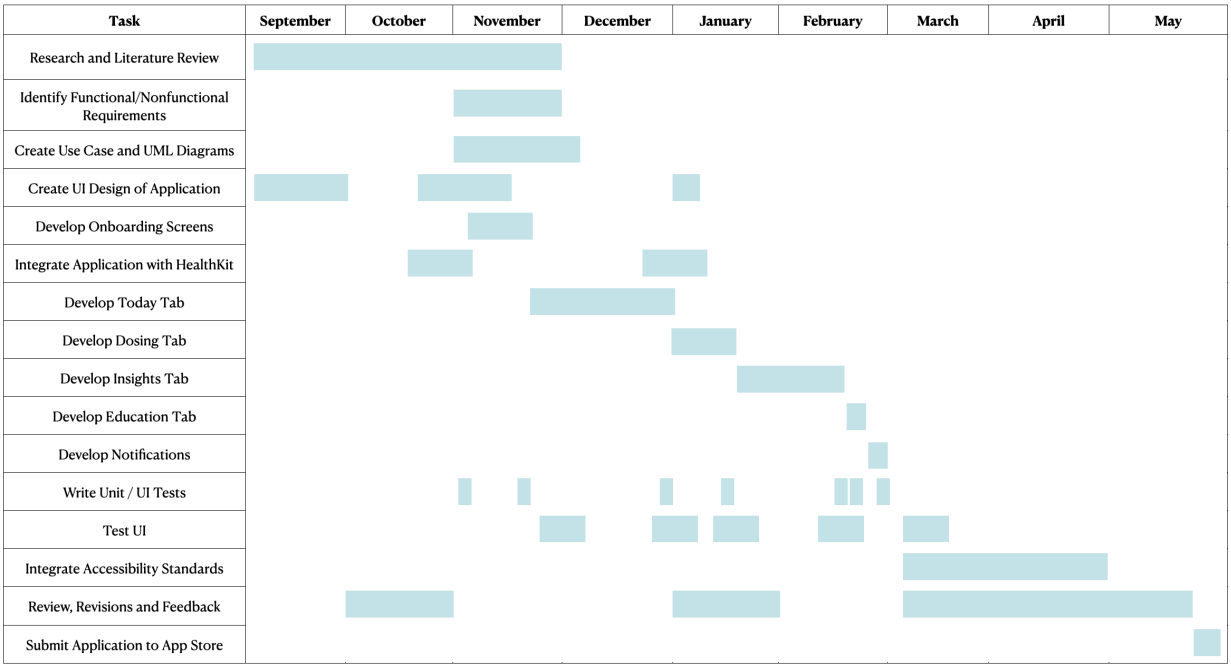


Figure 7.1: Project Gantt Chart

Chapter 8

Risk Analysis

The development process is inherently complex, marked by uncertainties and challenges that have the potential to influence the successful completion of any project. In recognizing the dynamic nature of software development, it becomes crucial to proactively identify and analyze potential risks. This approach not only facilitates early detection but also enables the formulation of effective mitigation strategies to safeguard the project's trajectory.

Risk analysis is an indispensable component of project management, providing a structured framework for understanding, assessing, and addressing potential threats. It involves a systematic evaluation of uncertainties that could impact project objectives, timelines, and deliverables. By undertaking a comprehensive risk analysis, project teams can anticipate, prioritize, and respond to potential issues, minimizing the likelihood of disruptions and ensuring a smoother project progression.

The risk analysis table presented in Table 8.1 serves as a strategic tool for capturing and categorizing risks that may manifest at various stages of the project lifecycle. Each entry in the table delineates a specific risk, accompanied by its probability of occurrence, severity, potential consequences, and recommended preventative measures. The probability is quantified on a scale from 0 to 1, severity is ranked from 1 to 10, and the impact is calculated as the product of probability and severity.

Risk	Probability	Severity	Impact	Consequences	Mitigation
Constraints are not met	0.5	8	4	Application is incomplete	Consult with advisor for clarification, and check initial progress before proceeding
Unable to integrate with Apple HealthKit	0.2	10	2	User is unable to view their data in the Health app, or share data with their doctor	Research HealthKit, and abstract the HealthKit portion of the code, so that if this area isn't implemented, the app still works
Unable to implement feature(s)	0.7	8	5.6	Code is unfinished, or is finished, but with less features than originally planned	Ask for assistance to keep up with timeline, and prioritize tasks, so that lower priority items can be dropped if needed
App store policies are not met	0.05	10	0.5	Unable to submit the application to the App Store	Regularly check App Store policies, and update app criteria as needed
Application is not compatible with iPhone model	0.1	10	1	User is unable to use the application	Ensure that the APIs and technologies used have an appropriate minimum iOS, and that the code is tested on various iPhone models
Source control error	0.4	4	1.6	Setback in development, due to loss of work	Back up work in GitHub, and ensure code is pushed after major code changes
Not enough time	0.5	7	3.5	Unable to complete the project by the due date	Allow for more time than expected for development, and keep to the schedule defined in the previous chapter
Bugs	1.0	4	4	Subpar user experience, and implementation not working fully	Develop unit and UI tests as features are developed, and check functionality of each feature before continuing to the next
Developer unable to contribute	0.2	10	2	Progress on the project becomes stagnant, as there is only a single developer	Regularly check in with advisor on progress, and prioritize tasks, so that if the developer becomes indisposed for a period of time, lower priority items can be dropped

Table 8.1: Risk Analysis Table

Chapter 9

The Final Product

I was able to stay on track with my schedule, implementing all aspects of my project that I had planned. Along the way, I learned how to add entitlements, ensuring my app had the necessary permissions and capabilities. Understanding HealthKit permissions and data handling was crucial for integrating health-related features seamlessly. Additionally, I gained insights into the differences between previewing, simulating, and deploying on an actual device, optimizing my app's performance for various environments. Moreover, I tackled the challenge of persisting images effectively, enhancing the user experience by securely storing visual data.

9.1 Differences Between Prototype and Final Product

Although the final version of my application remains faithful to its initial vision and incorporates many elements from the prototypes, the journey from conception to completion involved several noteworthy design refinements and feature enhancements.

9.1.1 UI Design Choices

One of the initial design enhancements I implemented in the application's user interface involved the utilization of emojis. I found the available selection of emojis for symptoms to be limiting, with many symptoms having similar emojis despite their distinctiveness. To address this issue, I opted for a combination of native emojis and custom emojis (see Figure 9.1). This approach allowed me to fill the gaps where native emojis fell short, ensuring a more comprehensive representation of symptoms.

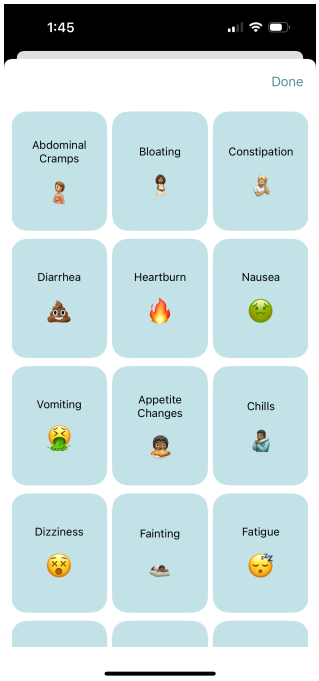


Figure 9.1: Unique Emojis for Symptoms

Furthermore, while there currently exists only one generic tree nut emoji, my application necessitated the individual distinction of various tree nuts. To avoid visual monotony, I introduced custom emojis tailored to each specific tree nut, as you can see in Figure 9.2. This customization not only enhanced the aesthetic appeal but also improved clarity and user experience.

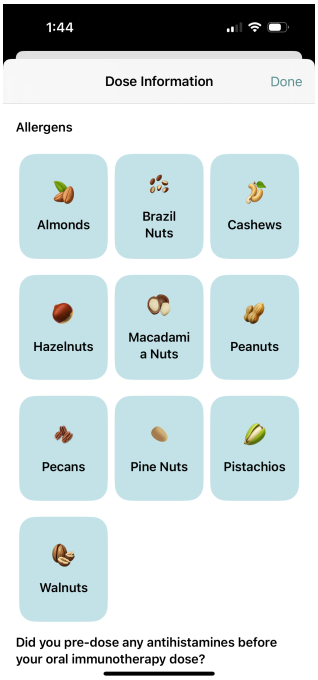


Figure 9.2: Unique Emojis for Nuts

Another enhancement I added to the UI of the application was tips. Utilizing Apple's TipKit, I was able to add helpful pop-ups and messages to the user, explaining how to use different features of the application (see Figure 9.3).

While many of the tips appear as the user is getting set up with the app, I was also able to utilize custom settings, like "displayFrequency" to customize how often new tips are presented after other tips have been displayed. This helped introduce the tips in stages, to not overwhelm the user with too many at once.

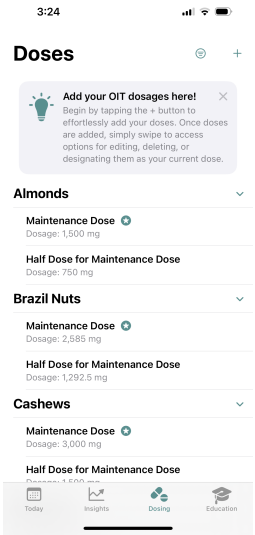


Figure 9.3: Doses Tab Tip

9.1.2 Additional Resources

As mentioned in Section 3.2: Consultation with Immunologists, many immunologists advised incorporating some type of feature to allow users to upload images of their emergency action plans to the Resources tab. So, I added an ”Important Documents” section to the Resources tab to allow patients to keep all of their important documents (e.g. Emergency Action Plan) in one place so that they can find them easily.

Users can easily tap on the ”Select Images” button to add more documents, tap on documents in the section to view them, and tap on the ”x” buttons below the images to remove them as needed (see Figures 9.4, 9.5, and 9.6).

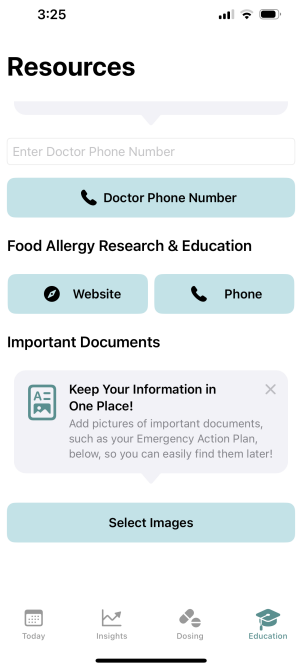


Figure 9.4: Important Documents Section

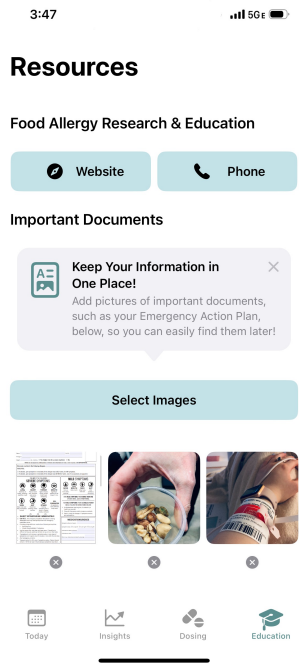


Figure 9.5: Important Documents Section With Documents Added

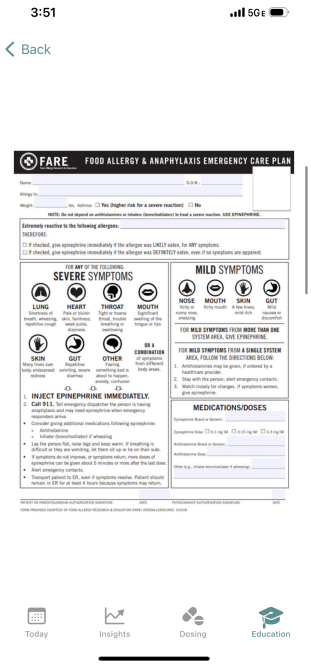


Figure 9.6: Tapping to View Important Document

Additionally, as you can see in Figure 9.7, I added a section for users to add their doctor's phone number, so they can easily reach out to them if needed.

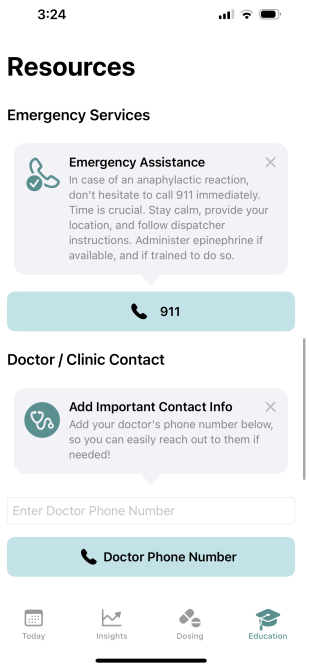


Figure 9.7: Doctor / Clinic Contact Info Section

9.1.3 Profile

While the profile section remained mostly the same as the prototypes, I added a few subtle changes. First, I added a section to add a profile picture (see Figures 9.8 and 9.9). I thought this added a needed element of personalization to the app and tied together the personal information section of the profile nicely.

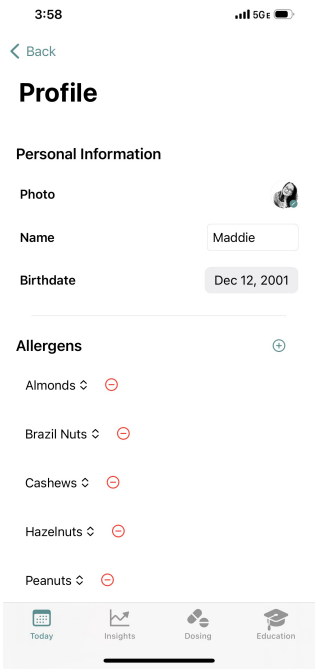


Figure 9.8: Add Profile Photo Section

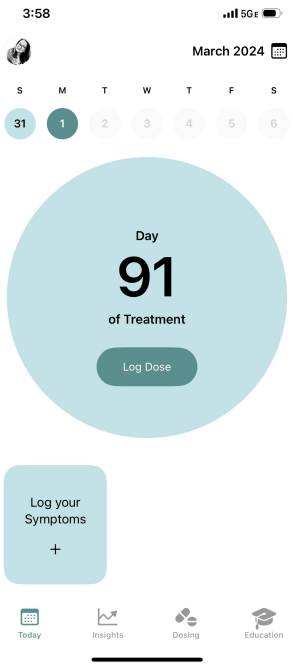


Figure 9.9: Profile Picture in Top Corner of App

Additionally, when handling users with less common allergens that weren't in the list of common allergens, I ran into issues with no emoji associated with the allergen, as well as issues with the data persistence. To fix this, I added a pop-up when the user selected "Other" from the list of allergens in their profile. This allowed them to enter their

allergen's name, as well as an emoji that correlated to that allergen (see Figures 9.10, 9.11, and 9.12).

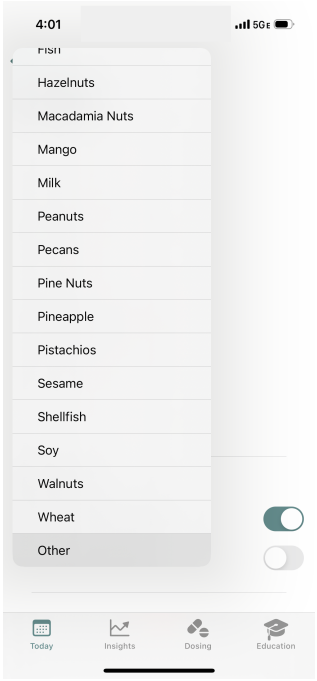


Figure 9.10: User Tapping Other Allergen from Common Allergens List

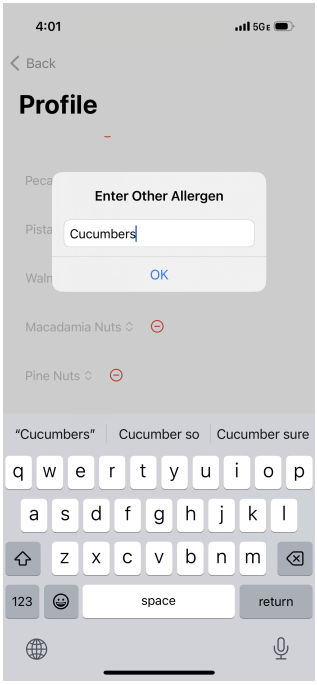


Figure 9.11: User Adding Other Allergen Name

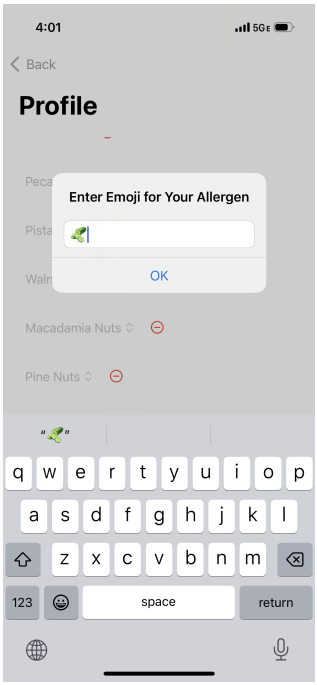


Figure 9.12: User Adding Other Allergen Emoji

Finally, my initial prototype had a ”Protect data with FaceID” button, however not every iOS device has FaceID enabled. So, I changed the toggle to say ”Protect data with Passcode, TouchID, or FaceID” to account for different iPhone models and security setups (see Figure 9.13). And, depending on the setup, the screen to authenticate will ask for the user’s passcode, TouchID, or FaceID.

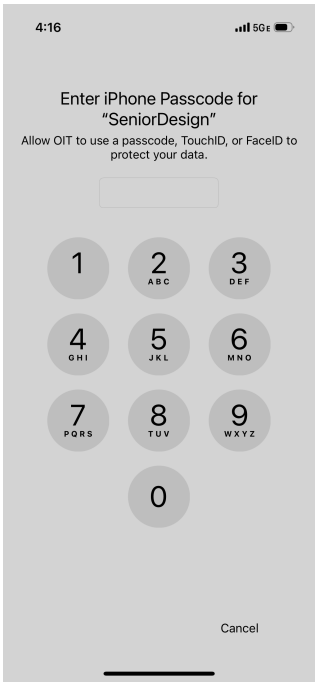


Figure 9.13: Screen to Enter Passcode

9.1.4 Dosing Tab Changes

The Dosing tab of the app went through several iterations after the prototypes were made. Instead of having a large list of all the allergens and doses, I separated the sections and made them collapsable, allowing the user to more easily navigate through the doses, as you can see in Figures 9.14, and 9.15.

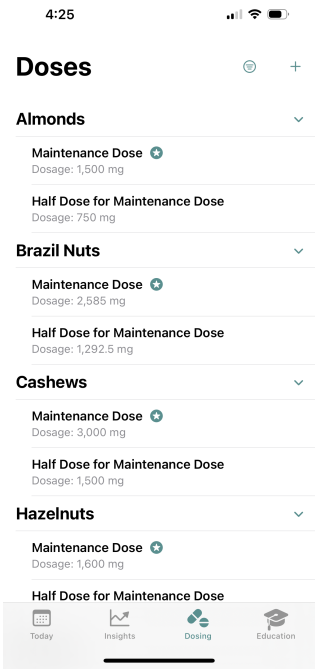


Figure 9.14: Uncollapsed Sections of Dosing Tab

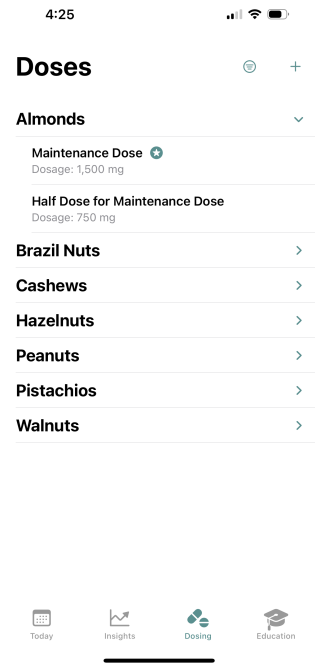


Figure 9.15: Collapsed Sections of Dosing Tab

Additionally, I added the ability to mark which dose is your current dose, filter the tab only to show current doses, and edit or delete doses (see Figures 9.16, 9.17, 9.18, and 9.19).

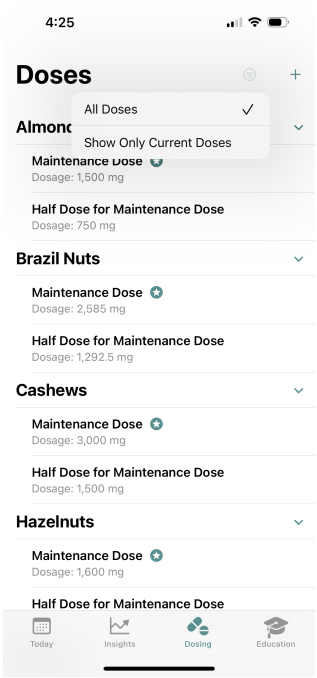


Figure 9.16: Dosing Tab Filters

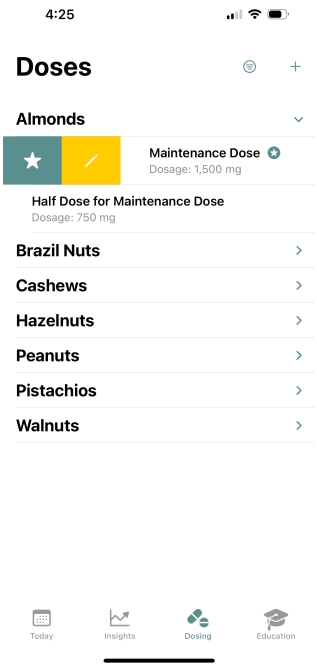


Figure 9.17: Dose Edit Actions

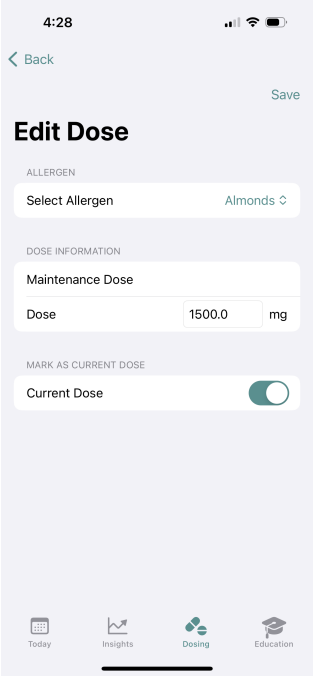


Figure 9.18: Dose Edit Screen

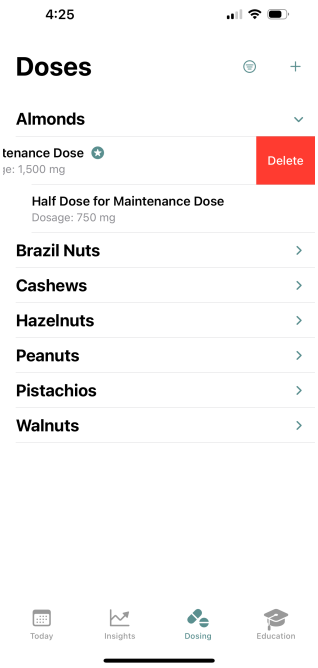


Figure 9.19: Dose Delete Action

9.2 User Guide

After finishing developing my application, I created a comprehensive user guide for it, as you can see in Figures 9.20 – 9.33.

Oral Immunotherapy Tracker User Guide

Everything you need to know to use the application!

Table of Contents

Figure 9.20: User Guide Page 1

Oral Immunotherapy Tracker User Guide

Everything you need to know to use the application!

Table of Contents (Continued)

Figure 9.21: User Guide Page 2

Set Up and Get Started

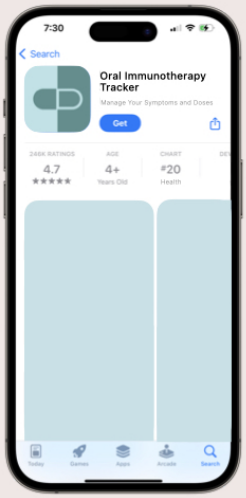
This section explains everything you need to know to get set up using the Oral Immunotherapy Tracker application!

Downloading the App

The OIT app is easy to download from the App Store!

Simply go to > Search, enter Oral Immunotherapy Tracker, select the app from the search results, and tap "Get" to install the application onto your phone.

Once installed, you can start getting set up with the application!



Setting Up Your Profile

When you first open the app, you'll see a screen with a "Get Started" button. Tapping this button will bring you to the Set Up screen.

Add Personal Information

On the Set Up screen, you can add personal information to personalize your app experience.

- Tap the + on the photo picker to select a profile photo
- Enter your preferred name in the name textfield
- Select your birthdate from the date picker

Additionally, in the "Allergens" section, you can add all of the allergens you're allergic to by tapping the + button and selecting your allergen from the picker.

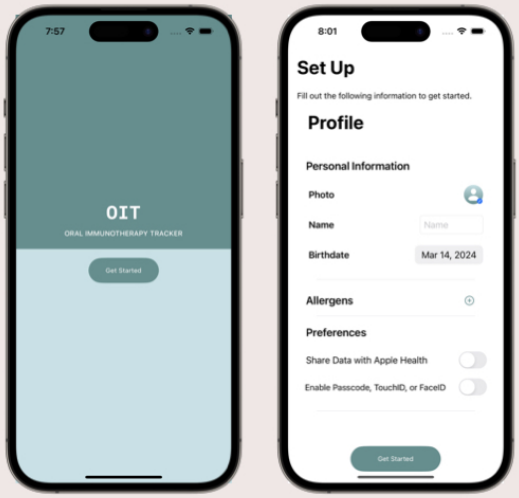


Figure 9.22: User Guide Page 3

Set Up and Get Started

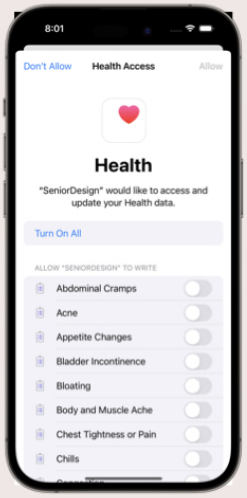
Share Data with Apple Health

In order to integrate your symptom and dose data with Apple Health, you need to allow Oral Immunotherapy Tracker to read and write relevant Health data.

Once you toggle the “Share Data with Apple Health” toggle, a pop up will be displayed to allow the application to access and update your Health data.

This screen breaks down all of the data the app could access, and you can opt to share some of the data, all of the data, or none* of the data.

***Note:** If you opt out of sharing data with Apple Health, the insights shown and overall app experience may be limited.



Protect Data with FaceID, TouchID, or Passcode

While all of your data in the app is encrypted and secure, you can further protect your data with FaceID, TouchID, or a Passcode (depending on your phone’s capabilities).

If you choose to toggle “Protect Data with FaceID, TouchID, or Passcode” on, then every time you enter the application, you will need to authenticate with your face, fingerprint, or passcode in order to use the app.

Figure 9.23: User Guide Page 4

Basics of the App

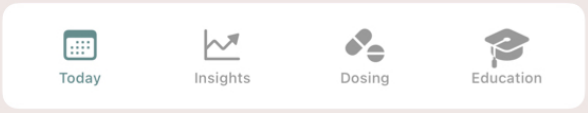
This section goes over some of the basics of using the app.

Navigating Between Tabs

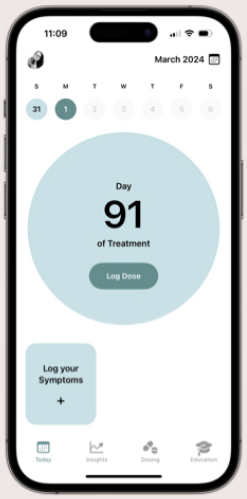
You can use the bottom tab bar to navigate between the Today, Insights, Dosing, and Education tabs.

Simply tap on one of the four icons to navigate to that tab!

The current tab you selected will be highlighted a teal color.



A Closer Look at the Tabs

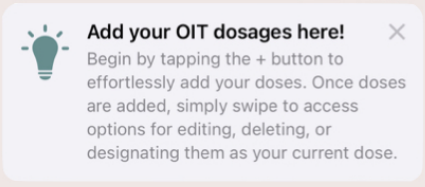


Dismissing Tips

As you navigate the application, you will find tips placed to help you learn about the app and guide you through using it.

Once you read through a tip, you can dismiss it by tapping the “x” in the top right corner of the tip.

Be careful, though! Once you dismiss a tip, you won’t be able to see it again!



An Example of a Tip

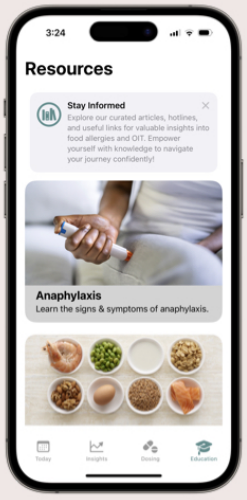


Figure 9.24: User Guide Page 5

Basics of the App

Dark and Light Mode

The application supports both dark and light modes, with colors adjusted slightly to optimize the viewing experience for your eyes.

The app's theme will automatically sync with your device settings; if your phone is in dark mode, the app will switch to dark mode accordingly, and vice versa.

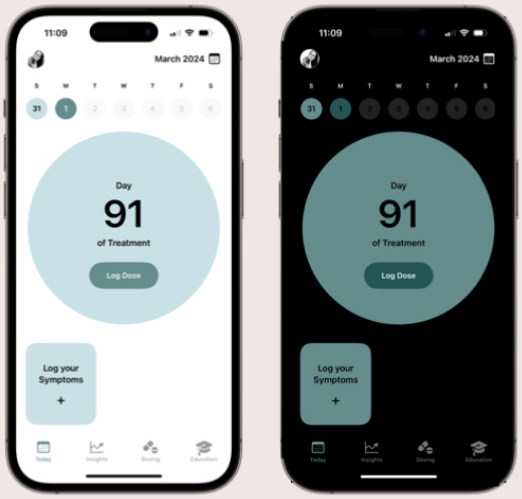


Figure 9.25: User Guide Page 6

Today Tab

The Today tab serves as a central hub for the most crucial information of your day. This section provides an overview of the key features of the Today Tab within the application.

Weekly Scroll View versus Monthly View

By default, upon entering the Today tab, you will encounter a weekly scroll view, with the current date pre-selected.

Within this view, you have the flexibility to tap on other days of the week or scroll back to previous weeks. Furthermore, you can easily identify the current day of treatment, log a dose for the selected date, or record symptoms specific to that date.

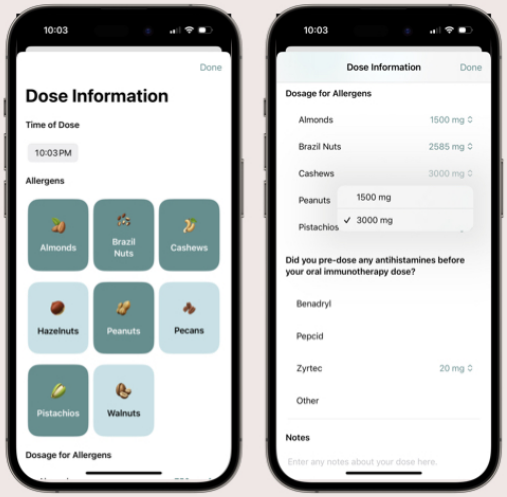
You also have the option of tapping the calendar button in the top right corner to enter a monthly view.

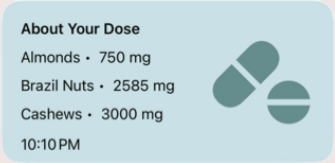
Logging a Dose

Tapping on the "Log Dose" button will prompt a pop-up where you can input information regarding the dose you've taken.

Within this pop-up, you can specify the time of the dose, select the allergens you've dosed, indicate the dosages for each allergen, and note if you've taken any medications prior to your OIT dose. And, you have the option to include any relevant notes that you may want to share with your doctor.

Once you've logged your dose, the Today tab will display a succinct overview of your dose.





An Example of a Dose Overview

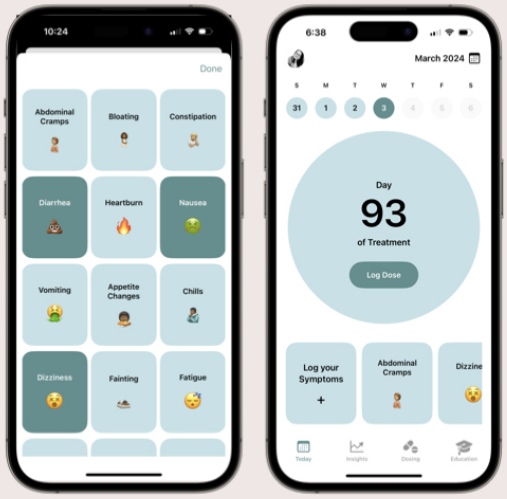
Figure 9.26: User Guide Page 7

Today Tab

Logging Symptoms

Tapping on the "Log your Symptoms" button will direct you to a pop-up displaying a list of symptoms. Simply tap on the symptoms you are experiencing to log them, then tap "Done" to save your entries.

After logging your symptoms, the Today tab will showcase them in a scroll view at the bottom of the screen for easy reference.



Profile

You can access the Profile section of the app by tapping on the icon located in the top left corner of the Today tab.

If you have set up a profile picture, it will be displayed as the icon. Otherwise, you will see a generic profile icon.

Generic Profile Icon

Editing Your Information

In the Profile tab, you have the option to edit all the information you provided during the setup process:

- Tap the "+" icon on the photo picker to choose a profile photo.
- Edit your preferred name in the name text field.
- Modify your birthdate using the date picker.

Moreover, within the "Allergens" section, you can add any allergens you're allergic to by tapping the "+" button and selecting your allergen from the picker, or remove allergens using the “-“ button.

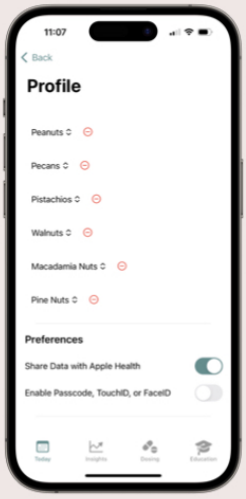
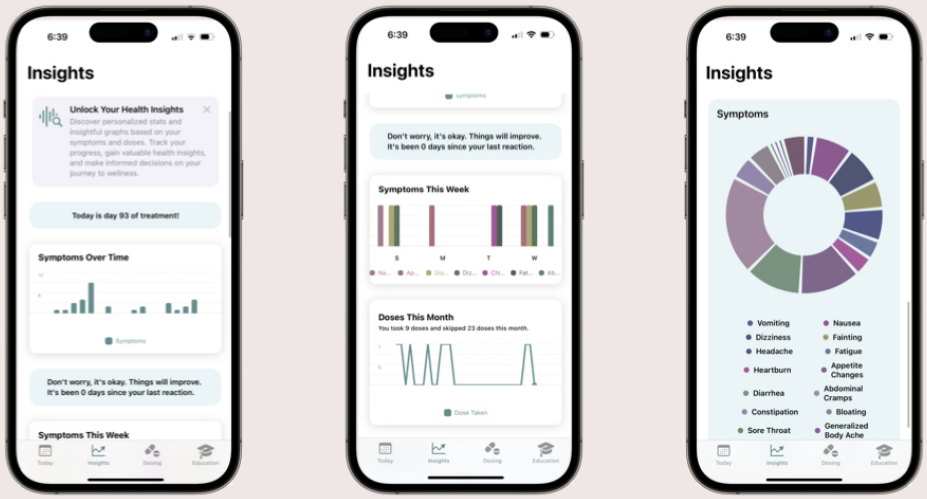


Figure 9.27: User Guide Page 8

Insights Tab

The Insights tab acts as a central hub where you can explore significant trends and insights derived from your recorded doses and symptoms.

Dynamically analyzing your symptom and dose data, the Insights tab displays different charts, messages, and more, based on how you have been doing.



If you don't see many charts or insights, don't worry! The app might need to collect a little bit more data from your doses and symptoms before it can display trends!

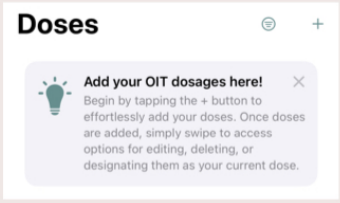
Figure 9.28: User Guide Page 9

Dosing Tab

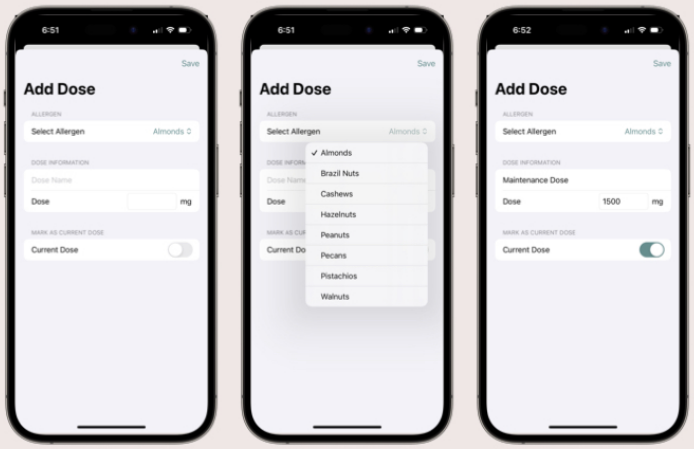
The Dosing tab is your dedicated space to input and monitor all the dosages prescribed by your doctor for your oral immunotherapy treatment.

Adding a Dose

When you initially start using the app, you'll find that none of your dosages are entered. You'll be prompted with a message encouraging you to input your dosages into the app.



Prompt to Add OIT Dosages



To do this, tap the "+" button located in the top right corner of the screen. A pop-up will then appear, allowing you to add a dose.

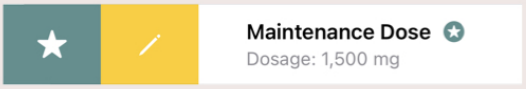
Within this pop-up, you can choose the allergen the dose pertains to, specify the dosage in milligrams, and indicate if it's your current dose for that allergen as necessary.

Once you've finished, simply tap "Save" to store your dosage!

Editing a Dose

Editing a dose you've added is straightforward if you've made a mistake or need to update it.

Just swipe right on the dose, then tap the yellow edit button. This action will present you with a sheet to modify the dose. Once you've made the necessary changes, tap "Save" to update your dosage!



Right Swipe Buttons for Dose

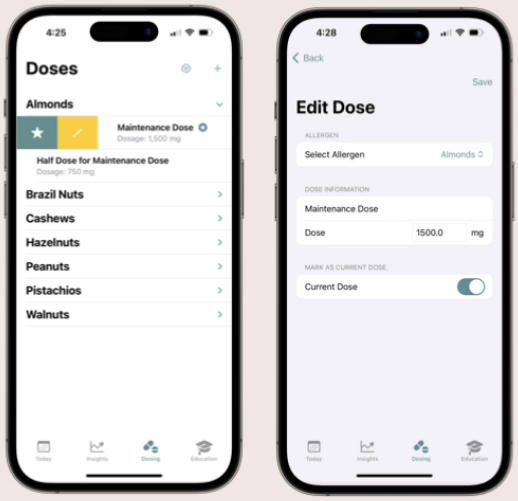


Figure 9.29: User Guide Page 10

Dosing Tab

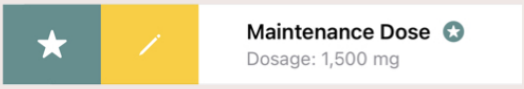
Deleting a Dose

To delete a dose, you can swipe left on the dose and select the red delete button.

Left Swipe Buttons for Dose

Marking a Dose as Current

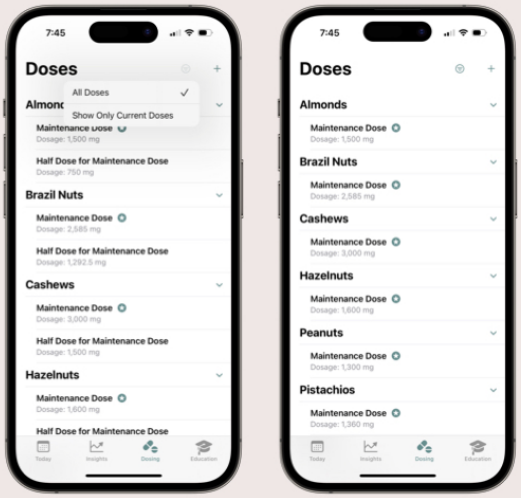
To assist you in keeping track of your doses, you can designate one dose of each allergen as your current dose. To do this, swipe right on the dose and tap the teal star button. This action will add a teal star symbol next to the dose, indicating that it's your current one.



Right Swipe Buttons for Dose

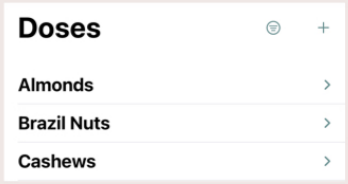
Filtering Doses

You have the option to filter the list of doses to display only your current doses. This feature allows you to conveniently locate your current dose for each allergen, especially if you have numerous dosages listed.



Collapsing Sections

Each allergen section in the tab is accompanied by arrows. To collapse or expand a section, simply tap on the arrow next to it. This allows for easy management of the display based on your preferences.



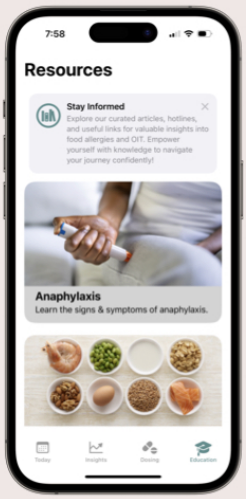
Collapsed Allergen Sections

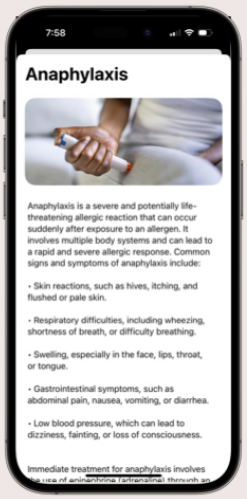
Figure 9.30: User Guide Page 11

Education Tab

The Education tab offers a wealth of educational resources, articles, and a convenient space for storing valuable information.

Viewing Articles





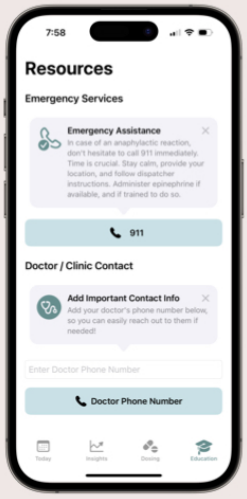
The app provides numerous articles to enhance your knowledge about allergies, anaphylaxis, and OIT.

To read an article, tap on its preview in the tab, and it will pop up for you to explore!

Resources

In addition to articles, you can access essential resources such as emergency services phone numbers, your doctor's contact information, and Food Allergy Research and Education's hotline and website within this tab. Simply scroll down past the articles to find them.

Furthermore, since the app isn't initially aware of your doctor's phone number, you can input it into the textbox provided in the Doctor/Clinic Contact section to include it in the app.



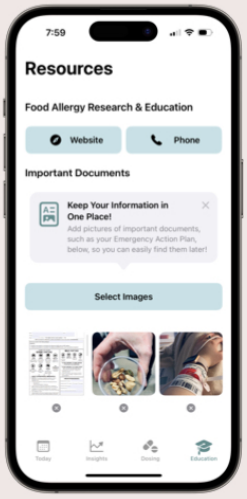


Figure 9.31: User Guide Page 12

Education Tab

Important Documents

The "Important Documents" section of the tab serves as a centralized location for storing all your crucial documents, such as an Emergency Action Plan, making them easily accessible.

You can effortlessly add more documents by tapping on the "Select Images" button, view documents by tapping on them within the section, and remove them as needed by tapping on the "x" buttons located below the images.

Tap the "x" underneath the document to delete it!

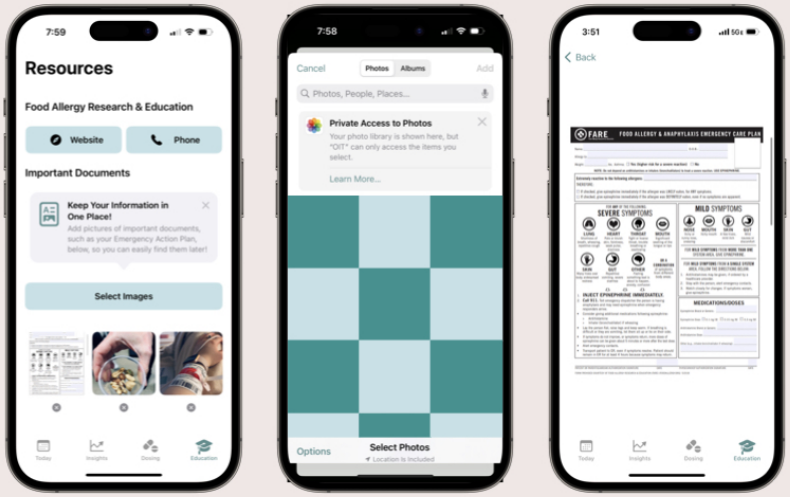


Figure 9.32: User Guide Page 13

Copyright

Figure 9.33: User Guide Page 14

Chapter 10

Testing

Testing is a pivotal component in the development lifecycle of the Oral Immunotherapy application, ensuring the reliability, functionality, and user experience align with the project goals. This chapter delves into the comprehensive testing strategies I employed throughout the project's development, encompassing Visual Testing, XCTests, and User Testing.

10.1 Visual Testing

Visual testing played a crucial role in ensuring the graphical elements and user interface (UI) components of the application met the desired standards. Leveraging Xcode's SwiftUI Canvas's previews, I systematically examined the visual representation of the app across different devices, screen sizes, phone orientations, and color schemes (e.g. light and dark modes). This approach allowed me to identify and rectify any layout inconsistencies, aesthetic issues, or UI element misalignments.

In Figures 10.1, 10.2, 10.3, 10.4, and 10.5 you can see an example of the previews given in Xcode's SwiftUI Canvas.

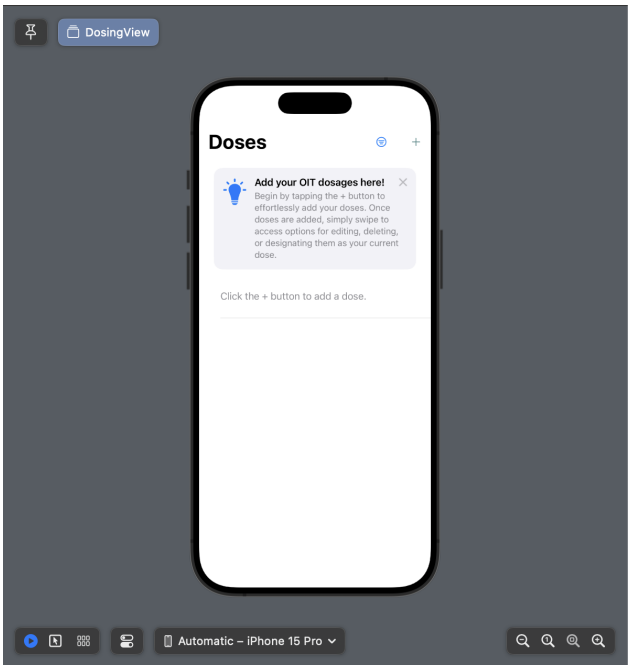


Figure 10.1: SwiftUI Canvas

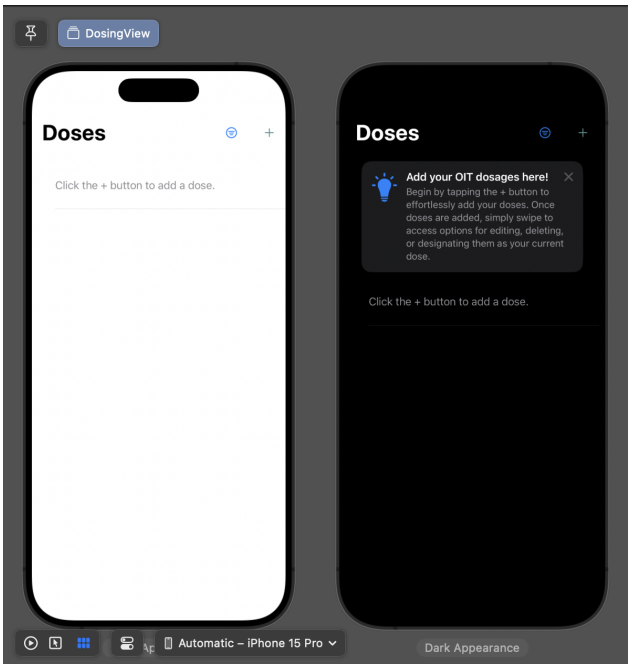


Figure 10.2: Swift UI Canvas Color Scheme Variants

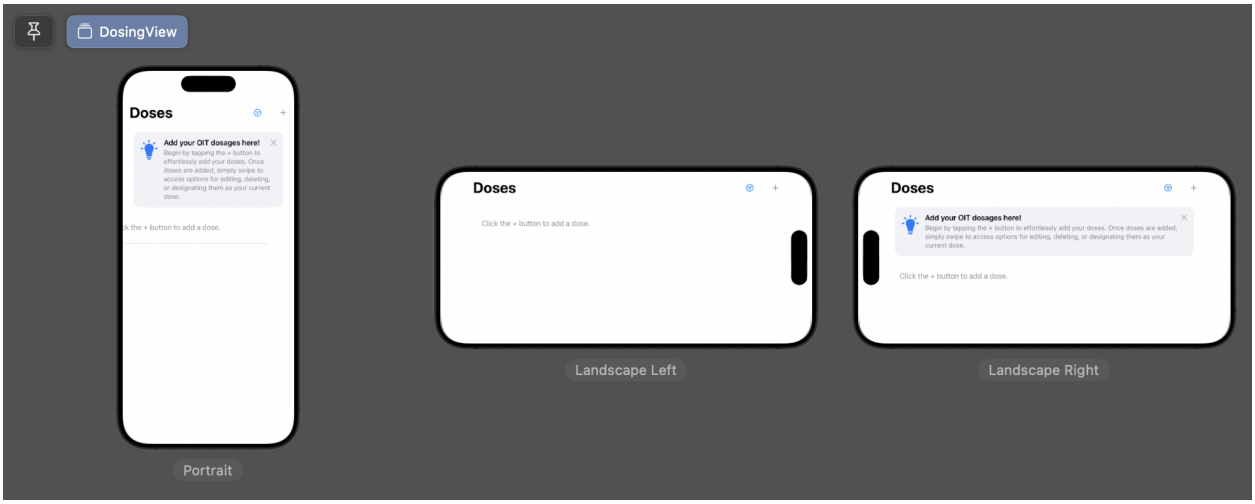


Figure 10.3: Swift UI Canvas Orientation Variants

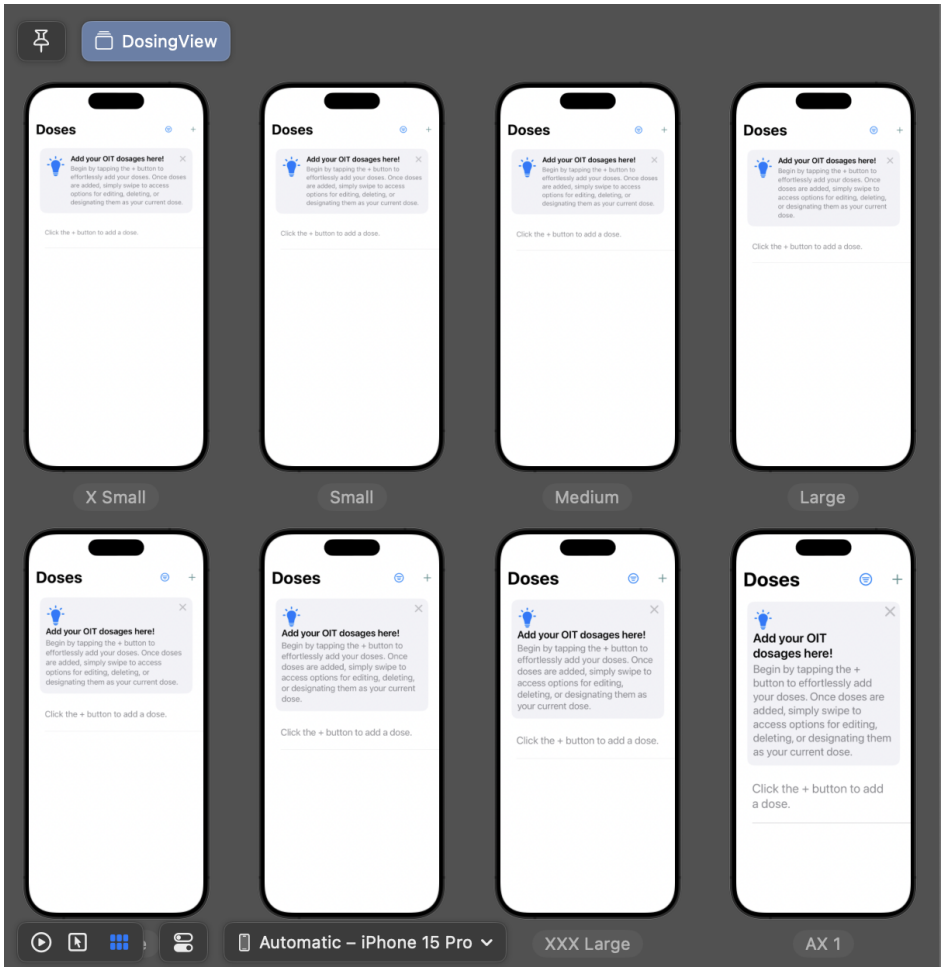


Figure 10.4: Swift UI Canvas Dynamic Type Variants

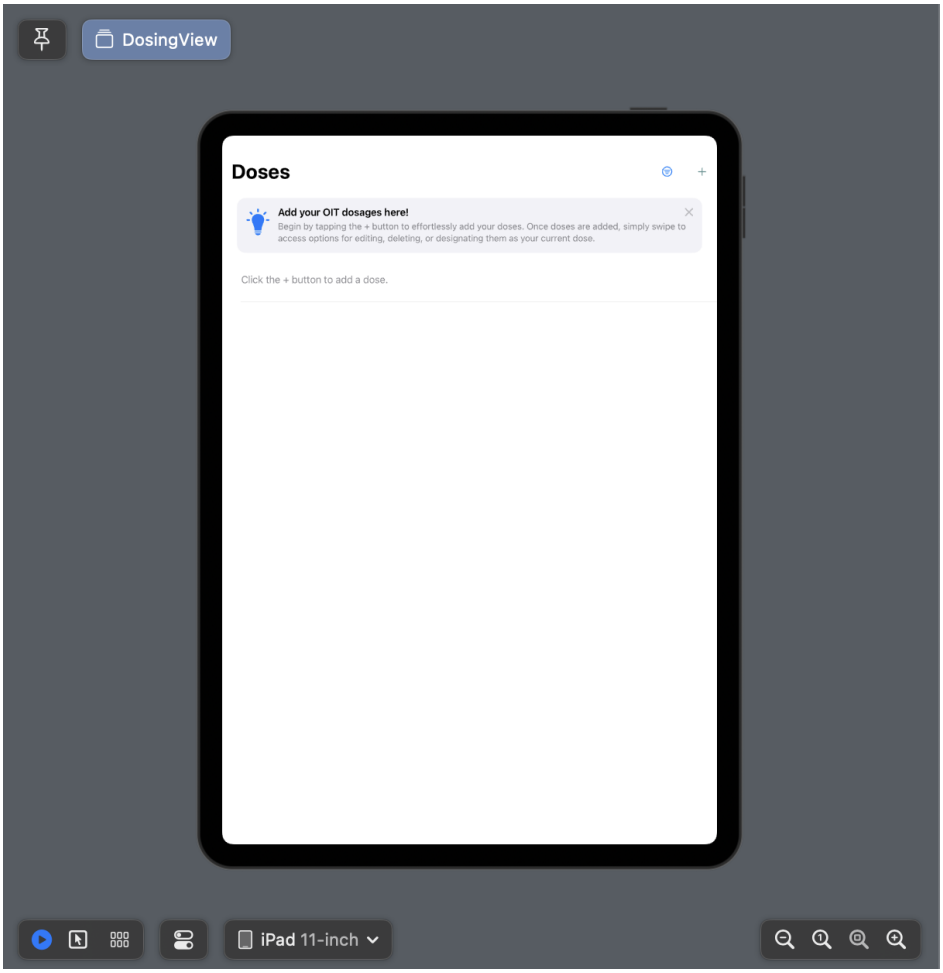


Figure 10.5: Swift UI Canvas Device Variant

Additionally, I deployed the application onto my own phone and simulator devices, in order to test the UI in a real world scenario. This was a very helpful way to test the UI and visuals, as certain issues only replicated on device and didn't show up in the Xcode previews.

By conducting comprehensive visual testing, I aimed to guarantee a visually cohesive and appealing user experience.

10.2 XCTests and XCUITests

XCTests and XCUITests served as integral tools for validating the functional aspects of the application throughout its development. These testing frameworks facilitated the creation and execution of unit tests, ensuring the reliability and correctness of the underlying codebase. Regularly running XCTests enabled me to catch and address potential bugs or regressions early in the development process, contributing to the overall stability of the app. Additionally, the use of XCUITests allowed for the automated testing of the app's user interface, simulating user interactions and verifying

the correct behavior of UI components.

10.3 User Testing

The user testing phase involved a personalized and hands-on approach to evaluating the application's practical utility.

As a patient undergoing daily maintenance doses of Oral Immunotherapy, I immersed myself in the user experience by utilizing the app in my daily routine. This enabled me to assess the real-world applicability of the application, focusing on aspects such as ease of use, convenience, and the integration of the app into my daily life. Throughout this process, I meticulously logged doses and symptoms, providing valuable insights into the app's functionality and user-friendlines

Although I didn't conduct a formal user study with other users, I did thoroughly design a study below, which could be implemented as a next step.

10.3.1 User Study

To extend the evaluation of the Oral Immunotherapy application beyond individual testing, a formal user study is proposed. The study aims to gather diverse feedback from a representative user population, ensuring a comprehensive understanding of the app's usability, effectiveness, and overall user satisfaction.

Objectives

The primary objectives of the user study are as follows:

- **Usability Assessment:** Evaluate the ease of use and user-friendliness of the application, focusing on navigation, feature discoverability, and overall interaction.
- **Effectiveness Evaluation:** Assess the effectiveness of the app in aiding users with daily maintenance doses of Oral Immunotherapy, emphasizing its ability to provide accurate information, timely reminders, and useful insights.
- **Integration into Daily Life:** Examine how well the application integrates into users' daily routines and lifestyles, considering factors such as convenience, time efficiency, and overall impact on their oral immunotherapy management.
- **Feedback Collection:** Gather qualitative and quantitative feedback from participants to identify strengths, weaknesses, and areas for improvement in the application

Participant Recruitment

Participants for the user study will be recruited from the target user demographic of individuals undergoing oral immunotherapy. Recruitment will aim for diversity in age, technological proficiency, and length of time they have been undergoing oral immunotherapy.

Methodology

The user study will adopt a mixed-methods approach, combining quantitative surveys and qualitative interviews to provide a comprehensive assessment.

- **Pre-Study Questionnaire:** Participants will complete a pre-study questionnaire to gather baseline information, including demographics, technological proficiency, and any prior experience with oral immunotherapy management apps.
- **Task-based Testing:** Participants will be given a set of predefined tasks within the application to perform. These tasks will cover essential functionalities, such as setting up their account, logging doses, and accessing relevant information.
- **Post-Task Surveys:** After completing each task, participants will provide feedback through structured surveys, rating their experience based on predefined criteria.
- **Interviews:** A subset of participants will be selected for in-depth interviews to explore their overall impressions, challenges encountered, and suggestions for improvement. These interviews will provide qualitative insights into user experiences and perceptions.

Data Analysis

Quantitative data from surveys will be analyzed using statistical tools to derive overall performance metrics and identify patterns. Qualitative data from interviews will be analyzed thematically to uncover nuanced feedback and user narratives.

10.3.2 Conclusion

While my personal testing provided valuable insights, the proposed user study is designed to offer a more extensive and diverse evaluation of the Oral Immunotherapy application. Implementing this study as the next step will not only enhance the application's refinement based on collective user feedback but also contribute to its broader acceptance and effectiveness within the target user population.

Chapter 11

Societal Issues

When developing a project like this, it is crucial to consider a broad spectrum of societal issues to ensure responsible and ethical innovation. This chapter explores various dimensions, including ethical, social, political, economic, health and safety, manufacturability, sustainability, environmental impact, usability, lifelong learning, and compassion.

11.1 Ethical

Ethical considerations are fundamental to the creation and utilization of any medical application. The oral immunotherapy application adheres to strict ethical standards in terms of user privacy, data security, and informed consent. The collection and handling of user data prioritize confidentiality and comply with relevant data protection regulations. Additionally, the application does not promote any form of discrimination or bias, ensuring fair and equitable access to its benefits. A full, more detailed ethical analysis of the application can be found in Chapter 12.

11.2 Social

The social dimension of the oral immunotherapy application focuses on its broader impact on individuals and the community. While the app does not include forums or community features, it still contributes significantly to the social aspect of oral immunotherapy management. By providing a tool for individuals to effectively manage their treatment, the app promotes a sense of autonomy and self-efficacy. This empowerment can lead to improved social interactions and a more active engagement with one's personal health journey. Moreover, the app facilitates better communication between patients and healthcare providers, fostering a collaborative and supportive relationship within the broader healthcare system. The social impact of the application extends beyond user interactions, playing a vital role in enhancing the overall quality of life for individuals undergoing oral immunotherapy.

11.3 Political

Political considerations encompass the alignment of the oral immunotherapy app with healthcare policies and regulations. The application complies with relevant medical standards and guidelines, contributing to the political goal of enhancing patient care and treatment adherence.

11.4 Economic

Economic factors focus on the financial implications of developing and adopting the oral immunotherapy app. The application is designed to be freely accessible to a wide range of users. Additionally, by promoting self-management and reducing healthcare visits, the app has the potential to alleviate economic burdens associated with oral immunotherapy treatments.

11.5 Health and Safety

Health and safety considerations are paramount in the development of a medical application. The app prioritizes user safety by providing accurate information, timely reminders, and emergency protocols. The app encourages responsible self-management while emphasizing the importance of consulting healthcare professionals for critical decisions. Finally, the application complies with Apple Health's constraints and standards, ensuring the data is stored safely and accurately.

11.6 Manufacturability

Manufacturability considerations revolve around the scalability and efficiency of the app's production and distribution. The oral immunotherapy app is designed to be easily scalable, with updates and improvements seamlessly integrated into the user experience. The manufacturing process, in this context, involves software development practices that prioritize reliability, security, and ease of deployment.

11.7 Sustainability

Sustainability considerations encompass the long-term viability and relevance of the oral immunotherapy app. The application is built with a modular and adaptable architecture, allowing for continuous updates and enhancements to meet evolving healthcare needs. Sustainable development practices ensure the app remains effective and relevant in the ever-changing landscape of oral immunotherapy.

11.8 Environmental Impact

The environmental impact of the oral immunotherapy app is predominantly associated with its digital nature, minimizing physical waste and resource consumption. By promoting digital interactions and reducing the need for physical documentation, the app aligns with environmentally conscious practices.

11.9 Usability

Usability is a crucial societal factor that influences the app's acceptance and effectiveness. The oral immunotherapy app prioritizes a user-friendly interface, intuitive navigation, and accessibility features to cater to a diverse user base. Enhancing usability contributes to the app's overall positive societal impact by ensuring broad accessibility and inclusivity.

11.10 Lifelong learning

Lifelong learning considerations emphasize the app's role in providing continuous education and support to users throughout their oral immunotherapy journey. The application incorporates educational resources, updates, and relevant information to foster ongoing learning and empowerment for users, promoting informed decision-making and self-management.

11.11 Compassion

Compassion is woven into the fabric of the oral immunotherapy app, recognizing the challenges individuals face in managing their oral immunotherapy treatments. The app is designed not only to provide practical support but also to empathize with users, acknowledging the emotional and physical aspects of their journey. Through features like personalized reminders and motivational messages, the app aims to instill a sense of compassion and understanding in its interaction with users.

Chapter 12

An Ethical Analysis

Ethical considerations play a crucial role in engineering projects, particularly in the realm of healthcare and medical technology. Engineering endeavors, especially those aimed at improving healthcare outcomes, inherently involve decisions that impact individuals' well-being, safety, and autonomy. Therefore, I seek to explore these ethical dimensions within the context of OIT application development, explaining how I designed a solution that not only enhances patient experience and treatment outcomes but also aligns with ethical principles governing healthcare innovation.

The following sections will outline the ethical justification for my project, identify significant ethical issues, provide reasons for decisions and actions, draw on ethical resources, and conclude with reflections on the ethical implications of the project for the broader healthcare landscape. Throughout this paper, I will draw upon established ethical frameworks, engineering codes of ethics, and additional ethical concepts to ensure a comprehensive analysis of the ethical considerations surrounding the development and implementation of the OIT application.

12.1 Ethical Justification for the Project

The very foundation of my project is built upon ethics – particularly the ethical principle of beneficence, which underscores the moral obligation to act for the benefit of others, promoting their well-being and preventing harm.

As philosopher Immanuel Kant eloquently stated, "Act in such a way that you treat humanity, whether in your own person or in the person of any other, never merely as a means to an end, but always at the same time as an end" [23]. This principle resonates deeply in the fields of healthcare and technology, where a commitment to improving human welfare should guide innovations. The development of my application embodies this principle by seeking to enhance the lives of individuals undergoing oral immunotherapy. By providing a user-friendly platform for dose tracking, symptom monitoring, and educational resources, my project aims to empower patients to manage their condition and care effectively. In doing so, it not only promotes patients' physical well-being but also acknowledges their autonomy and agency in healthcare decision-making.

Shifting gears to a utilitarian perspective, it is apparent that my application serves to maximize utility and happiness

for the greatest number of people. John Stuart Mill famously posited, "The creed which accepts as the foundation of morals, Utility, or the Greatest Happiness Principle, holds that actions are right in proportion as they tend to promote happiness, wrong as they tend to produce the reverse of happiness" [24]. From this standpoint, my project is justified by its potential to alleviate suffering and enhance the quality of life for a significant portion of the population affected by food allergies. Furthermore, as outlined in the Introduction, existing systems for logging OIT symptoms and doses rely on cumbersome handwritten logs and papers, causing frustration for both users managing the information and doctors analyzing it. Introducing a centralized digital platform to collect and aggregate this data not only maximizes happiness for patients by simplifying their experience, but also for healthcare providers, streamlining data management and analysis processes.

Drawing inspiration from contemporary philosopher Martha Nussbaum, my project also reflects a commitment to social justice by addressing disparities in healthcare access and information. As Nussbaum asserts, "Central human capabilities provide the normative focus for political principles and public policy" [25]. By facilitating access to vital resources and information through a mobile application, my project strives to promote equity in healthcare and empower individuals to advocate for their own well-being.

In essence, the ethical justification for my project extends beyond mere technological innovation – it embodies a commitment to compassion, autonomy, and social justice. By drawing upon philosophical insights on beneficence and social justice, my project seeks to contribute meaningfully to the advancement of healthcare and the promotion of human rights.

12.2 Identification of Significant Ethical Issues

The development and implementation of my application necessitate careful consideration of several significant ethical considerations. These encompass privacy and data security, informed consent, equitable access, and the accuracy and reliability of information. Each of these areas presents unique challenges and ethical dilemmas that must be addressed to ensure the ethical integrity of the application and the well-being of its users.

12.2.1 Privacy and Data Security

The issue of privacy and data security within the OIT application is multifaceted and demands careful deliberation. With the collection and storage of sensitive health information, users inherently entrust the application with personal data that can include medical history, allergy profiles, and treatment progress. This data is not only personal but also revealing of vulnerabilities and health conditions. Therefore, the responsibility falls on me, as the developer, to implement stringent security measures to safeguard this information against unauthorized access, breaches, or misuse.

Central to addressing privacy concerns is adherence to regulatory frameworks, such as the Health Insurance Portability and Accountability Act (HIPAA). Compliance with HIPAA standards ensures that patient confidentiality is

upheld, and appropriate safeguards are in place to protect health information from unauthorized disclosure. This includes measures such as encryption, access controls, audit trails, and regular security audits to mitigate the risk of data breaches.

Moreover, transparency in data handling practices is paramount to maintaining user trust and confidence in the OIT application. Users should be provided with clear and accessible information about how their data is collected, stored, and used within the application. This includes details on data retention policies and procedures for accessing or deleting personal information. By empowering users with knowledge and control over their data, the application should foster a sense of agency and accountability in its privacy practices.

12.2.2 Informed Consent

Informed consent stands as a cornerstone of ethical practice in healthcare technology, as well as computing in general. The ACM Code of Ethics states, "Computing professionals should establish transparent policies and procedures that allow individuals to understand what data is being collected and how it is being used, to give informed consent for automatic data collection, and to review, obtain, correct inaccuracies in, and delete their personal data" [26] This principle embodies the principle of respect for individuals' autonomy, acknowledging their right to make informed decisions about their participation in the use of the application and the sharing of their data.

When obtaining informed consent, it is vital that users are provided with comprehensive and comprehensible information about how their data will be collected, stored, and utilized within the application. This includes transparency regarding the purposes for which their data will be used, any potential risks or limitations associated with data sharing, and the mechanisms in place to safeguard their privacy and confidentiality. By arming users with this knowledge, they are better equipped to make informed decisions about their participation in the application and the extent to which they are comfortable sharing their personal information.

Furthermore, obtaining informed consent involves more than just providing information; it also requires ensuring that users have the capacity and opportunity to comprehend and deliberate upon the information presented to them. This is particularly relevant in the context of healthcare technology, where users may vary in their level of health literacy, technological proficiency, and cognitive abilities. As such, efforts should be made to employ clear language, visual aids, and accessible formats to facilitate understanding and decision-making.

Finally, the process of obtaining informed consent should be ongoing and iterative, rather than a one-time event. Users should be allowed to revisit and revise their consent preferences over time, as their circumstances, preferences, and understanding of the application evolve. This dynamic approach to informed consent not only respects users' autonomy but also acknowledges the fluid and evolving nature of their relationship with the application and their personal health information.

12.2.3 Equitable Access

Equitable access to my application is crucial to ensure that individuals of varying ages and technological backgrounds can benefit from its features. Age and digital literacy can significantly impact individuals' ability to access and utilize the application effectively. Given that many oral immunotherapy patients are young, and it's plausible that they, or their parents acting on their behalf, will engage with the application, accommodating varying levels of digital literacy becomes particularly pertinent in my project's design and usability. Another challenge lies in making the application accessible to diverse user groups, including those who may have limited access to technology or face barriers due to socioeconomic factors.

For older adults or individuals with limited digital literacy, the application should feature intuitive interfaces, clear instructions, and user-friendly design elements to facilitate ease of use. Providing comprehensive user support, such as tutorials or helplines, can further assist users in navigating the application regardless of their technological proficiency. Furthermore, considerations of affordability and cost should be addressed to ensure that my application remains accessible to individuals across socioeconomic backgrounds. Offering the application free of charge can help mitigate financial barriers and ensure equitable access for all.

In summary, ensuring equitable access to my application requires proactive measures to address disparities in digital literacy, socioeconomic status, and more. By prioritizing user accessibility and inclusivity in design and implementation, the application can effectively reach and benefit individuals of diverse backgrounds and circumstances.

12.2.4 Accuracy and Reliability of Information

As mentioned in the Introduction, my application serves as an educational resource for users, providing information about food allergies, oral immunotherapy, and related topics. Ensuring the accuracy and reliability of the information presented is paramount to prevent misinformation and promote informed decision-making among users. Regular updates and validation of content by reputable sources are essential to maintain the integrity of the educational materials.

12.3 Ethical Design Rationale

In light of the significant ethical issues identified in the previous section, I made several decisions regarding the development and features of my application to mitigate ethical quandaries. These decisions were guided by principles of privacy protection, user empowerment, accessibility, and accuracy of information. Below, I elaborate on the rationale behind each decision.

12.3.1 Integration with Apple Health and Other Apple Frameworks

The decision to integrate my app with Apple Health and other Apple frameworks, such as Core Data, demonstrates my commitment to protecting user privacy and ensuring data security within my application. By integrating and

aligning with Apple's ecosystem, which is renowned for its stringent privacy policies and robust security measures, my application adopts a proactive approach to safeguarding user data. Apple Health serves as a centralized repository for health-related information, allowing users to securely store and manage their medical data while maintaining control over its access and usage.

One of the key advantages of integrating with Apple Health is the assurance it provides users regarding the confidentiality and integrity of their sensitive health information. Apple has established itself as a trusted steward of user data, implementing encryption, access controls, and other advanced security mechanisms to protect against unauthorized access or breaches. By leveraging Apple's privacy infrastructure, my application can offer users a heightened level of confidence in the protection of their personal health data, fostering trust and credibility in its platform.

Additionally, integration with Core Data enhances the efficiency and reliability of data management within the OIT application. Core Data provides a robust and scalable solution for storing, querying, and manipulating application data, ensuring optimal performance and data integrity. By leveraging Core Data's capabilities, the application can streamline the data storage and retrieval processes while adhering to best practices for data security and privacy.

In essence, the decision to integrate with Apple Health and other Apple frameworks reflects a commitment to prioritizing user privacy and data security within the application. By aligning with Apple's ecosystem and leveraging its established privacy infrastructure, the application can offer users a trusted and secure platform for managing their health information.

12.3.2 Use of Emojis and Images for Enhanced Understanding

I decided to incorporate emojis and images throughout my application due to its diverse user base, which has the potential to span across different ages and backgrounds.

For users who may struggle with textual information due to language barriers, age, or limited literacy skills, emojis and images provide an alternative means of communication that is intuitive and easily understood. Emojis, in particular, convey emotions, actions, and concepts in a concise and visually appealing manner, allowing users to quickly interpret and relate to the content presented. By incorporating emojis strategically throughout the application, the app creates a more inclusive and user-friendly experience, catering to the needs of a diverse audience.

Moreover, the use of images within the application serves to complement textual information, reinforcing key concepts and enhancing comprehension. Visual representations can convey information more effectively than text alone, especially for complex or abstract ideas. By integrating images that illustrate concepts, procedures, or instructions, the application provides users with additional support in understanding and retaining information, regardless of their age or educational background.

12.3.3 Inclusion of Tips to Guide Users

The deliberate inclusion of tips within the OIT application represents a commitment to empowering users and enhancing their experience by offering valuable guidance and support. Recognizing the complexity of the application's features and functionalities, these tips serve as practical suggestions and insights aimed at helping new users navigate and utilize the application effectively.

The application's tips also work to promote informed decision-making and self-management of health among users. By providing users with actionable advice and recommendations, the application equips them with the knowledge and tools needed to make informed choices regarding their health and treatment journey. Whether it's tips on setting up the app, tracking progress, or adhering to treatment protocols, these insights enable users to maximize the benefits derived from the application and optimize their overall health outcomes.

Furthermore, the inclusion of tips reflects a user-centered approach to design and development, placing the needs and preferences of users at the forefront. By anticipating common challenges or questions that users may encounter, the application proactively addresses these concerns through timely and relevant tips. This proactive support not only enhances the user experience but also fosters a sense of confidence and autonomy among users in managing their health.

12.3.4 Creation of a User Guide

The creation of a comprehensive user guide for my application underscores a commitment to transparency, user empowerment, and informed consent. The user guide empowers users with the knowledge and understanding needed to navigate the application effectively. By offering detailed explanations of the application's features and functionalities, users are equipped to make informed choices about how they engage with the application and manage their health data. Moreover, the user guide serves as a reference tool, allowing users to access information about the application's capabilities and privacy practices at their convenience.

And, importantly, the user guide plays a critical role in facilitating informed consent among users. By providing transparency regarding data collection, storage, and utilization practices, the user guide ensures that users are fully aware of how their personal information is being managed within the application. This transparency enables users to make conscious decisions about their participation in the application and the extent to which they are comfortable sharing their data.

12.3.5 Making the App Free and Available on the App Store

The decision to offer my application free of charge and make it readily available on the App Store is driven by a commitment to promoting equitable access and ensuring that individuals from diverse socioeconomic backgrounds can benefit from its features without encountering any cost-related barriers. By offering the application for free,

financial constraints are eliminated as a barrier to access, ensuring that individuals of varying economic means can avail themselves of its functionalities. This approach aligns with principles of social justice and inclusivity, ensuring that essential health resources are accessible to all, regardless of their ability to pay.

Additionally, making my application available on the App Store enhances its accessibility and reach, as the platform serves as a central hub for users to discover and download a wide range of applications. By leveraging the widespread availability and convenience of the App Store, the application can reach a broader audience and fulfill its mission of providing valuable health resources to as many individuals as possible.

Offering the application on the App Store also ensures compliance with platform guidelines and standards, further enhancing user trust and credibility. By adhering to App Store regulations, the application demonstrates a commitment to quality, security, and user privacy, thereby instilling confidence in users regarding the integrity and reliability of the platform.

In addition to promoting equitable access for individuals, offering the application free of charge also opens avenues for healthcare providers, including doctors' offices, to distribute the app to their patients. By removing the financial barrier associated with app acquisition, healthcare professionals can readily recommend and provide access to the OIT application as a supplementary tool for patient care.

And, for doctors' offices, the ability to offer the application to patients aligns with a patient-centered approach to healthcare delivery. Healthcare providers can leverage the application as a resource to complement traditional treatment methods, empowering patients to take a more active role in managing their health. By equipping patients with access to the application, doctors' offices can enhance patient education, facilitate treatment adherence, and promote better health outcomes.

Furthermore, the availability of the application free of charge enables doctors' offices to integrate it seamlessly into their existing workflows and patient care protocols. Healthcare professionals can incorporate the application into patient consultations, providing personalized recommendations and guidance tailored to individual treatment plans and goals. This integration fosters a collaborative approach to healthcare, where patients and providers work together towards shared treatment objectives.

As you can see, offering my application for free encourages widespread adoption and utilization among healthcare providers and their patients. By reducing barriers to access, doctors' offices can promote the use of the application as a standard component of care for patients undergoing oral immunotherapy. This widespread adoption not only enhances patient engagement and satisfaction but also facilitates data collection and monitoring, enabling healthcare providers to track patient progress and adjust treatment plans accordingly.

12.3.6 Support for Different Font Sizes and Accessibility Features

Finally, the decision to incorporate support for different font sizes and accessibility features within the application underscores a commitment to promoting inclusivity and ensuring that all users can effectively engage with the application, regardless of their individual accessibility needs.

By offering support for different font sizes, the application acknowledges the diverse range of users who may require adjustments to accommodate visual impairments or preferences. This customization empowers users to personalize their experience with the application, ensuring that they can comfortably read and interact with content without encountering any barriers or difficulties. Whether users require larger font sizes for improved readability or prefer smaller font sizes for increased information density, the application's flexibility ensures that their needs are accommodated.

In addition to supporting different font sizes, the application also incorporates other accessibility features to enhance usability for individuals with diverse needs. By prioritizing accessibility in its design and functionality, the application ensures that all users, regardless of their abilities or limitations, can access its content and features without encountering any accessibility-related barriers.

Chapter 13

Conclusion

In conclusion, this project represents a significant step towards addressing the unmet needs of OIT patients through the development of an iPhone application. By delving into the challenges faced by individuals with life-threatening food allergies, the importance of innovative solutions is underscored. With a deeply personal connection to the subject matter, the project's overarching goal was established: to create an accessible, valuable tool that supports patients on their desensitization journey.

The comprehensive exploration of functional and non-functional requirements, detailed in Chapter 2, laid the foundation for the application's design. Key features such as integration with the Apple Health app, daily dose tracking, and intuitive data visualization were prioritized to ensure the app's usability and effectiveness. Moreover, the user-centric approach was further accentuated through the inclusion of educational resources, proactive notifications, and a keen focus on accessibility.

Throughout the project, numerous lessons were learned, ranging from technical intricacies to broader insights into patient needs and user experience. Notably, gaining proficiency in iOS development, understanding HealthKit permissions, and mastering data persistence techniques were among the technical learnings. Furthermore, insights into the challenges and nuances of managing food allergies, as well as the importance of empathy-driven design, significantly enriched the project's outcomes.

While the developed iPhone application represents a significant advancement in addressing the needs of OIT patients, there are still areas for improvement and refinement. Despite rigorous testing and user feedback, challenges such as ensuring seamless integration with diverse healthcare ecosystems and enhancing user engagement remain pertinent. Additionally, while the app strives for inclusivity, ongoing efforts are needed to address accessibility concerns and accommodate diverse user demographics effectively.

Looking ahead, future work could explore avenues for enhanced personalization, leveraging machine learning algorithms for predictive analytics, and fostering community engagement within the app. Furthermore, continued collaboration with healthcare professionals and allergy specialists can provide valuable insights for refining the app's

features and expanding its impact. Ultimately, the journey towards improving the lives of OIT patients is ongoing, and this project serves as a pivotal step towards that endeavor.

Appendix A

Code Files

A.1 OITApp.swift

```swift
//
//  SeniorDesignApp.swift
//  SeniorDesign
//
//  Created by Maddie on 9/25/23.
//

import SwiftUI
import LocalAuthentication

let hasAppBeenOpenedBeforeKey = "HasAppBeenOpenedBefore"

class AppState: ObservableObject {
    @AppStorage(hasAppBeenOpenedBeforeKey) var hasAppBeenOpenedBefore: Bool = false
}

@main
struct OITApp: App {
    // MARK: View Models
    @StateObject var profileViewModel = ProfileViewModel()
    @StateObject var healthKitViewModel = HealthKitViewModel()
    @StateObject var doseViewModel = DoseViewModel()
    @StateObject var profileImageViewModel = ProfileModel()
    @StateObject var appState = AppState()

    @State private var isUnlocked = false

    func authenticate() {
        let context = LAContext()
        var error: NSError?

        if context.canEvaluatePolicy(.deviceOwnerAuthenticationWithBiometrics, error: &error) {
            let reason = "Allow OIT to use a passcode, TouchID, or FaceID to protect your data."

            context.evaluatePolicy(.deviceOwnerAuthentication, localizedReason: reason) { success
    , authenticationError in
                if success {
                    isUnlocked = true
                    profileViewModel.loadProfileData()
                    profileImageViewModel.loadProfileImage()
                    doseViewModel.loadDoses()
                } else {
                    // Error
                }
            }
        } else if context.canEvaluatePolicy(.deviceOwnerAuthentication, error: &error) {
```

```
            let reason = "Allow OIT to use a passcode, TouchID, or FaceID to protect your data."

            context.evaluatePolicy(.deviceOwnerAuthentication, localizedReason: reason) { success, authenticationError in
                if success {
                    isUnlocked = true
                    profileViewModel.loadProfileData()
                    profileImageViewModel.loadProfileImage()
                    doseViewModel.loadDoses()
                } else {
                    // Error
                }
            }
        }
    }

    // MARK: View of App
    var body: some Scene {
        WindowGroup {
            if appState.hasAppBeenOpenedBefore {
                TabbedApplicationView()
                    .environmentObject(profileViewModel)
                    .environmentObject(healthKitViewModel)
                    .environmentObject(doseViewModel)
                    .environmentObject(profileImageViewModel)
                    .onAppear(perform: {
                        profileViewModel.loadProfileData()
                        profileImageViewModel.loadProfileImage()
                        doseViewModel.loadDoses()
                        if !isUnlocked && profileViewModel.profileData.useFaceID {
                            authenticate()
                        }
                    })
                    .onDisappear(perform: {
                        isUnlocked = false
                    })
            } else {
                GetStartedView()
                    .environmentObject(profileViewModel)
                    .environmentObject(healthKitViewModel)
                    .environmentObject(doseViewModel)
                    .environmentObject(appState)
                    .environmentObject(profileImageViewModel)
                    .onAppear {
                        profileViewModel.loadProfileData()
                        profileImageViewModel.loadProfileImage()
                        doseViewModel.loadDoses()
                    }
            }
        }
    }
}
```

A.2 TabbedApplicationView.swift

```
//
//  TabbedApplicationView.swift
//  SeniorDesign
//
//  Created by Maddie on 8/20/23.
//

import SwiftUI

struct TabbedApplicationView: View {
    // MARK: View Models
    @EnvironmentObject var profileViewModel: ProfileViewModel
```

```
    @EnvironmentObject var healthKitViewModel: HealthKitViewModel
    @EnvironmentObject var doseViewModel: DoseViewModel
    @EnvironmentObject var profileImageViewModel: ProfileModel

    var body: some View {
        TabView {
            // MARK: Home Tab
            NavigationStack {
                HomeView()
                    .environmentObject(profileViewModel)
                    .environmentObject(healthKitViewModel)
                    .environmentObject(doseViewModel)
                    .environmentObject(profileImageViewModel)
            }
            .tabItem {
                Label("Today", systemImage: "calendar")
            }

            // MARK: Insights Tab
            NavigationStack {
                InsightsView(healthKitViewModel: healthKitViewModel)
                    .environmentObject(profileViewModel)
                    .environmentObject(healthKitViewModel)
                    .environmentObject(doseViewModel)
            }
            .tabItem {
                Label("Insights", systemImage: "chart.line.uptrend.xyaxis")
            }

            // MARK: Dosing Tab
            NavigationStack {
                DosingView()
                    .environmentObject(profileViewModel)
                    .environmentObject(healthKitViewModel)
                    .environmentObject(doseViewModel)
            }
            .tabItem {
                Label("Dosing", systemImage: "pills.fill")
            }

            // MARK: Education Tab
            NavigationStack {
                EducationView()
                    .environmentObject(profileViewModel)
                    .environmentObject(healthKitViewModel)
                    .environmentObject(doseViewModel)
            }
            .tabItem {
                Label("Education", systemImage: "graduationcap.fill")
            }
        }
        .tint(.darkTeal)
        .onAppear(perform: {
            profileViewModel.loadProfileData()
            profileImageViewModel.loadProfileImage()
            doseViewModel.loadDoses()
        })
    }
}

// MARK: Preview
#Preview {
    TabbedApplicationView()
        .environmentObject(ProfileViewModel())
        .environmentObject(HealthKitViewModel())
        .environmentObject(DoseViewModel())
}
```

A.3 Extensions.swift

```swift
//
//  Extensions.swift
//  SeniorDesign
//
//  Created by Maddie on 9/25/23.
//

import Foundation
import SwiftUI

extension Color {
    static var lightGrey: Color { Color(hex: "FAFAFA") }
    static var mediumGrey: Color { Color(hex: "D3D3D3") }
    static var darkGrey: Color { Color(hex: "222222") }
    static var grey: Color { Color(hex: "D5D5D5") }
    static var darkTeal: Color { Color(hex: "5A8F8F") }
    static var darkerTeal: Color { Color(hex: "0a5757") }
    static var lightTeal: Color { Color(hex: "C3E2E7") }
    static var lightBlue: Color { Color(hex: "E9F5F9") }
    static var lightYellow: Color { Color(hex: "FFFEC8") }

    init(hex: String) {
        let scanner = Scanner(string: hex)
        var rgb: UInt64 = 0

        scanner.scanHexInt64(&rgb)

        let red = Double((rgb & 0xFF0000) >> 16) / 255.0
        let green = Double((rgb & 0x00FF00) >> 8) / 255.0
        let blue = Double(rgb & 0x0000FF) / 255.0

        self.init(red: red, green: green, blue: blue)
    }

    static var mutedRandom: Color {
        let red = CGFloat.random(in: 0.3...0.7)
        let green = CGFloat.random(in: 0.3...0.7)
        let blue = CGFloat.random(in: 0.3...0.7)

        return Color(red: Double(red), green: Double(green), blue: Double(blue))
    }
}

extension String {
    func splitCamelCase() -> String {
        var formattedString = self.replacingOccurrences(of: "HKCategoryTypeIdentifier", with: "")

        let pattern = "(\\w)([A-Z])"
        let regex = try! NSRegularExpression(pattern: pattern, options: [])
        let range = NSRange(location: 0, length: formattedString.utf16.count)

        let matches = regex.matches(in: formattedString, options: [], range: range)
        for match in matches.reversed() {
            let index = formattedString.index(formattedString.startIndex, offsetBy: match.range(
    at: 2).location)
            if index < formattedString.endIndex {
                formattedString.insert(" ", at: index)
            }
        }

        return formattedString
    }
}
```

A.4 AllergenWithDoses.swift

```swift
//
//  AllergenWithDoses.swift
//  SeniorDesign
//
//  Created by Maddie on 3/7/24.
//

import Foundation

struct AllergenWithDoses: Identifiable, Hashable {
    static func == (lhs: AllergenWithDoses, rhs: AllergenWithDoses) -> Bool {
            return lhs.id == rhs.id
        }

        func hash(into hasher: inout Hasher) {
            hasher.combine(id)
        }

    let id = UUID()
    let allergen: String
    let doses: [Dose]
}
```

A.5 AntihistamineDose.swift

```swift
//
//  AntihistamineDose.swift
//  SeniorDesign
//
//  Created by Maddie on 3/7/24.
//

import Foundation

struct AntihistamineDose: Hashable {
    let name: String
    let dose: String
}
```

A.6 AntihistamineDoseTransformer.swift

```swift
//
//  AntihistamineDoseTransformer.swift
//  SeniorDesign
//
//  Created by Maddie on 3/7/24.
//

import Foundation

class AntihistamineDoseTransformer: NSSecureUnarchiveFromDataTransformer {
    override class func allowsReverseTransformation() -> Bool {
        return true
    }

    override class func transformedValueClass() -> AnyClass {
        return NSData.self
    }

    override func transformedValue(_ value: Any?) -> Any? {
        guard let data = try? NSKeyedArchiver.archivedData(withRootObject: value as?
    AntihistamineDose ?? AntihistamineDose(name: "", dose: "")) else {
            return nil
```

```
        }
        return data as NSData
    }

    override func reverseTransformedValue(_ value: Any?) -> Any? {
        guard let data = value as? Data else { return nil }
        return try? NSKeyedUnarchiver.unarchiveTopLevelObjectWithData(data) as? AntihistamineDose
    }
}
```

A.7 Dose.swift

```
//
//  Dose.swift
//  SeniorDesign
//
//  Created by Maddie on 3/7/24.
//

import Foundation

struct Dose: Identifiable, Codable {
    var id: UUID {
        return UUID()
    }
    var allergen: String
    var doseType: String
    var doseAmount: Double
    var halfDose: Double {
        return doseAmount / 2.0
    }
    var isCurrentDose: Bool // Add property to track current dose
}
```

A.8 DoseTransformer.swift

```
//
//  DoseRecords.swift
//  SeniorDesign
//
//  Created by Maddie on 12/28/23.
//

import Foundation

class DosesTransformer: NSSecureUnarchiveFromDataTransformer {
    override func transformedValue(_ value: Any?) -> Any? {
        guard let doses = value as? [String: String],
              let data = try? NSKeyedArchiver.archivedData(withRootObject: doses,
    requiringSecureCoding: false) else {
            return nil
        }
        return data as NSData
    }

    override func reverseTransformedValue(_ value: Any?) -> Any? {
        guard let data = value as? Data,
              let doses = try? NSKeyedUnarchiver.unarchiveTopLevelObjectWithData(data) as? [
    String: String] else {
            return nil
        }
        return doses
    }
}
```

A.9 DoseViewModel.swift

```swift
//
//  DoseViewModel.swift
//  SeniorDesign
//
//  Created by Maddie on 1/1/24.
//

import Foundation

class DoseViewModel: ObservableObject {
    @Published var doses: [Dose] = []

    var allergensWithDoses: [AllergenWithDoses] {
        let groupedDoses = Dictionary(grouping: doses, by: { $0.allergen })
        let sortedAllergens = groupedDoses.keys.sorted()
        return sortedAllergens.map { allergen in
            AllergenWithDoses(allergen: allergen, doses: groupedDoses[allergen]!)
        }
    }

    // Function to add a dose with an option to mark it as current
    func addDose(allergen: String, doseType: String, doseAmount: Double, isCurrentDose: Bool) {
        let newDose = Dose(allergen: allergen, doseType: doseType, doseAmount: doseAmount, isCurrentDose: isCurrentDose)
        doses.append(newDose)

        let halfDoseAmount = doseAmount / 2.0
        let newHalfDose = Dose(allergen: allergen, doseType: "Half Dose for \(doseType)", doseAmount: halfDoseAmount, isCurrentDose: false)
        doses.append(newHalfDose)

        saveDoses()
    }

    func saveDoses() {
        let encoder = JSONEncoder()
        if let encoded = try? encoder.encode(doses) {
            UserDefaults.standard.set(encoded, forKey: "Doses")
        }
    }

    func loadDoses() {
        if let savedDoses = UserDefaults.standard.data(forKey: "Doses") {
            let decoder = JSONDecoder()
            if let loadedDoses = try? decoder.decode([Dose].self, from: savedDoses) {
                doses = loadedDoses
            }
        }
    }

    // Function to set the current dose for a specific allergen
    func setCurrentDose(allergen: String, doseType: String) {
        for index in 0..<doses.count {
            if doses[index].allergen == allergen && doses[index].doseType == doseType {
                doses[index].isCurrentDose = true
            } else {
                doses[index].isCurrentDose = false
            }
        }
        saveDoses()
    }

    func deleteDose(_ dose: Dose) {
        if let index = doses.firstIndex(where: { $0.doseType == dose.doseType }) {
            doses.remove(at: index)
```

```
            saveDoses()
        }
    }

    func setAsCurrentDose(_ dose: Dose) {
        for index in 0..<doses.count {
            if doses[index].allergen == dose.allergen {
                doses[index].isCurrentDose = doses[index].doseType == dose.doseType
            }
        }
        saveDoses()
    }

    func doseTypesForSelectedAllergens(selectedAllergens: [String]) -> [String] {
        var uniqueDoseTypes: [String] = []

        for allergen in selectedAllergens {
            let dosesForAllergen = doses.filter { $0.allergen == allergen }
            print(dosesForAllergen)
            let doseTypesForAllergen = Set(dosesForAllergen.map { "\($0.doseType)      \((String(
    format: "%.1f", $0.doseAmount))) mg" })
            print(doseTypesForAllergen)
            uniqueDoseTypes.append(contentsOf: doseTypesForAllergen)
        }

        return Array(Set(uniqueDoseTypes))
    }

    func updateDose(_ existingDose: Dose, allergen: String, doseType: String, doseAmount: Double,
     isCurrentDose: Bool) {
        if let index = doses.firstIndex(where: { $0.id == existingDose.id }) {
            doses[index].allergen = allergen
            doses[index].doseType = doseType
            doses[index].doseAmount = doseAmount
            doses[index].isCurrentDose = isCurrentDose
            saveDoses()
        }
    }
}
```

A.10 ProfileData

```
//
//  ProfileData.swift
//  SeniorDesign
//
//  Created by Maddie on 3/7/24.
//

import Foundation

class ProfileData: ObservableObject, Codable {
    @Published var name: String = ""
    @Published var dateOfBirth: Date = Date()
    @Published var allergens: [String] = []
    @Published var shareDataWithAppleHealth: Bool = false
    @Published var useFaceID: Bool = false
    @Published var commonAllergens = ["Milk", "Eggs", "Fish", "Shellfish", "Soy", "Peanuts", "
    Almonds", "Brazil Nuts", "Cashews", "Coconuts", "Hazelnuts", "Macadamia Nuts", "Pecans", "
    Pine Nuts", "Pistachios", "Walnuts", "Wheat", "Sesame"]
    @Published var allergenEmojiMap: [String: String] = [
        "Peanuts": "    ",
        "Almonds": "    ",
        "Brazil Nuts": "    ",
        "Cashews": "    ",
        "Coconuts": "    ",
        "Hazelnuts": "    ",
```

```
        "Macadamia Nuts": "    ",
        "Pecans": "    ",
        "Pine Nuts": "    ",
        "Pistachios": "    ",
        "Walnuts": "    ",
        "Milk": "    ",
        "Eggs": "    ",
        "Wheat": "    ",
        "Soy": "    ",
        "Fish": "    ",
        "Shellfish": "    ",
        "Sesame": "    ",
    ]

    func isNut(_ allergen: String) -> Bool {
        let nuts = ["Almonds", "Brazil Nuts", "Cashews", "Macadamia Nuts", "Pine Nuts", "
    Pistachios", "Pecans", "Walnuts"]
        return nuts.contains(allergen)
    }

    enum CodingKeys: String, CodingKey {
        case name
        case dateOfBirth
        case allergens
        case shareDataWithAppleHealth
        case useFaceID
        case commonAllergens
        case allergenEmojiMap
    }

    required init(from decoder: Decoder) throws {
        let container = try decoder.container(keyedBy: CodingKeys.self)
        name = try container.decode(String.self, forKey: .name)
        dateOfBirth = try container.decode(Date.self, forKey: .dateOfBirth)
        allergens = try container.decode([String].self, forKey: .allergens)
        shareDataWithAppleHealth = try container.decode(Bool.self, forKey: .
    shareDataWithAppleHealth)
        useFaceID = try container.decode(Bool.self, forKey: .useFaceID)
        commonAllergens = try container.decode([String].self, forKey: .commonAllergens)
        allergenEmojiMap = try container.decode([String : String].self, forKey: .allergenEmojiMap
    )
    }

    func encode(to encoder: Encoder) throws {
        var container = encoder.container(keyedBy: CodingKeys.self)
        try container.encode(name, forKey: .name)
        try container.encode(dateOfBirth, forKey: .dateOfBirth)
        try container.encode(allergens, forKey: .allergens)
        try container.encode(shareDataWithAppleHealth, forKey: .shareDataWithAppleHealth)
        try container.encode(useFaceID, forKey: .useFaceID)
        try container.encode(commonAllergens, forKey: .commonAllergens)
        try container.encode(allergenEmojiMap, forKey: .allergenEmojiMap)
    }

    init() { }
}
```

A.11 ProfileImageModel.swift

```
import SwiftUI
import PhotosUI
import CoreTransferable
import Foundation

@MainActor
class ProfileModel: ObservableObject {

```

```
// MARK: - Profile Details

  @Published var firstName: String = ""
  @Published var lastName: String = ""
  @Published var aboutMe: String = ""

  // MARK: - Profile Image

  enum ImageState {
      case empty
  case loading(Progress)
  case success(Image)
  case failure(Error)
  }

  enum TransferError: Error {
      case importFailed
  }

  struct ProfileImage: Transferable {
      let image: Image

      static var transferRepresentation: some TransferRepresentation {
          DataRepresentation(importedContentType: .image) { data in
          #if canImport(AppKit)
              guard let nsImage = NSImage(data: data) else {
                  throw TransferError.importFailed
              }
              let image = Image(nsImage: nsImage)
              return ProfileImage(image: image)
          #elseif canImport(UIKit)
              guard let uiImage = UIImage(data: data) else {
                  throw TransferError.importFailed
              }
              let image = Image(uiImage: uiImage)
              return ProfileImage(image: image)
          #else
              throw TransferError.importFailed
          #endif
          }
      }
  }

  @Published private(set) var imageState: ImageState = .empty

  @Published var imageSelection: PhotosPickerItem? = nil {
      didSet {
          if let imageSelection {
              let progress = loadTransferable(from: imageSelection)
              imageState = .loading(progress)
          } else {
              imageState = .empty
          }
      }
  }

// MARK: - Private Methods

  private func loadTransferable(from imageSelection: PhotosPickerItem) -> Progress {
      return imageSelection.loadTransferable(type: ProfileImage.self) { result in
          DispatchQueue.main.async {
              guard imageSelection == self.imageSelection else {
                  print("Failed to get the selected item.")
                  return
              }
              switch result {
              case .success(let profileImage?):
                  self.imageState = .success(profileImage.image)
```

```
                case .success(nil):
                    self.imageState = .empty
                case .failure(let error):
                    self.imageState = .failure(error)
                }
            }
        }
    }

    private let profileImageKey = "profileImage"

    init() {
        loadProfileImage()
    }

    func saveProfileImage() {
        if case let .success(profileImage) = imageState {
            if let data = try? NSKeyedArchiver.archivedData(withRootObject: profileImage,
    requiringSecureCoding: false) {
                UserDefaults.standard.set(data, forKey: profileImageKey)
            }
        }
    }

    func loadProfileImage() {
        if let data = UserDefaults.standard.data(forKey: profileImageKey),
           let profileImage = try? NSKeyedUnarchiver.unarchiveTopLevelObjectWithData(data) as?
    Image {
            imageState = .success(profileImage)
        }
    }
}
```

A.12 ProfileImageViewModel.swift

```
import SwiftUI
import PhotosUI
import CoreTransferable
import Foundation

@MainActor
class ProfileModel: ObservableObject {
    // MARK: - Profile Image

    enum ImageState {
        case empty
        case loading(Progress)
        case success(Image)
        case failure(Error)
    }

    enum TransferError: Error {
        case importFailed
    }

    struct ProfileImage: Transferable {
        let image: Image

        static var transferRepresentation: some TransferRepresentation {
            DataRepresentation(importedContentType: .image) { data in
            #if canImport(AppKit)
                guard let nsImage = NSImage(data: data) else {
                    throw TransferError.importFailed
                }
                let image = Image(nsImage: nsImage)
                return ProfileImage(image: image)
            #elseif canImport(UIKit)
```

```
                guard let uiImage = UIImage(data: data) else {
                    throw TransferError.importFailed
                }
                let image = Image(uiImage: uiImage)
                return ProfileImage(image: image)
            #else
                throw TransferError.importFailed
            #endif
            }
        }
    }

    @Published private(set) var imageState: ImageState = .empty

    @Published var imageSelection: PhotosPickerItem? = nil {
        didSet {
            if let imageSelection {
                let progress = loadTransferable(from: imageSelection)
                imageState = .loading(progress)
            } else {
                imageState = .empty
            }
        }
    }

    // MARK: - Private Methods

    private func loadTransferable(from imageSelection: PhotosPickerItem) -> Progress {
        return imageSelection.loadTransferable(type: ProfileImage.self) { result in
            DispatchQueue.main.async {
                guard imageSelection == self.imageSelection else {
                    print("Failed to get the selected item.")
                    return
                }
                switch result {
                case .success(let profileImage?):
                    self.imageState = .success(profileImage.image)
                case .success(nil):
                    self.imageState = .empty
                case .failure(let error):
                    self.imageState = .failure(error)
                }
            }
        }
    }

    private let profileImageKey = "profileImage"

    init() {
        loadProfileImage()
    }

    func saveProfileImage() {
        if case let .success(profileImage) = imageState {
            if let data = profileImage.getUIImage(newSize: CGSize(width: 100, height: 100))?.
    pngData() {
                UserDefaults.standard.setValue(data, forKey: profileImageKey)
            }
        }
    }

    func loadProfileImage() {
        if let data = UserDefaults.standard.data(forKey: profileImageKey),
           let uiImage = UIImage(data: data){
            imageState = .success(Image(uiImage: uiImage))
        }
    }
}
```

```swift

extension Image {
    @MainActor
    func getUIImage(newSize: CGSize) -> UIImage? {
        let image = resizable()
            .scaledToFill()
            .frame(width: newSize.width, height: newSize.height)
            .clipped()
        return ImageRenderer(content: image).uiImage
    }
}
```

A.13 ProfileViewModel.swift

```swift
//
//  PersonalInformation.swift
//  SeniorDesign
//
//  Created by Maddie on 10/20/23.
//

import Combine
import Foundation

class ProfileViewModel: ObservableObject {
    @Published var profileData = ProfileData()
    @Published var selectedAllergens: Set<String> = []
    @Published var numAllergens = 0 // Variable to track the number of allergens

    func saveProfileData() {
        let encoder = JSONEncoder()
        if let encoded = try? encoder.encode(profileData) {
            UserDefaults.standard.set(encoded, forKey: "ProfileData")
        }
    }

    func loadProfileData() {
        if let savedProfileData = UserDefaults.standard.data(forKey: "ProfileData") {
            let decoder = JSONDecoder()
            if let loadedProfile = try? decoder.decode(ProfileData.self, from: savedProfileData)
    {
                profileData = loadedProfile
                numAllergens = loadedProfile.allergens.count
            }
        }
    }

    func addNewAllergen() {
        profileData.allergens.append("Select an Allergen")
        numAllergens += 1 // Increment the allergen count
        saveProfileData()
    }

    func removeSelectedAllergen(at index: Int) {
        if profileData.allergens.indices.contains(index) {
            profileData.allergens.remove(at: index)
            numAllergens -= 1 // Decrement the allergen count
            saveProfileData()
        }
    }


    func toggleAllergenSelection(_ allergen: String) {
        if selectedAllergens.contains(allergen) {
            selectedAllergens.remove(allergen)
        } else {
            selectedAllergens.insert(allergen)
```

```
        }
    }

    func isSelected(_ allergen: String) -> Bool {
        return selectedAllergens.contains(allergen)
    }
}
```

A.14 SelectedAllergensTransformer

```
//
//  SelectedAllergensTransformer.swift
//  SeniorDesign
//
//  Created by Maddie on 3/7/24.
//

import Foundation

class SelectedAllergensTransformer: NSSecureUnarchiveFromDataTransformer {
    override static var allowedTopLevelClasses: [AnyClass] {
        return [NSSet.self]
    }
}
```

A.15 SymptomDataManager.swift

```
import SwiftUI
import CoreData

class SymptomDataManager: ObservableObject {
    @MainActor
    @Published var symptomRecords: [Date: [String]] = [:]

    @MainActor func saveSymptoms(for date: Date, symptoms: [String]) {
        // Update the symptom records dictionary with the new symptoms for the given date
        symptomRecords[date] = symptoms

        let context = persistentContainer.viewContext
        let symptomRecord = SymptomRecord(context: context)
        symptomRecord.date = date
        symptomRecord.symptoms = symptoms as NSObject

        do {
            try context.save()
        } catch {
            print("Failed to save symptoms: \(error.localizedDescription)")
        }
    }

    @MainActor func fetchSymptoms(for date: Date) -> [String] {
        if let symptoms = symptomRecords[date] {
            // Return symptoms from the dictionary if available
            return symptoms
        } else {
            let context = persistentContainer.viewContext
            let fetchRequest: NSFetchRequest<SymptomRecord> = SymptomRecord.fetchRequest()
            fetchRequest.predicate = NSPredicate(format: "date == %@", date as CVarArg)

            do {
                let result = try context.fetch(fetchRequest)
                if let record = result.first {
                    if let fetchedSymptoms = record.symptoms as? [String] {
                        // Update the symptom records dictionary with the fetched symptoms for
    caching
                        symptomRecords[date] = fetchedSymptoms
```

```
                    return fetchedSymptoms
                }
            }
        } catch {
            print("Failed to fetch symptoms: \(error.localizedDescription)")
        }
    }
    return []
}

// Function to fetch all stored symptoms from Core Data
@MainActor func fetchAllSymptoms() -> [String] {
    let context = persistentContainer.viewContext
    let fetchRequest: NSFetchRequest<SymptomRecord> = SymptomRecord.fetchRequest()

    do {
        let result = try context.fetch(fetchRequest)
        var allSymptoms: [String] = []

        // Iterate through fetched records and populate symptomRecords dictionary
        for record in result {
            if let date = record.date, let symptoms = record.symptoms as? [String] {
                symptomRecords[date] = symptoms
                allSymptoms.append(contentsOf: symptoms)
            }
        }

        return allSymptoms
    } catch {
        print("Failed to fetch all symptoms: \(error.localizedDescription)")
        return []
    }
}

func fetchAllSymptomsWithoutRefresh() -> [String] {
    let context = persistentContainer.viewContext
    let fetchRequest: NSFetchRequest<SymptomRecord> = SymptomRecord.fetchRequest()

    do {
        let result = try context.fetch(fetchRequest)
        let allSymptoms = result.flatMap { ($0.symptoms as? [String]) ?? [] }
        return allSymptoms
    } catch {
        print("Failed to fetch all symptoms: \(error.localizedDescription)")
        return []
    }
}

lazy var persistentContainer: NSPersistentContainer = {
    let container = NSPersistentContainer(name: "SymptomDataModel")
    container.loadPersistentStores { _, error in
        if let error = error {
            fatalError("Failed to load persistent store: \(error)")
        }
    }
    return container
}()

func lastReactionDate() -> Date? {
        let context = persistentContainer.viewContext
        let fetchRequest: NSFetchRequest<SymptomRecord> = SymptomRecord.fetchRequest()
        fetchRequest.sortDescriptors = [NSSortDescriptor(key: "date", ascending: false)]
        fetchRequest.fetchLimit = 1

        do {
            let result = try context.fetch(fetchRequest)
            if let record = result.first {
                return record.date
```

```
                }
            } catch {
                print("Failed to fetch last reaction date: \(error.localizedDescription)")
            }

            return nil
        }
}
```

A.16 AddDoseView.swift

```
//
//  AddDoseView.swift
//  SeniorDesign
//
//  Created by Maddie on 10/20/23.
//

import SwiftUI

struct AddDoseView: View {
    // MARK: View Models
    @EnvironmentObject var doseViewModel: DoseViewModel
    @EnvironmentObject var profileViewModel: ProfileViewModel

    // MARK: Variables
    @Environment(\.presentationMode) var presentationMode
    @State private var selectedAllergen = ""
    @State private var doseName = ""
    @State private var doseAmount = ""
    @State private var isCurrentDose = false
    @State var editingDose: Dose?

    // MARK: Add Dose View
    var body: some View {
        NavigationStack {
            Form {
                Section(header: Text("Allergen")) {
                    Picker("Select Allergen", selection: $selectedAllergen) {
                        ForEach(profileViewModel.profileData.allergens, id: \.self) { allergen in
                            Text(allergen)
                        }
                    }
                    .onAppear {
                        if selectedAllergen.isEmpty, let firstAllergen = profileViewModel.
    profileData.allergens.first {
                            selectedAllergen = firstAllergen
                        }
                    }
                    .pickerStyle(DefaultPickerStyle())
                }

                Section(header: Text("Dose Information")) {
                    TextField("Dose Name", text: $doseName)
                    HStack {
                        Text("Dose")
                        Spacer()
                        TextField("", text: $doseAmount)
                            .keyboardType(.decimalPad)
                            .textFieldStyle(RoundedBorderTextFieldStyle())
                            .frame(width: 100)
                        Text("mg")
                    }
                }

                Section(header: Text("Mark as Current Dose")) {
                    Toggle("Current Dose", isOn: $isCurrentDose)
```

```
                }
            }
            .onAppear {
                if let editingDose = editingDose {
                    selectedAllergen = editingDose.allergen
                    doseName = editingDose.doseType
                    doseAmount = "\(editingDose.doseAmount)"
                    isCurrentDose = editingDose.isCurrentDose
                }
            }
            .navigationTitle(editingDose != nil ? "Edit Dose" : "Add Dose")
                .navigationBarItems(trailing:
                    Button("Save") {
                        if let dosage = Double(doseAmount), !doseAmount.isEmpty {
                            if let editingDose = editingDose {
                                // Update existing dose
                                doseViewModel.updateDose(editingDose, allergen: selectedAllergen,
     doseType: doseName, doseAmount: dosage, isCurrentDose: isCurrentDose)
                            } else {
                                // Add new dose
                                doseViewModel.addDose(allergen: selectedAllergen, doseType:
    doseName, doseAmount: dosage, isCurrentDose: isCurrentDose)
                            }
                        }
                        presentationMode.wrappedValue.dismiss()
                    }
                )
        }
    }
}
```

A.17 DoseRowView.swift

```
//
//  DoseRowView.swift
//  SeniorDesign
//
//  Created by Maddie on 3/7/24.
//

import Foundation
import SwiftUI

struct DoseRowView: View {
    let dose: Dose
    let numberFormatter: NumberFormatter
    let viewModel: DoseViewModel

    @State private var createNewDose = false
    @State private var editingDose: Dose?

    var body: some View {
        VStack(alignment: .leading) {
            HStack {
                Text(dose.doseType)
                    .font(.headline)
                if dose.isCurrentDose {
                    Image(systemName: "star.circle.fill")
                        .foregroundColor(.darkTeal)
                }
            }
            if let formattedDosage = numberFormatter.string(for: dose.doseAmount) {
                Text("Dosage: \(formattedDosage) mg")
                    .font(.subheadline)
                    .foregroundColor(.gray)
            }
        }
```

```
        .swipeActions(edge: .leading) {
           Button {
               viewModel.setAsCurrentDose(dose)
           } label: {
               Label("Set as Current", systemImage: "star")
           }
           .tint(.darkTeal)

            Button(action: {
                editingDose = dose
            }, label: {
                Label("Edit", systemImage: "pencil")
            })
            .tint(.yellow)
       }
        .navigationBarHidden(true)
        .background(
            NavigationLink(
                destination: AddDoseView(editingDose: editingDose)
                    .environmentObject(viewModel),
                isActive: Binding<Bool>(
                    get: { editingDose != nil },
                    set: { _ in editingDose = nil }
                ),
                label: { EmptyView() }
            )
            .hidden()
        )
    }
}
```

A.18 DosingView.swift

```
//
//  DosingView.swift
//  SeniorDesign
//
//  Created by Maddie on 10/18/23.
//

import SwiftUI
import TipKit

struct DosingView: View {
    // MARK: View Models
    @EnvironmentObject var doseViewModel: DoseViewModel

    // MARK: Variables
    @State private var createNewDose = false
    @State private var isEditing = false
    @Environment(\.colorScheme) var colorScheme

    @State private var showFilterOptions = false
    @State private var selectedFilter: FilterOption = .all

    var dosingTip = DosingViewTip()

    enum FilterOption: String, CaseIterable {
        case all = "All Doses"
        case currentDoses = "Show Only Current Doses"
    }

    var filterButton: some View {
        Menu {
            ForEach(FilterOption.allCases, id: \.self) { option in
                Button(action: {
                    selectedFilter = option
```

```
                }) {
                    Label(option.rawValue, systemImage: selectedFilter == option ? "checkmark" :
""")
                }
            }
        } label: {
            Image(systemName: "line.horizontal.3.decrease.circle")
        }
        .padding()
    }

    var filteredDoses: [AllergenWithDoses] {
        var filteredAllergens = doseViewModel.allergensWithDoses

        switch selectedFilter {
        case .all:
            break
        case .currentDoses:
            filteredAllergens = filteredAllergens.map { allergenWithDoses in
                AllergenWithDoses(
                    allergen: allergenWithDoses.allergen,
                    doses: allergenWithDoses.doses.filter(\.isCurrentDose)
                )
            }
        }

        return filteredAllergens
    }
    var numberFormatter: NumberFormatter = {
        let formatter = NumberFormatter()
        formatter.numberStyle = .decimal
        formatter.maximumFractionDigits = 1
        return formatter
    }()

    // MARK: UI Elements
    var createNewDoseButton: some View {
        Button(action: {
            createNewDose = true
        }, label: {
            Image(systemName: "plus")
                .foregroundColor(.darkTeal)
        })
        .sheet(isPresented: $createNewDose, content: {
            AddDoseView()
                .environmentObject(doseViewModel)
        })
        .padding()
    }

    var header: some View {
        HStack {
            Text("Doses")
                .font(.largeTitle.bold())
                .padding()
            Spacer()
            filterButton
            createNewDoseButton
        }
    }

    @State private var expandedStates: [String: Bool] = [:]

    var listOfDoses: some View {
        List {
            if filteredDoses.isEmpty {
                Text("Click the + button to add a dose.")
                    .foregroundColor(.gray)
```

```
                    .padding()
            }
            ForEach(filteredDoses) { allergenWithDoses in
                DisclosureGroup(
                    isExpanded: Binding(
                        get: { expandedStates[allergenWithDoses.allergen] ?? false },
                                set: { expandedStates[allergenWithDoses.allergen] = $0 }
                            ),
                    content: {
                        ForEach(allergenWithDoses.doses) { dose in
                            DoseRowView(dose: dose, numberFormatter: numberFormatter, viewModel:
    doseViewModel)
                        }
                        .onDelete { indices in
                            indices.forEach { index in
                                doseViewModel.deleteDose(allergenWithDoses.doses[index])
                            }
                        }
                    },
                    label: {
                        Text(allergenWithDoses.allergen)
                            .font(.title2)
                            .bold()
                            .foregroundColor(colorScheme == .dark ? Color.white : Color.black)
                    }
                )
            }
        }
        .listStyle(PlainListStyle())
        .background(colorScheme == .dark ? Color.black : Color.white)
        .onAppear {
            doseViewModel.loadDoses()
            for allergenWithDoses in doseViewModel.allergensWithDoses {
                expandedStates[allergenWithDoses.allergen] = true
            }
        }
        .onDisappear(perform: {
            doseViewModel.saveDoses()
        })
    }

    // MARK: Dosing View
    var body: some View {
        VStack(alignment: .leading) {
            header
            if #available(iOS 17.0, *) {
                TipView(dosingTip, arrowEdge: .none)
                    .tipCornerRadius(15)
                    .padding(.leading, 25)
                    .padding(.trailing, 25)
            }
            listOfDoses
        }
        .task {
            if #available(iOS 17.0, *) {
                try? Tips.configure([
                    .displayFrequency(.immediate),
                    .datastoreLocation(.applicationDefault)
                ])
            }
        }
    }
}

// MARK: Preview
#Preview {
    DosingView()
        .environmentObject(DoseViewModel())
```

```
}
```

A.19 DosingViewTip.swift

```
//
//  DosingViewTip.swift
//  SeniorDesign
//
//  Created by Maddie on 3/7/24.
//

import Foundation
import TipKit

struct DosingViewTip: Tip, Identifiable {
    var id = UUID()
    var title: Text {
        Text("Add your OIT dosages here!")
    }

    var message: Text? {
        Text("Begin by tapping the + button to effortlessly add your doses. Once doses are added, simply swipe to access options for editing, deleting, or designating them as your current dose.")
    }

    var image: Image? {
        Image(systemName: "lightbulb.max.fill")
    }
}
```

A.20 ArticleRichLink.swift

```
//
//  ArticleRichLink.swift
//  SeniorDesign
//
//  Created by Maddie on 10/20/23.
//

import SwiftUI

struct ArticleRichLink: View {
    // MARK: Variables
    @State private var isShowingArticle = false
    var articleTitle: String
    var articleDescription: String
    var articleDisclaimer: AttributedString
    var articleContent: AttributedString
    var image: String

    @Environment(\.colorScheme) var colorScheme

    // MARK: Article Rich Link View
    var body: some View {
        NavigationLink(destination: ArticleView(articleTitle: articleTitle, articleDescription: articleContent, articleDisclaimer: articleDisclaimer, image: image))
        {
            ZStack(alignment: .bottom) {
                VStack(alignment: .leading) {
                    Image(image)
                        .resizable()
                        .aspectRatio(contentMode: .fill)
                        .frame(width: UIScreen.main.bounds.width - 30, height: 180)
                        .padding(.top, 0)
                        .padding(.bottom, 10)
```

```
                        .clipped()
                    HStack {
                        Text(articleTitle)
                            .font(.title2.bold())
                            .foregroundColor(colorScheme == .light ? .black : .white)
                            .padding(.leading, 20)
                        Spacer()
                    }
                    Text(articleDescription)
                        .font(.body)
                        .foregroundColor(colorScheme == .light ? .black : .white)
                        .padding(.leading, 20)
                    Spacer()
                }
                .frame(width: UIScreen.main.bounds.width - 30, height: 260)
                .background(colorScheme == .light ? Color.mediumGrey : Color.darkGrey)
                .cornerRadius(20)
            }
        }
        .buttonStyle(PlainButtonStyle())
    }
}
```

A.21 ArticleView.swift

```
//
//  ArticleView.swift
//  SeniorDesign
//
//  Created by Maddie on 10/20/23.
//

import SwiftUI

struct ArticleView: View {
    // MARK: Variables
    var articleTitle: String
    var articleDescription: AttributedString
    var articleDisclaimer: AttributedString
    var image: String

    @Environment(\.colorScheme) var colorScheme

    // MARK: Article View
    var body: some View {
        ScrollView {
            VStack {
                HStack {
                    Text(articleTitle)
                        .font(.largeTitle.bold())
                        .padding()
                    Spacer()
                }
                Image(image)
                    .resizable()
                    .aspectRatio(contentMode: .fill)
                    .frame(width: UIScreen.main.bounds.width - 30, height: 160)
                    .background(Color.lightTeal)
                    .cornerRadius(20)
                HStack {
                    Spacer()
                    VStack {
                        Text(articleDisclaimer)
                            .padding(.vertical, 20)
                            .padding(.horizontal, 20)
                            .frame(width: UIScreen.main.bounds.width - 30)
                            .background(Color.lightYellow)
```

```
                            .foregroundColor(.black)
                            .cornerRadius(20)
                            .padding(.top, 20)
                        Text(articleDescription)
                            .padding(20)
                    }
                    Spacer()
                }
                Spacer()
            }
        }
    }
}
```

A.22 DoctorPhoneTip.swift

```
//
//  DoctorPhoneTip.swift
//  SeniorDesign
//
//  Created by Maddie on 3/15/24.
//

import Foundation
import TipKit

struct DoctorPhoneTip: Tip, Identifiable {
    var id = UUID()
    var title: Text {
        Text("Add Important Contact Info")
    }

    var message: Text? {
        Text("Add your doctor's phone number below, so you can easily reach out to them if needed!")
    }

    var image: Image? {
        Image(systemName: "stethoscope.circle.fill")
    }
}
```

A.23 EducationView.swift

```
//
//  EducationView.swift
//  SeniorDesign
//
//  Created by Maddie on 10/18/23.
//

import SwiftUI
import TipKit

struct ImageData: Codable {
    var imageData: Data
}

struct EducationView: View {
    // MARK: Variables
    let content = EducationViewContent()
    var resourcesTip = ResourcesTip()
    var esTip = EmergencyServicesTip()
    var infoTip = ImportantDocumentsTip()
    var doctorTip = DoctorPhoneTip()

```

```
    @State var doctorPhoneNumber: String = ""
    @State var isShowingImagePicker: Bool = false
    @State private var selectedImages: [UIImage] = []
    @State private var storedImageData: [ImageData] = []

    @Environment(\.colorScheme) var colorScheme

    // MARK: UI Elements
    var articles: some View {
        VStack(alignment: .leading) {
            ArticleRichLink(articleTitle: "Anaphylaxis", articleDescription: "Learn the signs & symptoms of anaphylaxis.", articleDisclaimer: content.anaphylaxisDisclaimer, articleContent: content.anaphylaxisTips, image: "anaphylaxis")
                .padding(7)
            ArticleRichLink(articleTitle: "OIT Best Practices", articleDescription: "Tips and tricks for Oral Immunotherapy.", articleDisclaimer: content.oitDisclaimer, articleContent: content.oitTips, image: "oit")
                .padding(7)
            ArticleRichLink(articleTitle: "Avoiding Cross Contamination", articleDescription: "Tips to help avoid cross contamination.", articleDisclaimer: content.ccDisclaimer, articleContent: content.crossContaminationTips, image: "crossContamination")
                .padding(7)
        }
    }

    var emergencyServicesInformation: some View {
        VStack(alignment: .center) {
            HStack {
                Text("Emergency Services")
                    .font(.title3).bold()
                    .padding()
                Spacer()
            }
            if #available(iOS 17.0, *) {
                TipView(esTip, arrowEdge: .bottom)
                    .tipCornerRadius(15)
                    .padding(.leading, 25)
                    .padding(.trailing, 25)
            }
            Button(action: {
                if let url = URL(string: "tel://911") {
                    UIApplication.shared.open(url, options: [:], completionHandler: nil)
                }
            }) {
                Image(systemName: "phone.fill")
                    .resizable()
                    .frame(width: 20, height: 20)
                    .foregroundColor(.black)
                    .padding(.leading, 20)
                Text("911    ")
                    .font(.body)
                    .bold()
                    .foregroundColor(.black)
            }
            .frame(width: UIScreen.main.bounds.width - 30, height: 50)
            .background(colorScheme == .light ? Color.lightTeal : Color.darkTeal)
            .cornerRadius(10)
        }
    }

    var fareInformation: some View {
        VStack(alignment: .center) {
            HStack {
                Text("Food Allergy Research & Education")
                    .font(.title3).bold()
                    .padding()
                Spacer()
            }
```

```
            HStack{
                Button(action: {
                    if let url = URL(string: "https://www.foodallergy.org") {
                        UIApplication.shared.open(url)
                    }
                }) {
                    Image(systemName: "safari.fill")
                        .resizable()
                        .frame(width: 20, height: 20)
                        .foregroundColor(.black)
                        .padding(.leading, 20)
                    Text("Website    ")
                        .font(.body)
                        .bold()
                        .foregroundColor(.black)
                }
                .frame(width: (UIScreen.main.bounds.width - 40) / 2, height: 50)
                .background(colorScheme == .light ? Color.lightTeal : Color.darkTeal)
                .cornerRadius(10)

                Button(action: {
                    if let url = URL(string: "tel://18009294040") {
                        UIApplication.shared.open(url, options: [:], completionHandler: nil)
                    }
                }) {
                    Image(systemName: "phone.fill")
                        .resizable()
                        .frame(width: 20, height: 20)
                        .foregroundColor(.black)
                        .padding(.leading, 20)
                    Text("Phone       ")
                        .font(.body)
                        .bold()
                        .foregroundColor(.black)
                }
                .frame(width: (UIScreen.main.bounds.width - 40) / 2, height: 50)
                .background(colorScheme == .light ? Color.lightTeal : Color.darkTeal)
                .cornerRadius(10)
            }
        }
    }

    var doctorClinicInformation: some View {
        VStack(alignment: .center) {
            HStack {
                Text("Doctor / Clinic Contact")
                    .font(.title3).bold()
                    .padding()
                Spacer()
            }
            if #available(iOS 17.0, *) {
                TipView(doctorTip, arrowEdge: .bottom)
                    .tipCornerRadius(15)
                    .padding(.leading, 25)
                    .padding(.trailing, 25)
            }
            TextField("Enter Doctor Phone Number", text: $doctorPhoneNumber)
                .textFieldStyle(RoundedBorderTextFieldStyle())
                .keyboardType(.numberPad)
                .frame(width: UIScreen.main.bounds.width - 30, height: 50)
            Button(action: {
                if let url = URL(string: "tel://\(doctorPhoneNumber)") {
                    UIApplication.shared.open(url, options: [:], completionHandler: nil)
                }
            }) {
                Image(systemName: "phone.fill")
                    .resizable()
                    .frame(width: 20, height: 20)
```

```
                    .foregroundColor(.black)
                    .padding(.leading, 20)
                Text("Doctor Phone Number")
                    .font(.body)
                    .bold()
                    .foregroundColor(.black)
            }
            .frame(width: UIScreen.main.bounds.width - 30, height: 50)
            .background(colorScheme == .light ? Color.lightTeal : Color.darkTeal)
            .cornerRadius(10)
        }
    }

    var relevantDocuments: some View {
        VStack(alignment: .center) {
            HStack {
                Text("Important Documents")
                    .font(.title3).bold()
                    .padding()
                Spacer()
            }
            if #available(iOS 17.0, *) {
                TipView(infoTip, arrowEdge: .bottom)
                    .tipCornerRadius(15)
                    .padding(.leading, 25)
                    .padding(.trailing, 25)
            }
            Button(action: {
                isShowingImagePicker = true
            }) {
                Text("Select Images")
                    .font(.body)
                    .bold()
                    .foregroundColor(.black)
            }
            .frame(width: UIScreen.main.bounds.width - 30, height: 50)
            .background(colorScheme == .light ? Color.lightTeal : Color.darkTeal)
            .cornerRadius(10)
            .padding()
            .sheet(isPresented: $isShowingImagePicker) {
                MultiImagePicker(selectedImages: $selectedImages)
            }

            ScrollView {
                LazyVGrid(columns: Array(repeating: GridItem(), count: 3), spacing: 10) {
                    ForEach(selectedImages.indices, id: \.self) { index in
                        VStack {
                            NavigationLink(destination: Image(uiImage: selectedImages[index])
                                .resizable()
                                .aspectRatio(contentMode: .fit)
                                .navigationBarTitle("", displayMode: .inline)
                            ) {
                                Image(uiImage: selectedImages[index])
                                    .resizable()
                                    .aspectRatio(contentMode: .fill)
                                    .frame(width: UIScreen.main.bounds.width / 3 - 15, height:
    UIScreen.main.bounds.width / 3 - 15)
                                    .cornerRadius(10)
                            }
                            .buttonStyle(PlainButtonStyle())

                            Button(action: {
                                selectedImages.remove(at: index)
                            }) {
                                Image(systemName: "xmark.circle.fill")
                                    .foregroundColor(.gray)
                                    .background(colorScheme == .light ? Color.lightGrey : Color.
    darkGrey)
```

```
                                    .clipShape(Circle())
                                    .padding(1)
                            }
                            .frame(width: UIScreen.main.bounds.width / 3 - 15, height: 30,
alignment: .center)
                        }
                    }
                }
                .padding()
            }
        }
    }

    // MARK: Education Tab View
    var body: some View {
        VStack(alignment: .leading) {
            Text("Resources")
                .font(.largeTitle.bold())
                .padding()
            ScrollView {
                if #available(iOS 17.0, *) {
                    TipView(resourcesTip, arrowEdge: .none)
                        .tipCornerRadius(15)
                        .padding(.leading, 25)
                        .padding(.trailing, 25)
                }
                articles
                emergencyServicesInformation
                doctorClinicInformation
                fareInformation
                relevantDocuments
            }
        }
        .padding(.bottom, 20)
        .task {
            if #available(iOS 17.0, *) {
                try? Tips.configure([
                    .displayFrequency(.immediate),
                    .datastoreLocation(.applicationDefault)
                ])
            }
        }
        .onAppear {
            // Load saved images from UserDefaults
            if let data = UserDefaults.standard.data(forKey: "selectedImages") {
                do {
                    storedImageData = try JSONDecoder().decode([ImageData].self, from: data)
                    selectedImages = []
                    // Convert stored data to UIImage
                    for imageData in storedImageData {
                        if let image = UIImage(data: imageData.imageData) {
                            selectedImages.append(image)
                        }
                    }
                } catch {
                    print("Error decoding images: \(error)")
                }
            }
        }
        .onDisappear {
            // Save selected images to UserDefaults
            do {
                storedImageData.removeAll()
                for image in selectedImages {
                    if let imageData = image.jpegData(compressionQuality: 0.5) {
                        storedImageData.append(ImageData(imageData: imageData))
                    }
                }
```

```
                let encodedData = try JSONEncoder().encode(storedImageData)
                UserDefaults.standard.set(encodedData, forKey: "selectedImages")
            } catch {
                print("Error encoding images: \(error)")
            }
        }
    }
}

// MARK: Preview
#Preview {
    EducationView()
}
```

A.24 EducationViewContent.swift

```
//
//  EducationViewContent.swift
//  SeniorDesign
//
//  Created by Maddie on 1/1/24.
//

import Foundation

struct EducationViewContent {
    let anaphylaxisDisclaimer: AttributedString = try! AttributedString(markdown: "       **Disclaimer:** This information is provided for educational purposes only and should not be construed as medical advice. Always seek the advice of a qualified healthcare professional for any medical concerns or questions regarding anaphylaxis or its treatment.")

    let oitDisclaimer: AttributedString = try! AttributedString(markdown: "       **Disclaimer:** This information is provided for educational purposes only and should not be construed as medical advice. Always seek the advice of a qualified healthcare professional for any medical concerns or questions regarding Oral Immunotherapy or its treatment.")

    let ccDisclaimer: AttributedString = try! AttributedString(markdown: "       **Disclaimer:** This information is provided for educational purposes only and should not be construed as medical advice. Always seek the advice of a qualified healthcare professional for any medical concerns or questions regarding cross contamination.")

    let anaphylaxisTips: AttributedString = try! AttributedString(markdown: """
        Anaphylaxis is a severe and potentially life-threatening allergic reaction that can occur suddenly after exposure to an allergen. It involves multiple body systems and can lead to a rapid and severe allergic response.

        Common signs and symptoms of anaphylaxis include:

            Skin reactions, such as hives, itching, and flushed or pale skin.
            Respiratory difficulties, including wheezing, shortness of breath, or difficulty breathing.
            Swelling, especially in the face, lips, throat, or tongue.
            Gastrointestinal symptoms, such as abdominal pain, nausea, vomiting, or diarrhea.
            Low blood pressure, which can lead to dizziness, fainting, or loss of consciousness.

        Immediate treatment for anaphylaxis involves the use of epinephrine (adrenaline) through an auto-injector and seeking emergency medical assistance.

        With OIT, because you're ingesting your allergen(s), the risk of anaphylaxis is higher than usual. It is important to carry an epinephrine autoinjector with you at all times.

        If you or someone experiences symptoms of anaphylaxis, call 911 immediately.

        **Source:** [Mayo Clinic](https://www.mayoclinic.org/diseases-conditions/anaphylaxis/symptoms-causes/syc-20351468)
        """, options: AttributedString.MarkdownParsingOptions(interpretedSyntax: .inlineOnlyPreservingWhitespace))
```

```
    let oitTips: AttributedString = try! AttributedString(markdown: """
        Oral Immunotherapy (OIT) involves ingesting small, controlled amounts of an allergen to build tolerance over time. It is a treatment option for some individuals with food allergies.

        Some tips and best practices for OIT include:

            Avoiding exercise within 2 hours of taking a dose.
            Avoiding alcohol consumption near dose times.
            Taking doses with food to reduce the likelihood of stomach discomfort.
            Being consistent with dose schedules to maintain progress.
            Carrying an epinephrine auto-injector at all times.
            Informing healthcare providers about ongoing OIT treatment before any medical procedures or surgeries.

        Remember, OIT should be performed under the guidance and supervision of trained healthcare professionals. Follow their instructions closely and report any concerns or adverse reactions immediately.

        **Source:** [American Academy of Allergy Asthma & Immunology](https://www.aaaai.org/conditions-and-treatments/library/allergy-library/oral-immunotherapy)
        """, options: AttributedString.MarkdownParsingOptions(interpretedSyntax: .inlineOnlyPreservingWhitespace))

    let crossContaminationTips: AttributedString = try! AttributedString(markdown: """
        Cross-contamination poses a significant threat to individuals with food allergies, even in the smallest traces. It occurs when allergens unintentionally transfer to safe foods or surfaces during food preparation or handling. The risk lies in these tiny traces, which, if ingested, can lead to severe allergic reactions.

        The kitchen is a primary area susceptible to cross-contamination. Shared utensils, cutting boards, and countertops can harbor allergens, posing risks to allergic individuals. Similarly, public spaces like restaurants, schools, or workplaces carry potential cross-contamination risks that demand vigilance.

        At home, preventing cross-contamination involves meticulous organization. Designate separate areas for allergen-free cooking and use dedicated utensils, cutting boards, and cookware. Implement rigorous cleaning protocols to ensure surfaces, appliances, and hands remain allergen-free during food preparation.

        Restaurants and food establishments present unique challenges. Effective communication with staff about food allergies is crucial. Asking questions about food preparation methods and understanding restaurant protocols for preventing cross-contamination can significantly mitigate risks.

        When dealing with packaged foods, thorough label reading is essential. Check food labels for allergen information and be aware of advisory labels like "may contain" to gauge cross-contamination risks.

        Navigating social settings and eating out requires assertiveness. Choose restaurants with allergy-aware menus and communicate dietary needs. Similarly, in social gatherings, communicate allergy concerns to hosts and inquire about ingredient information.

        Educating others about the significance of cross-contamination prevention is pivotal. Inform family and friends about the importance of allergen safety and advocate for understanding and accommodation in public spaces.

        Emergency preparedness is paramount. Always carry epinephrine auto-injectors and understand their usage. Recognizing allergic reactions promptly and seeking medical assistance when necessary is critical for managing allergic incidents.

        In conclusion, proactive strategies to prevent cross-contamination are vital for individuals with food allergies. By implementing these measures and raising awareness, we can create safer environments and minimize allergic risks significantly.

        **Source:** [Food Allergy Research & Education (FARE)](https://www.foodallergy.org/living-food-allergies/food-allergy-essentials/cross-contact-cross-contamination)
        """, options: AttributedString.MarkdownParsingOptions(interpretedSyntax: .
```

```
    inlineOnlyPreservingWhitespace))
}
```

A.25 EmergencyServicesTip.swift

```
//
//  EmergencyServicesTip.swift
//  SeniorDesign
//
//  Created by Maddie on 3/7/24.
//

import Foundation
import TipKit

struct EmergencyServicesTip: Tip, Identifiable {
    var id = UUID()
    var title: Text {
        Text("Emergency Assistance")
    }

    var message: Text? {
        Text("In case of an anaphylactic reaction, don't hesitate to call 911 immediately. Time is crucial. Stay calm, provide your location, and follow dispatcher instructions. Administer epinephrine if available, and if trained to do so.")
    }

    var image: Image? {
        Image(systemName: "phone.badge.checkmark")
    }
}
```

A.26 ImportantDocumentsTip.swift

```
//
//  ImportantDocumentsTip.swift
//  SeniorDesign
//
//  Created by Maddie on 3/7/24.
//

import Foundation
import TipKit

struct ImportantDocumentsTip: Tip, Identifiable {
    var id = UUID()
    var title: Text {
        Text("Keep Your Information in One Place!")
    }

    var message: Text? {
        Text("Add pictures of important documents, such as your Emergency Action Plan, below, so you can easily find them later!")
    }

    var image: Image? {
        Image(systemName: "doc.richtext")
    }
}
```

A.27 MultiImagePicker.swift

```
//
//  MultiImagePicker.swift
```

```swift
//  SeniorDesign
//
//  Created by Maddie on 3/15/24.
//

import Foundation
import SwiftUI
import PhotosUI

struct MultiImagePicker: UIViewControllerRepresentable {
    @Binding var selectedImages: [UIImage]

    func makeUIViewController(context: Context) -> PHPickerViewController {
        var configuration = PHPickerConfiguration()
        configuration.filter = .images
        configuration.selectionLimit = 0 // Set to 0 to allow multiple selections
        let picker = PHPickerViewController(configuration: configuration)
        picker.delegate = context.coordinator
        return picker
    }

    func updateUIViewController(_ uiViewController: PHPickerViewController, context: Context) {}

    func makeCoordinator() -> Coordinator {
        Coordinator(self)
    }

    class Coordinator: NSObject, PHPickerViewControllerDelegate {
        let parent: MultiImagePicker

        init(_ parent: MultiImagePicker) {
            self.parent = parent
        }

        func picker(_ picker: PHPickerViewController, didFinishPicking results: [PHPickerResult]) {
            parent.selectedImages.removeAll()

            for result in results {
                if result.itemProvider.canLoadObject(ofClass: UIImage.self) {
                    result.itemProvider.loadObject(ofClass: UIImage.self) { (image, error) in
                        if let image = image as? UIImage {
                            DispatchQueue.main.async { [self] in
                                parent.selectedImages.append(image)
                            }
                        }
                    }
                }
            }

            DispatchQueue.main.async {
                picker.dismiss(animated: true)
            }
        }
    }
}
```

A.28 ResourcesTip.swift

```swift
//
//  ResourcesTip.swift
//  SeniorDesign
//
//  Created by Maddie on 3/7/24.
//

import Foundation
```

```swift
import TipKit

struct ResourcesTip: Tip, Identifiable {
    var id = UUID()
    var title: Text {
        Text("Stay Informed")
    }

    var message: Text? {
        Text("Explore our curated articles, hotlines, and useful links for valuable insights into food allergies and OIT. Empower yourself with knowledge to navigate your journey confidently!")
    }

    var image: Image? {
        Image(systemName: "books.vertical.circle")
    }
}
```

A.29 HealthKitManager.swift

```swift
//
//  HealthKitManager.swift
//  SeniorDesign
//
//  Created by Maddie on 10/19/23.
//

import Foundation
import HealthKit
import CoreData

class HealthKitManager {
    let healthStore = HKHealthStore()

    // MARK: Symptoms Array
    public let symptoms: [HKSampleType] = [
        HKObjectType.categoryType(forIdentifier: .abdominalCramps)!,
        HKObjectType.categoryType(forIdentifier: .bloating)!,
        HKObjectType.categoryType(forIdentifier: .constipation)!,
        HKObjectType.categoryType(forIdentifier: .diarrhea)!,
        HKObjectType.categoryType(forIdentifier: .heartburn)!,
        HKObjectType.categoryType(forIdentifier: .nausea)!,
        HKObjectType.categoryType(forIdentifier: .vomiting)!,
        HKObjectType.categoryType(forIdentifier: .appetiteChanges)!,
        HKObjectType.categoryType(forIdentifier: .chills)!,
        HKObjectType.categoryType(forIdentifier: .dizziness)!,
        HKObjectType.categoryType(forIdentifier: .fainting)!,
        HKObjectType.categoryType(forIdentifier: .fatigue)!,
        HKObjectType.categoryType(forIdentifier: .fever)!,
        HKObjectType.categoryType(forIdentifier: .generalizedBodyAche)!,
        HKObjectType.categoryType(forIdentifier: .hotFlashes)!,
        HKObjectType.categoryType(forIdentifier: .chestTightnessOrPain)!,
        HKObjectType.categoryType(forIdentifier: .coughing)!,
        HKObjectType.categoryType(forIdentifier: .rapidPoundingOrFlutteringHeartbeat)!,
        HKObjectType.categoryType(forIdentifier: .shortnessOfBreath)!,
        HKObjectType.categoryType(forIdentifier: .skippedHeartbeat)!,
        HKObjectType.categoryType(forIdentifier: .wheezing)!,
        HKObjectType.categoryType(forIdentifier: .lowerBackPain)!,
        HKObjectType.categoryType(forIdentifier: .headache)!,
        HKObjectType.categoryType(forIdentifier: .memoryLapse)!,
        HKObjectType.categoryType(forIdentifier: .moodChanges)!,
        HKObjectType.categoryType(forIdentifier: .lossOfSmell)!,
        HKObjectType.categoryType(forIdentifier: .lossOfTaste)!,
        HKObjectType.categoryType(forIdentifier: .runnyNose)!,
        HKObjectType.categoryType(forIdentifier: .soreThroat)!,
        HKObjectType.categoryType(forIdentifier: .sinusCongestion)!,
```

```
        HKObjectType.categoryType(forIdentifier: .acne)!,
        HKObjectType.categoryType(forIdentifier: .drySkin)!,
        HKObjectType.categoryType(forIdentifier: .hairLoss)!,
        HKObjectType.categoryType(forIdentifier: .nightSweats)!,
        HKObjectType.categoryType(forIdentifier: .sleepChanges)!,
        HKObjectType.categoryType(forIdentifier: .bladderIncontinence)!,
    ]

    let symptomEmojis: [String: String] = [
        "HKCategoryTypeIdentifierAbdominalCramps": "    ",
        "HKCategoryTypeIdentifierBloating": "    ",
        "HKCategoryTypeIdentifierConstipation": "    ",
        "HKCategoryTypeIdentifierDiarrhea": "    ",
        "HKCategoryTypeIdentifierHeartburn": "    ",
        "HKCategoryTypeIdentifierNausea": "    ",
        "HKCategoryTypeIdentifierVomiting": "    ",
        "HKCategoryTypeIdentifierAppetiteChanges": "       ",
        "HKCategoryTypeIdentifierChills": "    ",
        "HKCategoryTypeIdentifierDizziness": "    ",
        "HKCategoryTypeIdentifierFainting": "           ",
        "HKCategoryTypeIdentifierFatigue": "    ",
        "HKCategoryTypeIdentifierFever": "    ",
        "HKCategoryTypeIdentifierGeneralizedBodyAche": "    ",
        "HKCategoryTypeIdentifierHotFlashes": "    ",
        "HKCategoryTypeIdentifierChestTightnessOrPain": "    ",
        "HKCategoryTypeIdentifierCoughing": "    ",
        "HKCategoryTypeIdentifierRapidPoundingOrFlutteringHeartbeat": "    ",
        "HKCategoryTypeIdentifierShortnessOfBreath": "    ",
        "HKCategoryTypeIdentifierSkippedHeartbeat": "    ",
        "HKCategoryTypeIdentifierWheezing": "    ",
        "HKCategoryTypeIdentifierLowerBackPain": "    ",
        "HKCategoryTypeIdentifierHeadache": "    ",
        "HKCategoryTypeIdentifierMemoryLapse": "    ",
        "HKCategoryTypeIdentifierMoodChanges": "    ",
        "HKCategoryTypeIdentifierLossOfSmell": "    ",
        "HKCategoryTypeIdentifierLossOfTaste": "    ",
        "HKCategoryTypeIdentifierRunnyNose": "    ",
        "HKCategoryTypeIdentifierSoreThroat": "    ",
        "HKCategoryTypeIdentifierSinusCongestion": "    ",
        "HKCategoryTypeIdentifierAcne": "    ",
        "HKCategoryTypeIdentifierDrySkin": "    ",
        "HKCategoryTypeIdentifierHairLoss": "    ",
        "HKCategoryTypeIdentifierNightSweats": "    ",
        "HKCategoryTypeIdentifierSleepChanges": "    ",
        "HKCategoryTypeIdentifierBladderIncontinence": "    "
    ]

    func symptomImageNeeded(_ symptom: String) -> Bool {
        if symptom == "HKCategoryTypeIdentifierConstipation"
            || symptom == "HKCategoryTypeIdentifierAbdominalCramps"
            || symptom == "HKCategoryTypeIdentifierBladderIncontinence"
            || symptom == "HKCategoryTypeIdentifierCoughing"
            || symptom == "HKCategoryTypeIdentifierSoreThroat"
            || symptom == "HKCategoryTypeIdentifierSkippedHeartbeat"
            || symptom == "HKCategoryTypeIdentifierAcne"
            || symptom == "HKCategoryTypeIdentifierLowerBackPain"
            || symptom == "HKCategoryTypeIdentifierChills"
            || symptom == "HKCategoryTypeIdentifierFainting"
            || symptom == "HKCategoryTypeIdentifierBloating"
            || symptom == "HKCategoryTypeIdentifierAppetiteChanges" {
            return true
        }
        return false
    }

    // MARK: Symptom Functions
    func saveSymptom(_ symptom: String, for date: Date) {
        guard let symptomType = HKObjectType.categoryType(forIdentifier: HKCategoryTypeIdentifier
```

```
    (rawValue: symptom)) else {
            return
        }

        let sample = HKCategorySample(type: symptomType, value: HKCategoryValue.notApplicable.
    rawValue, start: date, end: date)
        healthStore.save(sample) { success, error in
            if let error = error {
                print("Error saving symptom data: \(error.localizedDescription)")
            } else {
                print("Symptom data saved successfully")
            }
        }
    }

    func fetchSymptomsForDate(_ date: Date, completion: @escaping ([String]) -> Void) {
        var symptomsData: [String] = []
        for symptom in symptoms {
            let predicate = HKQuery.predicateForSamples(withStart: date, end: date, options: .
    strictStartDate)
            let query = HKSampleQuery(sampleType: symptom, predicate: predicate, limit:
    HKObjectQueryNoLimit, sortDescriptors: nil) { (query, samples, error) in
                if let error = error {
                    print("Error fetching symptom data: \(error.localizedDescription)")
                    completion([])
                    return
                }

                if let samples = samples as? [HKCategorySample] {
                    for sample in samples {
                        symptomsData.append(sample.categoryType.identifier)
                    }
                }
            }
            healthStore.execute(query)
        }
        completion(symptomsData)
    }


    @MainActor func loadSymptomsForSelectedDate(selectedDate: Date, symptomDataManager:
    SymptomDataManager, completion: @escaping () -> Void) {
        fetchSymptomsForDate(selectedDate) { symptoms in
            symptomDataManager.symptomRecords[selectedDate] = symptoms
            completion()
        }
    }

    // MARK: Health Request Functions
    func setUpHealthRequest(healthStore: HKHealthStore, readSteps: @escaping () -> Void) {
        if HKHealthStore.isHealthDataAvailable() {
            healthStore.requestAuthorization(toShare: Set(symptoms), read: [HKObjectType.
    quantityType(forIdentifier: HKQuantityTypeIdentifier.heartRate)!]) { success, error in
                if success {
                } else if error != nil {
                    print(error ?? "Error")
                }
            }
        }
    }

    @MainActor func saveSymptomsToCoreData(for date: Date, symptoms: [String]) {
        let symptomDataManager = SymptomDataManager()
        symptomDataManager.saveSymptoms(for: date, symptoms: symptoms)
    }

    @MainActor func fetchSymptomsFromCoreData(for date: Date) -> [String] {
        let symptomDataManager = SymptomDataManager()
```

```
        return symptomDataManager.fetchSymptoms(for: date)
    }
}
```

A.30 HealthKitViewModel.swift

```
//
//  HealthKitViewModel.swift
//  SeniorDesign
//
//  Created by Maddie on 10/19/23.
//

import Foundation
import HealthKit
import CoreData

class HealthKitViewModel: ObservableObject {
    // MARK: Variables
    @Published var isAuthorized = false
    @Published var doseRecords: [Date: DoseRecord] = [:]
    @Published var totalTakenDoses: Int = 0
    @Published var totalSkippedDoses: Int = 0
    private var healthStore = HKHealthStore()
    private var healthKitManager = HealthKitManager()

    // MARK: Initializer
    init() {
        changeAuthorizationStatus()
    }

    // MARK: Dose Functions
    // CoreData container
    public let persistentContainer: NSPersistentContainer = {
        let container = NSPersistentContainer(name: "DoseDataModel")
        let antihistamineDoseTransformer = AntihistamineDoseTransformer()
        ValueTransformer.setValueTransformer(antihistamineDoseTransformer, forName:
    NSValueTransformerName(rawValue: "AntihistamineDoseTransformer"))

        let dosesTransformer = DosesTransformer()
        ValueTransformer.setValueTransformer(dosesTransformer, forName: NSValueTransformerName(
    rawValue: "DosesTransformer"))

        let selectedAllergensTransformer = SelectedAllergensTransformer()
        ValueTransformer.setValueTransformer(selectedAllergensTransformer, forName:
    NSValueTransformerName(rawValue: "SelectedAllergensTransformer"))
        container.loadPersistentStores(completionHandler: { (_, error) in
            if let error = error {
                fatalError("Failed to load persistent store: \(error)")
            }
        })
        return container
    }()

    // Function to save dose record using CoreData
    func saveDoseRecord(_ doseRecord: DoseRecord) {
        let context = persistentContainer.viewContext
        guard let entity = NSEntityDescription.entity(forEntityName: "DoseRecordEntity", in:
    context) else { return }

        let doseRecordEntity = NSManagedObject(entity: entity, insertInto: context)
        doseRecordEntity.setValue(doseRecord.date, forKey: "date")

        do {
            try context.save()
        } catch {
            print("Failed to save dose record: \(error.localizedDescription)")
```

```
        }
    }

    // Function to fetch dose records using CoreData
    func fetchDoseRecords() {
        let context = persistentContainer.viewContext
        let fetchRequest = NSFetchRequest<DoseRecord>(entityName: "DoseRecord")

        do {
            let fetchedRecords = try context.fetch(fetchRequest)
            var doseRecordsByDate: [Date: DoseRecord] = [:]
            for record in fetchedRecords {
                doseRecordsByDate[record.date!] = record
            }
            self.doseRecords = doseRecordsByDate
        } catch {
            print("Failed to fetch dose records: \(error.localizedDescription)")
        }
    }


    //MARK: - HealthKit Authorization Request Methods
    func healthRequest() {
        healthKitManager.setUpHealthRequest(healthStore: healthStore) {
            self.changeAuthorizationStatus()
        }
    }

    func changeAuthorizationStatus() {
        guard let stepQtyType = HKObjectType.quantityType(forIdentifier: .stepCount) else {
    return }
        let status = self.healthStore.authorizationStatus(for: stepQtyType)

        switch status {
        case .notDetermined:
            isAuthorized = false
        case .sharingDenied:
            isAuthorized = false
        case .sharingAuthorized:
            DispatchQueue.main.async {
                self.isAuthorized = true
            }
        @unknown default:
            isAuthorized = false
        }
    }

    @MainActor func saveSymptomsToCoreData(for date: Date, symptoms: [String]) {
            let symptomDataManager = SymptomDataManager()
            symptomDataManager.saveSymptoms(for: date, symptoms: symptoms)
        }

    @MainActor func fetchSymptomsFromCoreData(for date: Date) -> [String] {
            let symptomDataManager = SymptomDataManager()
            return symptomDataManager.fetchSymptoms(for: date)
        }
}
```

A.31 AboutYourDoseView.swift

```
//
//  AboutYourDoseView.swift
//  SeniorDesign
//
//  Created by Maddie on 10/20/23.
//

```

```swift
import SwiftUI

struct AboutYourDoseView: View {
    // MARK: Variables
    let allergenDoses: DoseRecord

    @Environment(\.colorScheme) var colorScheme

    // MARK: UI Elements
    private var header: some View {
        HStack {
            Text("About Your Dose")
                .font(.headline)
                .foregroundColor(colorScheme == .light ? Color.black : Color.black)
                .padding(.top, 20)
                .padding(.leading, 20)
                .padding(.bottom, 5)
            Spacer()
        }
    }

    func sortingFunction(_ allergen: String) -> Int {
        if allergen.hasPrefix("Zyrtec") || allergen.hasPrefix("Pepcid") || allergen.hasPrefix("Benadryl") {
            return 1
        } else {
            return 0
        }
    }

    private var dosesTaken: some View {
        VStack(alignment: .leading) {
            if let data = allergenDoses.doses as? Data,
               let doses = try? NSKeyedUnarchiver.unarchiveTopLevelObjectWithData(data) as? [String: String] {
                let sortedAllergens = doses.keys.sorted {
                    if sortingFunction($0) == sortingFunction($1) {
                        return $0 < $1 // Sort alphabetically within each group
                    } else {
                        return sortingFunction($0) < sortingFunction($1)
                    }
                }
                ForEach(sortedAllergens, id: \.self) { allergen in
                    if let dose = doses[allergen] {
                        let formattedDose = dose.components(separatedBy: "   ").last?
                            .replacingOccurrences(of: ".0", with: "") ?? ""
                            .trimmingCharacters(in: .whitespacesAndNewlines)
                        HStack {
                            Text("\(allergen)     \(formattedDose)")
                                .foregroundColor(colorScheme == .light ? Color.black : Color.black)
                                .padding(.leading, 20)
                                .padding(.bottom, 5)
                            Spacer()
                        }
                    }
                }
            }
        }
    }

    private var timeOfDose: some View {
        HStack {
            Text(timeFormatter.string(from: allergenDoses.time!))
                .foregroundColor(colorScheme == .light ? Color.black : Color.black)
                .padding(.leading, 20)
                .padding(.bottom, 10)
            Spacer()
```

```
        }
    }

    // MARK: About Your Dose View
    var body: some View {
        ZStack {
            VStack {
                header
                HStack {
                    dosesTaken
                        .frame(width: 230)
                    Spacer()
                }
                Spacer()
                timeOfDose
            }
            .frame(minWidth: 0, maxWidth: .infinity, minHeight: 0, maxHeight: .infinity,
    alignment: .topLeading)
            .background(colorScheme == .light ? Color.lightTeal : Color.darkTeal)
            .cornerRadius(20)
            .padding(.leading, 20)
            .padding(.trailing, 20)
            HStack {
                Spacer()
                Image(systemName: "pills.fill")
                    .font(.system(size: 90))
                    .foregroundColor(colorScheme == .light ? Color.darkTeal : Color.darkerTeal)
                    .padding(.trailing, 35)
            }
        }
    }

    // MARK: Functions
    private let timeFormatter: DateFormatter = {
        let formatter = DateFormatter()
        formatter.dateStyle = .none
        formatter.timeStyle = .short
        return formatter
    }()
}
```

A.32 CalendarDayView.swift

```
//
//  CalendarDayView.swift
//  SeniorDesign
//
//  Created by Maddie on 10/20/23.
//

import SwiftUI

struct CalendarDayView: View {
    // MARK: Variables
    @Binding var selectedDate: Date
    @Binding var selectedColorIndex: Int?
    @Environment(\.colorScheme) var colorScheme
    var date: Date
    var index: Int

    private var isSelected: Bool {
        selectedDate == date
    }

    private var isSelectable: Bool {
        return date <= Date()
    }
```

```
25
26      private var dayText: String {
27          let dateFormatter = DateFormatter()
28          dateFormatter.dateFormat = "E"
29          return String(dateFormatter.string(from: date).prefix(1))
30      }
31
32      private var dateText: String {
33          let dateFormatter = DateFormatter()
34          dateFormatter.dateFormat = "d"
35          return dateFormatter.string(from: date)
36      }
37
38      private var circleColor: Color {
39          if !isSelectable {
40              return colorScheme == .light ? Color.lightGrey : Color.darkGrey
41          }
42          if isSelected {
43              return colorScheme == .light ? .darkTeal : .darkerTeal // Color for selected circle
44          } else if let selectedColorIndex = selectedColorIndex, selectedColorIndex == index {
45              return colorScheme == .light ? .lightTeal : .darkTeal // Color for previously tapped
    circle
46          } else {
47              return colorScheme == .light ? .lightTeal : .darkTeal // Default color for other
    circles
48          }
49      }
50
51      private var circleTextColor: Color {
52          if !isSelectable {
53              return colorScheme == .light ? Color.mediumGrey : Color.black
54          }
55          if isSelected {
56              return colorScheme == .light ? Color.white : Color.black
57          } else if let selectedColorIndex = selectedColorIndex, selectedColorIndex == index {
58              return Color.black
59          } else {
60              return Color.black        }
61      }
62
63      // MARK: UI Elements
64      private var dayOfWeek: some View {
65          Text(dayText)
66              .font(.caption)
67              .bold()
68      }
69
70      private var dayCircle: some View {
71          Circle()
72              .frame(width: min(UIScreen.main.bounds.width / 10, 70), height: min(UIScreen.main.
    bounds.width / 7, 70))
73              .overlay(
74                  Text(dateText)
75                      .font(.subheadline)
76                      .foregroundColor(circleTextColor)
77                      .bold()
78              )
79              .onTapGesture {
80                  if isSelectable {
81                      selectedDate = date
82                      selectedColorIndex = index
83                  }
84              }
85              .foregroundColor(circleColor)
86              .cornerRadius(8)
87              .disabled(!isSelectable)
88      }
89
```

```
90
91      // MARK: Calendar Day View
92      var body: some View {
93          VStack {
94              dayOfWeek
95              dayCircle
96          }
97      }
98  }
```

A.33 DosingPopUp.swift

```
1   //
2   //  DosingPopUp.swift
3   //  SeniorDesign
4   //
5   //  Created by Maddie on 10/18/23.
6   //
7
8   import SwiftUI
9   import CoreData
10
11  struct DosingPopUp: View {
12      // MARK: View Models
13      @EnvironmentObject var profileViewModel: ProfileViewModel
14      @EnvironmentObject var healthKitViewModel: HealthKitViewModel
15      @EnvironmentObject var doseViewModel: DoseViewModel
16
17      // MARK: Variables
18      @Environment(\.presentationMode) var presentationMode
19
20      @State private var selectedDoses: [String: String] = [:]
21      @State private var selectedAntihistamineDoses: [AntihistamineDose] = []
22      @State private var doseTime = Date()
23      @State private var notes = ""
24      @State private var antihistamines: [String] = ["Benadryl", "Pepcid", "Zyrtec", "Other"]
25      @State private var hidden: [Bool] = [true, true, true, true]
26      @State private var dose: String = ""
27
28      private var selectedAllergensArray: [String] {
29          Array(profileViewModel.selectedAllergens)
30      }
31
32      @Environment(\.colorScheme) var colorScheme
33
34      let selectedDate: Date
35
36      // MARK: Initializer
37      init(selectedDate: Date) {
38          self.selectedDate = selectedDate
39      }
40
41      // MARK: UI Elements
42      var allergensSectionHeader: some View {
43          VStack {
44              if (profileViewModel.profileData.allergens.count == 0) {
45                  HStack {
46                      Text("Allergens")
47                          .font(.headline)
48                          .padding()
49                          .padding(.bottom, 0)
50                      Spacer()
51                  }
52                  HStack {
53                      Text("       You haven't recorded any allergens in your profile. To add
    allergens, go to your profile settings in the Home tab and update the allergens section.")
54                          .font(.body)
```

```
                        .padding()
                }
                .frame(width: UIScreen.main.bounds.width-20, height: 130)
                .background(Color.grey)
                .opacity(0.7)
                .padding(.top, 0)
                .cornerRadius(15)
            } else {
                HStack {
                    Text("Allergens")
                        .font(.headline)
                        .padding()
                    Spacer()
                }
            }
        }
    }

    var allergensSection: some View {
        LazyVGrid(columns: [GridItem(.flexible()), GridItem(.flexible()), GridItem(.flexible())], spacing: 20) {
            ForEach(profileViewModel.profileData.allergens.sorted(), id: \.self) { allergen in
                Button(action: {
                    profileViewModel.toggleAllergenSelection(allergen)
                }) {
                    VStack {
                        if profileViewModel.profileData.isNut(allergen) {
                            Image("\(allergen)")
                                .resizable()
                                .frame(width: 30, height: 30)
                                .aspectRatio(contentMode: .fit)
                                .padding(.bottom, 10)
                        } else {
                            Text(profileViewModel.profileData.allergenEmojiMap[allergen] ?? "")
                                .padding(.bottom, 10)
                                .font(.title)
                        }
                        Text(allergen)
                            .bold()
                            .foregroundColor(profileViewModel.isSelected(allergen) ? Color.white : Color.black)
                    }
                    .frame(width: 80, height: 100)
                    .padding()
                    .background(profileViewModel.isSelected(allergen) ? colorScheme == .light ? Color.darkTeal : Color.darkerTeal : colorScheme == .light ? Color.lightTeal : Color.darkTeal)
                    .cornerRadius(20)
                }
            }
        }
        .padding(.leading, 20)
        .padding(.trailing, 20)
    }

    var timeSectionHeader: some View {
        HStack {
            Text("Time of Dose")
                .font(.headline)
                .padding()
            Spacer()
        }
    }

    var timePicker: some View {
        DatePicker("Dose Time", selection: $doseTime, displayedComponents: .hourAndMinute)
            .datePickerStyle(CompactDatePickerStyle())
            .font(.body.bold())
            .labelsHidden()
```

```
            .padding(.leading, 20)
            .tint(.darkTeal)
    }

    var predoseHeader: some View {
        HStack {
            Text("Did you pre-dose any antihistamines before your oral immunotherapy dose?")
                .font(.headline)
                .padding()
            Spacer()
        }
    }

    var predoseSection: some View {
        VStack {
            ForEach(antihistamines.indices, id: \.self) { index in
                HStack {
                    Text(antihistamines[index])
                        .onTapGesture {
                            hidden[index].toggle()
                        }
                        .padding()
                    Spacer()
                    if !hidden[index] {
                        let selectedDose = Binding<String>(
                            get: {
                                selectedDoses[antihistamines[index]] ?? ""
                            },
                            set: { newValue in
                                selectedDoses[antihistamines[index]] = newValue
                            }
                        )
                        Picker("\(antihistamines[index])", selection: selectedDose) {
                            ForEach(["2.5", "5", "10", "12.5", "20", "25", "30", "40", "50"], id:
 \.self) { dosage in
                                Text("\(dosage) mg").tag("\(dosage) mg")
                            }
                        }
                        .pickerStyle(DefaultPickerStyle())
                        .tint(.darkTeal)
                    }
                }
                .padding(.horizontal, 20)
            }
            Divider()
                .padding(.horizontal, 20)
        }
    }



    var dosageSectionHeader: some View {
        HStack {
            if selectedAllergensArray.count > 0 {
                Text("Dosage for Allergens")
                    .font(.headline)
                    .padding()
                Spacer()
            }
        }
    }

    var dosageSection: some View {
        ForEach(selectedAllergensArray.sorted(), id: \.self) { allergen in
            HStack {
                Text(allergen)
                    .padding(.leading, 20)
                Spacer()
```

```
                Picker("", selection: Binding(
                    get: {
                        selectedDoses[allergen] ?? ""
                    },
                    set: { newValue in
                        selectedDoses[allergen] = newValue
                    }
                )) {
                    let doseTypesForSelectedAllergens = doseViewModel.
    doseTypesForSelectedAllergens(selectedAllergens: [allergen]).sorted()

                    ForEach(doseTypesForSelectedAllergens, id: \.self) { doseType in
                        let formattedDose = doseType.components(separatedBy: "    ").last?
                            .replacingOccurrences(of: ".0", with: "") ?? ""
                            .trimmingCharacters(in: .whitespacesAndNewlines)
                        Text(formattedDose)
                    }
                }
                .pickerStyle(MenuPickerStyle())
            }
            .padding(.leading, 20)
            .padding(.trailing, 20)
            .padding(.bottom, 10)
        }
    }



    var notesHeader: some View {
        HStack {
            Text("Notes")
                .font(.headline)
                .padding()
            Spacer()
        }
    }

    var notesSection: some View {
        TextField("Enter any notes about your dose here.", text: $notes)
            .textFieldStyle(DefaultTextFieldStyle())
            .padding(.leading, 20)
            .padding(.trailing, 20)
    }

    // MARK: Dosing Pop Up View
    var body: some View {
        NavigationStack {
            ScrollView {
                VStack(alignment: .leading) {
                    timeSectionHeader
                    timePicker

                    allergensSectionHeader
                    allergensSection

                    dosageSectionHeader
                    dosageSection

                    predoseHeader
                    predoseSection

                    notesHeader
                    notesSection
                    Spacer()
                }
                .navigationBarTitle("Dose Information")
                .navigationBarItems(trailing: Button("Done") {
                    saveDoseInformation()
```

```
                    presentationMode.wrappedValue.dismiss()
                })
            }
        }
    }

    // MARK: Functions
    func saveDoseInformation() {
        let context = healthKitViewModel.persistentContainer.viewContext
        guard let entity = NSEntityDescription.entity(forEntityName: "DoseRecord", in: context)
    else { return }
        let newDoseRecord = DoseRecord(entity: entity, insertInto: context)
        newDoseRecord.date = selectedDate
        newDoseRecord.time = doseTime
        newDoseRecord.notes = notes

        // Convert arrays to NSData or archive them before assigning to Core Data entity
        if let allergenData = try? NSKeyedArchiver.archivedData(withRootObject:
    selectedAllergensArray, requiringSecureCoding: false) as NSData? {
            newDoseRecord.selectedAllergens = allergenData
        }

        if let antihistamineData = try? NSKeyedArchiver.archivedData(withRootObject:
    selectedAntihistamineDoses, requiringSecureCoding: false) as NSData? {
            newDoseRecord.antihistamineDoses = antihistamineData
        }

        if let doseData = try? NSKeyedArchiver.archivedData(withRootObject: selectedDoses,
    requiringSecureCoding: false) as NSData? {
            newDoseRecord.doses = doseData
        }

        do {
            try context.save()
            _ = healthKitViewModel.fetchSymptomsFromCoreData(for: selectedDate)
            healthKitViewModel.fetchDoseRecords()
            doseViewModel.loadDoses()
            print("DoseRecord saved successfully.")
        } catch {
            print("Failed to save DoseRecord: \(error.localizedDescription)")
        }

        // Dismiss the pop-up
        presentationMode.wrappedValue.dismiss()
    }


}

// MARK: Preview
#Preview {
    DosingPopUp(selectedDate: Date())
        .environmentObject(ProfileViewModel())
        .environmentObject(HealthKitViewModel())
}
```

A.34 HomeView.swift

```
//
//  HomeView.swift
//  SeniorDesign
//
//  Created by Maddie on 8/20/23.
//

import SwiftUI

```

```
struct HomeView: View {
    // MARK: View Models
    @EnvironmentObject var profileViewModel: ProfileViewModel
    @EnvironmentObject var healthKitViewModel: HealthKitViewModel
    @EnvironmentObject var doseViewModel: DoseViewModel
    @EnvironmentObject var profileImageViewModel: ProfileModel

    let healthKitManager = HealthKitManager()

    // MARK: Variables
    @State private var selectedDate = Date()
    @State private var selectedColorIndex: Int?
    @State private var weekView = true
    @State private var logSymptoms = false
    @State private var logDose = false
    @State private var selectedSymptoms: Set<String> = []
    @State private var dayOfTreatment = 1 // Initial value
    @ObservedObject var symptomDataManager = SymptomDataManager()
    var symptomsForSelectedDate: [String] {
        return symptomDataManager.fetchSymptoms(for: selectedDate)
    }

    @Environment(\.colorScheme) var colorScheme

    // MARK: UI Elements
    var profileButton: some View {
        NavigationLink(destination: SettingsView().environmentObject(profileViewModel)
            .environmentObject(healthKitViewModel).environmentObject(profileImageViewModel)) {

                CircularProfileImage(imageState: profileImageViewModel.imageState)
                    .frame(width: 35, height: 35)
                    .foregroundColor(.secondary)
            }
            .padding(.leading, 16)
    }

    var monthCalendarButton: some View {
        HStack {
            Text(monthHeader)
                .font(.body)
                .bold()
            Image(systemName: "calendar")
                .resizable()
                .frame(width: 20, height: 20)
                .padding(.trailing, 20)
        }
        .onTapGesture {
            weekView.toggle()
        }
    }

    var header: some View {
        HStack {
            profileButton
            Spacer()
            monthCalendarButton
        }
    }

    var weekScrollView: some View {
        WeeklyScrollCalendarView
            .padding(.bottom, -10)
            .onAppear {
                selectedDate = todaysDate
            }
            .onChange(of: selectedDate) { _ in
                dayOfTreatment = calculateDayOfTreatment()
            }
```

```
    }

    var weekCircleView: some View {
        Circle()
            .padding(.leading, 20)
            .padding(.trailing, 20)
            .padding(.bottom, 10)
            .foregroundColor(colorScheme == .light ? .lightTeal : .darkTeal)
            .overlay(
                weekCircleViewContent
            )
    }

    var weekCircleViewContent: some View {
        VStack {
            Text("Day")
                .font(.body)
                .foregroundColor(colorScheme == .light ? Color.black : Color.black)
                .bold()
                .padding(.top, 30)
            Text("\(dayOfTreatment)")
                .font(.system(size: 70))
                .foregroundColor(colorScheme == .light ? Color.black : Color.black)
                .bold()
            Text("of Treatment")
                .font(.body)
                .foregroundColor(colorScheme == .light ? Color.black : Color.black)
                .bold()
            Button(action: {
                logDose.toggle()
            }) {
                Text("Log Dose          ")
                    .font(.subheadline).bold()
                    .foregroundColor(colorScheme == .light ? Color.white : Color.white)
                    .padding()
                    .background(colorScheme == .light ? Color.darkTeal : Color.darkerTeal)
                    .cornerRadius(30)
            }
            .fullScreenCover(isPresented: $logDose) {
                DosingPopUp(selectedDate: selectedDate)
                    .environmentObject(doseViewModel)
            }
            .padding()
        }
    }

    var monthCalendarView: some View {
        DatePicker("Select Date", selection: $selectedDate, in: ...Date(), displayedComponents:
    [.date])
            .datePickerStyle(GraphicalDatePickerStyle())
            .padding()
            .tint(.darkTeal)
    }

    var logSymptomsButton: some View {
        Button(action: {
            logSymptoms.toggle()
            selectedSymptoms = Set(symptomDataManager.fetchSymptoms(for: selectedDate))
        }) {
            VStack {
                Text("Log your Symptoms").bold()
                    .foregroundColor(colorScheme == .light ? Color.black : Color.black)
                    .padding(.bottom, 20)
                Image(systemName: "plus").bold()
                    .foregroundColor(colorScheme == .light ? Color.black : Color.black)
            }
            .frame(width: 130, height: 150)
            .background(colorScheme == .light ? Color.lightTeal : Color.darkTeal)
```

```
            .cornerRadius(20)
        }
        .fullScreenCover(isPresented: $logSymptoms) {
            NavigationStack {
                        SymptomsPopUp(selectedSymptoms: $selectedSymptoms, selectedDate:
    selectedDate)
                            .navigationBarItems(trailing: Button("Done") {
                                logSymptoms = false
                                saveSymptomsForSelectedDate()
                            })
                            .environmentObject(symptomDataManager) // Pass SymptomDataManager
    here
                    }
                }
    }

    var symptomsScrollView: some View {
        HStack {
            ScrollView(.horizontal, showsIndicators: false) {
                ScrollViewReader { proxy in
                    HStack(spacing: 10) {
                        logSymptomsButton
                        ForEach(Array(symptomsForSelectedDate), id: \.self) { symptom in
                            Button(action: {}) {
                                VStack {
                                    Text(formatSymptomName(symptom))
                                        .foregroundColor(colorScheme == .light ? Color.black :
    Color.black)
                                        .font(.system(size: 14)).bold()
                                        .padding(.leading, 10)
                                        .padding(.trailing, 10)
                                        .padding(.bottom, 10)
                                    if let emoji = healthKitManager.symptomEmojis[symptom] {
                                        if healthKitManager.symptomImageNeeded(symptom) {
                                            Image("\(symptom)")
                                                .resizable()
                                                .frame(width: 30, height: 30)
                                                .aspectRatio(contentMode: .fit)
                                                .padding(.top, 10)
                                        } else {
                                            Text(emoji)
                                                .font(.largeTitle)
                                                .padding(.top, 5)
                                        }
                                    }
                                }
                                .frame(width: 130, height: 150)
                                .background(colorScheme == .light ? Color.lightTeal : Color.
    darkTeal)
                                .cornerRadius(20)
                            }
                            .id(symptom)
                        }
                    }
                }
            }
        }
        .padding()
    }

    // MARK: Home Tab View
    var body: some View {
        NavigationStack {
            VStack {
                header
                VStack {
                    if weekView {
                        weekScrollView
```

```
                    }
                    ScrollView {
                        if weekView {
                            weekCircleView
                        } else {
                            monthCalendarView
                        }
                        if let dose = healthKitViewModel.doseRecords[selectedDate] {
                            if dose.doses != nil {
                                AboutYourDoseView(allergenDoses: dose)
                            }
                        }
                        symptomsScrollView
                    }
                }
            }
        }
        .onAppear(perform: {
            profileViewModel.loadProfileData()
            healthKitManager.loadSymptomsForSelectedDate(selectedDate: selectedDate, symptomDataManager: symptomDataManager) {
            }
            _ = healthKitViewModel.fetchSymptomsFromCoreData(for: selectedDate)
            healthKitViewModel.fetchDoseRecords()
            doseViewModel.loadDoses()
            profileImageViewModel.loadProfileImage()
        })

        Spacer()
    }

    // MARK: Symptom Functions
    private func saveSymptomsForSelectedDate() {
        symptomDataManager.saveSymptoms(for: selectedDate, symptoms: Array(selectedSymptoms))
    }

    private func saveHKSymptomsForSelectedDate() {
        for symptom in selectedSymptoms {
            healthKitManager.saveSymptom(symptom, for: selectedDate)
        }
    }

    private func formatSymptomName(_ identifier: String) -> String {
        let trimmed = identifier.replacingOccurrences(of: "HKCategoryTypeIdentifier", with: "")
        var formatted = ""
        for char in trimmed {
            if char.isUppercase {
                formatted += " " + String(char)
            } else {
                formatted += String(char)
            }
        }
        return formatted.trimmingCharacters(in: .whitespaces)
    }

    // MARK: Week View Functions
    private var WeeklyScrollCalendarView: some View {
        VStack {
            VStack {
                HStack(alignment: .center, spacing: 14) {
                    ForEach(0..<7, id: \.self) { index in
                        let day = Calendar.current.date(byAdding: .day, value: index, to: startOfWeek)!
                        CalendarDayView(selectedDate: $selectedDate, selectedColorIndex: $selectedColorIndex, date: day,  index: index)
                    }
                }
            }
```

```
            .padding()
        }
        .gesture(DragGesture(minimumDistance: 0, coordinateSpace: .local)
            .onEnded { value in
                let weekInterval: TimeInterval = 60 * 60 * 24 * 7
                let startOfNextWeekDate = startOfNextWeek(from: startOfWeek)
                if value.translation.width > 50 {
                    selectedDate = selectedDate.addingTimeInterval(-weekInterval)
                } else if value.translation.width < -50 && startOfNextWeekDate < Date() {
                    selectedDate = selectedDate.addingTimeInterval(weekInterval)
                }
            }
        )
    }

    private var startOfWeek: Date {
        let calendar = Calendar.current
        let components = calendar.dateComponents([.yearForWeekOfYear, .weekOfYear], from:
    selectedDate)
        var startComponents = DateComponents()
        startComponents.yearForWeekOfYear = components.yearForWeekOfYear
        startComponents.weekOfYear = components.weekOfYear
        startComponents.weekday = 1 // Sunday
        return calendar.date(from: startComponents)!
    }

    private var todaysDate: Date {
        let calendar = Calendar.current
        let today = calendar.startOfDay(for: Date())
        return today
    }

    private func startOfNextWeek(from date: Date) -> Date {
        let calendar = Calendar.current
        var startComponents = calendar.dateComponents([.yearForWeekOfYear, .weekOfYear, .weekday
    ], from: date)
        if let currentWeekStartDate = calendar.date(from: startComponents) {
            startComponents.weekOfYear? += 1 // Increment to the next week
            return calendar.date(from: startComponents) ?? currentWeekStartDate
        }
        return date
    }

    private var monthHeader: String {
        let dateFormatter = DateFormatter()
        dateFormatter.dateFormat = "MMMM yyyy"
        return dateFormatter.string(from: startOfWeek)
    }

    // MARK: Day of Treatment Functions
    func calculateDayOfTreatment() -> Int {
        if let firstSymptomDate = symptomDataManager.symptomRecords.keys.sorted().first,
           let firstDoseDate = healthKitViewModel.doseRecords.keys.sorted().first {

            let firstDates = [firstSymptomDate, firstDoseDate]
            let minDate = firstDates.min()!

            let calendar = Calendar.current
            let components = calendar.dateComponents([.day], from: minDate, to: selectedDate)
            return (components.day ?? 0) + 1 // Adding 1 to start from Day 1
        }

        if let firstSymptomDate = symptomDataManager.symptomRecords.keys.sorted().first {
            let calendar = Calendar.current
            let components = calendar.dateComponents([.day], from: firstSymptomDate, to:
    selectedDate)
            return (components.day ?? 0) + 1 // Adding 1 to start from Day 1
        }
```

```
339
340         if let firstDoseDate = healthKitViewModel.doseRecords.keys.sorted().first {
341             let calendar = Calendar.current
342             let components = calendar.dateComponents([.day], from: firstDoseDate, to:
    selectedDate)
343             return (components.day ?? 0) + 1 // Adding 1 to start from Day 1
344         }
345
346         return 1
347     }
348 }
349
350 // MARK: Preview
351 #Preview {
352     HomeView()
353         .environmentObject(ProfileViewModel())
354         .environmentObject(HealthKitViewModel())
355 }
```

A.35 ProfileImage.swift

```
1  import SwiftUI
2  import PhotosUI
3
4  struct ProfileImage: View {
5    let imageState: ProfileModel.ImageState
6
7    var body: some View {
8      switch imageState {
9      case .success(let image):
10       image.resizable()
11     case .loading:
12       ProgressView()
13     case .empty:
14       Image(systemName: "person.fill")
15         .font(.system(size: 25))
16         .foregroundColor(.white)
17     case .failure:
18       Image(systemName: "exclamationmark.triangle.fill")
19         .font(.system(size: 25))
20         .foregroundColor(.white)
21     }
22   }
23 }
24
25 struct CircularProfileImage: View {
26   let imageState: ProfileModel.ImageState
27
28   var body: some View {
29     ProfileImage(imageState: imageState)
30             .scaledToFill()
31       .clipShape(Circle())
32       .frame(width: 35, height: 35)
33       .background {
34         Circle().fill(
35           LinearGradient(
36             colors: [.lightTeal, .darkTeal],
37             startPoint: .top,
38             endPoint: .bottom
39           )
40         )
41       }
42   }
43 }
44
45 struct EditableCircularProfileImage: View {
46   @ObservedObject var viewModel: ProfileModel
```

```
47
48    var body: some View {
49      CircularProfileImage(imageState: viewModel.imageState)
50        .overlay(alignment: .bottomTrailing) {
51          PhotosPicker(selection: $viewModel.imageSelection,
52                 matching: .images,
53                 photoLibrary: .shared()) {
54            Image(systemName: "pencil.circle.fill")
55              .symbolRenderingMode(.multicolor)
56              .font(.system(size: 10))
57              .foregroundColor(.accentColor)
58          }
59                   .onDisappear {
60                       viewModel.saveProfileImage()
61                   }
62          .buttonStyle(.borderless)
63        }
64    }
65  }
```

A.36 SectionHeaderView.swift

```
1  //
2  //  SectionHeaderView.swift
3  //  SeniorDesign
4  //
5  //  Created by Maddie on 1/1/24.
6  //
7
8  import SwiftUI
9
10 struct SectionHeaderView: View {
11     let title: String
12
13     var body: some View {
14         HStack {
15             Text(title)
16                 .font(.title3.bold())
17                 .foregroundColor(.primary)
18             Spacer()
19         }
20         .padding(.vertical, 8)
21     }
22 }
```

A.37 SettingsView.swift

```
1  //
2  //  SettingsView.swift
3  //  SeniorDesign
4  //
5  //  Created by Maddie on 10/18/23.
6  //
7
8  import HealthKit
9  import SwiftUI
10
11 struct SettingsView: View {
12     // MARK: View Models
13     @EnvironmentObject var profileViewModel: ProfileViewModel
14     @EnvironmentObject var healthKitViewModel: HealthKitViewModel
15     @EnvironmentObject var doseViewModel: DoseViewModel
16     @EnvironmentObject var profileImageViewModel: ProfileModel
17
18     @State private var isAddingOtherAllergen = false
19     @State private var isAddingOtherAllergenEmoji = false
```

```
    @State private var otherAllergenName: String = ""
    @State private var otherAllergenEmoji: String = ""

    @StateObject var appState = AppState()

    @Environment(\.colorScheme) var colorScheme

    // MARK: UI Elements
    var personalInformationSection: some View {
        Section(header: Text("Personal Information").font(.title3).bold()) {
            HStack {
                Text("Photo").bold()
                    .padding(.leading, 5)
                Spacer()
                EditableCircularProfileImage(viewModel: profileImageViewModel)
                    .padding(.trailing, 5)
            }
            HStack {
                Text("Name")
                    .font(.body.bold())
                    .padding(.leading, 5)
                Spacer()
                TextField("Name", text: $profileViewModel.profileData.name)
                    .textFieldStyle(RoundedBorderTextFieldStyle())
                    .padding(.trailing, 5)
                    .onChange(of: profileViewModel.profileData.name) { _ in
                        profileViewModel.saveProfileData()
                    }
                    .frame(width: 120)
            }
            DatePicker("Birthdate", selection: $profileViewModel.profileData.dateOfBirth,
displayedComponents: .date)
            .datePickerStyle(CompactDatePickerStyle())
            .padding(.leading, 5)
            .padding(.trailing, 5)
            .font(.body.bold())
            .onChange(of: profileViewModel.profileData.dateOfBirth) { _ in
                profileViewModel.saveProfileData()
            }

        }
        .padding(.top, 10)
    }

    var allergensSection: some View {
        Section(header: HStack {
            SectionHeaderView(title: "Allergens")
            Spacer()
            Button(action: {
                profileViewModel.addNewAllergen()
            }) {
                HStack {
                    Image(systemName: "plus.circle")
                        .foregroundColor(.darkTeal)
                }
            }
            .padding(.trailing, 20)
        }) {
            ForEach(0..<max(1, profileViewModel.numAllergens), id: \.self) { index in
                if index < profileViewModel.numAllergens {
                    HStack {
                        VStack {
                            Picker("Select an Allergen", selection: $profileViewModel.profileData
.allergens[index]) {
                                ForEach($profileViewModel.profileData.commonAllergens, id: \.self
) { allergen in
                                    Text(allergen.wrappedValue).tag(allergen.wrappedValue)
                                }
```

```
                                Text("Other").tag("Other")
                            }
                            .tint(colorScheme == .dark ? .white : .black)
                            .pickerStyle(DefaultPickerStyle())
                            .padding(.bottom, 10)
                            .onChange(of: profileViewModel.profileData.allergens[index]) {
    newValue in
                                if newValue == "Other" {
                                    isAddingOtherAllergen = true
                                } else if !profileViewModel.profileData.allergens.contains(
    newValue) && newValue != "Select an Allergen" && newValue != "" && newValue != "Other"{
                                    profileViewModel.profileData.allergens.append(newValue)
                                    profileViewModel.profileData.allergens.removeAll { $0 == "
    Other" }
                                    profileViewModel.profileData.commonAllergens.sort()
                                    profileViewModel.saveProfileData()
                                }
                            }
                            .onAppear {
                                if profileViewModel.profileData.allergens[index].isEmpty,
                                   let firstCommonAllergen = $profileViewModel.profileData.
    commonAllergens.first {
                                    profileViewModel.profileData.allergens[index] =
    firstCommonAllergen.wrappedValue
                                }
                            }
                            Spacer()
                        }

                        VStack {
                            Button(action: {
                                profileViewModel.removeSelectedAllergen(at: index)
                            }) {
                                Image(systemName: "minus.circle").foregroundColor(.red)
                                    .padding(.top, 7)
                            }
                            Spacer()
                        }
                    }
                }
            }
        }
        .padding(.top, 10)
        .alert("Enter Other Allergen", isPresented: $isAddingOtherAllergen) {
            TextField("Other Allergen", text: $otherAllergenName)
            Button("OK", action: submitOtherAllergen)
        }
        .alert("Enter Emoji for Your Allergen", isPresented: $isAddingOtherAllergenEmoji) {
            TextField("Emoji", text: $otherAllergenEmoji)
            Button("OK", action: submitOtherEmoji)
        }
    }

    func submitOtherAllergen() {
        if !profileViewModel.profileData.allergens.contains(otherAllergenName) {
            profileViewModel.profileData.allergens.append(otherAllergenName)
            profileViewModel.profileData.commonAllergens.append(otherAllergenName)
            profileViewModel.profileData.allergens.removeAll { $0 == "Other" }
            profileViewModel.profileData.commonAllergens.sort()
            profileViewModel.saveProfileData()
        }
        isAddingOtherAllergen = false
        isAddingOtherAllergenEmoji = true
    }

    func submitOtherEmoji() {
        profileViewModel.profileData.allergenEmojiMap[otherAllergenName] = otherAllergenEmoji
        profileViewModel.profileData.commonAllergens.sort()
```

```
        profileViewModel.saveProfileData()
        isAddingOtherAllergenEmoji = false
    }

    var shareDataWithHealthToggle: some View {
        Toggle("Share Data with Apple Health", isOn: $profileViewModel.profileData.
    shareDataWithAppleHealth)
            .toggleStyle(SwitchToggleStyle(tint: .darkTeal))
            .onChange(of: profileViewModel.profileData.shareDataWithAppleHealth) { newValue in
                healthKitViewModel.healthRequest()
                profileViewModel.saveProfileData()
            }
    }

    var protectDataWithFaceID: some View {
        Toggle("Enable Passcode, TouchID, or FaceID", isOn: $profileViewModel.profileData.
    useFaceID)
            .toggleStyle(SwitchToggleStyle(tint: .darkTeal))
            .onChange(of: profileViewModel.profileData.useFaceID) { newValue in
                profileViewModel.saveProfileData()
            }
    }

    var preferencesSection: some View {
        Section(header: Text("Preferences").font(.title3).bold()) {
            shareDataWithHealthToggle
            protectDataWithFaceID
        }.padding(.top, 10)
    }

    var header: some View {
        HStack {
            Text("Profile")
                .font(.largeTitle.bold())
                .padding(.leading, 20)
                .padding(.top, 10)
            Spacer()
        }
    }

    // MARK: Settings View
    var body: some View {
        VStack {
            if appState.hasAppBeenOpenedBefore {
                header
            }
            ScrollView {
                VStack(alignment: .leading) {
                    VStack(alignment: .leading) {
                        personalInformationSection
                        Divider()
                            .padding(.leading, 20)
                            .padding(.trailing, 20)
                            .padding(.top, 20)
                        allergensSection
                        Spacer()
                        Divider()
                            .padding(.leading, 20)
                            .padding(.trailing, 20)
                        preferencesSection
                        Spacer()
                        Divider()
                            .padding(20)
//                        if appState.hasAppBeenOpenedBefore {
//                            HStack {
//                                Spacer()
//                                Text("Log Out")
//                                    .foregroundColor(.red)
```

```
//                                    .onTapGesture {
//                                        // TODO: Add logic to log out
//                                    }
//                                Spacer()
//                            }
//                            Spacer()
//                        }
                    }
                    .padding()
                }
            }
            .onDisappear {
                profileViewModel.profileData.allergens = profileViewModel.profileData.allergens.filter { !$0.isEmpty }
                profileViewModel.saveProfileData()
                profileImageViewModel.saveProfileImage()
            }
            .onAppear(perform: {
                profileViewModel.loadProfileData()
                profileImageViewModel.loadProfileImage()
                doseViewModel.loadDoses()
                profileViewModel.profileData.commonAllergens.sort()
            })
        }
    }
}

// MARK: Preview
#Preview {
    SettingsView()
        .environmentObject(HealthKitViewModel())
        .environmentObject(ProfileViewModel())
}
```

A.38 SymptomsPopUp.swift

```
import HealthKit
import SwiftUI

struct SymptomsPopUp: View {
    // MARK: Variables
    @EnvironmentObject var symptomDataManager: SymptomDataManager
    @Binding var selectedSymptoms: Set<String>
    let selectedDate: Date
    @State var contactDoctor = false
    let healthKitManager = HealthKitManager()
    let columns: [GridItem] = [GridItem(.flexible()), GridItem(.flexible()), GridItem(.flexible())]

    @Environment(\.presentationMode) var presentationMode
    @Environment(\.colorScheme) var colorScheme

    // MARK: Initializer
    init(selectedSymptoms: Binding<Set<String>>, selectedDate: Date) {
        self._selectedSymptoms = selectedSymptoms
        self.selectedDate = selectedDate
    }

    private func foregroundColor(for symptom: HKSampleType) -> Color {
        if selectedSymptoms.contains(symptom.identifier.description) {
            return colorScheme == .light ? Color.white : Color.white
        } else {
            return colorScheme == .light ? Color.black : Color.black
        }
    }

    private func backgroundColor(for symptom: HKSampleType) -> Color {
```

```
        if selectedSymptoms.contains(symptom.identifier.description) {
            return colorScheme == .light ? Color.darkTeal : Color.darkerTeal
        } else {
            return colorScheme == .light ? Color.lightTeal : Color.darkTeal
        }
    }

    // MARK: Symptoms Pop Up View
    var body: some View {
        ScrollView {
            LazyVGrid(columns: columns, spacing: 10) {
                ForEach(healthKitManager.symptoms, id: \.self) { symptom in
                    Button(action: {
                        toggleSymptom(symptom.identifier.description)
                    }) {
                        VStack {
                            Text(formatSymptomName(symptom.identifier.description))
                                .padding(.leading, 10)
                                .padding(.trailing, 10)
                                .font(.system(size: 14)).bold()
                                .foregroundColor(foregroundColor(for: symptom))
                            if let emoji = healthKitManager.symptomEmojis[symptom.identifier.
    description] {
                                if healthKitManager.symptomImageNeeded(symptom.identifier.
    description) {
                                    Image("\(symptom.identifier.description)")
                                        .resizable()
                                        .frame(width: 30, height: 30)
                                        .aspectRatio(contentMode: .fit)
                                        .padding(.top, 10)
                                } else {
                                    Text(emoji)
                                        .font(.largeTitle).bold()
                                        .padding(.top, 5)
                                }
                            }
                        }
                        .frame(width: UIScreen.main.bounds.width/3 - 10, height: 150)
                        .background(backgroundColor(for: symptom))
                        .cornerRadius(20)
                    }
                }
            }
        }
        .alert("Have you contacted your doctor?     ", isPresented: $contactDoctor) {
            Text("You have selected symptoms that may require immediate attention. Please contact
     your doctor or call 911.")
            Button("OK", action: {contactDoctor = false})
        }
        .onDisappear(perform: saveSymptomsForSelectedDate)
        .padding(10)
    }

    // MARK: Functions
    private func formatSymptomName(_ identifier: String) -> String {
        let trimmed = identifier.replacingOccurrences(of: "HKCategoryTypeIdentifier", with: "")
        var formatted = ""
        for char in trimmed {
            if char.isUppercase {
                formatted += " " + String(char)
            } else {
                formatted += String(char)
            }
        }
        return formatted.trimmingCharacters(in: .whitespaces)
    }

    private func saveSymptomsForSelectedDate() {
```

```
        symptomDataManager.saveSymptoms(for: selectedDate, symptoms: Array(selectedSymptoms))

        let worrisomeSymptoms = ["HKCategoryTypeIdentifierFever", "
    HKCategoryTypeIdentifierVomiting", "HKCategoryTypeIdentifierChestTightnessOrPain", "
    HKCategoryTypeIdentifierChills", "HKCategoryTypeIdentifierRapidPoundingOrFlutteringHeartbeat"
    , "HKCategoryTypeIdentifierShortnessOfBreath", "HKCategoryTypeIdentifierMemoryLapse", "
    HKCategoryTypeIdentifierMoodChanges", "HKCategoryTypeIdentifierFainting", "
    HKCategoryTypeIdentifierDizziness", "HKCategoryTypeIdentifierCoughing"]

            let hasWorrisomeSymptoms = selectedSymptoms.contains { worrisomeSymptoms.contains($0)
     }

            if hasWorrisomeSymptoms {
                contactDoctor = true
            }
    }

    private func toggleSymptom(_ identifier: String) {
        if selectedSymptoms.contains(identifier) {
            selectedSymptoms.remove(identifier)
        } else {
            selectedSymptoms.insert(identifier)
        }
    }
}
```

A.39 DosesForMonthView.swift

```
//
//  DosesForMonthView.swift
//  SeniorDesign
//
//  Created by Maddie on 3/7/24.
//
import SwiftUI
import CareKitUI

struct DosesForMonthView: View {
    @ObservedObject var healthKitViewModel: HealthKitViewModel
    @Environment(\.colorScheme) var colorScheme

    private var allDates: [Date] {
        let calendar = Calendar.current
        let currentDate = Date()

        guard let startDate = calendar.date(byAdding: .month, value: -1, to: currentDate) else {
            fatalError("Error calculating start date of the past month")
        }

        var dates: [Date] = []
        var currentDateIter = startDate
        while currentDateIter <= currentDate {
            dates.append(currentDateIter)
            currentDateIter = calendar.date(byAdding: .day, value: 1, to: currentDateIter)!
        }

        return dates
    }

    var body: some View {
        return VStack {
            DosesForMonthLinePlotView(allDates: allDates, healthKitViewModel: healthKitViewModel,
     colorScheme: colorScheme)
        }
    }
}

```

```
struct DosesForMonthLinePlotView: UIViewRepresentable {
    let allDates: [Date]
    let healthKitViewModel: HealthKitViewModel
    let colorScheme: ColorScheme

    func makeUIView(context: Context) -> OCKCartesianChartView {
        let chartView = OCKCartesianChartView(type: .line)
        chartView.headerView.titleLabel.text = "Doses This Month"

        var dataSeries = OCKDataSeries(values: allDates.map { date in
            let doseCount = Double(healthKitViewModel.doseRecords.values.filter { record in
                if let doseDate = record.date {
                    let components = Calendar.current.dateComponents([.day, .month, .year], from: doseDate)
                    return components == Calendar.current.dateComponents([.day, .month, .year], from: date)
                }
                return false
            }.count)
            return CGFloat(doseCount)
        }, title: "Dose Taken", color: colorScheme == .light ? UIColor(.darkTeal) : UIColor(.lightTeal))
        dataSeries.size = 3

        let (dosesTaken, dosesSkipped) = countDoses()
        chartView.headerView.detailLabel.text = "You took \(dosesTaken) doses and skipped \(dosesSkipped) doses this month."
        chartView.graphView.dataSeries = [dataSeries]
        chartView.graphView.yMaximum = 1
        chartView.graphView.yMinimum = 0
        return chartView
    }

    func updateUIView(_ uiView: OCKCartesianChartView, context: Context) {
        // Update the view if needed
    }

    private func countDoses() -> (Int, Int) {
        var dosesTaken = 0
        var dosesSkipped = 0

        for date in allDates {
            if healthKitViewModel.doseRecords.values.contains(where: { record in
                if let doseDate = record.date {
                    let components = Calendar.current.dateComponents([.day, .month, .year], from: doseDate)
                    return components == Calendar.current.dateComponents([.day, .month, .year], from: date)
                }
                return false
            }) {
                dosesTaken += 1
            } else {
                dosesSkipped += 1
            }
        }

        return (dosesTaken, dosesSkipped)
    }
}

struct DosesForMonthView_Previews: PreviewProvider {
    static var previews: some View {
        DosesForMonthView(healthKitViewModel: HealthKitViewModel())
    }
}
```

A.40 InsightsTip.swift

```swift
//
//  InsightsTip.swift
//  SeniorDesign
//
//  Created by Maddie on 3/7/24.
//

import Foundation
import TipKit

struct InsightsTip: Tip, Identifiable {
    var id = UUID()
    var title: Text {
        Text("Unlock Your Health Insights")
    }

    var message: Text? {
        Text("Discover personalized stats and insightful graphs based on your symptoms and doses. Track your progress, gain valuable health insights, and make informed decisions on your journey to wellness.")
    }

    var image: Image? {
        Image(systemName: "waveform.badge.magnifyingglass")
    }
}
```

A.41 InsightsView.swift

```swift
//
//  InsightsView.swift
//  SeniorDesign
//
//  Created by Maddie on 10/18/23.
//

import SwiftUI
import CoreData
import TipKit

struct InsightsView: View {
    // MARK: View Model
    @ObservedObject var healthKitViewModel: HealthKitViewModel
    @ObservedObject var symptomDataManager = SymptomDataManager()
    @State private var slicesWithLabels: [(Double, Color, String)] = []
    @Environment(\.colorScheme) var colorScheme

    var header: some View {
        HStack {
            Text("Insights")
                .font(.largeTitle.bold())
                .padding()
            Spacer()
        }
    }

    var insightsTip = InsightsTip()

    // MARK: Insights View
    var body: some View {
        VStack {
            header
            ScrollView {
                if #available(iOS 17.0, *) {
```

```
                    TipView(insightsTip, arrowEdge: .none)
                        .tipCornerRadius(15)
                        .padding(.leading, 25)
                        .padding(.trailing, 25)
                        .padding(.bottom, 20)
                }
                VStack {
                    VStack {
                        Spacer()
                        HStack {
                            Spacer()
                            Text("Today is day \(calculateDayOfTreatment()) of treatment!").bold
    ()
                                .padding(.leading, 20)
                                .padding(.trailing, 20)
                                .font(.subheadline)
                                .foregroundColor(colorScheme == .light ? .black : .black)
                            Spacer()
                        }
                        Spacer()
                    }
                    .frame(minWidth: 0, maxWidth: .infinity, minHeight: 50, maxHeight: .infinity,
     alignment: .topLeading)
                    .background(colorScheme == .light ? Color.lightBlue : Color.darkTeal)
                    .cornerRadius(20)
                    .padding(.leading, 20)
                    .padding(.trailing, 20)
                    .padding(.bottom, 20)
                    SymptomsOverTimeChartView(symptomDataManager: symptomDataManager)
                    .frame(minHeight: 200)
                    .padding(.leading, 20)
                    .padding(.trailing, 20)
                    .padding(.bottom, 20)
                    LastReactionInsight(symptomDataManager: symptomDataManager)
                    .padding(.leading, 20)
                    .padding(.trailing, 20)
                    .padding(.bottom, 20)
                    SymptomsThisWeekChartView(symptomDataManager: symptomDataManager)
                    .frame(minHeight: 200)
                    .padding(.leading, 20)
                    .padding(.trailing, 20)
                    .padding(.bottom, 20)
                    DosesForMonthView(healthKitViewModel: healthKitViewModel)
                    .frame(minHeight: 230)
                    .padding(.leading, 20)
                    .padding(.trailing, 20)
                    .padding(.bottom, 20)
                    VStack {
                        if !symptomDataManager.fetchAllSymptomsWithoutRefresh().isEmpty {
                            HStack {
                                Text("Symptoms").bold()
                                    .font(.title3)
                                    .foregroundColor(colorScheme == .light ? .black : .black)
                                    .padding()
                                Spacer()
                            }
                            SymptomsPieChartView(symptoms: symptomDataManager.
    fetchAllSymptomsWithoutRefresh(), slices: $slicesWithLabels)
                                .frame(height: 300)
                                .padding(.leading, 20)
                                .padding(.trailing, 20)
                            Legend(slicesWithLabels: slicesWithLabels)
                                .padding()
                        }
                    }
                    .frame(minWidth: 0, maxWidth: .infinity, minHeight: 0, maxHeight: .infinity,
    alignment: .topLeading)
                    .background(colorScheme == .light ? Color.lightBlue : Color.darkTeal)
```

```
                    .cornerRadius(20)
                    .padding(.leading, 20)
                    .padding(.trailing, 20)
                    .padding(.bottom, 20)
                    .onAppear {
                        self.slicesWithLabels = calculateSlices(symptoms: symptomDataManager.fetchAllSymptomsWithoutRefresh())
                    }
                }
            }
        }
        .onAppear(perform: {
            healthKitViewModel.fetchDoseRecords()
        })
        .task {
            if #available(iOS 17.0, *) {
                try? Tips.configure([
                    .displayFrequency(.immediate),
                    .datastoreLocation(.applicationDefault)
                ])
            }
        }
    }

    private func calculateSlices(symptoms: [String]) -> [(Double, Color, String)] {
        let groupedSymptoms = Dictionary(grouping: symptoms, by: { $0 })
        var calculatedSlices: [(Double, Color, String)] = []

        for (_, (key, value)) in groupedSymptoms.enumerated() {
            let formattedKey = key.splitCamelCase()
            let slice = (Double(value.count) / Double(symptoms.count), Color.mutedRandom, formattedKey)
            calculatedSlices.append(slice)
        }

        return calculatedSlices
    }

    func calculateDayOfTreatment() -> Int {
        if let firstSymptomDate = symptomDataManager.symptomRecords.keys.sorted().first,
           let firstDoseDate = healthKitViewModel.doseRecords.keys.sorted().first {

            let firstDates = [firstSymptomDate, firstDoseDate]
            let minDate = firstDates.min()!

            let calendar = Calendar.current
            let components = calendar.dateComponents([.day], from: minDate, to: Date())
            return (components.day ?? 0) + 1
        }

        if let firstSymptomDate = symptomDataManager.symptomRecords.keys.sorted().first {
            let calendar = Calendar.current
            let components = calendar.dateComponents([.day], from: firstSymptomDate, to: Date())
            return (components.day ?? 0) + 1
        }

        if let firstDoseDate = healthKitViewModel.doseRecords.keys.sorted().first {
            let calendar = Calendar.current
            let components = calendar.dateComponents([.day], from: firstDoseDate, to: Date())
            return (components.day ?? 0) + 1
        }

        return 1
    }
}
```

A.42 LastReactionInsight.swift

```swift
//
//  LastReactionInsight.swift
//  SeniorDesign
//
//  Created by Maddie on 3/7/24.
//

import Foundation
import SwiftUI

struct LastReactionInsight: View {
    @ObservedObject var symptomDataManager: SymptomDataManager
    @Environment(\.colorScheme) var colorScheme

    var body: some View {
        VStack {
            if let lastReactionDate = symptomDataManager.lastReactionDate() {
                let daysSinceLastReaction = Calendar.current.dateComponents([.day], from: lastReactionDate, to: Date()).day ?? 0

                HStack(alignment: .center) {
                    Spacer()
                    if daysSinceLastReaction <= 2 {
                        Text("Don't worry, it's okay. Things will improve. It's been \(daysSinceLastReaction) days since your last reaction.").bold()
                            .foregroundColor(colorScheme == .light ? .black : .black)
                    } else {
                        Text("Congratulations! It's been \(daysSinceLastReaction) days since your last reaction!").bold()
                            .foregroundColor(colorScheme == .light ? .black : .black)
                    }
                    Spacer()
                }
                .font(.subheadline)
                .padding()
            }
        }
        .frame(minWidth: 0, maxWidth: .infinity, minHeight: 0, maxHeight: .infinity, alignment: .topLeading)
        .background(colorScheme == .light ? Color.lightBlue : Color.darkTeal)
        .cornerRadius(20)
    }
}
```

A.43 Legend.swift

```swift
//
//  Legend.swift
//  SeniorDesign
//
//  Created by Maddie on 3/7/24.
//

import Foundation
import SwiftUI

struct Legend: View {
    let slicesWithLabels: [(Double, Color, String)]

    var body: some View {
        VStack(alignment: .leading) {
            let columns = 2
            let rows = (slicesWithLabels.count + columns - 1) / columns

```

```
            ForEach(0..<rows, id: \.self) { row in
                HStack(spacing: 0) {
                    Spacer()
                    ForEach(0..<columns, id: \.self) { column in
                        let index = row * columns + column
                        if index < slicesWithLabels.count {
                            let symptomName = slicesWithLabels[index].2
                            LegendItem(color: slicesWithLabels[index].1, label: "\(symptomName)")
                                .frame(alignment: .leading)
                        }
                        Spacer()
                    }
                }
                .frame(alignment: .leading)
            }
        }
        .padding(.top)
        .frame(maxWidth: 300)
    }
}
```

A.44 LegendItem.swift

```
//
//  LegendItem.swift
//  SeniorDesign
//
//  Created by Maddie on 3/7/24.
//

import Foundation
import SwiftUI

struct LegendItem: View {
    let color: Color
    let label: String

    @Environment(\.colorScheme) var colorScheme

    var body: some View {
        HStack {
            Circle()
                .fill(color)
                .frame(width: 10, height: 10)
            Text(label)
                .font(.system(size: 14)).bold()
                .foregroundColor(colorScheme == .light ? .black : .black)
        }
        .padding(.horizontal)
    }
}
```

A.45 SymptomsOverTimeChartView.swift

```
//
//  SymptomsOverTimeChartView.swift
//  OIT
//
//  Created by Maddie on 4/3/24.
//

import Foundation
import SwiftUI
import CareKitUI
import CareKit

```

```swift
struct SymptomsOverTimeChartView: UIViewRepresentable {
    let symptomDataManager: SymptomDataManager
    @Environment(\.colorScheme) var colorScheme

    func makeUIView(context: Context) -> OCKCartesianChartView {
        let chartView = OCKCartesianChartView(type: .bar)
        chartView.headerView.titleLabel.text = "Symptoms Over Time"

        _ = symptomDataManager.fetchAllSymptoms()
        let symptomRecordsByDate = symptomDataManager.symptomRecords
        var dataPoints: [(Date, Int)] = []
        for (date, symptoms) in symptomRecordsByDate {
            dataPoints.append((date, symptoms.count))
        }
        dataPoints.sort { $0.0 < $1.0 }
        _ = dataPoints.map { $0.0 }
        let counts = dataPoints.map { Double($0.1) }

        let dataSeries = OCKDataSeries(values: counts.map { CGFloat($0) }, title: "Symptoms",
    color: colorScheme == .light ? UIColor(.darkTeal) : UIColor(.lightTeal))
        chartView.graphView.dataSeries = [dataSeries]

        let dateFormatter = DateFormatter()
        dateFormatter.dateFormat = "MMM dd"
        return chartView
    }

    func updateUIView(_ uiView: OCKCartesianChartView, context: Context) {
        // Update the view if needed
    }
}
```

A.46 SymptomsThisWeekChartView.swift

```swift
//
//  SymptomsThisWeekChartView.swift
//  OIT
//
//  Created by Maddie on 4/3/24.
//

import Foundation
import SwiftUI
import CareKitUI
import CareKit

struct SymptomsThisWeekChartView: UIViewRepresentable {
    let symptomDataManager: SymptomDataManager

    func makeUIView(context: Context) -> OCKCartesianChartView {
        let chartView = OCKCartesianChartView(type: .bar)
        chartView.headerView.titleLabel.text = "Symptoms This Week"

        let oneWeekAgo = Calendar.current.date(byAdding: .day, value: -7, to: Date()) ?? Date()
        let symptomRecordsPastWeek = symptomDataManager.symptomRecords.filter { $0.key >=
    oneWeekAgo }
        let symptomRecordsByDate = symptomRecordsPastWeek.sorted(by: { $0.key < $1.key })
        var allSymptoms: Set<String> = []
        for (_, symptoms) in symptomRecordsByDate {
            allSymptoms.formUnion(symptoms)
        }

        for (index, symptom) in allSymptoms.enumerated() {
            let counts = symptomRecordsByDate.map { (_, symptoms) in
                symptoms.contains(symptom) ? 1.0 : 0.0
            }
            let dataSeries = OCKDataSeries(values: counts.map { CGFloat($0) }, title: symptom.
```

```
    splitCamelCase(), color: UIColor(Color.mutedRandom))
            chartView.graphView.dataSeries.append(dataSeries)
        }

        let dateFormatter = DateFormatter()
        dateFormatter.dateFormat = "EEEEE"
        chartView.graphView.horizontalAxisMarkers = symptomRecordsByDate.map { dateFormatter.
    string(from: $0.key) }
        chartView.graphView.yMaximum = 1
        chartView.graphView.yMinimum = 0

        return chartView
    }

    func updateUIView(_ uiView: OCKCartesianChartView, context: Context) {
        // Update the view if needed
    }
}
```

A.47 GetSetUpView.swift

```
//
//  GetSetUpView.swift
//  SeniorDesign
//
//  Created by Maddie on 9/25/23.
//

import SwiftUI

struct GetSetUpView: View {
    // MARK: View Models
    @EnvironmentObject var profileViewModel: ProfileViewModel
    @EnvironmentObject var healthKitViewModel: HealthKitViewModel
    @EnvironmentObject var appState: AppState


    // MARK: UI Elements
    var setUpTitle: some View {
        Text("Set Up")
            .font(.largeTitle.bold())
    }

    var setUpInstructions: some View {
        Text("Fill out the following information to get started.")
            .font(.subheadline)
            .padding(.top, 1)
    }

    var getStartedButton: some View {
        Text("Get Started                    ")
            .font(.caption)
            .foregroundColor(Color.white)
            .padding()
            .background(Color.darkTeal)
            .cornerRadius(30)
    }

    // MARK: Get Set Up View
    var body: some View {
        ScrollView {
            VStack(alignment: .leading) {
                setUpTitle
                setUpInstructions
                SettingsView()
                    .environmentObject(profileViewModel)
                    .environmentObject(healthKitViewModel)
```

```
                Spacer()
                HStack(alignment: .center) {
                    NavigationLink(destination: TabbedApplicationView().
    navigationBarBackButtonHidden(true)) {
                        Spacer()
                        getStartedButton
                            .onTapGesture {
                                appState.hasAppBeenOpenedBefore = true
                            }
                        Spacer()
                    }
                }
                .padding(.top, 20)
            }
            .padding()
        }
    }
}

// MARK: Preview
#Preview {
    GetSetUpView()
        .environmentObject(ProfileViewModel())
        .environmentObject(HealthKitViewModel())
}
```

A.48 GetStartedView.swift

```
//
//  GetStartedView.swift
//  SeniorDesign
//
//  Created by Maddie on 9/25/23.
//

import SwiftUI

struct GetStartedView: View {
    @EnvironmentObject var appState: AppState
    // MARK: UI Elements
    var getStartedButton: some View {
        NavigationLink(destination: GetSetUpView().navigationBarBackButtonHidden(true)) {
            Text("Get Started         ")
                .font(.caption)
                .foregroundColor(Color.white)
                .padding()
                .background(Color.darkTeal)
                .cornerRadius(30)
        }
        .padding()
    }

    var appTitle: some View {
        VStack {
            Spacer()
            Text("OIT")
                .font(.largeTitle.bold().monospaced())
                .foregroundColor(.white)
                .padding(.bottom, 1)
            Text("oral immunotherapy tracker")
                .font(.callout.lowercaseSmallCaps())
                .fontDesign(.rounded)
                .foregroundColor(.white)
                .padding(.bottom, 20)
        }
    }

```

```
    // MARK: Get Started View
    var body: some View {
        NavigationStack {
            VStack {
                ZStack {
                    Color.darkTeal
                        .frame(maxHeight: .infinity)
                    appTitle
                }
                VStack {}
                ZStack {
                    Color.lightTeal
                        .frame(maxHeight: .infinity)
                    VStack {
                        getStartedButton
                        Spacer()
                    }
                }
            }
            .edgesIgnoringSafeArea(.all)
        }
    }
}

// MARK: Preview
#Preview {
    GetStartedView()
}
```

Bibliography

[1] Food Allergy Research and Education, "Preventing Food Allergies: Early Interventions — FARE," *www.foodallergy.org.* https://www.foodallergy.org/research-innovation/accelerating-innovation/early-introduction-and-food-allergy-prevention

[2] D. Brown et al., "Food allergy-related bullying and associated peer dynamics among black and white children in the forward study," *Annals of Allergy, Asthma and Immunology, vol. 126, no. 3*, 2021. doi:10.1016/j.anai.2020.10.013

[3] American Academy of Allergy Asthma and Immunology, "The Current State of Oral Immunotherapy (OIT) for the Treatment of Food Allergy," *Aaaai.org*, 2020. https://www.aaaai.org/Tools-for-the-Public/Conditions-Library/Allergies/The-Current-State-of-Oral-Immunotherapy

[4] The American College of Allergy, Asthma and Immunology, "Food Allergy Avoidance," ACAAI Public Website. https://acaai.org/allergies/management-treatment/living-with-allergies/food-allergy-avoidance/

[5] M. Bilucaglia et al., "Looking through blue glasses: bioelectrical measures to assess the awakening after a calm situation," *2019 41st Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, Berlin, Germany, 2019, pp. 526-529, doi: 10.1109/EMBC.2019.8856486.

[6] K. Blumchen et al., "Post hoc analysis examining symptom severity reduction and symptom absence during food challenges in individuals who underwent oral immunotherapy for peanut allergy: results from three trials," Allergy, Asthma, and Clinical Immunology: Official Journal of the Canadian Society of Allergy and Clinical Immunology, vol. 19, no. 1, p. 21, Mar. 2023, doi: https://doi.org/10.1186/s13223-023-00757-8.

[7] N. Wanniang et al., "Immune signatures predicting the clinical outcome of peanut oral immunotherapy: where we stand," Frontiers in Allergy, vol. 4, Oct. 2023, doi: https://doi.org/10.3389/falgy.2023.1270344.

[8] S. A. Nairn, "Creating an (ethical) epistemic space for the normalization of clinical and 'real food' oral immunotherapy for food allergy," Health: An Interdisciplinary Journal for the Social Study of Health, Illness and Medicine, p. 136345932211096, Jul. 2022, doi: https://doi.org/10.1177/13634593221109679.

[9] T. Dominguez, “Food Allergy, Oral Food Challenges, and Oral Immunotherapy,” Physician Assistant Clinics, vol. 8, no. 4, pp. 675–684, Oct. 2023, doi: https://doi.org/10.1016/j.cpha.2023.05.003.

[10] M. Gilbert T. Chua et al., “Real-world safety and effectiveness analysis of low-dose preschool sesame oral immunotherapy,” Journal of Allergy and Clinical Immunology: Global, vol. 3, no. 1, p. 100171, Feb. 2024, Accessed: Nov. 29, 2023. [Online]. Available: https://doaj.org/article/419f5832508a49019ca3476a94bd4b3d

[11] M. D. J. Andrew Bird et al., “Long-term safety and immunologic outcomes of daily oral immunotherapy for peanut allergy,” Journal of Allergy and Clinical Immunology: Global, vol. 2, no. 3, p. 100120, Aug. 2023, Accessed: Nov. 29, 2023. [Online]. Available: https://doaj.org/article/880df47945f042aeb9a2cf79f96c1241

[12] “About the Unified Modeling Language Specification Version 2.5.1,” www.omg.org. https://www.omg.org/spec/UML/

[13] “IEEE Recommended Practice for Software Requirements Specifications,” IEEE Std 830-1998, pp. 1–40, Oct. 1998, doi: https://doi.org/10.1109/IEEESTD.1998.88286.

[14] ISO, “ISO/IEC 25010:2011,” ISO, 2011. https://www.iso.org/standard/35733.html

[15] Apple, “Human Interface Guidelines - Design - Apple Developer,” Apple.com, 2019. https://developer.apple.com/design/human-interface-guidelines/

[16] Apple, “App Store Review Guidelines - Apple Developer,” Apple.com, 2019. https://developer.apple.com/app-store/review/guidelines/

[17] Apple, “Accessibility - Apple Developer,” Apple.com, 2019. https://developer.apple.com/accessibility/

[18] H. Health, “U.S. Department of Health and Human Services,” HHS.gov, 2019. https://www.hhs.gov

[19] Apple, “HealthKit — Apple Developer Documentation,” developer.apple.com. https://developer.apple.com/documentation/healthkit

[20] Apple, “CareKit - Apple Developer,” developer.apple.com. https://developer.apple.com/carekit/

[21] [17]Apple, “Apple Developer Documentation,” developer.apple.com. https://developer.apple.com/documentation/charts

[22] Apple, “SwiftData,” Apple Developer Documentation. https://developer.apple.com/documentation/swiftdata

[23] Kant, Immanuel. ”Groundwork of the Metaphysics of Morals.” Cambridge University Press, 1998.

[24] Mill, John Stuart. ”Utilitarianism.” Edited by George Sher. Hackett Publishing Company, 2001.

[25] Nussbaum, Martha C. ”Creating Capabilities: The Human Development Approach.” Harvard University Press, 2011.

[26] ACM Code 2018 Task Force. “ACM Code of Ethics and Professional Conduct.” ACM: Association for Computing Machinery, 2018, https://www.acm.org/code-of-ethics.

[27] IEEE, ”IEEE Std 830-1998 - IEEE Recommended Practice for Software Requirements Specifications,” 1998.

[28] IEEE, ”IEEE Std 1016-2009 - IEEE Standard for Information Technology—Systems Design—Software Design Descriptions,” 2009.

[29] IEEE, ”IEEE Std 1471-2000 - IEEE Recommended Practice for Architectural Description of Software-Intensive Systems,” 2000.

[30] IEEE, ”IEEE Std 830.1-1993 - IEEE Recommended Practice for Software Requirements Specifications Extensions,” 1993.

[31] ISO/IEC, ”ISO/IEC 9126-1:2001 - Software Engineering - Product Quality - Part 1: Quality Model,” 2001.

[32] ISO/IEC, ”ISO/IEC 25000:2014 - Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) - Guide to SQuaRE,” 2014.

[33] ISO, ”ISO 9241-171:2008 - Ergonomics of human-system interaction - Part 171: Guidance on software accessibility,” 2008.