The University of Hong Kong

COMP4801

Final Year Project

Interim Report

An All-in-one HKU App for Students

Team Member

Yau Chin Pang (3035689257)

Supervisor

Dr. T.W. Chim

Date of Submission: 21st January 2024

Abstract

To foster a warm and close-knit student community in The University of Hong Kong (HKU) and integrate various student-centric functionalities into a unified platform, this project aims to create a mobile iOS application, named “Mane”. The application is characterized by a modern, intuitive design and an intelligent chatbot for information retrieval. The application is built with Swift and SwiftUI for the frontend, while employing Go, gRPC and Supabase for the backend. The backend would integrate OpenAI’s language model to power the chatbot, with similar context and information supplied to the model for auto-chat completions. Currently, the entire authentication setup and login process on the iOS App is completed, while more scraper, functionality and user interface are being added as well. The backend, written in Go, can be automatically built, and deployed after changes are made in the main branch of the codebase by using Dockerfile and Google Cloud Platform’s Cloud Build and Cloud Run services. The original plan was to co-operate with HKU’s Information Technology Services (ITS) to retrieve academic data. However, no replies have been received so far. As such, a web scraper is being implemented at the same time to retrieve student’s academic data. On the other hand, initial learning curve and setup process for both iOS app and the backend has taken more time than expected. As such, some features (with lower priority) may need to be cut from the final product. The development pace should speed up significantly in no time. Development of features on both Frontend and Backend platform are ongoing rapidly, with testing by other users (HKU students) commencing soon to help report any bugs or issues.

Acknowledgement

I would like to give my sincere gratitude to my supervisor Dr. T.W. Chim from HKU's Department of Computer Science. His support and guidance from the beginning of the project have been invaluable and laid a strong foundation for this project.

Special thanks to one of my personal friends who is studying in The Hong Kong University of Science and Technology to provide her university account's credentials to conduct reviews of "USThing" and "HKUST Student" application.

Table of Contents

List of Figures

List of Tables

Abbreviations

API	Application Programming Interface
GCP	Google Cloud Platform
GPT	Generative Pre-trained Transformers
gRPC	gRPC Remote Procedure Calls
HKU	The University of Hong Kong
ITS	HKU Information Technology Services
JWT	JSON Web Token
OS	Operating System
REST	Representational State Transfer
SIS	HKU's Student Information System
UI	User Interface
URL	Uniform Resource Locator
UX	User Experience

1. Introduction

This chapter provides an introduction to the project and outlines the progress reported herein. The background and motivation behind the project is discussed in Chapter 1.1. Objectives and deliverables for this project is presented in Chapter 1.2. Finally, a brief overview and the structure of the report is explained in Chapter 1.3 .

1.1 Background and Motivation

Enjoying a fulfilling campus life and being part of a warm community are essential aspects of a student's experience on campus. The vibrant and enjoyable activities organized by HKU's student societies contribute significantly to campus life. However, the promotion of these activities often lacks a centralized platform. Currently, students can stay connected with these societies by following their social media pages or sifting through numerous emails sent out by the societies every morning. This setup may deter students from exploring new interests or joining new societies.

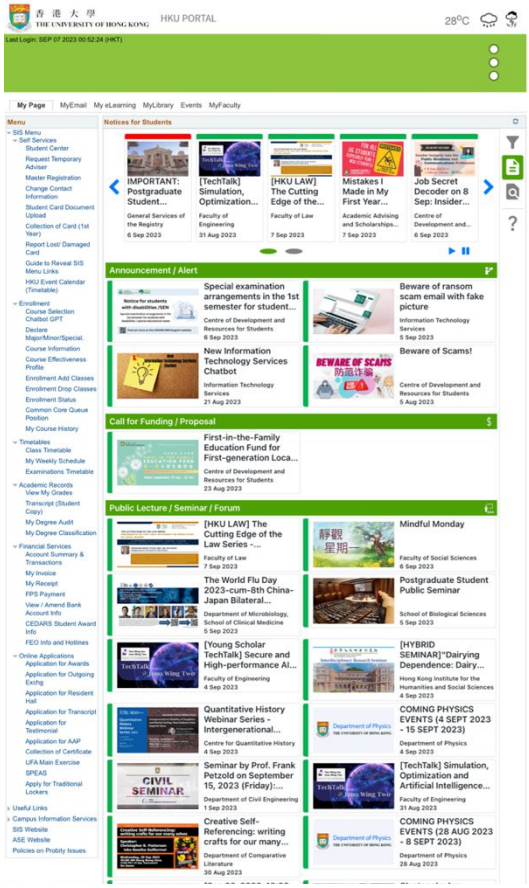

Figure 1. An smartphone view of the HKU Portal [1]

On the other hand, the University of Hong Kong provides only one platform - the HKU Portal, for students to access their academic data, course catalogue, and individual transcripts. Unfortunately, as depicted in Figure 1 above, the HKU Portal falls short in terms of mobile optimization and modern user interface design. It neither features a responsive design nor offers a distinct mobile site, making it challenging for students to check their data easily on their phones, especially since HKU does not provide a mobile application. Moreover, the process of booking facilities is quite cumbersome due to the lack of a centralized platform. Students often find themselves spending unnecessary time searching for the right resources online, which is both time-consuming and inconvenient.

In November 2022, OpenAI launched ChatGPT and unexpectedly soared in popularity, reaching 100 million users by January 2023. Its swift popularity was due to its sophisticated language understanding and generation capabilities, enabling more human-like and engaging interactions by generating different novel scenarios according to the given prompt [2]. This development has caused the emergence of chatbots that combined an organization's knowledge base with GPT-powered generation, offering personalized user experiences without requiring extensive application navigation. Developers have harnessed ChatGPT's advanced reasoning skills for comparing, extracting, and clustering diverse and different ideas, simplifying the creation of recommendation systems.

1.2 Objective and Deliverable

This project aims to address many of the problems outlined above by developing a mobile application, bolstered by a robust backend system to ensure speedy and secure operations. While some other universities in Hong Kong and globally offer mobile apps as alternatives, like USThing or PolyULife, this application aims to go a step further. Besides merely centralizing students' HKU academic data onto a single platform, the application also intends to introduce functionalities that foster tighter connections within the student communities. The planned functionalities for the application include:

1. An easy access of the user’s (student) data, such as transcript, course history, course grades or more.
2. Smart notifications to remind students of various event.
3. Book facilities with the application directly.

4. HKU course ratings or reviews.
    a. Only students who have completed the particular course are eligible to rate
    b. Leave comments, star ratings and what grades did they receive
    c. Tag the courses with simple categories such as high workload or "leng" (good) grade
    d. View the professor or lecturer's other courses as well
5. Current or future campus events:
    a. Student societies can publish their activities there so other non-members could join even though they are not a member.
    b. Students can sign up for the activities on the application directly.
6. Friendship pairing:
    a. Students can input the following for matching with or searching for other students:
        i. What sort of person they are looking for to be friends with
        ii. Common courses they have
        iii. Hobbies, sports
        iv. Other after-school activities
    b. They can browse the summarised list of other students
    c. The system will recommend similar students periodically
7. Lunch buddy pairing:
    a. Based on current location obtained via Global Positioning System
    b. Automatic filtering based on
        i. Dietary restrictions
        ii. Personal cuisine type preference
    c. Real-time chat feature for communication
8. Intelligent chatbot, powered by OpenAI language model
    a. Students can interact with the chatbot in a conversational manner
        i. Ask for general advice or just chit-chat
    b. The chatbot will utilize basic information of the particular student as context
    c. The chatbot will be able to reference scraped HKU data to ensure accuracy and avoid “hallucinations”
        i. Course catalogue
        ii. Exchange studies
        iii. Major, Minors

iv. Campus facilities

9. Security:
    i. Users will need to be authenticated to access the data
    ii. Encryption for sensitive data locally with Apple's Keychian
    iii. Secure and verified communication between the app and backend services
    iv. Minimal data storage in the backend database

1.3 Report Outline

The remainder of this paper is divided into three chapters. The first chapter describes the methodology used in this project, justifying and explaining the chosen tech stack to power the application. The second chapter discusses the current status of the project, followed by potential difficulties, concerns, and limitations. Finally, the third chapter concludes the paper, outlining the remaining work plans and immediate next steps.

2. Methodology

For the successful and secure deployment of the mobile application, the project requires a backend service to process requests from the application, manage authentication, database storage, and connection to OpenAI's API. A high-level overview of the system architecture is provided to give a general understanding of the architecture in Chapter 2.1. The procedures and justifications for building the app and the backend service are covered in Chapters 2.2 and 2.3, respectively.

2.1. Overview

The overall architecture is mainly divided by frontend and backend (see Figure 2). The frontend mainly consists of a native iOS application that would retrieve and update data via a scraper that are developing, whereas the backend is comprised of three different main technology. Supabase is used for authenticating users and host PostgreSQL, while GCP's Cloud Run is used to host the backend code, which is written in Go. The backend code then connects to OpenAI's APIs to provide intelligent functionalities to users.

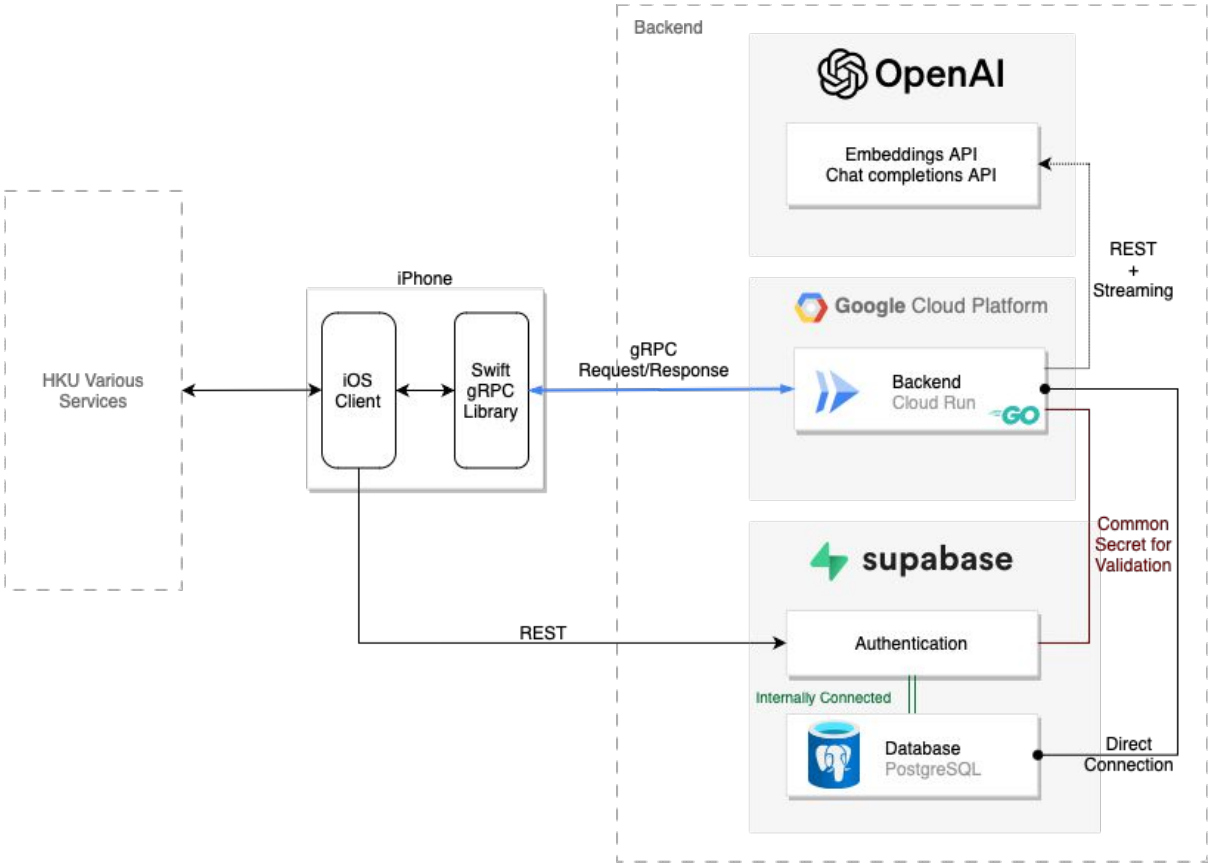

Figure 2. The overall system architecture design

2.2. Frontend

In the frontend (mobile application) side, the application is planned to be built natively on iOS. Although cross-platform app development tools do exist, native applications are still preferred due to their higher efficiency, deeper integration with the OS and the ability to connect to the native APIs [3], which can create a better UX to users. Comparing the two most prevalent mobile OS in Hong Kong from [4] (iOS and Android), iOS is more favorable to develop application on as S.Garg stated that "Android is more susceptible to security breaches and malware attacks" [5, p.13]. As one of the primary goal of this project is to build a smooth and efficient mobile application with security in mind, iOS is the better choice of mobile OS in this project.

Swift is the programming language, and SwiftUI is the environment to build the application natively on iOS via XCode. Apple offers UIKit and SwiftUI as the native UI framework [6], which allow developers to create UI in a simplified manner. UIKit is more customisable and robust as SwiftUI was launched recently in 2019. However, in [6], it stated that despite UIKit having better performance in most cases, SwiftUI can help reduce code length and accelerate development due to its declarative programming style. SwiftUI also offers modern features like Live Previews that can reflect UI changes immediately without the need to compile, making SwiftUI a great choice to utilise in this project to reduce development time. SwiftUI also provides a lot of convenient pre-sets like SecureField for inputting password or other Gradient background in a simple manner.

As for the architecture of the iOS app, Model View ViewModel (MVVM) design pattern is being chosen. MVVM design pattern separates functionality to help reduce confusion on where the source of truth is and separate business logic from UI, while using ViewModel (VM) has the logic to prepare the data for View object (i.e. the screen or UI) [7]. MVVM compacts the based code and makes it flexible [7]. For the VM, a new framework Observation, that came out in iOS 17, is used as it helps maintain a list of observers and notify them of specific or general state change [8], like "publishing" updates to a view when a variable's state changes. This framework is chosen as it came out recently and it is expected Apple would provide a longer support for this framework compared to older frameworks like ObservableObject. So fewer efforts are needed to maintain this project in the long term. Moreover, this project would also aim to use latest SwiftUI/Swift frameworks and features

like Navigation Stack or Sheet to provide a near-native iOS experience that is align with latest iOS trend, while minimising efforts needed to maintain in the long term as well.

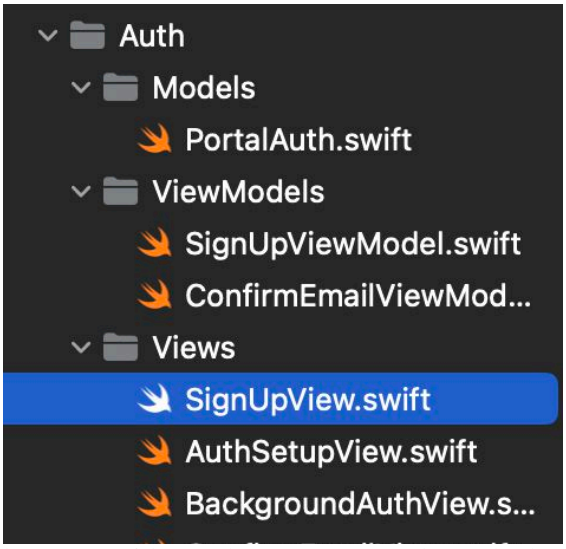

Figure 3. The file structure of the XCode project, structured accordingly to MVVM pattern.

Lastly, the project would try to make use external Swift libraries online to save time as these libraries usually provides simpler wrapper to use native iOS APIs, reducing learning curve. Alamofire is used to send or receive HTTP request, while AlertToast are used to create popup or toast alerts. KeychainAccess library is also used to simplify the use of Keychain.

2.3. Backend

2.3.1. gRPC

The connection between the client (iOS application) and the backend is facilitated by gRPC. gRPC, aided by protocol buffers, is an alternative way to exchange information between two systems. In comparison to the conventional approach of using JavaScript Object Notation (JSON) schema with RESTful API, gRPC with protocol buffers exhibits higher speed in certain scenarios, offers idiomatic client libraries in both Go and Swift [9], which ensures the client request is valid and the data types are correctly matched. gRPC provides easier developer experience as the client libraries allow the code to exchange information by just calling a function. Since the current plan is to only serve a single client type (iOS) and do not need to provide public read access, gRPC is more suitable for this project.

2.3.2. Cloud Run Service

Go is used to implement the backend service. The top 3 commonly-used programming languages used by professional developers that can be used as backend typically are JavaScript, TypeScript and Python [10]. However, they lack enforced type system checking built-into their roots, making them less safe compared to typed languages such as Go. Other

programming languages like Java or C++ are a bit dated in terms of their developer experience, syntax and lack of modern features.

GCP Cloud Run, hosted on Google's own cloud, is going to host the backend service. Cloud run is a serverless compute platform that allows developers to deploy code with auto-scaling, and the cost is request-based [11]. Compared to similar service from its competitor like Amazon Web Services, it is easier to deploy and manage scaling. Besides, it has full support for gRPC's requirement of utilizing HTTP/2. To make sure the cloud run service can authenticate the user request, a secret, a private key to check the signature of a message is valid or authenticated or not, can be obtained from Supabase. The cloud service can then access HKU's system to obtain and retrieve relevant information via a web scraper developed internally.

2.3.3. Supabase

Supabase is a platform that provides authentication, PostgreSQL database and bucket storage to its users to simplify backend service. Supabase packages a PostgreSQL database with its authentication product, whereas other alternatives like Firebase come with proprietary database solution. A significant advantage of utilizing a PostgreSQL database is its support for storage and indexing of vectors in a column through an extension called "pgvector." Vectors can be used to aid GPT models to retrieve similar contexts and help generating more precise responses. Supabase provides client libraries to Swift as well, so the iOS application can authenticate with Supabase directly without the need to implement an authentication service in our own cloud, which could mitigate potential bugs that may cause data leakage.

2.3.4. OpenAI APIs

For the chatbot to provide contextually relevant and accurate information with customized chat generation, the project will scrape various HKU resources such as exchange studies information, course details, timetables, academic offerings like available majors or minors, and campus facilities. Upon data retrieval and segmentation into different categories, the data will be fed into OpenAI's Embeddings API to convert them into embeddings—a vector list of floating-point numbers—which will then be stored within the Postgres database. These embeddings quantify the relatedness of text strings [12] and can be utilized to calculate the "distance" (i.e., the relatedness) between the user query and the actual data in the database.

This mechanism facilitates search, recommendation, and data clustering for the GPT model, enhancing the analysis and chat generation processes.

The project uses the Chat Completions API to generate conversations with the language model. Each of the messages sent to the API is assigned with roles as either "system," "assistant," or "user." The "assistant" and "user" roles correspond to the GPT model's messages and the actual user (student) respectively, while the "system" role serves to dictate the model's behaviour and provide specific instructional directives regarding response formatting or requirements. The "system" role will be used to provide relevant context for the GPT model to analyse and specify the appropriate format for message delivery, enriching the contextual understanding.

3. Results and Discussion

This chapter lays out the completed and ongoing tasks at the current stage. A brief update, including the status and completed research, is discussed in chapter 3.1. Limitations and potential difficulties are discussed in chapter 3.2.

3.1. Project status

The source code of the project can be found here. This includes the source code of the iOS app, backend services and protobuf definition.

Prior to starting development of the iOS App and backend services, research has been done via Postman (see Figure 4), an API platform with an API client to explore, debug, and test APIs [13], and various web inspector in different browsers, that can monitor network request and html file structure of the webpage, to study how the web browser communicate with HKU Portal or SIS's backend server. In the Postman API Client, request to SIS Server and response from the server are saved and studied to see how to extract appropriate info or login to the system. HKU portal also uses cookies to track user session. Alamofire is able to manage cookies automatically, so no further operations are needed for setup.

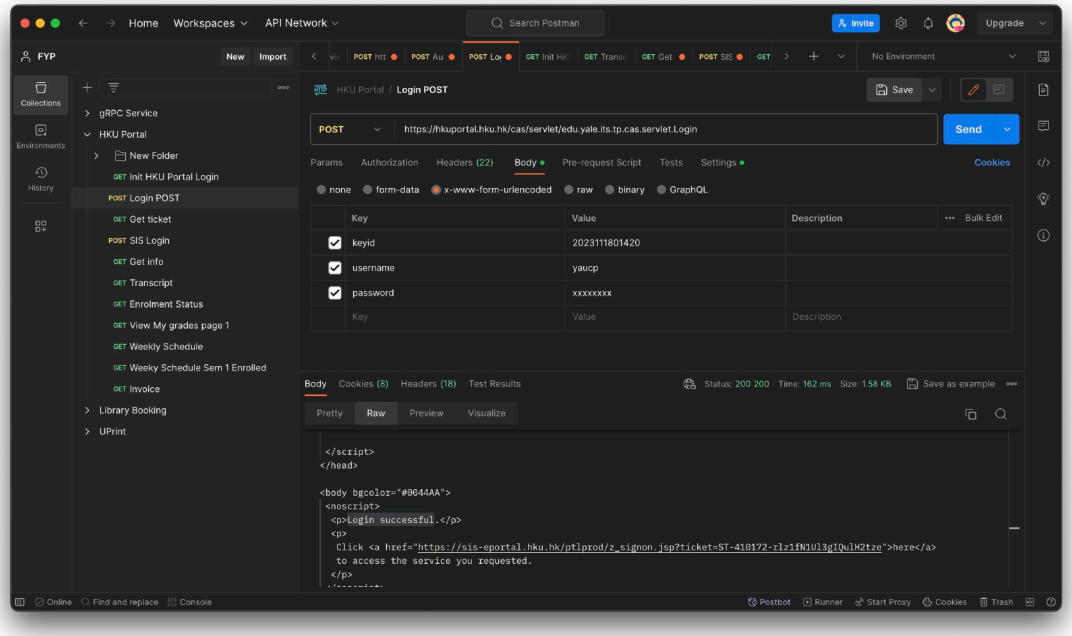

Figure 4. API Client in Postman.

It is used to test the correct request to send and the expected response that the scraper should receive.

Currently, research and exploration on how to sign in to HKU Portal, SIS and getting basic data like name, university number, transcript, grades and schedules are done. URLs for each endpoints are recorded as well.

3.1.1 Current Progress – iOS App

At the time of writing, the entire authentication process has completed. This includes signup, login, and initialization of the app itself when it is launched. When the app is launching, a *UserManager* instance is created and it would first go through Apple's Keychain services, which is an official API from Apple to store small bits of secrets or data in an encrypted database [14], and check whether a JWT is saved locally previously. These JWT are issued by Supabase previously when a user logged in. Whenever a user login, a JWT comprising of access token and refresh token is issued to the client. These tokens are used to verify user's identity and whether they are authenticated or not. Since they are stateless and are signed by Supabase, these tokens can be stored locally to help retrieve a new session whenever the user relaunches the application in the future. If no JWT are stored in Keychain (i.e. user has logged out or this is the first time the user has launched the app), the app would navigate to AuthSetup view (see the left of Figure 5) to let users sign up or login to our service. Otherwise, the user would navigate to Home view directly.

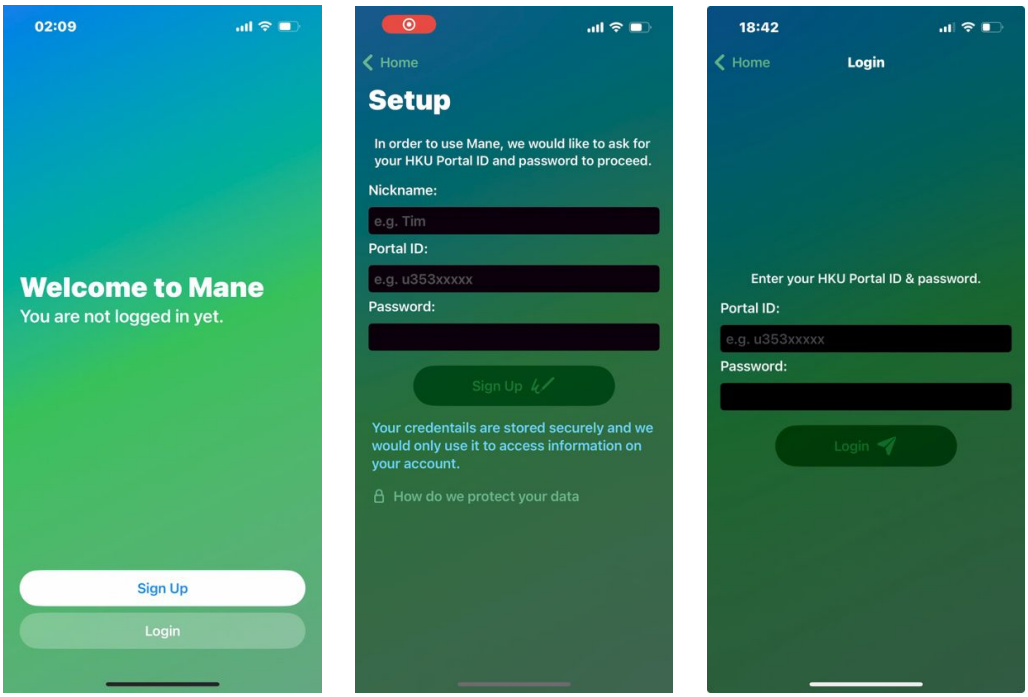

Figure 5. AuthSetup View (left) Signup View (Centre) Login View (Right)

User can then proceed to sign up view (see the centre of Figure 5) or login view (see the centre of Figure 5). In sign up view, user need to input their nickname, HKU portal id and

password as the scraper requires the user's credentials to access HKU portal and SIS on their behalf. After these fields are validated via several pre-set rules (like password length restrictions etc.), a scraper in the iOS app would login to the HKU portal with the given credentials in the background to validate them. If it is indeed able to login to HKU portal, it would also sign up the user in Supabase. Due to Supabase's security recommendation, an email with an one-time code is sent to user's HKU connect email and they would need to enter the code in the app (see Figure 6) as well to complete the signup process. The email is sent via Amazon Web Services' Simple Email Service with a bought domain (yaucp.dev).

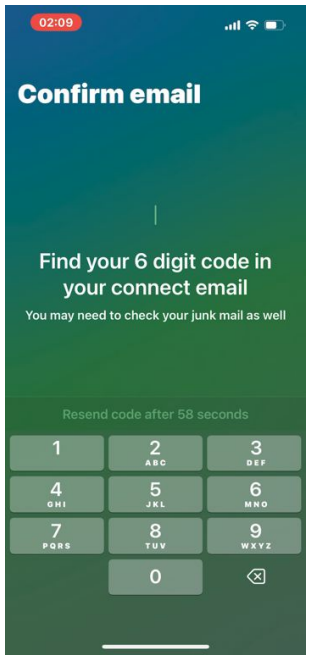

Figure 6. The view for user to enter the one-time code sent to their HKU connect email.

Various alerts or popup is also used to indicate something went wrong, e.g. wrong credentials or user has signed up before (see Figure 7).

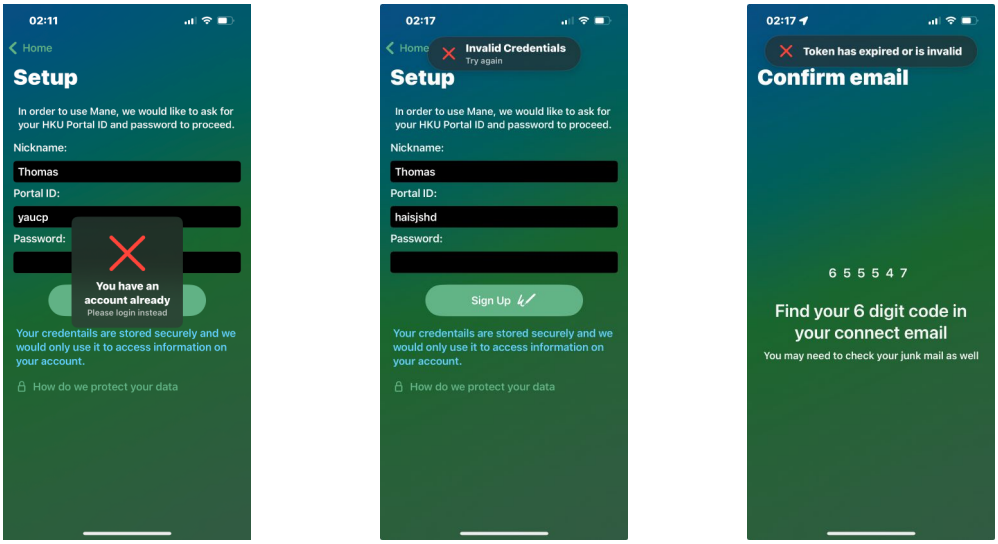

Figure 7. Various alerts or popups to alert user.

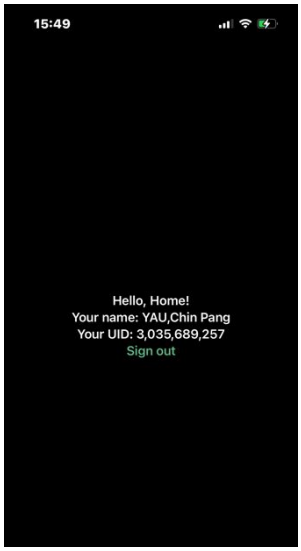

Figure 8. The Home View after the local scraper has logged in and scraped basic info from the received HTML file at the endpoint.

```swift
@Observable class HomeViewModel {
    var userInfo: UserInfo?

    init() {
        portalScraper.resetSession()
    }

    func initialLoginToSIS(using userManager: UserManager) async {
        guard let portalId = KeychainManager.shared.secureGet(key: .PortalId),
              let password = KeychainManager.shared.secureGet(key: .PortalPassword) else {
            print("Portal id or password doesn't exist")
            return
        }
        let signedIn = await portalScraper.signInSIS(portalId: portalId, password: password)

        if userManager.isAuthenticated && signedIn {
            self.userInfo = await portalScraper.getUserInfo()
        }
    }
}
```

Figure 9. The Swift code for Home View's ViewModel

After the user logged in or finished the sign up process, the Home view (see Figure 8) would be shown. The *HomeViewModel* (see Figure 9) would use *PortalScraper* class instance's functions and sign in to SIS asynchronously. After verifying that the client is indeed logged in by checking "Last Login" keyword, it would parse the HTML page received using SwiftSoup in *PortalScraper* and look for basic info of the students, including name and university number that would be stored locally for further use, while returning to the VM to be shown in the Home view.

3.1.2 Current Progress – Backend Service

Several PostgreSQL tables are set up and running. However, at the moment, it is not being utilized as much as hoped to as the features that are being developed do not require them. Triggers and constraint has been used to allow Supabase to automatically insert a row of user data to *profiles* whenever the user signup for easier management. Other tables like *urls* (for

updating the URL endpoint for the scrapers), *course_reviews* or *classes* have been setup as well.

The gRPC service to facilitate communications between frontend and backend service is already running in GCP using Cloud Run, which can be scaled to 0 instances and help save cost. With the help of Dockerfile and GCP's Cloud Built service, the gRPC service is automatically deployed and built whenever the main branch of the codebase has new commits. Currently, the service has several functions like healthcheck and GetUpdatedURLs running but work is being done to help seamlessly connect the iOS app to the service. To define what functions do the gRPC service support and the schema for the requests or responses, protocol buffers (protobuf) is used (see Figure 10). Protobuf help accelerate serialization or deserializations and can auto-generate relevant client or service stubs or types so developers can use the auto-generated functions and types directly. The protobuf is stored in a separate repository so both frontend and backend can use the protobuf to generate language-appropriate client stubs.

```
syntax = "proto3";

import "google/protobuf/timestamp.proto";

option go_package = "service/pb";

service MainService {
  rpc GetUpdatedURLs(GetUpdatedURLsRequest) returns (GetUpdatedURLsResponse);
}

message GetUpdatedURLsRequest {
  optional google.protobuf.Timestamp versionTimestamp = 1;
}

message GetUpdatedURLsResponse {
  message URLsList {
    google.protobuf.Timestamp versionTimestamp = 1;
    map<string, string> URLs = 2;
  }
  optional URLsList latestURLs = 1;
}
```

Figure 10. Part of the protobuf for MainService.

3.2 Limitations and Difficulties

The original project plan was to co-operate with ITS to retrieve's students' data from them directly. After sending emails to ITS asking for permission in September and October, no replies have been received apart from an automatically generated email. As they have not replied before the internal deadline of 31st October 2023, the project would instead utilize a web scraper developed internally to retrieve data. Some functionalities like booking facilities

may not work as expected. Research on how to develop an efficient web scraper are still undergoing, which may require some extra time.

Second, the initial setup has taken more time than expected. As SwiftUI was released a few years and there were quite a lot of new frameworks and changes made every year, tutorials and documentations on the newer technologies are quite scarce compared to other tools like Flutter or React Native. There were a few rewrites of the Swift/SwiftUI as the old methods were using hard-to-maintain and understand design patterns, so quite a bit of time is spent on reworking the structures and which frameworks to settle on. Dockerfile took a bit of time to configure to deploy the update automatically due to time needed to understand how Dockerfile works. Development pace should speed up soon as most of the setup work has completed.

3.3 Next Steps

The project is a bit off from the original schedule due to the reasons mentioned above. Development of scrapers (course schedule, transcripts, facility bookings) for the different endpoints are still undergoing, with UI is being developed alongside to present the scraped data to the users in a clear manner. After web scrapers were developed, the next phase would move to developing course reviews or campus events features. More focus will be put on developing backend service for these 2 features as they do not rely on external services like HKU. Due to tight time schedule, some lower-priority features like friendship or lunch buddy pairing may be cancelled or scaled down, whereas the GPT-powered chatbot would still remain a high priority. Development for the chatbot would start approximately at mid-Februray by scraping HKU data and transforming the data into embeddings. The new schedule can be found in the next page.

Period	Deliverables and Milestones
December 2023 – January 2024	**Implementation:** – Starting implementation of iOS app – Implementing backend services (web scraping – ongoing and facility bookings)
February 2024	**Implementation:** – Scraping HKU-related data and create embeddings (Exchange studies, courses data) – Implementing backend services (Course reviews/ratings, campus events and friendship/lunch buddy pairing)
March 2024	**Research:** – Prompt engineering (Training GPT-4 model to match similar people or personality and answering HKU-related data**)** **Implementation:** – Implementing OpenAI APIs (Moderation, creating embeddings, and chat completions) – Continue implementing backend services
April 2024	**Final Wrap-up:** – Bug-fixing and improving software products based on tester's feedback – Finalizing report, presentation, website and product

Table 1. Upcoming project plan schedule

4. Conclusion

This project aims to build a comprehensive iOS application to help bring HKU student community together and make looking up their own personal information easier than before and bringing the student community together. The iOS application is built with Swift and SwiftUI, chosen for their simplicity and ease of use for developers. The complete authentication (signup, login) process and the get basic personal info scraper is completed. The backend, written in Go, will connect to the application via gRPC and protobuf. The database tables' schema are preliminary designed and set up, whereas the gRPC service can be deployed and built automatically with Dockerfile. However, some extra times are spent on transition to using web scraper due to lack of reply from ITS. Initial setup time also takes more effort and time than expected, causing the schedule to shift. Some lower-priority features may need to be scaled down or cancelled. Development of other web scrapers are undergoing. The next phrase involves developing campus events and course reviews features, which primary focus will be put on backend service.

References

[1] “HKU Portal,” hkuportal.hku.hk. https://hkuportal.hku.hk/

[2] A. Chow, "How ChatGPT Managed to Grow Faster Than TikTok or Instagram," Time, Feb. 08, 2023. https://time.com/6253615/chatgpt-fastest-growing/.

[3] K. Shah, H. Sinha, and P. Mishra, “Analysis of Cross-Platform Mobile App Development Tools,” *IEEE Xplore*, Mar. 01, 2019. https://ieeexplore.ieee.org/abstract/document/9033872/

[4] S. Garg and N. Baliyan, “Comparative analysis of Android and iOS from security viewpoint,” *Computer Science Review*, vol. 40, p. 100372, May 2021, doi: https://doi.org/10.1016/j.cosrev.2021.100372.

[5] StatCounter Global Stats, “Mobile Operating System Market Share Hong Kong,” *StatCounter Global Stats*, Sep. 2023. https://gs.statcounter.com/os-market-share/mobile/hong-kong (accessed Oct. 24, 2023).

[6] B. Ronneling, “Performance analysis of SwiftUI and UIKit : A comparison of CPU time and memory usage for common user interface elements,” KTH Royal Institute of Technology, 2023. Accessed: Oct. 24, 2023. [Online]. Available: https://www.diva-portal.org/smash/record.jsf?pid=diva2%3A1789094&dswid=-9772

[7] M. Aljamea and M. Alkandari, “MMVMi: A Validation Model for MVC and MVVM Design Patterns in iOS Applications,” Aug. 2018. Available: https://www.iaeng.org/IJCS/issues_v45/issue_3/IJCS_45_3_03.pdf

[8] Google, “What Is gRPC?,” *gRPC*. https://grpc.io/docs/what-is-grpc/

[9] Apple, “Observation,” Apple Developer Documentation. https://developer.apple.com/documentation/observation (accessed Jan. 20, 2024).

[10] Stack Overflow, “Stack Overflow Developer Survey 2023,” *Stack Overflow*, May 2023. https://survey.stackoverflow.co/2023/

[11] Google Cloud Platform, “What is Cloud Run | Cloud Run Documentation,” *Google Cloud*. https://cloud.google.com/run/docs/overview/what-is-cloud-run

[12] OpenAI, “OpenAI API,” *platform.openai.com*. https://platform.openai.com/docs

[13] Postman, “Postman API Platform | Tools,” *Postman API Platform*. https://www.postman.com/product/tools/

[14] Apple, “Keychain services,” Apple Developer Documentation. https://developer.apple.com/documentation/security/keychain_services/ (accessed Jan. 20, 2024).