

**Team 18 – Melbourne Backpack**



# **FINAL REPORT**

**ISYS2101 Software Engineering Project Management**

**May 22, 2022 • RMIT University Vietnam**

**Members:**

**Tran Ngoc Anh Thu (s3879312)**

**Doan Yen Nhi (s3880599)**

**Du Duc Manh (s3878480)**

**Nguyen Hoang Linh (s3880313)**

**Instructor: Anna-Lyza Sancho Felipe**

# Table of Contents

<b>1. PRODUCT DESCRIPTION &amp; FUNCTIONAL BLOCK DIAGRAM .....</b>	<b>3</b>
1.1. Product/System name: Melbourne Backpack.....	3
1.2. Notable product properties .....	3
1.2.1. Non-functional requirements.....	3
1.2.2. Functionality.....	4
1.3. Short functional description .....	5
1.4. Software Architecture.....	7
<b>2. TECHNICAL SPECIFICATIONS .....</b>	<b>8</b>
2.1. Use Case Diagram.....	8
2.2. Activity Diagram.....	10
2.3. Data Model .....	12
2.4. Technology .....	13
2.5. The specific algorithm used in the project.....	14
2.6. Requirements .....	17
<b>3. USER GUIDE (MANUAL).....</b>	<b>18</b>
3.1. Installation .....	18
3.2. Set up procedure.....	18
3.3. User guide.....	19
<b>4. ACADEMIC POSTER.....</b>	<b>33</b>

<b>5. A4 TRIFOLD BROCHURE .....</b>	<b>34</b>
<b>REFERENCES .....</b>	<b>35</b>
<b>APPENDIX: RESOURCES .....</b>	<b>36</b>

---

# 1. Product description & functional block diagram

## 1.1. Product/System name: Melbourne Backpack

The name suggests a free platform that packs all necessary information about studying and living in Melbourne into a package (thus the name "Melbourne backpack") and makes it available across all mobile devices. The app, designed by RMIT students who will be transferring to RMIT Melbourne, would be ideal for anybody looking for practical tips to help them research about this information.

## 1.2. Notable product properties

### 1.2.1. Non-functional requirements

- Mobility and adaptability

This app can be run on both tablet and mobile devices. Since our app is developed using React Native, a cross-platform framework, it is very adaptable and compatible with both Android and iOS; it can run on both platforms. Furthermore, the app is tested on both Android and iOS devices to fix any available issues on only one platform.

- Security

The app uses Firestore, Storage, and Realtime Database to store data, and our data is protected by writing security rules to the services above. The security rules will help us control who can access our data and how much access they have over it. According to our security rules, unauthenticated users (users that are not logged in) cannot perform CRUD operations on our data. Authenticated users can view all the necessary data, such as housing and transportation information; however, they cannot delete any data. There are some data collections that users can create or update. For example, users can create and update the "users" collection as they can create a new account or update their profile information.

- Availability

The availability of our app depends on Firebase as it is our backend service. According to Google Cloud [1], the monthly uptime of Firestore Regional is a minimum of 99.99%; hence, the availability of our app's backend is also 99.99% per month. According to our calculation below, the downtime per month would be around 0.073 hours per month.

Based on our calculation, Firestore will have a downtime of 0.0073 hours per month:

Total hours per year	$365 * 24 = 8760$
Total available time in hours	$8760 * 99.99\% = 8759.124$
Total downtime in hours	$8760 - 8759.124 = 0.876$
Downtime per month	$0.876 / 12 = 0.073$

#### – Maintainability

Due to the need for maintainability, we will spend 1 hour per 2 weeks for maintenance, which means 2 hours per month. During this period, we need to check the performance of all app functions and check the consistency of the connection between the mobile app and the backend. In addition, there will be a release for bug fixing and feature upgrades, if possible, every month. Therefore, we will need 30 minutes of downtime to deploy it every month. We will use A/B testing if there are no significant problems that can crash the app to not interfere with the user's experience.

#### – Response time

Regarding the response time, the data of all screens are retrieved from Firebase and displayed within 1 - 2 seconds. However, images usually take 4 - 6 seconds to be fully rendered on our app, as some images have high resolution. For creating or updating data, such as creating a new user or posting a review, these operations are completed within 3 – 4 seconds.

### 1.2.2. Functionality

Most of our features are completed, and the app is ready to use. The functions that will need to be optimized and completed later are the chat and add friend, as our current chat feature can only perform basic tasks like sending messages, while the add friend feature is currently using UID, which is not very secure or private. Furthermore, most of our

functions can behave correctly, except for the display of the trending suggestion. There is currently no algorithm to get the most trending suggestion, so the trending section in Housing, Shopping, and Transport is only displaying the first data in the collection and not the most trending one.

### **1.3. Short functional description**

- Authentication

Users can login with their existing account or create a new account using a valid email and password to use the app. If they forgot their password, they could also reset it through the Reset password screen and continue to login with the new password.

- Personalization and Profile

When users first create an account, after signing up with an email and password, they will need to fill in personal information like campus and subjects of interest and upload an avatar. This information will be used to display the user profile on their profile screen and help users find similar students to them within the community. Users can also edit their profile if needed, and at the bottom of this screen is a button for users to logout.

- Community

This functionality allows users to see other users in the community and find people similar to them. They can also use the filter to filter out users based on campus, interests, and purpose of using the app. To view more information about another user, they can press on that user's picture to view the other user's profile.

- Chat

When the user views another user's profile, there is a button to add a new contact so that they can chat with each other. They can send normal messages to each other, and if the connection is good, the user will receive the message immediately as we are using a real-time database for the chat data. The user can also access their chat history by pressing the chat icon on their profile screen.

- Explore

This functionality fetches and displays all YouTube videos related to studying and living in Melbourne from RMIT University's official YouTube channel. Users can browse through all the videos and watch them directly within our app; they do not need to open YouTube. We also provide a RMIT detailed information including navigated button to YouTube channel, description, and useful links.

- Recommendation

This functionality recommends housing, shopping, and transportation options near RMIT for RMIT students and displays some basic information about each option. For the transportation screen, there is a filter that can filter out the options based on the preferred transportation mode.

There is a filter based on criteria like price, distance from RMIT Melbourne campus, and the number of bedrooms or bathrooms for housing. Since housing usually has much information, users can view the details of each housing, such as average rating, description, and online reviews, by pressing the image or name of the housing to access its housing detail screen. There are two review sections on each housing detail screen: one section displays the reviews for this housing from the internet, and the other shows our app users' comments and ratings. To leave a review for housing, users can go to that housing's detail screen, enter their comments about the house, and rate the house on a scale of 0 – 5. After submitting, the new comment and rating will be displayed immediately on the housing detail screen. The average housing rating will also be re-calculated to display the new number.

## 1.4. Software Architecture

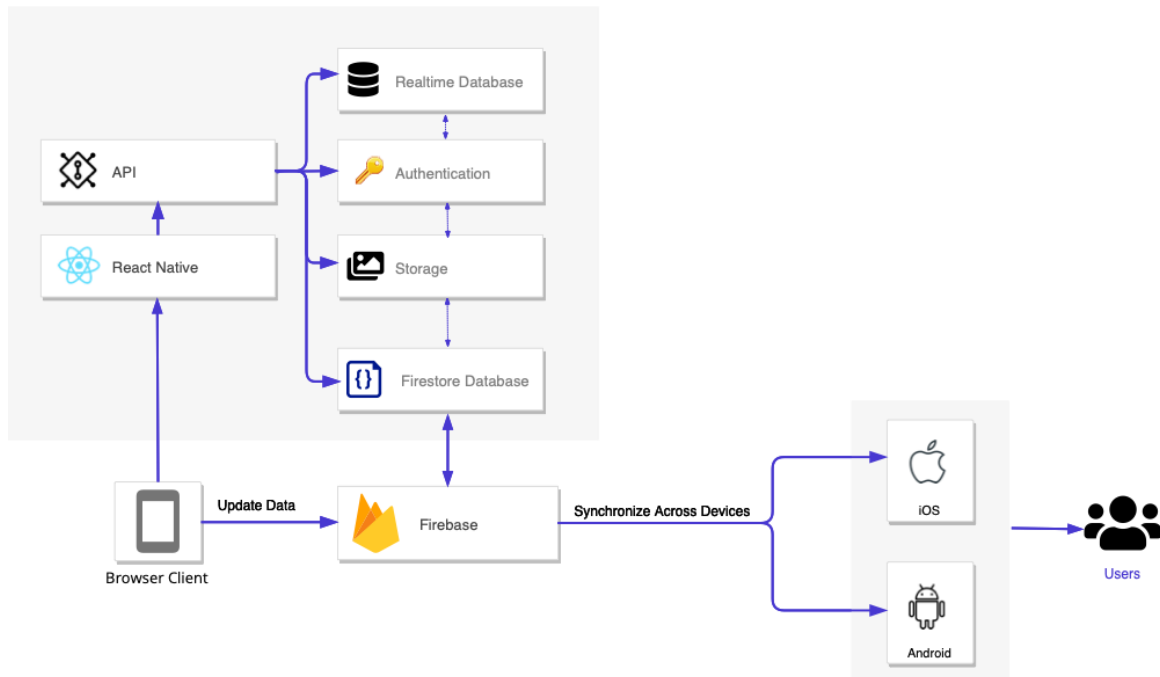


Figure 1: Software Architecture Diagram of Melbourne Backpack

Figure 1 depicts how the app's structural components interact with each other. Melbourne Backpack is a React Native app integrated with a Firebase backend. When the user interacts with the app that requires backend, the app will call the corresponding APIs to the backend services. Firebase acts as a "Mobile backend as a service" [2] application that provides Authentication, a Real-time database, a NoSQL database, and a Storage bucket. After receiving the requests, the corresponding backend service will perform the operations and return the results to our app. The results will then be rendered and displayed on the screen for users to see.



## 2. Technical Specifications

### 2.1. Use Case Diagram

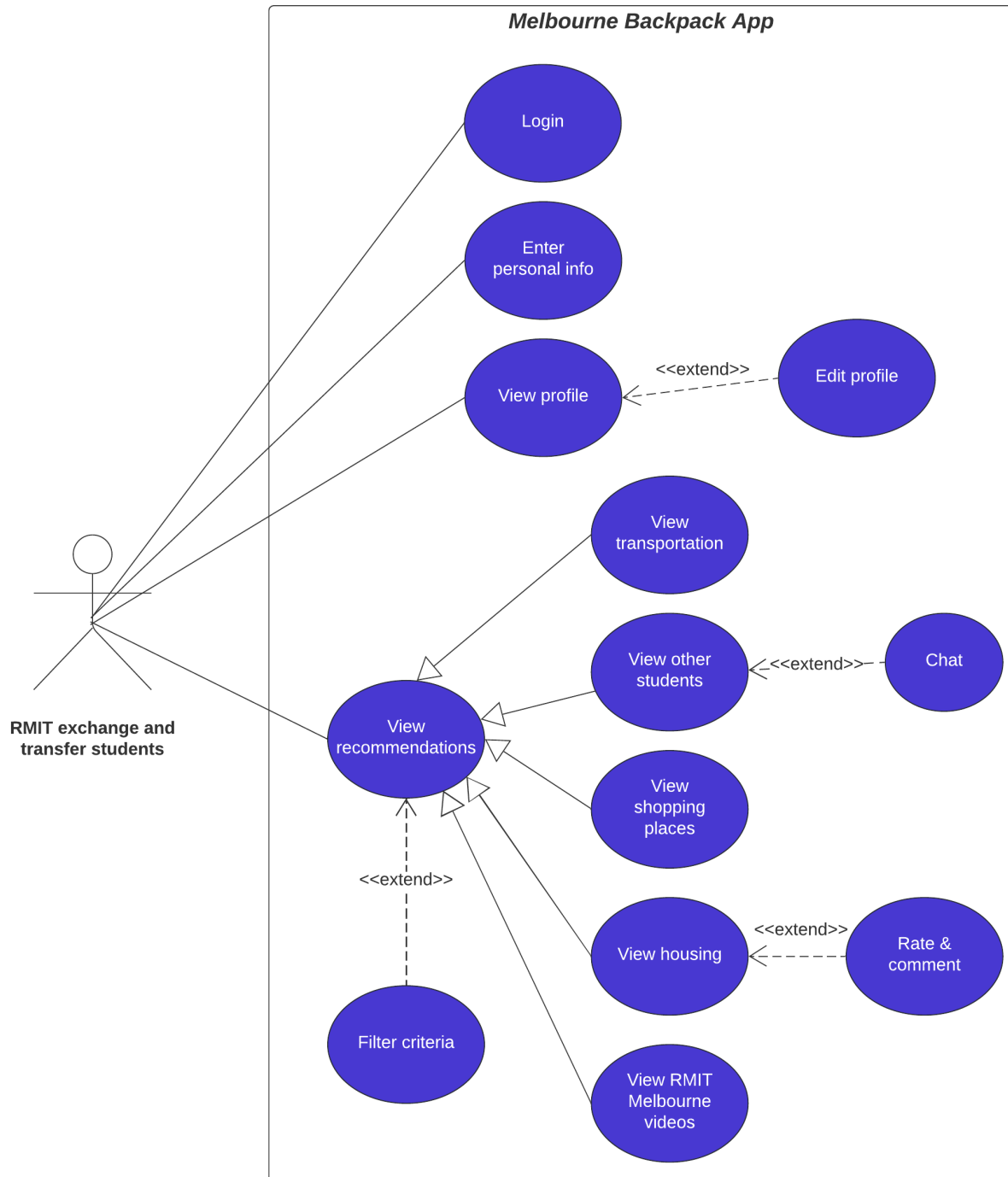


Figure 2: General Use Case Diagram of Melbourne Backpack

Figure 2 illustrates the high-level interactions between RMIT exchange or transfer students and our app. The main actors can login and enter their personal information, as well as view their profile and edit it if they want to. Students can also see our suggestions regarding accommodation, shopping places, available transportation, videos about life in Melbourne, and other students who want to study at RMIT Melbourne. Furthermore, the recommendations could be filtered based on specific criteria to find suitable suggestions quicker and easier. While looking at the Housing option, users can give star ratings and comments on the accommodation. In the community option, students can use our chat function to communicate with their peers.

## 2.2. Activity Diagram

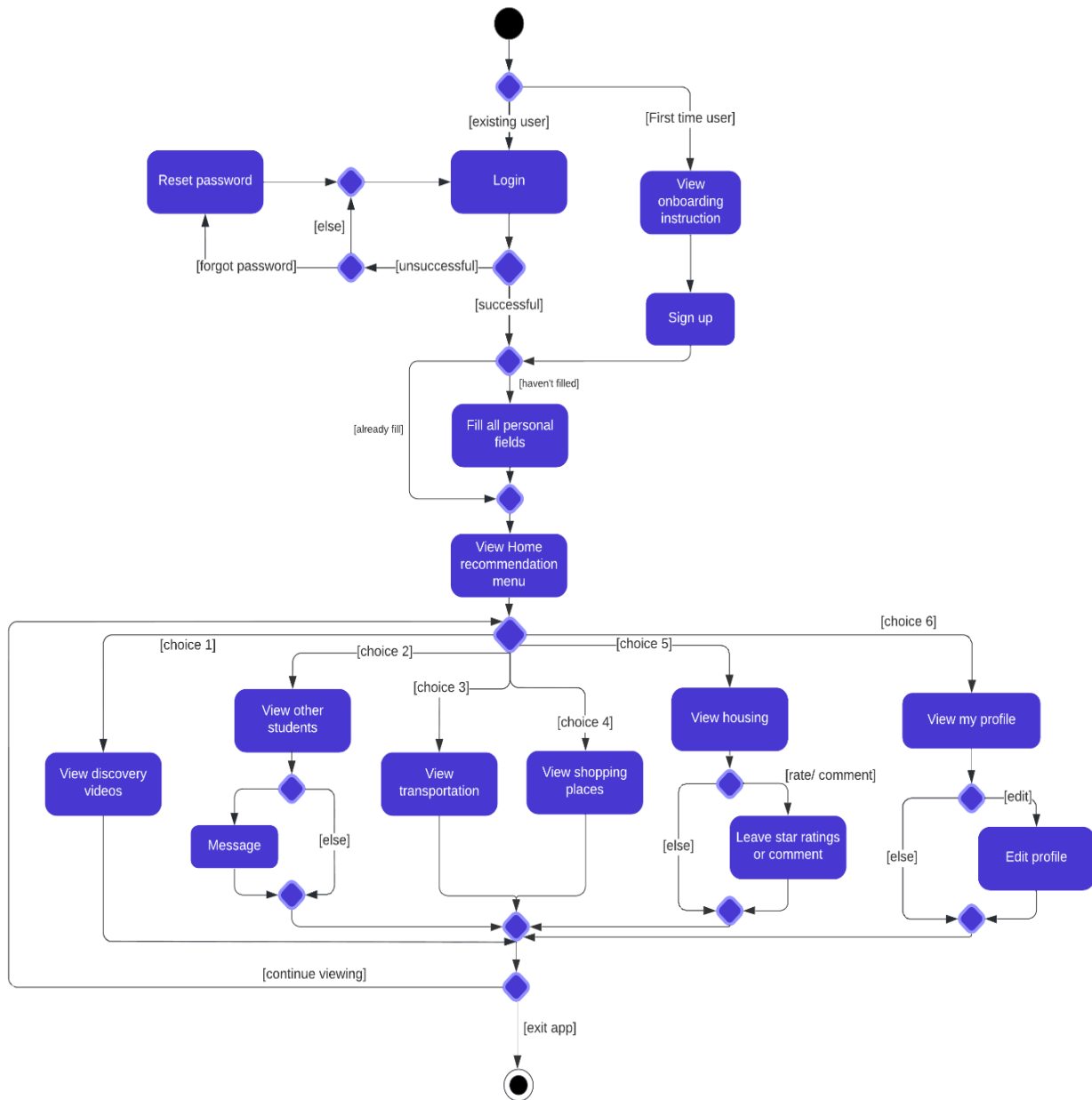


Figure 3: General Activity diagram of Melbourne Backpack

Figure 3 illustrates the activity flow of the whole app. To use the app, users need to have an account. If they have not created an account, the onboarding instructions will be shown to guide new users to use the app. After reading the instructions, they can sign up with a valid email and a strong password. Next, new users will have to fill in personal information like their bio, RMIT campus, subjects of interest, and their purpose for using this app.

After successfully filling all the fields, users will be redirected to the Home recommendation menu, where they will have access to other screens and functionalities of the app. If the user already has an account, they can login with their existing email and password; if they forgot their password, they could reset it and login with the new password. For existing users that have not filled in their personal information, they will have to fill it out before being able to access the Home recommendation menu.

On the Home recommendation menu, there are six options for users to choose from: Explore, Housing, Shopping, Transport, Community, and Profile. If users choose Explore, they can watch discovery videos about life in the city of Melbourne and RMIT Melbourne. If users want to see other students in the app, they can go to Community and view other students and their profiles and even chat with them if they want to. Pressing the Transport button will lead to a screen where users can view some transportation options near RMIT while pressing the Shopping button will help users see some shopping places in CBD (the university's area). Users also choose to view accommodation suggestions and their details and leave a review if they want to. The last option is Profile, where users can view their Profile and edit it if needed. After choosing an option, they can continue visiting other options using the navbar or exit the app.

## 2.3. Data Model

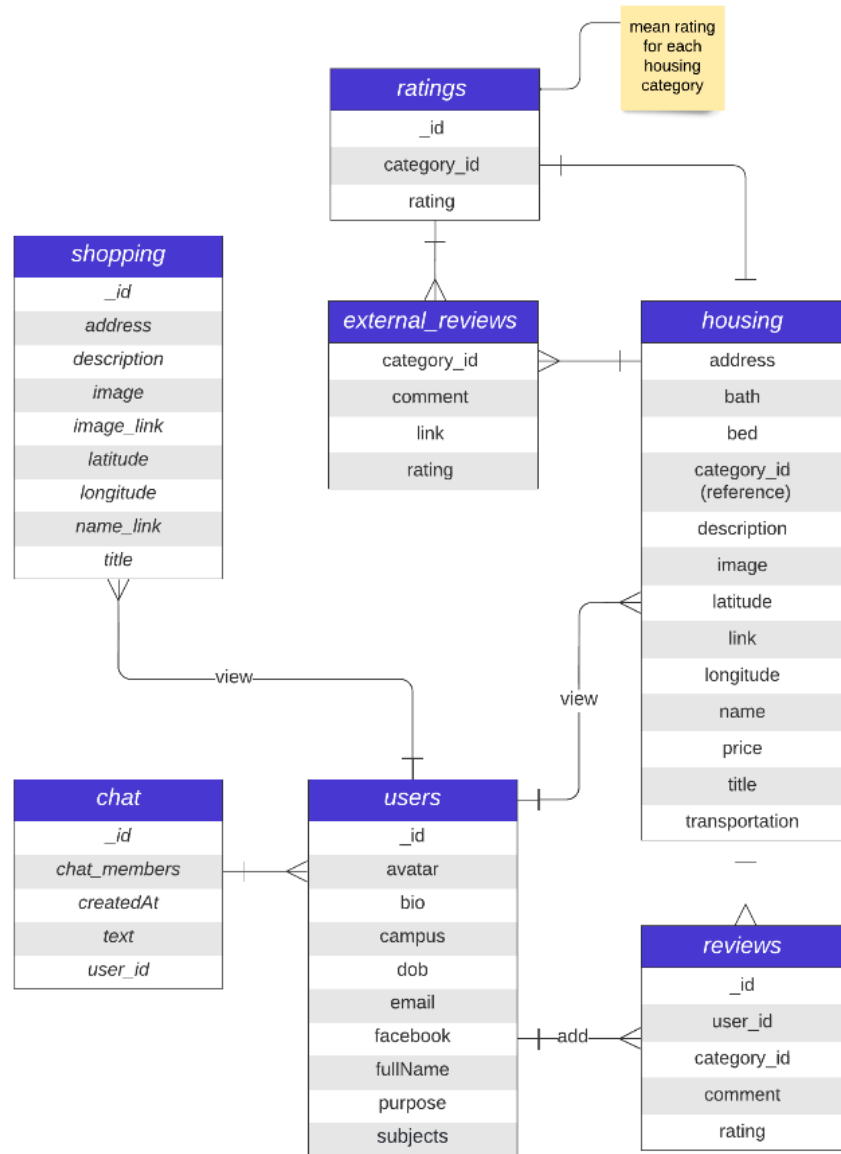


Figure 4: the Cloud Firestore Data model diagram in Melbourne Backpack

Firestore database does not have a rigid schema like other relational databases like MySQL. It has a data model that is exclusive to its category. Document databases may hold many collections in a single document. As the name indicates, our key-value database offers a straightforward data model. We have seven collections in the database, which corresponds to the seven tables in Figure 4 and the relationship between them.

Housing collection contains the cleaned data scraped from Melbourne students living. Each housing can have several reviews and external\_reviews collection representing ratings from internal users and other websites, respectively. The distinct rating collection indicates the mean calculated star ratings displayed on each accommodation category. Shopping collection includes data scraped from several shopping malls in CBD Melbourne. Chat collection comprises messages between the app's users inside the community forum. Finally, the users collection has users that have already signup and authenticated in the app via Firebase authentication.

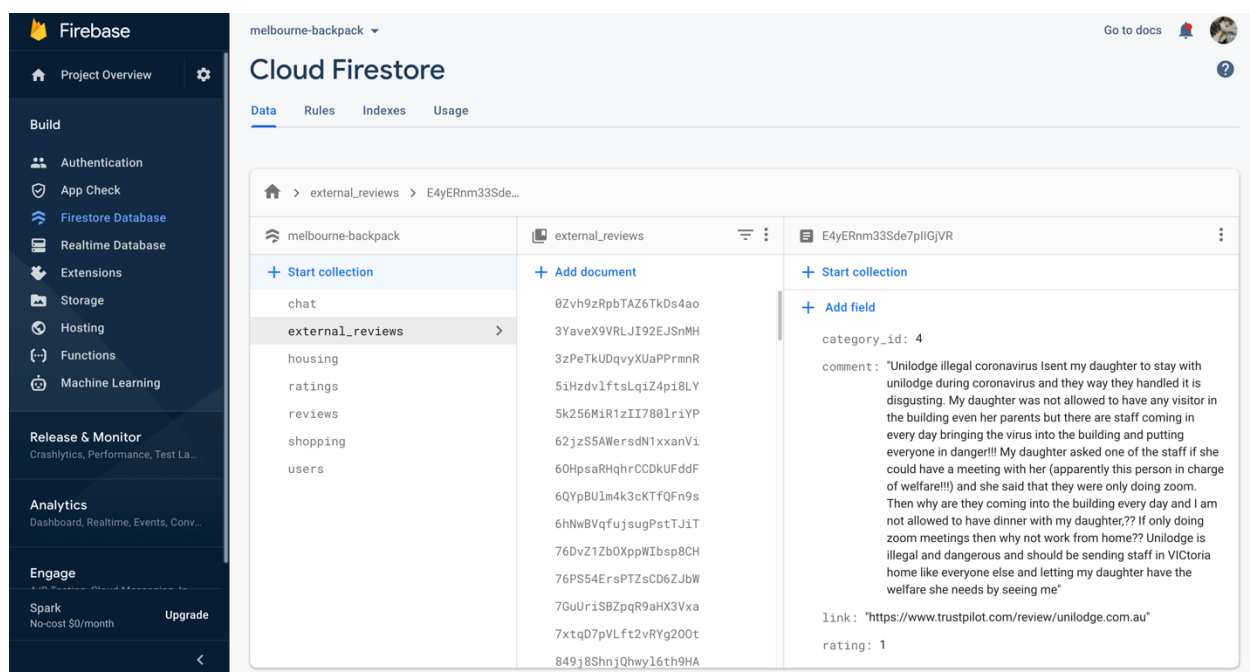


Figure 5: Screenshot of a field in external\_reviews collection

## 2.4. Technology

- Front-end:

Tools	Description
<b>WebStorm [3]</b>	We prefer to use a full-featured integrated development environment (IDE) to speed up the creation of mobile apps. WebStorm analysis identifies language and runtime issues and recommends fixes and enhancements. It also indexes our whole project and can find unneeded methods, variables, and other items.
<b>React Native</b>	We use React Native as the framework to develop our application since it is easier to construct, use, and maintain than Hybrid apps and less

<b>[4]</b>	expensive than Native apps. Most importantly, most of us have a web programming background in React, so it is easier for the team to migrate to React Native.
------------	---

- Back-end

Tools	Description
<b>Firestore</b> <b>[5]</b>	We chose Google Firestore as our go-to backend development platform because of various free services and technologies, including authentication, NoSQL database, storage, and real-time database, which is ideal for developing mobile apps.
<b>Jupyter Notebook</b> <b>[6]</b>	We use Jupyter Notebooks to scrape, extract, transform and load the data and machine learning model in a document-focused environment and then push it on Firestore. Jupyter notebooks show the analytic process in detail by arranging items such as code, photos, text, and output in a logical order but also aid us in documenting the thought process while working on analysis.

- Deployment

Tools	Description
<b>Expo</b> <b>[7]</b>	We use Expo to deploy because of the rapid development and testing features. Expo is a React Native toolchain that makes it simple to run apps. There is no native code, so we do not need to submit every update to the App Store or Google Play, saving extra costs.

## 2.5. The specific algorithm used in the project

First, our group implemented the natural language processing algorithm, Bidirectional Encoder Representations from Transformers, to understand what words in each scraped comment mean, but with all the nuances of context. Then, we apply the Naive Bayes algorithm to classify them into our defined star rating system. The extracted data is uploaded to the Firestore Database as dim external reviews and treated as a data warehouse for the housing fact feature.

In the housing and transportation screens, we allow users to select options in the filter by pressing buttons. Each button represents an option of the items' attribute on the screen. The filter for each screen is an object with matching attributes' names of the items on that screen. Each attribute will be an array of strings. The option will be pushed to the array when a button is pressed according to the attribute. After pressing multiple buttons, the

user can press the "Apply" button to apply the filter. The filter will check through the selected options one by one. Each item will be looped through to see if there are any attributes has the matching value to a selected option or has the selected option inside its array. If the button defines a range of numbers, the matching item's attribute will be checked if it is within that range. The distance of each item in the housing screen is stored in a list outside each item. Therefore, the value with the matching index to the item will be checked. Each item needs to satisfy at least one option per attribute that has at least one selected option to be selected after applying the filter.

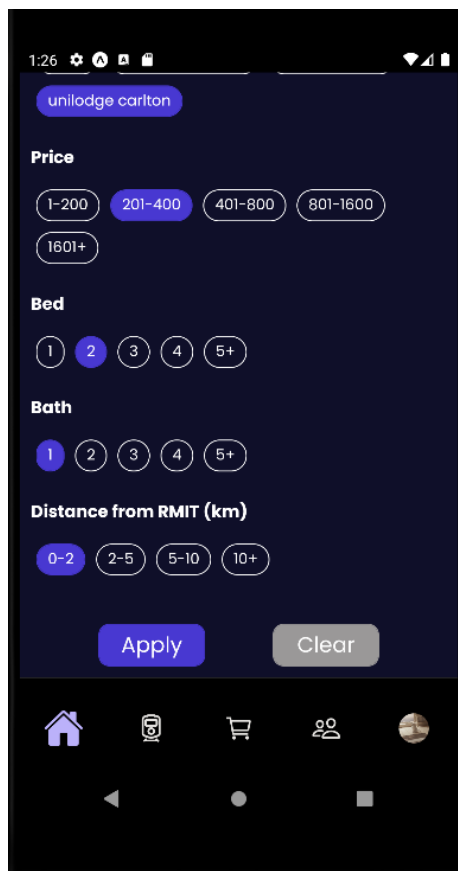


Figure 6: Housing filter

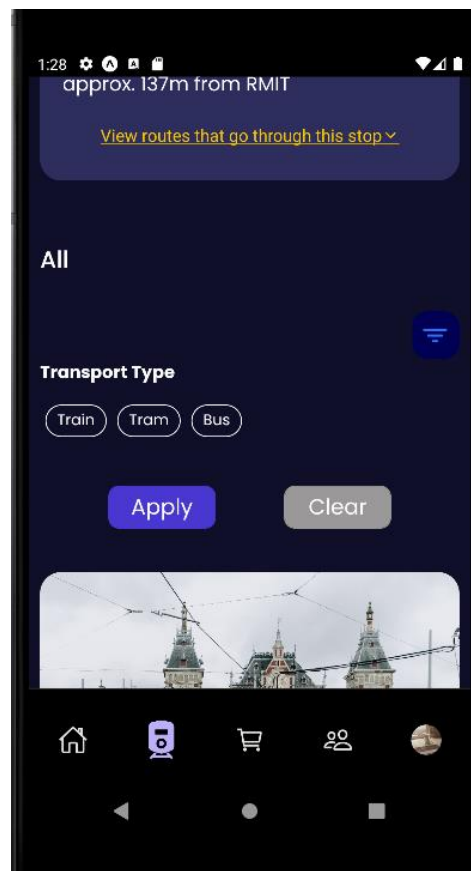


Figure 7: Transportation filter

There are reviews and ratings for each housing category for the housing screen. The average rating of each housing category is calculated using both the app users' reviews and external reviews from the internet. The program fetches the data of external and app users' reviews and loops through each review object to calculate the sum of all ratings and divide this number by the total number of ratings to get the average rating. The sum



and number of ratings are stored using the `useState` hook so that the program can re-calculate the average rating when a new review is submitted. When a user submits a review for a housing category, their rating will be added to the sum of ratings and divided by the number of the ratings added by 1 to get the new average rating.



Figure 8: Housing rating

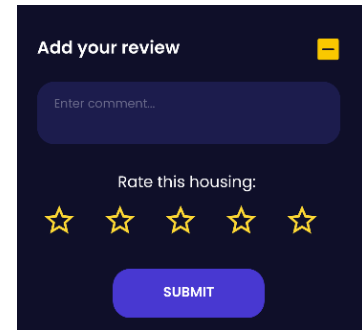


Figure 9: Add-review form

In the community screen, the users who share the most similarities in campus, subjects, and purpose of using the app will be recommended to the current user to befriend. For each similarity, the user that is not the current one will get one similarity point. The subjects attribute an array with three subjects, so 1 point for each similar subject. Then the users will be sorted from highest to the lowest point. 4 users with the highest points will be displayed on the most-like-you row of the community screen. The other rows use a filtered array to display the users matching the attribute and the selected option. There are three rows for the three mentioned attributes. Under the heading of each row, there are buttons for the options of that attribute. Pressing the button will change the cards on the row to match the selected option. Each row will show a maximum of 6 cards not to overload the page, and it will be easier to scroll from end to end. To see more results from each selected option, the user can press the “See more” option to see a whole result screen.

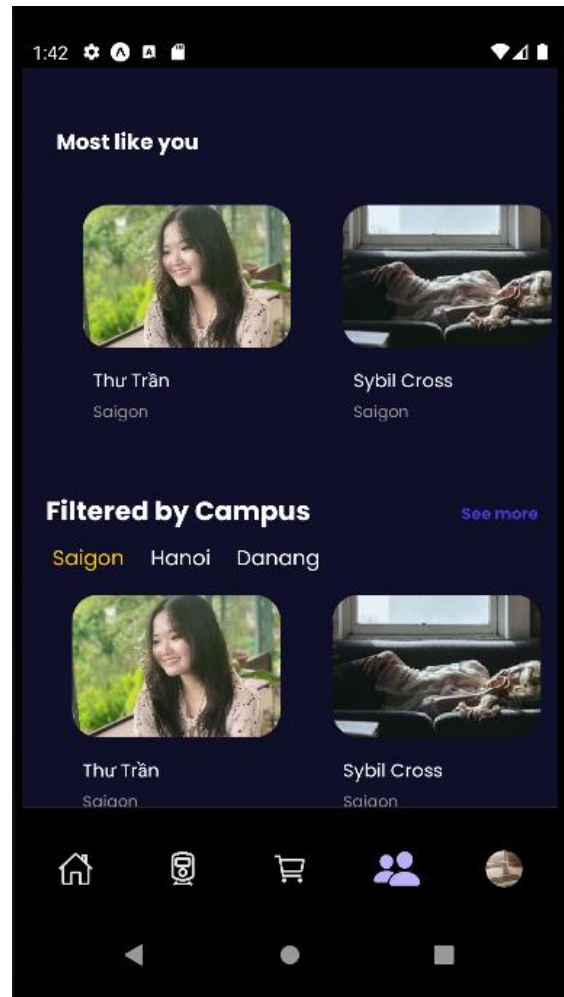


Figure 10: Community Screen

## 2.6. Requirements

This React Native app may target iOS 11.0 and Android 5.0 (API 21) or newer. The app support both the React Native CLI and the Expo CLI. The program is compatible with Windows, macOS, or Linux as the development operating system, though building and running iOS apps is limited to macOS. We install Expo [7] on Android and iPhone to test the development environment.

---

## 3. User Guide (Manual)

### 3.1. Installation

We used WebStorm as the IDE to code and test the program in the development environment, but an IDE is not needed to execute this program. However, the installation of node version 16.13.1 and expo-cli version 5.3.0 is mandatory for the project to run. No other software or libraries are required.

### 3.2. Set up procedure

- Step 1: type `git clone https://github.com/Melbourne-Backpack/melbourne-backpack.git` into terminal or download zip file from <https://github.com/Melbourne-Backpack/melbourne-backpack>, and unzip file to any folder.
- Step 2: Move to this folder using the command line, for example: `cd melbourne-backpack/`
- Step 3: type `npm install` into the terminal: this will install all npm packages with your package manager
- Step 4: Type `npm start` into the terminal from the project's root to start the program.
- Step 5: Download the Expo app on your phone
- Step 6: Scan the QR code in the terminal and open the app on Expo

### 3.3. User guide

**Splash:** An introduction splash screen containing app title, app description, images, and current version of our app. The splash screen appears for 3.6 seconds whenever the app is launching.

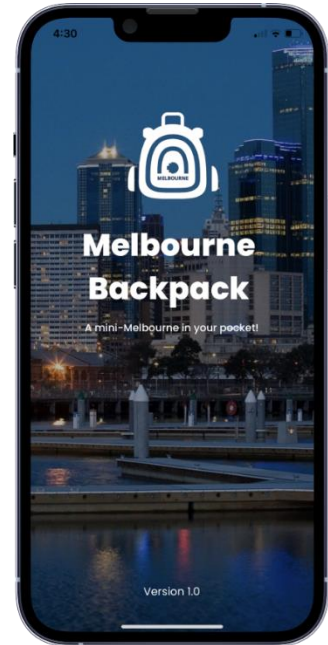


Figure 1: Splash


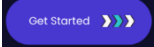


**Onboarding:** The onboarding screens aim to guide users to the most basic operations of our app. Users can press icon  to move on to next step and press  at the final step to move on to the Sign-in screen. Users also have an option to press icon  to go back previous step or  choose to skip to Sign-in screen immediately.



Figure 12: Onboarding - 2



Figure 13: Onboarding - 2



Figure 14: Onboarding – 3

**Authentication (Sign Up – Sign In – Forget Password):** Our app requires user to authenticate with an account on our system. For new user, they will need to Sign-up in pressing Sign-up button in the last text. The users will need to fill out email and password for registration (Figure 15). Our system will notify user with an alert if the validation of user email and password are not correct (Figure 17). If the information is valid, the system will appear a successful alert and navigate user to Personalization screen (Figure 18). On the other hand, for existing user, they can simply sign into our app using existed account (Figure 16). If they sign-in successfully, our system will pop-up successful alert (Figure 18) and auto navigate user to Home screen. In case user forgot password, they can press “Forget password” and fill out their email address (Figure 19). Our system will send a reset password link directly to the email that user put in (Figure 20).

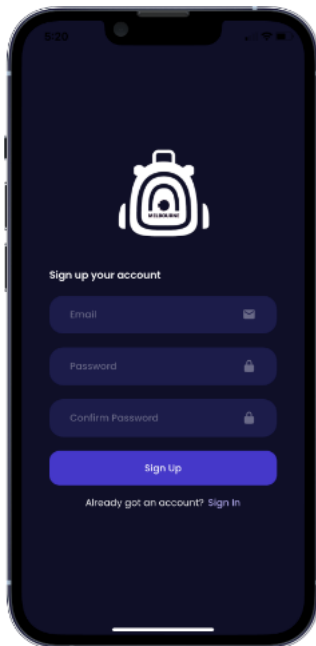


Figure 15: Sign Up

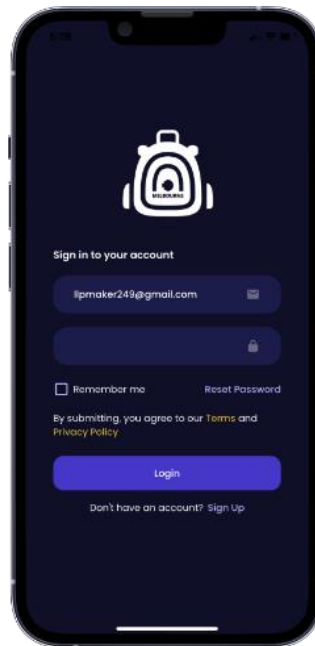


Figure 16: Sign In

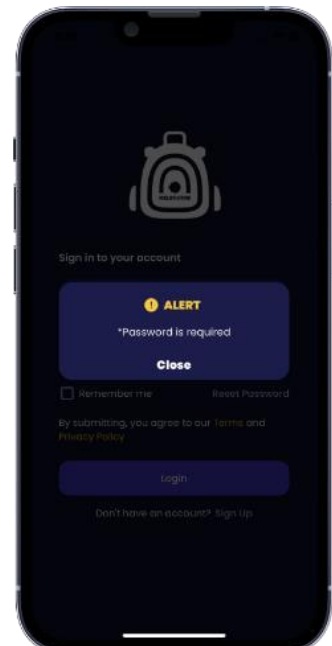


Figure 17: Error Alert

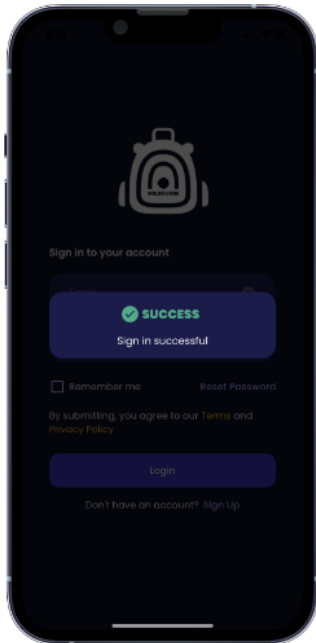


Figure 18: Success Alert

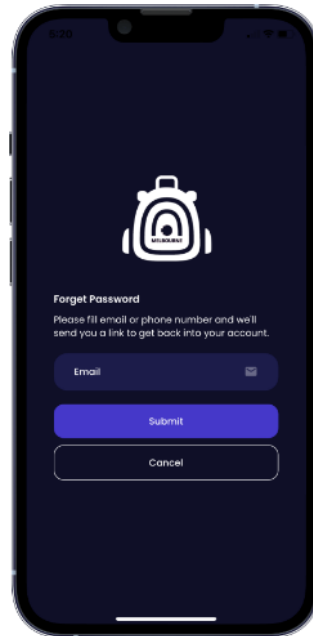


Figure 19: Forgot Password

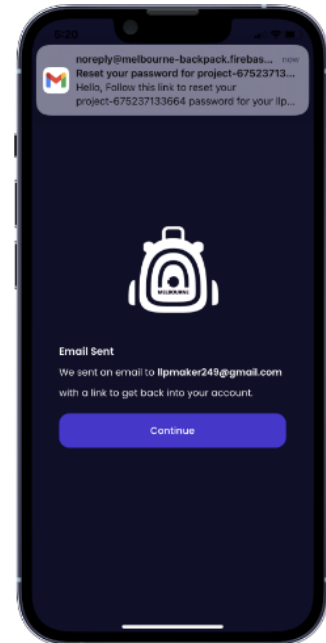


Figure 20: Email Sent

**Personalization:** For new user after signing in, they will be navigating to the personalization screen. We will collect more personal information of users in order to generate the best recommendations based on that information. User can press “Personalize your account” button to start fill out the upcoming data (Figure 21). The first screen requires users to choose their campus which they are currently studying, and they can move forward by pressing “Next button” (Figure 22). The next screen collects academic topics that users might interested in such as Engineering, Information Technology and the system requires the minimum of 3. Users can press “Next” button to go to next personalize screen or choose “Back” option to go back to Campus selection if they want to change their option (Figure 23). The last form requires users to fill out their personal information including avatar, full name, date of birth, introduction, and Facebook link as optional (Figure 24). Users need to check validate by pressing “Validate” button to ensure that all the information are filled as well as in the correct format. If the data is valid, the “Submit” option will appear and users can press it to submit the data to our database. Our app will navigate users to Finish screen and they can press Finish to go to our Home screen (Figure 25).

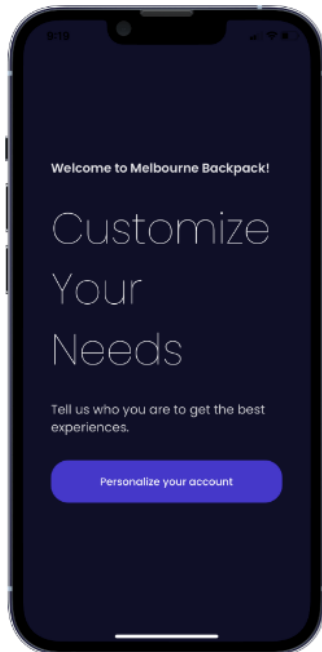


Figure 21: Personalization

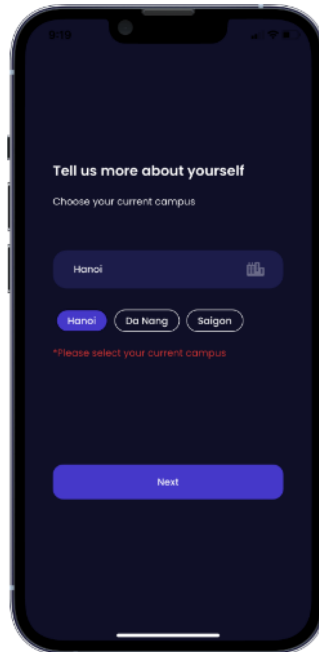


Figure 22: Campus

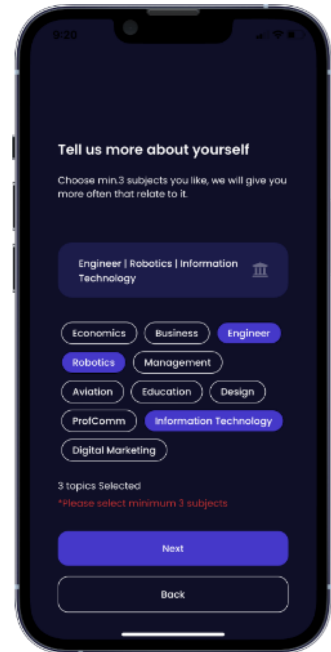


Figure 23: Subjects

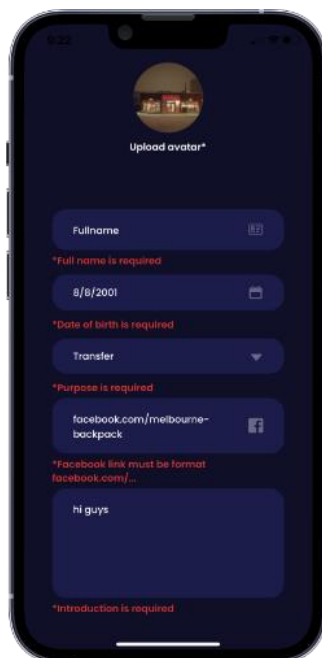


Figure 24: Form

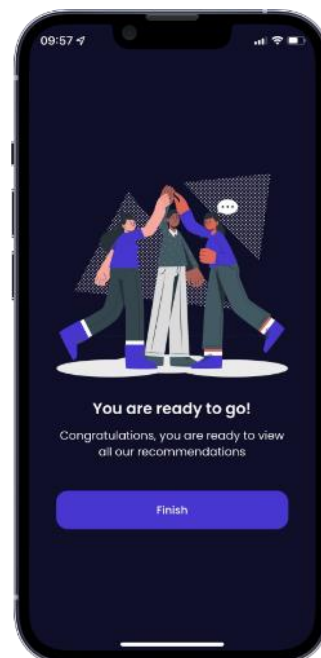
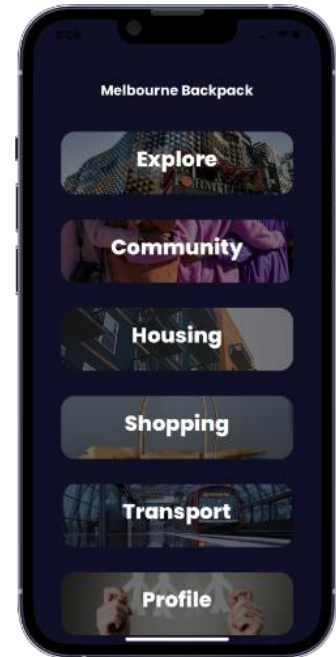


Figure 25: Ready to go

**Home:** After finish filling personal information, the system will navigate users to the Home screen. Home screen contains 6 features of our application which are Explore, Community, Housing, Shopping, Transport, and Profile (Figure 26). Users can press any button to navigate to the screen they want to access.



*Figure 26: Home*

**Explore:** If users press “Explore” button in Home screen, they will be navigating to this Explore screen (Figure 27). The Explore screen contains updated videos related to campus, exchange, international, global, and experience which we get from RMIT official YouTube channel. Each video card includes thumbnail image, title, and a small arrow button which is a switch to open that video card. We implement YouTube video player which support users to watch the video on our app (Figure 28). Users can also press RMIT University with a tick button to see more details of RMIT channel including the top title which will navigate to RMIT YouTube channel on press, detailed description, and multiple social media links (Figure 29).



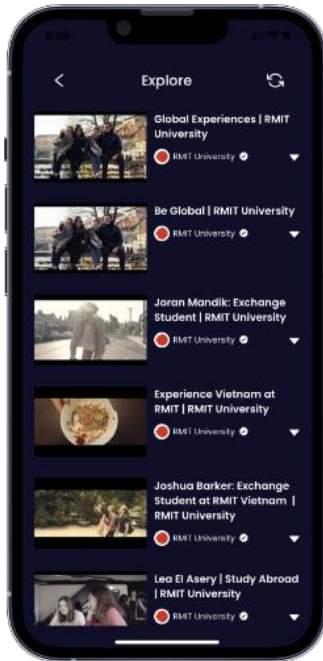


Figure 27: Explore

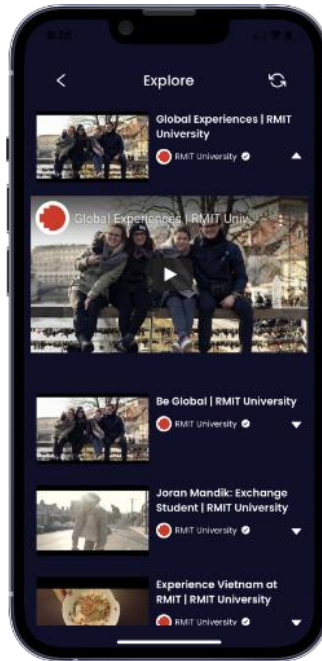


Figure 28: YouTube Video

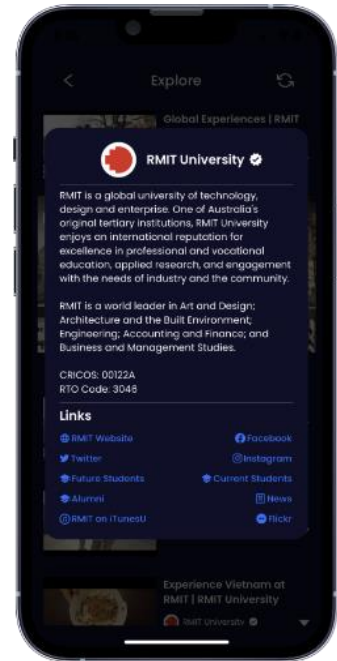


Figure 29: Channel Info

**Transport:** When users access Transport screen, the screen shows the most convenient transportation stop along with every nearest tram, train, and bus. Users can press the title of each card and the system will navigate them to Google Maps app on the phone with the location of the selected one. Users can also view the routes that go through that stop by pressing yellow text button “View routes that go through this stop” (Figure 30).

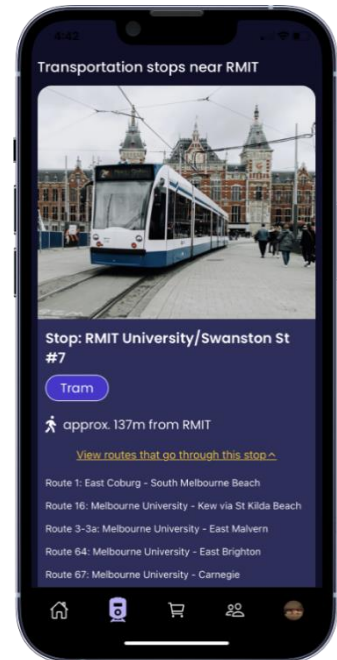



Figure 30: Transportation

**Shopping:** If users choose to see Shopping places, the screen contains some most popular and nearest marketplaces for users. Each card has an image and title as a website link which allows users to click on. Our app also provides the location of that shopping place where users can press to navigate to Google Maps. Users can also press yellow text-button “Read more” to see the short description of that place (Figure 31).



Figure 31: Shopping

**Housing & Housing Details:** One of the most important features of our app is the Housing screen. It has the same recommendation system as Transportation and Shopping screen with cards contain housing information. Each card displays a preview photo, type of accommodation, price, address, and rating star (Figure 32). On the Housing screen, we have a filter function  which allows user to search based on selected categories like name, price, number of beds & baths, and distance from RMIT campus (Figure 33). After selecting their desire option, they can press Apply and the system will generate the corresponding option. Users can press any housing card to view the detailed information. On the Housing Detail screen, each card appears with the same information on the previous Housing screen (Figure 34). However, the information contains detailed description and some additional features. Users can their personal review for the place as well as rate stars based on scale to 5 by pressing Add your review section (Figure 35). Furthermore, users can also review some online feedback that we crawl from online sources by pressing Online reviews section or they can view comments by previous RMIT students by choosing RMIT students' reviews option (Figure 36 & 37).

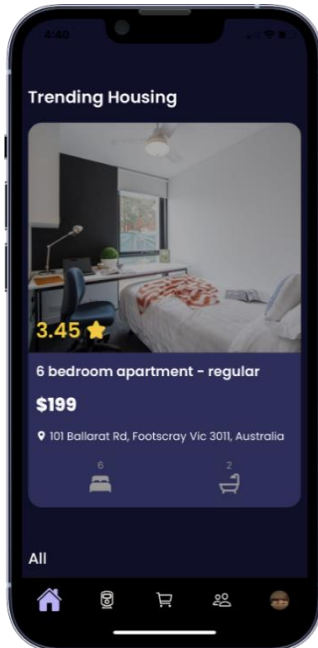


Figure 32: Housing

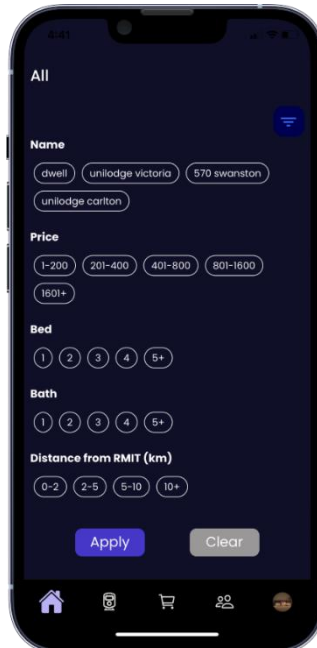


Figure 33: Housing Filter

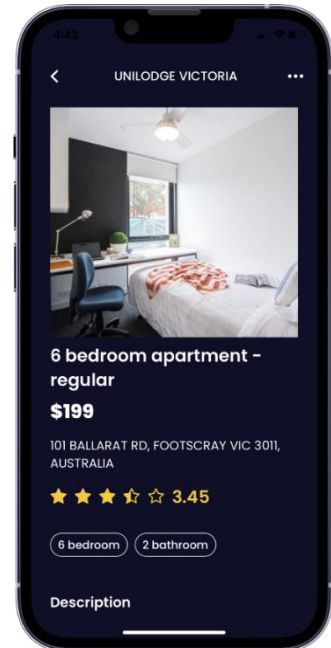


Figure 34: Housing Detail

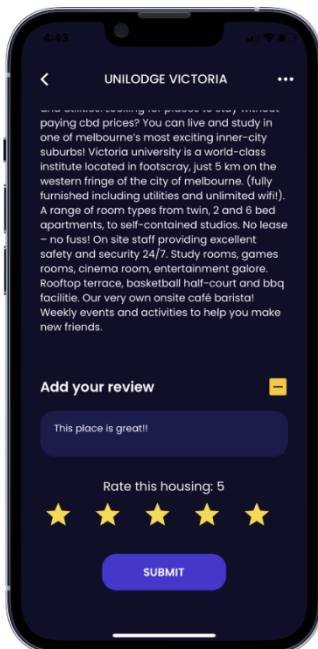


Figure 35: Add your review

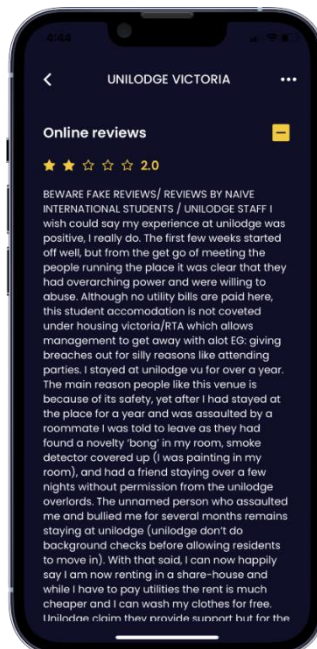


Figure 36: Online reviews

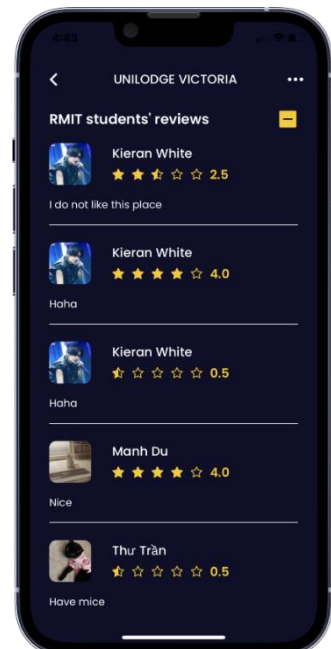




Figure 37: RMIT students' reviews

**Profile:** The Profile screen displays all the personal data that users give us in the previous steps for other users to view as well. There is an icon  button which can navigate

users back to Home screen and an  icon button to navigate to Message screen (Figure 38). Users can press a Copy icon next to user ID to copy their UID to clipboard for further use. At the bottom of the Profile screen, users can press the logout button and an alert will pop up (Figure 39). Users can sign out of their account by pressing Yes and will be navigating back to Sign in screen or they can choose Cancel to go back (Figure 40). Additionally, users can edit their personal data by pressing “Edit profile” button below the avatar and the system will navigate them to Edit Profile screen (Figure 38).

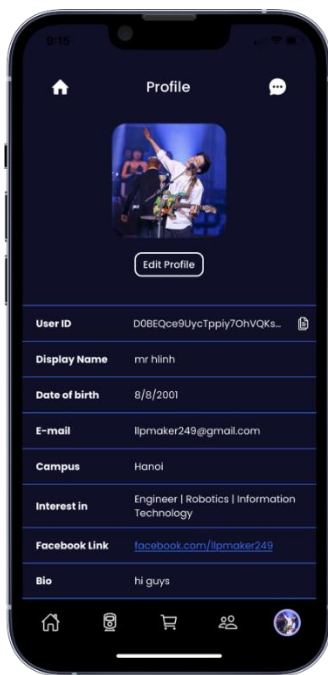


Figure 38: Profile

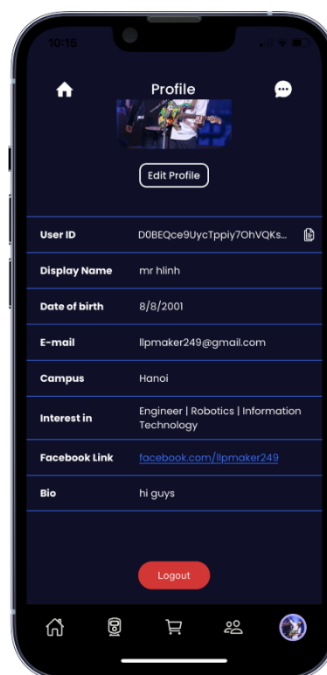


Figure 39: Profile Bottom

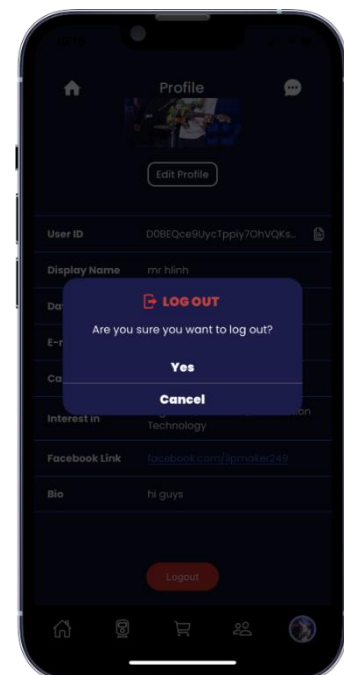



Figure 40: Sign out

**Edit Profile:** When users access the Edit Profile screen, they can press Back button to go back to Profile page without changing any information or press Done to go back with the new updated information. Our app allows users to change name, date of birth, campus, interested majors, Facebook link, and account (Figure 41). We have not implemented the updating avatar function yet, so the button Change Profile Photo is not available at the moment. Users can press this edit icon  to change that specific field. When the field is clicked, it will pop up a field input or multiple select options for

users to select (Figure 42 & 43). They can press Done button when they finish changing their data.

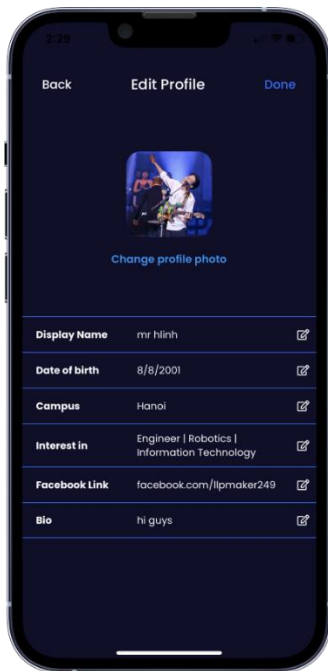


Figure 41: Edit Profile

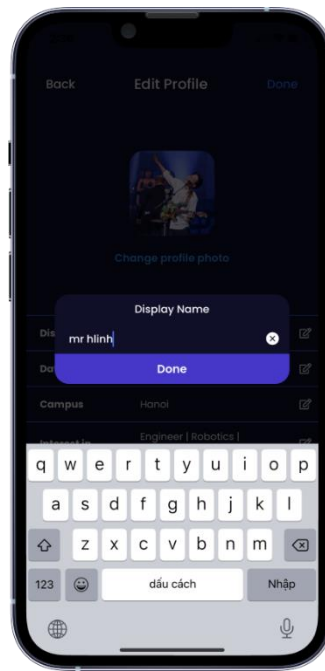


Figure 42: Edit Text

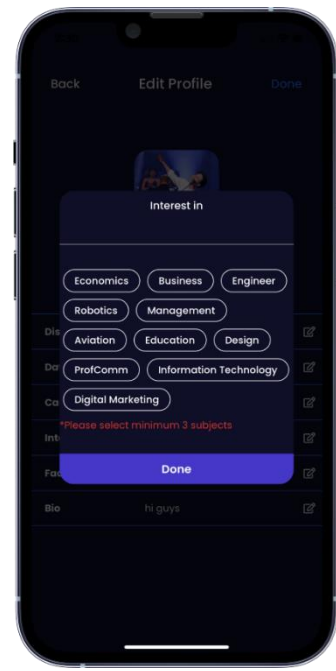

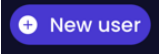


Figure 43: Edit Selections

**Message:** From the Profile screen, users can also go to Message screen. The Message screen contains every message between users and their friends. They can go back to Profile screen by pressing  icon at the top left corner. The main functions of the screen are a board displaying chat list including friends that users had messages and the  button for users to add new user to the chat list (Figure 44). When pressing the Add new user button, an input text required the user id will pop up (Figure 45). Users can fill in the user id of the friend they want to add if they already have the user ID. In case users do not have the user ID, they can go to Community screen and press the profile of that friend to get the user ID. The system will check if that user id is existed and valid to add to the chat list. User can press any person in the chat list to start messaging with them. The system will navigate to the correct Chat screen between user and that friend.



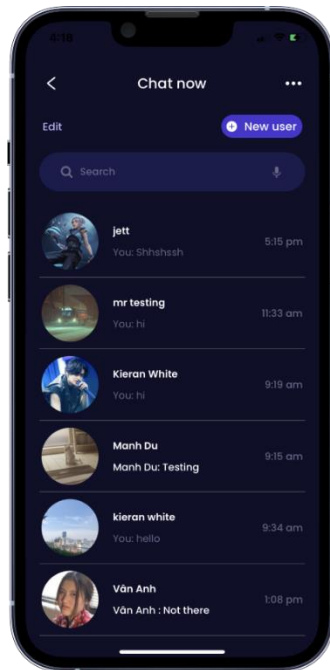


Figure 44: Message

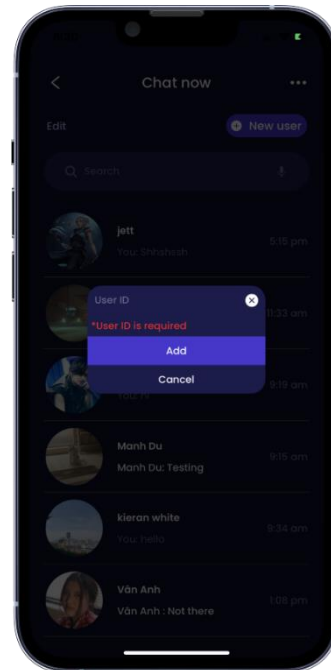





Figure 45: Add new user

**Chat:** In the Chat screen, users can go back to Message screen by pressing icon  at the top left corner. The Chat screen contains every existed message between 2 users. Each message box is customized to fit out theme and has text message and sent time stamp (Figure 46). At the bottom of the screen, there is a text bar which allows users to write a new message. When users click the text bar, it will be pushed up to make room for the keyboard (Figure 47). There will not be a sent button  until users write something. When users type a new message, the sent button  icon will appear (Figure 47). Users can send the message to their friend by pressing icon which the message will be push and update to our real-time database immediately (Figure 48). The new message that users

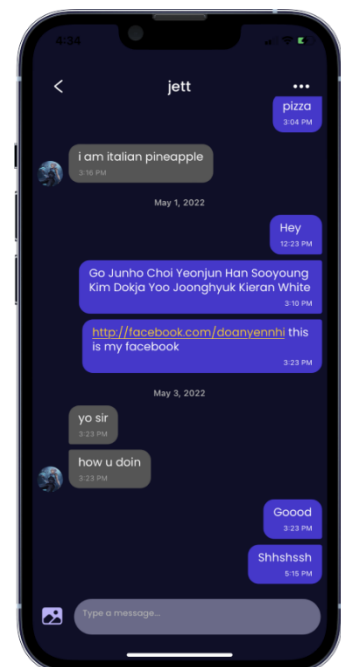


Figure 46: Chat

just sent will be displayed on the chat box and the friend will also notice the new message instantly (Figure 44).

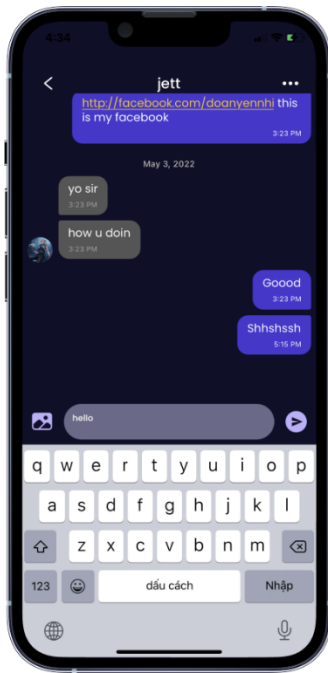


Figure 47: Chat bar



Figure 48: Sent a message

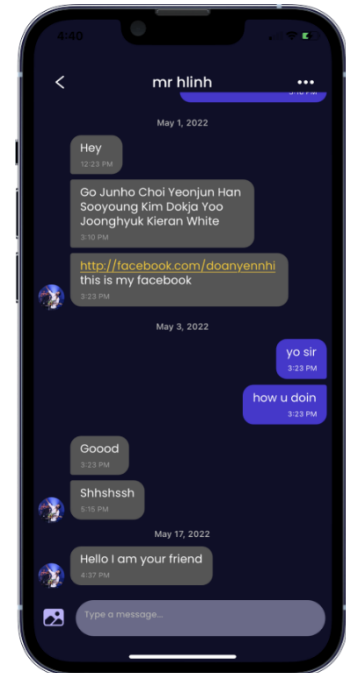


Figure 49: Friend Chat

**Community:** The Community Screen contains lists of user cards that we divide based on different categories. Our app has the recommendation list of user cards that have the most interested as you. Most like you list user cards displays at the top of the Community screen in horizontal view (Figure 50). Below are lists of user cards we filtered by Campus, Subjects, and Purpose. Users can select the option to search for the related users (Figures 50 & 51). For example, users can select Saigon, Hanoi, or Danang by pressing the text button to filtered users in that specific campus. The same applies to Filtered by Subjects and Filtered by Purpose where users can pick the option that they want. Our system displays 4 most related users on the screen. If users want to see more, they can press the blue See more button on the right side of each list to see all available users based on that category. Our system will navigate them to a new screen that includes all users (Figure 52). Furthermore, users can always press that user card to see the full

Profile of that user. Our system will show the same information of that user as in Profile screen. However, the Edit Profile button will be replaced with the Add to chat list button (Figure 53). When users click Add to chat list button, our system will automatically navigate to Message screen with pop-up user information to add them into friend list. Users can press Add to add that friend or Cancel to go back (Figure 54).

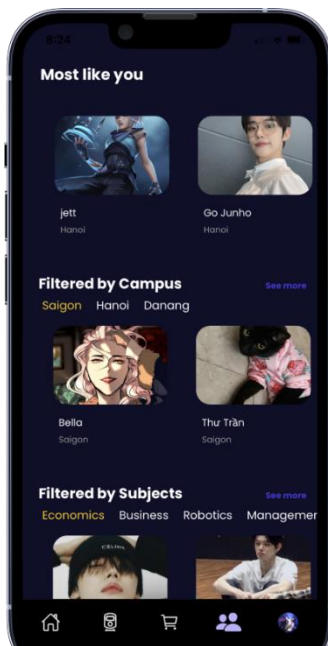


Figure 50: Community Top

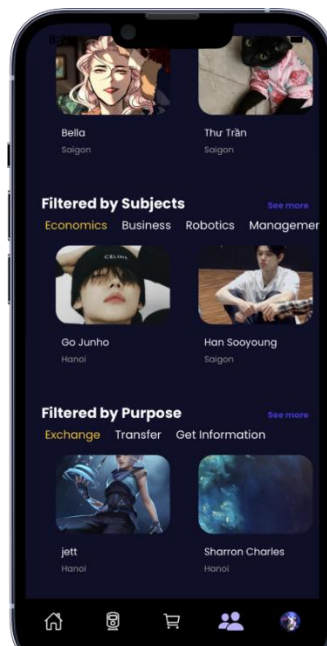


Figure 51: Community Bottom

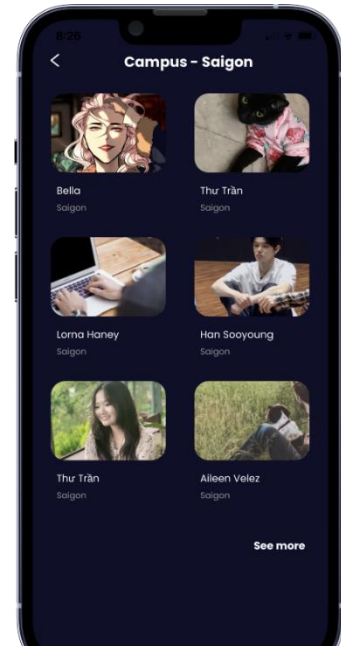


Figure 52: See more profiles



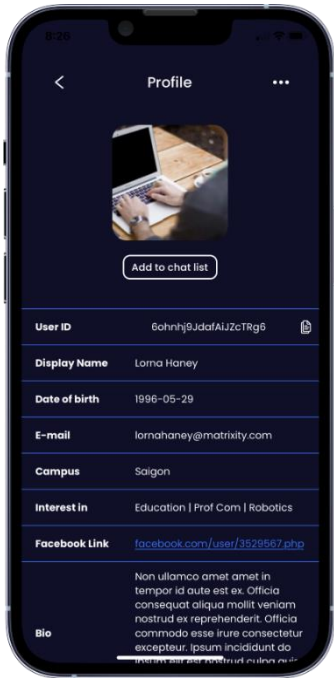


Figure 53: Profile of other users

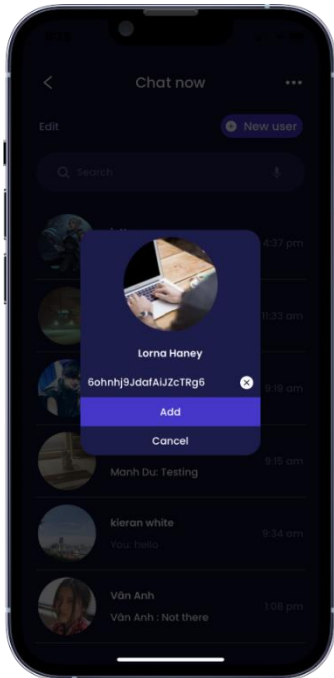


Figure 54: Add to chat list

## 4. Academic Poster





### MELBOURNE BACKPACK

**Authors**  
 Tran Ngoc Anh Thu  
 Doan Van My  
 Du Duc Minh  
 Nguyen Hoang Linh

**Affiliations**  
 School of Science, Engineering, & Technology, RMIT University Vietnam, 2022

**Supervisor:** Anna Felipe

Behance



Video demo



GitHub





**IMAGE**  
Melbourne backpack run on an iPhone and an Android

Sai Gon South  
 702 Nguyen Van Linh,  
 Tan Phong Ward, District 7, HCM  
 rmit.edu.vn

## Results



### INTRODUCTION

The information is spread out all over, including RMIT Mobility, Facebook pages, and RMIT's official website. A free cross-platform application packs everything about exchanging and transferring to RMIT Melbourne into one package: housing information, shopping locations, transport information, video exploration, and a supportive community that students can access and chat.

### Objective

Simplify the RMIT Melbourne research process via the simple and modern mobile app interface.

### Methodology

Agile, Kanban and Scrum development process




### Tools

- React Native: framework to develop the application
- Firebase: database, storage, authentication, realtime data
- Jupyter: interactive development environment for data pipeline
- Expo: making universal native apps for Android and iOS
- Webstorm: an IDE for code in the development environment







### Conclusion

- Completed the main five features: housing, community, shopping, videos, and transportation
- Responsive design to support all Mobile Phone screen sizes
- Deployed and tested the app on the Expo Go application on both Android and iPhone environments.

### Future Work

- Enhance the app's security
- Diverse accommodation types
- Integrate both collaborative and content filtering for the better recommender algorithm
- Add blogs section for successfully studying in Melbourne and getting scholarship content.

### Analysis

To determine the app's features, we examine business needs and user stories. The software architecture, on the other hand, is the technological stack that links these components together.



### Software Architecture

How components interact with each other



### Activity Diagram

The flow of the program on a high level

Figure 55: Academic poster

## 5. A4 Trifold Brochure

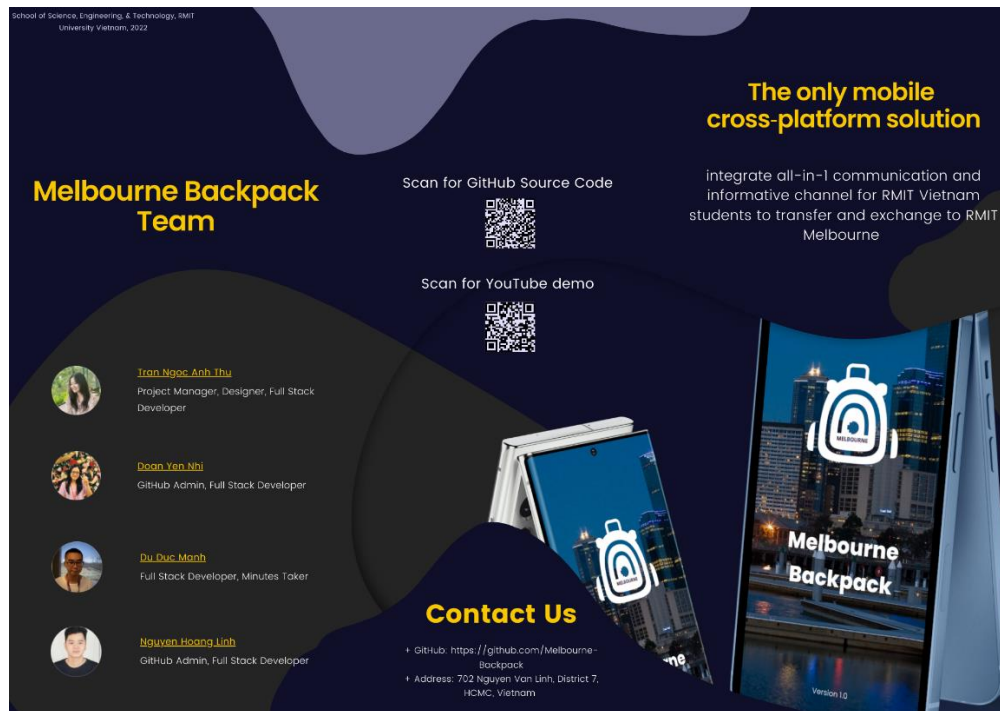


Figure 56: Front-side of brochure



Figure 57: Back-side of brochure

## References

- [1] Google Cloud. "Firestore Service Level Agreement (SLA)."  
[https://cloud.google.com/firestore/sla?fbclid=IwAR0pr0tbqgLONECLYm\\_kOYFZmKR4mqwGJfvz8cCeV5\\_1XuiLZRGxqpu2No](https://cloud.google.com/firestore/sla?fbclid=IwAR0pr0tbqgLONECLYm_kOYFZmKR4mqwGJfvz8cCeV5_1XuiLZRGxqpu2No) (accessed May 21, 2022).
- [2] C. Fanchi. "What Is Mobile Backend As A Service (MBaaS)?" Backendless.  
<https://backendless.com/what-is-mobile-backend-as-a-service-mbaas/> (accessed May 21, 2022).
- [3] Webstorm. "React Native." <https://www.jetbrains.com/help/webstorm/react-native.html#:~:text=WebStorm%20helps%20you%20create%2C%20edit,for%20React%20and%20Flow%20symbols> (accessed May 21, 2022).
- [4] React Native. "React Native · Learn once, write anywhere."  
<https://reactnative.dev/> (accessed May 21, 2022).
- [5] Firebase. "Firebase." <https://firebase.google.com/> (accessed May 21, 2022).
- [6] Jupyter. "Jupyter."  
<https://jupyter.org/#:~:text=The%20Jupyter%20Notebook%20is%20the,streamlined%2C%20document%2Dcentric%20experience> (accessed May 21, 2022).
- [7] Expo. "Make any app. Run it everywhere." <https://expo.dev/> (accessed May 21, 2022).

---

## Appendix: Resources

- Trello Project Management System: <https://trello.com/b/c85KQUtQ/team-18-melbourne-backpack-sepm>
- GitHub Code Repo: <https://github.com/Melbourne-Backpack/melbourne-backpack>
- YouTube video demo presentation: <https://youtu.be/XtpU9N3Mslo>
- Google Drive general folder for the whole project:  
[https://drive.google.com/drive/folders/1C5n\\_HWoQz421SLwNcJcfOa3aBZWP3gEj?usp=sharing](https://drive.google.com/drive/folders/1C5n_HWoQz421SLwNcJcfOa3aBZWP3gEj?usp=sharing)