

**SANTA CLARA UNIVERSITY**  
**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

Date: May 26, 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 26, 2024

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

Madeleine Waldie

Department of Computer Science and Engineering  
Santa Clara University  
May 26, 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

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Background . . . . .	1
1.2	Project Goal . . . . .	2
<b>2</b>	<b>Project Requirements</b>	<b>3</b>
2.1	Functional and Nonfunctional Requirements . . . . .	3
2.1.1	Functional Requirements . . . . .	3
2.1.2	Nonfunctional Requirements . . . . .	4
2.2	Use Cases . . . . .	5
<b>3</b>	<b>Research</b>	<b>17</b>
3.1	Research . . . . .	17
3.1.1	Background . . . . .	17
3.1.2	The Patient's Journey . . . . .	17
3.1.3	Predosing Medications and Precautions . . . . .	18
3.1.4	Adherence to Strict Rules and Lifestyle Adjustments . . . . .	18
3.1.5	Highly Personalized Treatment Plans . . . . .	19
3.2	Consultation with Immunologists . . . . .	19
<b>4</b>	<b>Design Overview</b>	<b>21</b>
4.1	General Design . . . . .	21
4.2	Onboarding View . . . . .	22
4.3	Today Tab . . . . .	22
4.4	Dose and Symptom Pop-Ups . . . . .	23
4.5	Insights Tab . . . . .	24
4.6	Dosing Tab . . . . .	24
4.7	Education Tab . . . . .	25
<b>5</b>	<b>Software Design</b>	<b>26</b>
5.1	General Architecture . . . . .	26
5.2	APIs . . . . .	27
5.3	Class diagrams . . . . .	27
<b>6</b>	<b>Constraints and Standards</b>	<b>36</b>
6.1	Constraints . . . . .	36
6.2	Standards . . . . .	37
<b>7</b>	<b>Schedule</b>	<b>40</b>
<b>8</b>	<b>Risk Analysis</b>	<b>41</b>

<b>9 The Final Product</b>	<b>43</b>
9.1 Differences Between Prototype and Final Product . . . . .	43
9.1.1 UI Design Choices . . . . .	43
9.1.2 Additional Resources . . . . .	45
9.1.3 Profile . . . . .	48
9.1.4 Dosing Tab Changes . . . . .	52
9.2 User Guide . . . . .	55
<b>10 Testing</b>	<b>70</b>
10.1 Visual Testing . . . . .	70
10.2 XCTest and XCUITests . . . . .	73
10.3 User Testing . . . . .	74
10.3.1 User Study . . . . .	74
10.3.2 Conclusion . . . . .	75
<b>11 Societal Issues</b>	<b>76</b>
11.1 Ethical . . . . .	76
11.2 Social . . . . .	76
11.3 Political . . . . .	77
11.4 Economic . . . . .	77
11.5 Health and Safety . . . . .	77
11.6 Manufacturability . . . . .	77
11.7 Sustainability . . . . .	77
11.8 Environmental Impact . . . . .	78
11.9 Usability . . . . .	78
11.10 Lifelong learning . . . . .	78
11.11 Compassion . . . . .	78
<b>12 An Ethical Analysis</b>	<b>79</b>
12.1 Ethical Justification for the Project . . . . .	79
12.2 Identification of Significant Ethical Issues . . . . .	80
12.2.1 Privacy and Data Security . . . . .	80
12.2.2 Informed Consent . . . . .	81
12.2.3 Equitable Access . . . . .	82
12.2.4 Accuracy and Reliability of Information . . . . .	82
12.3 Ethical Design Rationale . . . . .	82
12.3.1 Integration with Apple Health and Other Apple Frameworks . . . . .	83
12.3.2 Use of Emojis and Images for Enhanced Understanding . . . . .	83
12.3.3 Inclusion of Tips to Guide Users . . . . .	84
12.3.4 Creation of a User Guide . . . . .	84
12.3.5 Making the App Free and Available on the App Store . . . . .	85
12.3.6 Support for Different Font Sizes and Accessibility Features . . . . .	86
<b>13 Conclusion</b>	<b>87</b>
<b>A Code Files</b>	<b>89</b>
A.1 OITApp.swift . . . . .	89
A.2 TabbedApplicationView.swift . . . . .	90
A.3 Extensions.swift . . . . .	92
A.4 AllergenWithDoses.swift . . . . .	93
A.5 AntihistamineDose.swift . . . . .	93
A.6 AntihistamineDoseTransformer.swift . . . . .	93
A.7 Dose.swift . . . . .	94
A.8 DoseTransformer.swift . . . . .	94
A.9 DoseViewModel.swift . . . . .	95

A.10	ProfileData	96
A.11	ProfileImageModel.swift	97
A.12	ProfileImageViewModel.swift	99
A.13	ProfileViewModel.swift	101
A.14	SelectedAllergensTransformer	102
A.15	SymptomDataManager.swift	102
A.16	AddDoseView.swift	104
A.17	DoseRowView.swift	105
A.18	DosingView.swift	106
A.19	DosingViewTip.swift	109
A.20	ArticleRichLink.swift	109
A.21	ArticleView.swift	110
A.22	DoctorPhoneTip.swift	111
A.23	EducationView.swift	111
A.24	EducationViewContent.swift	116
A.25	EmergencyServicesTip.swift	118
A.26	ImportantDocumentsTip.swift	118
A.27	MultiImagePicker.swift	118
A.28	ResourcesTip.swift	119
A.29	HealthKitManager.swift	120
A.30	HealthKitViewModel.swift	123
A.31	AboutYourDoseView.swift	124
A.32	CalendarDayView.swift	126
A.33	DosingPopUp.swift	128
A.34	HomeView.swift	132
A.35	ProfileImage.swift	138
A.36	SectionHeaderView.swift	139
A.37	SettingsView.swift	139
A.38	SymptomsPopUp.swift	143
A.39	DosesForMonthView.swift	145
A.40	InsightsTip.swift	147
A.41	InsightsView.swift	147
A.42	LastReactionInsight.swift	150
A.43	Legend.swift	150
A.44	LegendItem.swift	151
A.45	SymptomsOverTimeChartView.swift	151
A.46	SymptomsThisWeekChartView.swift	152
A.47	GetSetUpView.swift	153
A.48	GetStartedView.swift	154

# List of Tables

2.1	Use Case 01: Initial App Onboarding . . . . .	6
2.2	Use Case 02: Sharing Data with Apple Health . . . . .	7
2.3	Use Case 03: Adding Allergen to Profile . . . . .	8
2.4	Use Case 04: Log a Dose Taken . . . . .	10
2.5	Use Case 05: Log Symptoms for Dose Taken . . . . .	11
2.6	Use Case 06: Add a Dosage for an Allergen . . . . .	13
2.7	Use Case 07: View Trends . . . . .	14
2.8	Use Case 08: Open Article . . . . .	14
2.9	Use Case 09: Navigating the Calendar View . . . . .	15
5.1	Class Diagram: OITApplication . . . . .	28
5.2	Class Diagram: TabbedApplicationView . . . . .	28
5.3	Class Diagram: GetStartedView . . . . .	28
5.4	Class Diagram: GetSetUpView . . . . .	28
5.5	Class Diagram: ProfileInformation . . . . .	29
5.6	Class Diagram: Allergen . . . . .	29
5.7	Class Diagram: Symptom . . . . .	29
5.8	Class Diagram: Medication . . . . .	29
5.9	Class Diagram: SymptomsPopUp . . . . .	30
5.10	Class Diagram: HomeView . . . . .	30
5.11	Class Diagram: WeeklyScrollView . . . . .	30
5.12	Class Diagram: CalendarDayView . . . . .	31
5.13	Class Diagram: AboutYourDoseView . . . . .	31
5.14	Class Diagram: Dose . . . . .	31
5.15	Class Diagram: DosingPopUp . . . . .	32
5.16	Class Diagram: SettingsView . . . . .	32
5.17	Class Diagram: HealthKitManager . . . . .	32
5.18	Class Diagram: HealthKitViewModel . . . . .	33
5.19	Class Diagram: InsightsView . . . . .	33
5.20	Class Diagram: ChartView . . . . .	33
5.21	Class Diagram: DosingView . . . . .	33
5.22	Class Diagram: AddDoseView . . . . .	34
5.23	Class Diagram: Dosage . . . . .	34
5.24	Class Diagram: EducationView . . . . .	34
5.25	Class Diagram: ArticleRichLink . . . . .	34
5.26	Class Diagram: ArticleView . . . . .	35
8.1	Risk Analysis Table . . . . .	42

# List of Figures

2.1 Use Case 02: Visual Diagram . . . . .	8
2.2 Use Case 03: Visual Diagram . . . . .	9
2.3 Use Case 04: Visual Diagram . . . . .	11
2.4 Use Case 05: Visual Diagram . . . . .	12
2.5 Use Case 09: Visual Diagram 1 . . . . .	15
2.6 Use Case 09: Visual Diagram 2 . . . . .	16
4.1 Color Scheme . . . . .	21
4.2 Onboarding Screens . . . . .	22
4.3 Today Tab Screens . . . . .	23
4.4 Dose and Symptoms Pop-Ups . . . . .	23
4.5 Insights Tab . . . . .	24
4.6 Dosing Tab Screens . . . . .	24
4.7 Education Tab Screens . . . . .	25
5.1 Overall File Architecture . . . . .	26
7.1 Project Gantt Chart . . . . .	40
9.1 Unique Emojis for Symptoms . . . . .	44
9.2 Unique Emojis for Nuts . . . . .	44
9.3 Doses Tab Tip . . . . .	45
9.4 Important Documents Section . . . . .	46
9.5 Important Documents Section With Documents Added . . . . .	46
9.6 Tapping to View Important Document . . . . .	47
9.7 Doctor / Clinic Contact Info Section . . . . .	47
9.8 Add Profile Photo Section . . . . .	48
9.9 Profile Picture in Top Corner of App . . . . .	49
9.10 User Tapping Other Allergen from Common Allergens List . . . . .	50
9.11 User Adding Other Allergen Name . . . . .	50
9.12 User Adding Other Allergen Emoji . . . . .	51
9.13 Screen to Enter Passcode . . . . .	52
9.14 Uncollapsed Sections of Dosing Tab . . . . .	53
9.15 Collapsed Sections of Dosing Tab . . . . .	53
9.16 Dosing Tab Filters . . . . .	54
9.17 Dose Edit Actions . . . . .	54
9.18 Dose Edit Screen . . . . .	55
9.19 Dose Delete Action . . . . .	55
9.20 User Guide Page 1 . . . . .	56
9.21 User Guide Page 2 . . . . .	57
9.22 User Guide Page 3 . . . . .	58
9.23 User Guide Page 4 . . . . .	59
9.24 User Guide Page 5 . . . . .	60
9.25 User Guide Page 6 . . . . .	61

9.26 User Guide Page 7 . . . . .	62
9.27 User Guide Page 8 . . . . .	63
9.28 User Guide Page 9 . . . . .	64
9.29 User Guide Page 10 . . . . .	65
9.30 User Guide Page 11 . . . . .	66
9.31 User Guide Page 12 . . . . .	67
9.32 User Guide Page 13 . . . . .	68
9.33 User Guide Page 14 . . . . .	69
10.1 SwiftUI Canvas . . . . .	71
10.2 Swift UI Canvas Color Scheme Variants . . . . .	71
10.3 Swift UI Canvas Orientation Variants . . . . .	72
10.4 Swift UI Canvas Dynamic Type Variants . . . . .	72
10.5 Swift UI Canvas Device Variant . . . . .	73

# 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.

<b>UC-01</b>	<b>Initial App Onboarding</b>	
<b>Dependencies</b>	<ul style="list-style-type: none"> <li>User has never opened the app before</li> </ul>	
<b>Description</b>	The system will behave in the following use case when the user initially opens the app.	
<b>Precondition</b>	The user has downloaded the app from the app store, and goes to use it.	
<b>Ordinary Sequence</b>	<b>Step</b>	<b>Action</b>
	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: <ul style="list-style-type: none"> <li>Name</li> <li>Birthdate</li> <li>Allergens</li> <li>Preferences               <ul style="list-style-type: none"> <li>Share Data with Apple Health</li> <li>Protect Application with FaceID</li> </ul> </li> </ul>
	4	User fills out the necessary information: <ul style="list-style-type: none"> <li>Name</li> <li>Birthdate</li> <li>Allergens</li> <li>Preferences               <ul style="list-style-type: none"> <li>Share Data with Apple Health</li> <li>Protect Application with FaceID</li> </ul> </li> </ul>
	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.
<b>Postcondition</b>	The user will be able to use the application.	
<b>Exceptions</b>	<b>Step</b>	<b>Action</b>
	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.
<b>Comments</b>	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

<b>UC-02</b>	<b>Sharing Data with Apple Health</b>	
<b>Dependencies</b>	<ul style="list-style-type: none"> <li>User hasn't enabled sharing data with Apple health yet</li> </ul>	
<b>Description</b>	The system will behave in the following use case when the user enables sharing data with Apple health	
<b>Precondition</b>	The user is going through the setup page.	
<b>Ordinary Sequence</b>	<b>Step</b>	<b>Action</b>
	1	Application displays a screen with basic information for the user to fill out: <ul style="list-style-type: none"> <li>Name</li> <li>Birthdate</li> <li>Allergens</li> <li>Preferences               <ul style="list-style-type: none"> <li>Share Data with Apple Health</li> <li>Protect Application with FaceID</li> </ul> </li> </ul>
	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"
<b>Postcondition</b>	The user will have all of the selected data shared with Apple Health.	
<b>Exceptions</b>	<b>Step</b>	<b>Action</b>
	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.
<b>Comments</b>	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

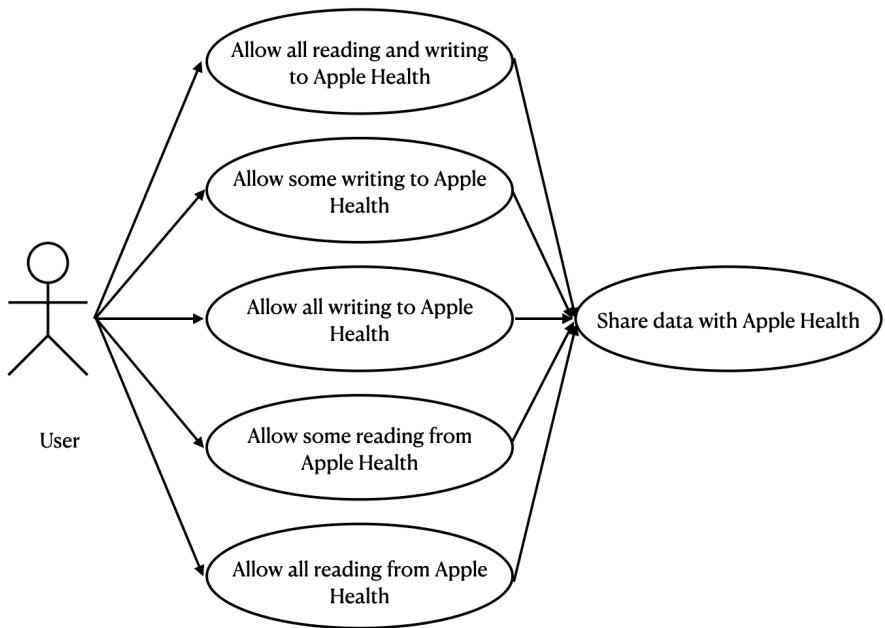


Figure 2.1: Use Case 02: Visual Diagram

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: <ul style="list-style-type: none"><li>• Name</li><li>• Birthdate</li><li>• Allergens</li><li>• Preferences<ul style="list-style-type: none"><li>○ Share Data with Apple Health</li><li>○ Protect Application with FaceID</li></ul></li></ul>
	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

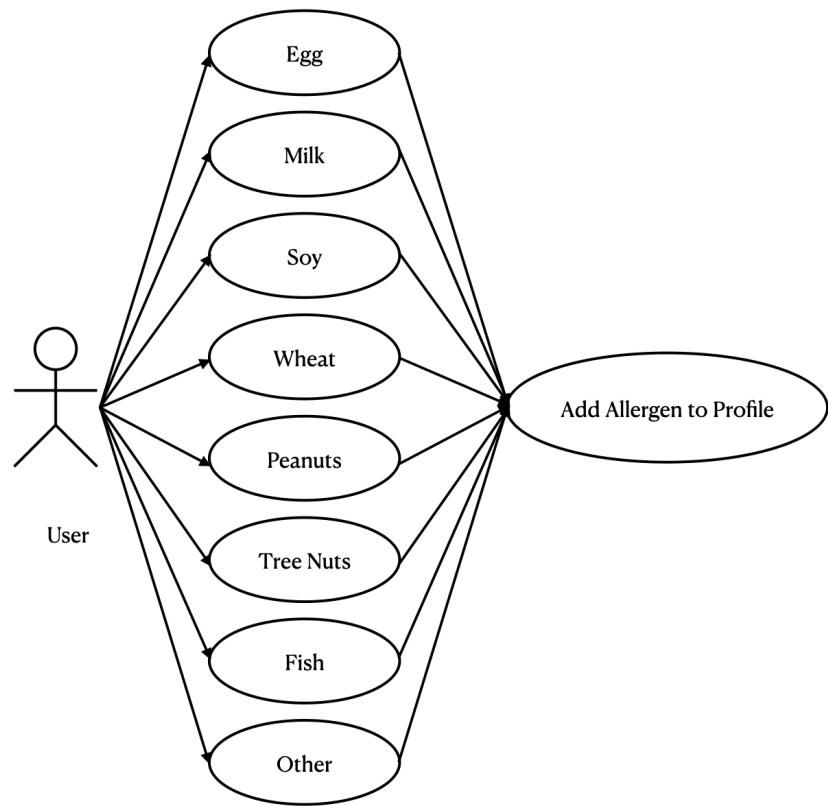


Figure 2.2: Use Case 03: Visual Diagram

<b>UC-04</b>	<b>Log a Dose Taken</b>	
<b>Dependencies</b>	<ul style="list-style-type: none"> <li>• Allergens patient is doing OIT for</li> <li>• List of possible doses patient could take for each allergen</li> <li>• Date of dose</li> </ul>	
<b>Description</b>	The app will behave in the following use case when the user goes to log a dose.	
<b>Precondition</b>	The patient has taken a dose and wants to log it, so they open their app, which opens to the “Home” tab.	
<b>Ordinary Sequence</b>	<b>Step</b>	<b>Action</b>
	1	User clicks the “Log Dose” button
	2	Application displays a sheet, with fields for the user to answer the following information: <ul style="list-style-type: none"> <li>• Allergens in the dose</li> <li>• Time of dose</li> <li>• Any medications taken before the dose</li> <li>• Dosage of each allergen</li> <li>• Notes</li> </ul>
	3	User inputs information for each of the following fields: <ul style="list-style-type: none"> <li>• Time of dose</li> <li>• Any medications taken before the dose</li> <li>• Dosage of each allergen</li> <li>• Notes</li> </ul>
	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
<b>Postcondition</b>	The user can see their dose was logged, and can quickly view the dose information for the day.	
<b>Exceptions</b>	<b>Step</b>	<b>Action</b>
	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.
<b>Comments</b>	Only one dose can be logged per day.	

Table 2.4: Use Case 04: Log a Dose Taken

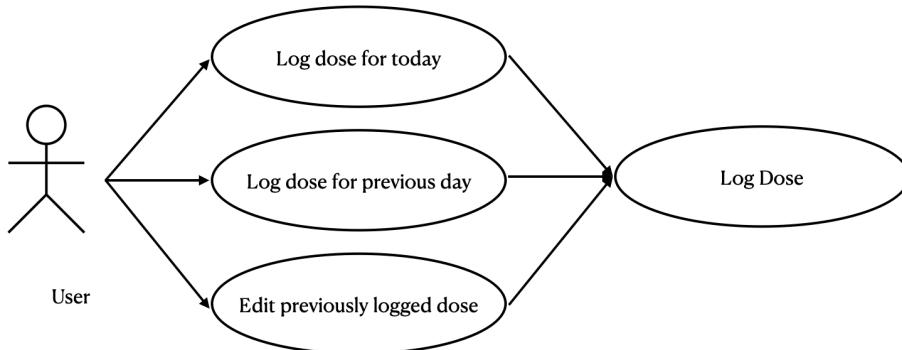


Figure 2.3: Use Case 04: Visual Diagram

UC-05	Log Symptoms for Dose Taken	
<b>Dependencies</b>	<ul style="list-style-type: none"> <li>• Date of dose</li> <li>• List of possible symptoms</li> </ul>	
<b>Description</b>	The app will behave in the following use case when the user goes to log symptoms they have experienced.	
<b>Precondition</b>	The patient wants to log symptoms they're experiencing after taking a dose, so they open the app.	
<b>Ordinary Sequence</b>	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
<b>Postcondition</b>	The user can see their symptoms were logged, and can quickly view their symptom information for the day.	
<b>Exceptions</b>	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.
<b>Comments</b>	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

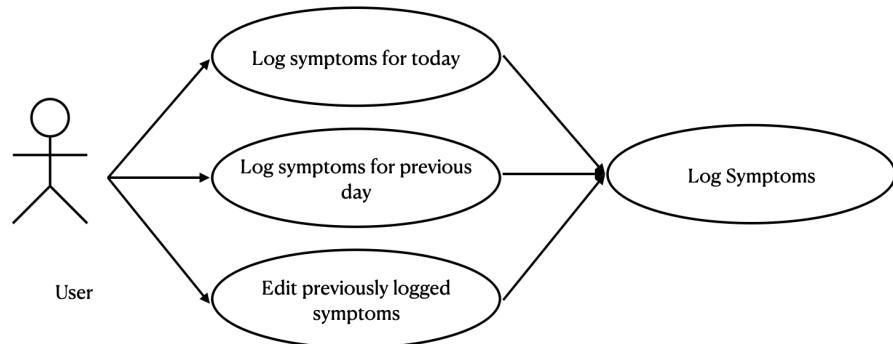


Figure 2.4: Use Case 05: Visual Diagram

<b>UC-06</b>	<b>Add a Dosage for an Allergen</b>	
<b>Dependencies</b>	<ul style="list-style-type: none"> <li>User has one or more allergens listed in their profile</li> </ul>	
<b>Description</b>	The system will behave in the following use case when the user wants to create a new dosage for an allergen.	
<b>Precondition</b>	The patient has received a new dosage for their OIT from their doctor, and wants to add it to the app.	
<b>Ordinary Sequence</b>	<b>Step</b>	<b>Action</b>
	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: <ul style="list-style-type: none"> <li>Allergen</li> <li>Name of Dose</li> <li>Dosage</li> <li>Notes</li> </ul>
	5	User enters the necessary information: <ul style="list-style-type: none"> <li>Allergen</li> <li>Name of Dose</li> <li>Dosage</li> <li>Notes</li> </ul>
	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.
<b>Postcondition</b>	The user can see their dose was added to the list of possible doses.	
<b>Exceptions</b>	<b>Step</b>	<b>Action</b>
	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.
<b>Comments</b>	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

<b>UC-07</b>	<b>View Trends</b>	
<b>Dependencies</b>	<ul style="list-style-type: none"> <li>• Doses have been logged</li> <li>• User allows connection with the Apple Health app</li> </ul>	
<b>Description</b>	The system will behave in the following use case when the user wants to view trends for their symptoms.	
<b>Precondition</b>	The user has opened the application.	
<b>Ordinary Sequence</b>	<b>Step</b>	<b>Action</b>
	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
<b>Postcondition</b>	The user can see the trends of their symptoms.	
<b>Exceptions</b>	<b>Step</b>	<b>Action</b>
	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.
<b>Comments</b>	The type of insights information displayed will depend on the information logged.	

Table 2.7: Use Case 07: View Trends

<b>UC-08</b>	<b>Open Article</b>	
<b>Description</b>	The system will behave in the following use case when the user wants to view one of the resource articles.	
<b>Precondition</b>	The user wants to learn more about the signs and symptoms of anaphylaxis.	
<b>Ordinary Sequence</b>	<b>Step</b>	<b>Action</b>
	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
<b>Postcondition</b>	User reads the article, learning new information	
<b>Comments</b>	Tapping on different articles will show different information.	

Table 2.8: Use Case 08: Open Article

<b>UC-09</b>	<b>Navigating the Calendar View</b>	
<b>Description</b>	The system will behave in the following use case when the user wants to see dose / symptom information for a day in the past.	
<b>Precondition</b>	The user has logged some doses over the past week, and wants to look at them again.	
<b>Ordinary Sequence</b>	<b>Step</b>	<b>Action</b>
	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
<b>Postcondition</b>	The user can see the dose and symptom data for the date they're interested in.	
<b>Exceptions</b>	<b>Step</b>	<b>Action</b>
	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.
<b>Comments</b>	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

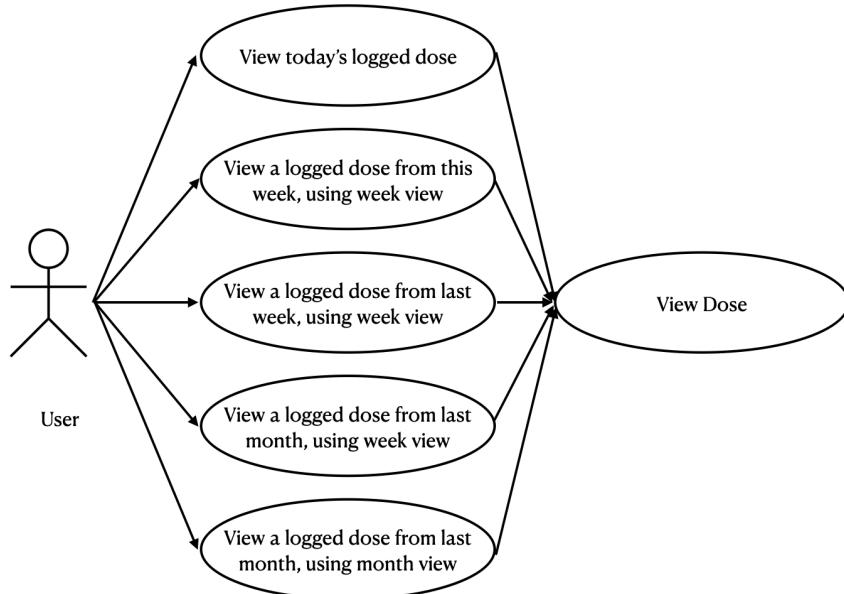


Figure 2.5: Use Case 09: Visual Diagram 1

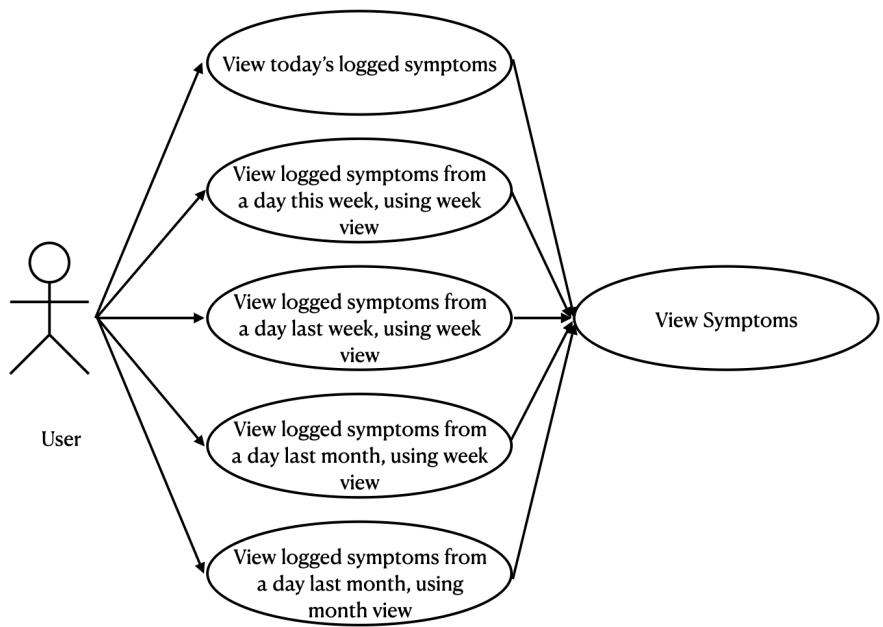


Figure 2.6: Use Case 09: Visual Diagram 2

# **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, 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 View 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.

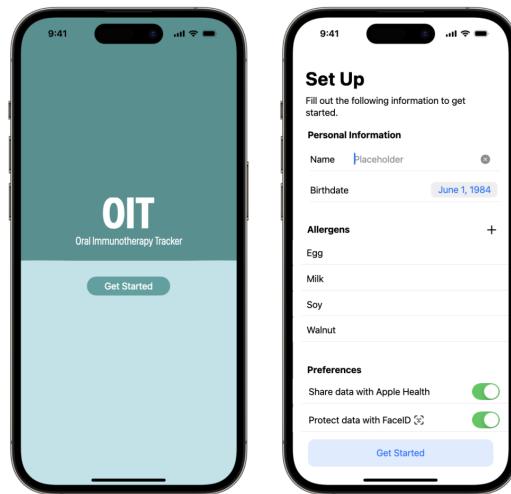


Figure 4.2: Onboarding Screens

## 4.3 Today Tab

The Today Tab serves 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.

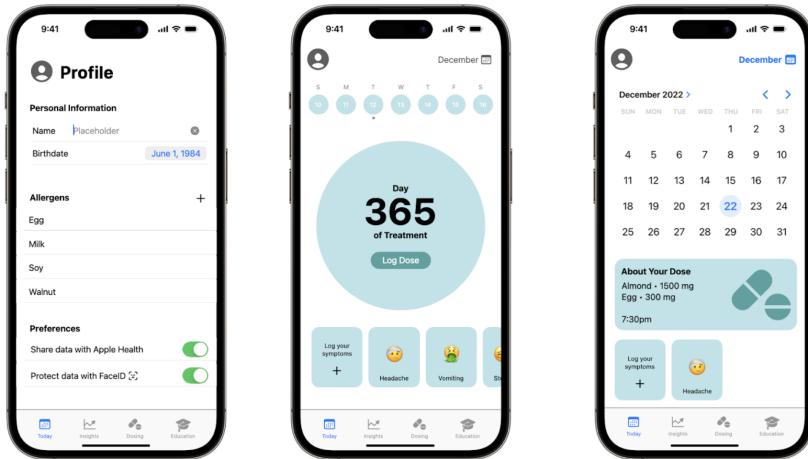


Figure 4.3: Today Tab Screens

## 4.4 Dose and Symptom Pop-Ups

The Dose and Symptom Pop-Ups 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.

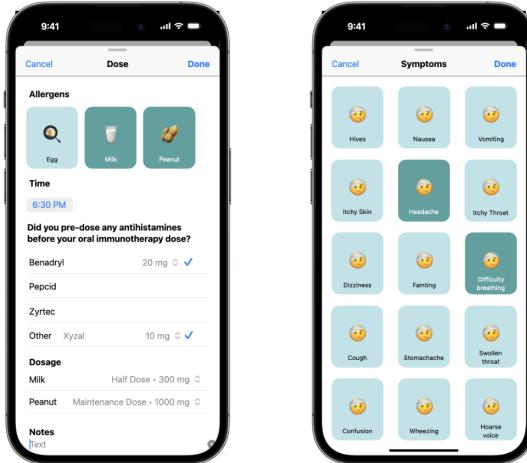


Figure 4.4: Dose and Symptoms Pop-Ups

## 4.5 Insights Tab

The Insights Tab 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.

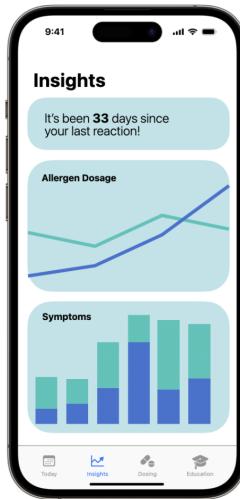


Figure 4.5: Insights Tab

## 4.6 Dosing Tab

The Dosing Tab 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.

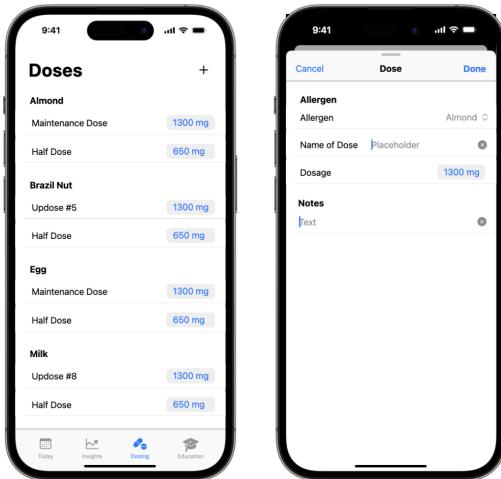


Figure 4.6: Dosing Tab Screens

## 4.7 Education Tab

The Education Tab 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.

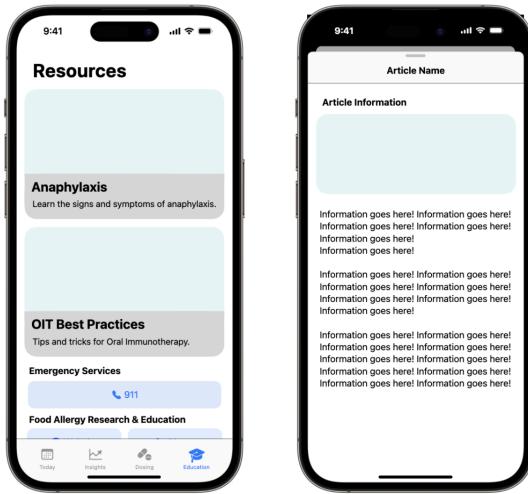


Figure 4.7: Education Tab Screens

# Chapter 5

## Software Design

### 5.1 General Architecture

My application will be written in Swift and SwiftUI. The file architecture diagram below 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.

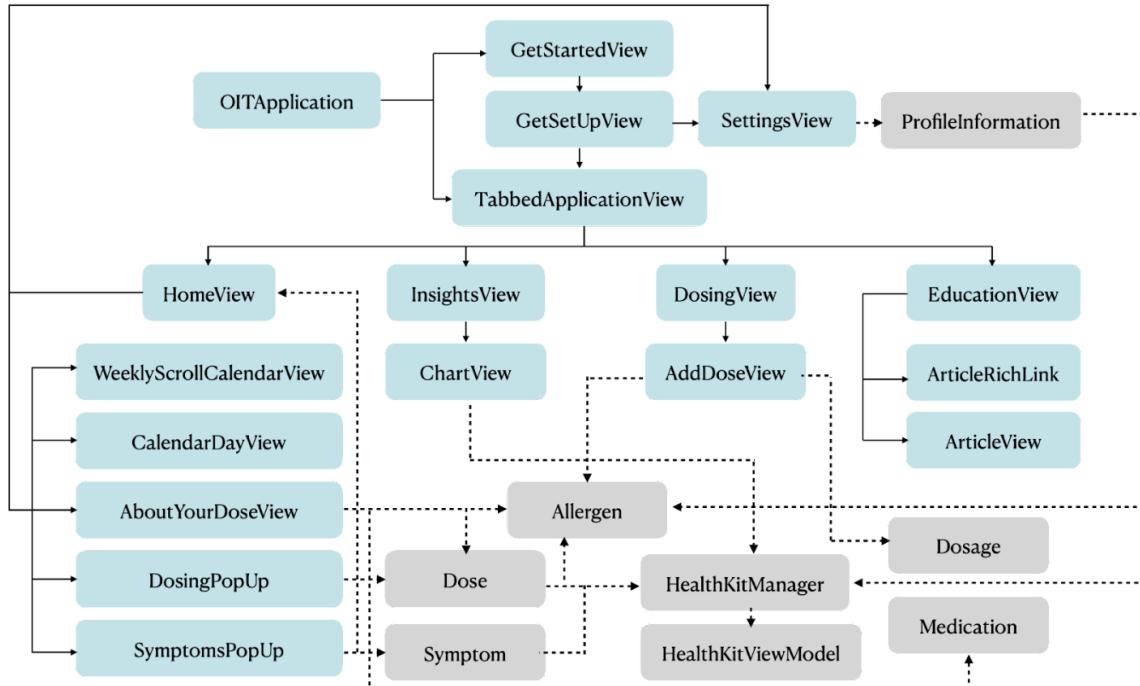


Figure 5.1: Overall File Architecture

## 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
- weeklyScrollView: WeeklyScrollView
- logDoseButton: Button
- logSymptomsButton: Button
- aboutYourDoseView: AboutYourDoseView
- dose: Dose
- symptoms: Symptoms

Table 5.11: Class Diagram: WeeklyScrollView

WeeklyScrollView
- 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: [CategoryType]
- setUpHealthRequest()

Table 5.18: Class Diagram: HealthKitViewModel

<b>HealthKitViewModel</b>
+ healthStore: HKHealthStore
+ healthKitManager: HKManager
+ isAuthorized: Bool
- - - - -
+ healthRequest()
+ changeAuthorizationStatus()

Table 5.19: Class Diagram: InsightsView

<b>InsightsView</b>
- title: Text
- charts: [ChartView]

Table 5.20: Class Diagram: ChartView

<b>ChartView</b>
- chartType: Chart
- chartData: [AnyObject]

Table 5.21: Class Diagram: DosingView

<b>DosingView</b>
- title: Text
- addNewDoseButton: Button
+ allergensAndDoses: [Allergen, [Dosage]]
- tableView: UITableView

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

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

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.

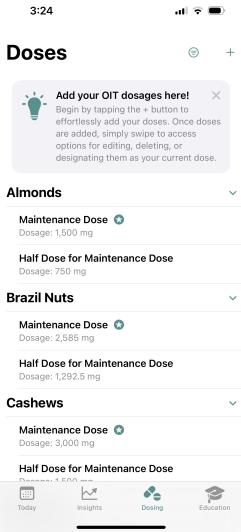


Figure 9.3: Doses Tab Tip

### 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.

3:25

## Resources

Enter Doctor Phone Number

📞 Doctor Phone Number

Food Allergy Research & Education

🌐 Website

📞 Phone

Important Documents



Keep Your Information in  
One Place!

Add pictures of important documents,  
such as your Emergency Action Plan,  
below, so you can easily find them later!

Select Images



Figure 9.4: Important Documents Section

3:47

## Resources

Food Allergy Research & Education

🌐 Website

📞 Phone

Important Documents



Keep Your Information in  
One Place!

Add pictures of important documents,  
such as your Emergency Action Plan,  
below, so you can easily find them later!

Select Images



Figure 9.5: Important Documents Section With Documents Added



Figure 9.6: Tapping to View Important Document

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

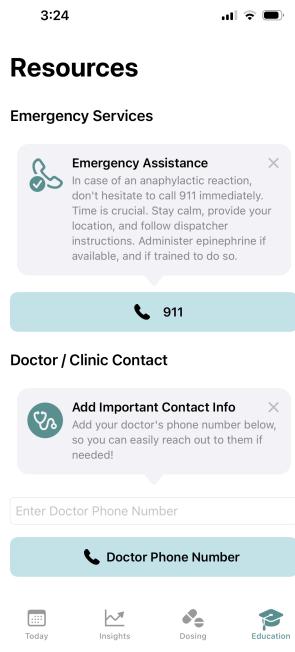


Figure 9.7: Doctor / Clinic Contact Info Section

### 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. I thought this added a needed element of personalization to the app and tied together the personal information section of the profile nicely.

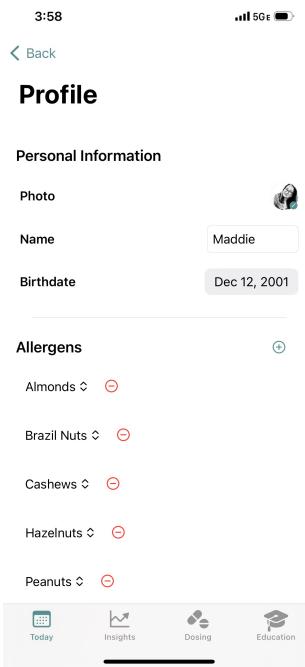


Figure 9.8: Add Profile Photo Section

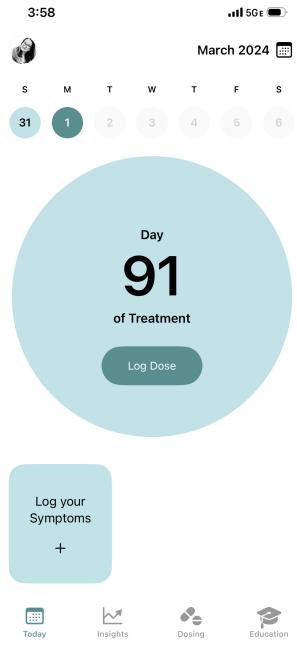


Figure 9.9: Profile Picture in Top Corner of App

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.

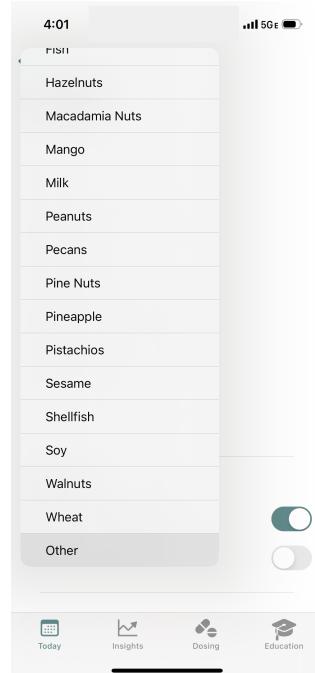


Figure 9.10: User Tapping Other Allergen from Common Allergens List

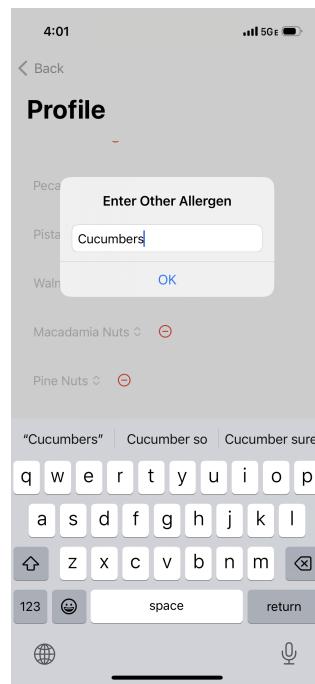


Figure 9.11: User Adding Other Allergen Name

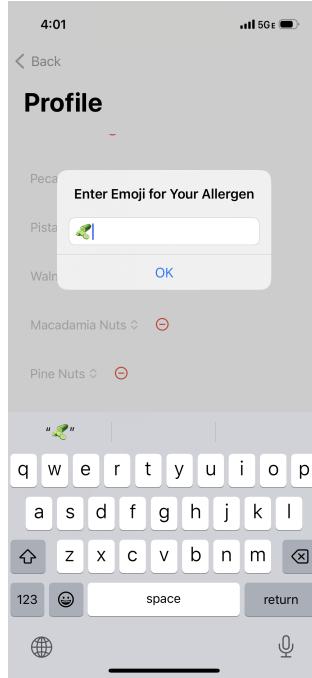


Figure 9.12: User Adding Other Allergen Emoji

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. And, depending on the setup, the screen to authenticate will ask for the user's passcode, TouchID, or FaceID.

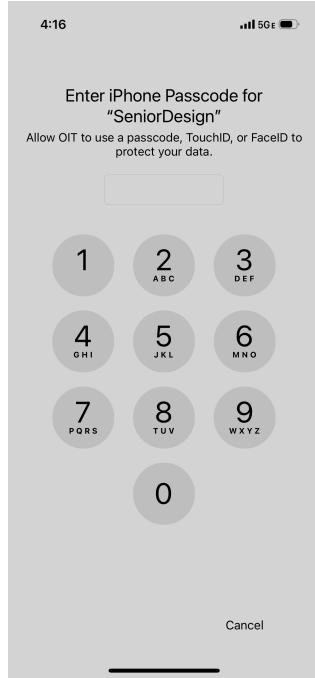


Figure 9.13: Screen to Enter Passcode

#### 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.

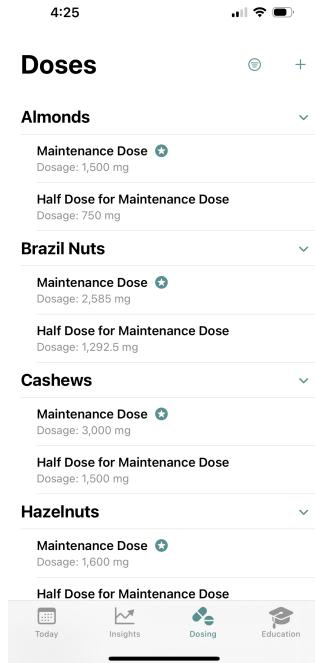


Figure 9.14: Uncollapsed Sections of Dosing Tab

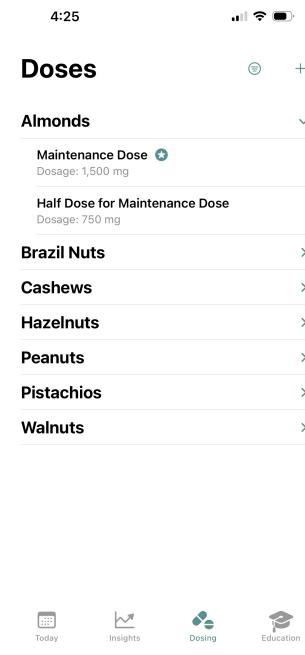


Figure 9.15: Collapsed Sections of Dosing Tab

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.

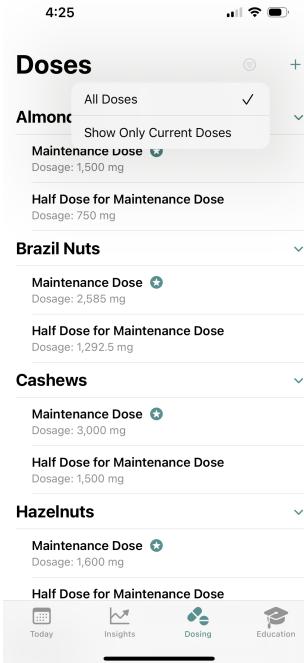


Figure 9.16: Dosing Tab Filters

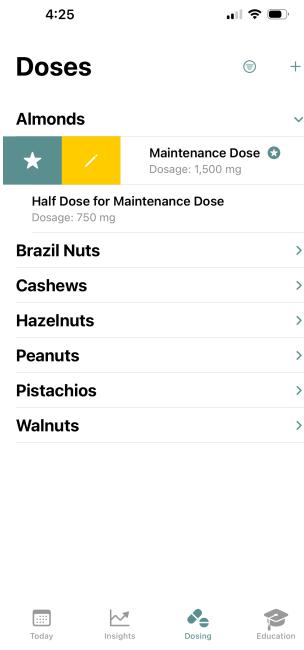


Figure 9.17: Dose Edit Actions

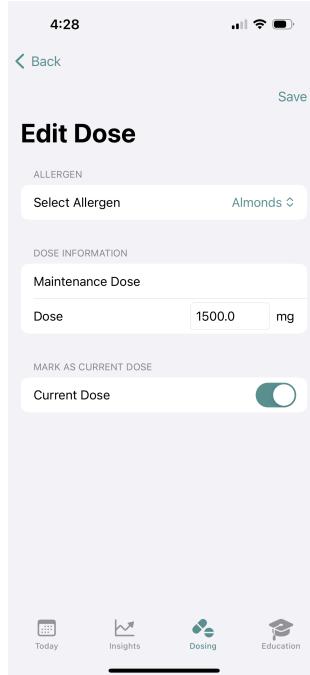


Figure 9.18: Dose Edit Screen

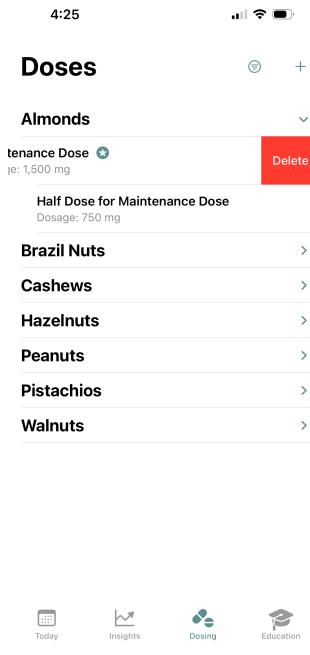


Figure 9.19: Dose Delete Action

## 9.2 User Guide

After finishing developing my application, I created a comprehensive user guide for it, as you can see below.

# Oral Immunotherapy Tracker User Guide

Everything you need to know to use the application!

## Table of Contents

Set Up and Get Started .....	3
Downloading the App .....	3
Setting Up Your Profile .....	3
Add Personal Information .....	3
Share Data with Apple Health .....	4
Protect Data with FaceID, TouchID, or Passcode .....	4
Basics of the App .....	5
Navigating Between Tabs .....	5
Dismissing Tips .....	5
Dark and Light Mode .....	6
Today Tab .....	7
Weekly Scroll View versus Monthly View .....	7
Logging a Dose .....	7
Logging Symptoms .....	8
Profile .....	8
Editing Your Information .....	8
Insights Tab .....	9
Dosing Tab .....	10
Adding a Dose .....	10
Editing a Dose .....	10
Deleting a Dose .....	11
Marking a Dose as Current .....	11

# **Oral Immunotherapy Tracker User Guide**

Everything you need to know to use the application!

## **Table of Contents (Continued)**

Filtering Doses .....	11
Collapsing Sections .....	11
Education Tab .....	12
Viewing Articles .....	12
Resources .....	12
Important Documents .....	13
Copyright .....	14

# Set Up and Get Started

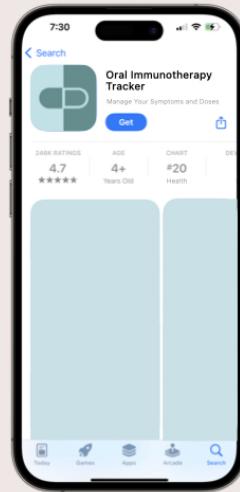
This section explains everything you need to know to get set up using the Oral Immunotherapy Tracker application!

## 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

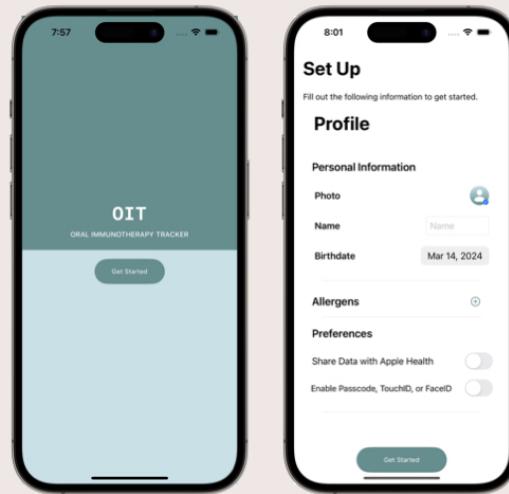
When you first open the app, you'll see a screen with a "Get Started" button. Tapping this button will bring you to the Set Up screen.

### 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.



# Set Up and Get Started

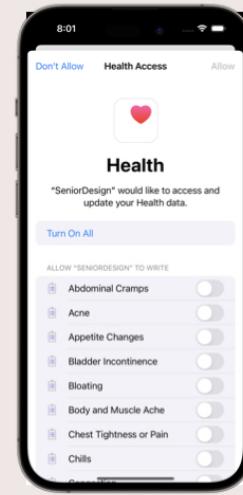
## Share Data with Apple Health

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.



# Basics of the App

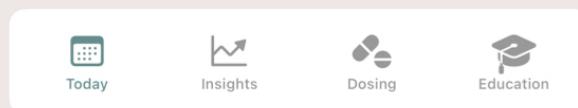
This section goes over some of the basics of using the app.

## Navigating Between Tabs

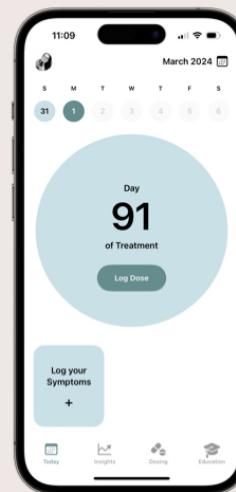
You can use the bottom tab bar to navigate between the Today, Insights, Dosing, and Education tabs.

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

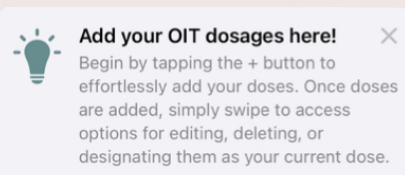


## Dismissing Tips

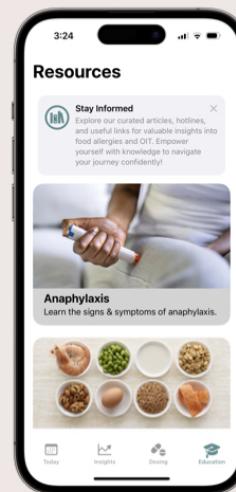
As you navigate the application, you will find tips placed to help you learn about the app and guide you through using it.

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

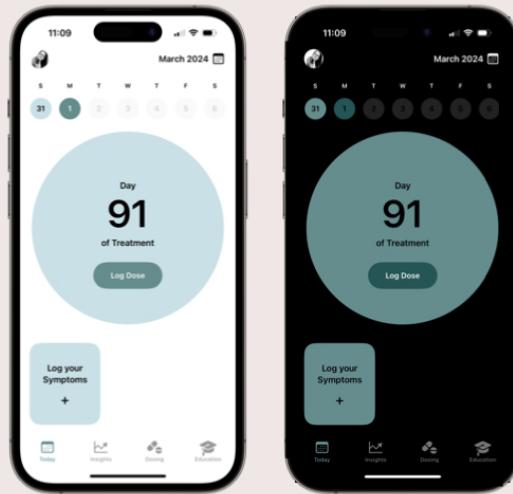


# Basics of the App

## 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.



# Today Tab

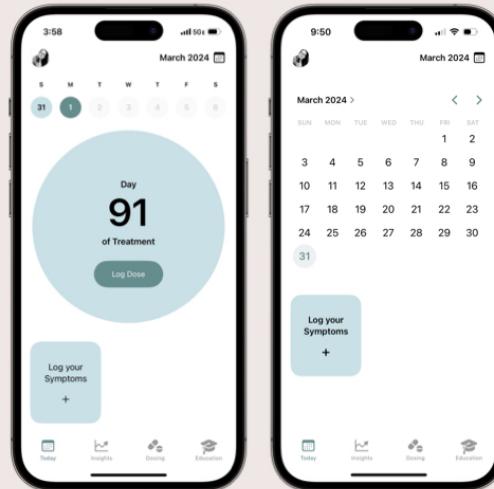
The Today tab serves as a central hub for the most crucial information of your day. This section provides an overview of the key features of the Today Tab within the application.

## 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.

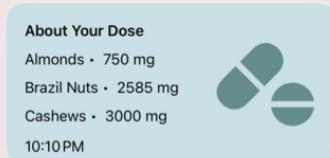


## Logging a Dose

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

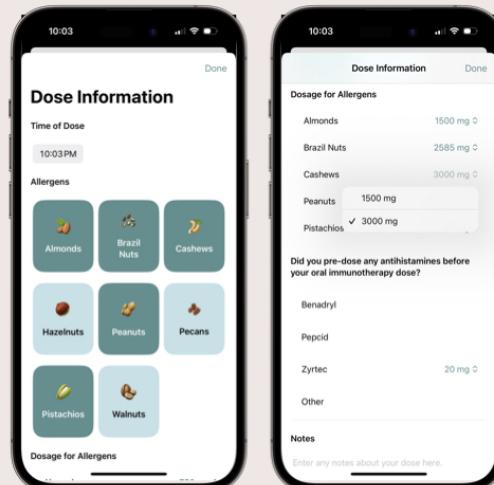


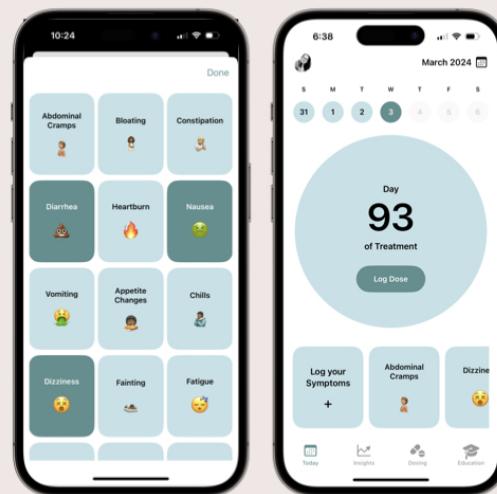
Figure 9.26: User Guide Page 7

# 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.



Generic Profile Icon

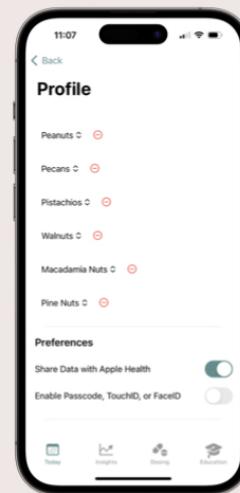
If you have set up a profile picture, it will be displayed as the icon. Otherwise, you will see a generic profile icon.

## 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.

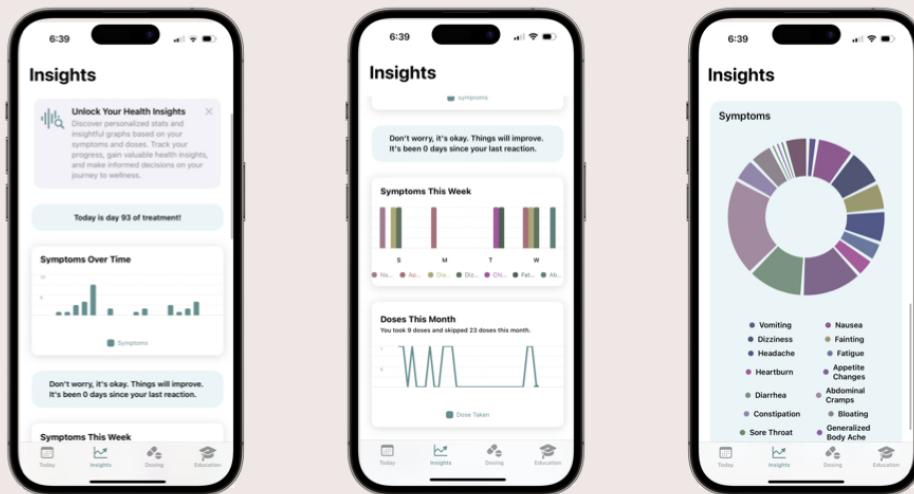
Moreover, within the "Allergens" section, you can add any allergens you're allergic to by tapping the "+" button and selecting your allergen from the picker, or remove allergens using the "-" button.



# 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.



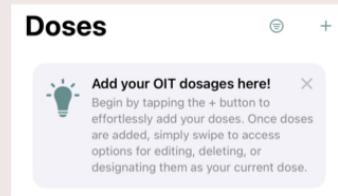
If you don't see many charts or insights, don't worry! The app might need to collect a little bit more data from your doses and symptoms before it can display trends!

# 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!

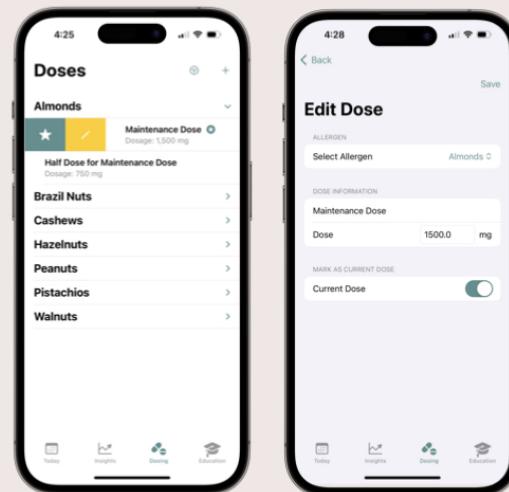
## Editing a Dose

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



# 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

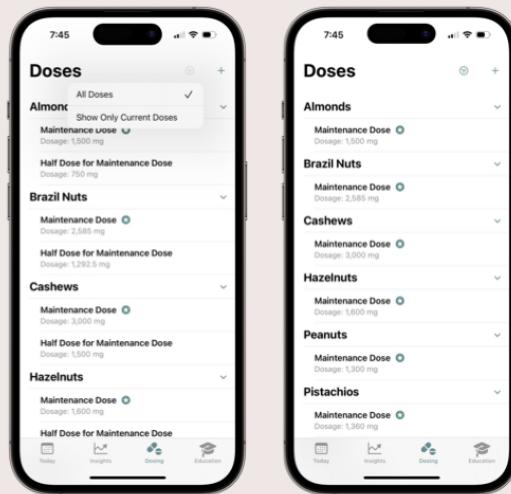
To assist you in keeping track of your doses, you can designate one dose of each allergen as your current dose. To do this, swipe right on the dose and tap the teal star button. This action will add a teal star symbol next to the dose, indicating that it's your current one.



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.

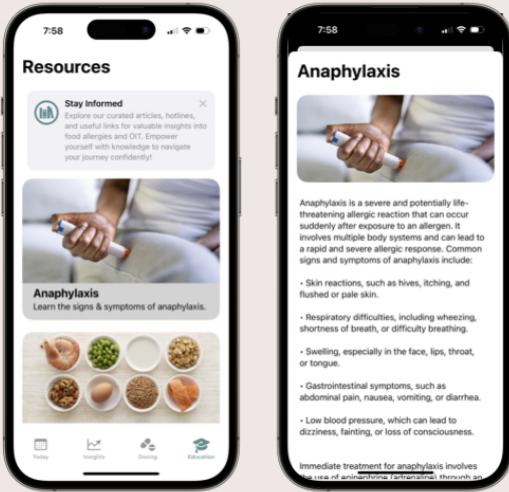


Collapsed Allergen Sections

# Education Tab

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

## Viewing Articles



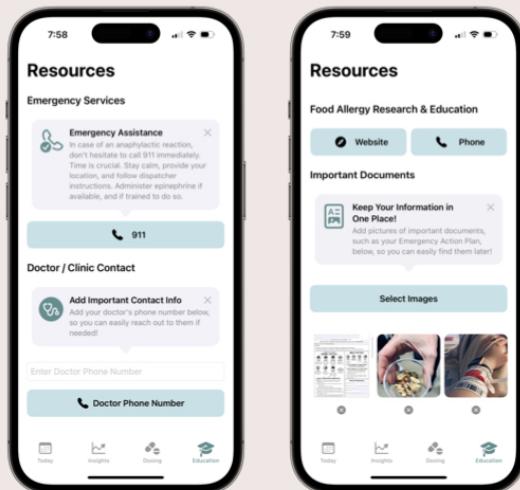
The app provides numerous articles to enhance your knowledge about allergies, anaphylaxis, and OIT.

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.



# Education Tab

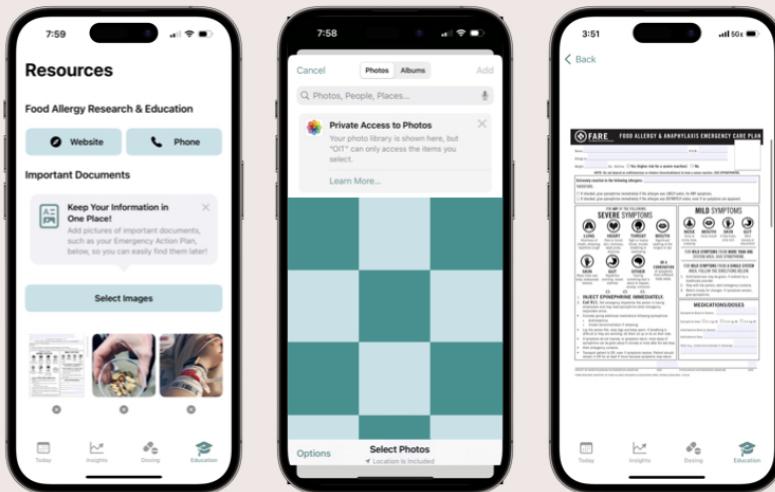
## Important Documents

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!



# **Copyright**

Copyright © Madeleine Waldie 2023 - 2024. All rights reserved.

# **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.

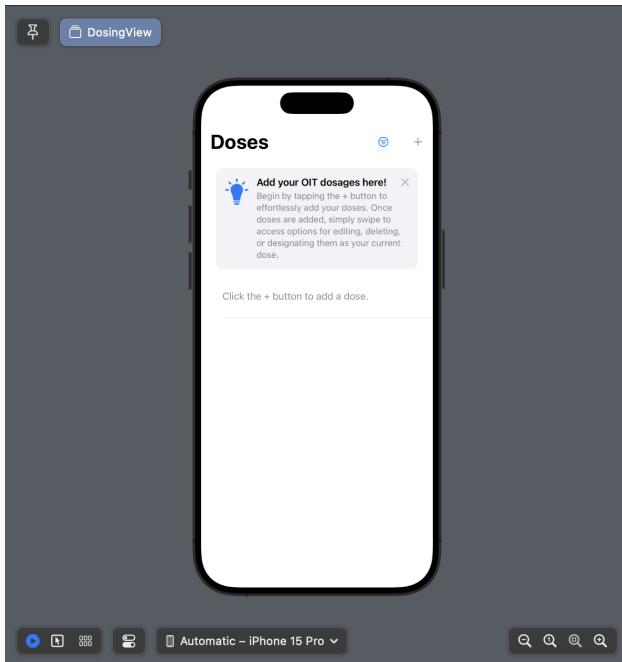


Figure 10.1: SwiftUI Canvas

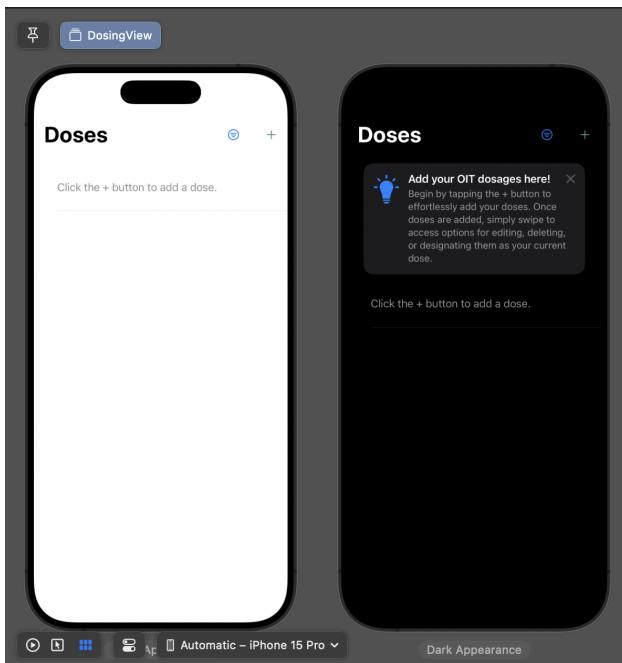


Figure 10.2: Swift UI Canvas Color Scheme Variants

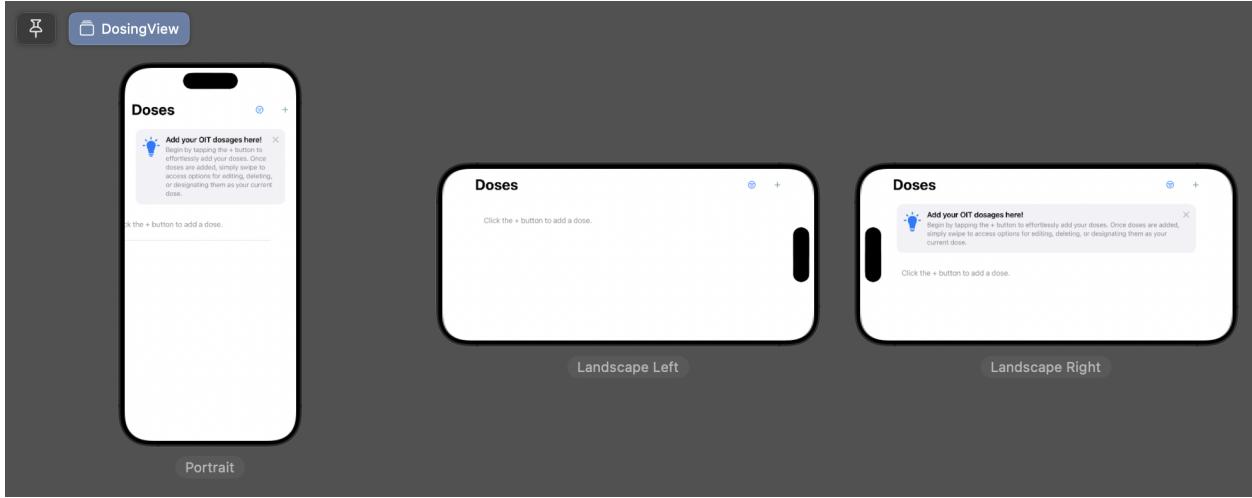


Figure 10.3: Swift UI Canvas Orientation Variants

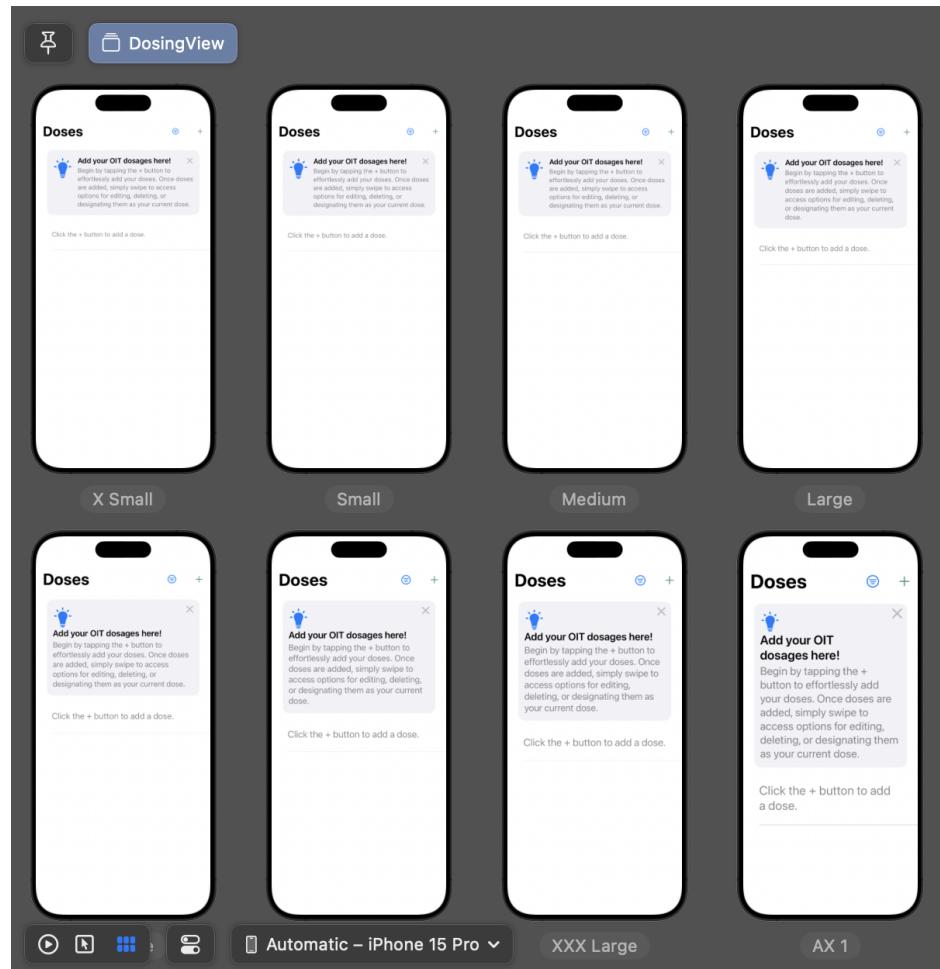


Figure 10.4: Swift UI Canvas Dynamic Type Variants

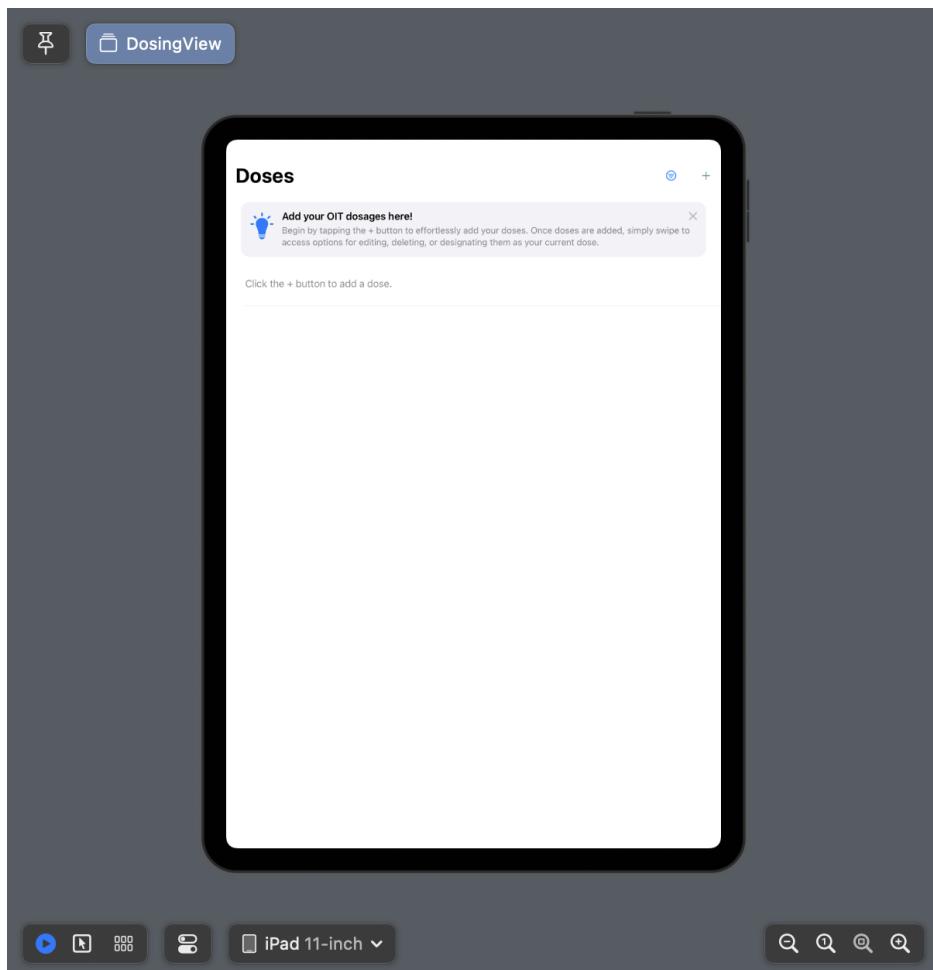


Figure 10.5: Swift UI Canvas Device Variant

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-friendliness.

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, in this paper, 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. 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

```
1 // SeniorDesignApp.swift
2 // SeniorDesign
3 //
4 // Created by Maddie on 9/25/23.
5 //
6
7
8 import SwiftUI
9 import LocalAuthentication
10
11 let hasAppBeenOpenedBeforeKey = "HasAppBeenOpenedBefore"
12
13 class AppState: ObservableObject {
14     @AppStorage(hasAppBeenOpenedBeforeKey) var hasAppBeenOpenedBefore: Bool = false
15 }
16
17 @main
18 struct OITApp: App {
19     // MARK: View Models
20     @StateObject var profileViewModel = ProfileViewModel()
21     @StateObject var healthKitViewModel = HealthKitViewModel()
22     @StateObject var doseViewModel = DoseViewModel()
23     @StateObject var profileImageViewModel = ProfileModel()
24     @StateObject var appState = AppState()
25
26     @State private var isUnlocked = false
27
28     func authenticate() {
29         let context = LAContext()
30         var error: NSError?
31
32         if context.canEvaluatePolicy(.deviceOwnerAuthenticationWithBiometrics, error: &error) {
33             let reason = "Allow OIT to use a passcode, TouchID, or FaceID to protect your data."
34
35             context.evaluatePolicy(.deviceOwnerAuthentication, localizedReason: reason) { success
36                 , authenticationError in
37                 if success {
38                     isUnlocked = true
39                     profileViewModel.loadProfileData()
40                     profileImageViewModel.loadProfileImage()
41                     doseViewModel.loadDoses()
42                 } else {
43                     // Error
44                 }
45             }
46         } else if context.canEvaluatePolicy(.deviceOwnerAuthentication, error: &error) {
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
```

```

46     let reason = "Allow OIT to use a passcode, TouchID, or FaceID to protect your data."
47
48     context.evaluatePolicy(.deviceOwnerAuthentication, localizedReason: reason) { success
49     , authenticationError in
50         if success {
51             isUnlocked = true
52             profileViewModel.loadProfileData()
53             profileImageViewModel.loadProfileImage()
54             doseViewModel.loadDoses()
55         } else {
56             // Error
57         }
58     }
59 }
60
// MARK: View of App
61 var body: some Scene {
62     WindowGroup {
63         if appState.hasAppBeenOpenedBefore {
64             TabbedApplicationView()
65                 .environmentObject(profileViewModel)
66                 .environmentObject(healthKitViewModel)
67                 .environmentObject(doseViewModel)
68                 .environmentObject(profileImageViewModel)
69                 .onAppear(perform: {
70                     profileViewModel.loadProfileData()
71                     profileImageViewModel.loadProfileImage()
72                     doseViewModel.loadDoses()
73                     if !isUnlocked && profileViewModel.profileData.useFaceID {
74                         authenticate()
75                     }
76                 })
77                 .onDisappear(perform: {
78                     isUnlocked = false
79                 })
80         } else {
81             GetStartedView()
82                 .environmentObject(profileViewModel)
83                 .environmentObject(healthKitViewModel)
84                 .environmentObject(doseViewModel)
85                 .environmentObject(appState)
86                 .environmentObject(profileImageViewModel)
87                 .onAppear {
88                     profileViewModel.loadProfileData()
89                     profileImageViewModel.loadProfileImage()
90                     doseViewModel.loadDoses()
91                 }
92         }
93     }
94 }
95 }
96 }
```

## A.2 TabbedApplicationView.swift

```

1 /**
2 //  TabbedApplicationView.swift
3 //  SeniorDesign
4 /**
5 //  Created by Maddie on 8/20/23.
6 /**
7
8 import SwiftUI
9
10 struct TabbedApplicationView: View {
11     // MARK: View Models
12     @EnvironmentObject var profileViewModel: ProfileViewModel
```

```

13 @EnvironmentObject var healthKitViewModel: HealthKitViewModel
14 @EnvironmentObject var doseViewModel: DoseViewModel
15 @EnvironmentObject var profileImageViewModel: ProfileViewModel
16
17 var body: some View {
18     TabView {
19         // MARK: Home Tab
20         NavigationStack {
21             HomeView()
22                 .environmentObject(profileImageViewModel)
23                 .environmentObject(healthKitViewModel)
24                 .environmentObject(doseViewModel)
25                 .environmentObject(profileImageViewModel)
26         }
27         .tabItem {
28             Label("Today", systemImage: "calendar")
29         }
30
31         // MARK: Insights Tab
32         NavigationStack {
33             InsightsView(healthKitViewModel: healthKitViewModel)
34                 .environmentObject(profileImageViewModel)
35                 .environmentObject(healthKitViewModel)
36                 .environmentObject(doseViewModel)
37         }
38         .tabItem {
39             Label("Insights", systemImage: "chart.line.uptrend.xyaxis")
40         }
41
42         // MARK: Dosing Tab
43         NavigationStack {
44             DosingView()
45                 .environmentObject(profileImageViewModel)
46                 .environmentObject(healthKitViewModel)
47                 .environmentObject(doseViewModel)
48         }
49         .tabItem {
50             Label("Dosing", systemImage: "pills.fill")
51         }
52
53         // MARK: Education Tab
54         NavigationStack {
55             EducationView()
56                 .environmentObject(profileImageViewModel)
57                 .environmentObject(healthKitViewModel)
58                 .environmentObject(doseViewModel)
59         }
60         .tabItem {
61             Label("Education", systemImage: "graduationcap.fill")
62         }
63     }
64     .tint(.darkTeal)
65     .onAppear(perform: {
66         profileImageViewModel.loadProfileData()
67         profileImageViewModel.loadProfileImage()
68         doseViewModel.loadDoses()
69     })
70 }
71 }
72
73 // MARK: Preview
74 #Preview {
75     TabbedApplicationView()
76         .environmentObject(ProfileViewModel())
77         .environmentObject(HealthKitViewModel())
78         .environmentObject(DoseViewModel())
79 }

```

### A.3 Extensions.swift

```
1 //  
2 // Extensions.swift  
3 // SeniorDesign  
4 //  
5 // Created by Maddie on 9/25/23.  
6 //  
7  
8 import Foundation  
9 import SwiftUI  
10  
11 extension Color {  
12     static var lightGrey: Color { Color(hex: "FAFAFA") }  
13     static var mediumGrey: Color { Color(hex: "D3D3D3") }  
14     static var darkGrey: Color { Color(hex: "222222") }  
15     static var grey: Color { Color(hex: "D5D5D5") }  
16     static var darkTeal: Color { Color(hex: "5A8F8F") }  
17     static var darkerTeal: Color { Color(hex: "0a5757") }  
18     static var lightTeal: Color { Color(hex: "C3E2E7") }  
19     static var lightBlue: Color { Color(hex: "E9F5F9") }  
20     static var lightYellow: Color { Color(hex: "FFFEC8") }  
21  
22     init(hex: String) {  
23         let scanner = Scanner(string: hex)  
24         var rgb: UInt64 = 0  
25  
26         scanner.scanHexInt64(&rgb)  
27  
28         let red = Double((rgb & 0xFF0000) >> 16) / 255.0  
29         let green = Double((rgb & 0x00FF00) >> 8) / 255.0  
30         let blue = Double(rgb & 0x0000FF) / 255.0  
31  
32         self.init(red: red, green: green, blue: blue)  
33     }  
34  
35     static var mutedRandom: Color {  
36         let red = CGFloat.random(in: 0.3...0.7)  
37         let green = CGFloat.random(in: 0.3...0.7)  
38         let blue = CGFloat.random(in: 0.3...0.7)  
39  
40         return Color(red: Double(red), green: Double(green), blue: Double(blue))  
41     }  
42 }  
43  
44 extension String {  
45     func splitCamelCase() -> String {  
46         var formattedString = self.replacingOccurrences(of: "HKCategoryTypeIdentifier", with: "")  
47  
48         let pattern = "(\\w)([A-Z])"  
49         let regex = try! NSRegularExpression(pattern: pattern, options: [])  
50         let range = NSRange(location: 0, length: formattedString.utf16.count)  
51  
52         let matches = regex.matches(in: formattedString, options: [], range: range)  
53         for match in matches.reversed() {  
54             let index = formattedString.index(formattedString.startIndex, offsetBy: match.range(at: 2).location)  
55             if index < formattedString.endIndex {  
56                 formattedString.insert(" ", at: index)  
57             }  
58         }  
59  
60         return formattedString  
61     }  
62 }
```

## A.4 AllergenWithDoses.swift

```
1 //  
2 //  AllergenWithDoses.swift  
3 //  SeniorDesign  
4 //  
5 //  Created by Maddie on 3/7/24.  
6 //  
7  
8 import Foundation  
9  
10 struct AllergenWithDoses: Identifiable, Hashable {  
11     static func == (lhs: AllergenWithDoses, rhs: AllergenWithDoses) -> Bool {  
12         return lhs.id == rhs.id  
13     }  
14  
15     func hash(into hasher: inout Hasher) {  
16         hasher.combine(id)  
17     }  
18  
19     let id = UUID()  
20     let allergen: String  
21     let doses: [Dose]  
22 }
```

## A.5 AntihistamineDose.swift

```
1 //  
2 //  AntihistamineDose.swift  
3 //  SeniorDesign  
4 //  
5 //  Created by Maddie on 3/7/24.  
6 //  
7  
8 import Foundation  
9  
10 struct AntihistamineDose: Hashable {  
11     let name: String  
12     let dose: String  
13 }
```

## A.6 AntihistamineDoseTransformer.swift

```
1 //  
2 //  AntihistamineDoseTransformer.swift  
3 //  SeniorDesign  
4 //  
5 //  Created by Maddie on 3/7/24.  
6 //  
7  
8 import Foundation  
9  
10 class AntihistamineDoseTransformer: NSSecureUnarchiveFromDataTransformer {  
11     override class func allowsReverseTransformation() -> Bool {  
12         return true  
13     }  
14  
15     override class func transformedValueClass() -> AnyClass {  
16         return NSData.self  
17     }  
18  
19     override func transformedValue(_ value: Any?) -> Any? {  
20         guard let data = try? NSKeyedArchiver.archivedData(withRootObject: value as?  
AntihistamineDose ?? AntihistamineDose(name: "", dose: "")) else {  
            return nil  
        }
```

```

22     }
23     return data as NSData
24 }
25
26 override func reverseTransformedValue(_ value: Any?) -> Any? {
27     guard let data = value as? Data else { return nil }
28     return try? NSKeyedUnarchiver.unarchiveTopLevelObjectWithData(data) as? AntihistamineDose
29 }
30 }
```

## A.7 Dose.swift

```

1 // Dose.swift
2 // SeniorDesign
3 //
4 // Created by Maddie on 3/7/24.
5 //
6
7
8 import Foundation
9
10 struct Dose: Identifiable, Codable {
11     var id: UUID {
12         return UUID()
13     }
14     var allergen: String
15     var doseType: String
16     var doseAmount: Double
17     var halfDose: Double {
18         return doseAmount / 2.0
19     }
20     var isCurrentDose: Bool // Add property to track current dose
21 }
```

## A.8 DoseTransformer.swift

```

1 // DoseRecords.swift
2 // SeniorDesign
3 //
4 // Created by Maddie on 12/28/23.
5 //
6
7
8 import Foundation
9
10 class DosesTransformer: NSSecureUnarchiveFromDataTransformer {
11     override func transformedValue(_ value: Any?) -> Any? {
12         guard let doses = value as? [String: String],
13             let data = try? NSKeyedArchiver.archivedData(withRootObject: doses,
14             requiringSecureCoding: false) else {
15             return nil
16         }
17         return data as NSData
18     }
19
20     override func reverseTransformedValue(_ value: Any?) -> Any? {
21         guard let data = value as? Data,
22             let doses = try? NSKeyedUnarchiver.unarchiveTopLevelObjectWithData(data) as? [
23             String: String] else {
24             return nil
25         }
26         return doses
27     }
28 }
```

## A.9 DoseViewModel.swift

```
1 //  
2 //  DoseViewModel.swift  
3 //  SeniorDesign  
4 //  
5 //  Created by Maddie on 1/1/24.  
6 //  
7  
8 import Foundation  
9  
10 class DoseViewModel: ObservableObject {  
11     @Published var doses: [Dose] = []  
12  
13     var allergensWithDoses: [AllergenWithDoses] {  
14         let groupedDoses = Dictionary(grouping: doses, by: { $0.allergen })  
15         let sortedAllergens = groupedDoses.keys.sorted()  
16         return sortedAllergens.map { allergen in  
17             AllergenWithDoses(allergen: allergen, doses: groupedDoses[allergen]!)  
18         }  
19     }  
20  
21     // Function to add a dose with an option to mark it as current  
22     func addDose(allergen: String, doseType: String, doseAmount: Double, isCurrentDose: Bool) {  
23         let newDose = Dose(allergen: allergen, doseType: doseType, doseAmount: doseAmount,  
24         isCurrentDose: isCurrentDose)  
25         doses.append(newDose)  
26  
27         let halfDoseAmount = doseAmount / 2.0  
28         let newHalfDose = Dose(allergen: allergen, doseType: "Half Dose for \(doseType)",  
29         doseAmount: halfDoseAmount, isCurrentDose: false)  
30         doses.append(newHalfDose)  
31  
32         saveDoses()  
33     }  
34  
35     func saveDoses() {  
36         let encoder = JSONEncoder()  
37         if let encoded = try? encoder.encode(doses) {  
38             UserDefaults.standard.set(encoded, forKey: "Doses")  
39         }  
40     }  
41  
42     func loadDoses() {  
43         if let savedDoses = UserDefaults.standard.data(forKey: "Doses") {  
44             let decoder = JSONDecoder()  
45             if let loadedDoses = try? decoder.decode([Dose].self, from: savedDoses) {  
46                 doses = loadedDoses  
47             }  
48         }  
49     }  
50  
51     // Function to set the current dose for a specific allergen  
52     func setCurrentDose(allergen: String, doseType: String) {  
53         for index in 0..54             if doses[index].allergen == allergen && doses[index].doseType == doseType {  
55                 doses[index].isCurrentDose = true  
56             } else {  
57                 doses[index].isCurrentDose = false  
58             }  
59         }  
60         saveDoses()  
61     }  
62  
63     func deleteDose(_ dose: Dose) {  
64         if let index = doses.firstIndex(where: { $0.doseType == dose.doseType }) {  
65             doses.remove(at: index)  
66         }  
67     }  
68 }
```

```

64         saveDoses()
65     }
66 }
67
68 func setAsCurrentDose(_ dose: Dose) {
69     for index in 0..

```

## A.10 ProfileData

```

1 // ProfileData.swift
2 // SeniorDesign
3 //
4 // Created by Maddie on 3/7/24.
5 //
6
7
8 import Foundation
9
10 class ProfileData: ObservableObject, Codable {
11     @Published var name: String = ""
12     @Published var dateOfBirth: Date = Date()
13     @Published var allergens: [String] = []
14     @Published var shareDataWithAppleHealth: Bool = false
15     @Published var useFaceID: Bool = false
16     @Published var commonAllergens = ["Milk", "Eggs", "Fish", "Shellfish", "Soy", "Peanuts", "Almonds", "Brazil Nuts", "Cashews", "Coconuts", "Hazelnuts", "Macadamia Nuts", "Pecans", "Pine Nuts", "Pistachios", "Walnuts", "Wheat", "Sesame"]
17     @Published var allergenEmojiMap: [String: String] = [
18         "Peanuts": "🥜",
19         "Almonds": "🥜",
20         "Brazil Nuts": "🥜",
21         "Cashews": "🥜",
22         "Coconuts": "🥥",
23         "Hazelnuts": "🌰",

```

```

24     "Macadamia Nuts": "    ",
25     "Pecans": "    ",
26     "Pine Nuts": "    ",
27     "Pistachios": "    ",
28     "Walnuts": "    ",
29     "Milk": "    ",
30     "Eggs": "    ",
31     "Wheat": "    ",
32     "Soy": "    ",
33     "Fish": "    ",
34     "Shellfish": "    ",
35     "Sesame": "    ",
36 ]
37
38 func isNut(_ allergen: String) -> Bool {
39     let nuts = ["Almonds", "Brazil Nuts", "Cashews", "Macadamia Nuts", "Pine Nuts", "
39 Pistachios", "Pecans", "Walnuts"]
40     return nuts.contains(allergen)
41 }
42
43 enum CodingKeys: String, CodingKey {
44     case name
45     case dateOfBirth
46     case allergens
47     case shareDataWithAppleHealth
48     case useFaceID
49     case commonAllergens
50     case allergenEmojiMap
51 }
52
53 required init(from decoder: Decoder) throws {
54     let container = try decoder.container(keyedBy: CodingKeys.self)
55     name = try container.decode(String.self, forKey: .name)
56     dateOfBirth = try container.decode(Date.self, forKey: .dateOfBirth)
57     allergens = try container.decode([String].self, forKey: .allergens)
58     shareDataWithAppleHealth = try container.decode(Bool.self, forKey: .
shareDataWithAppleHealth)
59     useFaceID = try container.decode(Bool.self, forKey: .useFaceID)
60     commonAllergens = try container.decode([String].self, forKey: .commonAllergens)
61     allergenEmojiMap = try container.decode([String : String].self, forKey: .allergenEmojiMap)
62 }
63
64 func encode(to encoder: Encoder) throws {
65     var container = encoder.container(keyedBy: CodingKeys.self)
66     try container.encode(name, forKey: .name)
67     try container.encode(dateOfBirth, forKey: .dateOfBirth)
68     try container.encode(allergens, forKey: .allergens)
69     try container.encode(shareDataWithAppleHealth, forKey: .shareDataWithAppleHealth)
70     try container.encode(useFaceID, forKey: .useFaceID)
71     try container.encode(commonAllergens, forKey: .commonAllergens)
72     try container.encode(allergenEmojiMap, forKey: .allergenEmojiMap)
73 }
74
75 init() { }
76 }

```

## A.11 ProfileImageModel.swift

```

1 import SwiftUI
2 import PhotosUI
3 import CoreTransferable
4 import Foundation
5
6 @MainActor
7 class ProfileModel: ObservableObject {
8

```

```

9 // MARK: - Profile Details
10
11 @Published var firstName: String = ""
12 @Published var lastName: String = ""
13 @Published var aboutMe: String = ""
14
15 // MARK: - Profile Image
16
17 enum ImageState {
18     case empty
19     case loading(Progress)
20     case success(Image)
21     case failure(Error)
22 }
23
24 enum TransferError: Error {
25     case importFailed
26 }
27
28 struct ProfileImage: Transferable {
29     let image: Image
30
31     static var transferRepresentation: some TransferRepresentation {
32         DataRepresentation(importedContentType: .image) { data in
33             #if canImport(AppKit)
34                 guard let nsImage = NSImage(data: data) else {
35                     throw TransferError.importFailed
36                 }
37                 let image = Image(nsImage: nsImage)
38                 return ProfileImage(image: image)
39             #elseif canImport(UIKit)
40                 guard let uiImage = UIImage(data: data) else {
41                     throw TransferError.importFailed
42                 }
43                 let image = Image(uiImage: uiImage)
44                 return ProfileImage(image: image)
45             #else
46                 throw TransferError.importFailed
47             #endif
48         }
49     }
50 }
51
52 @Published private(set) var imageState: ImageState = .empty
53
54 @Published var imageSelection: PhotosPickerItem? = nil {
55     didSet {
56         if let imageSelection {
57             let progress = loadTransferable(from: imageSelection)
58             imageState = .loading(progress)
59         } else {
60             imageState = .empty
61         }
62     }
63 }
64
65 // MARK: - Private Methods
66
67 private func loadTransferable(from imageSelection: PhotosPickerItem) -> Progress {
68     return imageSelection.loadTransferable(type: ProfileImage.self) { result in
69         DispatchQueue.main.async {
70             guard imageSelection == self.imageSelection else {
71                 print("Failed to get the selected item.")
72                 return
73             }
74             switch result {
75                 case .success(let profileImage):
76                     self.imageState = .success(profileImage.image)
77             }
78         }
79     }
80 }

```

```

77         case .success(nil):
78             self.imageState = .empty
79         case .failure(let error):
80             self.imageState = .failure(error)
81     }
82 }
83 }
84 }
85
86 private let profileImageKey = "profileImage"
87
88 init() {
89     loadProfileImage()
90 }
91
92 func saveProfileImage() {
93     if case let .success(profileImage) = imageState {
94         if let data = try? NSKeyedArchiver.archivedData(withRootObject: profileImage,
95             requiringSecureCoding: false) {
96             UserDefaults.standard.set(data, forKey: profileImageKey)
97         }
98     }
99 }
100
101 func loadProfileImage() {
102     if let data = UserDefaults.standard.data(forKey: profileImageKey),
103         let profileImage = try? NSKeyedUnarchiver.unarchiveTopLevelObjectWithData(data) as?
104     Image {
105         imageState = .success(profileImage)
106     }
107 }
108 }
```

## A.12 ProfileImageViewModel.swift

```

1 import SwiftUI
2 import PhotosUI
3 import CoreTransferable
4 import Foundation
5
6 @MainActor
7 class ProfileModel: ObservableObject {
8     // MARK: - Profile Image
9
10    enum ImageState {
11        case empty
12        case loading(Progress)
13        case success(Image)
14        case failure(Error)
15    }
16
17    enum TransferError: Error {
18        case importFailed
19    }
20
21    struct ProfileImage: Transferable {
22        let image: Image
23
24        static var transferRepresentation: some TransferRepresentation {
25            DataRepresentation(importedContentType: .image) { data in
26                #if canImport(AppKit)
27                    guard let nsImage = NSImage(data: data) else {
28                        throw TransferError.importFailed
29                    }
30                    let image = Image(nsImage: nsImage)
31                    return ProfileImage(image: image)
32                #elseif canImport/UIKit)
33            }
34        }
35    }
36 }
```

```

33     guard let uiImage = UIImage(data: data) else {
34         throw TransferError.importFailed
35     }
36     let image = Image(uiImage: uiImage)
37     return ProfileImage(image: image)
38 #else
39     throw TransferError.importFailed
40 #endif
41 }
42 }
43 }
44
45 @Published private(set) var imageState: ImageState = .empty
46
47 @Published var imageSelection: PhotosPickerItem? = nil {
48     didSet {
49         if let imageSelection {
50             let progress = loadTransferable(from: imageSelection)
51             imageState = .loading(progress)
52         } else {
53             imageState = .empty
54         }
55     }
56 }
57
58 // MARK: - Private Methods
59
60 private func loadTransferable(from imageSelection: PhotosPickerItem) -> Progress {
61     return imageSelection.loadTransferable(type: ProfileImage.self) { result in
62         DispatchQueue.main.async {
63             guard imageSelection == self.imageSelection else {
64                 print("Failed to get the selected item.")
65                 return
66             }
67             switch result {
68                 case .success(let profileImage?):
69                     self.imageState = .success(profileImage.image)
70                 case .success(nil):
71                     self.imageState = .empty
72                 case .failure(let error):
73                     self.imageState = .failure(error)
74             }
75         }
76     }
77 }
78
79 private let profileImageKey = "profileImage"
80
81 init() {
82     loadProfileImage()
83 }
84
85 func saveProfileImage() {
86     if case let .success(profileImage) = imageState {
87         if let data = profileImage.getUIImage(newSize: CGSize(width: 100, height: 100))?
88             pngData() {
89                 UserDefaults.standard.setValue(data, forKey: profileImageKey)
90             }
91     }
92 }
93
94 func loadProfileImage() {
95     if let data = UserDefaults.standard.data(forKey: profileImageKey),
96         let uiImage = UIImage(data: data){
97             imageState = .success(Image(uiImage: uiImage))
98     }
99 }

```

```

100
101 extension Image {
102     @MainActor
103     func getUIImage(newSize: CGSize) -> UIImage? {
104         let image = resizable()
105             .scaledToFill()
106             .frame(width: newSize.width, height: newSize.height)
107             .clipped()
108         return ImageRenderer(content: image).uiImage
109     }
110 }
```

## A.13 ProfileViewModel.swift

```

1 ///
2 //  PersonalInformation.swift
3 //  SeniorDesign
4 //-
5 //  Created by Maddie on 10/20/23.
6 //-
7
8 import Combine
9 import Foundation
10
11 class ProfileViewModel: ObservableObject {
12     @Published var profileData = ProfileData()
13     @Published var selectedAllergens: Set<String> = []
14     @Published var numAllergens = 0 // Variable to track the number of allergens
15
16     func saveProfileData() {
17         let encoder = JSONEncoder()
18         if let encoded = try? encoder.encode(profileData) {
19             UserDefaults.standard.set(encoded, forKey: "ProfileData")
20         }
21     }
22
23     func loadProfileData() {
24         if let savedProfileData = UserDefaults.standard.data(forKey: "ProfileData") {
25             let decoder = JSONDecoder()
26             if let loadedProfile = try? decoder.decode(ProfileData.self, from: savedProfileData)
27             {
28                 profileData = loadedProfile
29                 numAllergens = loadedProfile.allergens.count
30             }
31         }
32
33     func addNewAllergen() {
34         profileData.allergens.append("Select an Allergen")
35         numAllergens += 1 // Increment the allergen count
36         saveProfileData()
37     }
38
39     func removeSelectedAllergen(at index: Int) {
40         if profileData.allergens.indices.contains(index) {
41             profileData.allergens.remove(at: index)
42             numAllergens -= 1 // Decrement the allergen count
43             saveProfileData()
44         }
45     }
46
47     func toggleAllergenSelection(_ allergen: String) {
48         if selectedAllergens.contains(allergen) {
49             selectedAllergens.remove(allergen)
50         } else {
51             selectedAllergens.insert(allergen)
52 }
```

```

53         }
54     }
55
56     func isSelected(_ allergen: String) -> Bool {
57         return selectedAllergens.contains(allergen)
58     }
59 }
```

## A.14 SelectedAllergensTransformer

```

1 // SelectedAllergensTransformer.swift
2 // SeniorDesign
3 //
4 // Created by Maddie on 3/7/24.
5 //
6 //
7
8 import Foundation
9
10 class SelectedAllergensTransformer: NSSecureUnarchiveFromDataTransformer {
11     override static var allowedTopLevelClasses: [AnyClass] {
12         return [NSSet.self]
13     }
14 }
```

## A.15 SymptomDataManager.swift

```

1 import SwiftUI
2 import CoreData
3
4 class SymptomDataManager: ObservableObject {
5     @MainActor
6     @Published var symptomRecords: [Date: [String]] = [:]
7
8     @MainActor func saveSymptoms(for date: Date, symptoms: [String]) {
9         // Update the symptom records dictionary with the new symptoms for the given date
10        symptomRecords[date] = symptoms
11
12        let context = persistentContainer.viewContext
13        let symptomRecord = SymptomRecord(context: context)
14        symptomRecord.date = date
15        symptomRecord.symptoms = symptoms as NSObject
16
17        do {
18            try context.save()
19        } catch {
20            print("Failed to save symptoms: \(error.localizedDescription)")
21        }
22    }
23
24    @MainActor func fetchSymptoms(for date: Date) -> [String] {
25        if let symptoms = symptomRecords[date] {
26            // Return symptoms from the dictionary if available
27            return symptoms
28        } else {
29            let context = persistentContainer.viewContext
30            let fetchRequest: NSFetchedRequest<SymptomRecord> = SymptomRecord.fetchRequest()
31            fetchRequest.predicate = NSPredicate(format: "date == %@", date as CVarArg)
32
33            do {
34                let result = try context.fetch(fetchRequest)
35                if let record = result.first {
36                    if let fetchedSymptoms = record.symptoms as? [String] {
37                        // Update the symptom records dictionary with the fetched symptoms for
38                        caching
39                        symptomRecords[date] = fetchedSymptoms
40                    }
41                }
42            }
43        }
44    }
45 }
```

```

39             return fetchedSymptoms
40         }
41     }
42 } catch {
43     print("Failed to fetch symptoms: \(error.localizedDescription)")
44 }
45 }
46 return []
47 }

48 // Function to fetch all stored symptoms from Core Data
49 @MainActor func fetchAllSymptoms() -> [String] {
50     let context = persistentContainer.viewContext
51     let fetchRequest: NSFetchedRequest<SymptomRecord> = SymptomRecord.fetchRequest()
52
53     do {
54         let result = try context.fetch(fetchRequest)
55         var allSymptoms: [String] = []
56
57         // Iterate through fetched records and populate symptomRecords dictionary
58         for record in result {
59             if let date = record.date, let symptoms = record.symptoms as? [String] {
60                 symptomRecords[date] = symptoms
61                 allSymptoms.append(contentsOf: symptoms)
62             }
63         }
64     }
65
66     return allSymptoms
67 } catch {
68     print("Failed to fetch all symptoms: \(error.localizedDescription)")
69     return []
70 }
71 }

72 func fetchAllSymptomsWithoutRefresh() -> [String] {
73     let context = persistentContainer.viewContext
74     let fetchRequest: NSFetchedRequest<SymptomRecord> = SymptomRecord.fetchRequest()
75
76     do {
77         let result = try context.fetch(fetchRequest)
78         let allSymptoms = result.flatMap { ($0.symptoms as? [String]) ?? [] }
79         return allSymptoms
80     } catch {
81         print("Failed to fetch all symptoms: \(error.localizedDescription)")
82         return []
83     }
84 }
85 }

86 lazy var persistentContainer: NSPersistentContainer = {
87     let container = NSPersistentContainer(name: "SymptomDataModel")
88     container.loadPersistentStores { _, error in
89         if let error = error {
90             fatalError("Failed to load persistent store: \(error)")
91         }
92     }
93     return container
94 }
95 }

96 func lastReactionDate() -> Date? {
97     let context = persistentContainer.viewContext
98     let fetchRequest: NSFetchedRequest<SymptomRecord> = SymptomRecord.fetchRequest()
99     fetchRequest.sortDescriptors = [NSSortDescriptor(key: "date", ascending: false)]
100    fetchRequest.fetchLimit = 1
101
102    do {
103        let result = try context.fetch(fetchRequest)
104        if let record = result.first {
105            return record.date
106        }
107    }
108 }

```

```

107         }
108     } catch {
109         print("Failed to fetch last reaction date: \(error.localizedDescription)")
110     }
111
112     return nil
113 }
114 }
```

## A.16 AddDoseView.swift

```

1 //  

2 //  AddDoseView.swift  

3 //  SeniorDesign  

4 //  

5 //  Created by Maddie on 10/20/23.  

6 //  

7  

8 import SwiftUI  

9  

10 struct AddDoseView: View {
11     // MARK: View Models
12     @EnvironmentObject var doseViewModel: DoseViewModel
13     @EnvironmentObject var profileViewModel: ProfileViewModel
14  

15     // MARK: Variables
16     @Environment(\.presentationMode) var presentationMode
17     @State private var selectedAllergen = ""
18     @State private var doseName = ""
19     @State private var doseAmount = ""
20     @State private var isCurrentDose = false
21     @State var editingDose: Dose?
22  

23     // MARK: Add Dose View
24     var body: some View {
25         NavigationStack {
26             Form {
27                 Section(header: Text("Allergen")) {
28                     Picker("Select Allergen", selection: $selectedAllergen) {
29                         ForEach(profileViewModel.profileData.allergens, id: \.self) { allergen in
30                             Text(allergen)
31                         }
32                     }
33                     .onAppear {
34                         if selectedAllergen.isEmpty, let firstAllergen = profileViewModel.
35                         profileData.allergens.first {
36                             selectedAllergen = firstAllergen
37                         }
38                     }
39                     .pickerStyle(DefaultPickerStyle())
40                 }
41                 Section(header: Text("Dose Information")) {
42                     TextField("Dose Name", text: $doseName)
43                     HStack {
44                         Text("Dose")
45                         Spacer()
46                         TextField("", text: $doseAmount)
47                             .keyboardType(.decimalPad)
48                             .textFieldStyle(RoundedBorderTextFieldStyle())
49                             .frame(width: 100)
50                         Text("mg")
51                     }
52                 }
53  

54                 Section(header: Text("Mark as Current Dose")) {
55                     Toggle("Current Dose", isOn: $isCurrentDose)
56                 }
57             }
58         }
59     }
60 }
```

```

56         }
57     }
58     .onAppear {
59         if let editingDose = editingDose {
60             selectedAllergen = editingDose.allergen
61             doseName = editingDose.doseType
62             doseAmount = "\u{1d64}(editingDose.doseAmount)"
63             isCurrentDose = editingDose.isCurrentDose
64         }
65     }
66     .navigationTitle(editingDose != nil ? "Edit Dose" : "Add Dose")
67     .navigationBarItems(trailing:
68         Button("Save") {
69             if let dosage = Double(doseAmount), !doseAmount.isEmpty {
70                 if let editingDose = editingDose {
71                     // Update existing dose
72                     doseViewModel.updateDose(editingDose, allergen: selectedAllergen,
73                     doseType: doseName, doseAmount: dosage, isCurrentDose: isCurrentDose)
74                 } else {
75                     // Add new dose
76                     doseViewModel.addDose(allergen: selectedAllergen, doseType:
77                     doseName, doseAmount: dosage, isCurrentDose: isCurrentDose)
78                 }
79             }
80         }
81     }
82 }
83 }
```

## A.17 DoseRowView.swift

```

1 /**
2  // DoseRowView.swift
3  // SeniorDesign
4 /**
5  // Created by Maddie on 3/7/24.
6 /**
7
8 import Foundation
9 import SwiftUI
10
11 struct DoseRowView: View {
12     let dose: Dose
13     let numberFormatter: NumberFormatter
14     let viewModel: DoseViewModel
15
16     @State private var createNewDose = false
17     @State private var editingDose: Dose?
18
19     var body: some View {
20         VStack(alignment: .leading) {
21             HStack {
22                 Text(dose.doseType)
23                     .font(.headline)
24                 if dose.isCurrentDose {
25                     Image(systemName: "star.circle.fill")
26                         .foregroundColor(.darkTeal)
27                 }
28             }
29             if let formattedDosage = numberFormatter.string(for: dose.doseAmount) {
30                 Text("Dosage: \u{1d64}(formattedDosage) mg")
31                     .font(.subheadline)
32                     .foregroundColor(.gray)
33             }
34         }
35     }
36 }
```

```

35     .swipeActions(edge: .leading) {
36         Button {
37             viewModel.setAsCurrentDose(dose)
38         } label: {
39             Label("Set as Current", systemImage: "star")
40         }
41         .tint(.darkTeal)
42
43         Button(action: {
44             editingDose = dose
45         }, label: {
46             Label("Edit", systemImage: "pencil")
47         })
48         .tint(.yellow)
49     }
50     .navigationBarHidden(true)
51     .background(
52         NavigationLink(
53             destination: AddDoseView(editingDose: editingDose)
54             .environmentObject(viewModel),
55             isActive: Binding<Bool>(
56                 get: { editingDose != nil },
57                 set: { _ in editingDose = nil }
58             ),
59             label: { EmptyView() }
60         )
61         .hidden()
62     )
63 }
64 }
```

## A.18 DosingView.swift

```

1 /**
2 //  DosingView.swift
3 //  SeniorDesign
4 /**
5 //  Created by Maddie on 10/18/23.
6 /**
7
8 import SwiftUI
9 import TipKit
10
11 struct DosingView: View {
12     // MARK: View Models
13     @EnvironmentObject var doseViewModel: DoseViewModel
14
15     // MARK: Variables
16     @State private var createNewDose = false
17     @State private var isEditing = false
18     @Environment(\.colorScheme) var colorScheme
19
20     @State private var showFilterOptions = false
21     @State private var selectedFilter: FilterOption = .all
22
23     var dosingTip = DosingViewTip()
24
25     enum FilterOption: String, CaseIterable {
26         case all = "All Doses"
27         case currentDoses = "Show Only Current Doses"
28     }
29
30     var filterButton: some View {
31         Menu {
32             ForEach(FilterOption.allCases, id: \.self) { option in
33                 Button(action: {
34                     selectedFilter = option
35                 })
36             }
37         }
38     }
39
40     var body: some View {
41         NavigationView {
42             ...
43         }
44     }
45 }
```

```

35             })
36             Label(option.rawValue, systemImage: selectedFilter == option ? "checkmark" :
37         "")
38         }
39     } label: {
40         Image(systemName: "line.horizontal.3.decrease.circle")
41     }
42     .padding()
43 }
44
45 var filteredDoses: [AllergenWithDoses] {
46     var filteredAllergens = doseViewModel.allergensWithDoses
47
48     switch selectedFilter {
49     case .all:
50         break
51     case .currentDoses:
52         filteredAllergens = filteredAllergens.map { allergenWithDoses in
53             AllergenWithDoses(
54                 allergen: allergenWithDoses.allergen,
55                 doses: allergenWithDoses.doses.filter(\.isCurrentDose)
56             )
57         }
58     }
59
60     return filteredAllergens
61 }
62 var numberFormatter: NumberFormatter = {
63     let formatter = NumberFormatter()
64     formatter.numberStyle = .decimal
65     formatter.maximumFractionDigits = 1
66     return formatter
67 }()
68
69 // MARK: UI Elements
70 var createNewDoseButton: some View {
71     Button(action: {
72         createNewDose = true
73     }, label: {
74         Image(systemName: "plus")
75             .foregroundColor(.darkTeal)
76     })
77     .sheet(isPresented: $createNewDose, content: {
78         AddDoseView()
79             .environmentObject(doseViewModel)
80     })
81     .padding()
82 }
83
84 var header: some View {
85     HStack {
86         Text("Doses")
87             .font(.largeTitle.bold())
88             .padding()
89         Spacer()
90         filterButton
91         createNewDoseButton
92     }
93 }
94
95 @State private var expandedStates: [String: Bool] = [:]
96
97 var listOfDoses: some View {
98     List {
99         if filteredDoses.isEmpty {
100             Text("Click the + button to add a dose.")
101             .foregroundColor(.gray)

```

```

102         .padding()
103     }
104     ForEach(filteredDoses) { allergenWithDoses in
105         DisclosureGroup(
106             isExpanded: Binding(
107                 get: { expandedStates[allergenWithDoses.allergen] ?? false },
108                 set: { expandedStates[allergenWithDoses.allergen] = $0 }
109             ),
110             content: {
111                 ForEach(allergenWithDoses.doses) { dose in
112                     DoseRowView(dose: dose, numberFormatter: numberFormatter, viewModel:
113                         doseViewModel)
114                 }
115                 .onDelete { indices in
116                     indices.forEach { index in
117                         doseViewModel.deleteDose(allergenWithDoses.doses[index])
118                     }
119                 }
120             },
121             label: {
122                 Text(allergenWithDoses.allergen)
123                     .font(.title2)
124                     .bold()
125                     .foregroundColor(colorScheme == .dark ? Color.white : Color.black)
126             }
127         )
128     }
129     .listStyle(PlainListStyle())
130     .background(colorScheme == .dark ? Color.black : Color.white)
131     .onAppear {
132         doseViewModel.loadDoses()
133         for allergenWithDoses in doseViewModel.allergensWithDoses {
134             expandedStates[allergenWithDoses.allergen] = true
135         }
136     }
137     .onDisappear(perform: {
138         doseViewModel.saveDoses()
139     })
140 }
141
// MARK: Dosing View
142 var body: some View {
143     VStack(alignment: .leading) {
144         header
145         if #available(iOS 17.0, *) {
146             TipView(dosingTip, arrowEdge: .none)
147                 .tipCornerRadius(15)
148                 .padding(.leading, 25)
149                 .padding(.trailing, 25)
150         }
151         listOfDoses
152     }
153     .task {
154         if #available(iOS 17.0, *) {
155             try? Tips.configure([
156                 .displayFrequency(.immediate),
157                 .datastoreLocation(.applicationDefault)
158             ])
159         }
160     }
161 }
162 }
163 }
164
// MARK: Preview
165 #Preview {
166     DosingView()
167         .environmentObject(DoseViewModel())

```

169 }

## A.19 DosingViewTip.swift

```
1 //  
2 //  DosingViewTip.swift  
3 //  SeniorDesign  
4 //  
5 //  Created by Maddie on 3/7/24.  
6 //  
7  
8 import Foundation  
9 import TipKit  
10  
11 struct DosingViewTip: Tip, Identifiable {  
12     var id = UUID()  
13     var title: Text {  
14         Text("Add your OIT dosages here!")  
15     }  
16  
17     var message: Text? {  
18         Text("Begin by tapping the + button to effortlessly add your doses. Once doses are added,  
19             simply swipe to access options for editing, deleting, or designating them as your current  
20             dose.")  
21     }  
22  
23     var image: Image? {  
24         Image(systemName: "lightbulb.max.fill")  
25     }  
26 }
```

## A.20 ArticleRichLink.swift

```
1 //  
2 //  ArticleRichLink.swift  
3 //  SeniorDesign  
4 //  
5 //  Created by Maddie on 10/20/23.  
6 //  
7  
8 import SwiftUI  
9  
10 struct ArticleRichLink: View {  
11     // MARK: Variables  
12     @State private var isShowingArticle = false  
13     var articleTitle: String  
14     var articleDescription: String  
15     var articleDisclaimer: AttributedString  
16     var articleContent: AttributedString  
17     var image: String  
18  
19     @Environment(\.colorScheme) var colorScheme  
20  
21     // MARK: Article Rich Link View  
22     var body: some View {  
23         NavigationLink(destination: ArticleView(articleTitle: articleTitle, articleDescription:  
24             articleContent, articleDisclaimer: articleDisclaimer, image: image))  
25             {  
26                 ZStack(alignment: .bottom) {  
27                     VStack(alignment: .leading) {  
28                         Image(image)  
29                             .resizable()  
30                             .aspectRatio(contentMode: .fill)  
31                             .frame(width: UIScreen.main.bounds.width - 30, height: 180)  
32                             .padding(.top, 0)  
33                             .padding(.bottom, 10)
```

```

33         .clipped()
34     HStack {
35         Text(articleTitle)
36             .font(.title2.bold())
37             .foregroundColor(colorScheme == .light ? .black : .white)
38             .padding(.leading, 20)
39         Spacer()
40     }
41     Text(articleDescription)
42         .font(.body)
43         .foregroundColor(colorScheme == .light ? .black : .white)
44         .padding(.leading, 20)
45     Spacer()
46 }
47     .frame(width: UIScreen.main.bounds.width - 30, height: 260)
48     .background(colorScheme == .light ? Color.mediumGrey : Color.darkGrey)
49     .cornerRadius(20)
50 }
51 }
52     .buttonStyle(PlainButtonStyle())
53 }
54 }
```

## A.21 ArticleView.swift

```

1 ///
2 //  ArticleView.swift
3 //  SeniorDesign
4 //
5 //  Created by Maddie on 10/20/23.
6 //
7
8 import SwiftUI
9
10 struct ArticleView: View {
11     // MARK: Variables
12     var articleTitle: String
13     var articleDescription: NSAttributedString
14     var articleDisclaimer: NSAttributedString
15     var image: String
16
17     @Environment(\.colorScheme) var colorScheme
18
19     // MARK: Article View
20     var body: some View {
21         ScrollView {
22             VStack {
23                 HStack {
24                     Text(articleTitle)
25                         .font(.largeTitle.bold())
26                         .padding()
27                     Spacer()
28                 }
29                 Image(image)
30                     .resizable()
31                     .aspectRatio(contentMode: .fill)
32                     .frame(width: UIScreen.main.bounds.width - 30, height: 160)
33                     .background(Color.lightTeal)
34                     .cornerRadius(20)
35             HStack {
36                 Spacer()
37                 VStack {
38                     Text(articleDisclaimer)
39                         .padding(.vertical, 20)
40                         .padding(.horizontal, 20)
41                         .frame(width: UIScreen.main.bounds.width - 30)
42                         .background(Color.lightYellow)
43                 }
44             }
45         }
46     }
47 }
```

```

43         .foregroundColor(.black)
44         .cornerRadius(20)
45         .padding(.top, 20)
46     Text(articleDescription)
47         .padding(20)
48     }
49     Spacer()
50 }
51 Spacer()
52 }
53 }
54 }
55 }
```

## A.22 DoctorPhoneTip.swift

```

1 /**
2 //  DoctorPhoneTip.swift
3 //  SeniorDesign
4 //
5 //  Created by Maddie on 3/15/24.
6 //
7
8 import Foundation
9 import TipKit
10
11 struct DoctorPhoneTip: Tip, Identifiable {
12     var id = UUID()
13     var title: Text {
14         Text("Add Important Contact Info")
15     }
16
17     var message: Text? {
18         Text("Add your doctor's phone number below, so you can easily reach out to them if needed
19 !")
20     }
21
22     var image: Image? {
23         Image(systemName: "stethoscope.circle.fill")
24 }
```

## A.23 EducationView.swift

```

1 /**
2 //  EducationView.swift
3 //  SeniorDesign
4 //
5 //  Created by Maddie on 10/18/23.
6 //
7
8 import SwiftUI
9 import TipKit
10
11 struct ImageData: Codable {
12     var imageData: Data
13 }
14
15 struct EducationView: View {
16     // MARK: Variables
17     let content = EducationViewContent()
18     var resourcesTip = ResourcesTip()
19     var esTip = EmergencyServicesTip()
20     var infoTip = ImportantDocumentsTip()
21     var doctorTip = DoctorPhoneTip()
22 }
```

```

23 @State var doctorPhoneNumber: String = ""
24 @State var isShowingImagePicker: Bool = false
25 @State private var selectedImages: [UIImage] = []
26 @State private var storedImageData: [ImageData] = []
27
28 @Environment(\.colorScheme) var colorScheme
29
30 // MARK: UI Elements
31 var articles: some View {
32     VStack(alignment: .leading) {
33         ArticleRichLink(articleTitle: "Anaphylaxis", articleDescription: "Learn the signs &
34 symptoms of anaphylaxis.", articleDisclaimer: content.anaphylaxisDisclaimer, articleContent:
35 content.anaphylaxisTips, image: "anaphylaxis")
36             .padding(7)
37         ArticleRichLink(articleTitle: "OIT Best Practices", articleDescription: "Tips and
38 tricks for Oral Immunotherapy.", articleDisclaimer: content.oitDisclaimer, articleContent:
39 content.oitTips, image: "oit")
40             .padding(7)
41         ArticleRichLink(articleTitle: "Avoiding Cross Contamination", articleDescription: "
42 Tips to help avoid cross contamination.", articleDisclaimer: content.ccDisclaimer,
43 articleContent: content.crossContaminationTips, image: "crossContamination")
44             .padding(7)
45     }
46 }
47
48 var emergencyServicesInformation: some View {
49     VStack(alignment: .center) {
50         HStack {
51             Text("Emergency Services")
52                 .font(.title3).bold()
53                 .padding()
54             Spacer()
55         }
56         if #available(iOS 17.0, *) {
57             TipView(esTip, arrowEdge: .bottom)
58                 .tipCornerRadius(15)
59                 .padding(.leading, 25)
60                 .padding(.trailing, 25)
61         }
62         Button(action: {
63             if let url = URL(string: "tel://911") {
64                 UIApplication.shared.open(url, options: [:], completionHandler: nil)
65             }
66         }) {
67             Image(systemName: "phone.fill")
68                 .resizable()
69                 .frame(width: 20, height: 20)
70                 .foregroundColor(.black)
71                 .padding(.leading, 20)
72             Text("911")
73                 .font(.body)
74                 .bold()
75                 .foregroundColor(.black)
76         }
77         .frame(width: UIScreen.main.bounds.width - 30, height: 50)
78         .background(colorScheme == .light ? Color.lightTeal : Color.darkTeal)
79         .cornerRadius(10)
80     }
81 }
82
83 var fareInformation: some View {
84     VStack(alignment: .center) {
85         HStack {
86             Text("Food Allergy Research & Education")
87                 .font(.title3).bold()
88                 .padding()
89             Spacer()
90         }

```

```

85     HStack{
86         Button(action: {
87             if let url = URL(string: "https://www.foodallergy.org") {
88                 UIApplication.shared.open(url)
89             }
90         }) {
91             Image(systemName: "safari.fill")
92                 .resizable()
93                 .frame(width: 20, height: 20)
94                 .foregroundColor(.black)
95                 .padding(.leading, 20)
96             Text("Website")
97                 .font(.body)
98                 .bold()
99                 .foregroundColor(.black)
100        }
101        .frame(width: (UIScreen.main.bounds.width - 40) / 2, height: 50)
102        .background(colorScheme == .light ? Color.lightTeal : Color.darkTeal)
103        .cornerRadius(10)
104
105        Button(action: {
106            if let url = URL(string: "tel://18009294040") {
107                UIApplication.shared.open(url, options: [:], completionHandler: nil)
108            }
109        }) {
110            Image(systemName: "phone.fill")
111                 .resizable()
112                 .frame(width: 20, height: 20)
113                 .foregroundColor(.black)
114                 .padding(.leading, 20)
115             Text("Phone")
116                 .font(.body)
117                 .bold()
118                 .foregroundColor(.black)
119        }
120        .frame(width: (UIScreen.main.bounds.width - 40) / 2, height: 50)
121        .background(colorScheme == .light ? Color.lightTeal : Color.darkTeal)
122        .cornerRadius(10)
123    }
124}
125}
126
127 var doctorClinicInformation: some View {
128     VStack(alignment: .center) {
129         HStack {
130             Text("Doctor / Clinic Contact")
131                 .font(.title3).bold()
132                 .padding()
133             Spacer()
134         }
135         if #available(iOS 17.0, *) {
136             TipView(doctorTip, arrowEdge: .bottom)
137                 .tipCornerRadius(15)
138                 .padding(.leading, 25)
139                 .padding(.trailing, 25)
140         }
141         TextField("Enter Doctor Phone Number", text: $doctorPhoneNumber)
142             .textFieldStyle(RoundedBorderTextFieldStyle())
143             .keyboardType(.numberPad)
144             .frame(width: UIScreen.main.bounds.width - 30, height: 50)
145         Button(action: {
146             if let url = URL(string: "tel://\(doctorPhoneNumber)") {
147                 UIApplication.shared.open(url, options: [:], completionHandler: nil)
148             }
149        }) {
150            Image(systemName: "phone.fill")
151                 .resizable()
152                 .frame(width: 20, height: 20)

```

```

153         .foregroundColor(.black)
154         .padding(.leading, 20)
155     Text("Doctor Phone Number")
156         .font(.body)
157         .bold()
158         .foregroundColor(.black)
159     }
160     .frame(width: UIScreen.main.bounds.width - 30, height: 50)
161     .background(colorScheme == .light ? Color.lightTeal : Color.darkTeal)
162     .cornerRadius(10)
163   }
164 }
165
166 var relevantDocuments: some View {
167   VStack(alignment: .center) {
168     HStack {
169       Text("Important Documents")
170         .font(.title3).bold()
171         .padding()
172       Spacer()
173     }
174     if #available(iOS 17.0, *) {
175       TipView(infoTip, arrowEdge: .bottom)
176         .tipCornerRadius(15)
177         .padding(.leading, 25)
178         .padding(.trailing, 25)
179     }
180     Button(action: {
181       isShowingImagePicker = true
182     }) {
183       Text("Select Images")
184         .font(.body)
185         .bold()
186         .foregroundColor(.black)
187     }
188     .frame(width: UIScreen.main.bounds.width - 30, height: 50)
189     .background(colorScheme == .light ? Color.lightTeal : Color.darkTeal)
190     .cornerRadius(10)
191     .padding()
192     .sheet(isPresented: $isShowingImagePicker) {
193       MultiImagePicker(selectedImages: $selectedImages)
194     }
195
196   ScrollView {
197     LazyVGrid(columns: Array(repeating: GridItem(), count: 3), spacing: 10) {
198       ForEach(selectedImages.indices, id: \.self) { index in
199         VStack {
200           NavigationLink(destination: Image(uiImage: selectedImages[index]))
201             .resizable()
202             .aspectRatio(contentMode: .fit)
203             .navigationBarTitle("", displayMode: .inline)
204         ) {
205           Image(uiImage: selectedImages[index])
206             .resizable()
207             .aspectRatio(contentMode: .fill)
208             .frame(width: UIScreen.main.bounds.width / 3 - 15, height:
209               UIScreen.main.bounds.width / 3 - 15)
210             .cornerRadius(10)
211         }
212         .buttonStyle(PlainButtonStyle())
213
214         Button(action: {
215           selectedImages.remove(at: index)
216         }) {
217           Image(systemName: "xmark.circle.fill")
218             .foregroundColor(.gray)
219             .background(colorScheme == .light ? Color.lightGrey : Color.
220               darkGrey)

```

```

219             .clipShape(Circle())
220             .padding(1)
221         }
222     }
223     alignment: .center)
224     }
225   }
226   .padding()
227 }
228 }
229 }
230
// MARK: Education Tab View
231 var body: some View {
232     VStack(alignment: .leading) {
233         Text("Resources")
234             .font(.largeTitle.bold())
235             .padding()
236         ScrollView {
237             if #available(iOS 17.0, *) {
238                 TipView(resourcesTip, arrowEdge: .none)
239                     .tipCornerRadius(15)
240                     .padding(.leading, 25)
241                     .padding(.trailing, 25)
242             }
243             articles
244             emergencyServicesInformation
245             doctorClinicInformation
246             fareInformation
247             relevantDocuments
248         }
249     }
250     .padding(.bottom, 20)
251     .task {
252         if #available(iOS 17.0, *) {
253             try? Tips.configure([
254                 .displayFrequency(.immediate),
255                 .datastoreLocation(.applicationDefault)
256             ])
257         }
258     }
259     .onAppear {
260         // Load saved images from UserDefaults
261         if let data = UserDefaults.standard.data(forKey: "selectedImages") {
262             do {
263                 storedImageData = try JSONDecoder().decode([ImageData].self, from: data)
264                 selectedImages = []
265                 // Convert stored data to UIImage
266                 for imageData in storedImageData {
267                     if let image = UIImage(data: imageData.imageData) {
268                         selectedImages.append(image)
269                     }
270                 }
271             } catch {
272                 print("Error decoding images: \(\error)")
273             }
274         }
275     }
276     .onDisappear {
277         // Save selected images to UserDefaults
278         do {
279             storedImageData.removeAll()
280             for image in selectedImages {
281                 if let imageData = image.jpegData(compressionQuality: 0.5) {
282                     storedImageData.append(ImageData(imageData: imageData))
283                 }
284             }
285         }

```

```

286         let encodedData = try JSONEncoder().encode(storedImageData)
287         UserDefaults.standard.set(encodedData, forKey: "selectedImages")
288     } catch {
289         print("Error encoding images: \(error)")
290     }
291 }
292 }
293 }
294
295 // MARK: Preview
296 #Preview {
297     EducationView()
298 }
```

## A.24 EducationViewContent.swift

```

1 // 
2 //  EducationViewContent.swift
3 //  SeniorDesign
4 //
5 //  Created by Maddie on 1/1/24.
6 //
7
8 import Foundation
9
10 struct EducationViewContent {
11     let anaphylaxisDisclaimer: AttributedString = try! AttributedString(markdown: "      **Disclaimer:**\nDisclaimer:** This information is provided for educational purposes only and should not be\nconstrued as medical advice. Always seek the advice of a qualified healthcare professional\nfor any medical concerns or questions regarding anaphylaxis or its treatment.")
12
13     let oitDisclaimer: AttributedString = try! AttributedString(markdown: "      **Disclaimer:**\nThis information is provided for educational purposes only and should not be construed as\nmedical advice. Always seek the advice of a qualified healthcare professional for any medical\ncarens or questions regarding Oral Immunotherapy or its treatment.")
14
15     let ccDisclaimer: AttributedString = try! AttributedString(markdown: "      **Disclaimer:**\nThis information is provided for educational purposes only and should not be construed as\nmedical advice. Always seek the advice of a qualified healthcare professional for any medical\ncarens or questions regarding cross contamination.")
16
17     let anaphylaxisTips: AttributedString = try! AttributedString(markdown: """
18         Anaphylaxis is a severe and potentially life-threatening allergic reaction that can occur\nsuddenly after exposure to an allergen. It involves multiple body systems and can lead to a\nrapid and severe allergic response.
19
20         Common signs and symptoms of anaphylaxis include:
21
22             Skin reactions, such as hives, itching, and flushed or pale skin.\n             Respiratory difficulties, including wheezing, shortness of breath, or difficulty\nbreathing.\n             Swelling, especially in the face, lips, throat, or tongue.\n             Gastrointestinal symptoms, such as abdominal pain, nausea, vomiting, or diarrhea.\n             Low blood pressure, which can lead to dizziness, fainting, or loss of consciousness.
23
24             Immediate treatment for anaphylaxis involves the use of epinephrine (adrenaline) through\nan auto-injector and seeking emergency medical assistance.
25
26             With OIT, because you're ingesting your allergen(s), the risk of anaphylaxis is higher\nthan usual. It is important to carry an epinephrine autoinjector with you at all times.
27
28             If you or someone experiences symptoms of anaphylaxis, call 911 immediately.
29
30             **Source:** [Mayo Clinic](https://www.mayoclinic.org/diseases-conditions/anaphylaxis/\nsymptoms-causes/syc-20351468)
31             """ , options: AttributedString.MarkdownParsingOptions(interpretedSyntax: .\ninlineOnlyPreservingWhitespace))
```

```

36
37 let oitTips: AttributedString = try! AttributedString(markdown: """
38     Oral Immunotherapy (OIT) involves ingesting small, controlled amounts of an allergen to
39     build tolerance over time. It is a treatment option for some individuals with food allergies.
40
41     Some tips and best practices for OIT include:
42
43         Avoiding exercise within 2 hours of taking a dose.
44         Avoiding alcohol consumption near dose times.
45         Taking doses with food to reduce the likelihood of stomach discomfort.
46         Being consistent with dose schedules to maintain progress.
47         Carrying an epinephrine auto-injector at all times.
48         Informing healthcare providers about ongoing OIT treatment before any medical
49         procedures or surgeries.
50
51     Remember, OIT should be performed under the guidance and supervision of trained
52     healthcare professionals. Follow their instructions closely and report any concerns or
53     adverse reactions immediately.
54
55     **Source:** [American Academy of Allergy Asthma & Immunology](https://www.aaaai.org/
56     conditions-and-treatments/library/allergy-library/oral-immunotherapy)
57     """ , options: AttributedString.MarkdownParsingOptions(interpretedSyntax: .
58     inlineOnlyPreservingWhitespace))
59
60 let crossContaminationTips: AttributedString = try! AttributedString(markdown: """
61     Cross-contamination poses a significant threat to individuals with food allergies, even
62     in the smallest traces. It occurs when allergens unintentionally transfer to safe foods or
63     surfaces during food preparation or handling. The risk lies in these tiny traces, which, if
64     ingested, can lead to severe allergic reactions.
65
66     The kitchen is a primary area susceptible to cross-contamination. Shared utensils,
67     cutting boards, and countertops can harbor allergens, posing risks to allergic individuals.
68     Similarly, public spaces like restaurants, schools, or workplaces carry potential cross-
69     contamination risks that demand vigilance.
70
71     At home, preventing cross-contamination involves meticulous organization. Designate
72     separate areas for allergen-free cooking and use dedicated utensils, cutting boards, and
73     cookware. Implement rigorous cleaning protocols to ensure surfaces, appliances, and hands
74     remain allergen-free during food preparation.
75
76     Restaurants and food establishments present unique challenges. Effective communication
77     with staff about food allergies is crucial. Asking questions about food preparation methods
78     and understanding restaurant protocols for preventing cross-contamination can significantly
79     mitigate risks.
80
81     When dealing with packaged foods, thorough label reading is essential. Check food labels
82     for allergen information and be aware of advisory labels like "may contain" to gauge cross-
83     contamination risks.
84
85     Navigating social settings and eating out requires assertiveness. Choose restaurants with
86     allergy-aware menus and communicate dietary needs. Similarly, in social gatherings,
87     communicate allergy concerns to hosts and inquire about ingredient information.
88
89     Educating others about the significance of cross-contamination prevention is pivotal.
90     Inform family and friends about the importance of allergen safety and advocate for
91     understanding and accommodation in public spaces.
92
93     Emergency preparedness is paramount. Always carry epinephrine auto-injectors and
94     understand their usage. Recognizing allergic reactions promptly and seeking medical
95     assistance when necessary is critical for managing allergic incidents.
96
97     In conclusion, proactive strategies to prevent cross-contamination are vital for
98     individuals with food allergies. By implementing these measures and raising awareness, we can
99     create safer environments and minimize allergic risks significantly.
100
101    **Source:** [Food Allergy Research & Education (FARE)](https://www.foodallergy.org/living-
102    -food-allergies/food-allergy-essentials/cross-contact-cross-contamination)
103    """ , options: AttributedString.MarkdownParsingOptions(interpretedSyntax: .

```

```
    inlineOnlyPreservingWhitespace))
75 }
```

## A.25 EmergencyServicesTip.swift

```
1 //////////////////////////////////////////////////////////////////
2 ////////////////////////////////////////////////////////////////// EmergencyServicesTip.swift
3 ////////////////////////////////////////////////////////////////// SeniorDesign
4 //////////////////////////////////////////////////////////////////
5 ////////////////////////////////////////////////////////////////// Created by Maddie on 3/7/24.
6 //////////////////////////////////////////////////////////////////
7
8 import Foundation
9 import TipKit
10
11 struct EmergencyServicesTip: Tip, Identifiable {
12     var id = UUID()
13     var title: Text {
14         Text("Emergency Assistance")
15     }
16
17     var message: Text? {
18         Text("In case of an anaphylactic reaction, don't hesitate to call 911 immediately. Time
19             is crucial. Stay calm, provide your location, and follow dispatcher instructions. Administer
20             epinephrine if available, and if trained to do so.")
21     }
22
23     var image: Image? {
24         Image(systemName: "phone.badge.checkmark")
25     }
26 }
```

## A.26 ImportantDocumentsTip.swift

```
1 //////////////////////////////////////////////////////////////////
2 ////////////////////////////////////////////////////////////////// ImportantDocumentsTip.swift
3 ////////////////////////////////////////////////////////////////// SeniorDesign
4 //////////////////////////////////////////////////////////////////
5 ////////////////////////////////////////////////////////////////// Created by Maddie on 3/7/24.
6 //////////////////////////////////////////////////////////////////
7
8 import Foundation
9 import TipKit
10
11 struct ImportantDocumentsTip: Tip, Identifiable {
12     var id = UUID()
13     var title: Text {
14         Text("Keep Your Information in One Place!")
15     }
16
17     var message: Text? {
18         Text("Add pictures of important documents, such as your Emergency Action Plan, below, so
19             you can easily find them later!")
20     }
21
22     var image: Image? {
23         Image(systemName: "doc.richtext")
24     }
25 }
```

## A.27 MultiImagePicker.swift

```
1 //////////////////////////////////////////////////////////////////
2 ////////////////////////////////////////////////////////////////// MultiImagePicker.swift
```

```

3 // SeniorDesign
4 //
5 // Created by Maddie on 3/15/24.
6 //
7
8 import Foundation
9 import SwiftUI
10 import PhotosUI
11
12 struct MultiImagePicker: UIViewControllerRepresentable {
13     @Binding var selectedImages: [UIImage]
14
15     func makeUIViewController(context: Context) -> PHPickerViewController {
16         var configuration = PHPickerConfiguration()
17         configuration.filter = .images
18         configuration.selectionLimit = 0 // Set to 0 to allow multiple selections
19         let picker = PHPickerViewController(configuration: configuration)
20         picker.delegate = context.coordinator
21         return picker
22     }
23
24     func updateUIViewController(_ uiViewController: PHPickerViewController, context: Context) {}
25
26     func makeCoordinator() -> Coordinator {
27         Coordinator(self)
28     }
29
30     class Coordinator: NSObject, PHPickerViewControllerDelegate {
31         let parent: MultiImagePicker
32
33         init(_ parent: MultiImagePicker) {
34             self.parent = parent
35         }
36
37         func picker(_ picker: PHPickerViewController, didFinishPicking results: [PHPickerResult]) {
38             parent.selectedImages.removeAll()
39
40             for result in results {
41                 if result.itemProvider.canLoadObject(ofClass: UIImage.self) {
42                     result.itemProvider.loadObject(ofClass: UIImage.self) { (image, error) in
43                         if let image = image as? UIImage {
44                             DispatchQueue.main.async { [self] in
45                                 parent.selectedImages.append(image)
46                             }
47                         }
48                     }
49                 }
50             }
51
52             DispatchQueue.main.async {
53                 picker.dismiss(animated: true)
54             }
55         }
56     }
57 }

```

## A.28 ResourcesTip.swift

```

1 //
2 // ResourcesTip.swift
3 // SeniorDesign
4 //
5 // Created by Maddie on 3/7/24.
6 //
7
8 import Foundation

```

```

9 import TipKit
10
11 struct ResourcesTip: Tip, Identifiable {
12     var id = UUID()
13     var title: Text {
14         Text("Stay Informed")
15     }
16
17     var message: Text? {
18         Text("Explore our curated articles, hotlines, and useful links for valuable insights into
19             food allergies and OIT. Empower yourself with knowledge to navigate your journey confidently
20             !")
21     }
22
23     var image: Image? {
24         Image(systemName: "books.vertical.circle")
25     }
26 }
```

## A.29 HealthKitManager.swift

```

1 // 
2 //  HealthKitManager.swift
3 //  SeniorDesign
4 //
5 //  Created by Maddie on 10/19/23.
6 //
7
8 import Foundation
9 import HealthKit
10 import CoreData
11
12 class HealthKitManager {
13     let healthStore = HKHealthStore()
14
15     // MARK: Symptoms Array
16     public let symptoms: [HKObjectType] = [
17         HKObjectType.categoryType(forIdentifier: .abdominalCramps)!,  

18         HKObjectType.categoryType(forIdentifier: .bloating)!,  

19         HKObjectType.categoryType(forIdentifier: .constipation)!,  

20         HKObjectType.categoryType(forIdentifier: .diarrhea)!,  

21         HKObjectType.categoryType(forIdentifier: .heartburn)!,  

22         HKObjectType.categoryType(forIdentifier: .nausea)!,  

23         HKObjectType.categoryType(forIdentifier: .vomiting)!,  

24         HKObjectType.categoryType(forIdentifier: .appetiteChanges)!,  

25         HKObjectType.categoryType(forIdentifier: .chills)!,  

26         HKObjectType.categoryType(forIdentifier: .dizziness)!,  

27         HKObjectType.categoryType(forIdentifier: .fainting)!,  

28         HKObjectType.categoryType(forIdentifier: .fatigue)!,  

29         HKObjectType.categoryType(forIdentifier: .fever)!,  

30         HKObjectType.categoryType(forIdentifier: .generalizedBodyAche)!,  

31         HKObjectType.categoryType(forIdentifier: .hotFlashes)!,  

32         HKObjectType.categoryType(forIdentifier: .chestTightnessOrPain)!,  

33         HKObjectType.categoryType(forIdentifier: .coughing)!,  

34         HKObjectType.categoryType(forIdentifier: .rapidPoundingOrFlutteringHeartbeat)!,  

35         HKObjectType.categoryType(forIdentifier: .shortnessOfBreath)!,  

36         HKObjectType.categoryType(forIdentifier: .skippedHeartbeat)!,  

37         HKObjectType.categoryType(forIdentifier: .wheezing)!,  

38         HKObjectType.categoryType(forIdentifier: .lowerBackPain)!,  

39         HKObjectType.categoryType(forIdentifier: .headache)!,  

40         HKObjectType.categoryType(forIdentifier: .memoryLapse)!,  

41         HKObjectType.categoryType(forIdentifier: .moodChanges)!,  

42         HKObjectType.categoryType(forIdentifier: .lossOfSmell)!,  

43         HKObjectType.categoryType(forIdentifier: .lossOfTaste)!,  

44         HKObjectType.categoryType(forIdentifier: .runnyNose)!,  

45         HKObjectType.categoryType(forIdentifier: .soreThroat)!,  

46         HKObjectType.categoryType(forIdentifier: .sinusCongestion)!,
```

```

47    HKObjectType.categoryType(forIdentifier: .acne)!,  

48    HKObjectType.categoryType(forIdentifier: .drySkin)!,  

49    HKObjectType.categoryType(forIdentifier: .hairLoss)!,  

50    HKObjectType.categoryType(forIdentifier: .nightSweats)!,  

51    HKObjectType.categoryType(forIdentifier: .sleepChanges)!,  

52    HKObjectType.categoryType(forIdentifier: .bladderIncontinence)!,  

53 ]  

54  

55 let symptomEmojis: [String: String] = [  

56     "HKCategoryTypeIdentifierAbdominalCramps": "痢疾",  

57     "HKCategoryTypeIdentifierBloating": "胀气",  

58     "HKCategoryTypeIdentifierConstipation": "便秘",  

59     "HKCategoryTypeIdentifierDiarrhea": "腹泻",  

60     "HKCategoryTypeIdentifierHeartburn": "胃痛",  

61     "HKCategoryTypeIdentifierNausea": "恶心",  

62     "HKCategoryTypeIdentifierVomiting": "呕吐",  

63     "HKCategoryTypeIdentifierAppetiteChanges": "食欲变化",  

64     "HKCategoryTypeIdentifierChills": "寒颤",  

65     "HKCategoryTypeIdentifierDizziness": "眩晕",  

66     "HKCategoryTypeIdentifierFainting": "晕厥",  

67     "HKCategoryTypeIdentifierFatigue": "疲倦",  

68     "HKCategoryTypeIdentifierFever": "发热",  

69     "HKCategoryTypeIdentifierGeneralizedBodyAche": "全身酸痛",  

70     "HKCategoryTypeIdentifierHotFlashes": "潮热",  

71     "HKCategoryTypeIdentifierChestTightnessOrPain": "胸闷或疼痛",  

72     "HKCategoryTypeIdentifierCoughing": "咳嗽",  

73     "HKCategoryTypeIdentifierRapidPoundingOrFlutteringHeartbeat": "心跳加速或颤动",  

74     "HKCategoryTypeIdentifierShortnessOfBreath": "呼吸困难",  

75     "HKCategoryTypeIdentifierSkippedHeartbeat": "心跳跳停",  

76     "HKCategoryTypeIdentifierWheezing": "喘息",  

77     "HKCategoryTypeIdentifierLowerBackPain": "腰痛",  

78     "HKCategoryTypeIdentifierHeadache": "头痛",  

79     "HKCategoryTypeIdentifierMemoryLapse": "记忆力减退",  

80     "HKCategoryTypeIdentifierMoodChanges": "情绪变化",  

81     "HKCategoryTypeIdentifierLossOfSmell": "嗅觉丧失",  

82     "HKCategoryTypeIdentifierLossOfTaste": "味觉丧失",  

83     "HKCategoryTypeIdentifierRunnyNose": "流鼻涕",  

84     "HKCategoryTypeIdentifierSoreThroat": "喉咙痛",  

85     "HKCategoryTypeIdentifierSinusCongestion": "鼻塞",  

86     "HKCategoryTypeIdentifierAcne": "痤疮",  

87     "HKCategoryTypeIdentifierDrySkin": "干燥皮肤",  

88     "HKCategoryTypeIdentifierHairLoss": "脱发",  

89     "HKCategoryTypeIdentifierNightSweats": "夜汗",  

90     "HKCategoryTypeIdentifierSleepChanges": "睡眠变化",  

91     "HKCategoryTypeIdentifierBladderIncontinence": "尿失禁"
92 ]  

93  

94 func symptomImageNeeded(_ symptom: String) -> Bool {  

95     if symptom == "HKCategoryTypeIdentifierConstipation"  

96         || symptom == "HKCategoryTypeIdentifierAbdominalCramps"  

97         || symptom == "HKCategoryTypeIdentifierBladderIncontinence"  

98         || symptom == "HKCategoryTypeIdentifierCoughing"  

99         || symptom == "HKCategoryTypeIdentifierSoreThroat"  

100        || symptom == "HKCategoryTypeIdentifierSkippedHeartbeat"  

101        || symptom == "HKCategoryTypeIdentifierAcne"  

102        || symptom == "HKCategoryTypeIdentifierLowerBackPain"  

103        || symptom == "HKCategoryTypeIdentifierChills"  

104        || symptom == "HKCategoryTypeIdentifierFainting"  

105        || symptom == "HKCategoryTypeIdentifierBloating"  

106        || symptom == "HKCategoryTypeIdentifierAppetiteChanges" {  

107            return true
108        }
109        return false
110    }
111  

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

```

```

115     (rawValue: symptom)) else {
116         return
117     }
118
119     let sample = HKCategorySample(type: symptomType, value: HKCategoryValue.notApplicable.
120     rawValue, start: date, end: date)
121     healthStore.save(sample) { success, error in
122         if let error = error {
123             print("Error saving symptom data: \(error.localizedDescription)")
124         } else {
125             print("Symptom data saved successfully")
126         }
127     }
128
129     func fetchSymptomsForDate(_ date: Date, completion: @escaping ([String]) -> Void) {
130         var symptomsData: [String] = []
131         for symptom in symptoms {
132             let predicate = HKQuery.predicateForSamples(withStart: date, end: date, options: .
133             strictStartDate)
134             let query = HKSampleQuery(sampleType: symptom, predicate: predicate, limit:
135             HKObjectQueryNoLimit, sortDescriptors: nil) { (query, samples, error) in
136                 if let error = error {
137                     print("Error fetching symptom data: \(error.localizedDescription)")
138                     completion([])
139                     return
140                 }
141
142                 if let samples = samples as? [HKCategorySample] {
143                     for sample in samples {
144                         symptomsData.append(sample.categoryType.identifier)
145                     }
146                 }
147                 healthStore.execute(query)
148             }
149             completion(symptomsData)
150         }
151
152     @MainActor func loadSymptomsForSelectedDate(selectedDate: Date, symptomDataManager:
153     SymptomDataManager, completion: @escaping () -> Void) {
154         fetchSymptomsForDate(selectedDate) { symptoms in
155             symptomDataManager.symptomRecords[selectedDate] = symptoms
156             completion()
157         }
158     }
159
160     // MARK: Health Request Functions
161     func setUpHealthRequest(healthStore: HKHealthStore, readSteps: @escaping () -> Void) {
162         if HKHealthStore.isHealthDataAvailable() {
163             healthStore.requestAuthorization(toShare: Set(symptoms), read: [HKObjectType.
164             quantityType(forIdentifier: HKQuantityTypeIdentifier.heartRate)!]) { success, error in
165                 if success {
166                     } else if error != nil {
167                         print(error ?? "Error")
168                     }
169                 }
170             }
171
172             @MainActor func saveSymptomsToCoreData(for date: Date, symptoms: [String]) {
173                 let symptomDataManager = SymptomDataManager()
174                 symptomDataManager.saveSymptoms(for: date, symptoms: symptoms)
175             }
176
177             @MainActor func fetchSymptomsFromCoreData(for date: Date) -> [String] {
178                 let symptomDataManager = SymptomDataManager()

```

```

177         return symptomDataManager.fetchSymptoms(for: date)
178     }
179 }
```

## A.30 HealthKitViewModel.swift

```

1 // HealthKitViewModel.swift
2 // SeniorDesign
3 //
4 // Created by Maddie on 10/19/23.
5 //
6 //
7
8 import Foundation
9 import HealthKit
10 import CoreData
11
12 class HealthKitViewModel: ObservableObject {
13     // MARK: Variables
14     @Published var isAuthorized = false
15     @Published var doseRecords: [Date: DoseRecord] = [:]
16     @Published var totalTakenDoses: Int = 0
17     @Published var totalSkippedDoses: Int = 0
18     private var healthStore = HKHealthStore()
19     private var healthKitManager = HealthKitManager()
20
21     // MARK: Initializer
22     init() {
23         changeAuthorizationStatus()
24     }
25
26     // MARK: Dose Functions
27     // CoreData container
28     public let persistentContainer: NSPersistentContainer = {
29         let container = NSPersistentContainer(name: "DoseDataModel")
30         let antihistamineDoseTransformer = AntihistamineDoseTransformer()
31         ValueTransformer.setValueTransformer(antihistamineDoseTransformer, forName:
32             NSValueTransformerName(rawValue: "AntihistamineDoseTransformer"))
33
34         let dosesTransformer = DosesTransformer()
35         ValueTransformer.setValueTransformer(dosesTransformer, forName: NSValueTransformerName(
36             rawValue: "DosesTransformer"))
37
38         let selectedAllergensTransformer = SelectedAllergensTransformer()
39         ValueTransformer.setValueTransformer(selectedAllergensTransformer, forName:
40             NSValueTransformerName(rawValue: "SelectedAllergensTransformer"))
41         container.loadPersistentStores(completionHandler: { (_, error) in
42             if let error = error {
43                 fatalError("Failed to load persistent store: \(error)")
44             }
45         })
46         return container
47     }()
48
49     // Function to save dose record using CoreData
50     func saveDoseRecord(_ doseRecord: DoseRecord) {
51         let context = persistentContainer.viewContext
52         guard let entity = NSEntityDescription.entity(forEntityName: "DoseRecordEntity", in:
53             context) else { return }
54
55         let doseRecordEntity = NSManagedObject(entity: entity, insertInto: context)
56         doseRecordEntity.setValue(doseRecord.date, forKey: "date")
57
58         do {
59             try context.save()
60         } catch {
61             print("Failed to save dose record: \(error.localizedDescription)")
62         }
63     }
64 }
```

```

58     }
59 }
60
61 // Function to fetch dose records using CoreData
62 func fetchDoseRecords() {
63     let context = persistentContainer.viewContext
64     let fetchRequest = NSFetchedResultsController<DoseRecord>(entityName: "DoseRecord")
65
66     do {
67         let fetchedRecords = try context.fetch(fetchRequest)
68         var doseRecordsByDate: [Date: DoseRecord] = [:]
69         for record in fetchedRecords {
70             doseRecordsByDate[record.date!] = record
71         }
72         self.doseRecords = doseRecordsByDate
73     } catch {
74         print("Failed to fetch dose records: \(error.localizedDescription)")
75     }
76 }
77
78
79 //MARK: - HealthKit Authorization Request Methods
80 func healthRequest() {
81     healthKitManager.setUpHealthRequest(healthStore: healthStore) {
82         self.changeAuthorizationStatus()
83     }
84 }
85
86 func changeAuthorizationStatus() {
87     guard let stepQtyType = HKObjectType.quantityType(forIdentifier: .stepCount) else {
88         return
89     }
90     let status = self.healthStore.authorizationStatus(for: stepQtyType)
91
92     switch status {
93     case .notDetermined:
94         isAuthorized = false
95     case .sharingDenied:
96         isAuthorized = false
97     case .sharingAuthorized:
98         DispatchQueue.main.async {
99             self.isAuthorized = true
100        }
101    @unknown default:
102        isAuthorized = false
103    }
104
105 @MainActor func saveSymptomsToCoreData(for date: Date, symptoms: [String]) {
106     let symptomDataManager = SymptomDataManager()
107     symptomDataManager.saveSymptoms(for: date, symptoms: symptoms)
108 }
109
110 @MainActor func fetchSymptomsFromCoreData(for date: Date) -> [String] {
111     let symptomDataManager = SymptomDataManager()
112     return symptomDataManager.fetchSymptoms(for: date)
113 }
114 }
```

## A.31 AboutYourDoseView.swift

```

1 /**
2  // AboutYourDoseView.swift
3  // SeniorDesign
4  //
5  // Created by Maddie on 10/20/23.
6  //
7 */
```

```

8 import SwiftUI
9
10 struct AboutYourDoseView: View {
11     // MARK: Variables
12     let allergenDoses: DoseRecord
13
14     @Environment(\.colorScheme) var colorScheme
15
16     // MARK: UI Elements
17     private var header: some View {
18         HStack {
19             Text("About Your Dose")
20                 .font(.headline)
21                 .foregroundColor(colorScheme == .light ? Color.black : Color.black)
22                 .padding(.top, 20)
23                 .padding(.leading, 20)
24                 .padding(.bottom, 5)
25             Spacer()
26         }
27     }
28
29     func sortingFunction(_ allergen: String) -> Int {
30         if allergen.hasPrefix("Zyrtec") || allergen.hasPrefix("Pepcid") || allergen.hasPrefix("Benadryl") {
31             return 1
32         } else {
33             return 0
34         }
35     }
36
37     private var dosesTaken: some View {
38         VStack(alignment: .leading) {
39             if let data = allergenDoses.doses as? Data,
40                 let doses = try? NSKeyedUnarchiver.unarchiveTopLevelObjectWithData(data) as? [
41                 String: String] {
42                 let sortedAllergens = doses.keys.sorted {
43                     if sortingFunction($0) == sortingFunction($1) {
44                         return $0 < $1 // Sort alphabetically within each group
45                     } else {
46                         return sortingFunction($0) < sortingFunction($1)
47                     }
48                 }
49                 ForEach(sortedAllergens, id: \.self) { allergen in
50                     if let dose = doses[allergen] {
51                         let formattedDose = dose.components(separatedBy: " ").last?
52                             .replacingOccurrences(of: ".0", with: "") ?? ""
53                             .trimmingCharacters(in: .whitespacesAndNewlines)
54                         HStack {
55                             Text("\(allergen)    \(formattedDose)")
56                                 .foregroundColor(colorScheme == .light ? Color.black : Color.
57 black)
58                                 .padding(.leading, 20)
59                                 .padding(.bottom, 5)
60                         Spacer()
61                     }
62                 }
63             }
64         }
65
66     private var timeOfDose: some View {
67         HStack {
68             Text(timeFormatter.string(from: allergenDoses.time!))
69                 .foregroundColor(colorScheme == .light ? Color.black : Color.black)
70                 .padding(.leading, 20)
71                 .padding(.bottom, 10)
72         Spacer()

```

```

73         }
74     }
75
76     // MARK: About Your Dose View
77     var body: some View {
78         ZStack {
79             VStack {
80                 header
81                 HStack {
82                     dosesTaken
83                         .frame(width: 230)
84                     Spacer()
85                 }
86                 Spacer()
87                 timeOfDose
88             }
89             .frame(minWidth: 0, maxWidth: .infinity, minHeight: 0, maxHeight: .infinity,
90 alignment: .topLeading)
91             .background(colorScheme == .light ? Color.lightTeal : Color.darkTeal)
92             .cornerRadius(20)
93             .padding(.leading, 20)
94             .padding(.trailing, 20)
95             HStack {
96                 Spacer()
97                 Image(systemName: "pills.fill")
98                     .font(.system(size: 90))
99                     .foregroundColor(colorScheme == .light ? Color.darkTeal : Color.darkerTeal)
100                    .padding(.trailing, 35)
101            }
102        }
103
104    // MARK: Functions
105    private let dateFormatter: DateFormatter = {
106        let formatter = DateFormatter()
107        formatter.dateStyle = .none
108        formatter.timeStyle = .short
109        return formatter
110    }()
111 }

```

## A.32 CalendarDayView.swift

```

1 /**
2 //  CalendarDayView.swift
3 //  SeniorDesign
4 //
5 //  Created by Maddie on 10/20/23.
6 //
7
8 import SwiftUI
9
10 struct CalendarDayView: View {
11     // MARK: Variables
12     @Binding var selectedDate: Date
13     @Binding var selectedIndex: Int?
14     @Environment(\.colorScheme) var colorScheme
15     var date: Date
16     var index: Int
17
18     private var isSelected: Bool {
19         selectedDate == date
20     }
21
22     private var isSelectable: Bool {
23         return date <= Date()
24     }

```

```

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
46             circle
47         } else {
48             return colorScheme == .light ? .lightTeal : .darkTeal // Default color for other
49             circles
50         }
51     }
52
53     private var circleTextColor: Color {
54         if !isSelectable {
55             return colorScheme == .light ? Color.mediumGrey : Color.black
56         }
57         if isSelected {
58             return colorScheme == .light ? Color.white : Color.black
59         } else if let selectedColorIndex = selectedColorIndex, selectedColorIndex == index {
60             return Color.black
61         } else {
62             return Color.black
63         }
64
65     // MARK: UI Elements
66     private var dayOfWeek: some View {
67         Text(dayText)
68             .font(.caption)
69             .bold()
70     }
71
72     private var dayCircle: some View {
73         Circle()
74             .frame(width: min(UIScreen.main.bounds.width / 10, 70), height: min(UIScreen.main.
75             bounds.width / 7, 70))
76             .overlay(
77                 Text(dateText)
78                     .font(.subheadline)
79                     .foregroundColor(circleTextColor)
80                     .bold()
81             )
82             .onTapGesture {
83                 if isSelectable {
84                     selectedDate = date
85                     selectedColorIndex = index
86                 }
87             }
88             .foregroundColor(circleColor)
89             .cornerRadius(8)
90             .disabled(!isSelectable)
91     }

```

```

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)
55                 }
56             }
57         }
58     }
59 }
```

```

55             .padding()
56         }
57         .frame(width: UIScreen.main.bounds.width-20, height: 130)
58         .background(Color.grey)
59         .opacity(0.7)
60         .padding(.top, 0)
61         .cornerRadius(15)
62     } else {
63         HStack {
64             Text("Allergens")
65                 .font(.headline)
66                 .padding()
67             Spacer()
68         }
69     }
70 }
71
72 var allergensSection: some View {
73     LazyVGrid(columns: [GridItem(.flexible()), GridItem(.flexible()), GridItem(.flexible())],
74     spacing: 20) {
75         ForEach(profileViewModel.profileData.allergens.sorted(), id: \.self) { allergen in
76             Button(action: {
77                 profileViewModel.toggleAllergenSelection(allergen)
78             }) {
79                 VStack {
80                     if profileViewModel.profileData.isNut(allergen) {
81                         Image("\u{1f3c1}")
82                             .resizable()
83                             .frame(width: 30, height: 30)
84                             .aspectRatio(contentMode: .fit)
85                             .padding(.bottom, 10)
86                     } else {
87                         Text(profileViewModel.profileData.allergenEmojiMap[allergen] ?? "")
88                             .padding(.bottom, 10)
89                             .font(.title)
90                     }
91                     Text(allergen)
92                         .bold()
93                         .foregroundColor(profileViewModel.isSelected(allergen) ? Color.white
94 : Color.black)
95                     }
96                     .frame(width: 80, height: 100)
97                     .padding()
98                     .background(profileViewModel.isSelected(allergen) ? colorScheme == .light ?
99 Color.darkTeal : Color.darkerTeal : colorScheme == .light ? Color.lightTeal : Color.darkTeal)
100                    .cornerRadius(20)
101                }
102            }
103            .padding(.leading, 20)
104            .padding(.trailing, 20)
105        }
106
107 var timeSectionHeader: some View {
108     HStack {
109         Text("Time of Dose")
110             .font(.headline)
111             .padding()
112             Spacer()
113     }
114
115 var timePicker: some View {
116     DatePicker("Dose Time", selection: $doseTime, displayedComponents: .hourAndMinute)
117     .datePickerStyle(CompactDatePickerStyle())
118     .font(.body.bold())
119     .labelsHidden()

```

```

120         .padding(.leading, 20)
121         .tint(.darkTeal)
122     }
123
124     var predoseHeader: some View {
125         HStack {
126             Text("Did you pre-dose any antihistamines before your oral immunotherapy dose?")
127                 .font(.headline)
128                 .padding()
129             Spacer()
130         }
131     }
132
133     var predoseSection: some View {
134         VStack {
135             ForEach(antihistamines.indices, id: \.self) { index in
136                 HStack {
137                     Text(antihistamines[index])
138                         .onTapGesture {
139                             hidden[index].toggle()
140                         }
141                         .padding()
142                     Spacer()
143                     if !hidden[index] {
144                         let selectedDose = Binding<String>(
145                             get: {
146                                 selectedDoses[antihistamines[index]] ?? ""
147                             },
148                             set: { newValue in
149                                 selectedDoses[antihistamines[index]] = newValue
150                             }
151                         )
152                         Picker("\(antihistamines[index])", selection: selectedDose) {
153                             ForEach(["2.5", "5", "10", "12.5", "20", "25", "30", "40", "50"], id:
154                             \.self) { dosage in
155                                 Text("\(dosage) mg").tag("\(dosage) mg")
156                             }
157                         .pickerStyle(DefaultPickerStyle())
158                         .tint(.darkTeal)
159                     }
160                     .padding(.horizontal, 20)
161                 }
162             Divider()
163                 .padding(.horizontal, 20)
164         }
165     }
166 }
167
168
169     var dosageSectionHeader: some View {
170         HStack {
171             if selectedAllergensArray.count > 0 {
172                 Text("Dosage for Allergens")
173                     .font(.headline)
174                     .padding()
175                 Spacer()
176             }
177         }
178     }
179 }
180
181     var dosageSection: some View {
182         ForEach(selectedAllergensArray.sorted(), id: \.self) { allergen in
183             HStack {
184                 Text(allergen)
185                     .padding(.leading, 20)
186                 Spacer()

```

```

187         Picker("", selection: Binding(
188             get: {
189                 selectedDoses[allergen] ?? ""
190             },
191             set: { newValue in
192                 selectedDoses[allergen] = newValue
193             }
194         )) {
195             let doseTypesForSelectedAllergens = doseViewModel.
196 doseTypesForSelectedAllergens(selectedAllergens: [allergen]).sorted()
197
198             ForEach(doseTypesForSelectedAllergens, id: \.self) { doseType in
199                 let formattedDose = doseType.components(separatedBy: " ").last?
200                     .replacingOccurrences(of: ".0", with: "") ?? ""
201                     .trimmingCharacters(in: .whitespacesAndNewlines)
202                     Text(formattedDose)
203             }
204             .pickerStyle(MenuPickerStyle())
205         }
206         .padding(.leading, 20)
207         .padding(.trailing, 20)
208         .padding(.bottom, 10)
209     }
210 }
211
212
213
214 var notesHeader: some View {
215     HStack {
216         Text("Notes")
217             .font(.headline)
218             .padding()
219         Spacer()
220     }
221 }
222
223 var notesSection: some View {
224     TextField("Enter any notes about your dose here.", text: $notes)
225         .textFieldStyle(DefaultTextFieldStyle())
226         .padding(.leading, 20)
227         .padding(.trailing, 20)
228 }
229
// MARK: Dosing Pop Up View
230 var body: some View {
231     NavigationStack {
232         ScrollView {
233             VStack(alignment: .leading) {
234                 timeSectionHeader
235                 timePicker
236
237                 allergensSectionHeader
238                 allergensSection
239
240                 dosageSectionHeader
241                 dosageSection
242
243                 predoseHeader
244                 predoseSection
245
246                 notesHeader
247                 notesSection
248                 Spacer()
249             }
250             .navigationBarTitle("Dose Information")
251             .navigationBarItems(trailing: Button("Done") {
252                 saveDoseInformation()

```

```

254             presentationMode.wrappedValue.dismiss()
255         })
256     }
257   }
258 }
259
// MARK: Functions
260 func saveDoseInformation() {
261   let context = healthKitViewModel.persistentContainer.viewContext
262   guard let entity = NSEntityDescription.entity(forEntityName: "DoseRecord", in: context)
263   else { return }
264   let newDoseRecord = DoseRecord(entity: entity, insertInto: context)
265   newDoseRecord.date = selectedDate
266   newDoseRecord.time = doseTime
267   newDoseRecord.notes = notes
268
269 // Convert arrays to NSData or archive them before assigning to Core Data entity
270 if let allergenData = try? NSKeyedArchiver.archivedData(withRootObject:
271 selectedAllergensArray, requiringSecureCoding: false) as NSData? {
272   newDoseRecord.selectedAllergens = allergenData
273 }
274
if let antihistamineData = try? NSKeyedArchiver.archivedData(withRootObject:
275 selectedAntihistamineDoses, requiringSecureCoding: false) as NSData? {
276   newDoseRecord.antihistamineDoses = antihistamineData
277 }
278
if let doseData = try? NSKeyedArchiver.archivedData(withRootObject: selectedDoses,
279 requiringSecureCoding: false) as NSData? {
280   newDoseRecord.doses = doseData
281 }
282
do {
283   try context.save()
284   _ = healthKitViewModel.fetchSymptomsFromCoreData(for: selectedDate)
285   healthKitViewModel.fetchDoseRecords()
286   doseViewModel.loadDoses()
287   print("DoseRecord saved successfully.")
288 } catch {
289   print("Failed to save DoseRecord: \(error.localizedDescription)")
290 }
291
// Dismiss the pop-up
292 presentationMode.wrappedValue.dismiss()
293 }
294
295
296
297 }
298
299 // MARK: Preview
300 #Preview {
301   DosingPopUp(selectedDate: Date())
302     .environmentObject(ProfileViewModel())
303     .environmentObject(HealthKitViewModel())
304 }

```

## A.34 HomeView.swift

```

1 //
2 //  HomeView.swift
3 //  SeniorDesign
4 //
5 //  Created by Maddie on 8/20/23.
6 //
7
8 import SwiftUI
9

```

```

10 struct HomeView: View {
11     // MARK: View Models
12     @EnvironmentObject var profileViewModel: ProfileViewModel
13     @EnvironmentObject var healthKitViewModel: HealthKitViewModel
14     @EnvironmentObject var doseViewModel: DoseViewModel
15     @EnvironmentObject var profileImageViewModel: ProfileModel
16
17     let healthKitManager = HealthKitManager()
18
19     // MARK: Variables
20     @State private var selectedDate = Date()
21     @State private var selectedIndex: Int?
22     @State private var weekView = true
23     @State private var logSymptoms = false
24     @State private var logDose = false
25     @State private var selectedSymptoms: Set<String> = []
26     @State private var dayOfTreatment = 1 // Initial value
27     @ObservedObject var symptomDataManager = SymptomDataManager()
28     var symptomsForSelectedDate: [String] {
29         return symptomDataManager.fetchSymptoms(for: selectedDate)
30     }
31
32     @Environment(\.colorScheme) var colorScheme
33
34     // MARK: UI Elements
35     var profileButton: some View {
36         NavigationLink(destination: SettingsView().environmentObject(profileViewModel)
37             .environmentObject(healthKitViewModel).environmentObject(profileImageViewModel)) {
38
39             CircularProfileImage(imageState: profileImageViewModel.imageState)
40                 .frame(width: 35, height: 35)
41                 .foregroundColor(.secondary)
42         }
43         .padding(.leading, 16)
44     }
45
46     var monthCalendarButton: some View {
47         HStack {
48             Text(monthHeader)
49                 .font(.body)
50                 .bold()
51             Image(systemName: "calendar")
52                 .resizable()
53                 .frame(width: 20, height: 20)
54                 .padding(.trailing, 20)
55         }
56         .onTapGesture {
57             weekView.toggle()
58         }
59     }
60
61     var header: some View {
62         HStack {
63             profileButton
64             Spacer()
65             monthCalendarButton
66         }
67     }
68
69     var weekScrollView: some View {
70         WeeklyScrollView
71             .padding(.bottom, -10)
72             .onAppear {
73                 selectedDate = todaysDate
74             }
75             .onChange(of: selectedDate) { _ in
76                 dayOfTreatment = calculateDayOfTreatment()
77             }

```

```

78     }
79
80     var weekCircleView: some View {
81         Circle()
82             .padding(.leading, 20)
83             .padding(.trailing, 20)
84             .padding(.bottom, 10)
85             .foregroundColor(colorScheme == .light ? .lightTeal : .darkTeal)
86             .overlay(
87                 weekCircleViewContent
88             )
89     }
90
91     var weekCircleViewContent: some View {
92         VStack {
93             Text("Day")
94                 .font(.body)
95                 .foregroundColor(colorScheme == .light ? Color.black : Color.black)
96                 .bold()
97                 .padding(.top, 30)
98             Text("\(dayOfTreatment)")
99                 .font(.system(size: 70))
100                .foregroundColor(colorScheme == .light ? Color.black : Color.black)
101                .bold()
102             Text("of Treatment")
103                 .font(.body)
104                 .foregroundColor(colorScheme == .light ? Color.black : Color.black)
105                 .bold()
106             Button(action: {
107                 logDose.toggle()
108             }) {
109                 Text("Log Dose      ")
110                     .font(.subheadline).bold()
111                     .foregroundColor(colorScheme == .light ? Color.white : Color.white)
112                     .padding()
113                     .background(colorScheme == .light ? Color.darkTeal : Color.darkerTeal)
114                     .cornerRadius(30)
115             }
116             .fullScreenCover(isPresented: $logDose) {
117                 DosingPopUp(selectedDate: selectedDate)
118                     .environmentObject(doseViewModel)
119             }
120             .padding()
121         }
122     }
123
124     var monthCalendarView: some View {
125         DatePicker("Select Date", selection: $selectedDate, in: ...Date(), displayedComponents: [.date])
126             .datePickerStyle(GraphicalDatePickerStyle())
127             .padding()
128             .tint(.darkTeal)
129     }
130
131     var logSymptomsButton: some View {
132         Button(action: {
133             logSymptoms.toggle()
134             selectedSymptoms = Set(symptomDataManager.fetchSymptoms(for: selectedDate))
135         }) {
136             VStack {
137                 Text("Log your Symptoms").bold()
138                     .foregroundColor(colorScheme == .light ? Color.black : Color.black)
139                     .padding(.bottom, 20)
140                 Image(systemName: "plus").bold()
141                     .foregroundColor(colorScheme == .light ? Color.black : Color.black)
142             }
143             .frame(width: 130, height: 150)
144             .background(colorScheme == .light ? Color.lightTeal : Color.darkTeal)

```

```

145     .cornerRadius(20)
146 }
147     .fullScreenCover(isPresented: $logSymptoms) {
148         NavigationStack {
149             SymptomsPopUp(selectedSymptoms: $selectedSymptoms, selectedDate:
150             selectedDate)
151                 .navigationBarItems(trailing: Button("Done") {
152                     logSymptoms = false
153                     saveSymptomsForSelectedDate()
154                 })
155                 .environmentObject(symptomDataManager) // Pass SymptomDataManager
156             here
157         }
158     }
159     var symptomsScrollView: some View {
160         HStack {
161             ScrollView(.horizontal, showsIndicators: false) {
162                 ScrollViewReader { proxy in
163                     HStack(spacing: 10) {
164                         logSymptomsButton
165                         ForEach(Array(symptomsForSelectedDate), id: \.self) { symptom in
166                             Button(action: {}) {
167                                 VStack {
168                                     Text(formatSymptomName(symptom))
169                                     .foregroundColor(colorScheme == .light ? Color.black :
170                                         Color.black)
171                                     .font(.system(size: 14)).bold()
172                                     .padding(.leading, 10)
173                                     .padding(.trailing, 10)
174                                     .padding(.bottom, 10)
175                                     if let emoji = healthKitManager.symptomEmojis[symptom]
176                                     if healthKitManager.symptomImageNeeded(symptom) {
177                                         Image("\(symptom)")
178                                         .resizable()
179                                         .frame(width: 30, height: 30)
180                                         .aspectRatio(contentMode: .fit)
181                                         .padding(.top, 10)
182                                     } else {
183                                         Text(emoji)
184                                         .font(.largeTitle)
185                                         .padding(.top, 5)
186                                     }
187                                 }
188                                 .frame(width: 130, height: 150)
189                                 .background(colorScheme == .light ? Color.lightTeal : Color.
190                                     darkTeal)
191                                 .cornerRadius(20)
192                             }
193                         .id(symptom)
194                     }
195                 }
196             }
197         }
198     .padding()
199 }
200
// MARK: Home Tab View
201 var body: some View {
202     NavigationStack {
203         VStack {
204             header
205             VStack {
206                 if weekView {
207                     weekScrollView

```

```

209         }
210         ScrollView {
211             if weekView {
212                 weekCircleView
213             } else {
214                 monthCalendarView
215             }
216             if let dose = healthKitViewModel.doseRecords[selectedDate] {
217                 if dose.doses != nil {
218                     AboutYourDoseView(allergenDoses: dose)
219                 }
220             }
221             symptomsScrollView
222         }
223     }
224 }
225 .onAppear(perform: {
226     profileViewModel.loadProfileData()
227     healthKitManager.loadSymptomsForSelectedDate(selectedDate: selectedDate,
228 symptomDataManager: symptomDataManager)
229     }
230     _ = healthKitViewModel.fetchSymptomsFromCoreData(for: selectedDate)
231     healthKitViewModel.fetchDoseRecords()
232     doseViewModel.loadDoses()
233     profileImageViewModel.loadProfileImage()
234 })
235
236     Spacer()
237 }
238
// MARK: Symptom Functions
239 private func saveSymptomsForSelectedDate() {
240     symptomDataManager.saveSymptoms(for: selectedDate, symptoms: Array(selectedSymptoms))
241 }
242
243 private func saveHKSymptomsForSelectedDate() {
244     for symptom in selectedSymptoms {
245         healthKitManager.saveSymptom(symptom, for: selectedDate)
246     }
247 }
248
249 private func formatSymptomName(_ identifier: String) -> String {
250     let trimmed = identifier.replacingOccurrences(of: "HKCategoryTypeIdentifier", with: "")
251     var formatted = ""
252     for char in trimmed {
253         if char.isUppercase {
254             formatted += " " + String(char)
255         } else {
256             formatted += String(char)
257         }
258     }
259     return formatted.trimmingCharacters(in: .whitespaces)
260 }
261
262
// MARK: Week View Functions
263 private var WeeklyScrollView: some View {
264     VStack {
265         VStack {
266             HStack(alignment: .center, spacing: 14) {
267                 ForEach(0..<7, id: \.self) { index in
268                     let day = Calendar.current.date(byAdding: .day, value: index, to:
269                         startOfWeek)!
270                     CalendarDayView(selectedDate: $selectedDate, selectedIndex:
271                         $selectedColorIndex, date: day, index: index)
272                 }
273             }

```

```

274     .padding()
275   }
276   .gesture(DragGesture(minimumDistance: 0, coordinateSpace: .local)
277     .onEnded { value in
278       let weekInterval: TimeInterval = 60 * 60 * 24 * 7
279       let startOfNextWeekDate = startOfNextWeek(from: startOfWeek)
280       if value.translation.width > 50 {
281         selectedDate = selectedDate.addingTimeInterval(-weekInterval)
282       } else if value.translation.width < -50 && startOfNextWeekDate < Date() {
283         selectedDate = selectedDate.addingTimeInterval(weekInterval)
284       }
285     }
286   )
287 }
288
289 private var startOfWeek: Date {
290   let calendar = Calendar.current
291   let components = calendar.dateComponents([.yearForWeekOfYear, .weekOfYear], from:
292 selectedDate)
293   var startComponents = DateComponents()
294   startComponents.yearForWeekOfYear = components.yearForWeekOfYear
295   startComponents.weekOfYear = components.weekOfYear
296   startComponents.weekday = 1 // Sunday
297   return calendar.date(from: startComponents)!
298 }
299
300 private var todaysDate: Date {
301   let calendar = Calendar.current
302   let today = calendar.startOfDay(for: Date())
303   return today
304 }
305
306 private func startOfNextWeek(from date: Date) -> Date {
307   let calendar = Calendar.current
308   var startComponents = calendar.dateComponents([.yearForWeekOfYear, .weekOfYear, .weekday
309 ], from: date)
310   if let currentWeekStartDate = calendar.date(from: startComponents) {
311     startComponents.weekOfYear? += 1 // Increment to the next week
312     return calendar.date(from: startComponents) ?? currentWeekStartDate
313   }
314   return date
315 }
316
317 private var monthHeader: String {
318   let dateFormatter = DateFormatter()
319   dateFormatter.dateFormat = "MMMM yyyy"
320   return dateFormatter.string(from: startOfWeek)
321 }
322
323 // MARK: Day of Treatment Functions
324 func calculateDayOfTreatment() -> Int {
325   if let firstSymptomDate = symptomDataManager.symptomRecords.keys.sorted().first,
326     let firstDoseDate = healthKitViewModel.doseRecords.keys.sorted().first {
327
328     let firstDates = [firstSymptomDate, firstDoseDate]
329     let minDate = firstDates.min()!
330
331     let calendar = Calendar.current
332     let components = calendar.dateComponents([.day], from: minDate, to: selectedDate)
333     return (components.day ?? 0) + 1 // Adding 1 to start from Day 1
334   }
335
336   if let firstSymptomDate = symptomDataManager.symptomRecords.keys.sorted().first {
337     let calendar = Calendar.current
338     let components = calendar.dateComponents([.day], from: firstSymptomDate, to:
339 selectedDate)
340     return (components.day ?? 0) + 1 // Adding 1 to start from Day 1
341   }

```

```

339
340     if let firstDoseDate = healthKitViewModel.doseRecords.keys.sorted().first {
341         let calendar = Calendar.current
342         let components = calendar.dateComponents([.day], from: firstDoseDate, to:
343             selectedDate)
344         return (components.day ?? 0) + 1 // Adding 1 to start from Day 1
345     }
346
347     return 1
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             .scaledToFit()
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
```

```

20 @State private var otherAllergenName: String = ""
21 @State private var otherAllergenEmoji: String = ""
22
23 @StateObject var appState = AppState()
24
25 @Environment(\.colorScheme) var colorScheme
26
27 // MARK: UI Elements
28 var personalInformationSection: some View {
29     Section(header: Text("Personal Information").font(.title3).bold()) {
30         HStack {
31             Text("Photo").bold()
32                 .padding(.leading, 5)
33             Spacer()
34             EditableCircularProfileImage(viewModel: profileImageViewModel)
35                 .padding(.trailing, 5)
36         }
37         HStack {
38             Text("Name")
39                 .font(.body.bold())
40                 .padding(.leading, 5)
41             Spacer()
42             TextField("Name", text: $profileViewModel.profileData.name)
43                 .textFieldStyle(RoundedBorderTextFieldStyle())
44                 .padding(.trailing, 5)
45                 .onChange(of: profileViewModel.profileData.name) { _ in
46                     profileViewModel.saveProfileData()
47                 }
48                 .frame(width: 120)
49         }
50         DatePicker("Birthdate", selection: $profileViewModel.profileData.dateOfBirth,
51 displayedComponents: .date)
52             .datePickerStyle(CompactDatePickerStyle())
53             .padding(.leading, 5)
54             .padding(.trailing, 5)
55             .font(.body.bold())
56             .onChange(of: profileViewModel.profileData.dateOfBirth) { _ in
57                 profileViewModel.saveProfileData()
58             }
59     }
60     .padding(.top, 10)
61 }
62
63 var allergensSection: some View {
64     Section(header: HStack {
65         SectionHeaderView(title: "Allergens")
66         Spacer()
67         Button(action: {
68             profileViewModel.addNewAllergen()
69         }) {
70             HStack {
71                 Image(systemName: "plus.circle")
72                     .foregroundColor(.darkTeal)
73             }
74         }
75         .padding(.trailing, 20)
76     }) {
77         ForEach(0..<max(1, profileViewModel.numAllergens), id: \.self) { index in
78             if index < profileViewModel.numAllergens {
79                 HStack {
80                     VStack {
81                         Picker("Select an Allergen", selection: $profileViewModel.profileData
82 .allergens[index])
83                         .ForEach($profileViewModel.profileData.commonAllergens, id: \.self
84 ) { allergen in
85                             Text(allergen.wrappedValue).tag(allergen.wrappedValue)
86                         }
87                     }
88                 }
89             }
90         }
91     }
92 }
```

```

85             Text("Other").tag("Other")
86         }
87         .tint(colorScheme == .dark ? .white : .black)
88         .pickerStyle(DefaultPickerStyle())
89         .padding(.bottom, 10)
90         .onChange(of: profileViewModel.profileData.allergens[index]) {
91             newValue in
92                 if newValue == "Other" {
93                     isAddingOtherAllergen = true
94                 } else if !profileViewModel.profileData.allergens.contains(
95                     newValue) && newValue != "Select an Allergen" && newValue != "" && newValue != "Other"{
96                     profileViewModel.profileData.allergens.append(newValue)
97                     profileViewModel.profileData.allergens.removeAll { $0 == "Other" }
98                     profileViewModel.profileData.commonAllergens.sort()
99                     profileViewModel.saveProfileData()
100                }
101            }
102            .onAppear {
103                if profileViewModel.profileData.allergens[index].isEmpty,
104                    let firstCommonAllergen = $profileViewModel.profileData.
105                    commonAllergens.first {
106                        profileViewModel.profileData.allergens[index] =
107                            firstCommonAllergen.wrappedValue
108                    }
109            }
110            Spacer()
111        }
112        VStack {
113            Button(action: {
114                profileViewModel.removeSelectedAllergen(at: index)
115            }) {
116                Image(systemName: "minus.circle").foregroundColor(.red)
117                .padding(.top, 7)
118            }
119        }
120    }
121    .padding(.top, 10)
122    .alert("Enter Other Allergen", isPresented: $isAddingOtherAllergen) {
123        TextField("Other Allergen", text: $otherAllergenName)
124        Button("OK", action: submitOtherAllergen)
125    }
126    .alert("Enter Emoji for Your Allergen", isPresented: $isAddingOtherAllergenEmoji) {
127        TextField("Emoji", text: $otherAllergenEmoji)
128        Button("OK", action: submitOtherEmoji)
129    }
130}
131
132 func submitOtherAllergen() {
133     if !profileViewModel.profileData.allergens.contains(otherAllergenName) {
134         profileViewModel.profileData.allergens.append(otherAllergenName)
135         profileViewModel.profileData.commonAllergens.append(otherAllergenName)
136         profileViewModel.profileData.allergens.removeAll { $0 == "Other" }
137         profileViewModel.profileData.commonAllergens.sort()
138         profileViewModel.saveProfileData()
139     }
140     isAddingOtherAllergen = false
141     isAddingOtherAllergenEmoji = true
142 }
143
144 func submitOtherEmoji() {
145     profileViewModel.profileData.allergenEmojiMap[otherAllergenName] = otherAllergenEmoji
146     profileViewModel.profileData.commonAllergens.sort()
147 }
```

```

148     profileViewModel.saveProfileData()
149     isAddingOtherAllergenEmoji = false
150 }
151
152 var shareDataWithHealthToggle: some View {
153     Toggle("Share Data with Apple Health", isOn: $profileViewModel.profileData.
154     shareDataWithAppleHealth)
155         .toggleStyle(SwitchToggleStyle(tint: .darkTeal))
156         .onChange(of: profileViewModel.profileData.shareDataWithAppleHealth) { newValue in
157             healthKitViewModel.healthRequest()
158             profileViewModel.saveProfileData()
159         }
160 }
161
162 var protectDataWithFaceID: some View {
163     Toggle("Enable Passcode, TouchID, or FaceID", isOn: $profileViewModel.profileData.
164     useFaceID)
165         .toggleStyle(SwitchToggleStyle(tint: .darkTeal))
166         .onChange(of: profileViewModel.profileData.useFaceID) { newValue in
167             profileViewModel.saveProfileData()
168         }
169 }
170
171 var preferencesSection: some View {
172     Section(header: Text("Preferences").font(.title3).bold()) {
173         shareDataWithHealthToggle
174         protectDataWithFaceID
175     }.padding(.top, 10)
176 }
177
178 var header: some View {
179     HStack {
180         Text("Profile")
181             .font(.largeTitle.bold())
182             .padding(.leading, 20)
183             .padding(.top, 10)
184         Spacer()
185     }
186 }
187
188 // MARK: Settings View
189 var body: some View {
190     VStack {
191         if appState.hasAppBeenOpenedBefore {
192             header
193         }
194         ScrollView {
195             VStack(alignment: .leading) {
196                 VStack(alignment: .leading) {
197                     personalInformationSection
198                     Divider()
199                         .padding(.leading, 20)
200                         .padding(.trailing, 20)
201                         .padding(.top, 20)
202                     allergensSection
203                     Spacer()
204                     Divider()
205                         .padding(.leading, 20)
206                         .padding(.trailing, 20)
207                     preferencesSection
208                     Spacer()
209                     Divider()
210                         .padding(20)
211                     if appState.hasAppBeenOpenedBefore {
212                         HStack {
213                             Spacer()
214                             Text("Log Out")
215                             .foregroundColor(.red)

```

```

214 //          .onTapGesture {
215 //              // TODO: Add logic to log out
216 //          }
217 //      }
218 //      Spacer()
219 //      Spacer()
220 //  }
221 //  .padding()
222 //}
223 }
224 .onDisappear {
225     profileViewModel.profileData.allergens = profileViewModel.profileData.allergens.
226     filter { !$0.isEmpty }
227     profileViewModel.saveProfileData()
228     profileImageViewModel.saveProfileImage()
229 }
230 .onAppear(perform: {
231     profileViewModel.loadProfileData()
232     profileImageViewModel.loadProfileImage()
233     doseViewModel.loadDoses()
234     profileViewModel.profileData.commonAllergens.sort()
235 })
236 }
237 }
238 }
239
240 // MARK: Preview
241 #Preview {
242     SettingsView()
243     .environmentObject(HealthKitViewModel())
244     .environmentObject(ProfileViewModel())
245 }

```

## A.38 SymptomsPopUp.swift

```

1 import HealthKit
2 import SwiftUI
3
4 struct SymptomsPopUp: View {
5     // MARK: Variables
6     @EnvironmentObject var symptomDataManager: SymptomDataManager
7     @Binding var selectedSymptoms: Set<String>
8     let selectedDate: Date
9     @State var contactDoctor = false
10    let healthKitManager = HealthKitManager()
11    let columns: [GridItem] = [GridItem(.flexible()), GridItem(.flexible()), GridItem(.flexible())]
12
13    @Environment(\.presentationMode) var presentationMode
14    @Environment(\.colorScheme) var colorScheme
15
16    // MARK: Initializer
17    init(selectedSymptoms: Binding<Set<String>>, selectedDate: Date) {
18        self._selectedSymptoms = selectedSymptoms
19        self.selectedDate = selectedDate
20    }
21
22    private func foregroundColor(for symptom: HKSampleType) -> Color {
23        if selectedSymptoms.contains(symptom.identifier.description) {
24            return colorScheme == .light ? Color.white : Color.white
25        } else {
26            return colorScheme == .light ? Color.black : Color.black
27        }
28    }
29
30    private func backgroundColor(for symptom: HKSampleType) -> Color {

```

```

31     if selectedSymptoms.contains(symptom.identifier.description) {
32         return colorScheme == .light ? Color.darkTeal : Color.darkerTeal
33     } else {
34         return colorScheme == .light ? Color.lightTeal : Color.darkTeal
35     }
36 }
37
38 // MARK: Symptoms Pop Up View
39 var body: some View {
40     ScrollView {
41         LazyVGrid(columns: columns, spacing: 10) {
42             ForEach(healthKitManager.symptoms, id: \.self) { symptom in
43                 Button(action: {
44                     toggleSymptom(symptom.identifier.description)
45                 }) {
46                     VStack {
47                         Text(formatSymptomName(symptom.identifier.description))
48                             .padding(.leading, 10)
49                             .padding(.trailing, 10)
50                             .font(.system(size: 14)).bold()
51                             .foregroundColor(foregroundColor(for: symptom))
52                         if let emoji = healthKitManager.symptomEmojis[symptom.identifier.
53                             description] {
54                             if healthKitManager.symptomImageNeeded(symptom.identifier.
55                                 description) {
56                                 Image("\(symptom.identifier.description)")
57                                     .resizable()
58                                     .frame(width: 30, height: 30)
59                                     .aspectRatio(contentMode: .fit)
60                                     .padding(.top, 10)
61                             } else {
62                                 Text(emoji)
63                                     .font(.largeTitle).bold()
64                                     .padding(.top, 5)
65                             }
66                         }
67                         .frame(width: UIScreen.main.bounds.width/3 - 10, height: 150)
68                         .background(backgroundColor(for: symptom))
69                         .cornerRadius(20)
70                     }
71                 }
72             }
73             .alert("Have you contacted your doctor?      ", isPresented: $contactDoctor) {
74                 Text("You have selected symptoms that may require immediate attention. Please contact
75                     your doctor or call 911.")
76                 Button("OK", action: {contactDoctor = false})
77             }
78             .onDisappear(perform: saveSymptomsForSelectedDate)
79             .padding(10)
80         }
81
82 // MARK: Functions
83 private func formatSymptomName(_ identifier: String) -> String {
84     let trimmed = identifier.replacingOccurrences(of: "HKCategoryTypeIdentifier", with: "")
85     var formatted = ""
86     for char in trimmed {
87         if char.isUppercase {
88             formatted += " " + String(char)
89         } else {
90             formatted += String(char)
91         }
92     }
93     return formatted.trimmingCharacters(in: .whitespaces)
94 }
95
96 private func saveSymptomsForSelectedDate() {

```

```

96     symptomDataManager.saveSymptoms(for: selectedDate, symptoms: Array(selectedSymptoms))
97
98     let worrisomeSymptoms = ["HKCategoryTypeIdentifierFever", "HKCategoryTypeIdentifierVomiting", "HKCategoryTypeIdentifierChestTightnessOrPain", "HKCategoryTypeIdentifierChills", "HKCategoryTypeIdentifierRapidPoundingOrFlutteringHeartbeat", "HKCategoryTypeIdentifierShortnessOfBreath", "HKCategoryTypeIdentifierMemoryLapse", "HKCategoryTypeIdentifierMoodChanges", "HKCategoryTypeIdentifierFainting", "HKCategoryTypeIdentifierDizziness", "HKCategoryTypeIdentifierCoughing"]
99
100    let hasWorrisomeSymptoms = selectedSymptoms.contains { worrisomeSymptoms.contains($0) }
101
102    if hasWorrisomeSymptoms {
103        contactDoctor = true
104    }
105
106
107    private func toggleSymptom(_ identifier: String) {
108        if selectedSymptoms.contains(identifier) {
109            selectedSymptoms.remove(identifier)
110        } else {
111            selectedSymptoms.insert(identifier)
112        }
113    }
114 }

```

## A.39 DosesForMonthView.swift

```

1 // DosesForMonthView.swift
2 // SeniorDesign
3 //
4 // Created by Maddie on 3/7/24.
5 //
6 import SwiftUI
7 import CareKitUI
8
9 struct DosesForMonthView: View {
10     @ObservedObject var healthKitViewModel: HealthKitViewModel
11     @Environment(\.colorScheme) var colorScheme
12
13     private var allDates: [Date] {
14         let calendar = Calendar.current
15         let currentDate = Date()
16
17         guard let startDate = calendar.date(byAdding: .month, value: -1, to: currentDate) else {
18             fatalError("Error calculating start date of the past month")
19         }
20
21         var dates: [Date] = []
22         var currentDateIter = startDate
23         while currentDateIter <= currentDate {
24             dates.append(currentDateIter)
25             currentDateIter = calendar.date(byAdding: .day, value: 1, to: currentDateIter)!
26         }
27
28         return dates
29     }
30
31     var body: some View {
32         return VStack {
33             DosesForMonthLinePlotView(allDates: allDates, healthKitViewModel: healthKitViewModel,
34             colorScheme: colorScheme)
35         }
36     }
37 }
38

```

```

39 struct DosesForMonthLinePlotView: UIViewRepresentable {
40     let allDates: [Date]
41     let healthKitViewModel: HealthKitViewModel
42     let colorScheme: ColorScheme
43
44     func makeUIView(context: Context) -> OCKCartesianChartView {
45         let chartView = OCKCartesianChartView(type: .line)
46         chartView.headerView.titleLabel.text = "Doses This Month"
47
48         var dataSeries = OCKDataSeries(values: allDates.map { date in
49             let doseCount = Double(healthKitViewModel.doseRecords.values.filter { record in
50                 if let doseDate = record.date {
51                     let components = Calendar.current.dateComponents([.day, .month, .year], from:
52                         doseDate)
53                     return components == Calendar.current.dateComponents([.day, .month, .year],
54                         from: date)
55                 }
56                 return false
57             }.count)
58             return CGFloat(doseCount)
59         }, title: "Dose Taken", color: colorScheme == .light ? UIColor(.darkTeal) : UIColor(.lightTeal))
60         dataSeries.size = 3
61
62         let (dosesTaken, dosesSkipped) = countDoses()
63         chartView.headerView.detailLabel.text = "You took \(dosesTaken) doses and skipped \(dosesSkipped) doses this month."
64         chartView.graphView.dataSeries = [dataSeries]
65         chartView.graphView.yMaximum = 1
66         chartView.graphView.yMinimum = 0
67         return chartView
68     }
69
70     func updateUIView(_ uiView: OCKCartesianChartView, context: Context) {
71         // Update the view if needed
72     }
73
74     private func countDoses() -> (Int, Int) {
75         var dosesTaken = 0
76         var dosesSkipped = 0
77
78         for date in allDates {
79             if healthKitViewModel.doseRecords.values.contains(where: { record in
80                 if let doseDate = record.date {
81                     let components = Calendar.current.dateComponents([.day, .month, .year], from:
82                         doseDate)
83                     return components == Calendar.current.dateComponents([.day, .month, .year],
84                         from: date)
85                 }
86                 return false
87             }) {
88                 dosesTaken += 1
89             } else {
90                 dosesSkipped += 1
91             }
92         }
93
94         return (dosesTaken, dosesSkipped)
95     }
96
97     struct DosesForMonthView_Previews: PreviewProvider {
98         static var previews: some View {
99             DosesForMonthView(healthKitViewModel: HealthKitViewModel())
100        }
101    }
102}

```

## A.40 InsightsTip.swift

```
1 //  
2 //  InsightsTip.swift  
3 //  SeniorDesign  
4 //  
5 //  Created by Maddie on 3/7/24.  
6 //  
7  
8 import Foundation  
9 import TipKit  
10  
11 struct InsightsTip: Tip, Identifiable {  
12     var id = UUID()  
13     var title: Text {  
14         Text("Unlock Your Health Insights")  
15     }  
16  
17     var message: Text? {  
18         Text("Discover personalized stats and insightful graphs based on your symptoms and doses.  
19             Track your progress, gain valuable health insights, and make informed decisions on your  
20             journey to wellness.")  
21     }  
22  
23     var image: Image? {  
24         Image(systemName: "waveform.badge.magnifyingglass")  
25     }  
26 }
```

## A.41 InsightsView.swift

```
1 //  
2 //  InsightsView.swift  
3 //  SeniorDesign  
4 //  
5 //  Created by Maddie on 10/18/23.  
6 //  
7  
8 import SwiftUI  
9 import CoreData  
10 import TipKit  
11  
12 struct InsightsView: View {  
13     // MARK: View Model  
14     @ObservedObject var healthKitViewModel: HealthKitViewModel  
15     @ObservedObject var symptomDataManager = SymptomDataManager()  
16     @State private var slicesWithLabels: [(Double, Color, String)] = []  
17     @Environment(\.colorScheme) var colorScheme  
18  
19     var header: some View {  
20         HStack {  
21             Text("Insights")  
22                 .font(.largeTitle.bold())  
23                 .padding()  
24             Spacer()  
25         }  
26     }  
27  
28     var insightsTip = InsightsTip()  
29  
30     // MARK: Insights View  
31     var body: some View {  
32         VStack {  
33             header  
34             ScrollView {  
35                 if #available(iOS 17.0, *) {
```



```

100         .cornerRadius(20)
101         .padding(.leading, 20)
102         .padding(.trailing, 20)
103         .padding(.bottom, 20)
104         .onAppear {
105             self.slicesWithLabels = calculateSlices(symptoms: symptomDataManager.
106             fetchAllSymptomsWithoutRefresh())
107         }
108     }
109 }
110 .onAppear(perform: {
111     healthKitViewModel.fetchDoseRecords()
112 })
113 .task {
114     if #available(iOS 17.0, *) {
115         try? Tips.configure([
116             .displayFrequency(.immediate),
117             .datastoreLocation(.applicationDefault)
118         ])
119     }
120 }
121 }
122
123 private func calculateSlices(symptoms: [String]) -> [(Double, Color, String)] {
124     let groupedSymptoms = Dictionary(grouping: symptoms, by: { $0 })
125     var calculatedSlices: [(Double, Color, String)] = []
126
127     for (_, value) in groupedSymptoms.enumerated() {
128         let formattedKey = key.splitCamelCase()
129         let slice = (Double(value.count) / Double(symptoms.count), Color.mutedRandom,
130         formattedKey)
131         calculatedSlices.append(slice)
132     }
133
134     return calculatedSlices
135 }
136
137 func calculateDayOfTreatment() -> Int {
138     if let firstSymptomDate = symptomDataManager.symptomRecords.keys.sorted().first,
139         let firstDoseDate = healthKitViewModel.doseRecords.keys.sorted().first {
140
141         let firstDates = [firstSymptomDate, firstDoseDate]
142         let minDate = firstDates.min()!
143
144         let calendar = Calendar.current
145         let components = calendar.dateComponents([.day], from: minDate, to: Date())
146         return (components.day ?? 0) + 1
147     }
148
149     if let firstSymptomDate = symptomDataManager.symptomRecords.keys.sorted().first {
150         let calendar = Calendar.current
151         let components = calendar.dateComponents([.day], from: firstSymptomDate, to: Date())
152         return (components.day ?? 0) + 1
153     }
154
155     if let firstDoseDate = healthKitViewModel.doseRecords.keys.sorted().first {
156         let calendar = Calendar.current
157         let components = calendar.dateComponents([.day], from: firstDoseDate, to: Date())
158         return (components.day ?? 0) + 1
159     }
160
161     return 1
162 }

```

## A.42 LastReactionInsight.swift

```
1 //  
2 //  LastReactionInsight.swift  
3 //  SeniorDesign  
4 //  
5 //  Created by Maddie on 3/7/24.  
6 //  
7  
8 import Foundation  
9 import SwiftUI  
10  
11 struct LastReactionInsight: View {  
12     @ObservedObject var symptomDataManager: SymptomDataManager  
13     @Environment(\.colorScheme) var colorScheme  
14  
15     var body: some View {  
16         VStack {  
17             if let lastReactionDate = symptomDataManager.lastReactionDate() {  
18                 let daysSinceLastReaction = Calendar.current.dateComponents([.day], from:  
lastReactionDate, to: Date()).day ?? 0  
19  
20                 HStack(alignment: .center) {  
21                     Spacer()  
22                     if daysSinceLastReaction <= 2 {  
23                         Text("Don't worry, it's okay. Things will improve. It's been \(  
daysSinceLastReaction) days since your last reaction.").bold()  
24                             .foregroundColor(colorScheme == .light ? .black : .black)  
25                     } else {  
26                         Text("Congratulations! It's been \(daysSinceLastReaction) days since your  
last reaction!").bold()  
27                             .foregroundColor(colorScheme == .light ? .black : .black)  
28                     }  
29                     Spacer()  
30                 }  
31                 .font(.subheadline)  
32                 .padding()  
33             }  
34         }  
35         .frame(minWidth: 0, maxWidth: .infinity, minHeight: 0, maxHeight: .infinity, alignment: .  
topLeading)  
36         .background(colorScheme == .light ? Color.lightBlue : Color.darkTeal)  
37         .cornerRadius(20)  
38     }  
39 }
```

## A.43 Legend.swift

```
1 //  
2 //  Legend.swift  
3 //  SeniorDesign  
4 //  
5 //  Created by Maddie on 3/7/24.  
6 //  
7  
8 import Foundation  
9 import SwiftUI  
10  
11 struct Legend: View {  
12     let slicesWithLabels: [(Double, Color, String)]  
13  
14     var body: some View {  
15         VStack(alignment: .leading) {  
16             let columns = 2  
17             let rows = (slicesWithLabels.count + columns - 1) / columns  
18         }  
19     }  
20 }
```

```

19     ForEach(0..<rows, id: \.self) { row in
20         HStack(spacing: 0) {
21             Spacer()
22             ForEach(0..<columns, id: \.self) { column in
23                 let index = row * columns + column
24                 if index < slicesWithLabels.count {
25                     let symptomName = slicesWithLabels[index].2
26                     LegendItem(color: slicesWithLabels[index].1, label: "\u{symptomName}")
27                         .frame(alignment: .leading)
28                 }
29             }
30         }
31     }
32     .frame(alignment: .leading)
33 }
34 }
35 .padding(.top)
36 .frame(maxWidth: 300)
37 }
38 }
```

## A.44 LegendItem.swift

```

1 /**
2 //  LegendItem.swift
3 //  SeniorDesign
4 /**
5 //  Created by Maddie on 3/7/24.
6 /**
7
8 import Foundation
9 import SwiftUI
10
11 struct LegendItem: View {
12     let color: Color
13     let label: String
14
15     @Environment(\.colorScheme) var colorScheme
16
17     var body: some View {
18         HStack {
19             Circle()
20                 .fill(color)
21                 .frame(width: 10, height: 10)
22             Text(label)
23                 .font(.system(size: 14)).bold()
24                 .foregroundColor(colorScheme == .light ? .black : .white)
25         }
26         .padding(.horizontal)
27     }
28 }
```

## A.45 SymptomsOverTimeChartView.swift

```

1 /**
2 //  SymptomsOverTimeChartView.swift
3 //  OIT
4 /**
5 //  Created by Maddie on 4/3/24.
6 /**
7
8 import Foundation
9 import SwiftUI
10 import CareKitUI
11 import CareKit
12
```

```

13 struct SymptomsOverTimeChartView: UIViewRepresentable {
14     let symptomDataManager: SymptomDataManager
15     @Environment(\.colorScheme) var colorScheme
16
17     func makeUIView(context: Context) -> OCKCartesianChartView {
18         let chartView = OCKCartesianChartView(type: .bar)
19         chartView.headerView.titleLabel.text = "Symptoms Over Time"
20
21         _ = symptomDataManager.fetchAllSymptoms()
22         let symptomRecordsByDate = symptomDataManager.symptomRecords
23         var dataPoints: [(Date, Int)] = []
24         for (date, symptoms) in symptomRecordsByDate {
25             dataPoints.append((date, symptoms.count))
26         }
27         dataPoints.sort { $0.0 < $1.0 }
28         _ = dataPoints.map { $0.0 }
29         let counts = dataPoints.map { Double($0.1) }
30
31         let dataSeries = OCKDataSeries(values: counts.map { CGFloat($0) }, title: "Symptoms",
32             color: colorScheme == .light ? UIColor(.darkTeal) : UIColor(.lightTeal))
33         chartView.graphView.dataSeries = [dataSeries]
34
35         let dateFormatter = DateFormatter()
36         dateFormatter.dateFormat = "MMM dd"
37         return chartView
38     }
39
40     func updateUIView(_ uiView: OCKCartesianChartView, context: Context) {
41         // Update the view if needed
42     }
43 }
```

## A.46 SymptomsThisWeekChartView.swift

```

1 /**
2  *  SymptomsThisWeekChartView.swift
3  *  OIT
4  */
5 // Created by Maddie on 4/3/24.
6 //
7
8 import Foundation
9 import SwiftUI
10 import CareKitUI
11 import CareKit
12
13 struct SymptomsThisWeekChartView: UIViewRepresentable {
14     let symptomDataManager: SymptomDataManager
15
16     func makeUIView(context: Context) -> OCKCartesianChartView {
17         let chartView = OCKCartesianChartView(type: .bar)
18         chartView.headerView.titleLabel.text = "Symptoms This Week"
19
20         let oneWeekAgo = Calendar.current.date(byAdding: .day, value: -7, to: Date()) ?? Date()
21         let symptomRecordsPastWeek = symptomDataManager.symptomRecords.filter { $0.key >= oneWeekAgo }
22         let symptomRecordsByDate = symptomRecordsPastWeek.sorted(by: { $0.key < $1.key })
23         var allSymptoms: Set<String> = []
24         for (_, symptoms) in symptomRecordsByDate {
25             allSymptoms.formUnion(symptoms)
26         }
27
28         for (index, symptom) in allSymptoms.enumerated() {
29             let counts = symptomRecordsByDate.map { (_, symptoms) in
30                 symptoms.contains(symptom) ? 1.0 : 0.0
31             }
32             let dataSeries = OCKDataSeries(values: counts.map { CGFloat($0) }, title: symptom.
33         }
34 }
```

```

33     splitCamelCase(), color: UIColor(Color.mutedRandom))
34         chartView.graphView.dataSeries.append(dataSeries)
35     }
36
37     let dateFormatter = DateFormatter()
38     dateFormatter.dateFormat = "EEEE"
39     chartView.graphView.horizontalAxisMarkers = symptomRecordsByDate.map { dateFormatter.
40     string(from: $0.key) }
41     chartView.graphView.yMaximum = 1
42     chartView.graphView.yMinimum = 0
43
44     return chartView
45 }
46
47 func updateUIView(_ uiView: OCKCartesianChartView, context: Context) {
48     // Update the view if needed
49 }
50 }
```

## A.47 GetSetUpView.swift

```

1 // GetSetUpView.swift
2 // SeniorDesign
3 //
4 // Created by Maddie on 9/25/23.
5 //
6
7
8 import SwiftUI
9
10 struct GetSetUpView: View {
11     // MARK: View Models
12     @EnvironmentObject var profileViewModel: ProfileViewModel
13     @EnvironmentObject var healthKitViewModel: HealthKitViewModel
14     @EnvironmentObject var appState: AppState
15
16
17     // MARK: UI Elements
18     var setUpTitle: some View {
19         Text("Set Up")
20             .font(.largeTitle.bold())
21     }
22
23     var setUpInstructions: some View {
24         Text("Fill out the following information to get started.")
25             .font(.subheadline)
26             .padding(.top, 1)
27     }
28
29     var getStartedButton: some View {
30         Text("Get Started")
31             .font(.caption)
32             .foregroundColor(Color.white)
33             .padding()
34             .background(Color.darkTeal)
35             .cornerRadius(30)
36     }
37
38     // MARK: Get Set Up View
39     var body: some View {
40         ScrollView {
41             VStack(alignment: .leading) {
42                 setUpTitle
43                 setUpInstructions
44                 SettingsView()
45                     .environmentObject(profileViewModel)
46                     .environmentObject(healthKitViewModel)
47             }
48         }
49     }
50 }
```

```

47     Spacer()
48     HStack(alignment: .center) {
49         NavigationLink(destination: TabbedApplicationView().navigationBarBackButtonHidden(true)) {
50             Spacer()
51             getStartedButton
52                 .onTapGesture {
53                     appState.hasAppBeenOpenedBefore = true
54                 }
55             Spacer()
56         }
57     }
58     .padding(.top, 20)
59 }
60     .padding()
61 }
62 }
63 }
64
// MARK: Preview
65 #Preview {
66     GetSetUpView()
67         .environmentObject(ProfileViewModel())
68         .environmentObject(HealthKitViewModel())
69 }
70

```

## A.48 GetStartedView.swift

```

1 /**
2 //  GetStartedView.swift
3 //  SeniorDesign
4 //
5 //  Created by Maddie on 9/25/23.
6 //
7
8 import SwiftUI
9
10 struct GetStartedView: View {
11     @EnvironmentObject var appState: AppState
12     // MARK: UI Elements
13     var getStartedButton: some View {
14         NavigationLink(destination: GetSetUpView().navigationBarBackButtonHidden(true)) {
15             Text("Get Started")
16                 .font(.caption)
17                 .foregroundColor(Color.white)
18                 .padding()
19                 .background(Color.darkTeal)
20                 .cornerRadius(30)
21         }
22         .padding()
23     }
24
25     var appTitle: some View {
26         VStack {
27             Spacer()
28             Text("OIT")
29                 .font(.largeTitle.bold().monospaced())
30                 .foregroundColor(.white)
31                 .padding(.bottom, 1)
32             Text("oral immunotherapy tracker")
33                 .font(.callout.lowercaseSmallCaps())
34                 .fontDesign(.rounded)
35                 .foregroundColor(.white)
36                 .padding(.bottom, 20)
37         }
38     }
39

```

```
40 // MARK: Get Started View
41 var body: some View {
42     NavigationStack {
43         VStack {
44             ZStack {
45                 Color.darkTeal
46                     .frame(maxHeight: .infinity)
47                     appTitle
48             }
49             VStack {}
50             ZStack {
51                 Color.lightTeal
52                     .frame(maxHeight: .infinity)
53                 VStack {
54                     getStartedButton
55                     Spacer()
56                 }
57             }
58         }
59         .edgesIgnoringSafeArea(.all)
60     }
61 }
62 }
63
64 // MARK: Preview
65 #Preview {
66     GetStartedView()
67 }
```

# Bibliography

- [1] Food Allergy Research and Education, “Preventing Food Allergies: Early Interventions — FARE,” [www.foodallergy.org](http://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 Web-site. <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/spec/UML/](http://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>.