Assignment 3 - Group Project

Written Report (35%)

Assessment Due Date: 10:00 A.M 23 September 2024

By: Droplet

Pham Hoang Long - s3938007

Do Phan Nhat Anh - s3915034

Dinh Le Hong Tin - s3932134

Ngo Ngoc Thinh - s3879364

Table of Contents

I. Introduction

a. Goal

The "Smart Pharmacy" application's main goal is to bridge the gap in the access of healthcare in rural areas and the more developed cities, by offering pharmaceutical services on digital platforms. This application will allow users, especially people in under-served areas, to have access to key medications, health consultations, and prescriptions. With the use of mobile technology, Smart Pharmacy aspires to improve health outcomes in areas where medical services and medical professionals are scarce.

b. Inspiration

Vietnam is currently still a developing country, in remote places or in some rural areas, healthcare services are not evenly distributed. People living in these regions often struggle to access medical professionals, healthcare facilities, and necessary medications. This disparity can lead to preventable health complications, delayed treatment, and overall poor health outcomes. The Smart Pharmacy app was inspired by the urgent need to address this healthcare inequality and to provide a solution that leverages modern technology to make healthcare more accessible for everyone.

With the integration of digital technologies, Smart Pharmacy could be considered a creative and innovative solution to one of the most current acute social problems in the sphere of pharmaceutical services. Even though services of telemedicine and other online pharmacy options exist, Smart Pharmacy has been devised to respond to specific needs related to the rural population of Vietnam. Its originality lies in the ability to incorporate the local pharmacy into the nation-wide network, which assures the inclusion of small or local business entities in medicine distribution. This will also ensure tailored solutions to geographical and medicinal needs of different communities with deep insight into the landscape of healthcare in Vietnam.

Consequently, Smart Pharmacy brings novelty to the convenience and access of health services. It increases access to healthcare resources for people who could not have had them easily, either by geography or economics, while at the same time facilitating improvements in health outcomes among the most vulnerable. It is a transformative tool to reduce health disparities within the Vietnamese healthcare system.

c. Competitors

Online pharmaceutical services are no sort of new invention in the current day, including Vietnam. There are large businesses that revolve around medication who have expanded their marketplace to the digital world. For example, e-Pharmacy apps in Vietnam such as Medigo and Pharmacity already provide online access to pharmaceutical services integrated with an order management for medication delivery across the country. Besides the e-Pharmacy apps, there are telemedicine platforms that offer teleconsultations and access to healthcare professionals; platforms such as Jio Health also provide an abundance of pharmaceutical services as well as online purchasing of medicines.

In comparison to Smart Pharmacy, most existing applications or platforms have yet to integrate AI technology, which has become one of the most prominent and widely discussed topics in recent years. The selling point of Smart Pharmacy is its built-in AI-integrated medical assistant. which provides a highly personalized and interactive healthcare experience. In the current version, it is designed to assist users in managing common health concerns, such as headache, stomachache, and sleep…, those are not strictly requiring a doctor appointment. The medical assistant also understands the medicines in the database and will suggest users with prescriptions. This AI-powered assistant brings a level of convenience and personalization that most competitors currently lack, proposing a prominent solution to the unevenly distributed healthcare services in Vietnam.

II. Implementation Details:

1. Main Features:
    a. User Management and Customization:

"Smart Pharmacist" leverages Firebase Authentication to provide secure user management and customization options. Users can sign up or log in through email, Google, or Facebook accounts. The app also integrates Firebase Firestore to store user data, allowing users to easily update their profiles by changing their name, date of birth, phone number, address, and avatar. In addition, when users sign up with Facebook or Google, their names and avatars will be synchronized from these platforms to the application's database.

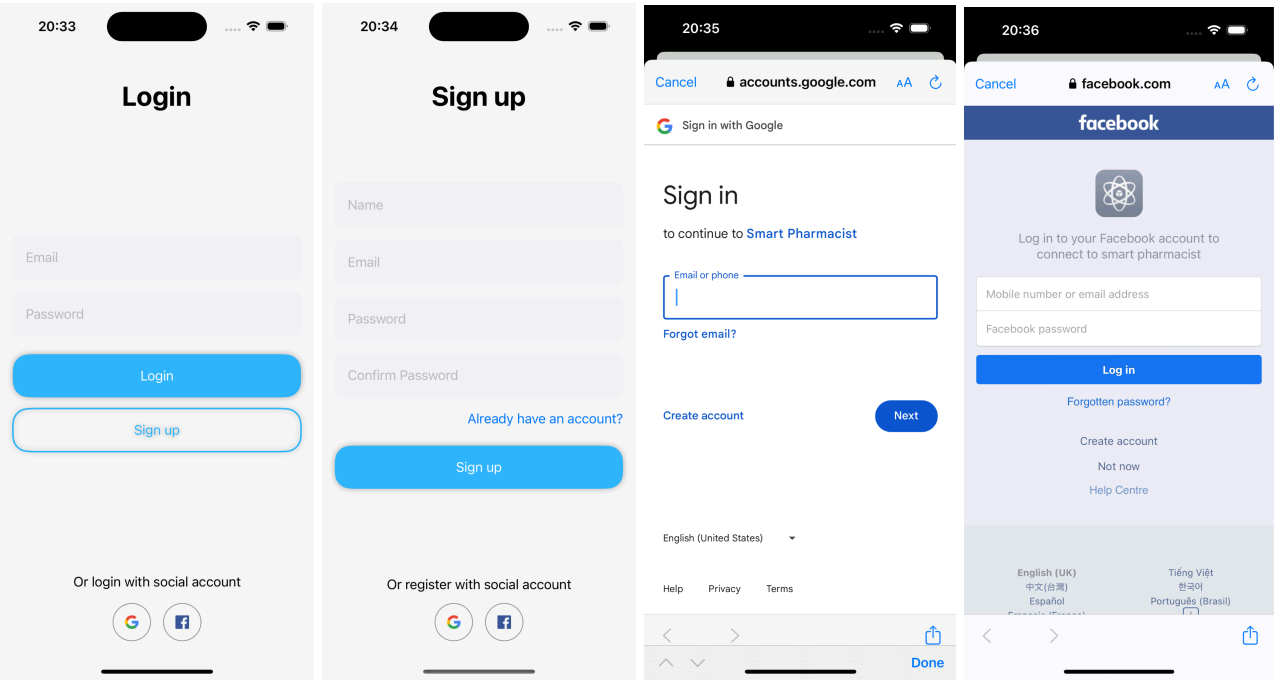


Figure 1: Sign In View + Sign Up View + Sign In with Google account + Sign In with Facebook account

The *UserProfileView* displays user information and provides navigation to detailed settings. In *UserSettingsView*, users can edit their profile (name, email, phone, address), customize their app appearance (light/dark mode), and set notification preferences. Furthermore, user data is stored in Firebase, with the *UserService* handling CRUD operations. For the admin role, it will have additional features: Add New Medicine View, for adding the new product to the database and updating order status. Lastly, the app uses *UserDefaults* for local storage of preferences such as dark mode settings.

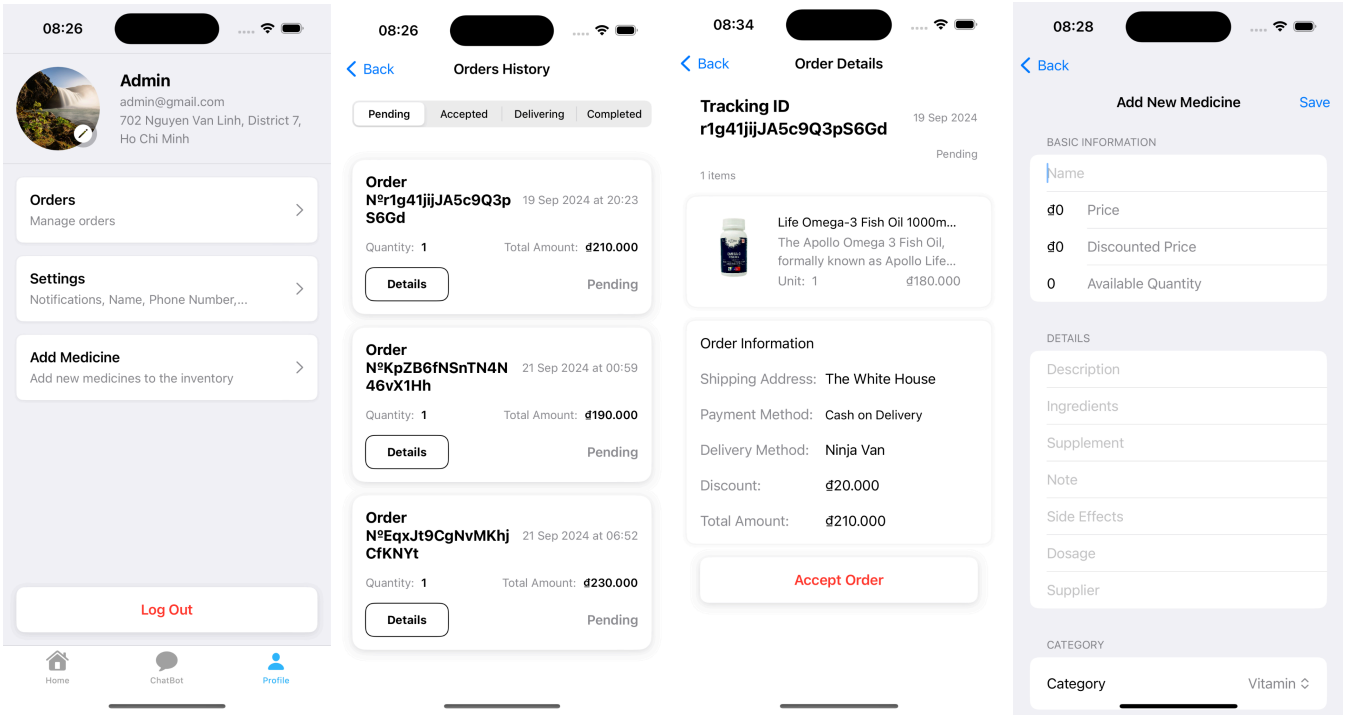


Figure 2: Admin Profile View + Manage Order View + Order Detail View + Add New Medicine View

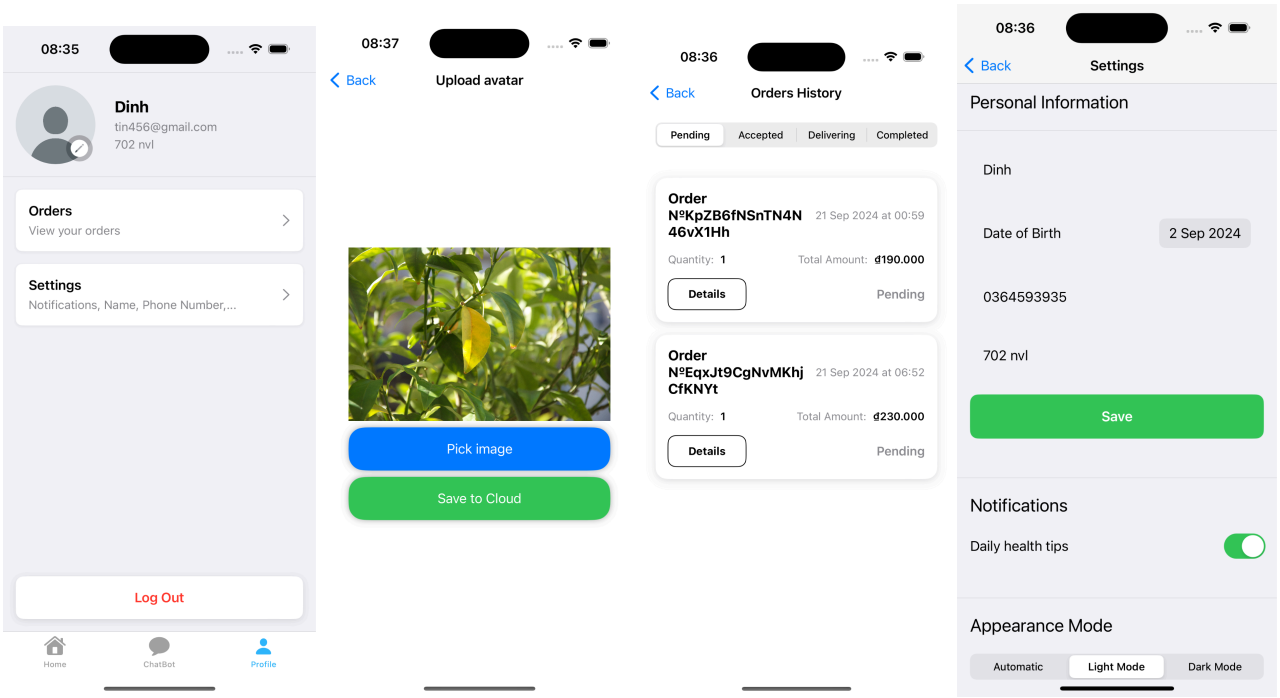


Figure 3: User Profile View + Upload Avatar View + Order History View + Profile Settings View

b. Search and Filter:

The application allows users to search for products, and filter them by 7 available categories.

- Search Product: This one is implemented using a *TextField* bound to the searchText property in the MedicineViewModel. The *filterMedicines()* function is triggered whenever the user types, which filters the medicines array based on whether the medicine name contains the search text. The filtered results are then displayed in real-time in the *MedicineView*, providing immediate response to the user.

- Filter Product: Users can select categories (*Flash Sales, New Releases, Vitamins, Calcium, etc.*) via the *CategorySectionView*. When a filter is selected, the *filterMedicines()* function applies the corresponding criteria to the medicines list, updating the filteredMedicines array. This filtered list is then used to display only the relevant products in the *MedicineView*, which will allow users to quickly narrow down their product view based on specific attributes or promotions.

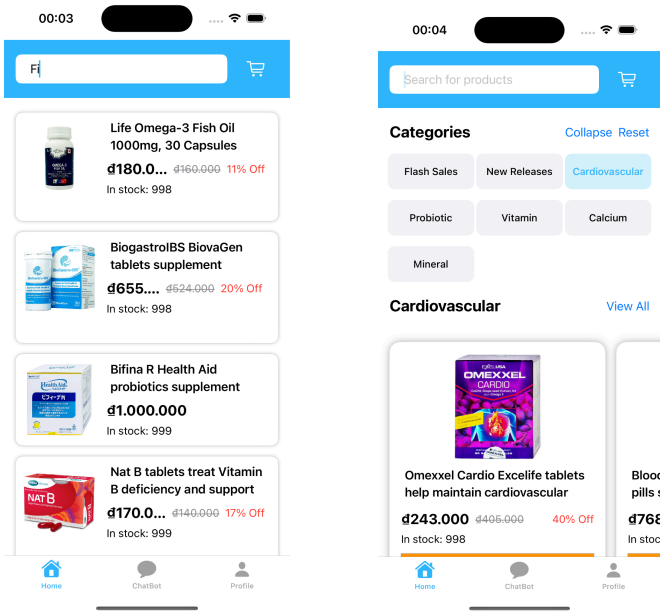


Figure 4: Search View with "Fi" text + Filter View with "Cardiovascular" category

c. Data Persistence & CRUD operations:

Medicine (): The application implements data persistence, CRUD operations, and user data synchronization for *Medicine* by using Firebase as its backend. A *MedicineService* class, extending the generic *CRUDService*, handles medicine-related database operations. The *Medicine* model represents individual products, storing details like name, price, description, and inventory. CRUD operations are implemented asynchronously, allowing for efficient creation, retrieval, updating, and deletion of medical data. The *MedicineViewModel* manages the state of medicine-related views, fetching and updating data as needed.

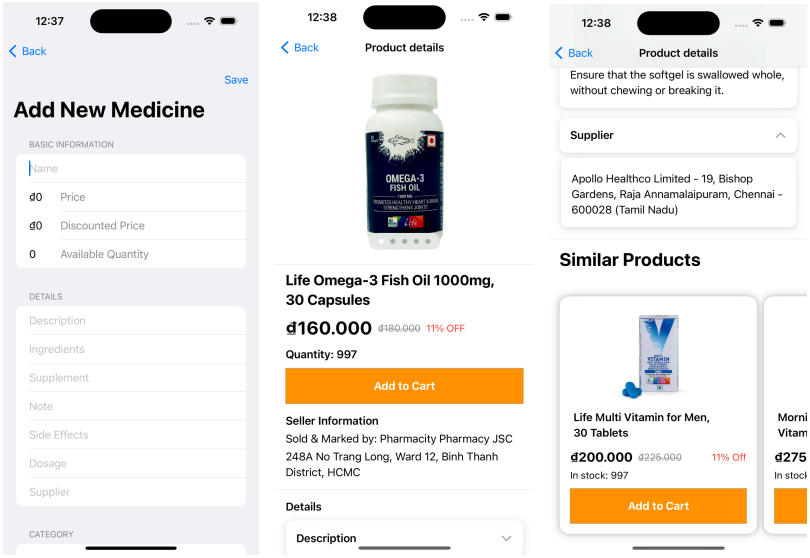


Figure 5: Add New Medicine View + Medicine Detailed View + Similar Products in Product Detailed View

Cart (): The application implements data persistence, CRUD operations, and user data synchronization for the Cart by using Firebase as its backend. It employs a generic *CRUDService* class for basic database operations, extended by specific services including *CartService* and *CartItemService*. These services manage cart-related data using asynchronous methods with Swift's async/await pattern. The *CartDeliveryViewModel* acts as an intermediary, managing the cart's state and synchronizing data between the UI and Firebase.

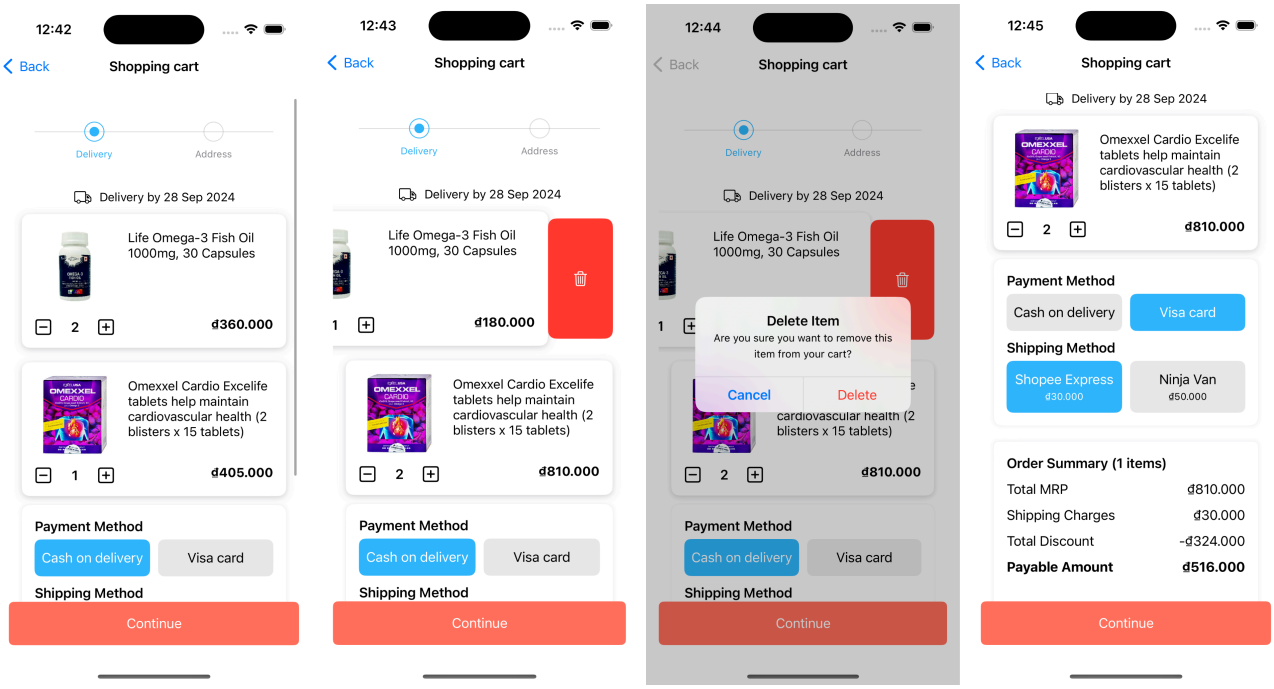


Figure 6: Cart View + Delete Cart + Confirmation Alert + Order Summary in Cart View

User Data Synchronisation: "Smart Pharmacist" implements user data synchronization using Firebase services, primarily FirebaseAuth and Firestore. It employs a combination of local and remote data management strategies. User authentication is handled through multiple methods (email + password, Google, or Facebook), with user data stored both locally for offline access and in Firestore for cloud synchronization. For example, user authentication ensures that cart operations are associated with the correct user, while real-time updates allow changes to be immediately reflected across devices. Another example would be the medicine view, which enables real-time updates to medicine information, ensuring that product details, inventory levels, and prices are always current across the app. The architecture supports features such as browsing medicines and viewing details during the shopping process, with all changes persisting in the database and synchronizing across user sessions.

d. Order Status:

The "Order" in this application is implemented using the MVVM architecture pattern with SwiftUI. While the core data structure is represented by the *Order* model, the *OrderViewModel* acts as an intermediary, handling business logic such as fetching order history and updating statuses. Views like OrderView and OrderDetailView

present the data to the user, utilizing custom UI components for enhanced presentation. The implementation leverages SwiftUI's state management properties and navigation system, while employing async/await for asynchronous operations.

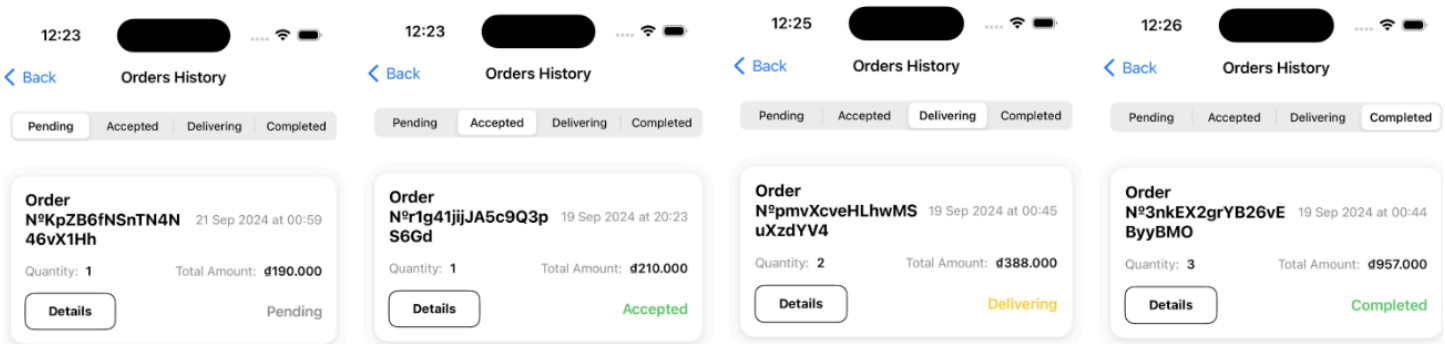


Figure 7: Orders History with 4 statuses "Pending", "Accepted", "Delivering", "Completed"

e. Dark Mode Support:

The application takes advantage of SwiftUI's built-in *colorScheme* environment variable and a custom *DarkLightModeService*. It responds to system-wide dark mode changes and also allows users to manually set their preferences. This preference is stored using *UserDefaults* and applied throughout the app using custom color sets that adapt to the current mode.

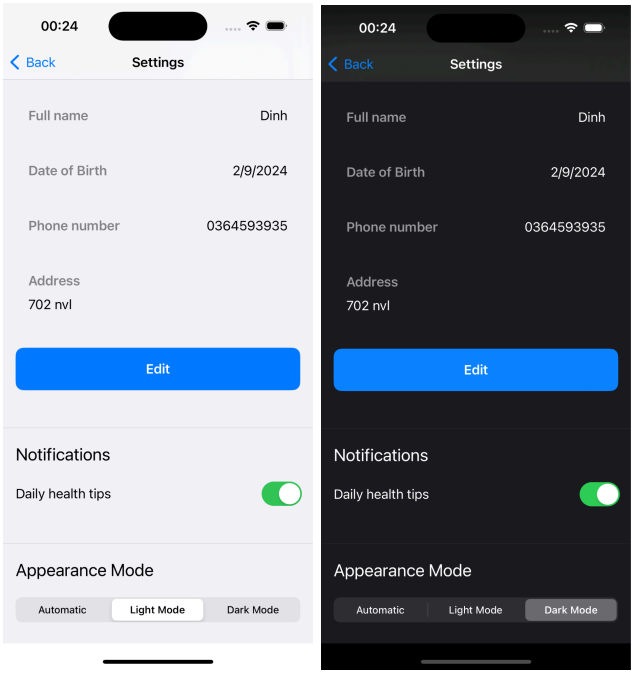


Figure 8: Dark Mode options in Setting View

f. Animated Splash Screen:

This screen uses SwiftUI animations to create a visually appealing intro screen with a scaling and rotating pharmacy logo. It employs a *LinearGradient* background and animated text appearance. After a fake delay

simulated with *DispatchQueue*, it automatically navigates to the login screen using SwiftUI's NavigationStack. This creates a smooth transition from the app launch to the main interface, providing a polished and professional first impression to users. The animation of this screen will be demonstrated in our presentation.

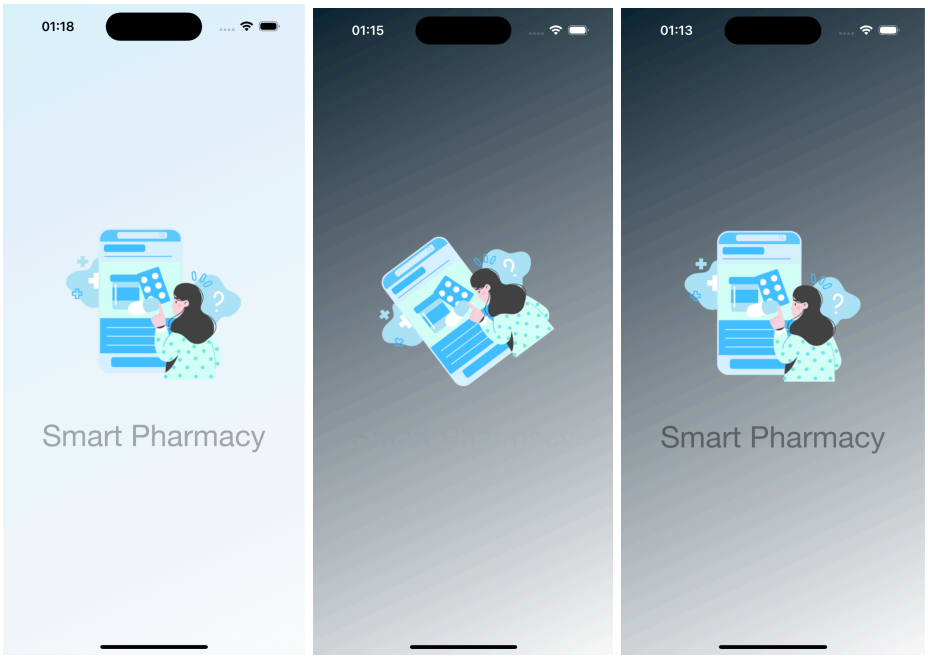


Figure 9: Splash Screen in Light Mode + Rotating + Dark Mode

2. Advanced Features:
    a. AI Chatbot:

In this feature, OpenAI's GPT model is integrated into the application. It maintains a conversation history as an array of messages, each containing HyperLinkResponse objects. User input is captured in a TextField and sent to the *OpenAIService* when submitted. The service processes the input and returns a response, which is then displayed in the chat interface. In addition to that, the chat supports *hyperlinked text* that can navigate to *medicine details* when tapped. The user interface also shows a loading indicator while waiting for the AI's response, making the user experience smooth. Under the hood, before the user interacts with the AI Assistant, its memory will be updated with the latest medicine database so that it can suggest related products to the user. Besides that, the AI will be prompted smartly so that its text responses can be parsed and mapped into appropriate products available in the Smart Pharmacy resulting in clickable hyperlink texts.

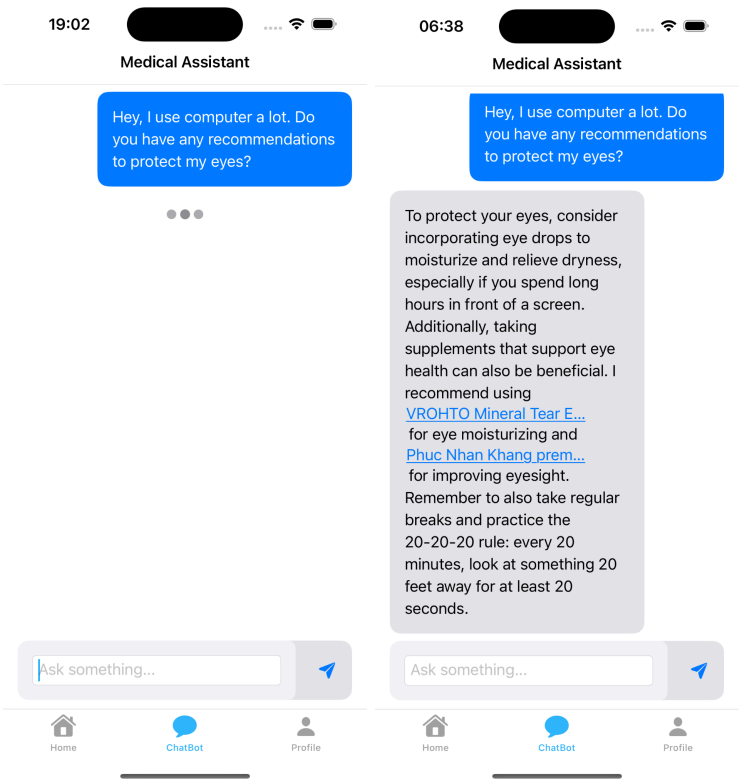


Figure 10: AI Chat Box with Loading + Instruction and Medicine Recommendations

b. Real Payment with Stripe:

In this feature, the *StripePaymentViewModel* manages the payment logic, including creating a *PaymentIntent* via a *Firebase Cloud Function* and configuring the *PaymentSheet*. When a user proceeds to payment, the *initiatePayment()* function is called, which sets up the *PaymentSheet* with the necessary customer detailed information securely. Upon successful payment, the order is processed, ensuring a seamless and secure checkout experience integrated directly into the app's workflow. However, please note that this feature is only available when users choose the payment method of their Visa card.

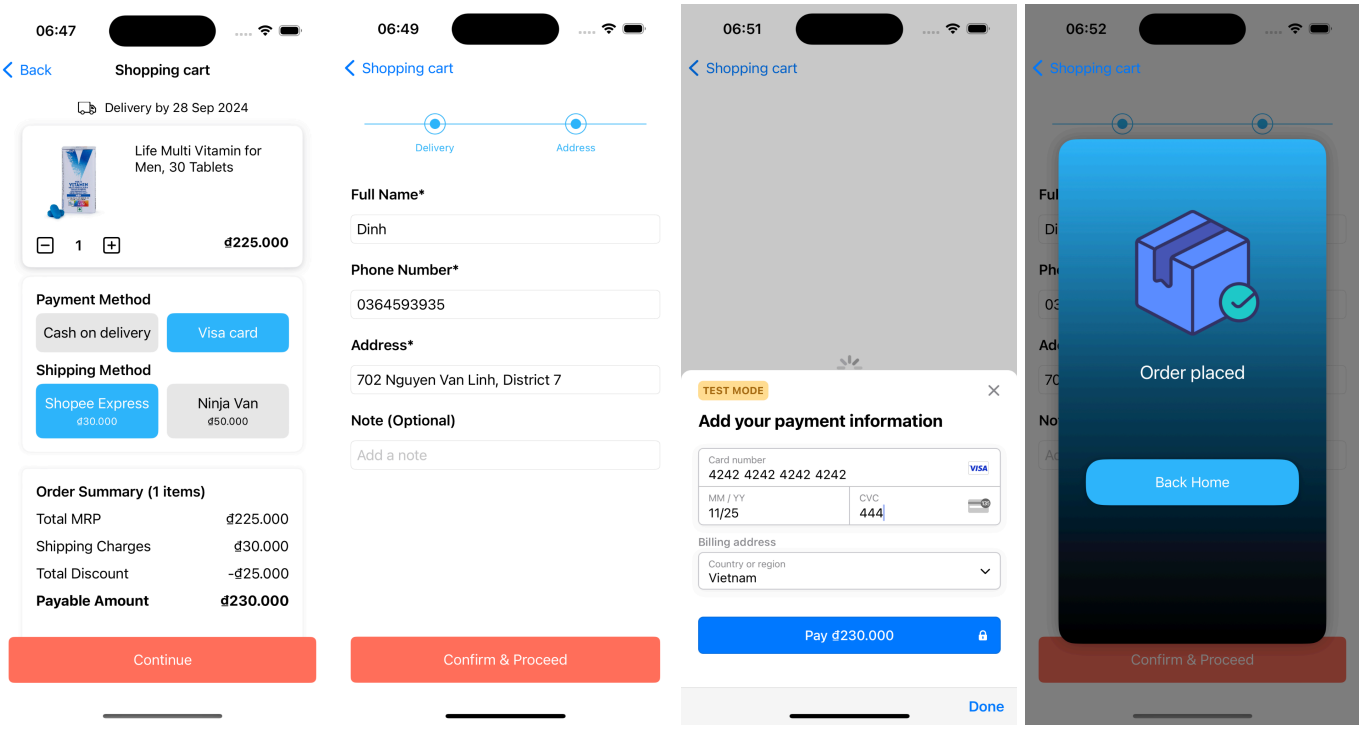


Figure 11: Shopping Cart with 230.000đ + Stripe Payment View with data + Order Successfully Modal

c. Local Notifications (daily health reminder notification):

The *NotificationService* scheduled local notifications using *UNUserNotificationCenter*, allowing the app to send daily health tips to users. The content for these notifications is generated dynamically every user opens the application using the *OpenAIService* above, ensuring varied and relevant health advice. More specifically, every time user opens the application, in the background, all the pending notifications will be canceled and the application will prompt OpenAI to provide health quotes such as "Pretend that you are a professional doctor, suggest me a health quote about 20 words to help me have a good day start" and use OpenAI's response to schedule new notifications in the morning (7 AM) and night (9 PM). Besides, users can customize their notification preferences in the app settings, and the service handles permission requests, scheduling, and cancellation of notifications. Hereby, this feature provides users with regular health reminders, which will enhance engagement and promote wellness through the app.

Figure 12: Daily Notifications in Lock Screen (left) and Home Screen (right) at 9 P.M

III. Application Flow

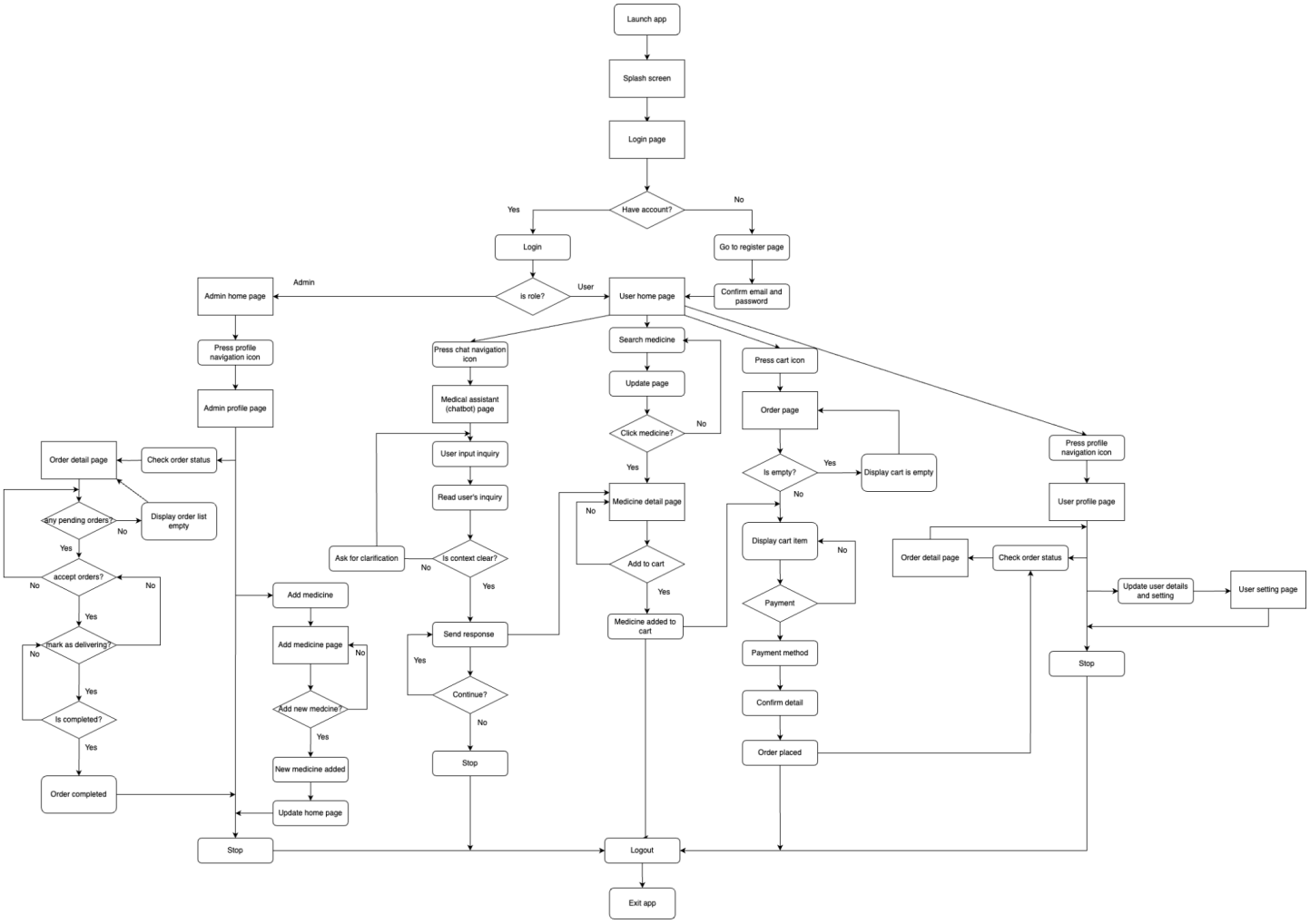


Figure 13. Application flow diagram (draw.io)

The flow begins at the splash screen, followed by the login page. Users who do not have an account can navigate to the register page to create one by confirming their email and password. After successful registration or login, the system checks the role—whether they are a user or an admin—and directs them to the same home page. While both users and admins share this home page, their functionalities differ based on their roles. Admins do not go through a registration process and login with predefined credentials, but they also land on the same home page as users. However, for admins, this page serves as a preview where they can view and manage products as a user would.

For users, the home page provides several actions. They can search for medicines, view the search results, and select a specific product, which takes them to the medicine detail page. If they choose to buy the medicine, they can add the product to their cart. By pressing the cart icon, users navigate to the order page, where they can proceed with payment if the cart contains items, or see a message indicating an empty cart. After confirming the

payment method and order details, the user places the order and is redirected to the order detail page to check the status of the purchase. Users can also access their profile page to update personal details or settings. If assistance is required, users can interact with the chatbot on the medical assistant page, where they can input inquiries and receive relevant responses. Once their tasks are completed, users can log out or exit the app.

For admins, although they land on the same home page as users, they have additional capabilities. The home page functions as a preview, allowing them to see how products appear from a user's perspective. Additionally, admins can update the product catalog by navigating to the add medicine page, where they can add new medicines to the system. After adding products, admins can return to the home page to verify that the new medicine has been successfully added, simulating the user experience to ensure the product is visible and ready for purchase. Admins can also manage user orders by accessing the order detail page, where they can view pending orders, accept them, mark them as "delivering," and complete the process once delivery is confirmed. Like users, admins can access their profile page to manage their account details and, once their administrative tasks are completed, they can log out or stop their session.

IV. Backend Stack

After comprehensive considerations, we have decided that we will use Firebase as the main storage of the application combined with a small part of data that will be stored with UserDefault. The adoption of Firebase as the main data persistence solution for the **"Smart Pharmacy"** app is reasonable due to numerous essential advantages and features that correspond with the specification and technological demands of a cloud-based mobile application. Below are some of the justifications:

Real-time Data Synchronisation

Firestore from Firebase offers real-time synchronization, enabling instantaneous updates of user data, such as profiles, orders, and cart information, across various devices and sessions. This capability is particularly vital in an application such as Smart Pharmacy, where users engage with real-time inventories, submit orders, and manage profiles. Real-time data synchronization guarantees that information remains up-to-date, enhancing user experience across devices. For instance, if a customer adds things into their carts, Firebase Firestore guarantees that these alterations will be consistent across different customer's devices without manually updating.

Scalability and Flexibility

The NoSQL architecture of Firebase's Cloud Firestore offers a highly scalabilty solution and flexibility in the nature of NoSQL, which are crucial for applications expected to expand in number of users and features like "**Smart Pharmacy**" in the future.

Conversely, although Core Data is an effective instrument for local data persistence, it is mostly intended for small to medium-scale applications and is inadequate for extensive, real-time, cloud-based operations. Firebase offers a highly scalable cloud architecture, providing free options for application in the beginning phase with

pay-as-you-go scalability. Backed by Google, Firebase has proven its stability and durability over the years, exceeding Core Data's capabilities in cloud scalability.

User Authentication and Security

Firebase Authentication is seamlessly integrated into the platform, facilitating the implementation of secure login functions through Google, Facebook, or email with ease and reducing development time in the context of a hackathon. Firebase Authentication provides Smart Pharmacy with secure management and synchronization of user data across devices, making Firebase a superior option to Core Data, which lacks native user authentication capabilities. Please note that Firebase will only handle user's authentication, in terms of authorization (normal user and admin in this case) will be handled at the application level.

Firebase Storage and UserDefaults

Firebase Firestore handles not only domain specific data such as products, orders and users information but also user's preferences such as notifications's On/Off state. In combination with UserDefaults for smaller, device-centric settings such as dark/light mode. More specifically, UserDefaults is only for storing device-oriented preferences, for example sometimes users want to use dark mode in iPhone but light mode in iPad this separation of storage will offer that functionality.

Image storage

A significant advantage of Firebase is its seamless interaction with Firebase FileStore, offering effective and scalable cloud storage solutions for user-generated media, such as image. Within the context of "**Smart Pharmacy**" which covers functionalities such as the uploading and management or user avatars, Firebase FileStore provides a versatile solution for the handling of significant amounts of image data. In contrast, Core Data does not provide an inherent capability for managing media files such as photos except serialized-binary storage which makes the media storage inefficient.

V. Database Design

As shown in Appendix 3, the application follows an object-oriented programming style [1] that models all the entities within Smart Pharmacy and treats them as objects. The figure describes the relationships between entities. Specifically, the entity relationships will strictly follow third normal form practices to avoid data duplication [2].

Each entity or model in the application is associated with a distinct service responsible for performing CRUD (Create, Read, Update, Delete) operations on records related to that model. For example, services like UserService, CartService, and CartItemService each manage their corresponding entities. These services are intended to derive from a foundational service class, CRUDService, which offers essential functionality for basic functions such as createDocument(), updateDocument(), and deleteDocument().

The application improves code reusability and maintainability by utilizing CRUDService as a foundational component (Appendix 3). It guarantees that all services use the conventional CRUD operations while permitting customization according to particular business logic requirements in the sub-inherited services. For example, CartItemService contains the essential CRUD actions derived from CRUDService, while also integrating further functionality, such as increasing or decreasing items' quantities.

VI. External Libraries and Install Instructions

Smart Pharmacy adopts some required dependencies for Firebase, Google, and Facebook. Besides, some external SwiftUI-component libraries are also used to create a good user experience in a manner of short development time. Below is the comprehensive list of external libraries:

Name	**Description**	**Dependency link**
FirebaseAnalytics	Firebase analytics framework, not used within the app but is required	https://github.com/firebase/firebase-ios-sdk.git
FirebaseAuth	User authentication with email and password	
FirebaseFirestore	Firebase database to save data related to entities/models	
FirebaseStorage	Media storage such as images of users's avatars or product images	
FirebaseFunctions FirebaseMessaging	Firebase serverless function script and messaging required for establishing the connection to Stripe	
GoogleSignIn GoogleSignInSwift	Google authentication integrated with Firebase Authentication	https://github.com/google/GoogleSignIn-iOS
FacebookCore FacebookLogin	Facebook core and authentication integrated with Firebase Authentication	https://github.com/facebook/facebook-ios-sdk
ChatGPTSwift	ChatGPT iOS client to perform API request and response cycle with ease	https://github.com/alfianlosari/ChatGPTSwift.git
Stripe StripePaymentSheet	Stripe core library and payment sheet UI component, in cooperation with Firebase function to perform real checkout flow	https://github.com/stripe/stripe-ios-spm
ActivityIndicatorView	The library contains lots of loading	https://github.com/exyte/ActivityIndicator

	animation used within the application to make the application responsive to user interactions	View.git
PopupView	The library contains a customizable popup view component that allows the application to display popups where needed such as Order Completion	https://github.com/exyte/PopupView.git
WrappingHStack	Bring text wrap effect (auto insert a new line when the text of hyperlink and normal text exceed a row width) in the chat box view	https://github.com/dkk/WrappingHStack

Table 1: Application's dependencies table

Install Instructions:
Before the project can be run, all the mentioned dependencies in *Table 1* must be installed. Here are step-by-step instructions to install those dependencies using Swift Package Manager and run project in XCode:

1. Clone and open the XCode project
2. Navigate the Package Dependencies
    a. Click on the project in the **Project Navigator**.
    b. Select the app target (in this case **pharmacist_project**) and go to the **Package Dependencies** tab.
3. Add all the required packages from top to bottom. For each "dependency link":
    a. Click the **"+" button** at the bottom of the package list.
    b. In the search field, paste the URL in the "dependency link" column. For example: "https://github.com/firebase/firebase-ios-sdk.git"
    c. Select the dependency rule "**Up to next major version**" then click "**Add Package**", and then a list of modules inside that package will be shown.
    d. Select the appropriate modules associated with the "dependency link" following the dependencies table in the column "Name". For example, with the dependency link "https://github.com/google/GoogleSignIn-iOS", we must choose both modules "GoogleSignIn" and "GoogleSignSwift". For each selected module, under "**Add to Target**", select the current target (pharmacist_project).
    e. Click "**Add Package**" to add dependencies to the project. Repeat from step 3.a to 3.e for all the dependency links.
4. Run the project with Command + R.

Note: The above steps are only required when the XCode can not recognize any required dependencies, in case XCode has already recognized all the dependencies, we can skip all those steps and in the navigator go to **File -> Swift Packages -> Resolve Package Versions** and wait for the dependencies to be automatically resolved.

VII. Design Elements and User Experience

Visual Appeal

Color theme: The application employs a clean and professional color palette, suitable for a medical and pharmaceutical setting. In light mode, the background is primarily white, complemented by blue and orange highlights. The text is rendered in black or dark gray to enhance readability against the bright background. In dark mode, the application transitions to a dark gray or black background with light text (white or light gray), reducing eye strain, particularly in dim light environments. The use of orange and blue for interactive elements, including buttons and icons, is consistent throughout all modes, ensuring visual coherence and adherence to the health concept.

Font: A contemporary sans-serif font is employed consistently across the application, guaranteeing clarity and readability. The font size is suitable for mobile devices, with bigger headers to direct the user's focus and smaller body text for descriptions. The fonts are consistent throughout bright and dark modes, with contrast well-adjusted to provide legibility in both settings.

Layout: The layout is user-friendly and systematically organized. Crucial parts such as "Search," "Medication List," and "Prescription Management" are shown with sufficient spacing between components to avoid a cluttered interface. Buttons are positioned near the bottom for accessibility, with significant activities like making orders or reading information highlighted by color and size. Each screen has a distinct structure, positioning the most essential information at the top and subordinate activities or details behind it, facilitating smooth navigation.

Intuitive User Interface

User-friendly design: The application emphasizes an intuitive design to facilitate navigation for all users, especially those with limited proficiency in digital technologies. Essential elements, like medicine search, prescription information, and pharmacy contact data, are prominently displayed. Clear icons with large buttons assist users in navigating critical activities. The design reduces the processes required to accomplish a task, hence improving user efficiency.

Search and filter: The application offers a search functionality enabling users to swiftly locate medications or pharmacies. The filtering options, like sorting by location, price, or availability, enable consumers to refine their selections. These filters are collapsible, minimizing clutter and providing users with a simplified interface while not in use.

Usability considerations: The application is developed with accessibility as a priority. Large buttons and an intuitive navigation mechanism facilitate user interaction with the interface, even on smaller screens. The back button is consistently accessible, allowing users to easily return to previous screens. The text's readability is meticulously assessed with the contrast between the text and the background guaranteeing clarity in both bright and dark modes. Icons are easy to recognize, and interactive elements are clearly distinguishable from non-interactive ones.

Consistency

Colors: The application employs a consistent palette of blues, oranges, and grays to indicate interactive elements, backgrounds, and notifications. The color scheme of action buttons is identical across screens, facilitating users' instant recognition of available actions. The colors stay uniform between light and dark modes, ensuring familiarity irrespective of the mode employed.

Fonts and typography: The application ensures consistency in its selection of fonts and typographic hierarchy. Headings are emphasized by bold style and increased size to distinguish them from the smaller, lighter body substance. The clear differentiation of headers, subheadings, and body text fosters an organized flow of information that is easy to understand.

Imagery: Each medicine is illustrated by an image, offering consumers visual indicators for easy product identification. The graphics display consistency in style and presentation, facilitating users' quick recognition of common drugs and enabling informed decision-making. The use of medicine visuals enhances practicality and user-friendliness, facilitating improved interaction, particularly when users seek specific brands or varieties.

User-Centered Design

The application emphasizes user-centered design by concentrating on usability, accessibility, and functionality. It serves a diverse audience, ranging from technologically proficient individuals to people with limited expertise in-app navigation. The organized style, legible text, and readily recognizable symbols enhance the user experience.

Prioritizing accessibility: The application employs big, clickable buttons and clear labeling to facilitate user interaction with the UI. The capability to switch between light and dark modes accommodates diverse user preferences, enhancing usability under varied lighting conditions.

Task efficiency: The design minimizes the number of actions required to perform common tasks such as searching for medications or submitting a prescription, ensuring that users can accomplish what they need with minimal effort. The application’s functionalities are systematically arranged, with straightforward navigation that minimizes the learning curve for new users.

Consistency of experience: The application fosters user trust and minimizes errors by maintaining a consistent color scheme, font, and layout across all screens, ensuring users know what to expect. The use of images for

medications further improves user engagement, allowing them to visually confirm their choices before proceeding with an action.

VIII. Known Bugs/Problems:

Currently, “Smart Pharmacist” runs with no issues as expected, there are no found bugs or warnings. However, the application has not been tested on other Apple devices including iPad, Apple TV, and Apple Watch yet.

IX. Conclusion

In conclusion, “Smart Pharmacist” helps our team comprehensively approach iOS development with Swift and SwiftUI frameworks. We can apply fundamental concepts such as MVVM architecture, Firebase integration for authentication and data storage, networking with external APIs, state management, and custom UI components. Our team also incorporates more advanced features like third-party integrations for social login and payment processing, image handling, and accessibility considerations. Therefore, the project gives us a solid understanding of modern iOS development practices and techniques for building complex, feature-rich applications.

In the future, our team aims to focus on improving user experience, expanding functionality, and leveraging cutting-edge technologies with more 3 key features below:

- *Prescription Management:* Add new features for users to upload, manage, and renew prescriptions directly through the app. In addition to that, we will take advantage of existing daily reminders to notify users to take medicine on time.
- *Offline Mode:* The application will add the capability for basic app functionality without an Internet connection, and then it will sync data when connectivity is restored.
- *VoiceOver support:* The application will be implemented with VoiceOver support, Dynamic Type, and other accessibility features to cater to users with various disabilities such as the visually impaired.

X. Project Responsibilities

Group Member	**Role**	**Contribution(%)**
Pham Hoang Long - s3938007	Project Manager	25%
Do Phan Nhat Anh - s3915034	Front-end Developer	25%
Dinh Le Hong Tin - s3932134	Front-end Developer	25%
Ngo Ngoc Thinh - s3879364	Back-end Developer	25%

Task	Subtask	
Technical Implementation		
Application prototype	Design Figma	Anh Do, Tin Dinh
	Database modeling	Long Pham, Thinh Ngo
Backend Services	Setup Firebase project	Thinh Ngo
	Basic CRUD services	
	Data models	
Authentication	Login, logout UI	Anh Do
	Authentication Service	Thinh Ngo
	Integrate view model	Thinh Ngo
Home	Main UI components	Anh Do, Long Pham
	Medicine view	Anh Do
	Medicine service	Thinh Ngo
	Integrate medicine view model	Anh Do, Long Pham
Cart	Main UI components	Tin Dinh
	Cart view	Tin Dinh
	Cart Service	Thinh Ngo
	Integrate view model	Tin Dinh, Long Pham
	Stripe payment service	Long Pham
Order	Main UI components	Long Pham
	Order view	Long Pham
	Order Service	Thinh Ngo

	Integrate view model	Long Pham
User profile	User profile view	Long Pham
	User Service	Thinh Ngo
	Integrate view model	Long Pham
Admin	Create medicine view	Tin Dinh
	Integrate view model	Tin Dinh, Thinh Ngo
AI Medical Assistant	Chat box view	Anh Do
	AI Service	Thinh Ngo
	Integrate view model	Thinh Ngo
Notification	Push notification on schedule	Thinh Ngo
Splash screen	Design a splash screen	Anh Do
Dark/Light theme	Colour theme service	Anh Do, Thinh Ngo
	Integrate color theme to all views	Anh Do
Report		
Introduction	Write goal, project inspiration, competitors	Long Pham
Implementation Details	Explain the technical implementation of each feature	Tin Dinh
	Provide screenshots from different views of the app	
Database	Justify database choice	Thinh Ngo
	Database design	

Technologies	List all of the Swift, SwiftUI components and External Libraries used	Thinh Ngo
	Provide step by step instructions to setup project	
Application Flow	Create a diagram showing how different components/views interact.	Anh Do
Design Elements & User Experience	Explain the overall design concept : Visual Appeal, Intuitive UI, Consistency, User-Centered Design.	Anh Do
Conclusion	Reflect on what was learned from the project	Tin Dinh
	Suggest potential future improvements	
Known bugs/problems	Provides the list of known bugs/problems they haven't fixed	Tin Dinh
Video presentation	Design a simple PowerPoint slide	Anh Do
	Project detail presentation	Long Pham, Tin Dinh, Thinh Ngo, Anh Do
	Application demo	Anh Do

XI. The Video Presentation

Link Youtube - Presentation

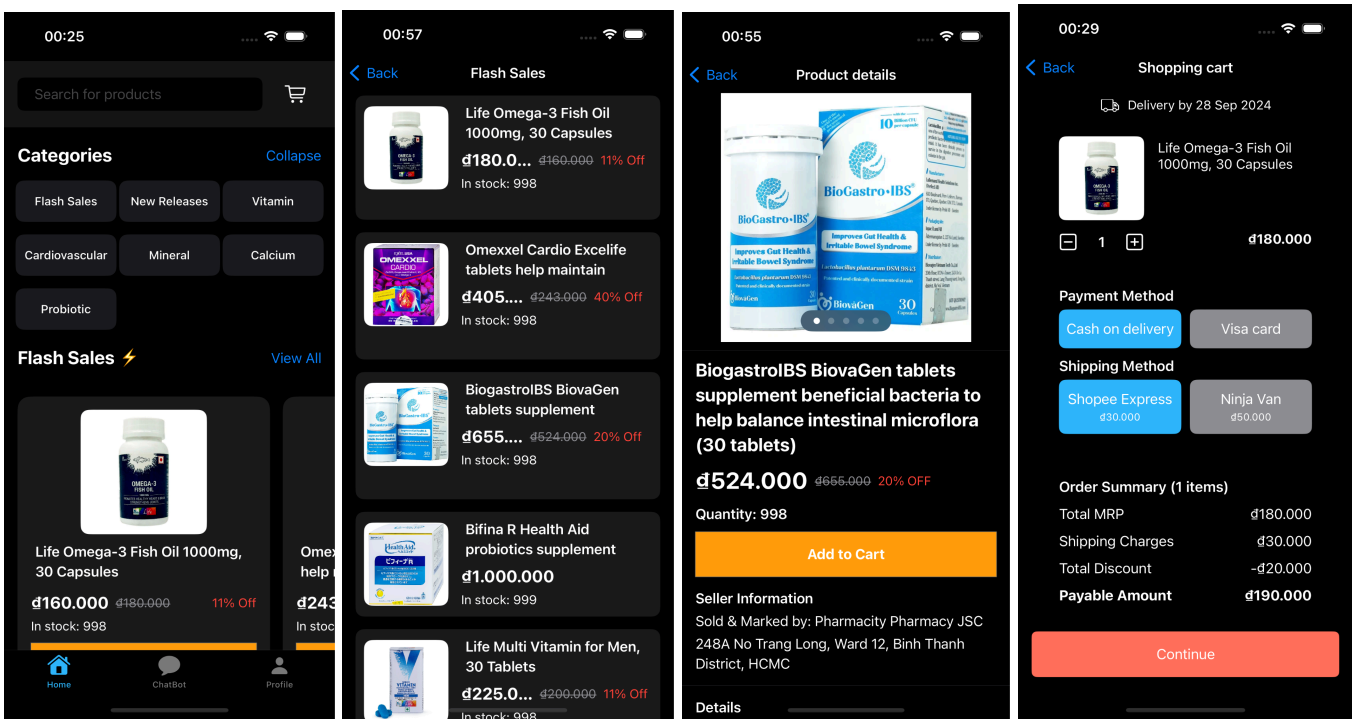
XII. Reference

[1] A. S. Gillis and S. Lewis, “What is object-oriented programming (OOP)?: Definition from TechTarget,” App Architecture, https://www.techtarget.com/searchapparchitecture/definition/object-oriented-programming-OOP (accessed Sep. 18, 2024).

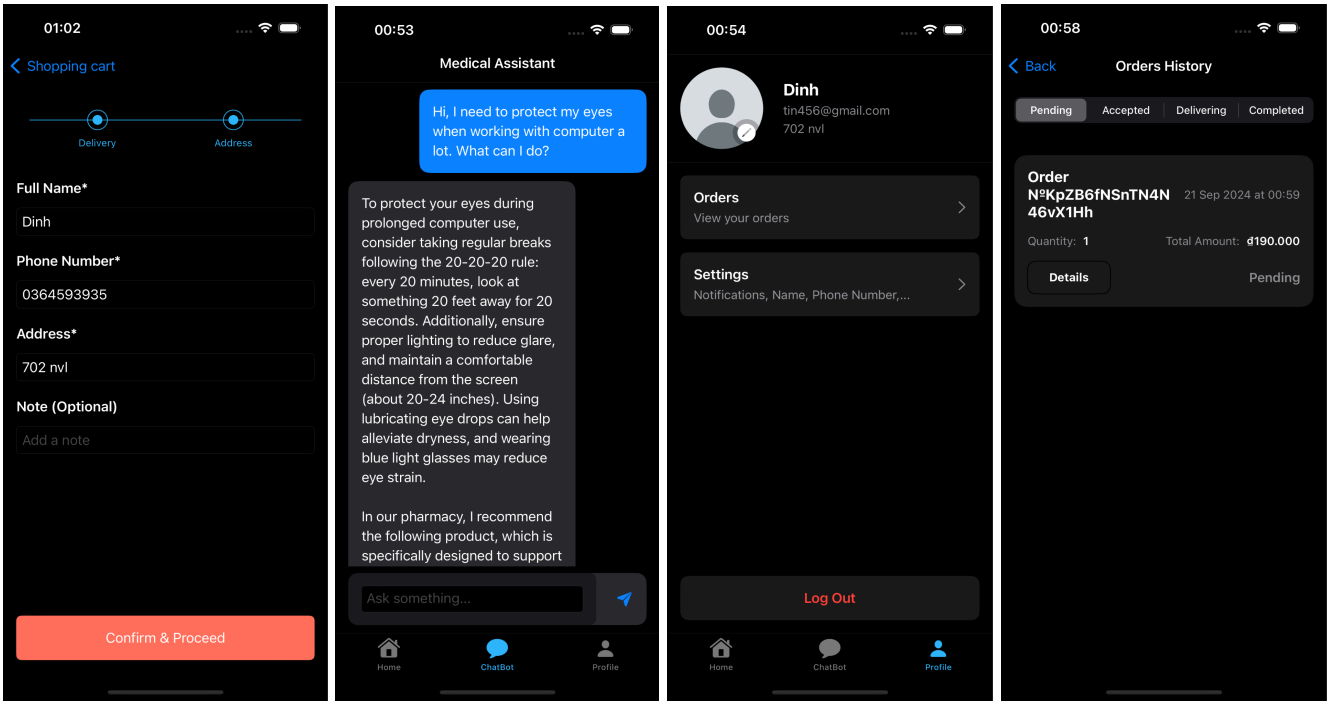
[2] M. Upadhyay, “Third normal form (3NF),” GeeksforGeeks, https://www.geeksforgeeks.org/third-normal-form-3nf/ (accessed Sep. 18, 2024).

XIII. Appendices

Appendix 1: Home View + Category View + Product Details View + Card View in Dark Mode



Appendix 2: Cart Delivery View + AI Chatbot View + Profile View + Order History View in Dark Mode



Appendix 3: Application database and services design (draw.io)