UNIVERSITY OF MALAYA

Project Report

WIX2007 MOBILE APPLICATION DEVELOPMENT

SEMESTER 1 SESSION 2024/2025

LinkCamp

Group Member:

1. LEE XIANG WEN 23064284
2. ONG HAN LIN 23004928
3. CHAI JIE SHENG 22100945
4. CYNTHIA ONG KA 23004897
5. ENG KE JIA 22089176

TABLE OF CONTENT

1.0 Introduction

The LinkCamp project fits under the United Nations SDG 4: Quality Education as well as SGD 8: Decent Work and Economic Growth. Education of both girl child as well as disabled and in this scramble to make schooling for all a reality is supported by this goal of fostering fair quality education that is also inclusive of the disabled and that encourages lifelong learning.

LinkCamp is an innovative concept that blends learning and employment resources to offer all stakeholders full solutions at this intersection. It represents the place for gaining knowledge and skills as well as providing the access to career opportunities as the cooperation of academic and professional spheres.

Target Audience

The application primarily target two groups:

1) Teenagers: In order to help them develop the resources and tools they need to change and continue their education.
2) Workers: To promote the acquisition of new knowledge and skills by employees.

Objective

The main aim of this project is to create a dominant platform that acts as a source of quality education and job vacancies. This will involve provision of a broad variety of courses, certifications and degrees which are recognized by various universal bodies and institutions to enhance the participant's career needs and styles. Content delivery routes will be customized with an instructional idea that organizes content according to users' skill level and desired product outcomes.

Besides education, one will be able to search for a job with the help of the platform, as well as search for internships, and receive information about a certain career. Learning platforms will suggest courses for skills that learners lack and career counseling and mentorship will provide support for any career issues. This will also mean that more users will be able to converse, and share experiential knowledge with each other through the platform affordances as well as being able to network with their peers.

To this end, the users will have the capacity to check on the amount of progress through certifications and badges. People will receive analytical data, which will help them reach better and well-grounded decisions of their further actions. In the end, the platform will allow creating an ecosystem with the necessary resources for the development of the person as a learner, in the educational process and as a professional who wants to achieve success in his/her career.

2.0 Planning

2.1 Role and Responsibilities

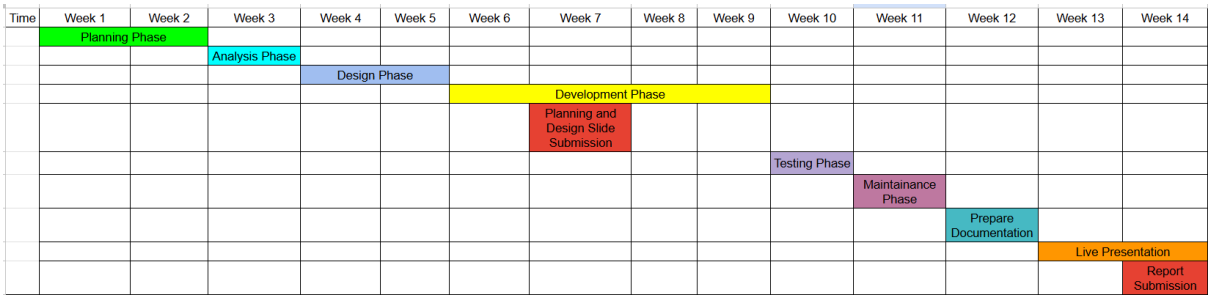
Individual assignments were also made to the players in the team with a view of decentralizing authority and responsibility to achieve project goals. The roles include project coordination, program development, space design, and data validation.

Role	Pic	Responsibilities
Project Manager	Lee Xiang Wen	- Plan and oversee the project workflow - Ensure tasks are completed according to schedule - Coordinate communication between team members and stakeholders - Assist in resolving technical and project issues - Track project progress and report to supervisors
Backend Developer	Ong Han Lin	- Develop and maintain the backend system - Integrate APIs and handle server-side logic - Optimize application performance and scalability - Collaborate with frontend and database developers for integration - Perform debugging and testing of backend code
Database Developer	Chai Jie Sheng	- Design and develop the database structure for the application - Analyze requirements to create Entity-Relationship Diagrams (ERD) - Optimize database queries and ensure data integrity - Implement database security, backups, and recovery mechanisms - Troubleshoot and resolve database-related issues
GUI Designer	Cynthia Ong Ka	- Design the user interface (UI) layout and graphical elements - Ensure the UI aligns with user experience (UX) best practices - Develop wireframes, mockups, and prototypes - Collaborate with developers to maintain consistency in design implementation - Conduct usability tests and iterate designs based on feedback

GUI Developer	Eng Ke Jia	- Implement the graphical user interface based on design specifications - Develop responsive and interactive UI components - Integrate frontend elements with backend functionality - Debug and optimize frontend code for better performance - Ensure cross-platform and cross-browser compatibility
Tester	All Members	- Develop and execute test cases to validate application functionality - Identify, report, and track bugs in the application - Perform functional, performance, and security testing - Collaborate with developers to ensure bug fixes and improvements - Document testing results and provide quality assurance feedback

2.2 Gantt Chart

A Gantt chart was created to visualize the timeline of the project. This detailed timeline tracks each phase of the project, including requirements gathering, design, development, and testing phases, ensuring milestones are achieved on time.



2.3 Database Design and Data Structure

- Database: The project uses Firebase and SQLLite for database management, which provides scalability and real-time data synchronization.
- Data Structure: The Hash Map data structure is utilized for efficient storage and retrieval of user data, supporting the application's dynamic functionalities.

3.0 Functional Requirements (Modules)

The application is composed of the following core modules, each with specific functional requirements:

3.1 User Authentication Module

This module ensures secure access to the platform.

- Users can register using an email address and password
- A two-factor authentication mechanism enhance security
- Login categories:
    1) Lecture: Facilitates classroom management and communications
    2) Company: Allows companies to browse and interact with potential candidates
    3) End-user: General users access learning and career resources.

3.2 User Profile Module

The profile module allows users to manage their personal information.

- Users can view and update their profiles
- The module supports uploading profile pictures and education certificates
- Company users can access and review end-user profiles.

3.3 Messaging Module

This module enables real-time communication.

- Users can send and receive messages instantly
- Lecturers can create broadcast channels and facilitate classroom announcements.
- Each message contains details, ensuring transparency in communication.

3.4 Notification Module

Notifications keep users informed and engaged.

- Push notifications alert users about new messages and events.
- Notifications are sent for interview requests and updates about class participation.
- Lecturers are notified when students join their classes.

3.5 Payment Module

This module provides financial functionalities within the application.

- Users can make donations to other users.
- Users can view donations made and donations received in a list.
- Users can withdraw the donation received.

4.0 Non-functional Requirements

4.1 Performance

- The application ensures a fast and responsive experience, with the main screen loading in under 2 seconds under normal network conditions.
- It is designed to handle a large number of concurrent users without compromising functionality.

4.2 Security

- User data is securely stored within the Firebase database system.
- Two-factor authentication provides an additional layer of security, ensuring that only authorized users gain access.

5.0 UI Design

The user interface design focuses on simplicity and usability. Key pages include:

5.1 User Authentication Module

The login and registration pages provide a seamless entry point for users, with clear instructions and a straightforward layout.

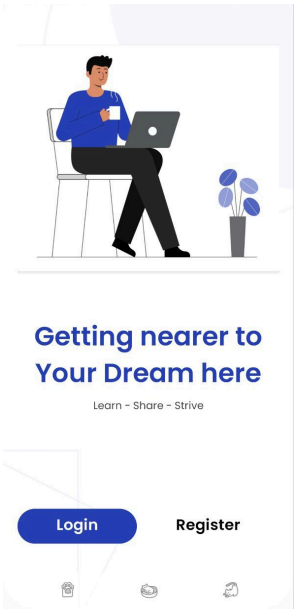


Figure 5.1.1

Welcome Activity
This page acts as the welcome screen when users open our app. It provides options to navigate to either the login or register page. First-time users can opt to register and create an account at the register page (Figure 5.1.) to save their progress in the app. For users who already have an account, they can simply click the login button to go to the login page (Figure 5.1.2) and log into their account.

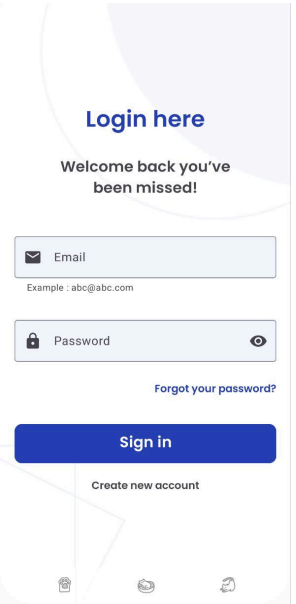


Figure 5.1.2

Login Activity
This page serves as the login page where users can login to their account through this page. After users have entered their credentials, the login page will authenticate the information. If the entered email address and password match, the page will proceed to the OTP verification page (Figure 5.1.2). Otherwise, an error message will be displayed, indicating unsuccessful login. The 'Forgot your password?' link is available for users who need to reset their password. Additionally, there is a link 'Create new account' option guiding users to the register page for new account creation.

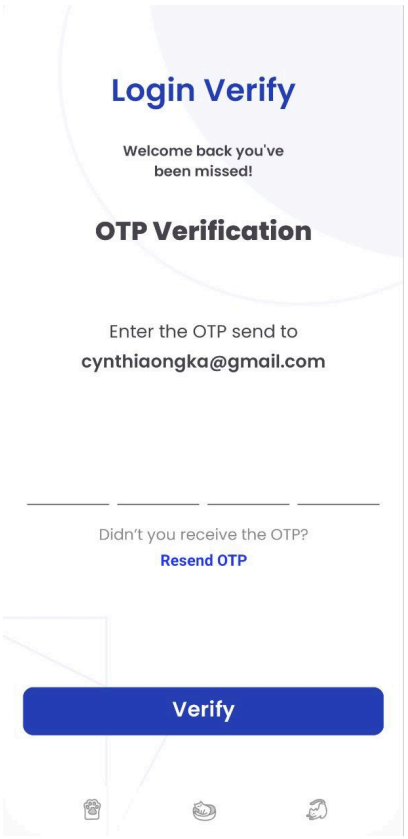


Figure 5.1.3

Verification Activity
After the user enters their email and password on the login page, they will be directed to the OTP verification page (Figure 5.1.3). On this page, an OTP code will be sent to the user's email. The user must enter the OTP code in the provided field. If the entered OTP matches the code sent, the user can log in successfully by clicking the "Verify" button. If the OTP does not match, an error message will be displayed, and the user will have two more attempts to enter the correct code. If the user does not receive the OTP, they can click the "Resend OTP" link to request the system to send the code again.

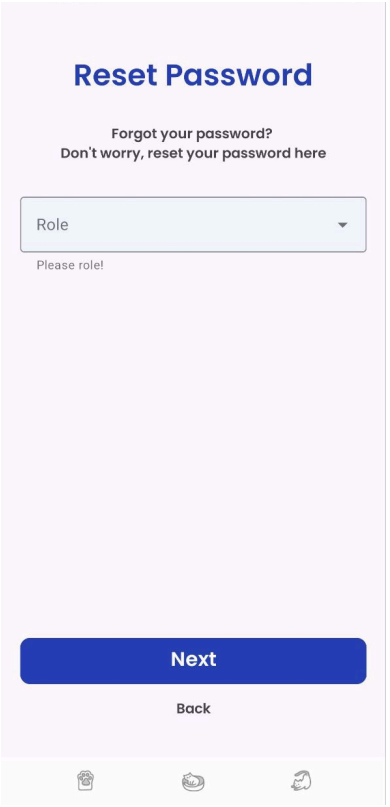


Figure 5.1.4.0

Reset Password Activity
Users experiencing difficulty with their password can initiate the password reset process on the 'Reset Password' page (Figure 5.1.4.0).

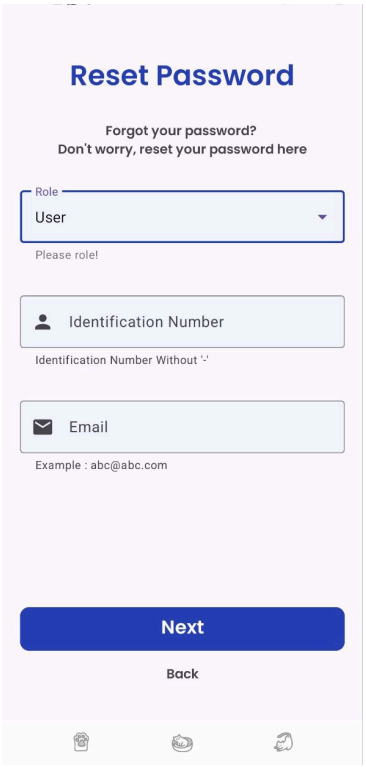


Figure 5.1.4.1

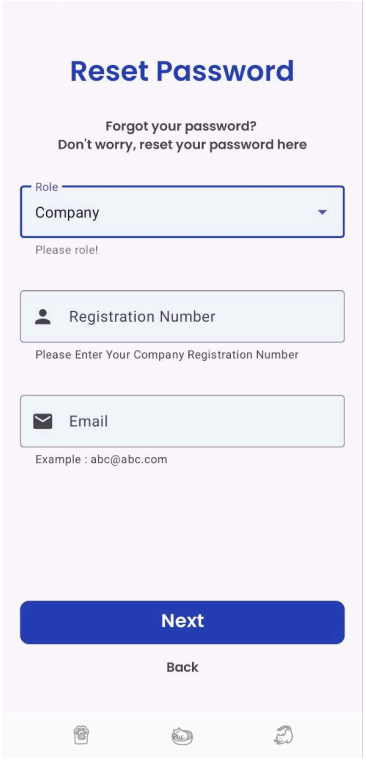


Figure 5.1.4.2

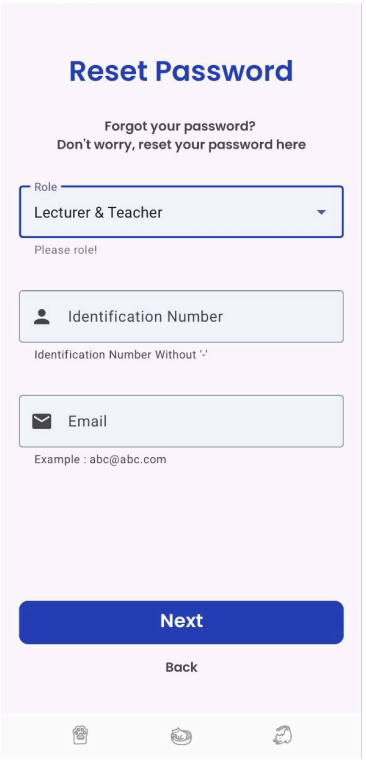


Figure 5.1.4.3

Reset Password Activity
The required information varies based on the selected role.

- For users, an identification number and email are required (Figure 5.1.4.1).
- For companies, a registration number and email are required (Figure 5.1.4.2).
- For lecturers and teachers, an identification number and email are required (Figure 5.1.4.3).

If the entered information is correct, the user will proceed to the OTP verification page. Otherwise, an error message will be displayed.

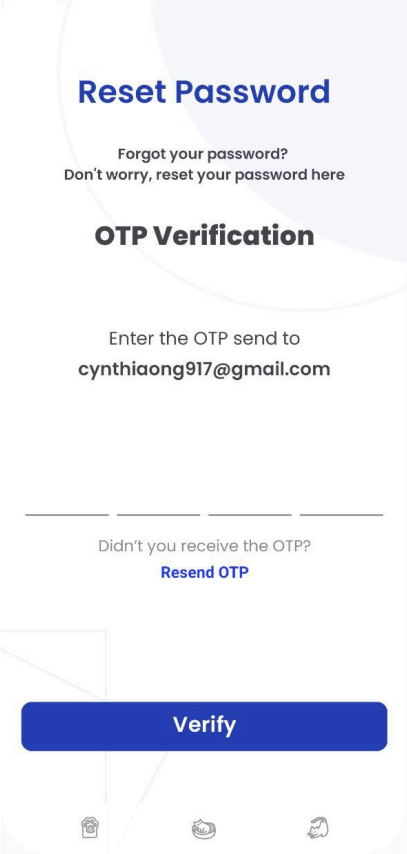


Figure 5.1.4.4

Reset Password Activity (Verification)
After the user enters their information on the reset password page, they will be directed to the OTP verification page (Figure 5.1.4.4). On this page, an OTP code will be sent to the user's email. The user must enter the OTP code in the provided field. If the entered OTP matches the code sent, the user can proceed with reset password by clicking the "Verify" button. If the OTP does not match, an error message will be displayed, and the user will have two more attempts to enter the correct code. If the user does not receive the OTP, they can click the "Resend OTP" link to request the system to send the code again.

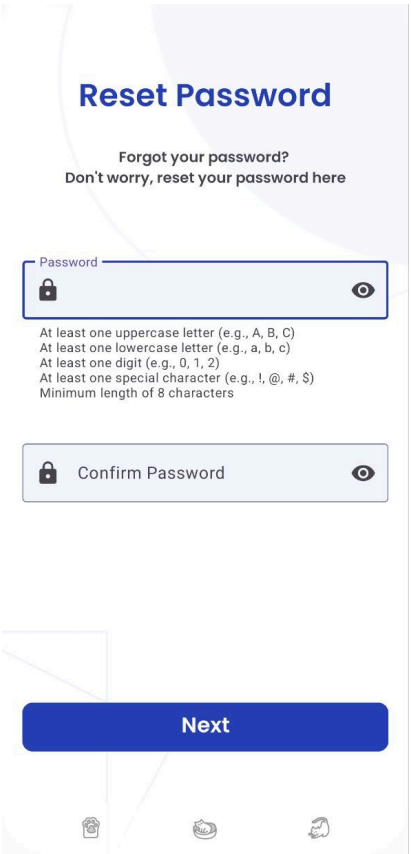


Figure 5.1.4.5

Reset Password Activity
Users can reset their password by entering a new password that meets the specified requirements (Figure 5.1.4.5). If the password does not comply, an error message will be displayed. Users are also required to re-enter the password for confirmation. If both entries match, the password will be successfully changed, and a success message will be displayed. Otherwise, an error message will be shown.

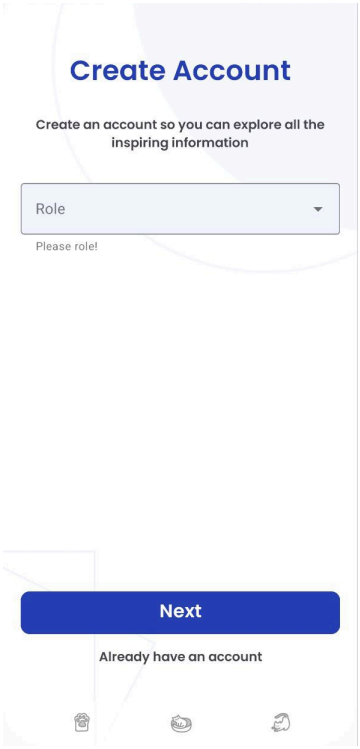


Figure 5.1.6

Register Activity
New users must complete the registration process before they can use the app. The registration process is different for users, companies and lecturers. The "Already have an account" link redirects existing users to the login page.

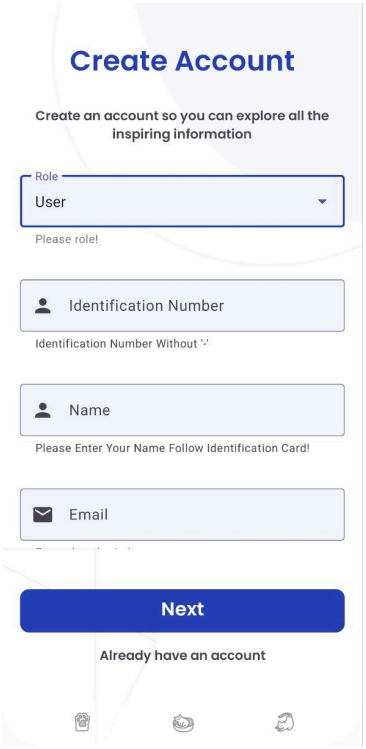


Figure 5.1.7.0

User Registration Activity
Users are required to provide their details, including their identification number, name, and email. The system will verify that the identification number and email are unique and not already registered. The "Already have an account" link redirects existing users to the login page.

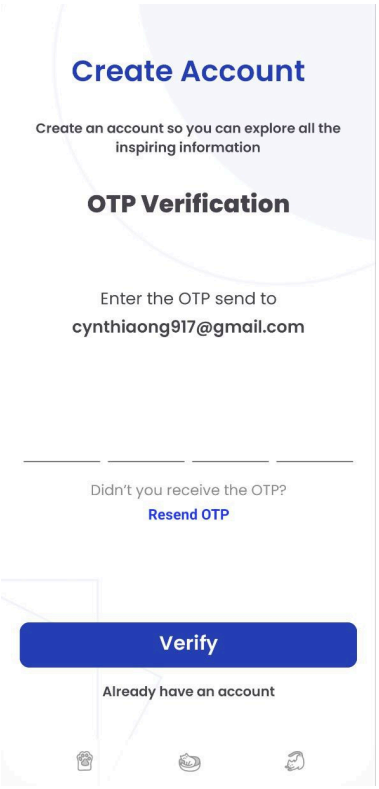


Figure 5.1.7.1

User Registration Activity (Verification)
After the user enters their information on the registration page, they will be directed to the OTP verification page (Figure 5.1.7.1). On this page, an OTP code will be sent to the user's email. The user must enter the OTP code in the provided field. If the entered OTP matches the code sent, the user can proceed with registration by clicking the "Verify" button. If the OTP does not match, an error message will be displayed, and the user will have two more attempts to enter the correct code. If the user does not receive the OTP, they can click the "Resend OTP" link to request the system to send the code again. The "Already have an account" link redirects existing users to the login page.

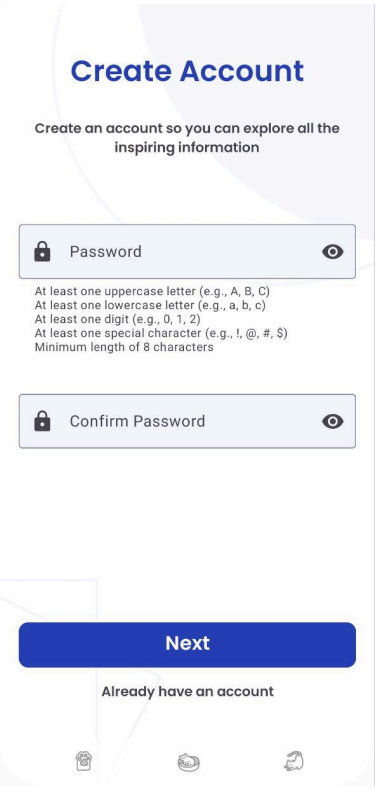


Figure 5.1.7.2

User Registration Activity
Users can set their password by entering a password that meets the specified requirements (Figure 5.1.7.2). If the password does not comply, an error message will be displayed. Users are also required to re-enter the password for confirmation. If both entries match, the password will be successfully set, and a success message will be displayed. Otherwise, an error message will be shown. The "Already have an account" link redirects existing users to the login page.

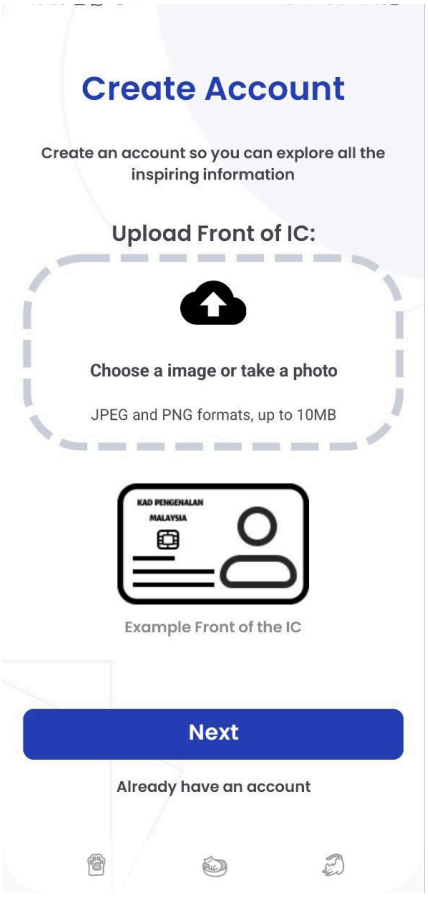


Figure 5.1.7.3

User Registration Activity
Users are required to upload the front side of their identification card (IC) to verify the authenticity of their account. They can choose to upload an image from their device or take a photo directly. The "Already have an account" link redirects existing users to the login page.

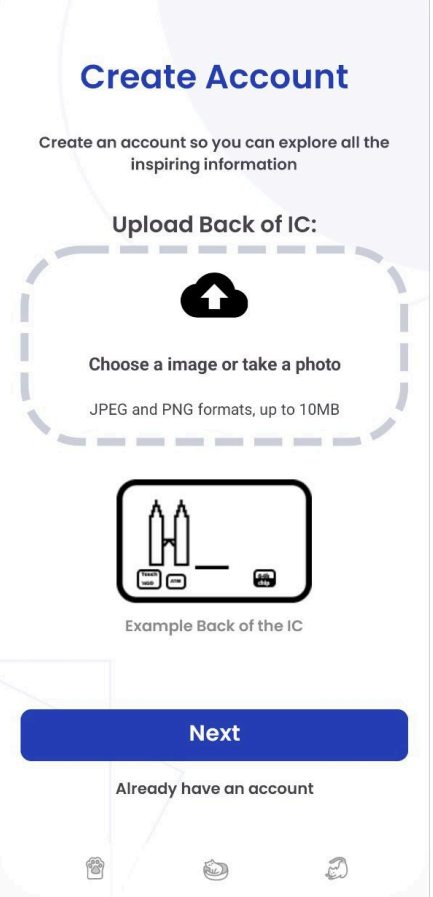


Figure 5.1.7.4

User Registration Activity
Users are required to upload the back side of their identification card (IC) to verify the authenticity of their account. They can choose to upload an image from their device or take a photo directly. The "Already have an account" link redirects existing users to the login page.

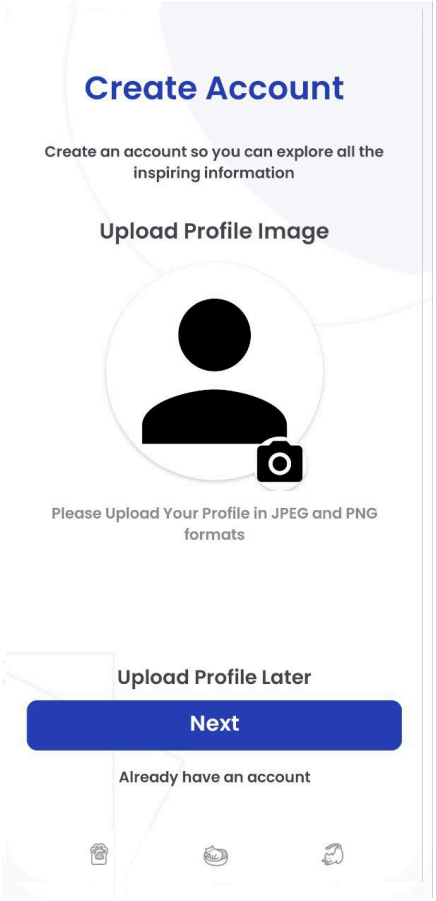


Figure 5.1.7.5

User Registration Activity
Users are required to upload profile images. They can choose to upload an image from their device or take a photo directly. The "Already have an account" link redirects existing users to the login page.

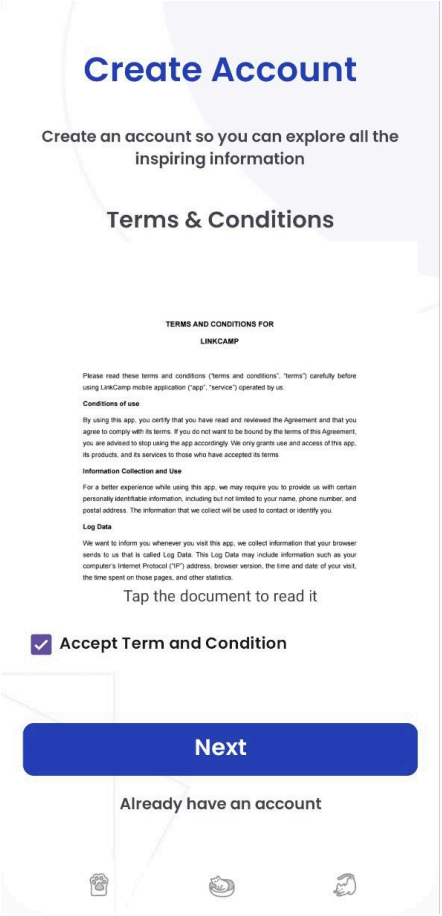


Figure 5.1.7.6

User Registration Activity
Users must read and accept the terms and conditions in order to complete the registration process. The "Already have an account" link redirects existing users to the login page.

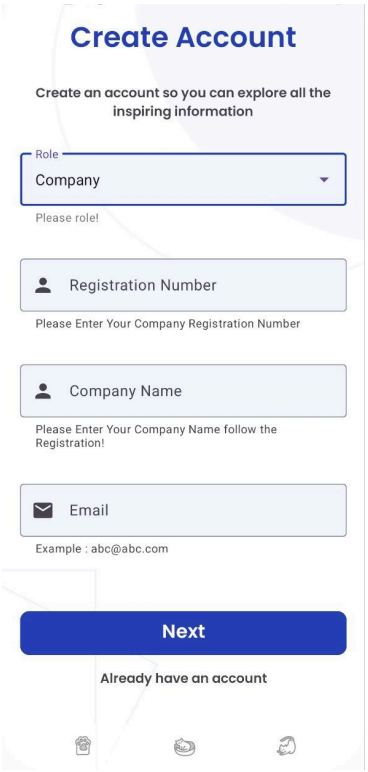


Figure 5.1.8.0

Company Registration Activity
Companies are required to provide their details, including their registration number, company name, and email. The system will verify that the registration number and email are unique and not already registered. The "Already have an account" link redirects existing companies to the login page.

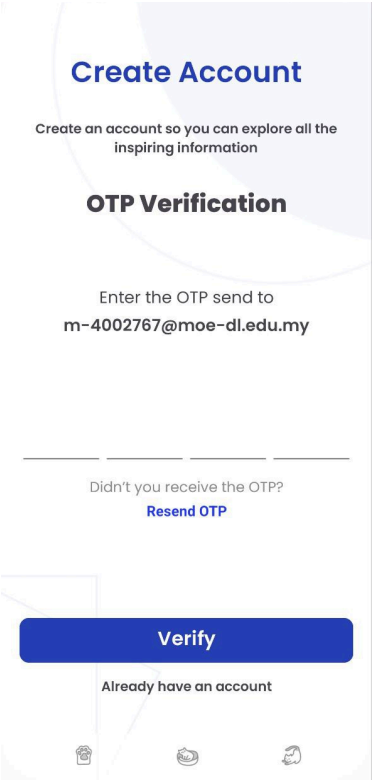


Figure 5.1.8.1

Company Registration Activity (Verification)
After the companies enter their information on the registration page, they will be directed to the OTP verification page (Figure 5.1.8.1). On this page, an OTP code will be sent to the company's email. The company must enter the OTP code in the provided field. If the entered OTP matches the code sent, the company can proceed with registration by clicking the "Verify" button. If the OTP does not match, an error message will be displayed, and the company will have two more attempts to enter the correct code. If the company does not receive the OTP, they can click the "Resend OTP" link to request the system to send the code again. The "Already have an account" link redirects existing companies to the login page.

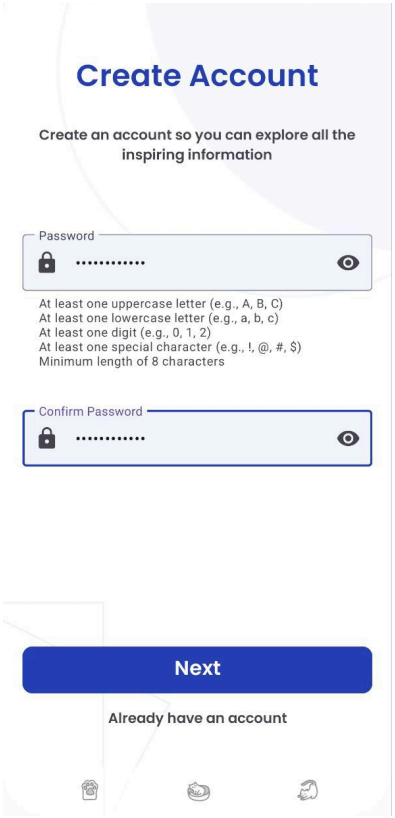


Figure 5.1.8.2

Company Registration Activity

Companies can set their password by entering a password that meets the specified requirements (Figure 5.1.8.2). If the password does not comply, an error message will be displayed. Companies are also required to re-enter the password for confirmation. If both entries match, the password will be successfully set, and a success message will be displayed. Otherwise, an error message will be shown. The "Already have an account" link redirects existing companies to the login page.

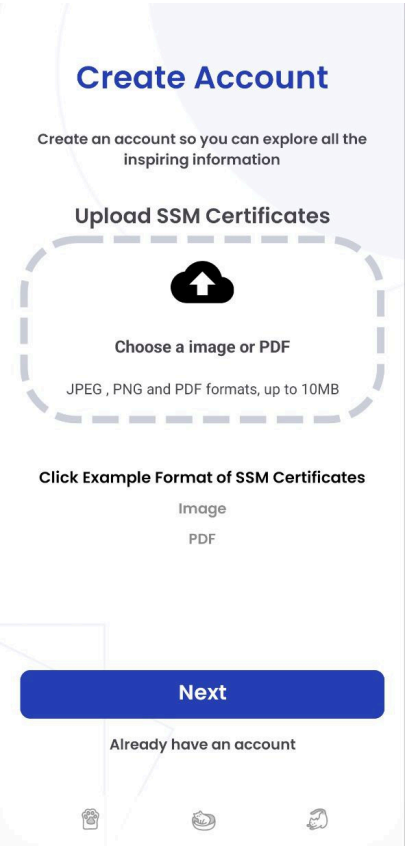


Figure 5.1.8.3

Company Registration Activity

Companies are required to upload their SSM certificates to verify the authenticity of their account. They can choose to upload an image or pdf from their device. The "Already have an account" link redirects existing companies to the login page.

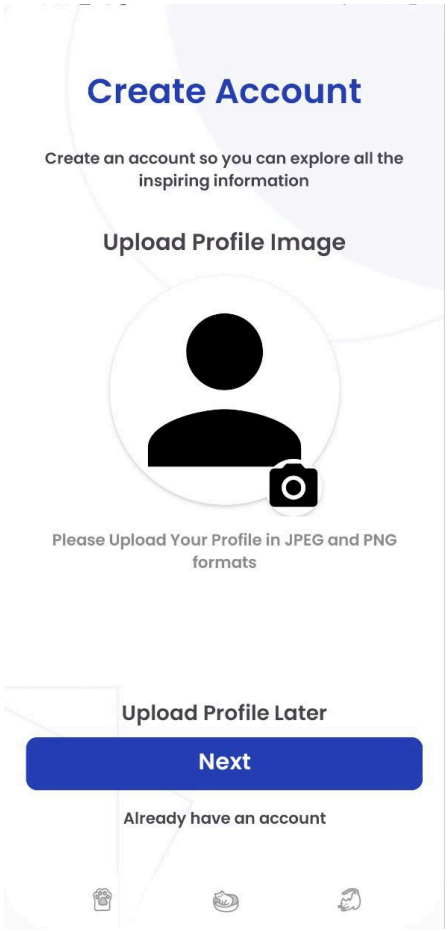


Figure 5.1.8.4

Company Registration Activity
Companies are required to upload profile images. They can choose to upload an image from their device or take a photo directly. The "Already have an account" link redirects existing companies to the login page.

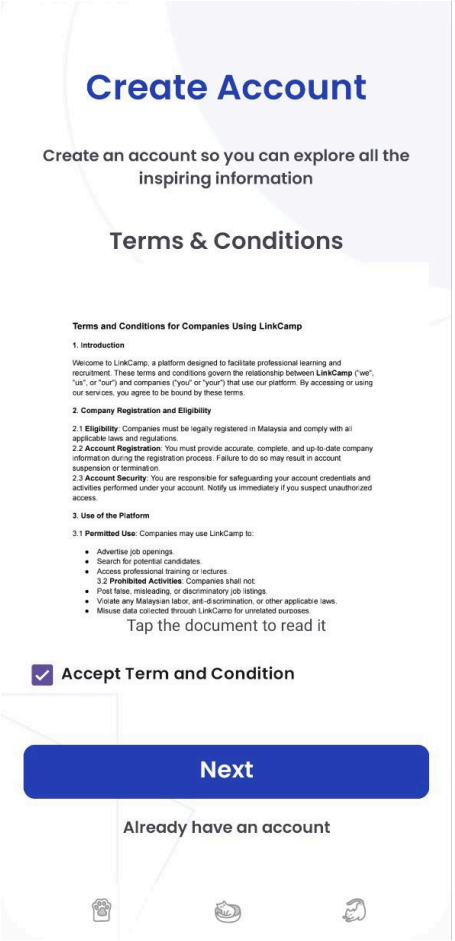


Figure 5.1.8.5

Company Registration Activity
Companies must read and accept the terms and conditions in order to complete the registration process. The "Already have an account" link redirects existing companies to the login page.

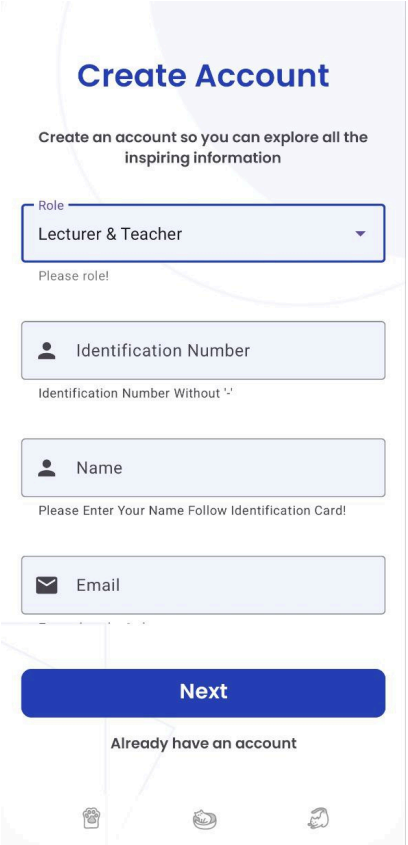


Figure 5.1.9.0

Lecturer Registration Activity

Lecturers are required to provide their details, including their identification number, name, and email. The system will verify that the identification number and email are unique and not already registered. The "Already have an account" link redirects existing lecturers to the login page.

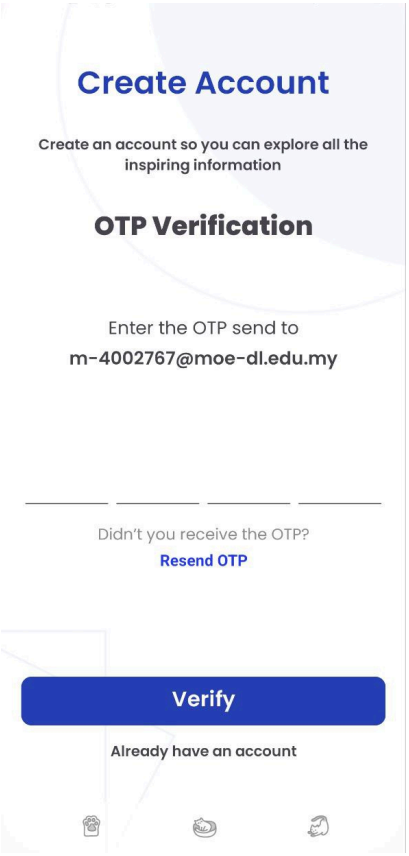


Figure 5.1.9.1

Lecturer Registration Activity (Verification)

After the lecturers enter their information on the registration page, they will be directed to the OTP verification page (Figure 5.1.9.1). On this page, an OTP code will be sent to the lecturer's email. The lecturers must enter the OTP code in the provided field. If the entered OTP matches the code sent, the lecturer can proceed with registration by clicking the "Verify" button. If the OTP does not match, an error message will be displayed, and the lecturer will have two more attempts to enter the correct code. If the lecturer does not receive the OTP, they can click the "Resend OTP" link to request the system to send the code again. The "Already have an account" link redirects existing lecturers to the login page.

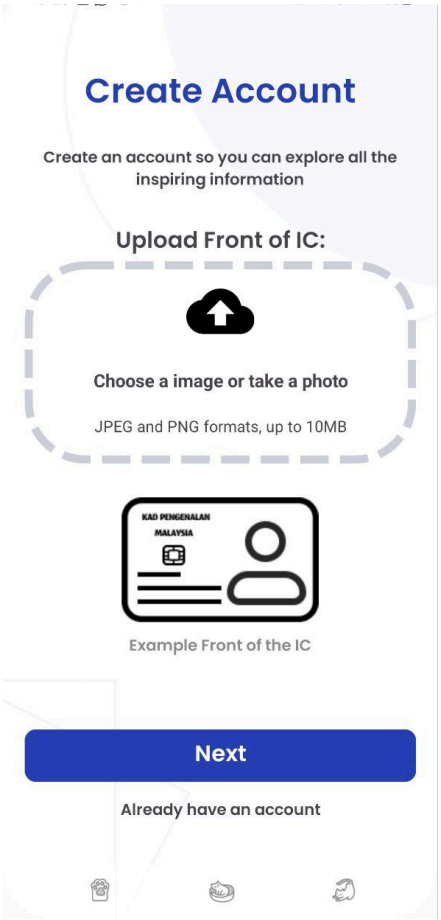


Figure 5.1.9.2

Lecturer Registration Activity

Lecturers are required to upload the front side of their identification card (IC) to verify the authenticity of their account. They can choose to upload an image from their device or take a photo directly. The "Already have an account" link redirects existing lecturers to the login page.

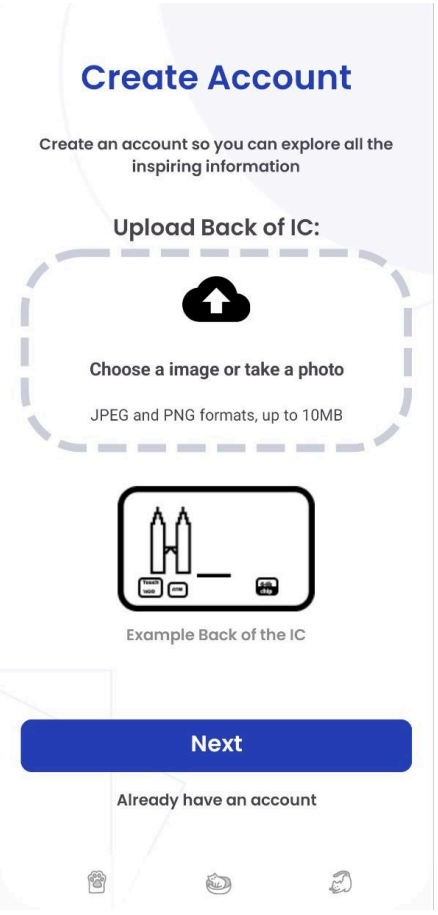


Figure 5.1.9.3

Lecturer Registration Activity

Lecturers are required to upload the back side of their identification card (IC) to verify the authenticity of their account. They can choose to upload an image from their device or take a photo directly. The "Already have an account" link redirects existing lecturers to the login page.

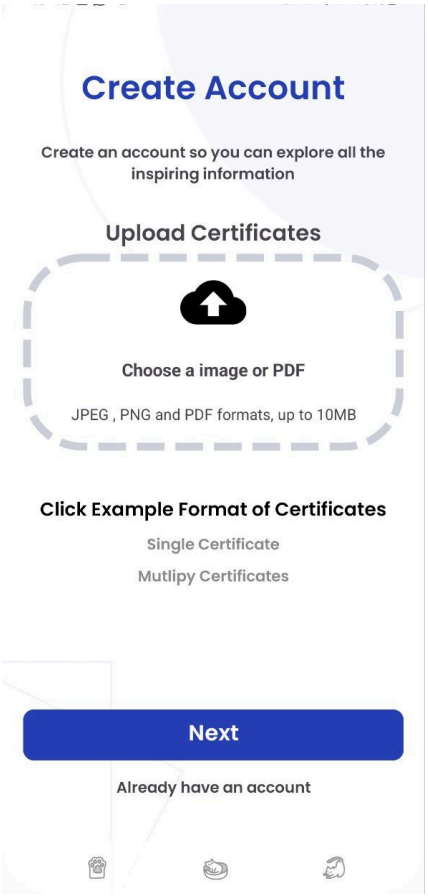


Figure 5.1.9.4

Lecturer Registration Activity

Companies are required to upload their SSM certificates to verify the authenticity of their account. They can choose to upload an image or pdf from their device. The "Already have an account" link redirects existing companies to the login page.

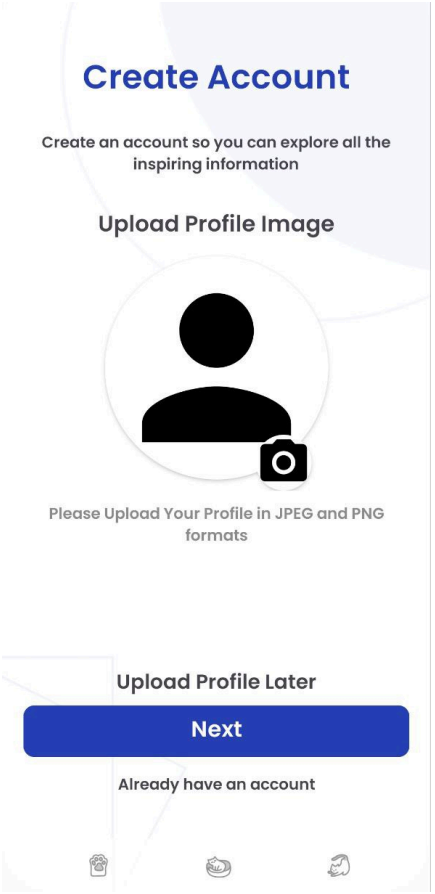


Figure 5.1.9.5

Lecturer Registration Activity

Lecturers are required to upload profile images. They can choose to upload an image from their device or take a photo directly. The "Already have an account" link redirects existing lecturers to the login page.

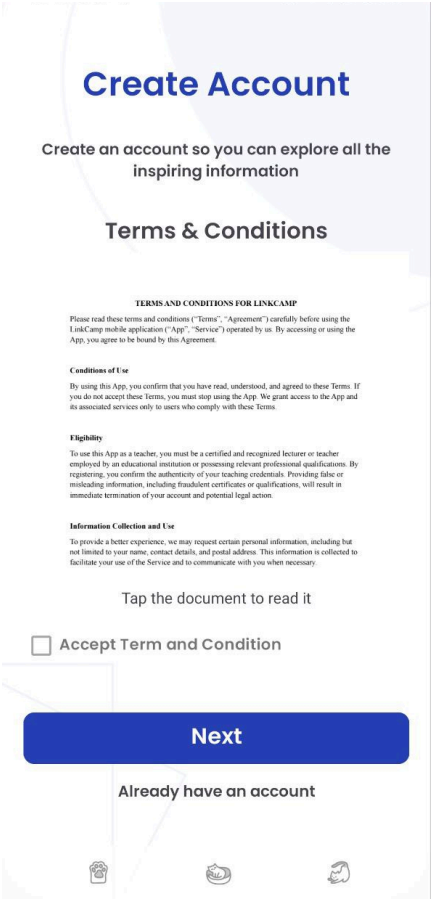


Figure 5.1.9.6

Lecturer Registration Activity

Lecturers must read and accept the terms and conditions in order to complete the registration process. The "Already have an account" link redirects existing lecturers to the login page.

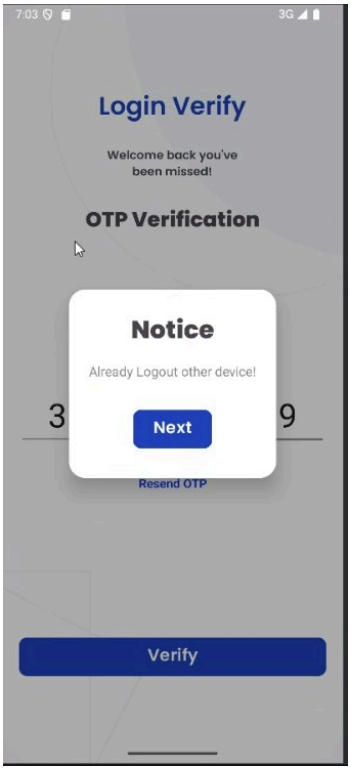


Figure 5.1.10.0

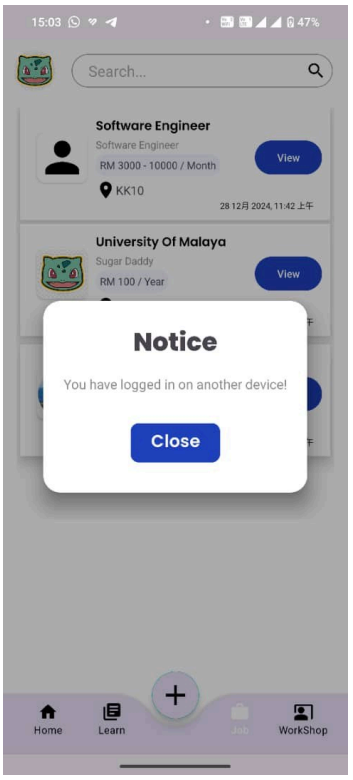


Figure 5.1.10.1

Account Device Management

When a user logs into their account on a new device (Figure 5.1.10.0), the previously logged-in device will be automatically logged out (Figure 5.1.10.1). A notification will be displayed on both devices to inform the user of this action.

5.2 User Profile Module

5.2.1 Profile Activity

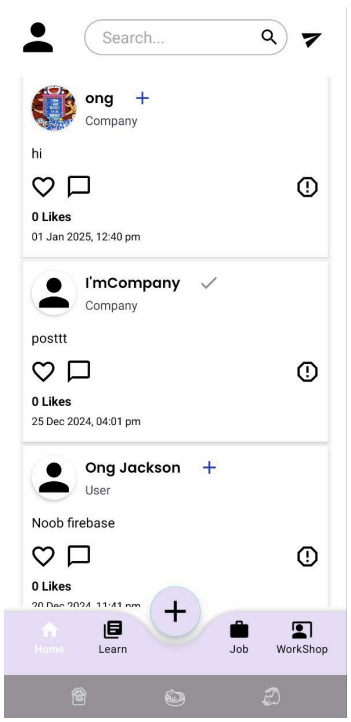


Figure 5.2.1.1.0

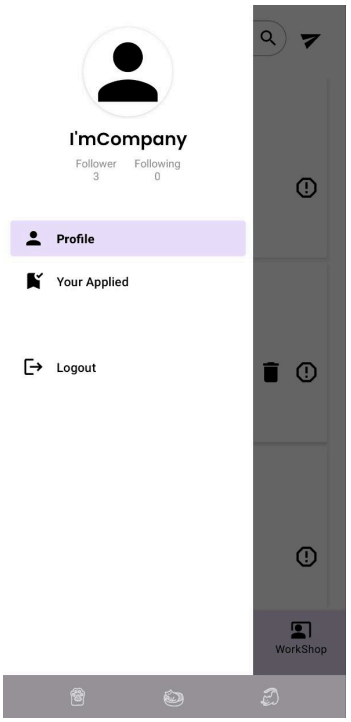


Figure 5.2.1.1.1

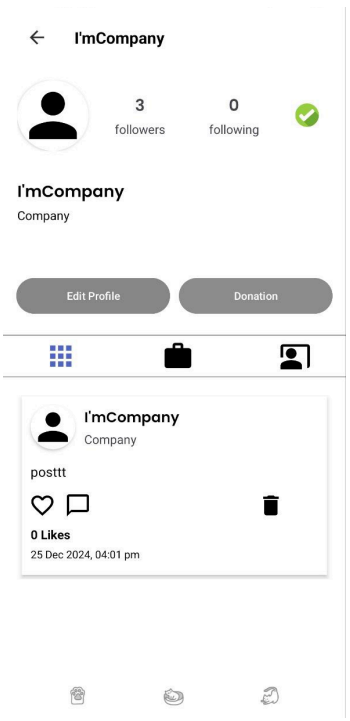


Figure 5.2.1.1.2

Navigate to Profile Activity

Users can access the profile activity by clicking the profile icon located in the upper left corner of the app bar (Figure 5.2.1.1.0). Next, they can select the profile button from the sidebar menu (Figure 5.2.1.1.1), which will redirect them to the profile page (Figure 5.2.1.1.2).

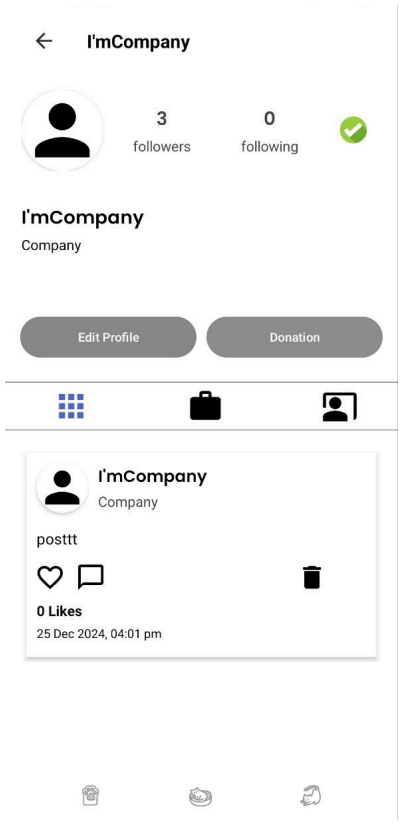


Figure 5.2.1.2.0

Edit Profile Activity

Users can edit their profile by clicking the “Edit Profile” button in their profile page (Figure *5.2.1.1.2*).

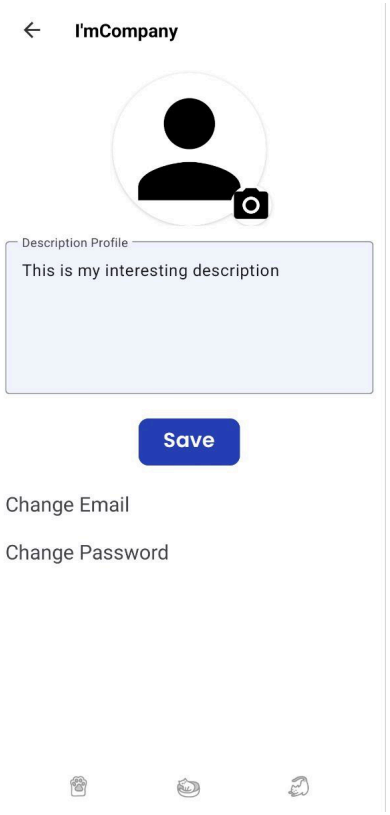


Figure 5.2.1.2.1

Edit Profile Activity

Clicking the "Edit Profile" button will direct users to this page (Figure 5.2.1.2.0). Here, users can update their profile description by entering text into the provided edit box and clicking the "Save" button to save the changes. Additionally, users can update their email by selecting the "Change Email" link or modify their password by clicking the "Change Password" link.

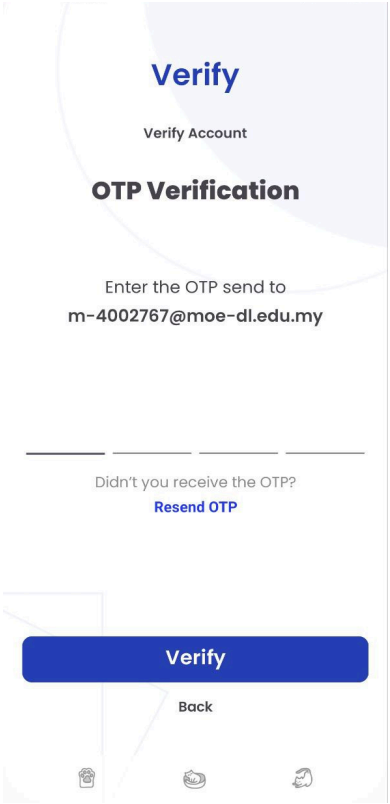


Figure 5.2.1.3.0

Change Email Activity

After clicking the “Change Email” link, users will be directed to this page (Figure *Figure 5.2.1.3.0*). An OTP code will be sent to the user’s email, which they must enter into the designated field. If the entered OTP matches the code sent, users can proceed to change their email by clicking the "Verify" button. If the OTP does not match, an error message will appear, and the user will have two additional attempts to enter the correct code. If the OTP is not received, users can click the "Resend OTP" link to request a new code. The “Back” button will redirect users to the previous page.

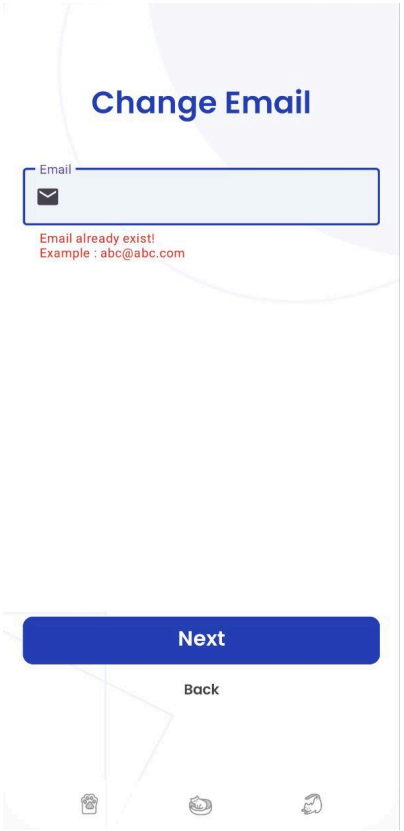


Figure 5.2.1.3.1

Change Email Activity

After the OTP code is successfully verified, users can enter their new email address, ensuring it adheres to the required format. The new email must not be an address already associated with an existing account. After entering the new email, users can click the “Next” button to proceed. The “Back” button will redirect users to the previous page.

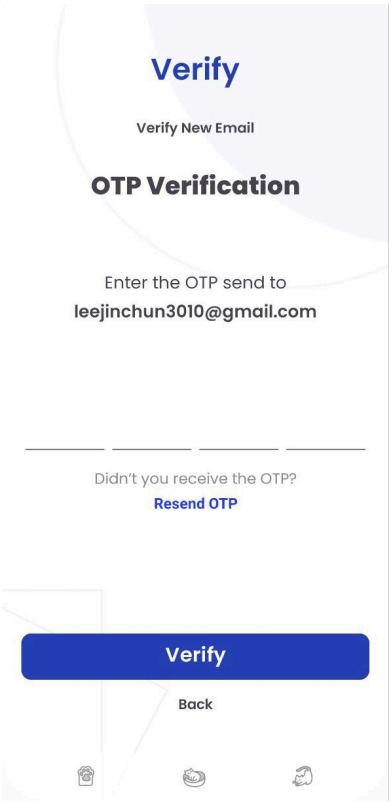


Figure 5.2.1.3.2

Change Email Activity
After clicking the "Next" button (Figure 5.8.3.2), users will be directed to this page (Figure 5.8.3.0). An OTP code will be sent to the user's new email, which must be entered in the designated field. If the entered OTP matches the sent code, users can successfully change their email by clicking the "Verify" button, after which a success notification will be displayed. If the OTP does not match, an error message will appear, and users will have two additional attempts to enter the correct code. If the OTP is not received, users can click the "Resend OTP" link to request a new code. The "Back" button will redirect users to the previous page.

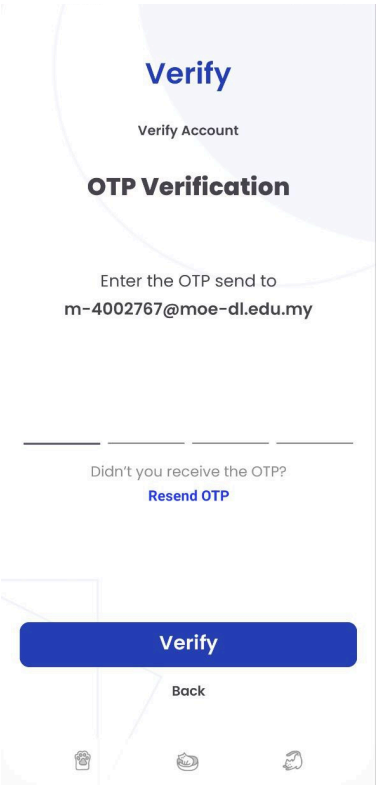


Figure 5.2.1.4.0

Change Password Activity
After clicking the "Change Password" link, users will be directed to this page (Figure 5.8.3.0). An OTP code will be sent to the user's email, which they must enter into the designated field. If the entered OTP matches the code sent, users can proceed to change their email by clicking the "Verify" button. If the OTP does not match, an error message will appear, and the user will have two additional attempts to enter the correct code. If the OTP is not received, users can click the "Resend OTP" link to request a new code. The "Back" button will redirect users to the previous page.

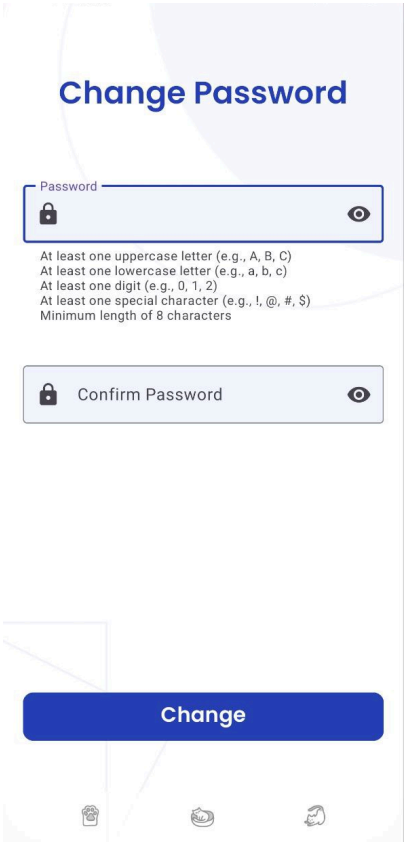


Figure 5.2.1.4.1

Change Password Activity

Users can change their password by entering a new password that meets the specified requirements (Figure 5.2.1.4.1). If the password does not comply, an error message will be displayed. Users are also required to re-enter the password for confirmation. If both entries match, the password will be successfully changed, and a success message will be displayed. Otherwise, an error message will be shown.

5.2.2 Job Application Pages

This page simplifies the process of applying for jobs, displaying opportunities in an organized and easy-to-navigate manner.

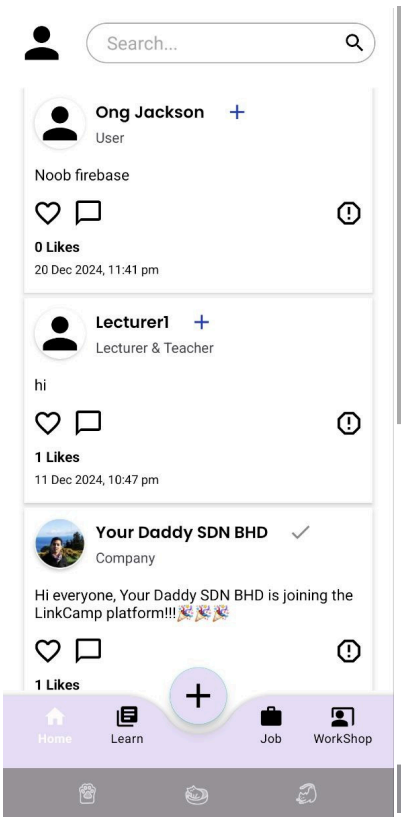


Figure 5.2.2.1.0

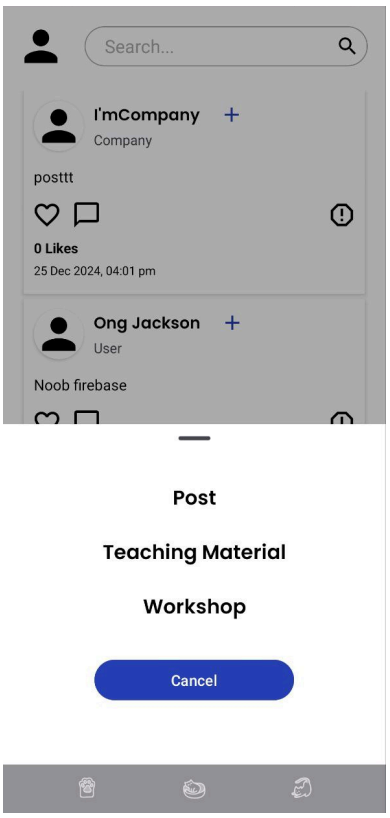


Figure 5.2.2.1.1

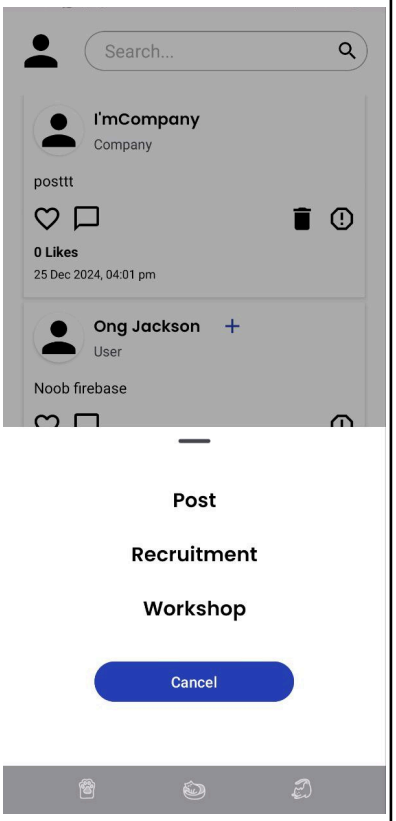


Figure 5.2.2.1.2

Home Page Activity

- By clicking the "+" symbol at the lower center, **users** will be directed to the post page.
- By clicking the "+" symbol at the lower center, **lecturers** can create posts, teaching materials, or workshops. Each option will direct the lecturer to a different page (Figure 5.2.2.1.1).
- By clicking the "+" symbol at the lower center, **companies** can create posts, recruitment posts, or workshops. Each option will direct the company to a different page (Figure 5.2.2.1.2).

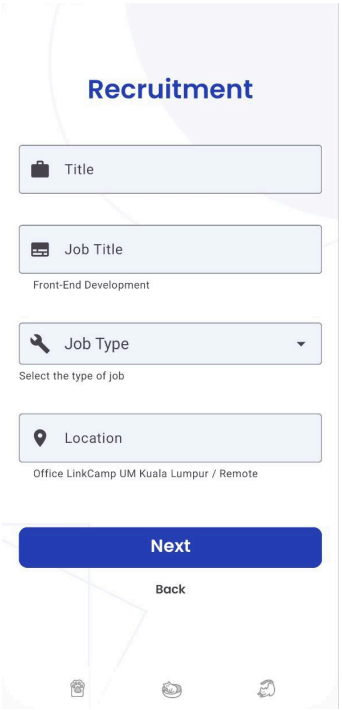


Figure 5.2.2.2.0

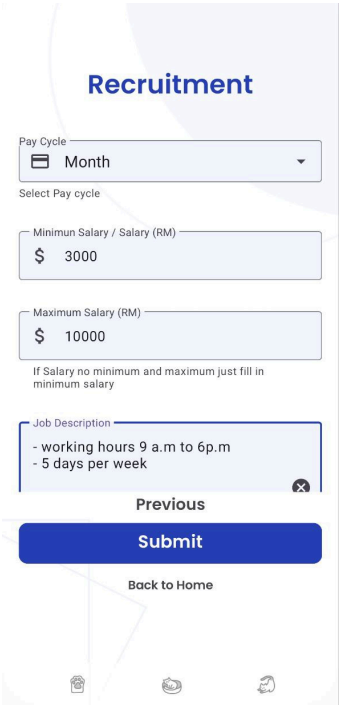


Figure 5.2.2.2.1

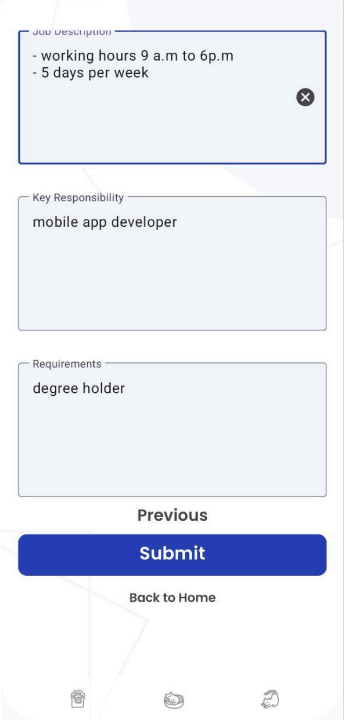


Figure 5.2.2.2.2

Create Recruitment Activity

By clicking the "+" symbol at the lower center, companies can create recruitment posts (Figure 5.2.2.2.0). Companies are required to fill in details such as the title, job title, job type (part-time, internship, full-time, or freelance), and location. After completing this step, they can click the "Next" button to proceed. The "Back" button will return the company to the previous page (Figure 5.2.2.2.0).

In the next step, companies need to provide additional information, including the pay cycle (year, month, week, day, or hour), minimum and maximum salary, job description, key responsibilities, and requirements. Once completed, they can click the "Submit" button to finalize and upload the recruitment details. The "Previous" button will navigate back to the previous page, while the "Back to Home" button will return to the home page (Figures 5.2.2.2.1 and 5.2.2.2.2).

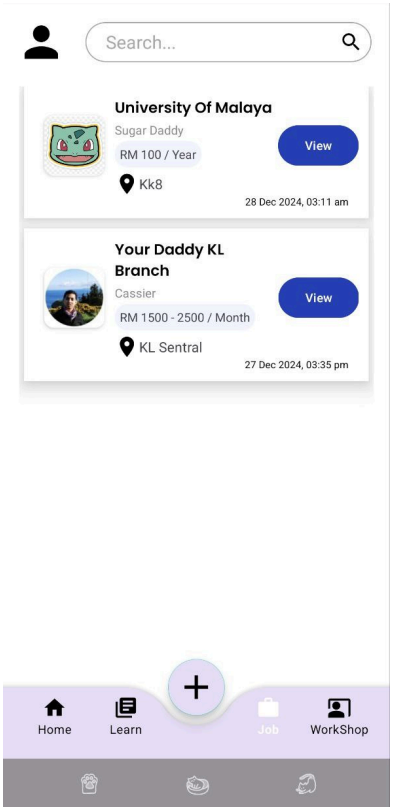


Figure 5.2.2.3

Job Page Activity

All users can click on the "Job" icon in the bottom navigation bar to view details of available job applications. They can see information such as the company offering the job, the job title, a brief description, and the date and time the job was posted. By clicking the "View" button, users can access more detailed information.

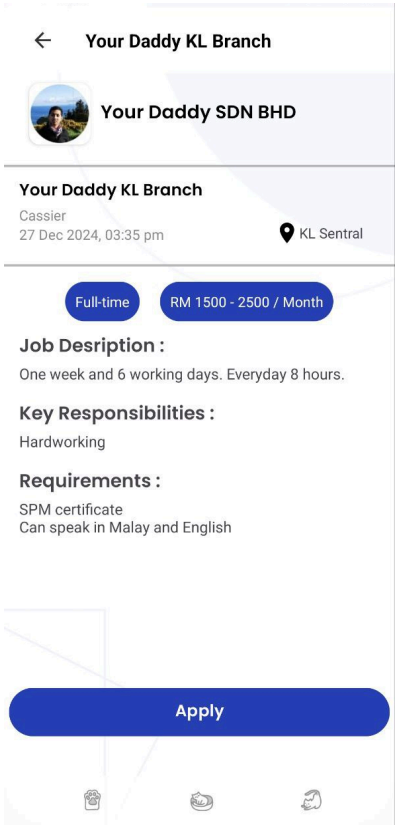


Figure 5.2.2.4.0

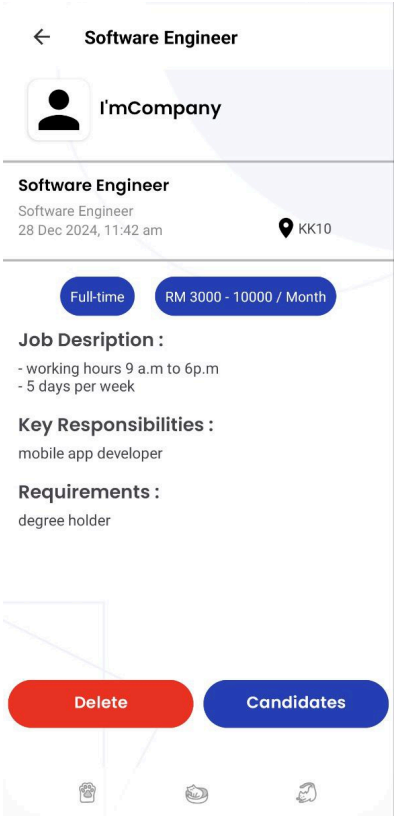


Figure 5.2.2.4.1

View Job Activity

After clicking the "View" button on the job page (Figure 5.2.2.3), users or lecturers are directed to the job details page (Figure 5.2.2.4.0). Here, they can view detailed job information, including the company offering it, the title, job title, location, job type, salary range, pay cycle, job description, key responsibilities, and requirements.

Users and lecturers can apply for the job by clicking the "Apply" button. **Only users and lecturers are eligible to apply.**

For **companies** that posted the job, clicking the "View" button redirects them to a similar job page (Figure 5.2.2.4.1), where they can view the job details listed above, delete the job post by clicking the "Delete" button and view applicants by clicking the "Candidates" button.

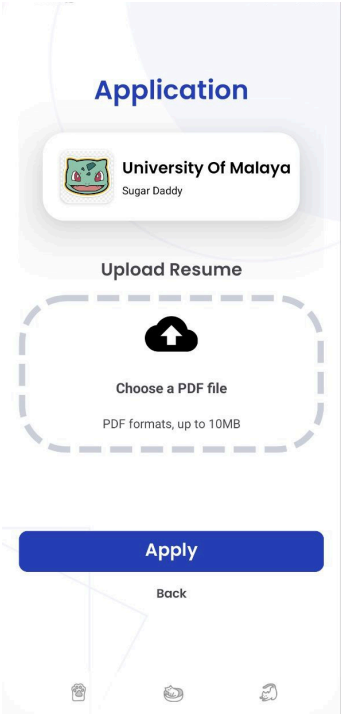


Figure 5.2.2.5

Apply Job Activity
After clicking the "Apply" button on the job details page (Figure 5.2.2.2.4.0), users or lecturers will be directed to this page (Figure 5.2.2.5). Here, they are required to upload their resume in PDF format. Once the resume is uploaded, they can apply for the job by clicking the "Apply" button.

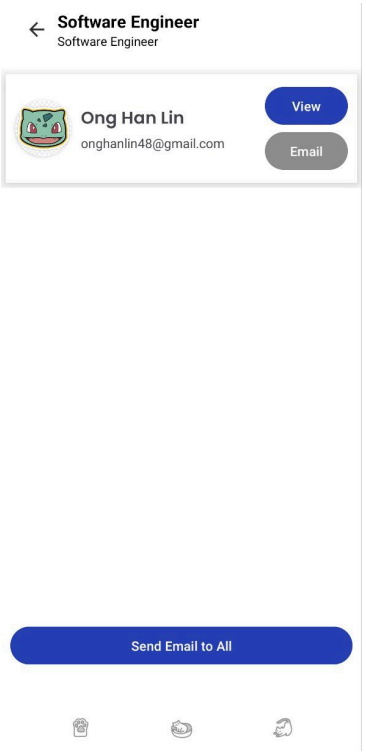


Figure 5.2.2.6

View Candidates Activity
After clicking the "Candidates" button on the job details page (Figure 5.2.2.4.1), the company offering the job can view a list of applicants, including their names and email addresses. The company can click the "View" button to access a candidate's uploaded resume, the "Email" button to send an email to an individual candidate, or the "Send Email to All" button to email all candidates simultaneously.

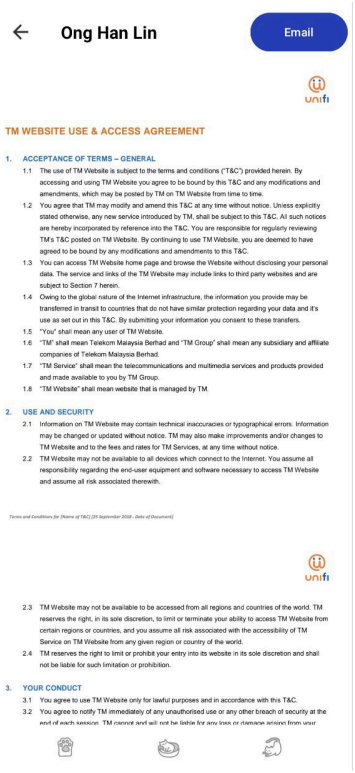


Figure 5.2.2.7

View Resume Activity

After clicking the "View" button on the candidates page (Figure 5.2.2.6), the candidate's resume will be displayed (Figure 5.2.2.7).

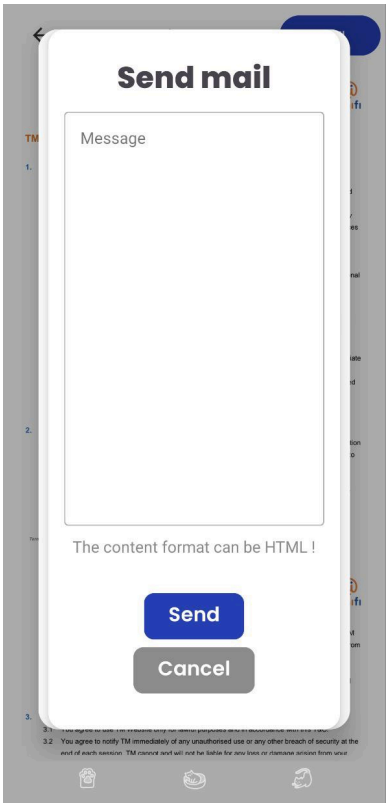


Figure 5.2.2.8

Send Email Activity

After clicking the "Send Email" or "Send Email to All" button, the company will be directed to this activity. Here, the company can compose the email content in the "Message" text editor. Clicking the "Send" button will send the email, while clicking the "Cancel" button will discard the email and exit the activity.

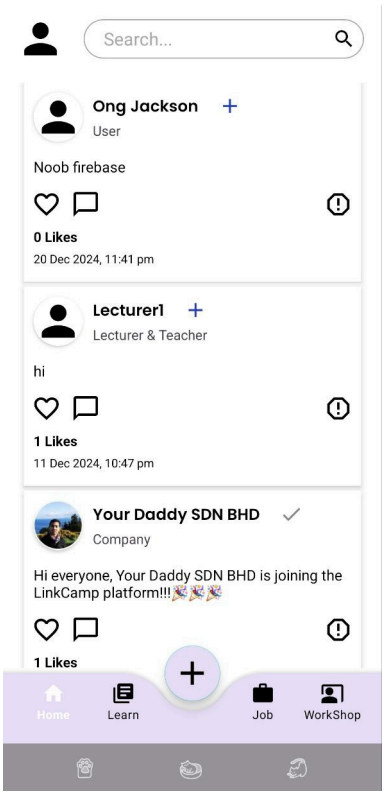


Figure 5.2.2.9.0

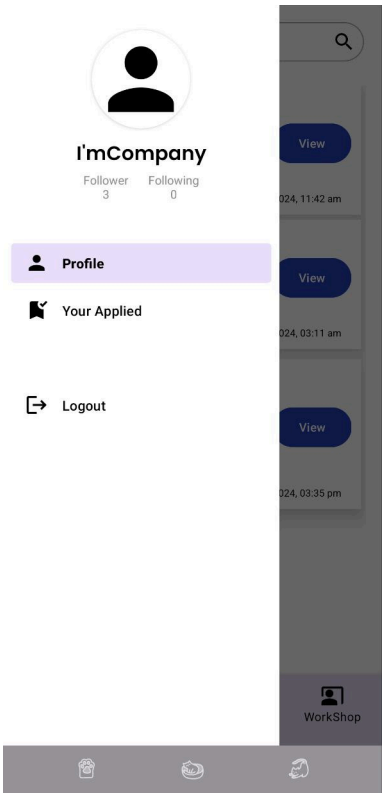


Figure 5.2.2.9.1

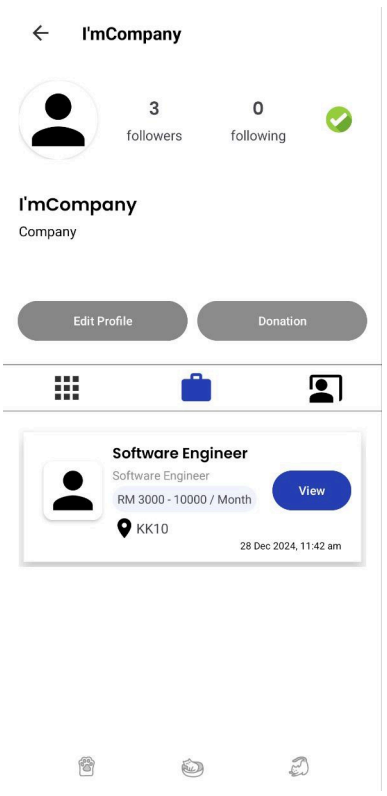
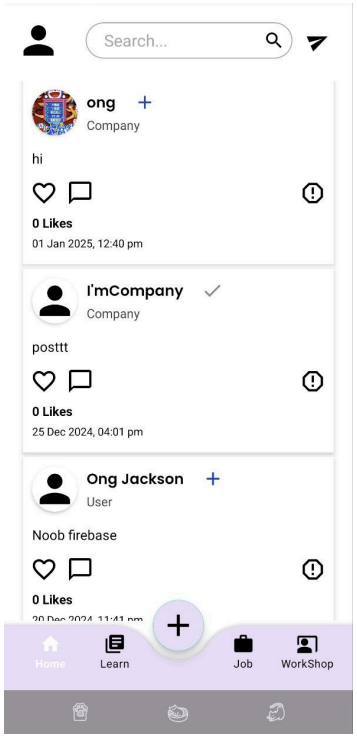


Figure 5.2.2.9.2

Profile Created Job Activity

Companies can view their posted recruitment activities on the profile page by clicking the profile icon located in the upper left corner (Figure 5.2.2.9.0). After clicking the icon, a sidebar will appear (Figure 5.2.2.9.1). Selecting the "Profile" link will redirect the user to the profile page (Figure 5.2.2.9.2).

On the profile page, creators can click the center icon in the row of three icons to display their posted recruitment posts. By clicking the "View" button, creators will be redirected to the detailed recruitment activity page (Figure 5.2.2.4.1).



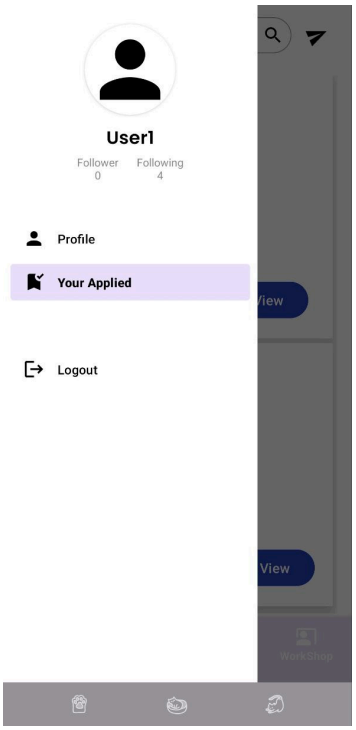


Figure 5.2.2.10.0 *Figure 5.2.2.10.1* *Figure 5.2.2.10.2*

View Applied Job Activity

Users can view applied jobs by clicking the profile icon in the app bar (Figure 5.2.2.10.0), then selecting "Your Applied" from the sidebar menu (Figure 5.2.2.10.1). This will direct them to the applied page (Figure 5.2.2.10.2). By clicking the job button, users will be taken to the applied job page (Figure 5.3.2.10.2).

5.2.3 Workshop Creation and Suggestions Pages

These features allow lecturers and users to create and manage workshops, with suggestions tailored to user preferences for an enhanced experience.

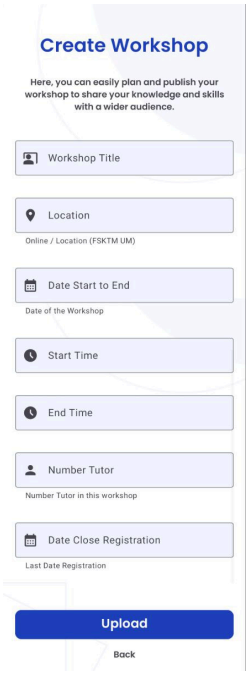


Figure 5.2.3.1.0

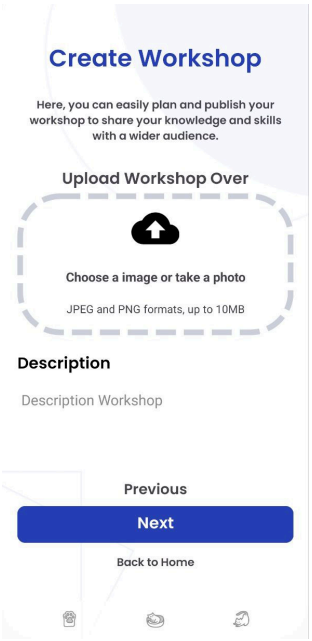


Figure 5.2.3.1.1

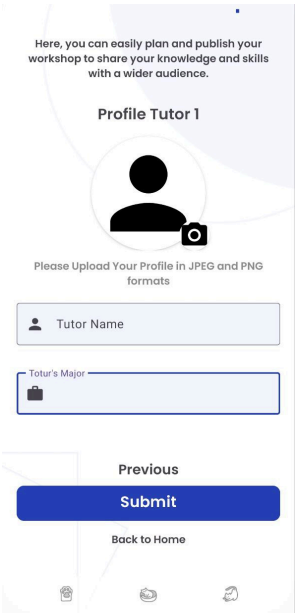


Figure 5.2.3.1.2

Create Workshop Activity

By clicking the "+" symbol at the lower center on the home page, lecturers can create workshops (Figure 5.2.1.1.0). On this page, lecturers and companies can create a workshop by providing details such as the workshop title, location, start and end dates, start and end times, number of tutors, and the registration closing date (Figure 5.2.3.1.0). The "Back" link will return them to the previous page.

After entering the details, they can click the "Next" button to proceed. They will then have the option to upload a workshop image and provide a description of the workshop. (Figure 5.2.3.1.1)

After entering the details, they can click the "Next" button to proceed. They will have the option to upload a profile image, either by selecting a file from their device or by taking a photo directly. Additionally, they need to enter the tutor's name and major. Once completed, they can click the "Submit" button to create the workshop. The "Previous" link will return them to the previous page, and the "Back to Home" link will direct them to the home page (Figure 5.2.3.1.2).

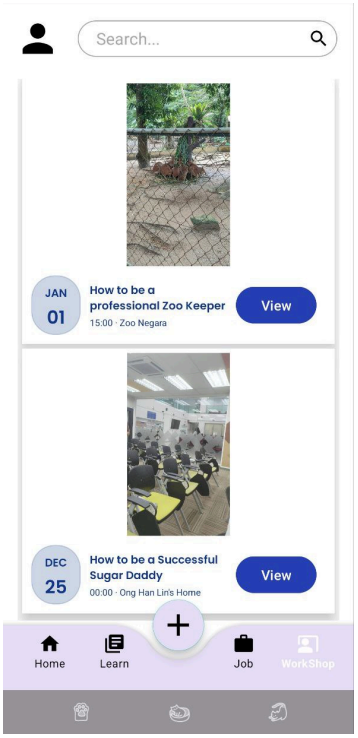


Figure 5.2.3.2

Workshop Activity

On this page, all users can view workshops created by lecturers and companies (Figure 5.2.3.2). Details such as the workshop image, start date, start time, and location will be displayed. By clicking the "View" button, users will be directed to the relevant workshop page.

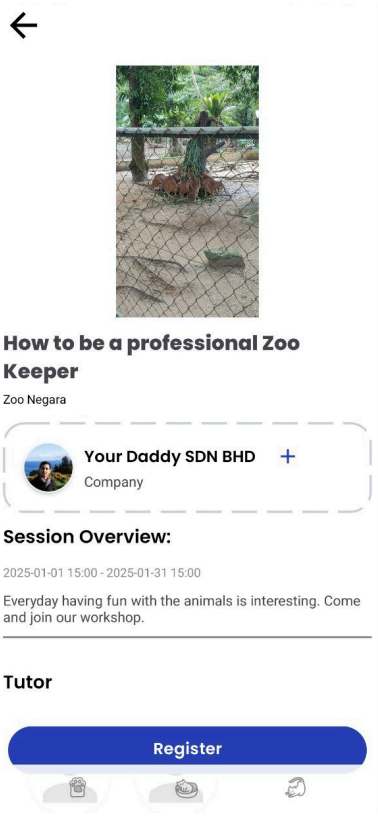


Figure 5.2.3.3.0

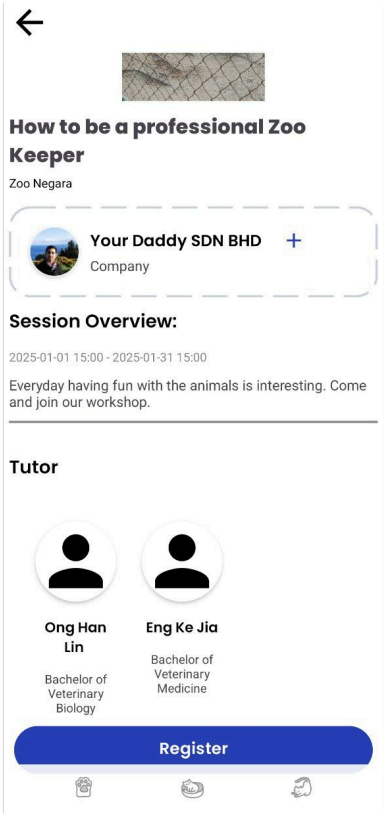


Figure 5.2.3.3.1

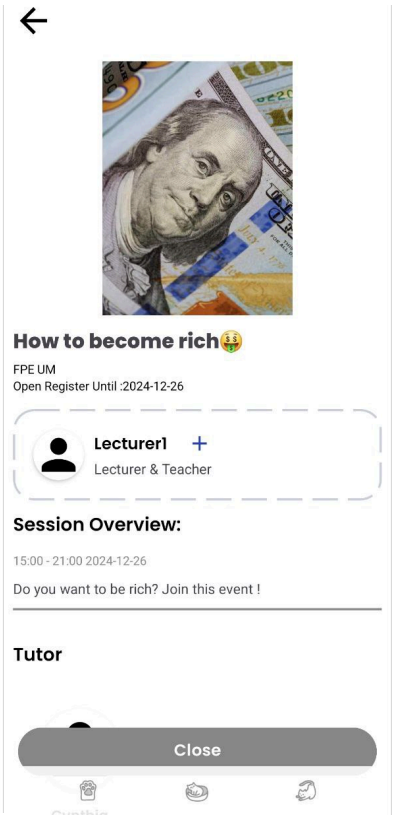


Figure 5.2.3.3.2

Register Workshop Activity

After clicking the "View" button on the workshop page (Figure 5.2.3.2), users will be directed to the related workshop page, which displays detailed information about the workshop. This includes the workshop image, title, location, description, the company or lecturer hosting the workshop, and the tutor(s) involved (Figures 5.2.3.3.0 and 5.2.3.3.1).

All users can follow the company or lecturer hosting the workshop by clicking the "+" symbol next to their name.

Additionally, users can register for the workshop by clicking the "Register" button at the lower center of the page. If they wish to unregister, they can click the same button again to cancel their registration.

The workshop registration will close on the day the workshop is held (Figure 5.2.3.3.2).

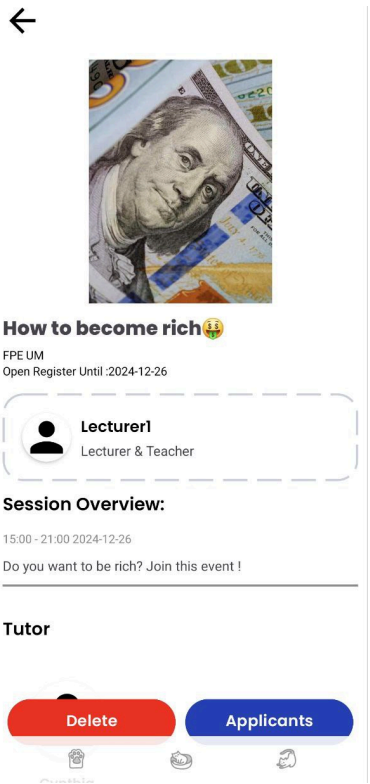


Figure 5.2.3.4

Created Workshop Activity

After clicking the "View" button on the workshop page (Figure 5.2.3.3), the workshop creator will be directed to the related workshop page, which displays detailed information about the workshop. This includes the workshop image, title, location, description, the company or lecturer hosting the workshop, and the tutor(s) involved.

The workshop creator has the option to delete the workshop by clicking the "Delete" button or view the list of applicants by clicking the "Applicants" button. (Figure 5.2.3.4)

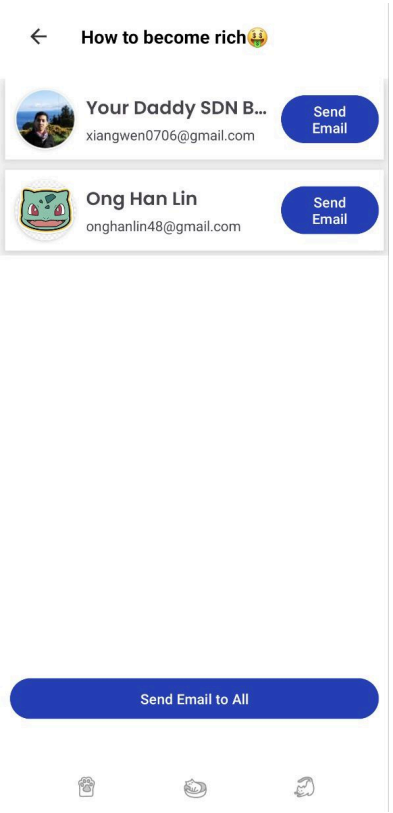


Figure 5.2.3.5

Contact Applicants Activity

After clicking the "Applicants" button on the created workshop page (Figure 5.2.3.4), a list of workshop applicants will be displayed. The workshop creator can send an email to individual applicants by clicking the "Send Email" button next to their name or send an email to all applicants at once by clicking the "Send Email to All" button at the lower center of the page.

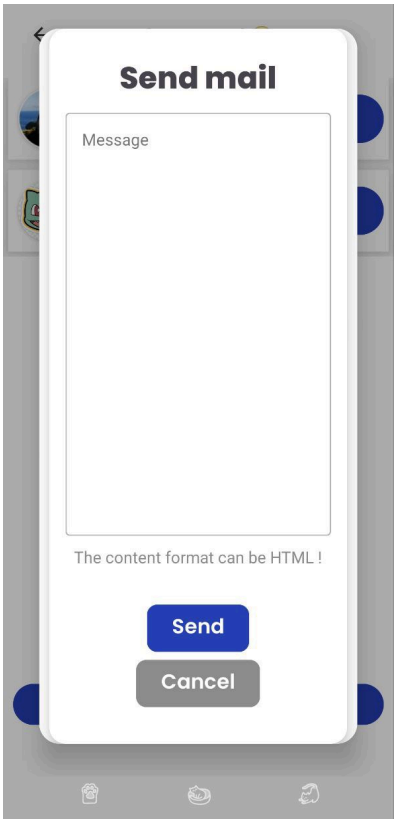


Figure 5.2.3.6

Send Email Activity
After clicking the "Send Email" or "Send Email to All" button, the workshop creator will be directed to this activity. Here, the workshop creator can compose the email content in the "Message" text editor. Clicking the "Send" button will send the email, while clicking the "Cancel" button will discard the email and exit the activity.

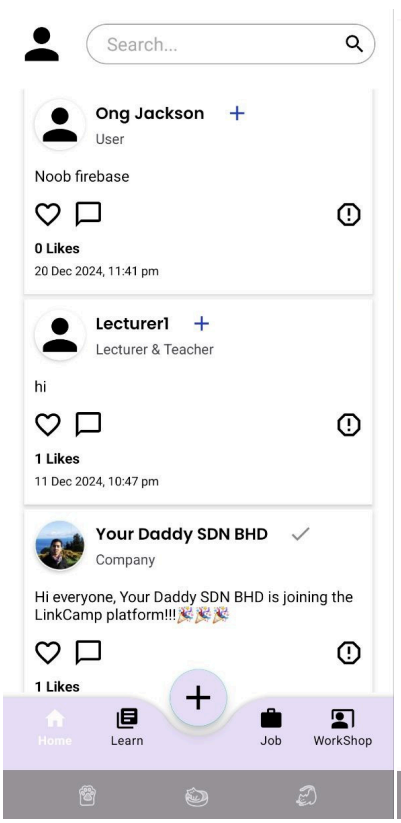


Figure 5.2.3.7.0

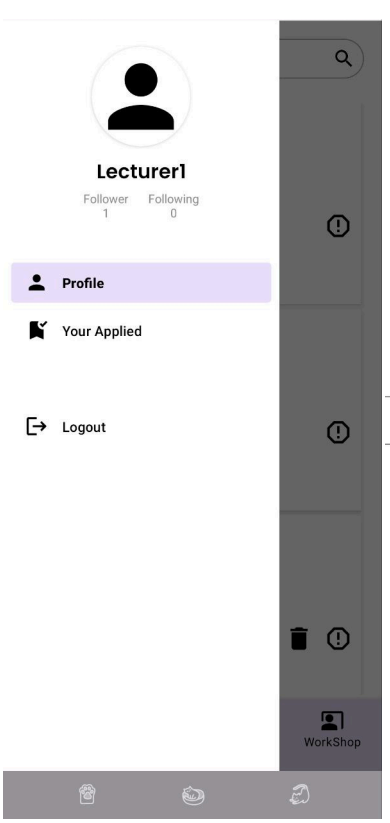


Figure 5.2.3.7.1

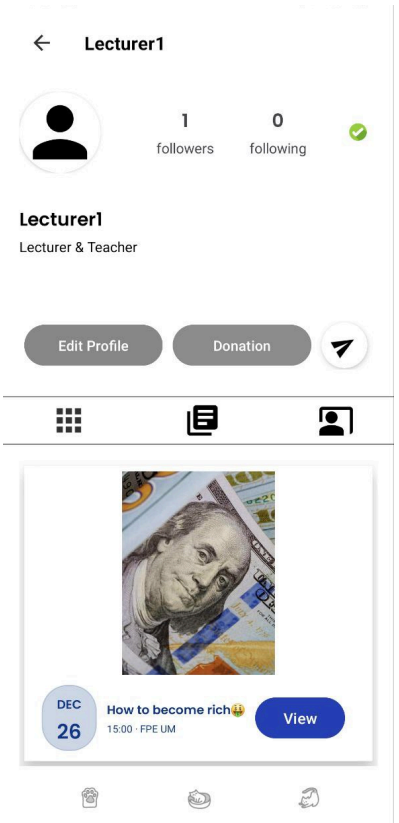


Figure 5.2.3.7.2

Profile Created Workshop Activity

Workshop creators can view their created workshops on the profile page by clicking the profile icon located in the upper left corner (Figure 5.2.3.7.0). After clicking the icon, a sidebar will appear (Figure 5.2.3.7.1). Selecting the "Profile" link will redirect the user to the profile page (Figure 5.2.3.7.2).

On the profile page, creators can click the rightmost icon in the row containing three icons to display their created workshops. By clicking the “view” button creators will be redirect to created workshop activity (Figure 5.2.3.4).

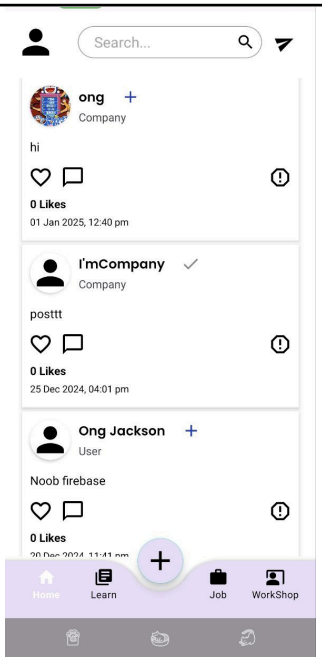


Figure 5.2.3.8.0

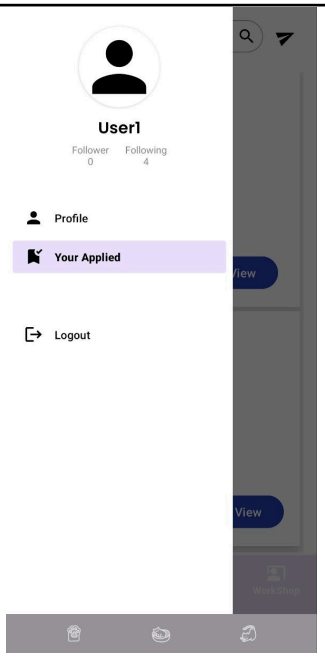


Figure 5.2.3.8.1

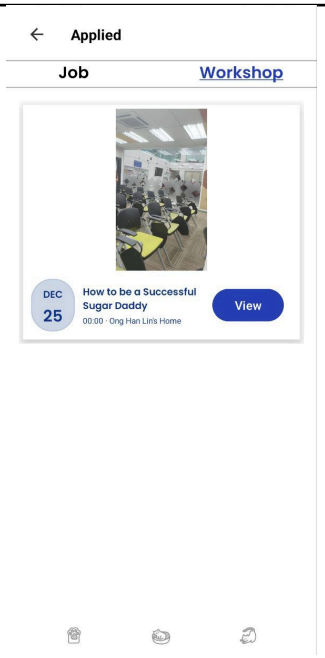


Figure 5.2.3.8.2

View Applied Workshop Activity

Users can view applied workshops by clicking the profile icon in the app bar (Figure 5.2.3.8.0), then selecting "Your Applied" from the sidebar menu (Figure 5.2.3.8.1). This will direct them to the applied page (Figure 5.2.3.8.2). By clicking the workshop button, users will be taken to the applied workshop page (Figure 5.2.3.8.2).

5.2.4 Post Creation and Discussion Pages

These features allow users, companies and lecturers to create and manage posts and discussion between them.

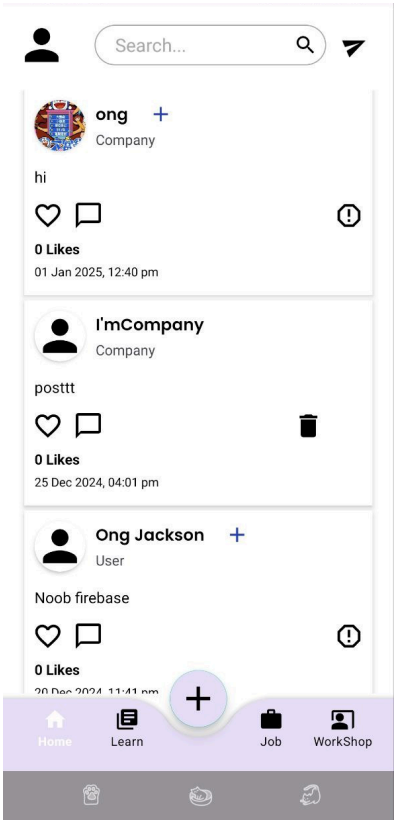


Figure 5.2.4.1

Home Page View Post Activity
All users can review both their own and others' posts on this page (Figure 5.2.4.1). They can like a post by clicking the heart icon, comment on it by clicking the chat box icon, and follow the post's author by clicking the "＋" icon next to the author's name. Additionally, users can report a post by clicking the report icon with the "!" symbol. Users can delete their own post by clicking the dustbin icon on their post.

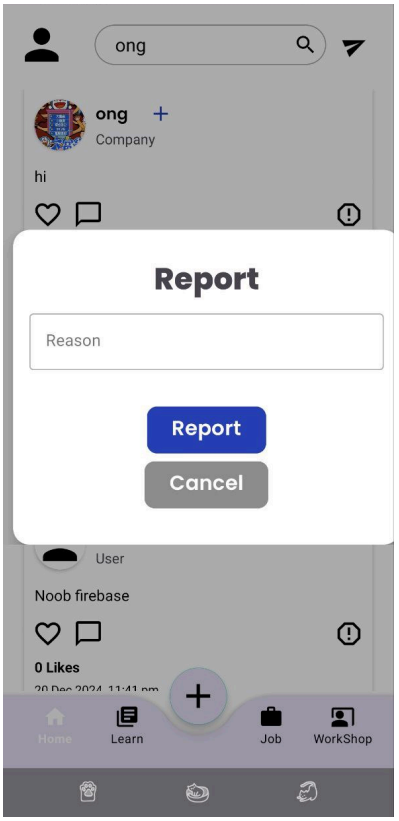


Figure 5.2.4.2

Report Post Activity
By clicking the report icon (Figure 5.2.4.1), users can enter the reason for the report and click the "Report" button to submit it. If users wish to cancel the report, they can click the "Cancel" button.

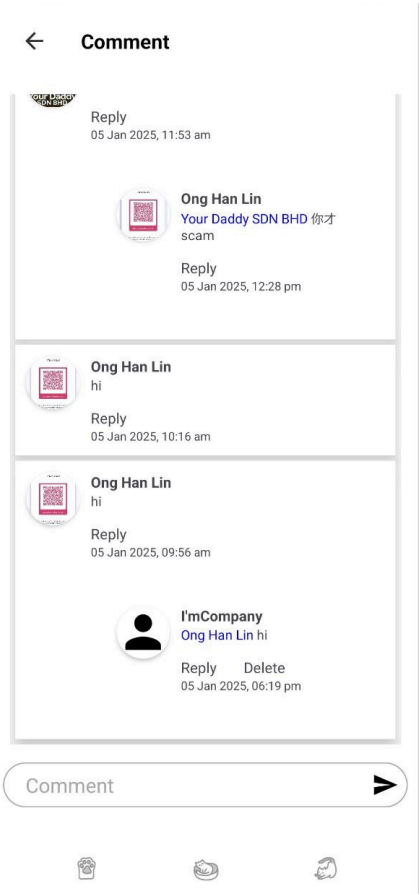


Figure 5.2.4.3

Comment Post Activity

By clicking the chat box icon on a post (Figure 5.2.4.1), users can leave a comment. They can type their comment in the provided EditText box and click the arrow to send it. Users can also reply to other comments by selecting the "reply" link beneath a comment. To reply to their own comment, they can click the "reply" link, and to delete their comment, they can click the "delete" link.

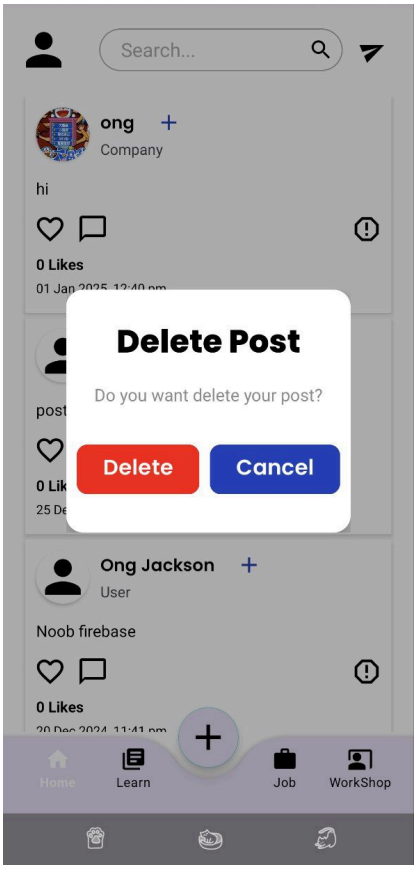


Figure 5.2.4.4

Delete Post Activity

By clicking the dustbin icon on their post (Figure 5.2.4.1), a notification will appear. The user can choose to delete the post by clicking the "Delete" button, or if they wish to cancel, they can click the "Cancel" button.

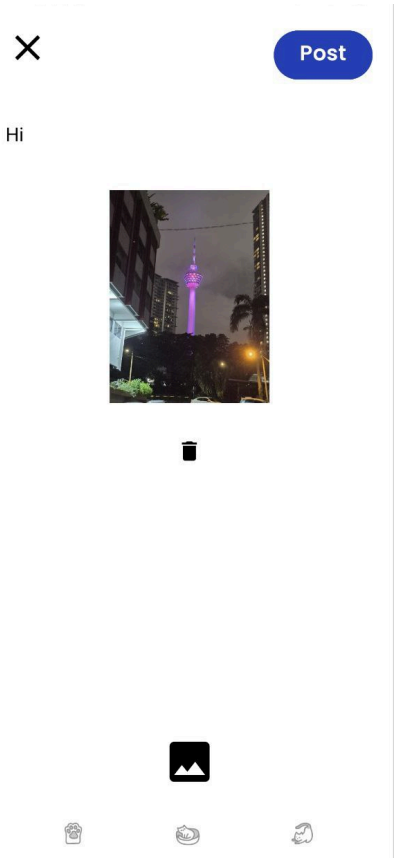


Figure 5.2.4.5

Create Post Activity

By clicking the "+" symbol at the lower center (Figure 5.2.4.1), users will be directed to the post page. All users can write their posts on this page. They can also add a picture to their post by clicking the picture symbol at the lower center. To delete the picture, users can click the trash symbol. Once ready, users can click "Post" to publish their content.

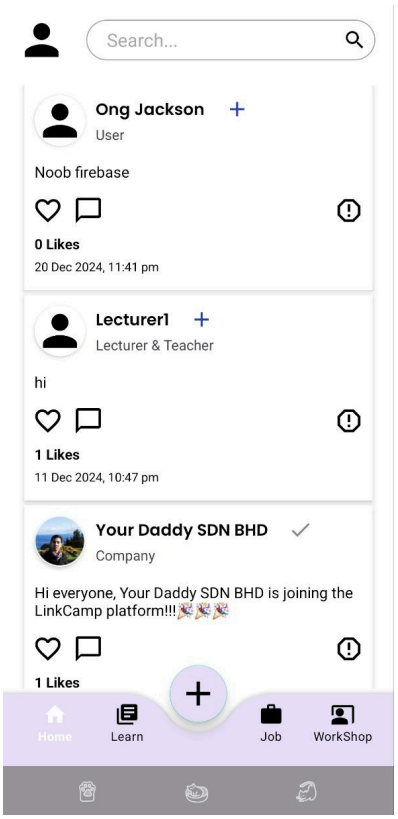


Figure 5.2.4.6.0

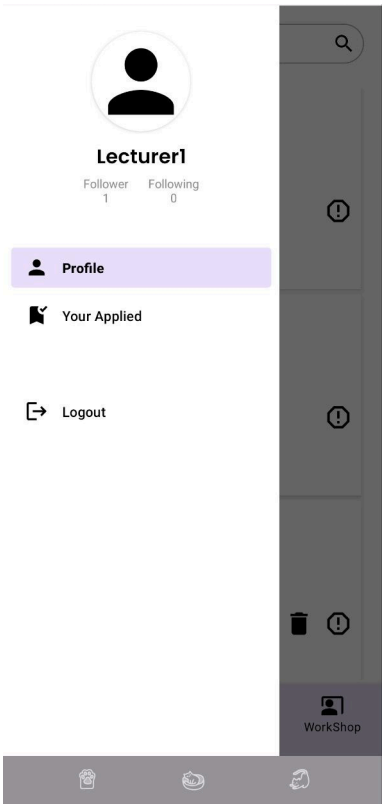


Figure 5.2.4.6.1

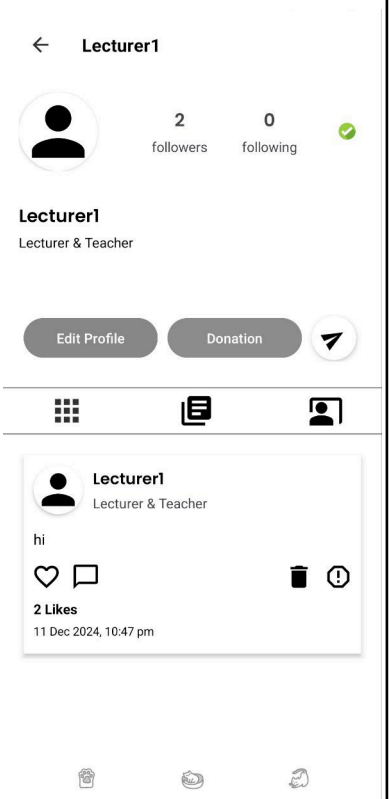


Figure 5.2.4.6.2

Profile Created Post Activity

Post creators can view their created post on the profile page by clicking the profile icon located in the upper left corner (Figure 5.2.4.6.0). After clicking the icon, a sidebar will appear (Figure 5.2.4.6.1). Selecting the "Profile" link will redirect the user to the profile page (Figure 5.2.4.6.2).

On the profile page, creators can click the leftmost icon in the row containing three icons to display their created post. Creators can manage their posts by deleting them using the trash icon, liking them by clicking the heart icon, or commenting using the chatbox icon. Information about their posts, such as the number of likes and comments, is also visible.

5.2.5 Learning Materials Pages

These features allow lecturers to upload learning materials.

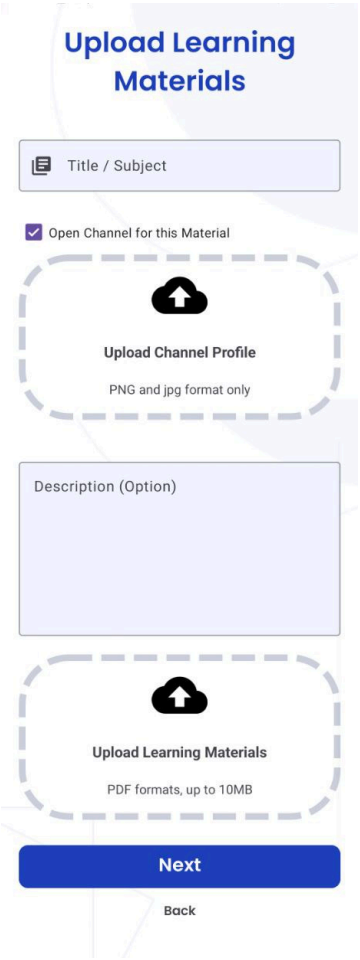


Figure 5.2.5.1

Upload Learning Materials Activity
Lecturers can upload learning materials on this page by providing a title, description, and uploading the materials in PDF format. They may also choose to create a channel for the material by selecting the checkbox, which requires uploading a channel profile. Once all the necessary details are provided, lecturers can click the "Next" button to proceed with the upload process.

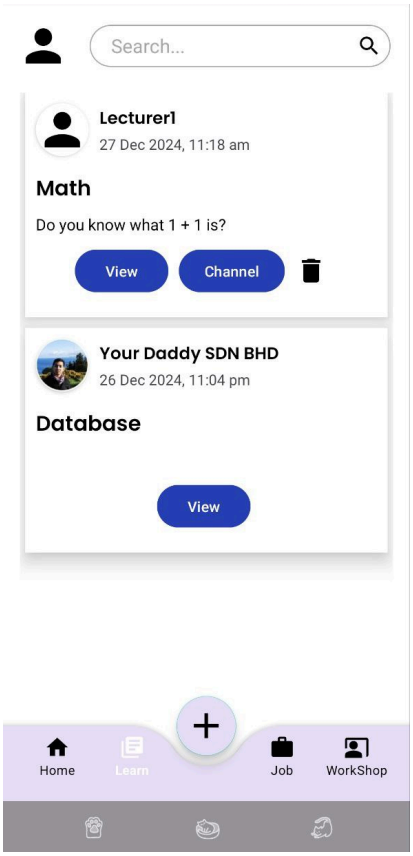


Figure 5.2.5.2

Learn Page
All users can click on the “Learn” icon in the menu to view details of all uploaded learning materials. Users can see the lecturer who uploaded the material, the title, and the description. By clicking the “View” button, users can access the learning material. By clicking the “View” button, users can access the learning material. By clicking the “Channel” button, users can join the learning channel.

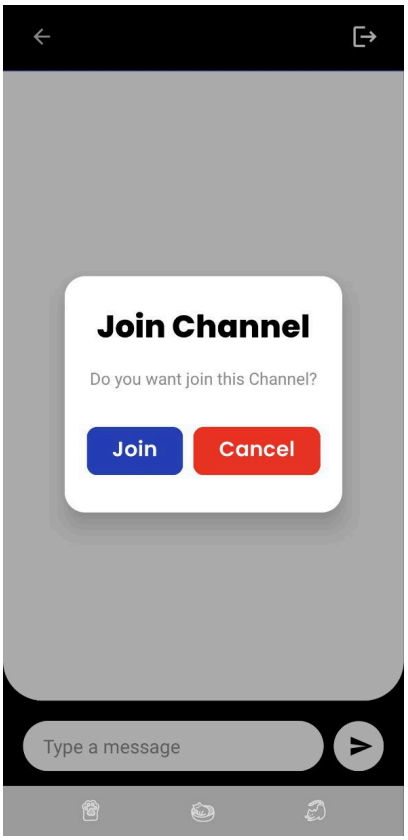


Figure 5.2.5.3

Join Channel Activity

After clicking the "Channel" button on the Learn page (Figure 5.5.2), users will be directed to this page (Figure 5.5.3). A notification will appear, asking if the user wants to join the channel. Clicking the "Join" button will allow the user to join the channel, while clicking the "Cancel" button will return the user to the previous page.

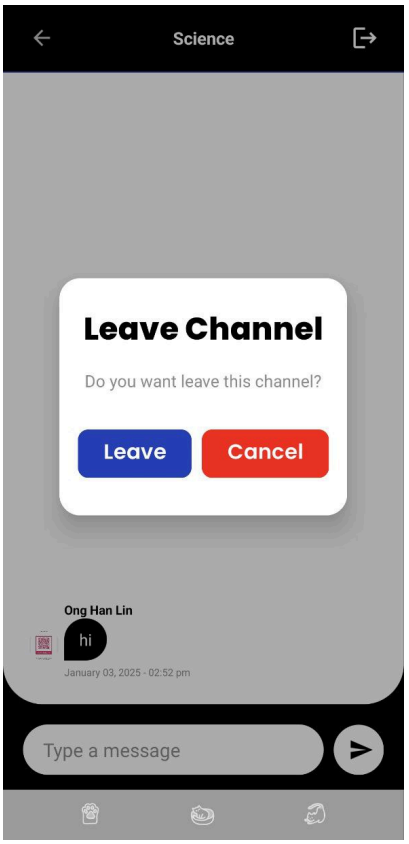


Figure 5.2.5.4

Leave Channel Activity

Users can leave the channel by clicking the leave icon located at the upper right corner of the channel page. A notification will prompt the user to confirm their decision to leave the channel. Clicking the "Leave" button will confirm and complete the action, while clicking the "Cancel" button will dismiss the prompt and keep the user on the current page.

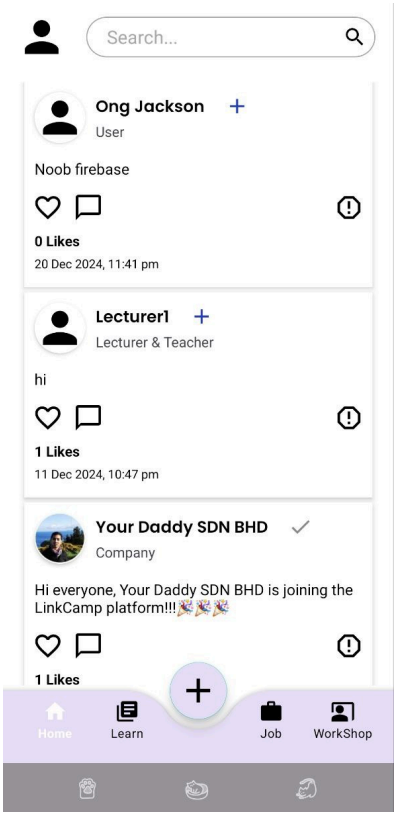


Figure 5.2.5.5.0

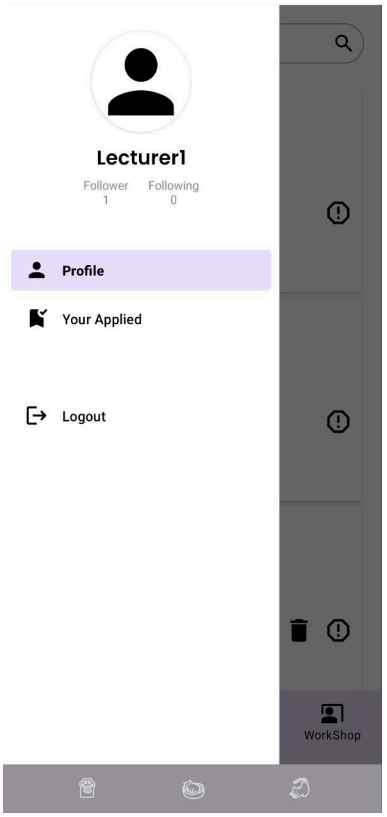


Figure 5.2.5.5.1

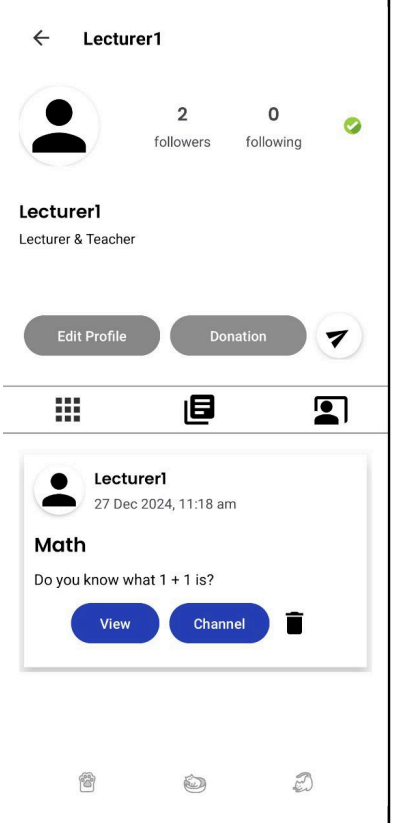


Figure 5.2.5.5.2

Profile Uploaded Teaching Material Activity

Lecturers can view uploaded teaching material on the profile page by clicking the profile icon located in the upper left corner (Figure 5.2.5.5.0). After clicking the icon, a sidebar will appear (Figure 5.2.5.5.1). Selecting the "Profile" link will redirect the lecturer to the profile page (Figure 5.2.5.5.2).

On the profile page, lecturers can click the center icon in the row containing three icons to display their uploaded teaching material. Lecturers can manage their teaching material by deleting them using the trash icon. By clicking the “view” button, lecturers can view the teaching material.

5.2.6 Search Pages

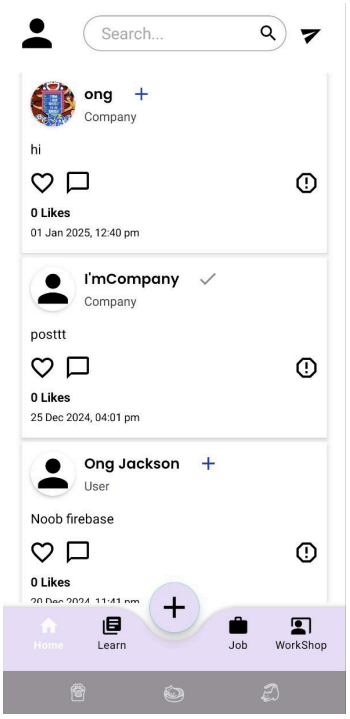


Figure 5.2.6.1.0

Search Function
Users can utilize the search function by clicking the search bar at the top of the page.

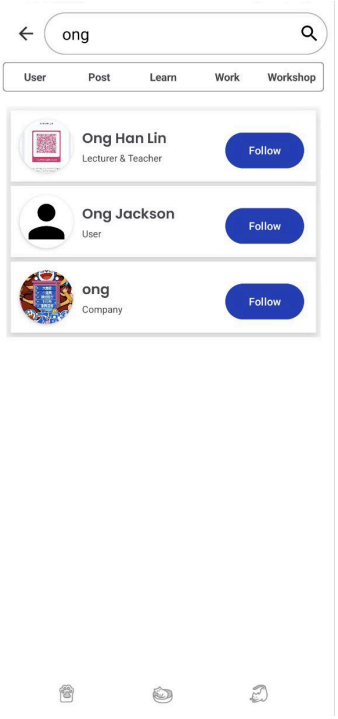


Figure 5.2.6.1.1

Search Function
Users can enter keywords in the search bar and click the search icon to initiate a search. They can also select a category for their search, including "User," "Post," "Learn," "Work," and "Workshop." The "User" category will display user names matching the keyword, along with their roles and a "Follow" button, allowing users to follow them (Figure 5.2.6.1.1). Users can also view a profile by clicking on the profile picture.

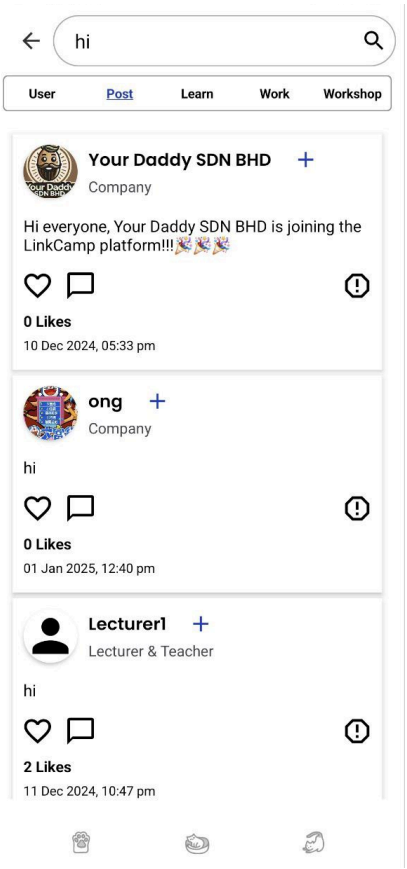


Figure 5.2.6.1.2

Search Function

The "Post" category will display posts that match the keyword, along with the author's name and role. Users can like a post by clicking the heart icon, comment on it by clicking the chat box icon, and follow the author by clicking the "＋" icon next to the author's name. Additionally, users can report a post by clicking the report icon, marked with an "!" symbol.

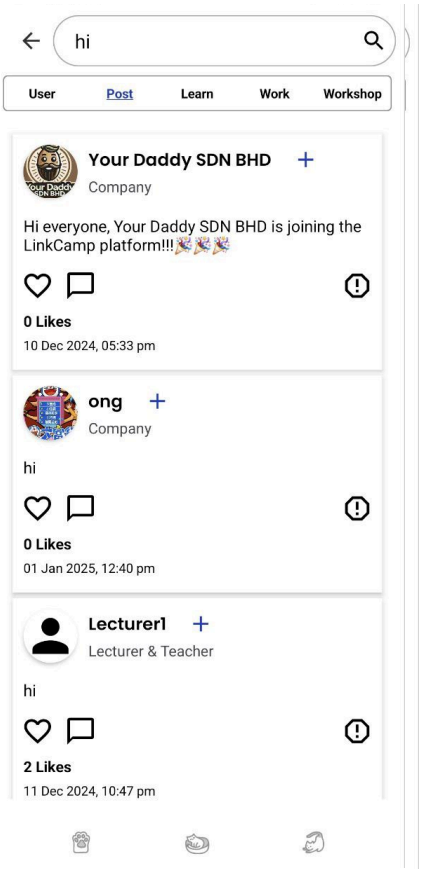


Figure 5.2.6.1.3

Search Function

The "Learn" category will display learning materials that match the keyword, along with the lecturer's name, the date and time the material was uploaded, the title of the material, and its description. Users can follow the lecturer by clicking the "＋" icon next to the lecturer's name. Additionally, users can view the learning material by clicking the "View" button and join the learning channel by clicking the "Channel" button.

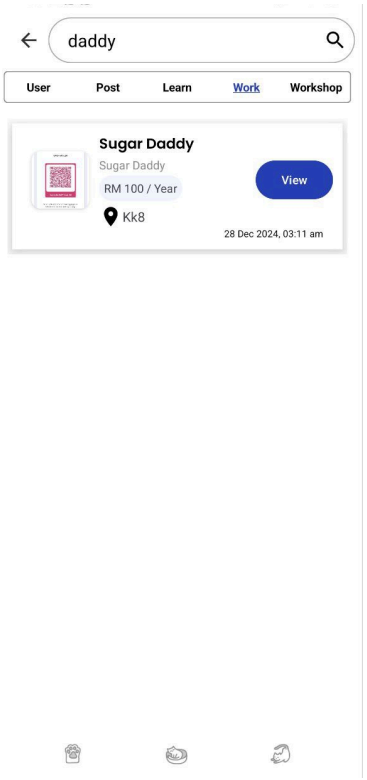


Figure 5.2.6.1.4

Search Function
The "Work" category will display job listings that match the keyword, including the job title, salary, location, and the date and time the job was posted. Users can view the job details by clicking the "View" button.

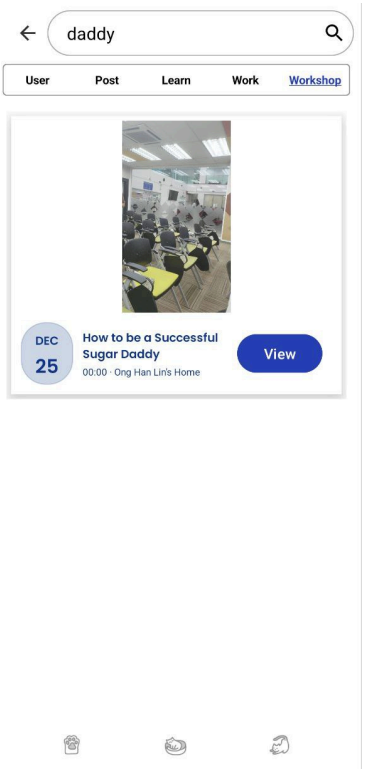


Figure 5.2.6.1.5

Search Function
The "Workshop" category will display workshops that match the keyword, along with the workshop title, location, and the date and time it is scheduled. Users can view the workshop details by clicking the "View" button.

5.2.7 Admin Pages

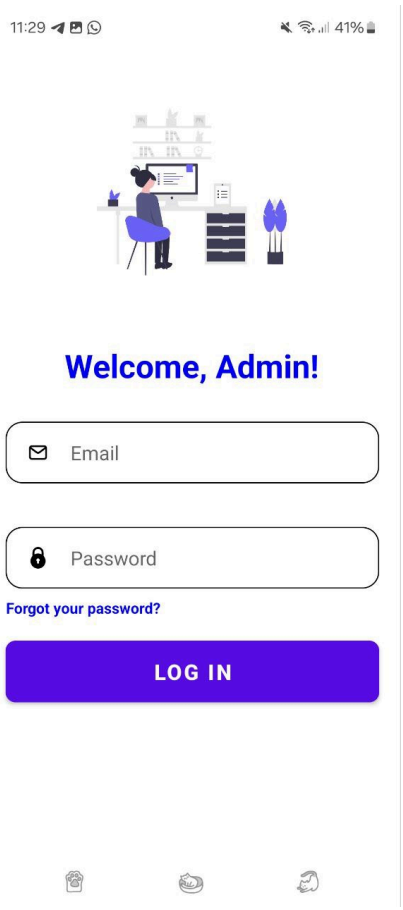


Figure 5.2.7.1

Login Activity

This page serves as the login interface for administrators. Admins can log in to their accounts by entering their email and password in the provided fields.

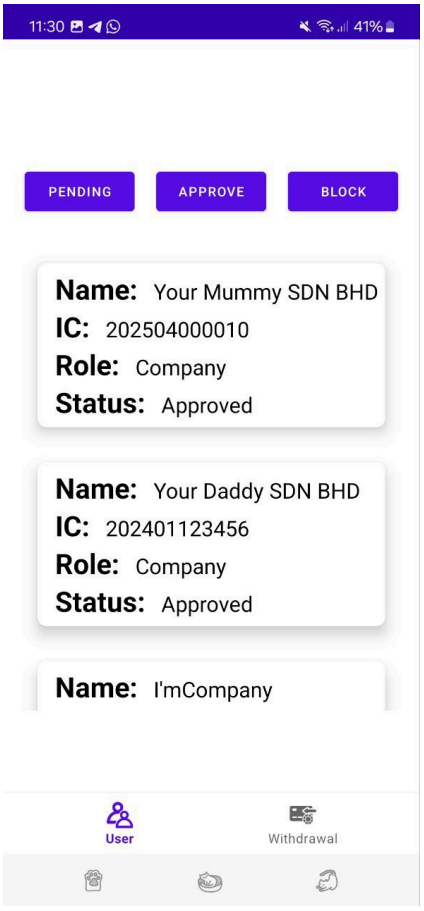


Figure 5.2.7.2

User Page

This page allows administrators to view all users registered on the app, categorized into pending users, approved users, and blocked users.

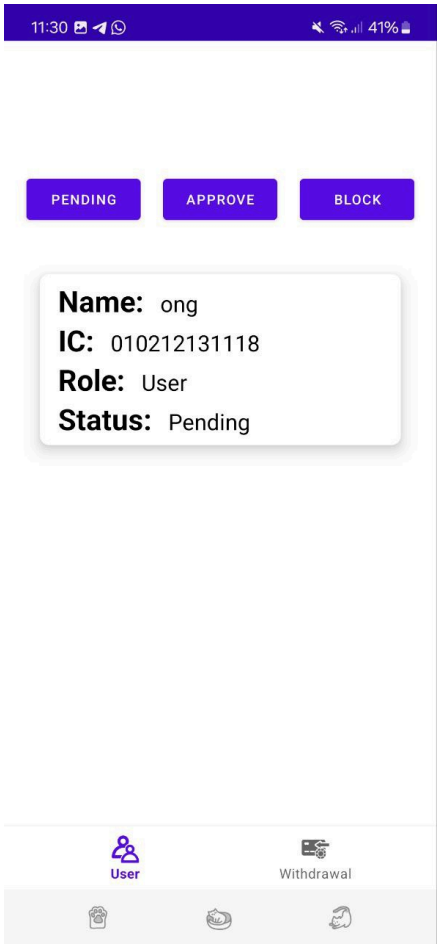


Figure 5.2.7.3

User Pending Page

This page displays users who have registered for the app but are still in the pending status. User information such as name, IC number, role, and current status is shown. Each user's information box is clickable, allowing administrators to navigate to another page for more detailed information about the user.

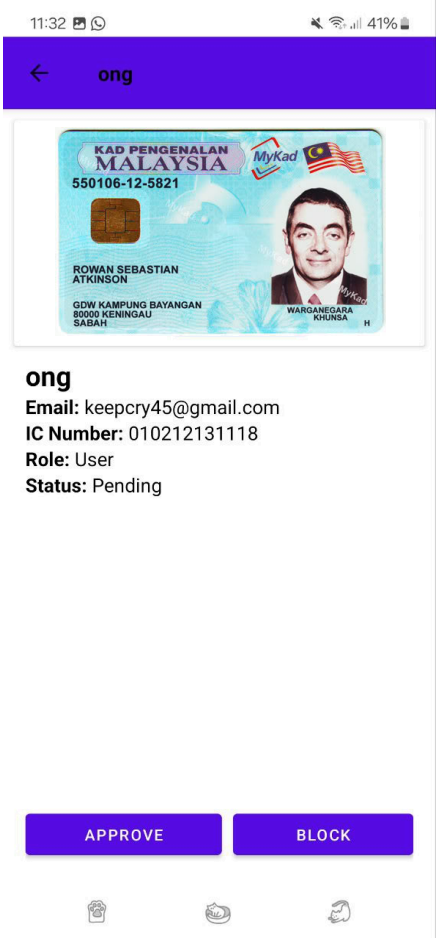


Figure 5.2.7.4

Pending User Page

After the user information box is clicked on the user pending page (Figure 5.2.7.4), the admin is directed to this page (Figure 5.2.7.5). On this page, the front image of the user's IC is displayed, and the back image can be viewed by swiping. Additionally, the user's email, IC number, role, and status are shown. The "Approve" button enables the admin to approve the user's registration, while the "Block" button allows the admin to block the pending user.

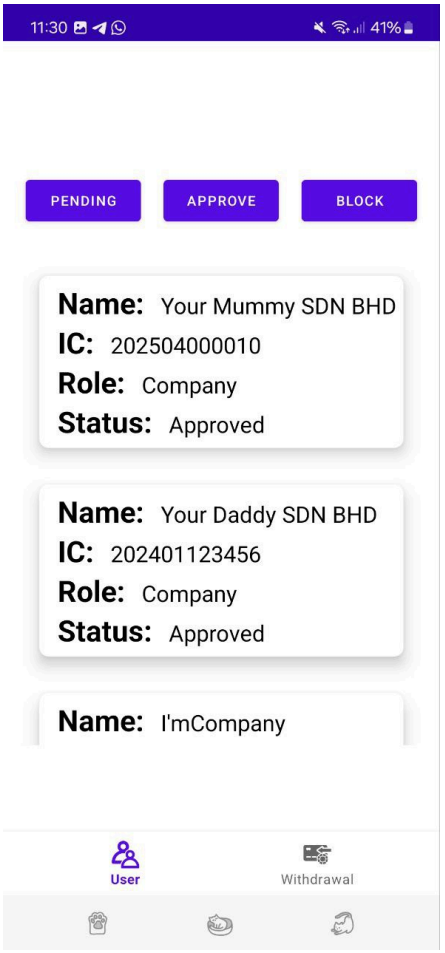


Figure 5.2.7.5

User Approve Page

This page displays users who have registered for the app and have an approved status. Information such as the user's name, IC number, role, and current status is provided. Each user's information box is interactive, enabling administrators to click and navigate to a detailed page for more comprehensive user information.

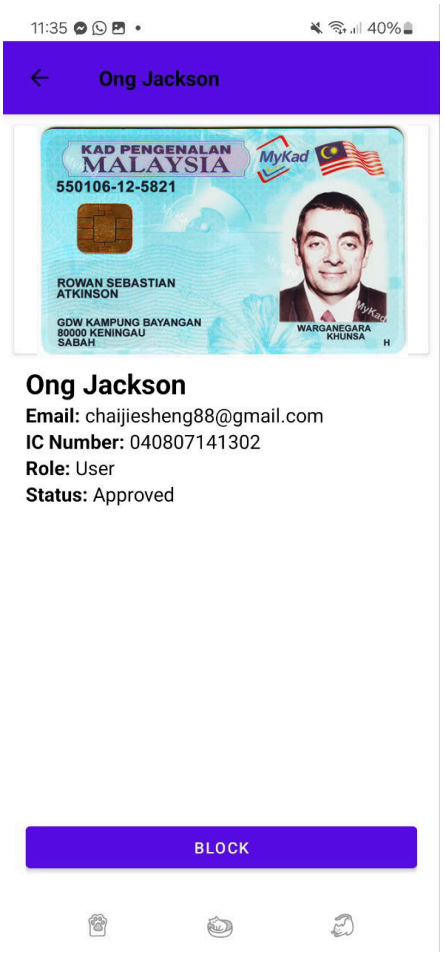


Figure 5.2.7.6

Approved User Page

After the user information box is clicked on the user pending page (Figure 5.2.7.5), the admin is directed to this page (Figure 5.2.7.6). On this page, the front image of the user's IC is displayed, and the back image can be viewed by swiping. Additionally, the user's email, IC number, role, and status are shown. The "Block" button allows the admin to block the user.

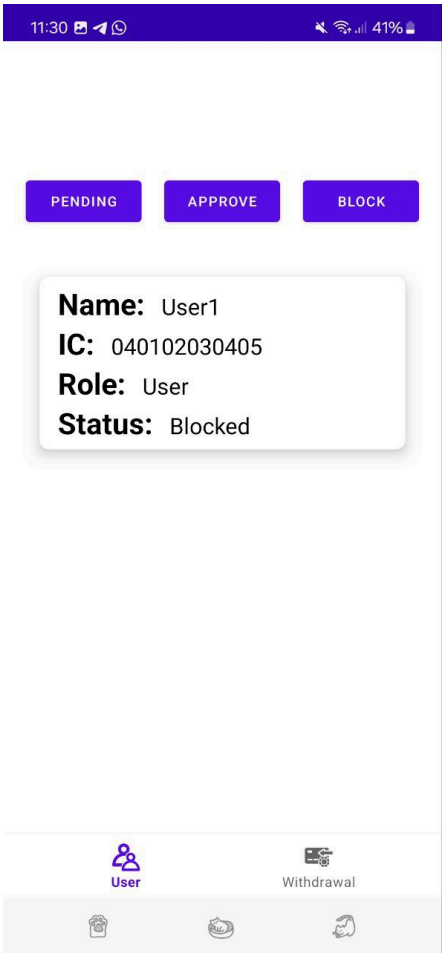


Figure 5.2.7.7

User Block Page

This page displays users who have been blocked. Information such as the user's name, IC number, role, and current status is provided. Each user's information box is interactive, enabling administrators to click and navigate to a detailed page for more comprehensive user information.

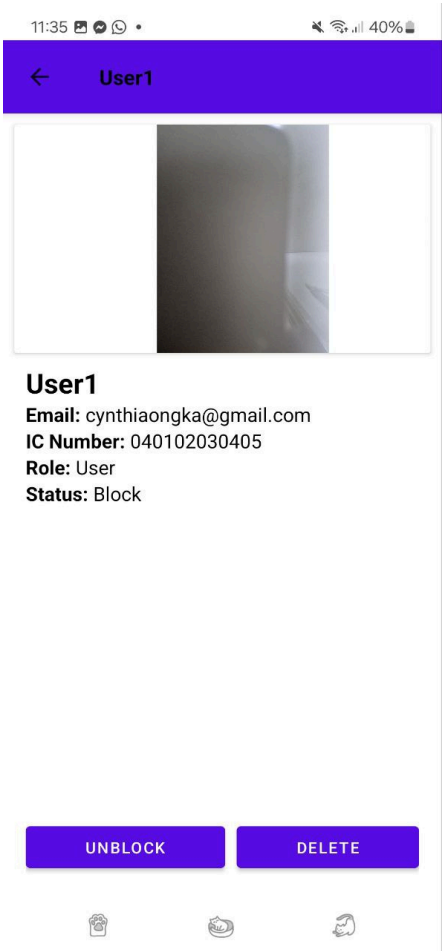


Figure 5.2.7.8

Blocked User Page

After the user information box is clicked on the user pending page (Figure 5.2.7.7), the admin is directed to this page (Figure 5.2.7.8). On this page, the front image of the user's IC is displayed, and the back image can be viewed by swiping. Additionally, the user's email, IC number, role, and status are shown. The "Unblock" button enables the admin to unblock the user, while the "Delete" button allows the admin to delete the user.

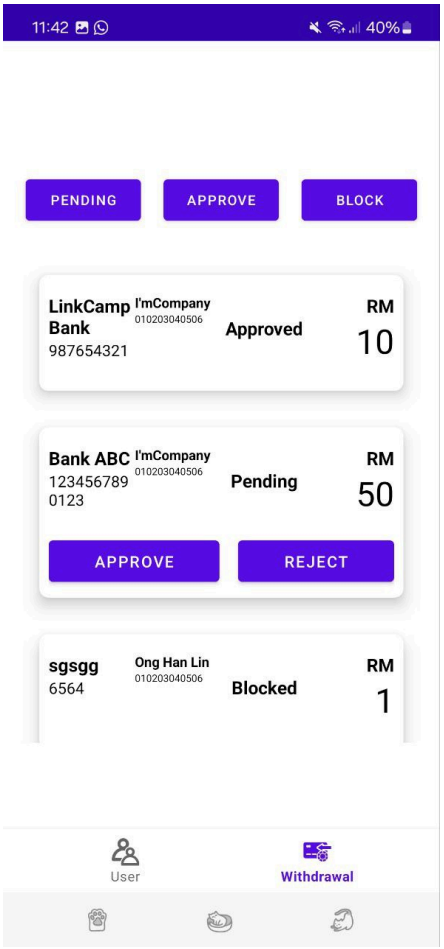


Figure 5.2.7.9

Withdrawal Page

This page displays withdrawal requests made by users. Information such as the user's name, bank account number, bank name, current status, and withdrawal amount is provided.

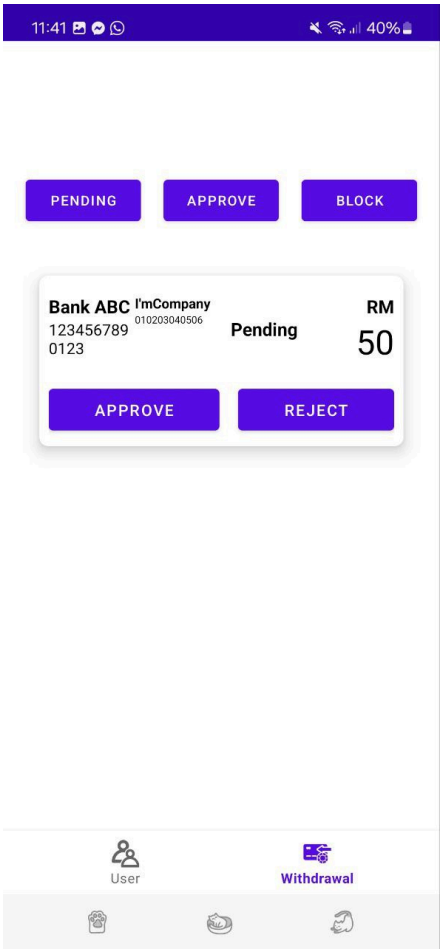


Figure 5.2.7.10

Withdrawal Pending Page

This page displays withdrawal requests made by users that are currently in pending status. Information such as the user's name, bank account number, bank name, current status, and withdrawal amount is provided. The “Approve” button enables the admin to approve the transaction, while the “Reject” button allows the admin to reject it.

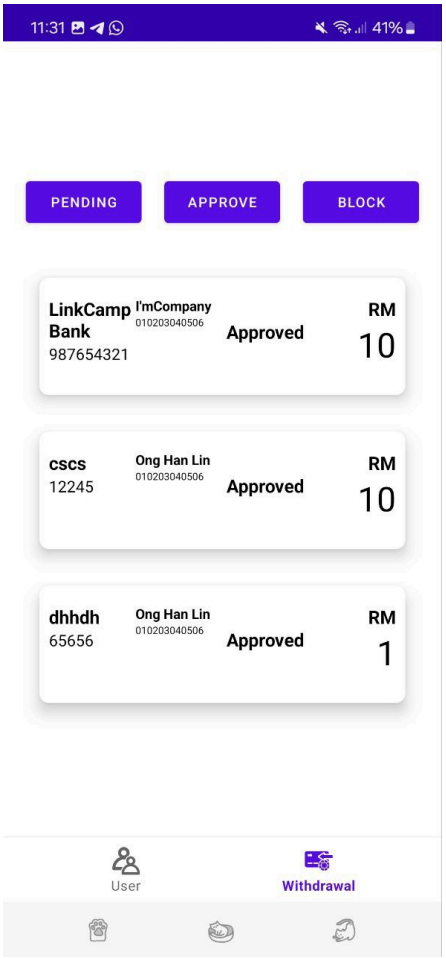


Figure 5.2.7.11

Withdrawal Approve Page

This page displays withdrawal requests made by users that are in approved status. Information such as the user's name, bank account number, bank name, current status, and withdrawal amount is provided.

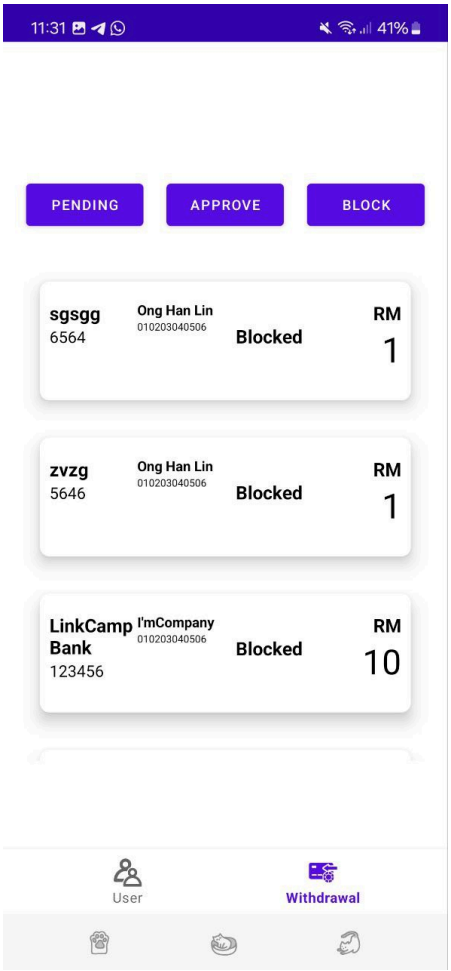


Figure 5.2.7.12

Withdrawal Block Page

This page displays withdrawal requests made by users that have been blocked. Information such as the user's name, bank account number, bank name, current status, and withdrawal amount is provided.

5.3 Messaging Module

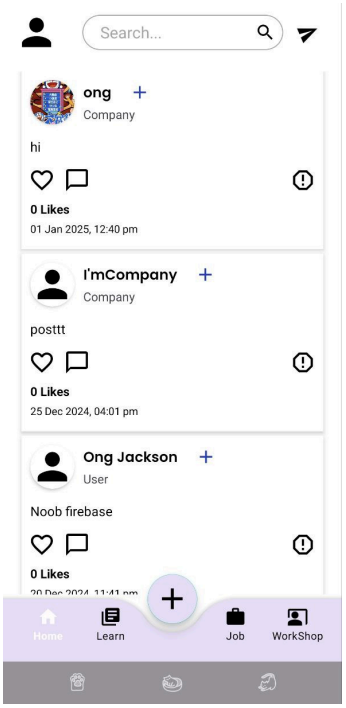


Figure 5.3.1.0

Send Message Activity

All users can send messages to other users through the Send Message activity, which can be accessed by clicking the arrow icon in the upper right corner of the app bar (Figure 5.3.1.0).

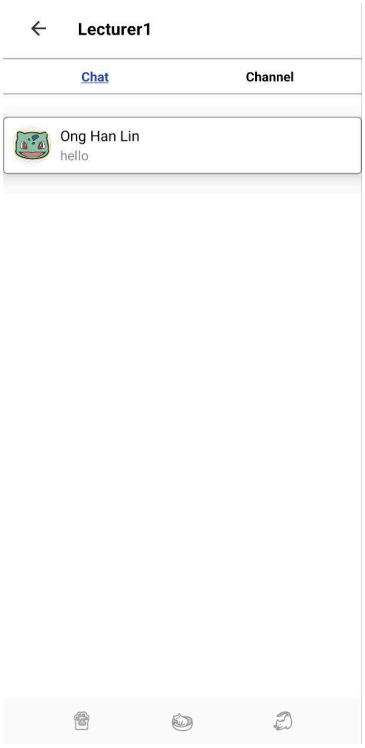


Figure 5.3.1.1

Send Message Activity

After clicking the arrow icon in the app bar (Figure 5.3.1.0), users will be directed to the Send Message interface (Figure 5.3.1.1). Here, users can view a list of individuals who have messaged them. By selecting a user's name, they can send a message to that person.

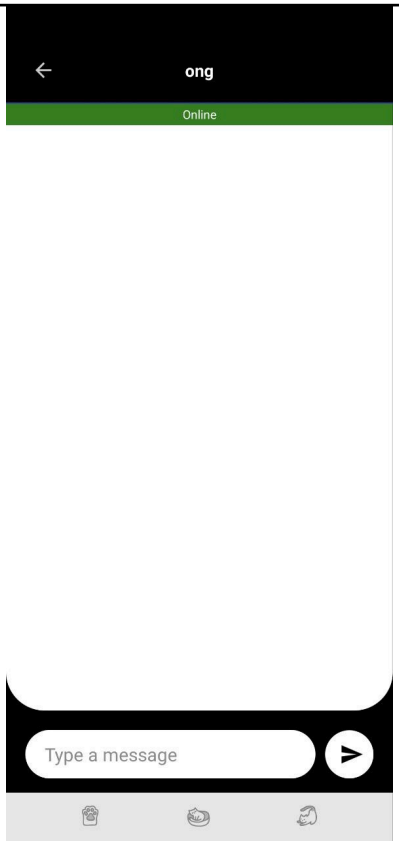


Figure 5.3.1.2

Send Message Activity

After selecting a user's name, they will be directed to a page where they can read and send messages to that person (Figure 5.3.1.2). Users can type their message in the text box and click the send arrow to deliver the message. A green indicator will appear next to the user's name in the chat, indicating that they are online.

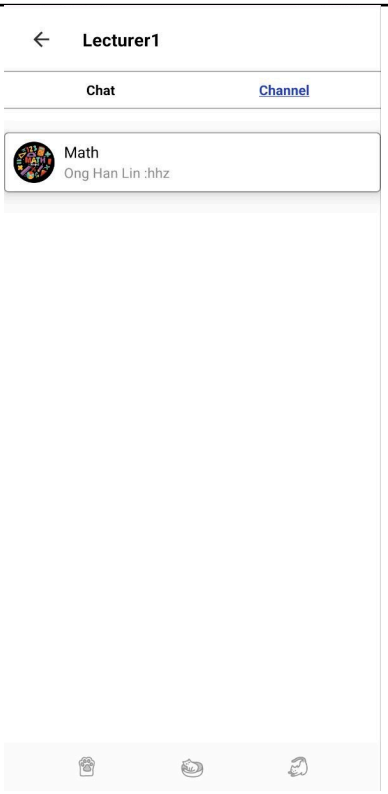


Figure 5.3.2.1

Send Message In Channel Activity

After clicking the arrow icon in the app bar (Figure 5.3.1.0), users will be directed to the Send Message interface (Figure 5.3.1.1). By selecting the "Channel" option in the upper right corner, users can view a list of channels they have joined. To send a message within a channel, users can click on the channel name (Figure 5.3.2.1).

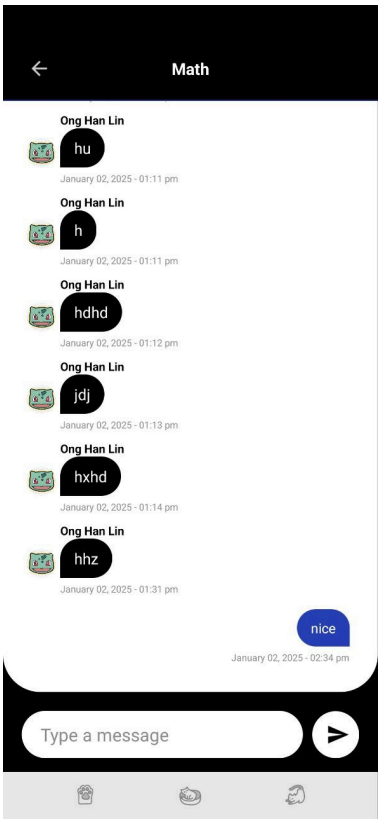


Figure 5.3.2.2

Send Message In Channel Activity

After selecting a user's name, they will be directed to a page where they can read and send messages in the channel (Figure 5.3.2.2). Users can type their message in the text box and click the send arrow to deliver the message.

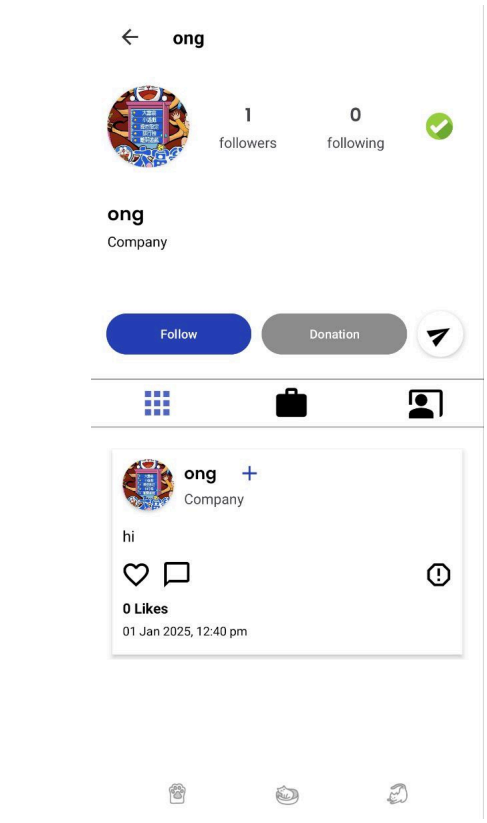


Figure 5.3.3

Send Message Activity By Profile

Users can send messages to others by clicking on their profile picture, which directs them to the user's profile (Figure 5.3.3). From there, users can click the arrow icon (next to the donation button) to initiate a message. They will then be directed to a page where they can read and send messages to that person.

5.4 Notification Module

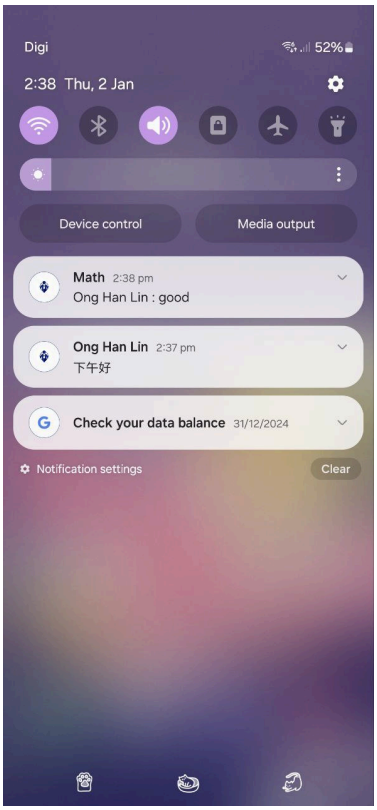


Figure 5.4.1.0

Notification Activity
Users will receive a notification when someone sends them a message. Notification permissions must be enabled for this feature to work

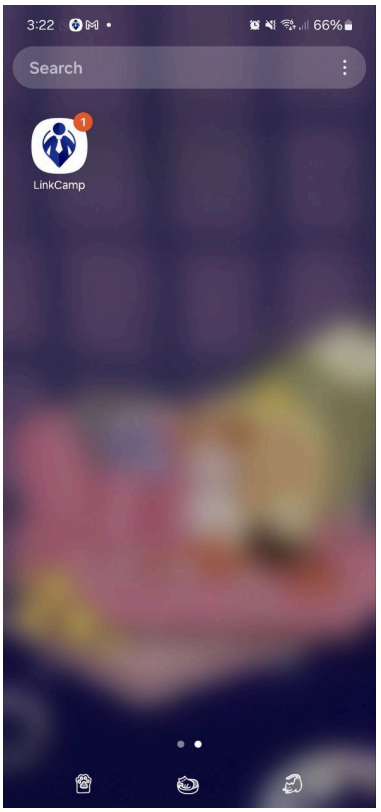


Figure 5.4.1.1

Notification Activity
The app icon on the phone will display a red badge with the number of notifications in the upper right corner whenever there is a new notification.

5.5 Payment Module

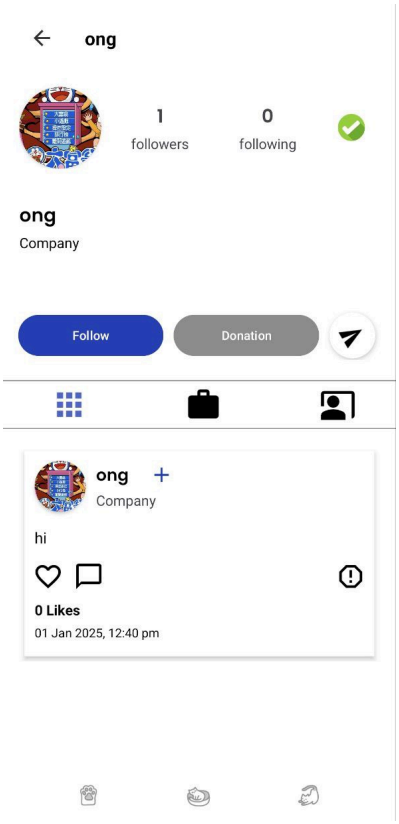


Figure 5.5.1.0

Donation Activity
Users can donate to other users by clicking their profile picture, which will direct them to the user's profile page (Figure 5.5.1.0). From there, users can click the "Donation" button to make a donation.

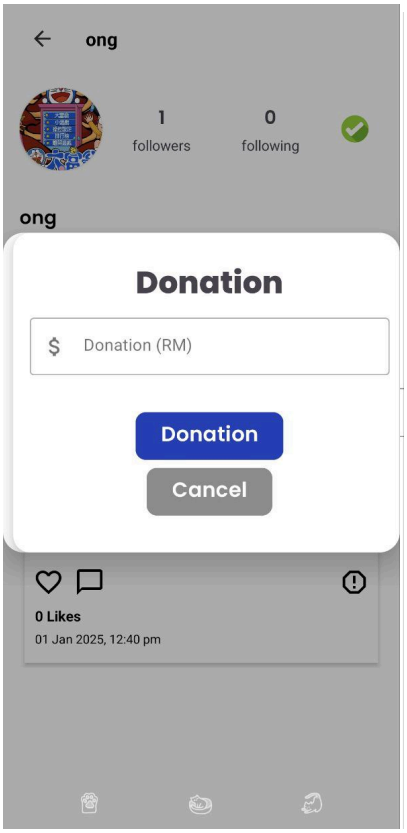


Figure 5.5.1.1

Donation Activity
Users can enter the amount they wish to donate and click the "Donation" button to proceed. If they wish to cancel, they can click the "Cancel" button (Figure 5.5.1.1).

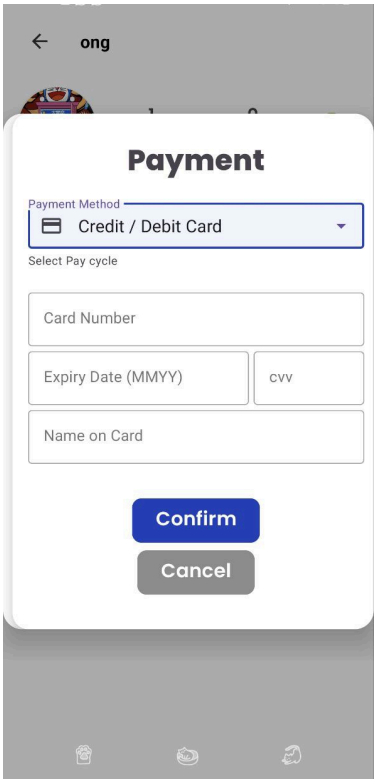


Figure 5.5.1.2

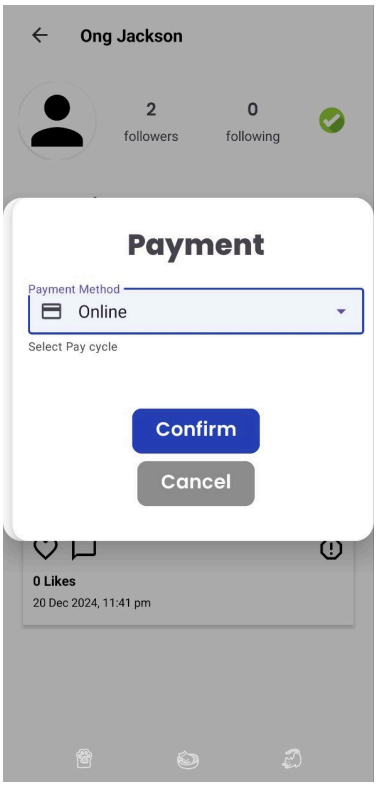


Figure 5.5.1.3

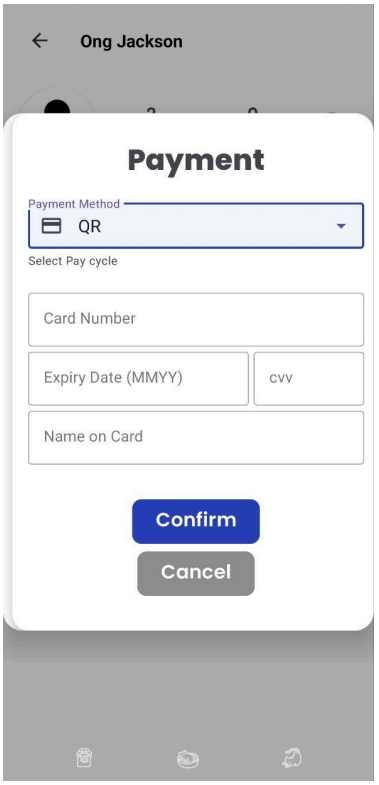


Figure 5.5.1.4

Donation Activity

Users will proceed to select a payment method, which includes "Credit/Debit Card," "Online," and "QR."

For "Credit/Debit Card" (Figure 5.5.1.2) and "QR" (Figure5.5.1.4), users will need to enter their card number, expiry date, CVV, and the name on the card.

For the "Online" option, no additional information is required.

Users can click the "Confirm" button to complete the donation process. If they wish to cancel, they can click the "Cancel" button.

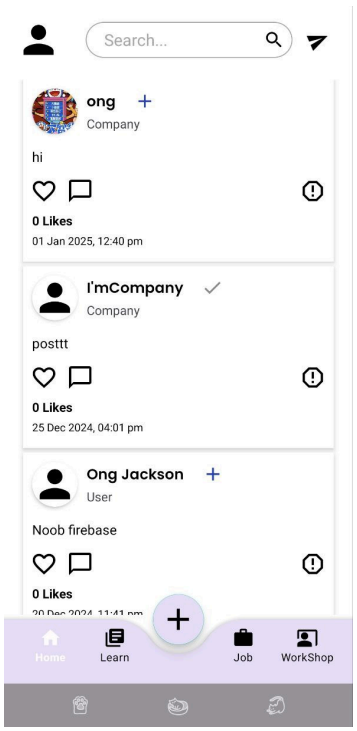


Figure 5.5.2.0

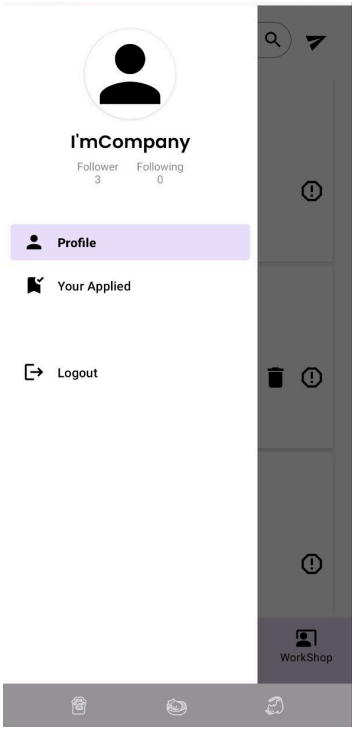


Figure 5.5.2.2

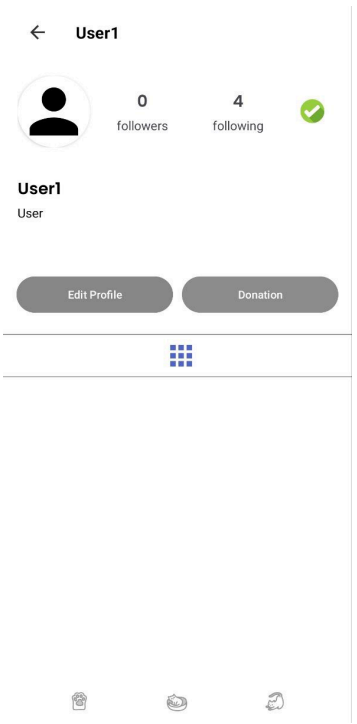


Figure 5.5.2.2

View Donation History Activity

Users can view donation activity by clicking the profile icon in the app bar (Figure 5.5.2.0), then selecting "Profile" from the sidebar menu (Figure 5.5.2.1). This will direct them to the profile page (Figure 5.5.2.2). By clicking the donation button, users will be taken to the donation page.

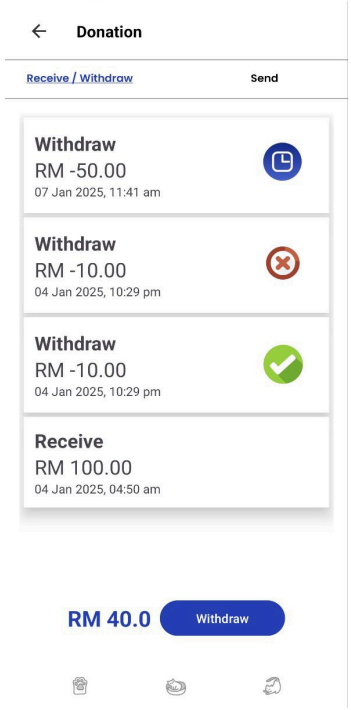


Figure 5.5.3.0

Donation History Activity

After clicking the donation button on the profile page (Figure 5.5.2.2), users will be directed to the donation page, where they can view their receive and withdrawal history (Figure 5.5.3.0). The blue icon beside the withdrawal information box indicates that the withdrawal is in pending status, the red icon indicates that the withdrawal has been blocked by the admin, and the green icon indicates that the withdrawal has been approved by the admin.

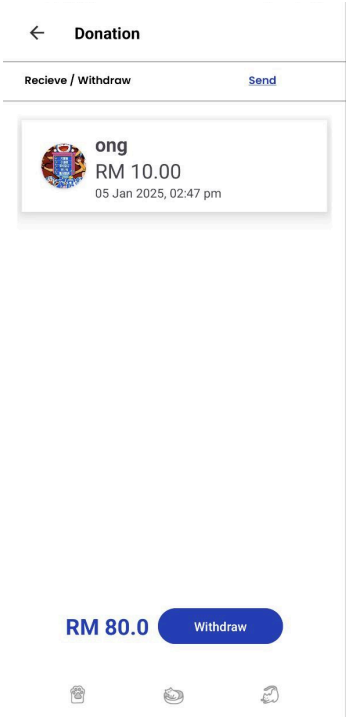


Figure 5.5.3.1

Donation History Activity

After clicking the donation button on the profile page (Figure 5.5.2.2), users will be directed to the donation page (Figure 5.5.3.0). From there, clicking the “Send” link will navigate users to the "Send" page, where they can view the history of money they have sent to other users (Figure 5.5.3.1).

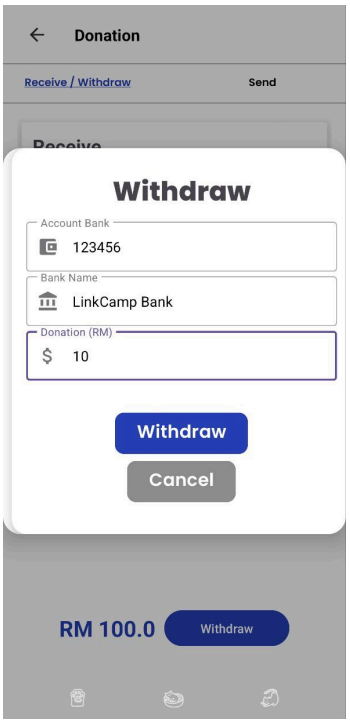


Figure 5.5.4

Withdraw Activity

Users can withdraw money by clicking the withdraw button on the donation page (Figure 5.5.3.0 or Figure 5.5.3.1). After entering the required details—"account bank," "bank name," and "donation (RM)"—users can click the withdraw button to initiate the process. If the donation amount exceeds the available balance, an error message will occur. Upon successful submission, a confirmation notice will appear. Withdrawals require admin approval, and the pending withdrawal status will be displayed on the donation page (Figure 5.5.3.0).

6.0 System Implementation

6.1 Development Environment

1) Android Manifest

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:tools="http://schemas.android.com/tools">

   <uses-permission android:name="android.permission.INTERNET" />
   <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
   <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
   <uses-permission android:name="android.permission.POST_NOTIFICATIONS" />

   <application
       android:allowBackup="true"
       android:dataExtractionRules="@xml/data_extraction_rules"
       android:fullBackupContent="@xml/backup_rules"
       android:icon="@mipmap/ic_launcher"
       android:label="@string/app_name"
       android:roundIcon="@mipmap/ic_launcher_round"
       android:supportsRtl="true"
       android:theme="@style/Theme.LinkCamp"
       tools:targetApi="31">
       <activity
           android:name=".Comment"
           android:windowSoftInputMode="adjustPan" />
       <activity
           android:name=".Search_"
           android:exported="false" />
       <activity
           android:name=".Change_email"
           android:exported="false" />
       <activity
           android:name=".Edit_profile"
           android:exported="false" />
       <activity
           android:name=".Channel"
           android:windowSoftInputMode="adjustPan" />
       <activity
           android:name=".Chat_Home"
           android:exported="false" />
       <activity
           android:name=".Chat"
           android:windowSoftInputMode="adjustPan" />
       <activity
           android:name=".View_Donation"
           android:exported="false" />
       <activity
           android:name=".View_Applied"
           android:exported="false" />
```

```
        <activity
            android:name=".View_Resume"
            android:exported="false" />
        <activity
            android:name=".View_Apply"
            android:exported="false" />
        <activity
            android:name=".Apply_Work"
            android:exported="false" />
        <activity
            android:name=".View_Work"
            android:exported="false" />
        <activity
            android:name=".create_work_salary"
            android:exported="false" />
        <activity
            android:name=".View_learn"
            android:exported="false" />
        <activity
            android:name=".Create_learning"
            android:exported="false" />
        <activity
            android:name=".Workshop_Applicant"
            android:exported="false" />
        <activity
            android:name=".Create_work"
            android:exported="false" />
        <activity
            android:name=".Profile_view"
            android:exported="false" />
        <activity
            android:name=".View_Workshop"
            android:exported="false" />
        <activity
            android:name=".Create_WorkShop_tutor"
            android:exported="false" />
        <activity
            android:name=".Create_Workshop_Cover"
            android:exported="false" />
        <activity
            android:name=".Create_Workshop"
            android:exported="false" />

        <meta-data

android:name="com.google.firebase.messaging.default_notification_channel_id"
            android:value="default" />
        <meta-data
            android:name="com.google.android.gms.version"
            android:value="@integer/google_play_services_version" />

        <activity
            android:name=".Posting"
            android:exported="false" />
        <activity
            android:name=".Home"
            android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
```

```
            </intent-filter>
        </activity>
        <activity
            android:name=".forgot_password"
            android:exported="false" />
        <activity
            android:name=".SSM_Certificates"
            android:exported="false"
            android:theme="@style/Theme.LinkCamp" />
        <activity
            android:name=".Upload_Certificates"
            android:exported="false" />
        <activity
            android:name=".term"
            android:exported="false" />
        <activity
            android:name=".Upload_profile"
            android:exported="false" />
        <activity
            android:name=".Upload_IC_Back"
            android:exported="false" />
        <activity
            android:name=".Upload_IC_Front"
            android:exported="false" />
        <activity
            android:name=".Create_password"
            android:exported="false" />
        <activity
            android:name=".OTPv"
            android:exported="false" />
        <activity
            android:name=".Create_account"
            android:exported="false" />
        <activity
            android:name=".login_page"
            android:exported="false" />
        <activity
            android:name=".MainActivity"
            android:exported="false" />

        <meta-data
            android:name="CLOUDINARY_URL"

android:value="cloudinary://184762337876237:SM6tkXj_NUCpZF5qCHL7dV-N8LI@dakrinvpf
" />

        <service
            android:name=".FCMNotificationService"
            android:exported="true">
            <intent-filter>
                <action android:name="com.google.firebase.MESSAGING_EVENT" />
            </intent-filter>
        </service>
    </application>

</manifest>
```

2) Convention

a) colors.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
   <color name="black">#FF000000</color>
   <color name="red">#FF0000</color>
   <color name="green1">#2AD06F</color>
   <color name="white">#FFFFFFFF</color>
   <color name="lavender">#8692f7</color>
   <color name="circle1">#F8F9FF</color>
   <color name="blue1">#1F41BB</color>
   <color name="blue2">#F1F4FF</color>
   <color name="blue3">#1D3BFF</color>
   <color name="blue4">#4A6DFF</color>
   <color name="grey">#8E8D8D</color>
   <color name="gray">#B8B8B8</color>
   <color name="grey1">#E8DEF8</color>
   <color name="less">#00FFFFFF</color>
   <color name="blue5">#3300388C</color>
   <color name="blue6">#00388C</color>
</resources>
```

b) dimens.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
   <dimen name="recycler_item_spacing">5dp</dimen>
</resources>
```

c) ic_launcher_background.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
   <color name="ic_launcher_background">#FFFFFF</color>
</resources>
```

d) strings.xml

```
<resources>
   <string name="app_name">LinkCamp</string>
         <string  name="getting">Getting  nearer  to\nYour  Dream
here</string>
   <string name="lss">Learn - Share - Strive </string>
   <string name="login">Login</string>
   <string name="register">Register</string>
   <string name="login_here">Login here</string>
   <string name="welcome">Welcome back you've\nbeen missed!</string>
   <string name="email">Email</string>
   <string name="password">Password</string>
   <string name="forgot_p">Forgot your password?</string>
   <string name="signin">Sign in</string>
   <string name="create_a">Create new account</string>
   <string name="create">Create Account</string>
    <string name="create_note">Create an account so you can explore
all the\ninspiring information</string>
   <string name="name">Name</string>
   <string name="next">Next</string>
   <string name="already">Already have an account</string>
   <string name="otp_verification">OTP Verification</string>
       <string  name="enter_the_otp_send_to">Enter  the  OTP  send
to</string>
   <string name="dint">Didn't you receive the OTP?</string>
   <string name="resend">Resend OTP</string>
   <string name="verify">Verify</string>
   <string name="role">Role</string>
   <string name="confirm_password">Confirm Password</string>
   <string name="ic_front">Upload Front of IC:</string>
   <string name="ic_back">Upload Back of IC:</string>
       <string  name="choose_image">Choose  a  image  or  take  a
photo</string>
   <string name="png">JPEG and PNG formats, up to 10MB</string>
            <string    name="identification_number">Identification
Number</string>
   <string name="example_front">Example Front of the IC</string>
   <string name="example_back">Example Back of the IC</string>
   <string name="upload_profile">Upload Profile Image</string>
```

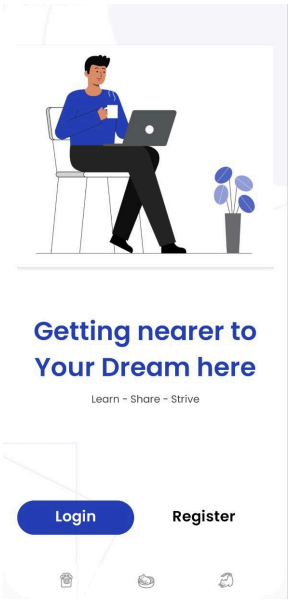
```
    <string name="upload_pro">Please Upload Your Profile in JPEG and
PNG formats</string>
   <string name="later">Upload Profile Later</string>
   <string name="term">Terms &amp;amp; Conditions</string>
   <string name="check_term">Accept Term and Condition </string>
   <string name="upload_c">Upload Certificates</string>
   <string name="choose_c">Choose a image or PDF</string>
     <string name="upload_ce">JPEG , PNG and PDF formats, up to
10MB</string>
          <string name="example_c">Click Example Format of
Certificates</string>
   <string name="single">Single Certificate</string>
   <string name="muti">Mutlipy Certificates</string>
   <string name="upload_ssm">Upload SSM Certificates</string>
       <string name="example_ssm">Click Example Format of SSM
Certificates</string>
   <string name="image">Image</string>
   <string name="pdf">PDF</string>
   <string name="reset_password">Reset Password</string>
     <string name="forgot_detail">Forgot your password? \nDon\'t
worry, reset your password here</string>
   <string name="back">Back</string>
   <string name="cancel">Cancel</string>
   <string name="conform">Confirm</string>
   <string name="post">Post</string>
   <string name="drawer_open">open\n</string>
   <string name="drawer_close">close</string>
   <string name="description">Description</string>
   <string name="description_w">Description Workshop</string>
   <string name="workshop">Create Workshop</string>
    <string name="workshop_desciption">Here, you can easily plan and
publish your workshop to share your knowledge and skills with a
wider audience.</string>
   <string name="view">View</string>
   <string name="overview">Session Overview:</string>
   <string name="tutor">Tutor</string>
   <string name="upload_pdf">PDF formats, up to 10MB</string>
   <!-- TODO: Remove or change this placeholder text -->
   <string name="hello_blank_fragment">Hello blank fragment</string>
   <string name="apply">Apply</string>
   <string name="delete">Delete</string>
```

```
    <string name="job_desription">Job Desription :</string>
        <string   name="key_responsibilities">Key   Responsibilities
:</string>
    <string name="requirements">Requirements :</string>
    <string name="application">Application</string>
    <string name="applied">Applied</string>
</resources>
```

6.2 Key Feature Implementation

6.2.1 User Authentication Module

Welcome Activity




```
package com.um.linkcamp;

import static function.function.hashPassword;
import static function.function.random;
import static function.function.sendmail;
import static function.function.verify_email;

import android.app.Dialog;
import android.content.Intent;
import android.content.pm.ActivityInfo;
import android.content.res.ColorStateList;
import android.graphics.Color;
import android.os.Bundle;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.TextView;

import androidx.activity.EdgeToEdge;
import androidx.appcompat.app.AppCompatActivity;

import com.google.android.material.textfield.TextInputEditText;
import com.google.android.material.textfield.TextInputLayout;
import com.google.firebase.firestore.FirebaseFirestore;
import com.google.firebase.firestore.QueryDocumentSnapshot;

import java.io.IOException;
import java.security.NoSuchAlgorithmException;
import java.util.Objects;

import data.gui;
import data.register;

public class login_page extends AppCompatActivity {
   private FirebaseFirestore db;
   @Override
   protected void onCreate(Bundle savedInstanceState) {
       super.onCreate(savedInstanceState);
       EdgeToEdge.enable(this);

       setContentView(R.layout.activity_login_page);
       final TextView BtnCreate = findViewById(R.id.BtnCreate);
       final TextView BtnForgot =findViewById(R.id.BtnForgot);
       BtnCreate.setOnClickListener(v -> {
                                                Intent intent = new
Intent(login_page.this,Create_account.class);
           startActivity(intent);
           finish();
       });
       BtnForgot.setOnClickListener(v -> {
                                                Intent intent = new
Intent(login_page.this,forgot_password.class);
           startActivity(intent);
```

```
            finish();
        });
        Button btnLogin = findViewById(R.id.button);
        btnLogin.setOnClickListener(v -> {
                                            TextInputEditText    txt_email    =
findViewById(R.id.text_input_email);
                                        TextInputEditText    txt_password    =
findViewById(R.id.text_input_password);
                                                        String     email     =
Objects.requireNonNull(txt_email.getText()).toString();
                                                         String     pass     =
Objects.requireNonNull(txt_password.getText()).toString();
            email = email.toLowerCase();
            boolean check = false;

                                                TextInputLayout    l_email    =
findViewById(R.id.text_input_layout_email);
                                               TextInputLayout    l_password    =
findViewById(R.id.text_input_layout_password);
            if(email.isEmpty() ){
                l_email.setHelperText("Please fill you email");

l_email.setHelperTextColor(ColorStateList.valueOf(Color.RED));
                check = true;
            }else{
                if(!verify_email(email)){
                    l_email.setHelperText("Invalid email");

l_email.setHelperTextColor(ColorStateList.valueOf(Color.RED));
                    check = true;
                }else{
                    l_email.setHelperText(null);
                }
            }
            if(pass.isEmpty()){
                l_password.setHelperText("Please fill you Password");

l_password.setHelperTextColor(ColorStateList.valueOf(Color.RED));
                check = true;
            }else {
                l_password.setHelperText(null);
            }

            if(check){
                return;
            }
            db = FirebaseFirestore.getInstance();
            String finalEmail = email;

            db.collection("Users").whereEqualTo("email",finalEmail).get()
                    .addOnSuccessListener(querySnapshot ->{
                        if(querySnapshot.isEmpty()){
                            Dialog dialog = new Dialog(login_page.this);
                            dialog.setContentView(R.layout.create_dialog);
                            Objects.requireNonNull(dialog.getWindow()).

setLayout(ViewGroup.LayoutParams.WRAP_CONTENT,ViewGroup.LayoutParams.WRAP
_CONTENT);

dialog.getWindow().setBackgroundDrawable(getDrawable(R.drawable.dialog));
                                                        TextView  title  =
```

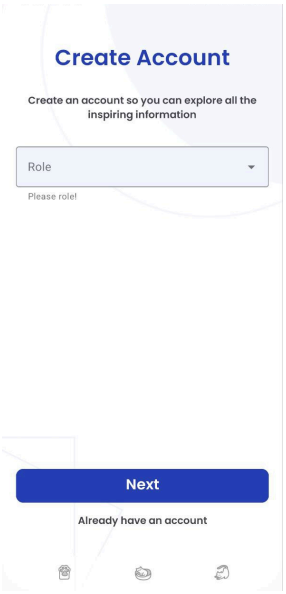
```
dialog.findViewById(R.id.tittle);
                                                        TextView  detail  =
dialog.findViewById(R.id.detail);
                            title.setText("Warning");
                            detail.setText("Email or Password Invalid");
                                                          Button  close  =
dialog.findViewById(R.id.confirm);
                            close.setText("Close");

close.setBackgroundTintList(ColorStateList.valueOf(Color.RED));
                            close.setOnClickListener(v1 -> {
                                dialog.dismiss();
                            });
                            dialog.show();
                        }else{
                            String salt = null,dPass = null;
                            int status = 0;
                                     for (QueryDocumentSnapshot document :
querySnapshot) {
                                    salt = document.getString("salt");
                                                                    dPass  =
document.getString("password");
                                                                   status  =
document.getLong("status").intValue();
                                String fPass;
                                try {
                                    fPass = hashPassword(pass, salt);
                                } catch (NoSuchAlgorithmException e) {
                                    throw new RuntimeException(e);
                                }
                                if (fPass.equals(dPass)) {

                                    // Handle pending status
                                    if (status == 1) {
                                                        Dialog dialog = new
Dialog(login_page.this);

dialog.setContentView(R.layout.create_dialog);

Objects.requireNonNull(dialog.getWindow()).

setLayout(ViewGroup.LayoutParams.WRAP_CONTENT,ViewGroup.LayoutParams.WRAP
_CONTENT);

dialog.getWindow().setBackgroundDrawable(getDrawable(R.drawable.dialog));
                                                          TextView title  =
dialog.findViewById(R.id.tittle);
                                                         TextView detail  =
dialog.findViewById(R.id.detail);
                                        title.setText("Pending");
                                             detail.setText("Please waiting
admin approve!");
                                                           Button close  =
dialog.findViewById(R.id.confirm);
                                        close.setText("Close");

close.setBackgroundTintList(ColorStateList.valueOf(Color.RED));
                                        close.setOnClickListener(v1 -> {
                                            dialog.dismiss();
                                        });
                                        dialog.show();
```

```
                                    return;
                                }

                                // Handle blocked status
                                if (status != 2) {
                                                Dialog dialog = new
Dialog(login_page.this);

dialog.setContentView(R.layout.create_dialog);

Objects.requireNonNull(dialog.getWindow()).

setLayout(ViewGroup.LayoutParams.WRAP_CONTENT,ViewGroup.LayoutParams.WRAP
_CONTENT);

dialog.getWindow().setBackgroundDrawable(getDrawable(R.drawable.dialog));
                                                TextView title =
dialog.findViewById(R.id.tittle);
                                                TextView detail =
dialog.findViewById(R.id.detail);
                                    title.setText("Block");
                                         detail.setText("You account is
block");
                                                Button close =
dialog.findViewById(R.id.confirm);
                                    close.setText("Close");

close.setBackgroundTintList(ColorStateList.valueOf(Color.RED));
                                    close.setOnClickListener(v1 -> {
                                        dialog.dismiss();
                                    });
                                    dialog.show();
                                    return;
                                }

                                // Generate OTP
                                                String otp1 =
String.valueOf(random());
                                                String otp2 =
String.valueOf(random());
                                String otpHash;
                                try {
                                          otpHash = hashPassword(otp1,
otp2);
                                } catch (NoSuchAlgorithmException e) {
                                    throw new RuntimeException(e);
                                }

                                // Send OTP email
                                gui gui = new gui();
                                try {
                                        sendmail(finalEmail, gui.otp_t,
gui.otp_content_login(otp1));
                                } catch (IOException e) {
                                    throw new RuntimeException(e);
                                }


                                     // Store user data in the register
singleton
                                                register register =
data.register.getInstance();
```

```
                                    register.clear();

register.setName(document.getString("name"));
                                    register.setEmail(finalEmail);
                                    register.setPassword(fPass);

register.setIc(document.getString("ic"));
                                                                String role =
document.getString("role");
                                    register.setRole(role);

register.setProfile(document.getString("profile"));

                                    if ("Company".equals(role)) {

register.setCertificates(document.getString("certificates"));
                                    } else {

register.setIc_front(document.getString("front"));

register.setIc_back(document.getString("back"));
                                        if (!"User".equals(role)) {

register.setCertificates(document.getString("certificates"));
                                        }
                                    }

                                    register.setPage(3);
                                    register.setOtp(3);
                                    register.setSalt(otp2);
                                    register.setHashOTp(otpHash);

                                    // Navigate to OTP page
                                                            Intent intent = new
Intent(login_page.this, OTPv.class);
                                    startActivity(intent);
                                    finish();
                                    return;
                                } else {
                                                            Dialog dialog = new
Dialog(login_page.this);

dialog.setContentView(R.layout.create_dialog);

Objects.requireNonNull(dialog.getWindow()).

setLayout(ViewGroup.LayoutParams.WRAP_CONTENT,ViewGroup.LayoutParams.WRAP
_CONTENT);

dialog.getWindow().setBackgroundDrawable(getDrawable(R.drawable.dialog));
                                                              TextView title =
dialog.findViewById(R.id.tittle);
                                                            TextView detail =
dialog.findViewById(R.id.detail);
                                    title.setText("Warning");
                                         detail.setText("Email or Password
Invalid");
                                                               Button close =
dialog.findViewById(R.id.confirm);
                                    close.setText("Close");
```

```
close.setBackgroundTintList(ColorStateList.valueOf(Color.RED));
                                    close.setOnClickListener(v1 -> {
                                        dialog.dismiss();
                                    });
                                    dialog.show();
                                }
                            }
                        }

                    })
                    .addOnFailureListener(e -> {
                        Dialog dialog = new Dialog(login_page.this);
                        dialog.setContentView(R.layout.create_dialog);
                        Objects.requireNonNull(dialog.getWindow()).

setLayout(ViewGroup.LayoutParams.WRAP_CONTENT,ViewGroup.LayoutParams.WRAP
_CONTENT);

dialog.getWindow().setBackgroundDrawable(getDrawable(R.drawable.dialog));
                        TextView title = dialog.findViewById(R.id.tittle);
                                                    TextView  detail  =
dialog.findViewById(R.id.detail);
                        title.setText("Error");
                             detail.setText("Failed to connect. Please try
again later.");
                        Button close = dialog.findViewById(R.id.confirm);
                        close.setText("Close");

close.setBackgroundTintList(ColorStateList.valueOf(Color.RED));
                        close.setOnClickListener(v1 -> {
                            dialog.dismiss();
                        });
                        dialog.show();
                    });
        });
    }
}
```

Create Account Activity




```
package com.um.linkcamp;

import static function.function.check_ic_format;
import static function.function.check_ssm_format;
import static function.function.hashPassword;
import static function.function.random;
import static function.function.sendmail;
import static function.function.verify_email;

import android.content.Intent;
import android.content.res.ColorStateList;
import android.graphics.Color;
import android.os.Bundle;
import android.view.View;
import android.widget.ArrayAdapter;
import android.widget.AutoCompleteTextView;
import android.widget.Button;
import android.widget.TextView;

import androidx.activity.EdgeToEdge;
import androidx.appcompat.app.AppCompatActivity;
import androidx.constraintlayout.widget.ConstraintLayout;

import com.google.android.material.snackbar.Snackbar;
import com.google.android.material.textfield.TextInputEditText;
import com.google.android.material.textfield.TextInputLayout;

import com.google.firebase.firestore.FirebaseFirestore;
import com.google.firebase.firestore.QueryDocumentSnapshot;


import java.io.IOException;
import java.security.NoSuchAlgorithmException;
import java.util.ArrayList;
import java.util.List;
import java.util.Objects;


public class Create_account extends AppCompatActivity {

   private static final List<String> Role = new ArrayList<>();
   String role = null;
   TextInputLayout name_l,email_l,ic_l;

   private FirebaseFirestore db;
   @Override
   protected void onCreate(Bundle savedInstanceState) {
       super.onCreate(savedInstanceState);
       EdgeToEdge.enable(this);
       setContentView(R.layout.activity_create_account);

       db = FirebaseFirestore.getInstance();

       final TextView BtnAlready = findViewById(R.id.already);
       final Button BtnNext = findViewById(R.id.next);
       final ConstraintLayout constraintLayout = findViewById(R.id.main);

       name_l = findViewById(R.id.text_input_layout_name);
```

```
        email_l = findViewById(R.id.text_input_layout_email);
        ic_l = findViewById(R.id.text_input_layout_ic);

        name_l.setVisibility(View.GONE);
        email_l.setVisibility(View.GONE);
        ic_l.setVisibility(View.GONE);

        db.collection("Role")
                .get()
                .addOnSuccessListener(queryDocumentSnapshots -> {
                    Role.clear();
                                    for (QueryDocumentSnapshot document :
queryDocumentSnapshots) {
                        String role = document.getString("role");
                        if (role != null) {
                            Role.add(role);
                        }
                    }
                });
        data.register register = data.register.getInstance();
        register.clear();

        ArrayAdapter<String> adapter = new ArrayAdapter<>(this,
                android.R.layout.simple_dropdown_item_1line, Role);
        AutoCompleteTextView textView = findViewById(R.id.text_input_role);
        textView.setAdapter(adapter);

        textView.setOnItemClickListener((parent, view, position, id) -> {
            role = adapter.getItem(position);
            assert role != null;
            if(role.equals("Company")){
                ic_l.setHint("Registration Number");
                  ic_l.setHelperText("Please Enter Your Company Registration
Number");
                name_l.setHint("Company Name");
                 name_l.setHelperText("Please Enter Your Company Name follow
the Registration!");
            }else{
                ic_l.setHint("Identification Number");
                ic_l.setHelperText("Identification Number Without '-'");
                name_l.setHint("Name");
                        name_l.setHelperText("Please Enter Your Name Follow
Identification Card!");
            }
            name_l.setVisibility(View.VISIBLE);
            email_l.setVisibility(View.VISIBLE);
            ic_l.setVisibility(View.VISIBLE);
        });

        BtnAlready.setOnClickListener(v -> {
            register.clear();
                                                Intent    intent    =    new
Intent(Create_account.this,login_page.class);
            startActivity(intent);
            finish();
        });

        BtnNext.setOnClickListener(v -> {
                                                TextInputLayout    txtrole    =
findViewById(R.id.text_input_layout_role);
            if(role == null){
```

```
txtrole.setHelperTextColor(ColorStateList.valueOf(Color.RED));
                return;
            }
            txtrole.setHelperTextColor(ColorStateList.valueOf(Color.BLACK));
            TextInputEditText txt_name = findViewById(R.id.text_input_name);
                                                TextInputEditText    txt_email    =
findViewById(R.id.text_input_email);
            TextInputEditText txt_ic = findViewById(R.id.text_input_ic);

                                                                String     name     =
Objects.requireNonNull(txt_name.getText()).toString();
                                                               String     email     =
Objects.requireNonNull(txt_email.getText()).toString();
            String ic = Objects.requireNonNull(txt_ic.getText()).toString();
            email = email.toLowerCase();

            if(!valid(ic,name,email,ic_l,name_l,email_l)){
               return;
            }

            String finalEmail = email;
            checkIfEmailExists(email, emailExists -> {

                if (emailExists) {
                      email_l.setHelperText("Email already exist!\nExample :
abc@abc.com");

email_l.setHelperTextColor(ColorStateList.valueOf(Color.RED));
                }else{
                    email_l.setHelperText(null);
                }
                checkIfICExists(ic,icExists ->{
                    if (icExists) {
                        if(role.equals("Company")){
                             ic_l.setHelperText("Registration Number already
exist");

ic_l.setHelperTextColor(ColorStateList.valueOf(Color.RED));
                        }else{
                                    ic_l.setHelperText("Identification Number
already exist");

ic_l.setHelperTextColor(ColorStateList.valueOf(Color.RED));
                        }
                    }else{
                        ic_l.setHelperText(null);
                    }
                    if(!emailExists && !icExists){
                        try {
                            int number = random();
                            int number_2 = random();
                            data.gui gui = new data.gui();
                                                                  String  password  =
hashPassword(String.valueOf(number),String.valueOf(number_2));
                                                        sendmail(finalEmail,gui.otp_t,
gui.otp_content(String.valueOf(number)));
                                        Snackbar.make(constraintLayout,"OTP send
successfully!",Snackbar.LENGTH_SHORT).show();
```

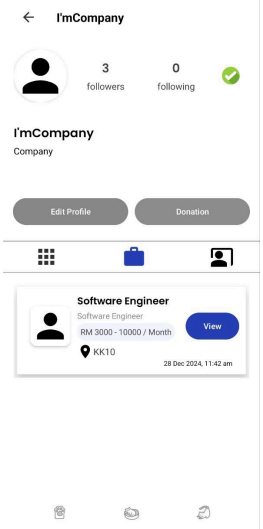
```
                                register.setIc(ic);
                                register.setEmail(finalEmail);
                                register.setName(name);
                                register.setRole(role);
                                register.setPassword(password);
                                register.setSalt(String.valueOf(number_2));
                                register.setOtp(3);
                                register.setPage(1);

                                  Intent intent =new Intent(Create_account.this,
OTPv.class);
                                startActivity(intent);
                                finish();
                            } catch (IOException | NoSuchAlgorithmException e) {
                                  Snackbar.make(constraintLayout,"Failed to send
OTP!",Snackbar.LENGTH_SHORT).show();
                                throw new RuntimeException(e);
                            }
                        }
                    });
                });
            });
    }

        private boolean valid(String ic,String name, String email,
TextInputLayout ly_ic,TextInputLayout ly_name, TextInputLayout ly_email) {
        int number = 0;
        if(ic.isEmpty()){
            if(role.equals("Company")){
                ly_ic.setHelperText("Please fill in Registration Number!");
            }else{
                    ly_ic.setHelperText("Please fill in Identification Number
Without '-' !");
            }
            ly_ic.setHelperTextColor(ColorStateList.valueOf(Color.RED));
            number++;
        } else if (!role.equals("Company")) {
            if(!check_ic_format(ic)){
                ly_ic.setHelperText("Invalid Identification Number!");
                ly_ic.setHelperTextColor(ColorStateList.valueOf(Color.RED));
                number++;
            }else{
                ly_ic.setHelperText(null);
            }
        }else if(role.equals("Company")){
            if(!check_ssm_format(ic)){
                ly_ic.setHelperText("Invalid Registration Number!");
                ly_ic.setHelperTextColor(ColorStateList.valueOf(Color.RED));
                number++;
            }else{
                ly_ic.setHelperText(null);
            }
        } else{
            ly_ic.setHelperText(null);
        }
        if(name.isEmpty()){
            if(role.equals("Company")){
                  ly_name.setHelperText("Please fill in company name follow
the SSM!");
            }else{
                  ly_name.setHelperText("Please fill in your name follow you
```

```
identification card!");
            }

            ly_name.setHelperTextColor(ColorStateList.valueOf(Color.RED));
            number ++;
        }else{
            ly_name.setHelperText(null);
        }
        if(email.isEmpty()){
              ly_email.setHelperText("Please fill in your email!\nExample :
abc@abc.com");
            ly_email.setHelperTextColor(ColorStateList.valueOf(Color.RED));
            number ++;
        } else if (!verify_email(email)) {
            ly_email.setHelperText("Invalid email!\nExample : abc@abc.com");
            ly_email.setHelperTextColor(ColorStateList.valueOf(Color.RED));
            number ++;
        } else{

            ly_email.setHelperText(null);
        }
        return number <= 0;
    }

     private void checkIfEmailExists(String emailToCheck, FirebaseCallback
callback) {
        db.collection("Users")
                .whereEqualTo("email", emailToCheck) // Example condition
                .get()
                .addOnSuccessListener(querySnapshot -> {
                    if (querySnapshot.isEmpty()) {
                        try {
                            callback.onCallback(false);
                        } catch (IOException e) {
                            throw new RuntimeException(e);
                        } catch (NoSuchAlgorithmException e) {
                            throw new RuntimeException(e);
                        }
                    } else {
                        try {
                            callback.onCallback(true);
                        } catch (IOException e) {
                            throw new RuntimeException(e);
                        } catch (NoSuchAlgorithmException e) {
                            throw new RuntimeException(e);
                        }
                    }
                })
                .addOnFailureListener(e -> {
                    try {
                        callback.onCallback(true);
                    } catch (IOException ev) {
                        throw new RuntimeException(ev);
                    } catch (NoSuchAlgorithmException ev) {
                        throw new RuntimeException(ev);
                    }
                });
    }
    private void checkIfICExists(String i, FirebaseCallback callback) {
        db.collection("Users")
                .whereEqualTo("ic", i) // Example condition
```

```
                .get()
                .addOnSuccessListener(querySnapshot -> {
                    if (querySnapshot.isEmpty()) {
                        try {
                            callback.onCallback(false);
                        } catch (IOException e) {
                            throw new RuntimeException(e);
                        } catch (NoSuchAlgorithmException e) {
                            throw new RuntimeException(e);
                        }
                    } else {
                        try {
                            callback.onCallback(true);
                        } catch (IOException e) {
                            throw new RuntimeException(e);
                        } catch (NoSuchAlgorithmException e) {
                            throw new RuntimeException(e);
                        }
                    }
                })
                .addOnFailureListener(e -> {
                    try {
                        callback.onCallback(true);
                    } catch (IOException ev) {
                        throw new RuntimeException(ev);
                    } catch (NoSuchAlgorithmException ev) {
                        throw new RuntimeException(ev);
                    }
                });
    }
}
```

6.2.2 User Profile Module

Profile Page




```
package com.um.linkcamp;

import static android.graphics.Color.BLACK;
import static android.graphics.Color.GRAY;
import static android.graphics.Color.RED;

import static function.function.sendmail;

import android.app.Dialog;
import android.content.Intent;
import android.content.res.ColorStateList;
import android.database.Cursor;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Color;
import android.os.Bundle;
import android.text.InputFilter;
import android.util.Base64;
import android.util.Log;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.AutoCompleteTextView;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.LinearLayout;
import android.widget.TextView;

import androidx.activity.EdgeToEdge;
import androidx.appcompat.app.AppCompatActivity;
import androidx.cardview.widget.CardView;
import androidx.constraintlayout.widget.ConstraintSet;
import androidx.core.content.ContextCompat;
import androidx.core.graphics.Insets;
import androidx.core.view.ViewCompat;
import androidx.core.view.WindowInsetsCompat;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import
com.google.android.material.textfield.TextInputEditText;
import com.google.android.material.textfield.TextInputLayout;
import com.google.firebase.Timestamp;
import com.google.firebase.firestore.CollectionReference;
import com.google.firebase.firestore.DocumentChange;
import com.google.firebase.firestore.DocumentReference;
import com.google.firebase.firestore.FieldValue;
import com.google.firebase.firestore.FirebaseFirestore;
import com.google.firebase.firestore.QueryDocumentSnapshot;
import com.google.firebase.firestore.SetOptions;


import org.w3c.dom.Text;

import java.io.IOException;
import java.util.ArrayList;
```

```
import java.util.Date;
import java.util.HashMap;
import java.util.List;
import java.util.Objects;
import java.util.concurrent.atomic.AtomicReference;

import Adapter.LearnAdapter;
import Adapter.PostAdapter;
import Adapter.WorkAdapter;
import Adapter.WorkshopAdapter;
import data.DatabaseHelper;
import function.DecimalDigitsInputFilter;
import function.SpacingItemDecoration;
import function.VerifyLogin;
import model.Learn;
import model.Post;
import model.Work;
import model.Workshop;

public class Profile_view extends AppCompatActivity {
   private DatabaseHelper dbHelper;
   String pay;
     String ic = null,userID = null,name=null,email_user =
null,email_user_ = null;
   RecyclerView item_view,learn_view,work_view,workshop_view;
   ArrayList<Post> postList;
   ArrayList<Workshop> workshopList;
   ArrayList<Learn> learnList;
   ArrayList<Work> workList;
   PostAdapter postAdapter;
   WorkshopAdapter workshopAdapter;
   LearnAdapter learnAdapter;
   WorkAdapter workAdapter;
   @Override
   protected void onCreate(Bundle savedInstanceState) {
       super.onCreate(savedInstanceState);
       EdgeToEdge.enable(this);
       setContentView(R.layout.activity_profile_view);

ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.m
ain), (v, insets) -> {
                                        Insets    systemBars    =
insets.getInsets(WindowInsetsCompat.Type.systemBars());
               v.setPadding(systemBars.left, systemBars.top,
systemBars.right, systemBars.bottom);
           return insets;
       });
       dbHelper = new DatabaseHelper(Profile_view.this);
                          VerifyLogin    verifyLogin    =    new
VerifyLogin(Profile_view.this);
       if (verifyLogin.isDatabaseExist()) {
           verifyLogin.verify(result -> {
               if ("other".equals(result)) {
                   dbHelper.clearUserData();
                  finish();
               }else if (!("login".equals(result))) {
                   dbHelper.clearUserData();
                   finish();
               }
           });
       }else{
```

```
            finish();
            return;
        }
        Cursor cursor = dbHelper.getUserData();
        if (cursor.moveToFirst()) {
                                                        ic     =
cursor.getString(cursor.getColumnIndex("ic"));
                                                    email_user     =
cursor.getString(cursor.getColumnIndex("email"));
        }
                        AtomicReference<Intent>   intent   =   new
AtomicReference<>(getIntent());
        userID = intent.get().getStringExtra("UserID");
        if(userID.isEmpty()){
            finish();
        }else if (userID == null){
            finish();
        }


        Button btn1 = findViewById(R.id.btn1);
        Button btn2 = findViewById(R.id.btn2);


        TextView txt_username = findViewById(R.id.username);
        TextView txt_name = findViewById(R.id.txt_name);
        TextView txt_follower = findViewById(R.id.follower);
        TextView txt_following = findViewById(R.id.following);
        TextView txt_role = findViewById(R.id.txt_role);
                                TextView    txt_description    =
findViewById(R.id.description);

        ImageView post_item = findViewById(R.id.post_item);
        ImageView learn_item = findViewById(R.id.learn_item);
        ImageView work_item = findViewById(R.id.work_item);
                                 ImageView    workshop_item    =
findViewById(R.id.workshop_item);
                                    ImageView     imgProfile     =
findViewById(R.id.profile_image);
        ImageView btnBack = findViewById(R.id.back);

                LinearLayoutManager  linearLayoutManager  =new
LinearLayoutManager(Profile_view.this);
        linearLayoutManager.setReverseLayout(true);
        linearLayoutManager.setStackFromEnd(true);

                                     int     spacingInPixels     =
getResources().getDimensionPixelSize(R.dimen.recycler_item_sp
acing);

                LinearLayoutManager  itemLayoutManager  =  new
LinearLayoutManager(Profile_view.this);
        itemLayoutManager.setReverseLayout(true);
        itemLayoutManager.setStackFromEnd(true);


        item_view = findViewById(R.id.item_view);
        learn_view = findViewById(R.id.learn_view);
        work_view = findViewById(R.id.work_view);
        workshop_view = findViewById(R.id.workshop_view);
```

```
                                setupRecyclerView(item_view,    new
LinearLayoutManager(this), spacingInPixels);
                               setupRecyclerView(learn_view,    new
LinearLayoutManager(this), spacingInPixels);
                                setupRecyclerView(work_view,    new
LinearLayoutManager(this), spacingInPixels);
                           setupRecyclerView(workshop_view,    new
LinearLayoutManager(this), spacingInPixels);

        item_view.setVisibility(View.VISIBLE);

        postList = new ArrayList<>();
                                        postAdapter     =     new
PostAdapter(Profile_view.this,postList);

        item_view.setAdapter(postAdapter);

        workshopList = new ArrayList<>();
                                     workshopAdapter     =     new
WorkshopAdapter(Profile_view.this,workshopList);

        workshop_view.setAdapter(workshopAdapter);

        learnList = new ArrayList<>();
                                       learnAdapter     =     new
LearnAdapter(Profile_view.this,learnList,ic);

        learn_view.setAdapter(learnAdapter);

        workList = new ArrayList<>();
                                       workAdapter     =     new
WorkAdapter(Profile_view.this,workList);

        work_view.setAdapter(workAdapter);

        UpdatePost();
        CardView cardView =findViewById(R.id.send_message);

        post_item.setOnClickListener(v -> {
            showRecyclerView(item_view,post_item);

            UpdatePost();
        });
        workshop_item.setOnClickListener(v -> {
            showRecyclerView(workshop_view,workshop_item);

            UpdateWorkshop();
        });
        learn_item.setOnClickListener(v -> {
            showRecyclerView(learn_view,learn_item);
            UpdateLearn();
        });
        work_item.setOnClickListener(v -> {
            showRecyclerView(work_view,work_item);
            UpdateWork();
        });


        if(check(userID,ic)){
            btn1.setText("Edit Profile");
            btn1.setBackgroundColor(GRAY);
```

```
            btn1.setOnClickListener(v -> {
                                        Intent  intent1  =  new
Intent(Profile_view.this,Edit_profile.class);
                startActivity(intent1);
                finish();
            });
            btn2.setOnClickListener(v -> {
                                        Intent  intent1  =  new
Intent(Profile_view.this,View_Donation.class);
                startActivity(intent1);
            });
            cardView.setVisibility(View.GONE);
        }else{
            isFollow(userID,ic,btn1);
            btn1.setOnClickListener(v -> {
                if((btn1.getTag()).equals("Follow")){
                        HashMap<String, Object> hashMap = new
HashMap<>();
                    hashMap.put(userID, true);

                    FirebaseFirestore.getInstance()
                            .collection("Follow")
                            .document(ic)
                            .collection("following")
                            .document(userID)
                            .set(hashMap, SetOptions.merge());

                       HashMap<String, Object> f_hashMap = new
HashMap<>();
                    f_hashMap.put(ic, true);

                    FirebaseFirestore.getInstance()
                            .collection("Follow")
                            .document(userID)
                            .collection("follower")
                            .document(ic)
                                                .set(f_hashMap,
SetOptions.merge());
                    btn1.setText("Following");

btn1.setBackgroundColor(ContextCompat.getColor(Profile_view.t
his, R.color.grey));
                    btn1.setTag("Following");
                }else{
                    FirebaseFirestore.getInstance()
                            .collection("Follow")
                            .document(userID)
                            .collection("follower")
                            .document(ic)
                            .delete();

                    FirebaseFirestore.getInstance()
                            .collection("Follow")
                            .document(ic)
                            .collection("following")
                            .document(userID)
                            .delete();
                    btn1.setText("Follow");

btn1.setBackgroundColor(ContextCompat.getColor(Profile_view.t
his, R.color.blue1));
```

```
                    btn1.setTag("Follow");
                }
            });
            btn2.setOnClickListener(v -> {
                dialog_donation();
            });
        }
        cardView.setOnClickListener(v -> {
                                    Intent   intent1   =   new
Intent(Profile_view.this,Chat.class);
            intent1.putExtra("userId",userID);
            intent1.putExtra("name",name);
            startActivity(intent1);
        });
        ImageView send = findViewById(R.id.send);
        send.setOnClickListener(v -> {
                                    Intent   intent1   =   new
Intent(Profile_view.this,Chat.class);
            intent1.putExtra("name",name);
            intent1.putExtra("userId",userID);
            startActivity(intent1);
        });

                                  FirebaseFirestore     db     =
FirebaseFirestore.getInstance();
        db.collection("Users").whereEqualTo("ic",userID).get()
                .addOnSuccessListener(querySnapshot ->{
                    if(querySnapshot.isEmpty()){
                        finish();
                    }else{
                         for (QueryDocumentSnapshot document :
querySnapshot) {
                            name = document.getString("name");
                                                email_user_  =
document.getString("email");
                                               String  role  =
document.getString("role");
                                           String description =
document.getString("description");
                                               String  profile  =
document.getString("profile");
                                                int  status  =
Objects.requireNonNull(document.getLong("status")).intValue()
;
                                         ImageView  img_status  =
findViewById(R.id.status);
                            if(status == 1){

img_status.setImageResource(R.drawable.pending);
                            } else if (status == 2) {

img_status.setImageResource(R.drawable.acept);
                            }else{

img_status.setImageResource(R.drawable.block);
                            }

followN(txt_following,txt_follower);
                            txt_username.setText(name);
                            txt_name.setText(name);
                            txt_role.setText(role);
```

```
                            if(description == null){
                                txt_description.setText(null);
                                                    }else
if(description.equals("null")){
                                txt_description.setText(null);
                            } else{

txt_description.setText(description);
                            }
                            if(!"skip".equals(profile)){
                                try {
                                            byte[] imageBytes =
Base64.decode(profile, Base64.DEFAULT);
                                                Bitmap bitmap =
BitmapFactory.decodeByteArray(imageBytes,                    0,
imageBytes.length);

imgProfile.setImageBitmap(bitmap);
                                } catch (Exception e) {
                                    e.printStackTrace();
                                }
                            }
                            if("Company".equals(role)){

workshop_item.setVisibility(View.VISIBLE);

work_item.setVisibility(View.VISIBLE);
                                        } else if ("Lecturer &
Teacher".equals(role)) {

workshop_item.setVisibility(View.VISIBLE);

learn_item.setVisibility(View.VISIBLE);
                            }
                        }
                    }
                });

        btnBack.setOnClickListener(v -> {
            finish();
        });
    }
    private void showDialog(String title, String message) {
        Dialog dialog = new Dialog(Profile_view.this);
        dialog.setContentView(R.layout.create_dialog);

Objects.requireNonNull(dialog.getWindow()).setLayout(ViewGrou
p.LayoutParams.WRAP_CONTENT,
ViewGroup.LayoutParams.WRAP_CONTENT);

dialog.getWindow().setBackgroundDrawable(getDrawable(R.drawab
le.dialog));


        TextView t = dialog.findViewById(R.id.tittle);
        TextView d = dialog.findViewById(R.id.detail);
        Button btnC = dialog.findViewById(R.id.confirm);

        t.setText(title);
        d.setText(message);
```

```
        btnC.setText("Close");
        btnC.setOnClickListener(v -> {
            dialog.dismiss();
        });

        dialog.show();
    }
     private void setupRecyclerView(RecyclerView recyclerView,
LinearLayoutManager layoutManager, int spacingInPixels) {
        recyclerView.setHasFixedSize(true);
        recyclerView.setLayoutManager(layoutManager);
                              recyclerView.addItemDecoration(new
SpacingItemDecoration(spacingInPixels));
        recyclerView.setVisibility(View.GONE);
    }
                private    void    showRecyclerView(RecyclerView
recyclerView,ImageView imageView) {
        item_view.setVisibility(View.GONE);
        learn_view.setVisibility(View.GONE);
        work_view.setVisibility(View.GONE);
        workshop_view.setVisibility(View.GONE);

        ImageView post_item = findViewById(R.id.post_item);
        ImageView learn_item = findViewById(R.id.learn_item);
        ImageView work_item = findViewById(R.id.work_item);
                                ImageView    workshop_item    =
findViewById(R.id.workshop_item);


post_item.setColorFilter(ContextCompat.getColor(Profile_view.
this, R.color.black));

learn_item.setColorFilter(ContextCompat.getColor(Profile_view
.this, R.color.black));

work_item.setColorFilter(ContextCompat.getColor(Profile_view.
this, R.color.black));

workshop_item.setColorFilter(ContextCompat.getColor(Profile_v
iew.this, R.color.black));



imageView.setColorFilter(ContextCompat.getColor(Profile_view.
this, R.color.blue1));

        recyclerView.setVisibility(View.VISIBLE);
    }
    private void dialog_donation(){
        Dialog dialog = new Dialog(Profile_view.this);
        dialog.setContentView(R.layout.donation_dialog);

Objects.requireNonNull(dialog.getWindow()).setLayout(ViewGrou
p.LayoutParams.WRAP_CONTENT,
ViewGroup.LayoutParams.WRAP_CONTENT);

dialog.getWindow().setBackgroundDrawable(getDrawable(R.drawab
le.dialog));

        dialog.setCancelable(false);
```

```
        Button btnC = dialog.findViewById(R.id.confirm);
        Button btnCancel = dialog.findViewById(R.id.cancel);
                                    TextInputEditText    amount    =
dialog.findViewById(R.id.content);
                        amount.setFilters(new   InputFilter[]{new
DecimalDigitsInputFilter(2)});
        btnC.setOnClickListener(v -> {
                                                 String    amount_    =
Objects.requireNonNull(amount.getText()).toString().trim();
                                   TextInputLayout   l_amount   =
dialog.findViewById(R.id.amount);
            if (amount_ == null){

l_amount.setHelperTextColor(ColorStateList.valueOf(RED));
                l_amount.setHelperText("Please fill in Amount
(RM)");
            } else if (amount_.isEmpty()) {

l_amount.setHelperTextColor(ColorStateList.valueOf(RED));
                l_amount.setHelperText("Please fill in Amount
(RM)");
            }else{

l_amount.setHelperTextColor(ColorStateList.valueOf(BLACK));
                l_amount.setHelperText("");
                dialog.dismiss();
                payment(amount_);

            }
        });
        btnCancel.setOnClickListener(v -> {
            dialog.dismiss();
        });

        dialog.show();
    }
    private void payment(String amount_){
        Dialog dialog = new Dialog(Profile_view.this);
        dialog.setContentView(R.layout.dialog_payment);

Objects.requireNonNull(dialog.getWindow()).setLayout(ViewGrou
p.LayoutParams.WRAP_CONTENT,
ViewGroup.LayoutParams.WRAP_CONTENT);

dialog.getWindow().setBackgroundDrawable(getDrawable(R.drawab
le.dialog));
        dialog.setCancelable(false);

        Button confirm = dialog.findViewById(R.id.confirm);

                                TextInputEditText    txt_card    =
dialog.findViewById(R.id.card_number);
                                TextInputEditText    txt_date    =
dialog.findViewById(R.id.date);
                                 TextInputEditText    txt_cvv    =
dialog.findViewById(R.id.cvv);
                                TextInputEditText    txt_name    =
dialog.findViewById(R.id.name);


                                    FirebaseFirestore     db     =
```

```
FirebaseFirestore.getInstance();

                AutoCompleteTextView  autoCompleteTextView  =
dialog.findViewById(R.id.auto_complete_pay_cycle);
        List<String> Pay = new ArrayList<>();

        db.collection("Payment")
                .get()
                  .addOnSuccessListener(queryDocumentSnapshots
-> {
                    Pay.clear();
                         for (QueryDocumentSnapshot document :
queryDocumentSnapshots) {
                                                  String  type  =
document.getString("Payment");
                        if (type != null) {
                            Pay.add(type);
                        }
                    }
                });
                      ArrayAdapter<String>   adapter   =   new
ArrayAdapter<>(this,
                    android.R.layout.simple_dropdown_item_1line,
Pay);
        autoCompleteTextView.setAdapter(adapter);

                 autoCompleteTextView.setOnClickListener(v  ->
autoCompleteTextView.showDropDown());

          autoCompleteTextView.setOnItemClickListener((parent,
view, position, id) ->{
                                          LinearLayout   card   =
dialog.findViewById(R.id.card);
            pay = adapter.getItem(position);
            if("Online".equals(pay)){
                card.setVisibility(View.GONE);
            }else{
                card.setVisibility(View.VISIBLE);
            }
        });

        confirm.setOnClickListener(v -> {

            if("Online".equals(pay)){
                submit(amount_);
                dialog.dismiss();
            }else {
                                                String   number   =
Objects.requireNonNull(txt_card.getText()).toString().trim();
                                                  String   date   =
Objects.requireNonNull(txt_date.getText()).toString().trim();
                                                   String   cvv   =
Objects.requireNonNull(txt_cvv.getText()).toString().trim();
                                                  String   name   =
Objects.requireNonNull(txt_name.getText()).toString().trim();

                                       TextInputLayout  l_number  =
dialog.findViewById(R.id.l_card_number);
                                         TextInputLayout  l_date  =
dialog.findViewById(R.id.l_date);
                                          TextInputLayout  l_cvv  =
```

```
dialog.findViewById(R.id.l_cvv);
                                        TextInputLayout  l_name  =
dialog.findViewById(R.id.l_name);


                boolean check = true;

                if(number == null){
                    check = false;

l_number.setHelperTextColor(ColorStateList.valueOf(Color.RED)
);
                        l_number.setHelperText("Please fill in
Card number!");
                } else if (number.isEmpty()) {
                    check = false;

l_number.setHelperTextColor(ColorStateList.valueOf(Color.RED)
);
                        l_number.setHelperText("Please fill in
Card number!");
                }else{
                                if(number.length()  >=  13  &&
number.length() <= 19){

l_number.setHelperTextColor(ColorStateList.valueOf(Color.BLAC
K));
                        l_number.setHelperText(" ");
                    }else{
                        check = false;

l_number.setHelperTextColor(ColorStateList.valueOf(Color.RED)
);
                            l_number.setHelperText("Card Number
between 13 digit and 19 digit!");
                    }
                }

                if(date == null){
                    check = false;

l_date.setHelperTextColor(ColorStateList.valueOf(Color.RED));
                        l_date.setHelperText("Please fill in
Expiry Date (MMYY)!");
                } else if (date.isEmpty()) {
                    check = false;

l_date.setHelperTextColor(ColorStateList.valueOf(Color.RED));
                        l_date.setHelperText("Please fill in
Expiry Date (MMYY)!");
                }else{
                    if(date.length() != 4){
                        check = false;

l_date.setHelperTextColor(ColorStateList.valueOf(Color.RED));
                            l_date.setHelperText("Expiry Date
format error!");
                    }else{

l_date.setHelperTextColor(ColorStateList.valueOf(Color.BLACK)
);
```

```
                        l_date.setHelperText(" ");
                    }
                }

                if(cvv == null){
                    check = false;

l_cvv.setHelperTextColor(ColorStateList.valueOf(Color.RED));
                        l_cvv.setHelperText("Please fill in
CVV!");
                } else if (cvv.isEmpty()) {
                    check = false;

l_cvv.setHelperTextColor(ColorStateList.valueOf(Color.RED));
                        l_cvv.setHelperText("Please fill in
CVV!");
                }else{
                    if(cvv.length() != 3){
                        check = false;

l_cvv.setHelperTextColor(ColorStateList.valueOf(Color.RED));
                         l_cvv.setHelperText("CVV have 3 digit
number!");
                    }else{

l_cvv.setHelperTextColor(ColorStateList.valueOf(Color.BLACK))
;
                        l_cvv.setHelperText(" ");
                    }
                }

                if(name == null){
                    check = false;

l_name.setHelperTextColor(ColorStateList.valueOf(Color.RED));
                     l_name.setHelperText("Please fill Name on
Card");
                } else if (name.isEmpty()) {
                    check = false;

l_name.setHelperTextColor(ColorStateList.valueOf(Color.RED));
                     l_name.setHelperText("Please fill Name on
Card");
                }else{

l_name.setHelperTextColor(ColorStateList.valueOf(Color.BLACK)
);
                    l_name.setHelperText(" ");
                }

                if(check){
                    submit(amount_);
                }

            }
        });

        Button cancel = dialog.findViewById(R.id.cancel);
        cancel.setOnClickListener(v -> {
            dialog.dismiss();
        });
```

```
        dialog.show();
    }
    private void submit(String amount_){

                                    FirebaseFirestore     db     =
FirebaseFirestore.getInstance();
                               CollectionReference    postsRef    =
db.collection("Donation");
        String donationID = postsRef.document().getId();

        try {
            data.gui gui = new data.gui();
                                    sendmail(email_user,gui.d_t_r,
gui.donation_content_receive(amount_,name,donationID));
                                     sendmail(email_user,gui.d_t,
gui.donation_content(amount_,name,donationID));
        } catch (IOException e) {
            throw new RuntimeException(e);
        }

        HashMap<String, Object> hashMap = new HashMap<>();
        hashMap.put("Send", ic);
        hashMap.put("To", userID);
        hashMap.put("Amount", amount_);
        hashMap.put("timestamp", new Date());
        hashMap.put("id", donationID);

        postsRef.document(donationID).set(hashMap)
                .addOnSuccessListener(aVoid -> {
                            showDialog("Success", "Successfully
Donation!");
                })
                .addOnFailureListener(e -> {
                          showDialog("Failed", "Please contact
admin!");
                });
    }
       private  void  followN(TextView  txt_following,TextView
txt_follower){
                             DocumentReference    documentRef    =
FirebaseFirestore.getInstance()
                .collection("Follow")
                .document(userID);
                        CollectionReference    followingRef    =
documentRef.collection("following");

followingRef.addSnapshotListener((queryDocumentSnapshots,  e)
-> {
            if (e != null) {
                Log.w("Firestore", "Listen failed", e);
                txt_following.setText("0");
                return;
            }

                        if  (queryDocumentSnapshots  !=  null  &&
!queryDocumentSnapshots.isEmpty()) {
                                              int   followingCount   =
queryDocumentSnapshots.size();
                txt_following.setText(""+followingCount);
```

```
            } else {
                txt_following.setText("0");
            }
        });

                        DocumentReference documentfollower =
FirebaseFirestore.getInstance()
                .collection("Follow")
                .document(userID);
                         CollectionReference followerRef =
documentfollower.collection("follower");

followerRef.addSnapshotListener((queryDocumentSnapshots, e)
-> {
            if (e != null) {
                Log.w("Firestore", "Listen failed", e);
                txt_follower.setText("0");
                return;
            }

                       if (queryDocumentSnapshots != null &&
!queryDocumentSnapshots.isEmpty()) {
                                              int followerCount =
queryDocumentSnapshots.size();
                txt_follower.setText(""+followerCount);
            } else {
                txt_follower.setText("0");
            }
        });
    }
    private Boolean check(String x,String y){
        return x.equals(y);
    }
       private void isFollow(String userId,String ic,Button
button){

                           DocumentReference documentRef =
FirebaseFirestore.getInstance()
                .collection("Follow")
                .document(ic);
                        CollectionReference followingRef =
documentRef.collection("following");

followingRef.addSnapshotListener((queryDocumentSnapshots,e)
-> {
            if(e != null){
                return;
            }

            if (!queryDocumentSnapshots.isEmpty()) {
                                         boolean isFollowing =
queryDocumentSnapshots.getDocuments().stream()
                                                 .anyMatch(doc ->
userId.equals(doc.getId()));

                if (isFollowing) {
                    button.setText("Following");

button.setBackgroundColor(ContextCompat.getColor(Profile_view
.this, R.color.grey));
                    button.setTag("Following");
```

```
                } else {
                    button.setText("Follow");

button.setBackgroundColor(ContextCompat.getColor(Profile_view
.this, R.color.blue1));
                    button.setTag("Follow");
                }
            }else {
                button.setText("Follow");

button.setBackgroundColor(ContextCompat.getColor(Profile_view
.this, R.color.blue1));
                button.setTag("Follow");
            }

        });

    }

    private void UpdatePost(){
        RemovePost();
                    FirebaseFirestore firebaseFirestore =
FirebaseFirestore.getInstance();
                                CollectionReference ref =
firebaseFirestore.collection("Posts");
                                  ref.whereEqualTo("publisher",
userID).get().addOnCompleteListener(task -> {
            postList.clear();

            if (task.isSuccessful()) {
                        for (QueryDocumentSnapshot document :
task.getResult()) {
                    Post post = document.toObject(Post.class);
                    postList.add(post);
                }

                postAdapter.notifyDataSetChanged();
            }
        });
    }
    private void RemovePost(){
                    FirebaseFirestore firebaseFirestore =
FirebaseFirestore.getInstance();
                                CollectionReference Ref =
firebaseFirestore.collection("Posts");
                                  Ref.whereEqualTo("publisher",
userID).addSnapshotListener((querySnapshot, e) -> {
            if (e != null) {
                                      Log.e("Firestore Error",
Objects.requireNonNull(e.getMessage()));
                return;
            }
            if (querySnapshot != null) {
                                   for (DocumentChange change :
querySnapshot.getDocumentChanges()) {
                                    if (change.getType() ==
DocumentChange.Type.REMOVED) {

                                        Post postToRemove =
change.getDocument().toObject(Post.class);
```

```
                        int index = -1;
                          for (int i = 0; i < postList.size();
i++) {
                                                                  if
(postList.get(i).getPostId().equals(postToRemove.getPostId())
) {
                                    index = i;
                                    break;
                                }
                            }

                            if (index != -1) {
                                postList.remove(index);

postAdapter.notifyItemRemoved(index);
                            }
                                    }else if (change.getType() ==
DocumentChange.Type.ADDED) {
                                                          Post post =
change.getDocument().toObject(Post.class);
                            postList.add(post);
                            int position = postList.size() - 1;

postAdapter.notifyItemInserted(position);
                        }
                    }
                }
            });
    }
    private void UpdateWorkshop(){
        RemoveWorkshop();
                        FirebaseFirestore firebaseFirestore =
FirebaseFirestore.getInstance();
                                    CollectionReference ref =
firebaseFirestore.collection("Workshop");

ref.whereEqualTo("publisher",userID).get().addOnCompleteListe
ner(task -> {
                workshopList.clear();
                if (task.isSuccessful()) {
                            for (QueryDocumentSnapshot document :
task.getResult()) {
                        String id = document.getString("id");
                                                    String cover =
document.getString("Cover");
                                                    String title =
document.getString("Title");
                        String date = document.getString("Date");
                                                    String start =
document.getString("Start");
                                                 String location =
document.getString("Location");
                                                String publisher =
document.getString("publisher");

                                            Workshop workshop = new
Workshop(id,cover,title,date,start,location,publisher);

                        workshopList.add(workshop);

                    }
```

```
                workshopAdapter.notifyDataSetChanged();
            }
        });
    }
    private void RemoveWorkshop(){
                    FirebaseFirestore firebaseFirestore =
FirebaseFirestore.getInstance();
                                CollectionReference Ref =
firebaseFirestore.collection("Workshop");
                                 Ref.whereEqualTo("publisher",
userID).addSnapshotListener((querySnapshot, e) -> {
            if (e != null) {
                                        Log.e("Firestore Error",
Objects.requireNonNull(e.getMessage()));
                return;
            }
            if (querySnapshot != null) {
                                 for (DocumentChange change :
querySnapshot.getDocumentChanges()) {
                                     if (change.getType() ==
DocumentChange.Type.REMOVED) {
                                                String id =
change.getDocument().getString("id");
                        int index = -1;
                                           for (int i = 0; i <
workshopList.size(); i++) {
                                                                if
(workshopList.get(i).getId().equals(id)) {
                                    index = i;
                                    break;
                                }
                            }
                            if (index != -1) {
                                workshopList.remove(index);

workshopAdapter.notifyItemRemoved(index);
                            }
                                    }else if (change.getType() ==
DocumentChange.Type.ADDED) {
                                                String id =
change.getDocument().getString("id");
                                            String cover =
change.getDocument().getString("Cover");
                                            String title =
change.getDocument().getString("Title");
                                             String date =
change.getDocument().getString("Date");
                                            String start =
change.getDocument().getString("Start");
                                         String location =
change.getDocument().getString("Location");
                                        String publisher =
change.getDocument().getString("publisher");
                                       Timestamp timestamp =
change.getDocument().getTimestamp("timestamp");

                                     Workshop workshop = new
Workshop(id,cover,title,date,start,location,publisher,timesta
mp);
                        workshopList.add(workshop);
```

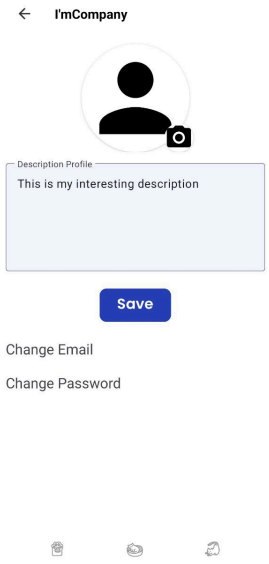
```
workshopAdapter.notifyDataSetChanged();
                    }
                }

            }
        });
    }
    private void UpdateLearn(){
        RemoveLearn();
                    FirebaseFirestore firebaseFirestore =
FirebaseFirestore.getInstance();
                                CollectionReference ref =
firebaseFirestore.collection("Learning");

ref.whereEqualTo("publisher",userID).get().addOnCompleteListe
ner(task -> {
            learnList.clear();
            if (task.isSuccessful()) {
                        for (QueryDocumentSnapshot document :
task.getResult()) {
                    String id = document.getString("id");
                                                String title =
document.getString("title");
                                            String description =
document.getString("description");
                                                Boolean channel =
document.getBoolean("channel");
                                            Timestamp timestamp =
document.getTimestamp("timestamp");
                                                String publisher =
document.getString("publisher");

                                                Learn learn = new
Learn(id,title,description,publisher,channel,timestamp);

                    learnList.add(learn);
                }

                learnAdapter.notifyDataSetChanged();
            }
        });
    }
    private void RemoveLearn(){
                    FirebaseFirestore firebaseFirestore =
FirebaseFirestore.getInstance();
                                CollectionReference Ref =
firebaseFirestore.collection("Learning");
                                  Ref.whereEqualTo("publisher",
userID).addSnapshotListener((querySnapshot, e) -> {
            if (e != null) {
                                        Log.e("Firestore Error",
Objects.requireNonNull(e.getMessage()));
                return;
            }
            if (querySnapshot != null) {
                                 for (DocumentChange change :
querySnapshot.getDocumentChanges()) {
                                     if (change.getType() ==
DocumentChange.Type.REMOVED) {
                                                String id =
change.getDocument().getString("id");
```

```
                        int index = -1;
                        System.out.println(id);
                         for (int i = 0; i < learnList.size();
i++) {
                                                                if
(learnList.get(i).getId().equals(id)) {
                                    index = i;
                                    break;
                                }
                        }
                        if (index != -1) {
                            learnList.remove(index);

learnAdapter.notifyItemRemoved(index);
                        }
                                }else if (change.getType() ==
DocumentChange.Type.ADDED) {
                                                    String id =
change.getDocument().getString("id");
                                                String title =
change.getDocument().getString("title");
                                            String description =
change.getDocument().getString("description");
                                             Boolean channel =
change.getDocument().getBoolean("channel");
                                           Timestamp timestamp =
change.getDocument().getTimestamp("timestamp");
                                            String publisher =
change.getDocument().getString("publisher");
                                            Learn learn = new
Learn(id,title,description,publisher,channel,timestamp);
                        learnList.add(learn);
                        int position = learnList.size() - 1;

learnAdapter.notifyItemInserted(position);
                    }
                }

            }
        });
    }
    private void UpdateWork(){
        RemoveWork();

                      FirebaseFirestore firebaseFirestore =
FirebaseFirestore.getInstance();
                                CollectionReference ref =
firebaseFirestore.collection("Work");

ref.whereEqualTo("publisher",userID).get().addOnCompleteListe
ner(task -> {
            workList.clear();
            if (task.isSuccessful()) {
                        for (QueryDocumentSnapshot document :
task.getResult()) {
                    String id = document.getString("id");
                                            String title =
document.getString("title");
                                        String location =
document.getString("location");
                                         String minimum =
```

```
document.getString("minimum");
                                                    String  maximum  =
document.getString("maximum");
                    String pay = document.getString("pay");
                                                    String  job_title  =
document.getString("job_title");
                                                    String  publisher  =
document.getString("publisher");
                                                Timestamp  timestamp  =
document.getTimestamp("timestamp");
                    String salary;
                    if(maximum == null){
                        salary = "RM "+minimum + " / " + pay;
                    }else if(maximum.isEmpty()){
                        salary = "RM "+minimum + " / " + pay;
                    }else{
                                salary ="RM "+ minimum + " - " +
maximum + " / " + pay;

                    }

                                                    Work  work  =  new
Work(id,title,job_title,salary,publisher,location,timestamp);
                    workList.add(work);
                }

                workAdapter.notifyDataSetChanged();
            }
        });
    }
    private void RemoveWork(){
                        FirebaseFirestore  firebaseFirestore  =
FirebaseFirestore.getInstance();
                                    CollectionReference  Ref  =
firebaseFirestore.collection("Work");
                                      Ref.whereEqualTo("publisher",
userID).addSnapshotListener((querySnapshot, e) -> {
            if (e != null) {
                                            Log.e("Firestore  Error",
Objects.requireNonNull(e.getMessage()));
                return;
            }
            if (querySnapshot != null) {
                                     for  (DocumentChange  change  :
querySnapshot.getDocumentChanges()) {
                                         if  (change.getType()  ==
DocumentChange.Type.REMOVED) {
                                                        String  id  =
change.getDocument().getString("id");
                        int index = -1;
                        System.out.println(id);
                          for (int i = 0; i < workList.size();
i++) {
                                                                if
(workList.get(i).getId().equals(id)) {
                                    index = i;
                                    break;
                                }
                            }
                        System.out.println(index);
                        if (index != -1) {
```

```
                            workList.remove(index);

workAdapter.notifyItemRemoved(index);
                        }
                                }else if (change.getType() ==
DocumentChange.Type.ADDED) {
                                                    String id =
change.getDocument().getString("id");
                                                  String title =
change.getDocument().getString("title");
                                                 String location =
change.getDocument().getString("location");
                                                  String minimum =
change.getDocument().getString("minimum");
                                                  String maximum =
change.getDocument().getString("maximum");
                                                     String pay =
change.getDocument().getString("pay");
                                                 String job_title =
change.getDocument().getString("job_title");
                                                 String publisher =
change.getDocument().getString("publisher");
                                               Timestamp timestamp =
change.getDocument().getTimestamp("timestamp");
                        String salary;
                        if(maximum == null){
                               salary = "RM "+minimum + " / " +
pay;
                        }else if(maximum.isEmpty()){
                               salary = "RM "+minimum + " / " +
pay;
                        }else{
                               salary ="RM "+ minimum + " - " +
maximum + " / " + pay;

                        }

                                                  Work work = new
Work(id,title,job_title,salary,publisher,location,timestamp);
                        workList.add(work);
                        int position = workList.size() - 1;

workAdapter.notifyItemInserted(position);
                    }
                }

            }
        });
    }
}
```

Edit Profile Activity




```
package com.um.linkcamp;

import static function.convert.encodeImageToBase64;
import static function.function.sendmail;

import android.app.Dialog;
import android.content.ContentResolver;
import android.content.Intent;
import android.database.Cursor;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.net.Uri;
import android.os.Bundle;
import android.util.Base64;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.TextView;

import androidx.activity.EdgeToEdge;
import androidx.annotation.Nullable;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.graphics.Insets;
import androidx.core.view.ViewCompat;
import androidx.core.view.WindowInsetsCompat;

import com.github.dhaval2404.imagepicker.ImagePicker;
import
com.google.android.material.textfield.TextInputEditText;
import com.google.common.base.FinalizablePhantomReference;
import com.google.firebase.firestore.DocumentReference;
import com.google.firebase.firestore.FieldValue;
import com.google.firebase.firestore.FirebaseFirestore;

import org.w3c.dom.Text;

import java.io.ByteArrayOutputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.InputStream;
import java.security.NoSuchAlgorithmException;
import java.util.HashMap;
import java.util.Map;
import java.util.Objects;

import data.DatabaseHelper;
import data.gui;
import data.register;
import function.VerifyLogin;

public class Edit_profile extends AppCompatActivity {
    private DatabaseHelper dbHelper;
    String ic,name,profile,description,email,password;
    FirebaseFirestore firebaseFirestore;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        EdgeToEdge.enable(this);
```

```
        setContentView(R.layout.activity_edit_profile);

ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id
.main), (v, insets) -> {
            Insets systemBars =
insets.getInsets(WindowInsetsCompat.Type.systemBars());
            v.setPadding(systemBars.left, systemBars.top,
systemBars.right, systemBars.bottom);
            return insets;
        });

        dbHelper = new DatabaseHelper(Edit_profile.this);
        VerifyLogin verifyLogin = new
VerifyLogin(Edit_profile.this);
        if (verifyLogin.isDatabaseExist()) {
            verifyLogin.verify(result -> {
                if ("other".equals(result)) {
                    dbHelper.clearUserData();
                    finish();
                }else if (!("login".equals(result))) {
                    dbHelper.clearUserData();
                    finish();
                }
            });
        }else{
            finish();
            return;
        }
        Cursor cursor = dbHelper.getUserData();
        if (cursor.moveToFirst()) {
            ic =
cursor.getString(cursor.getColumnIndex("ic"));
            name =
cursor.getString(cursor.getColumnIndex("name"));
            email =
cursor.getString(cursor.getColumnIndex("email"));
            password =
cursor.getString(cursor.getColumnIndex("password"));
        }

        ImageView back = findViewById(R.id.back);

        back.setOnClickListener(v -> {
            Intent intent1 = new
Intent(Edit_profile.this,Profile_view.class);
            intent1.putExtra("UserID",ic);
            startActivity(intent1);
            finish();
        });

        TextView txtName = findViewById(R.id.username);
        txtName.setText(name);

        ImageView imgProfile = findViewById(R.id.profile);

        TextInputEditText txt_description =
findViewById(R.id.text_input_description);

        firebaseFirestore = FirebaseFirestore.getInstance();
        DocumentReference Ref =
firebaseFirestore.collection("Users").document(ic);
```

```
        Ref.get().addOnSuccessListener(Snapshot -> {
            if(Snapshot.exists()){
                profile = Snapshot.getString("profile");
                description =
Snapshot.getString("description");
                if(profile.equals("skip")){

imgProfile.setImageResource(R.drawable.icon_person);
                }else {
                    try {
                        byte[] imageBytes =
Base64.decode(profile, Base64.DEFAULT);
                        Bitmap bitmap =
BitmapFactory.decodeByteArray(imageBytes, 0,
imageBytes.length);
                        imgProfile.setImageBitmap(bitmap);
                    } catch (Exception e) {
                        e.printStackTrace();
                    }
                }
                txt_description.setText(description);
            }
        });

        Button btnSave = findViewById(R.id.next);
        btnSave.setOnClickListener(v -> {

            description =
Objects.requireNonNull(txt_description.getText()).toString(
).trim();
            Map<String, Object> updates = new HashMap<>();
            updates.put("profile", profile);
            updates.put("description", description);

            Ref.update(updates).addOnCompleteListener(task
-> {
                if (task.isSuccessful()) {
                    showDialog("Success","Successfully
updated!");
                } else {
                    System.out.println(task.getException());
                    showDialog("Failed", "Please Contact
admin!");
                }
            });
        });

        imgProfile.setOnClickListener(v -> {
            ImagePicker.with(Edit_profile.this)
                    .cropSquare()
                    .compress(1024)
                    .maxResultSize(1080, 1080)
                    .start();
        });

        ImageView upload_profile =findViewById(R.id.upload);
        upload_profile.setOnClickListener(v -> {
            ImagePicker.with(Edit_profile.this)
                    .cropSquare()
                    .compress(1024)
                    .maxResultSize(1080, 1080)
```

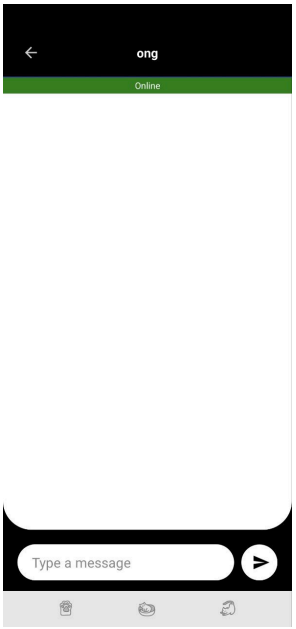
```
                    .start();
        });

        TextView txt_pass = findViewById(R.id.Change_pass);
        TextView txt_email =
findViewById(R.id.Change_email);

        txt_email.setOnClickListener(v -> {
            gui gui = new gui();
            String number =
String.valueOf(function.function.random());
            String number2 =
String.valueOf(function.function.random());
            try {
                sendmail(email,gui.otp_t,
gui.otp_v(number));
            } catch (IOException e) {
                throw new RuntimeException(e);
            }
            register r = register.getInstance();
            try {
                String password =
function.function.hashPassword(number, number2);
                r.setPassword(password);
            } catch (NoSuchAlgorithmException e) {
                throw new RuntimeException(e);
            }
            r.setEmail(email);
            r.setOtp(3);
            r.setPage(5);
            r.setProfile(password);
            r.setIc(ic);
            r.setSalt(number2);
            Intent intent = new
Intent(Edit_profile.this,OTPv.class);
            startActivity(intent);
        });

        txt_pass.setOnClickListener(v -> {
            gui gui = new gui();
            String number =
String.valueOf(function.function.random());
            String number2 =
String.valueOf(function.function.random());
            try {
                sendmail(email,gui.otp_t,
gui.otp_v(number));
            } catch (IOException e) {
                throw new RuntimeException(e);
            }
            register r = register.getInstance();
            try {
                String password =
function.function.hashPassword(number, number2);
                r.setPassword(password);
            } catch (NoSuchAlgorithmException e) {
                throw new RuntimeException(e);
            }
            r.setEmail(email);
            r.setOtp(3);
            r.setPage(7);
```

```
            r.setProfile(password);
            r.setIc(ic);
            r.setSalt(number2);
            Intent intent = new
Intent(Edit_profile.this,OTPv.class);
            startActivity(intent);
        });


    }
    @Override
    protected void onActivityResult(int requestCode, int
resultCode, @Nullable Intent data) {
        super.onActivityResult(requestCode, resultCode,
data);
        if(data != null){
            ImageView imageProfile =
findViewById(R.id.profile);
            Uri selectedImageUri = data.getData();

            if (selectedImageUri != null) {

                profile =
encodeImageToBase64(selectedImageUri);

                imageProfile.setImageURI(selectedImageUri);
            }

        }

    }
    private void showDialog(String title, String message) {
        Dialog dialog = new Dialog(Edit_profile.this);
        dialog.setContentView(R.layout.create_dialog);

Objects.requireNonNull(dialog.getWindow()).setLayout(ViewGr
oup.LayoutParams.WRAP_CONTENT,
ViewGroup.LayoutParams.WRAP_CONTENT);

dialog.getWindow().setBackgroundDrawable(getDrawable(R.draw
able.dialog));


        TextView t = dialog.findViewById(R.id.tittle);
        TextView d = dialog.findViewById(R.id.detail);
        Button btnC = dialog.findViewById(R.id.confirm);

        t.setText(title);
        d.setText(message);
        btnC.setText("Close");
        btnC.setOnClickListener(v -> {
            dialog.dismiss();
        });

        dialog.show();
    }

    private String encodeImageToBase64(Uri imageUri) {
        String base64String = null;
        try {
            // Open the InputStream
```

```
            InputStream inputStream =
Edit_profile.this.getContentResolver().openInputStream(imag
eUri);
            if (inputStream != null) {
                // Decode the image to Bitmap
                Bitmap bitmap =
BitmapFactory.decodeStream(inputStream);
                if (bitmap != null) {
                    // Compress and convert Bitmap to byte
array
                    ByteArrayOutputStream
byteArrayOutputStream = new ByteArrayOutputStream();

bitmap.compress(Bitmap.CompressFormat.JPEG, 10,
byteArrayOutputStream);
                    byte[] imageBytes =
byteArrayOutputStream.toByteArray();

                    base64String =
Base64.encodeToString(imageBytes, Base64.DEFAULT);
                } else {
                    System.out.println("Failed to decode
image.");
                }
            } else {
                System.out.println("Failed to open
InputStream.");
            }
        } catch (IOException e) {
            System.out.println("Error encoding image to
Base64.  " + e);
        }
        return base64String;
    }

}
```

6.2.3 Messaging Module

Message Page




```
package com.um.linkcamp;

import android.content.Intent;
import android.database.Cursor;
import android.os.Bundle;
import android.view.View;

import com.google.android.gms.tasks.OnCompleteListener;
import com.google.firebase.firestore.DocumentChange;
import com.google.firebase.firestore.DocumentSnapshot;
import com.google.firebase.firestore.EventListener;
import com.google.firebase.firestore.FieldValue;
import com.google.firebase.firestore.QuerySnapshot;

import androidx.activity.EdgeToEdge;
import androidx.core.graphics.Insets;
import androidx.core.view.ViewCompat;
import androidx.core.view.WindowInsetsCompat;

import com.google.firebase.firestore.DocumentReference;
import com.google.firebase.firestore.FirebaseFirestore;
import com.um.linkcamp.databinding.ActivityChatBinding;

import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Collections;
import java.util.Date;
import java.util.HashMap;
import java.util.List;
import java.util.Locale;
import java.util.Objects;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;
import java.util.concurrent.atomic.AtomicReference;

import Adapter.ChatAdapter;
import data.Constants;
import data.DatabaseHelper;
import function.ChatNotificationSender;
import function.VerifyLogin;
import model.ChatMessage;

public class Chat extends BaseActivity {
    private ActivityChatBinding binding;
    String userID, name,profile,ic,s_name,s_profile,token;
    private List<ChatMessage> chatMessages;
    private ChatAdapter chatAdapter;
    private FirebaseFirestore database;
    DatabaseHelper dbHelper;
    private String conversionId = null;
    private Boolean isReceiverAvailable = true;

    @Override
```

```
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        EdgeToEdge.enable(this);
        binding =
ActivityChatBinding.inflate(getLayoutInflater());
        setContentView(binding.getRoot());

ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.
main), (v, insets) -> {
            Insets systemBars =
insets.getInsets(WindowInsetsCompat.Type.systemBars());
            v.setPadding(systemBars.left, systemBars.top,
systemBars.right, systemBars.bottom);
            return insets;
        });

        dbHelper = new DatabaseHelper(Chat.this);
        VerifyLogin verifyLogin = new VerifyLogin(Chat.this);
        if (verifyLogin.isDatabaseExist()) {
            verifyLogin.verify(result -> {
                if ("other".equals(result)) {
                    dbHelper.clearUserData();
                    finish();
                }else if (!("login".equals(result))) {
                    dbHelper.clearUserData();
                    finish();
                }
            });
        }else{
            finish();
            return;
        }
        binding.imageInfo.setVisibility(View.INVISIBLE);
        Cursor cursor = dbHelper.getUserData();
        if (cursor.moveToFirst()) {
            ic =
cursor.getString(cursor.getColumnIndex("ic"));
            s_name =
cursor.getString(cursor.getColumnIndex("name"));
            s_profile =
cursor.getString(cursor.getColumnIndex("profile"));
        }

        loadReceiverDetails();

        setListeners();
        getInfo();
    }
    private void loadReceiverDetails(){
        AtomicReference<Intent> intent = new
AtomicReference<>(getIntent());
        userID = intent.get().getStringExtra("userId");
        name = intent.get().getStringExtra("name");
        if(userID.isEmpty()){
            finish();
        }else if (userID == null){
            finish();
        }
        FirebaseFirestore db =
FirebaseFirestore.getInstance();
        db.collection("countMessage")
```

```
                .whereEqualTo("receiverId", ic)
                .whereEqualTo("senderId", userID)
                .get()
                .addOnSuccessListener(querySnapshot -> {
                    if (!querySnapshot.isEmpty()) {
                        for (DocumentSnapshot
documentSnapshot : querySnapshot) {
                            String documentId =
documentSnapshot.getId();
                            db.collection("countMessage")
                                    .document(documentId)
                                    .delete();
                        }
                    }
                });
        binding.textName.setText(name);

    }
    private void setListeners(){
        binding.imageBack.setOnClickListener(v ->
onBackPressed());
        binding.layoutSend.setOnClickListener(v ->
sendMessage());
    }
    private void init(){
        chatMessages = new ArrayList<>();
        chatAdapter = new ChatAdapter(
                profile,
                ic,
                chatMessages,
                0,
                Chat.this
        );
        binding.chatRecyclerView.setAdapter(chatAdapter);
        database = FirebaseFirestore.getInstance();
    }
    private void sendMessage(){
        HashMap<String,Object> message = new HashMap<>();
        message.put(Constants.KEY_SENDER_ID,ic);
        message.put(Constants.KEY_RECEIVER_ID,userID);

message.put(Constants.KEY_MESSAGE,binding.inputMessage.getTe
xt().toString());
        message.put(Constants.KEY_TIMESTAMP, new Date());

database.collection(Constants.KEY_COLLECTION_CHAT).add(messa
ge);
        if(conversionId != null){

updateConversion(binding.inputMessage.getText().toString());
        }else{
            HashMap<String,Object> conversion = new
HashMap<>();
            conversion.put(Constants.KEY_SENDER_ID,ic);
            conversion.put(Constants.KEY_SENDER_NAME,s_name);

conversion.put(Constants.KEY_SENDER_IMAGE,s_profile);
            conversion.put(Constants.KEY_RECEIVER_ID,userID);
            conversion.put(Constants.KEY_RECEIVER_NAME,name);

conversion.put(Constants.KEY_RECEIVER_IMAGE,profile);
```

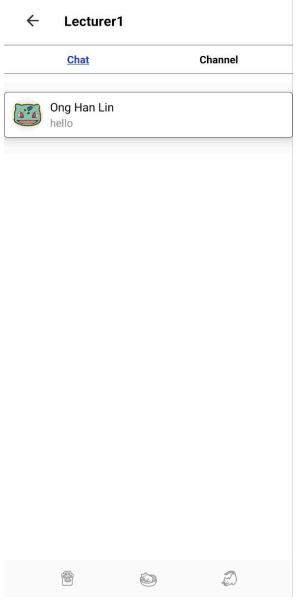
```
conversion.put(Constants.KEY_LAST_MESSAGE,binding.inputMessa
ge.getText().toString());
            conversion.put(Constants.KEY_TIMESTAMP,new
Date());
            addConversion(conversion);
        }
        if(!isReceiverAvailable) {
            database.collection("countMessage")
                    .whereEqualTo("receiverId", userID)
                    .whereEqualTo("senderId", ic)
                    .get()
                    .addOnSuccessListener(snapshot -> {
                        HashMap<String, Object> count = new
HashMap<>();
                        if (!snapshot.isEmpty()) {
                            // Assuming only one document
matches the query
                            DocumentSnapshot doc =
snapshot.getDocuments().get(0);

                            Long countValue =
doc.getLong("count"); // Safely get the count field
                            int count2 = (countValue != null)
? countValue.intValue() : 0;

                            count.put("receiverId", userID);
                            count.put("count", count2 + 1);
                            count.put("senderId", ic);

                            // Update the existing document

database.collection("countMessage")
                                    .document(doc.getId())
                                    .set(count);
                        } else {
                            count.put("receiverId", userID);
                            count.put("count", 1);
                            count.put("senderId", ic);

                            // Add a new document

database.collection("countMessage")
                                    .add(count);
                        }
                    });
            String body =
binding.inputMessage.getText().toString();
            if ("new".equals(token)) {
                HashMap<String,Object> laterSend = new
HashMap<>();
                laterSend.put("senderName",s_name);
                laterSend.put("receiverId",userID);
                laterSend.put("message",body);
                laterSend.put("senderId",ic);
                laterSend.put("timestamp",
FieldValue.serverTimestamp());
                laterSend.put("channel","Channel");
                database.collection("later").add(laterSend);
            }else{
                ExecutorService executor =
```

```
Executors.newSingleThreadExecutor();
                executor.execute(() -> {

ChatNotificationSender.sendNotification(Chat.this,token,s_na
me,body,ic,"Chat");
                });
            }
        }
        binding.inputMessage.setText(null);
    }
    private void listenAvailabilityOfReceiver(){
        FirebaseFirestore database =
FirebaseFirestore.getInstance();
        database.collection("Users").document(userID)
                .addSnapshotListener(Chat.this,((value,
error) -> {
                    if(error != null){
                        return;
                    }
                    if (value != null){

if(value.getLong(Constants.KEY_AVAILABILITY) != null){
                            int available =
Objects.requireNonNull(

value.getLong(Constants.KEY_AVAILABILITY)
                            ).intValue();
                            isReceiverAvailable = available
== 1;
                        }
                    }
                    if (isReceiverAvailable){

binding.textAvailability.setVisibility(View.VISIBLE);
                    }else{

binding.textAvailability.setVisibility(View.GONE);
                    }

                }));
    }
    private void listenMessage(){
        database.collection(Constants.KEY_COLLECTION_CHAT)
                .whereEqualTo(Constants.KEY_SENDER_ID,ic)

.whereEqualTo(Constants.KEY_RECEIVER_ID,userID)
                .addSnapshotListener(eventListener);
        database.collection(Constants.KEY_COLLECTION_CHAT)
                .whereEqualTo(Constants.KEY_SENDER_ID,userID)
                .whereEqualTo(Constants.KEY_RECEIVER_ID,ic)
                .addSnapshotListener(eventListener);
    }
    private final EventListener<QuerySnapshot> eventListener
= (value, error) -> {
        if (error != null) {
            System.err.println("Error listening for changes:
" + error.getMessage());
            return;
        }
        if(value != null){
            int count = chatMessages.size();
```

```
            for (DocumentChange documentChange :
value.getDocumentChanges()){
                if(documentChange.getType() ==
DocumentChange.Type.ADDED){
                    ChatMessage chatMessage = new
ChatMessage();
                    chatMessage.senderId =
documentChange.getDocument().getString(Constants.KEY_SENDER_
ID);
                    chatMessage.receiverId =
documentChange.getDocument().getString(Constants.KEY_RECEIVE
R_ID);
                    chatMessage.message =
documentChange.getDocument().getString(Constants.KEY_MESSAGE
);
                    chatMessage.dateTime =
getReadableDateTime(documentChange.getDocument().getDate(Con
stants.KEY_TIMESTAMP));
                    chatMessage.dateObject =
documentChange.getDocument().getDate(Constants.KEY_TIMESTAMP
);
                    chatMessages.add(chatMessage);
                }
            }

Collections.sort(chatMessages,(obj1,obj2)->obj1.dateObject.c
ompareTo(obj2.dateObject));
            if(count == 0){
                chatAdapter.notifyDataSetChanged();
            }else{

chatAdapter.notifyItemRangeChanged(chatMessages.size(),chatM
essages.size());

binding.chatRecyclerView.smoothScrollToPosition(chatMessages
.size() - 1);
            }

binding.chatRecyclerView.setVisibility(View.VISIBLE);
        }
        binding.progressBar.setVisibility(View.GONE);
        if(conversionId == null){
            checkForConversion();
        }
    };
    private String getReadableDateTime(Date date){
        return new SimpleDateFormat("MMMM dd, yyyy - hh:mm
a", Locale.getDefault()).format(date);
    }

    private void getInfo(){
        FirebaseFirestore firebaseFirestore =
FirebaseFirestore.getInstance();
        DocumentReference Ref =
firebaseFirestore.collection("Users").document(userID);
        Ref.get().addOnSuccessListener(Snapshot -> {
            if(Snapshot.exists()){
                profile = Snapshot.getString("profile");
                token = Snapshot.getString("Token");
                init();
                listenMessage();
```

```
            }
        });
    }
    private void addConversion(HashMap<String,Object>
conversion){

database.collection(Constants.KEY_COLLECTION_CONVERSATIONS)
                .add(conversion)
                .addOnSuccessListener(documentReference ->
conversionId = documentReference.getId());
    }
    private void updateConversion(String message){
        DocumentReference documentReference =

database.collection(Constants.KEY_COLLECTION_CONVERSATIONS).
document(conversionId);
        documentReference.update(
                Constants.KEY_LAST_MESSAGE,message,
                Constants.KEY_TIMESTAMP,new Date()
        );
    }
    private void checkForConversion(){
        if(chatMessages.size() != 0){
            checkForConversionRemotely(
                    userID,
                    ic
            );
            checkForConversionRemotely(
                    ic,
                    userID
            );
        }
    }
    private void checkForConversionRemotely(String
senderId,String receiverId){

database.collection(Constants.KEY_COLLECTION_CONVERSATIONS)

.whereEqualTo(Constants.KEY_SENDER_ID,senderId)

.whereEqualTo(Constants.KEY_RECEIVER_ID,receiverId)
                .get()

.addOnCompleteListener(conversionOnCompleteListener);
    }
    private final OnCompleteListener<QuerySnapshot>
conversionOnCompleteListener = task -> {
        if (task.isSuccessful() && task.getResult() != null
&& task.getResult().getDocuments().size() > 0){
            DocumentSnapshot documentSnapshot =
task.getResult().getDocuments().get(0);
            conversionId = documentSnapshot.getId();
        }
    };

    @Override
    protected void onResume() {
        super.onResume();
        listenAvailabilityOfReceiver();
    }
}
```

View Chat Page




```
package com.um.linkcamp;

import android.database.Cursor;
import android.os.Bundle;
import android.text.SpannableString;
import android.text.style.UnderlineSpan;
import android.view.View;
import android.widget.ImageView;
import android.widget.ProgressBar;
import android.widget.TextView;

import androidx.activity.EdgeToEdge;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.graphics.Insets;
import androidx.core.view.ViewCompat;
import androidx.core.view.WindowInsetsCompat;
import androidx.recyclerview.widget.RecyclerView;

import com.google.firebase.firestore.DocumentChange;
import com.google.firebase.firestore.EventListener;
import com.google.firebase.firestore.FirebaseFirestore;
import com.google.firebase.firestore.QuerySnapshot;

import java.util.ArrayList;
import java.util.Collections;
import java.util.List;

import Adapter.RecentConversationsAdapter;
import data.Constants;
import data.DatabaseHelper;
import function.VerifyLogin;
import model.ChatMessage;

public class Chat_Home extends AppCompatActivity {
    DatabaseHelper dbHelper;
    TextView chat,channel;
    String ic,name;
    RecyclerView chatRecycler,channelRecycle;
    ProgressBar progressBar;

    private List<ChatMessage> conversations,channelList;
    private RecentConversationsAdapter
conversationsAdapter,channelAdapter;
    private FirebaseFirestore database;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        EdgeToEdge.enable(this);
        setContentView(R.layout.activity_chat_home);

ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.i
d.main), (v, insets) -> {
            Insets systemBars =
insets.getInsets(WindowInsetsCompat.Type.systemBars());
            v.setPadding(systemBars.left, systemBars.top,
systemBars.right, systemBars.bottom);
            return insets;
        });

        dbHelper = new DatabaseHelper(Chat_Home.this);
```

```
        VerifyLogin verifyLogin = new
VerifyLogin(Chat_Home.this);
        if (verifyLogin.isDatabaseExist()) {
            verifyLogin.verify(result -> {
                if ("other".equals(result)) {
                    dbHelper.clearUserData();
                    finish();
                }else if (!("login".equals(result))) {
                    dbHelper.clearUserData();
                    finish();
                }
            });
        }else{
            finish();
            return;
        }
        Cursor cursor = dbHelper.getUserData();
        if (cursor.moveToFirst()) {
            ic =
cursor.getString(cursor.getColumnIndex("ic"));
            name =
cursor.getString(cursor.getColumnIndex("name"));
        }
        init();

        ImageView btnBack = findViewById(R.id.back);
        btnBack.setOnClickListener(v -> {
            finish();
        });

        TextView txtName = findViewById(R.id.title);
        txtName.setText(name);

        chat = findViewById(R.id.chat);
        channel = findViewById(R.id.channel);

        selected("Chat",chat);

        chat.setOnClickListener(v -> {
            selected("Chat",chat);
            chatRecycler.setVisibility(View.GONE);
            progressBar.setVisibility(View.VISIBLE);
            channelRecycle.setVisibility(View.GONE);
            listenConversations();
        });

        channel.setOnClickListener(v -> {
            selected("Channel",channel);
            chatRecycler.setVisibility(View.GONE);
            progressBar.setVisibility(View.VISIBLE);
            channelRecycle.setVisibility(View.GONE);
            progressBar = findViewById(R.id.progressBar);
            listenChannelConversations();
        });
        listenConversations();
    }
    private void init(){
        conversations = new ArrayList<>();
        conversationsAdapter = new
RecentConversationsAdapter(Chat_Home.this,conversations,ic
,0);
```

```
        chatRecycler =
findViewById(R.id.conversationRecyclerView);
        chatRecycler.setAdapter(conversationsAdapter);
        database = FirebaseFirestore.getInstance();

        channelList = new ArrayList<>();
        channelAdapter = new
RecentConversationsAdapter(Chat_Home.this,channelList,ic,1
);
        channelRecycle =
findViewById(R.id.channelRecyclerView);
        channelRecycle.setAdapter(channelAdapter);

        progressBar = findViewById(R.id.progressBar);
    }
    private void listenConversations(){
        conversations.clear();

database.collection(Constants.KEY_COLLECTION_CONVERSATIONS
)
                .whereEqualTo(Constants.KEY_SENDER_ID,ic)
                .addSnapshotListener(eventListener);

database.collection(Constants.KEY_COLLECTION_CONVERSATIONS
)
                .whereEqualTo(Constants.KEY_RECEIVER_ID,ic)
                .addSnapshotListener(eventListener);
    }
    private final EventListener<QuerySnapshot>
eventListener = (value, error) -> {
        if (error != null){
            return;
        }
        if (value != null){
            for (DocumentChange documentChange :
value.getDocumentChanges()){
                if(documentChange.getType() ==
DocumentChange.Type.ADDED){
                    String senderId =
documentChange.getDocument().getString(Constants.KEY_SENDE
R_ID);
                    String receiverId =
documentChange.getDocument().getString(Constants.KEY_RECEI
VER_ID);
                    ChatMessage chatMessage = new
ChatMessage();
                    chatMessage.senderId = senderId;
                    chatMessage.receiverId = receiverId;
                    if (ic.equals(senderId)){
                        chatMessage.conversionImage =
documentChange.getDocument().getString(Constants.KEY_RECEI
VER_IMAGE);
                        chatMessage.conversionName =
documentChange.getDocument().getString(Constants.KEY_RECEI
VER_NAME);
                        chatMessage.conversionId =
documentChange.getDocument().getString(Constants.KEY_RECEI
VER_ID);
                    }else{
                        chatMessage.conversionImage =
documentChange.getDocument().getString(Constants.KEY_SENDE
```

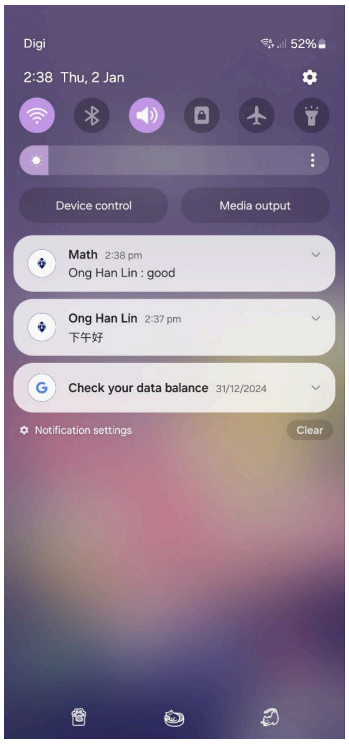
```
R_IMAGE);
                        chatMessage.conversionName =
documentChange.getDocument().getString(Constants.KEY_SENDE
R_NAME);
                        chatMessage.conversionId =
documentChange.getDocument().getString(Constants.KEY_SENDE
R_ID);
                    }
                    chatMessage.message =
documentChange.getDocument().getString(Constants.KEY_LAST_
MESSAGE);
                    chatMessage.dateObject =
documentChange.getDocument().getDate(Constants.KEY_TIMESTA
MP);
                    conversations.add(chatMessage);
                } else if (documentChange.getType() ==
DocumentChange.Type.MODIFIED){
                    for (int i = 0; i <
conversations.size(); i++) {
                        String senderId =
documentChange.getDocument().getString(Constants.KEY_SENDE
R_ID);
                        String receiverId =
documentChange.getDocument().getString(Constants.KEY_RECEI
VER_ID);
                        if
(conversations.get(i).senderId.equals(senderId) &&
conversations.get(i).receiverId.equals(receiverId)){
                            conversations.get(i).message =
documentChange.getDocument().getString(Constants.KEY_LAST_
MESSAGE);
                            conversations.get(i).dateObject
=
documentChange.getDocument().getDate(Constants.KEY_TIMESTA
MP);
                            break;
                        }
                    }
                }
            }
            Collections.sort(conversations,(obj1,obj2)->
obj2.dateObject.compareTo(obj1.dateObject));
            conversationsAdapter.notifyDataSetChanged();
            chatRecycler.smoothScrollToPosition(0);
            chatRecycler.setVisibility(View.VISIBLE);

            progressBar.setVisibility(View.GONE);
        }
    };

    private void listenChannelConversations(){
        channelList.clear();
        database.collection("Channel")

.whereEqualTo(ic,true).addSnapshotListener(((value, error)
-> {
                    if (error != null){

                        return;
                    }
                    if (value != null){
```

```
                        boolean fieldExists = false;
                        for (DocumentChange documentChange
: value.getDocumentChanges()){
                            if(documentChange.getType() ==
DocumentChange.Type.ADDED){
                                fieldExists = true;
                                String channelID =
documentChange.getDocument().getId();

database.collection("channelConversations")

.whereEqualTo(Constants.KEY_RECEIVER_ID,channelID)

.addSnapshotListener(eventChannelListener);
                            } else if
(documentChange.getType() == DocumentChange.Type.REMOVED)
{
                                fieldExists = true;
                                String channelID =
documentChange.getDocument().getId();
                                for (int i = 0; i <
channelList.size(); i++) {

if(channelList.get(i).receiverId.equals(channelID)){

channelList.remove(i);

channelAdapter.notifyDataSetChanged();
                                        break;
                                    }
                                }
                            }
                        }
                        if (!fieldExists) {
                            // No documents with the field
`ic` set to `true`

progressBar.setVisibility(View.GONE);
                            // Optionally inform the user
                        }
                    }
                }));
    }
    private final EventListener<QuerySnapshot>
eventChannelListener = (value, error) -> {
        if (error != null){
            return;
        }
        if (value != null){
            for (DocumentChange documentChange :
value.getDocumentChanges()){

                if(documentChange.getType() ==
DocumentChange.Type.ADDED){
                    String senderId =
documentChange.getDocument().getString(Constants.KEY_SENDE
R_ID);
                    String receiverId =
documentChange.getDocument().getString(Constants.KEY_RECEI
VER_ID);
                    ChatMessage chatMessage = new
```

```
ChatMessage();
                    chatMessage.senderId = senderId;
                    chatMessage.receiverId = receiverId;
                    chatMessage.conversionImage =
documentChange.getDocument().getString(Constants.KEY_RECEI
VER_IMAGE);
                    chatMessage.conversionName =
documentChange.getDocument().getString(Constants.KEY_RECEI
VER_NAME);
                    chatMessage.conversionId =
documentChange.getDocument().getString(Constants.KEY_RECEI
VER_ID);
                    if (ic.equals(senderId)){
                        chatMessage.message = "You :" +
documentChange.getDocument().getString(Constants.KEY_LAST_
MESSAGE);
                    }else{
                        chatMessage.message =
documentChange.getDocument().getString(Constants.KEY_SENDE
R_NAME)+" :"
                                +
documentChange.getDocument().getString(Constants.KEY_LAST_
MESSAGE);
                    }
                    chatMessage.dateObject =
documentChange.getDocument().getDate(Constants.KEY_TIMESTA
MP);
                    channelList.add(chatMessage);
                } else if (documentChange.getType() ==
DocumentChange.Type.MODIFIED){
                    for (int i = 0; i < channelList.size();
i++) {
                        String senderId =
documentChange.getDocument().getString(Constants.KEY_SENDE
R_ID);
                        String receiverId =
documentChange.getDocument().getString(Constants.KEY_RECEI
VER_ID);
                        System.out.println(receiverId);
                        if
(channelList.get(i).receiverId.equals(receiverId)){
                            if(senderId.equals(ic)){
                                channelList.get(i).message
= "You :" +
documentChange.getDocument().getString(Constants.KEY_LAST_
MESSAGE);
                            }else{
                                channelList.get(i).message
=
documentChange.getDocument().getString(Constants.KEY_SENDE
R_NAME)+" :"
                                        +
documentChange.getDocument().getString(Constants.KEY_LAST_
MESSAGE);
                            }
                            channelList.get(i).dateObject =
documentChange.getDocument().getDate(Constants.KEY_TIMESTA
MP);
                            break;
                        }
                    }
```

```
                }
            }
            Collections.sort(channelList,(obj1,obj2)->
obj2.dateObject.compareTo(obj1.dateObject));
            channelAdapter.notifyDataSetChanged();
            channelRecycle.smoothScrollToPosition(0);
            channelRecycle.setVisibility(View.VISIBLE);

            progressBar.setVisibility(View.GONE);
        }
    };

    public void selected(String title,TextView textView){
        chat.setText("Chat");
        channel.setText("Channel");


chat.setTextColor(getResources().getColor(R.color.black,
null));

channel.setTextColor(getResources().getColor(R.color.black
, null));

        SpannableString spannableString = new
SpannableString(title);
        spannableString.setSpan(new UnderlineSpan(), 0,
title.length(), 0);
        textView.setText(spannableString);

textView.setTextColor(getResources().getColor(R.color.blue
1, null));
    }
}
```

6.2.4 Notification Module

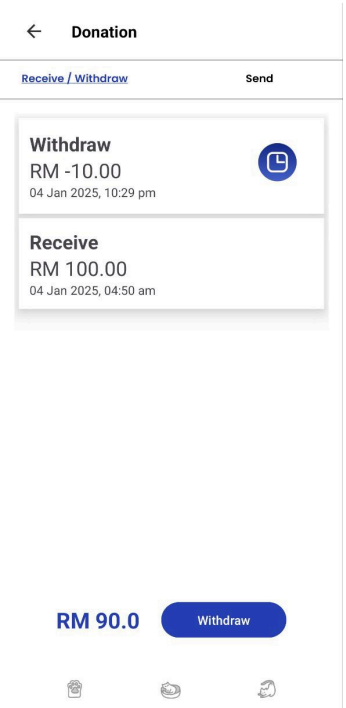



```
package com.um.linkcamp;

import android.app.NotificationChannel;
import android.app.NotificationManager;
import android.app.PendingIntent;
import android.content.Intent;
import android.os.Build;

import androidx.core.app.NotificationCompat;

import
com.google.firebase.messaging.FirebaseMessagingService;
import com.google.firebase.messaging.RemoteMessage;

public class FCMNotificationService extends
FirebaseMessagingService {
   @Override
   public void onMessageReceived(RemoteMessage remoteMessage)
{
       super.onMessageReceived(remoteMessage);

       String title =
remoteMessage.getNotification().getTitle();
       String body =
remoteMessage.getNotification().getBody();
       String channel1 =
remoteMessage.getData().get("channel");
       Intent intent = null;
       System.out.println( channel1 + " " + title );
       if("Channel".equals(channel1)){
           intent = new Intent(this, Channel.class);
           intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
           intent.putExtra("channelName",title);

intent.putExtra("channelID",remoteMessage.getData().get("user
Id"));
       }else{
           System.out.println("hi");
           // Pass data to MainActivity or show a
notification
            intent = new Intent(this, Chat.class);

           intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
           intent.putExtra("name",title);

intent.putExtra("userId",remoteMessage.getData().get("userId"
));
       }

       PendingIntent pendingIntent =
PendingIntent.getActivity(this, 0, intent,
PendingIntent.FLAG_ONE_SHOT | PendingIntent.FLAG_IMMUTABLE);

       NotificationManager notificationManager =
(NotificationManager) getSystemService(NOTIFICATION_SERVICE);
       String channelId = "default_channel";
```

```
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
            NotificationChannel channel = new
NotificationChannel(
                    channelId, "Default Channel",
NotificationManager.IMPORTANCE_HIGH
            );

notificationManager.createNotificationChannel(channel);
        }

        NotificationCompat.Builder notificationBuilder = new
NotificationCompat.Builder(this, channelId)
                .setSmallIcon(R.drawable.ic_notification)
                .setContentTitle(title)
                .setContentText(body)
                .setAutoCancel(true)
                .setContentIntent(pendingIntent);

        notificationManager.notify(0,
notificationBuilder.build());
    }

    @Override
    public void onNewToken(String token) {
        super.onNewToken(token);
        // Send the token to your server or update the
database
    }
}
```

6.2.5 Payment Module

View Donation Page




```
package com.um.linkcamp;

import static android.graphics.Color.BLACK;
import static android.graphics.Color.RED;

import android.app.Dialog;
import android.content.res.ColorStateList;
import android.database.Cursor;
import android.os.Bundle;
import android.text.InputFilter;
import android.text.SpannableString;
import android.text.style.UnderlineSpan;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.TextView;

import androidx.activity.EdgeToEdge;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.graphics.Insets;
import androidx.core.view.ViewCompat;
import androidx.core.view.WindowInsetsCompat;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import
com.google.android.material.textfield.TextInputEditText;
import com.google.android.material.textfield.TextInputLayout;
import com.google.firebase.Timestamp;
import com.google.firebase.firestore.CollectionReference;
import com.google.firebase.firestore.DocumentChange;
import com.google.firebase.firestore.EventListener;
import com.google.firebase.firestore.FieldValue;
import com.google.firebase.firestore.FirebaseFirestore;
import com.google.firebase.firestore.Query;
import com.google.firebase.firestore.QueryDocumentSnapshot;
import com.google.firebase.firestore.QuerySnapshot;

import java.math.BigDecimal;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Collections;
import java.util.Date;
import java.util.HashMap;
import java.util.List;
import java.util.Locale;
import java.util.Objects;

import Adapter.ReceiveAdapter;
import Adapter.SendAdapter;
import data.Constants;
import data.DatabaseHelper;
import function.DecimalDigitsInputFilter;
import function.SpacingItemDecoration;
```

```
import function.VerifyLogin;
import model.Post;
import model.Receive;

public class View_Donation extends AppCompatActivity {
   private DatabaseHelper dbHelper;
   FirebaseFirestore firestore;
   double total = 0;
   String ic;
   String account = null,bank_name = null,amount_w = null;
   ReceiveAdapter receiveAdapter;
   TextView txt_get,txt_send;
   List<Receive> receiveList;
   List<Receive> sendList;
   SendAdapter sendAdapter;
   @Override
   protected void onCreate(Bundle savedInstanceState) {
       super.onCreate(savedInstanceState);
       EdgeToEdge.enable(this);
       setContentView(R.layout.activity_view_donation);

ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.m
ain), (v, insets) -> {
           Insets systemBars =
insets.getInsets(WindowInsetsCompat.Type.systemBars());
           v.setPadding(systemBars.left, systemBars.top,
systemBars.right, systemBars.bottom);
           return insets;
       });
       dbHelper = new DatabaseHelper(View_Donation.this);
       VerifyLogin verifyLogin = new
VerifyLogin(View_Donation.this);
       if (verifyLogin.isDatabaseExist()) {
           verifyLogin.verify(result -> {
               if ("other".equals(result)) {
                   dbHelper.clearUserData();
                   finish();
               } else if (!("login".equals(result))) {
                   dbHelper.clearUserData();
                   finish();
               }
           });
       } else {
           finish();
           return;
       }
       Cursor cursor = dbHelper.getUserData();
       if (cursor.moveToFirst()) {
           ic =
cursor.getString(cursor.getColumnIndex("ic"));
       }
       ImageView btnBack = findViewById(R.id.back);
       btnBack.setOnClickListener(v -> finish());

       RecyclerView receiveView =
findViewById(R.id.receive_view);
       RecyclerView sendView = findViewById(R.id.send_view);
       int spacingInPixels =
getResources().getDimensionPixelSize(R.dimen.recycler_item_sp
acing);
       setupRecyclerView(receiveView, new
```

```
LinearLayoutManager(this), spacingInPixels);
        setupRecyclerView(sendView, new
LinearLayoutManager(this), spacingInPixels);

        receiveList = new ArrayList<>();
        receiveAdapter = new ReceiveAdapter(receiveList);
        receiveView.setAdapter(receiveAdapter);
        receiveView.setVisibility(View.VISIBLE);

        sendList = new ArrayList<>();
        sendAdapter = new
SendAdapter(View_Donation.this,sendList);
        sendView.setAdapter(sendAdapter);

        firestore = FirebaseFirestore.getInstance();
        readReceive();
        sendReceive();

        Button btnW = findViewById(R.id.withdraw);
        btnW.setOnClickListener(v -> {
            showWithdraw();
        });
        txt_get = findViewById(R.id.get);
        txt_send = findViewById(R.id.give);

        selected("Receive / Withdraw",txt_get);

        txt_get.setOnClickListener(v -> {
            readReceive();
            selected("Receive / Withdraw",txt_get);
            receiveView.setVisibility(View.VISIBLE);
            sendView.setVisibility(View.GONE);
        });
        txt_send.setOnClickListener(v -> {
            selected("Send",txt_send);
            receiveView.setVisibility(View.GONE);
            sendView.setVisibility(View.VISIBLE);
        });
    }
    private void setupRecyclerView(RecyclerView recyclerView,
LinearLayoutManager layoutManager, int spacingInPixels) {
        recyclerView.setHasFixedSize(true);
        recyclerView.setLayoutManager(layoutManager);
        recyclerView.addItemDecoration(new
SpacingItemDecoration(spacingInPixels));
        recyclerView.setVisibility(View.GONE);
    }
    private void showWithdraw(){
        Dialog dialog = new Dialog(View_Donation.this);
        dialog.setContentView(R.layout.withdraw_dialog);

Objects.requireNonNull(dialog.getWindow()).setLayout(ViewGrou
p.LayoutParams.WRAP_CONTENT,
ViewGroup.LayoutParams.WRAP_CONTENT);

dialog.getWindow().setBackgroundDrawable(getDrawable(R.drawab
le.dialog));

        TextInputEditText bankAccount =
dialog.findViewById(R.id.bank_number);
        TextInputEditText bankName =
```

```
dialog.findViewById(R.id.bank_name);
       TextInputEditText amount =
dialog.findViewById(R.id.amount_value);

       amount.setFilters(new InputFilter[]{new
DecimalDigitsInputFilter(2)});

       Button btnC = dialog.findViewById(R.id.confirm);
       btnC.setOnClickListener(v -> {
           account = bankAccount.getText().toString().trim();
           bank_name = bankName.getText().toString().trim();
           amount_w = amount.getText().toString().trim();

           if(checkEmtpy(dialog)){
               FirebaseFirestore db =
FirebaseFirestore.getInstance();
               CollectionReference postsRef =
db.collection("Donation");
               String donationID =
postsRef.document().getId();
               HashMap<String, Object> hashMap = new
HashMap<>();
               hashMap.put("Send", "Withdraw");
               hashMap.put("To", ic);
               hashMap.put("Amount", amount_w);
               hashMap.put("timestamp", new Date());
               hashMap.put("Account", account);
               hashMap.put("Bank_Name", bank_name);
               hashMap.put("status", 1);
               hashMap.put("id", donationID);

               postsRef.document(donationID).set(hashMap)
                       .addOnSuccessListener(aVoid -> {
                           showDialog("Success", "Admin will
approve in 1-3 day!");
                           dialog.dismiss();
                       })
                       .addOnFailureListener(e -> {
                           showDialog("Failed", "Please
contact admin!");
                           dialog.dismiss();
                       });
           }
       });

       Button cancel = dialog.findViewById(R.id.cancel);
       cancel.setOnClickListener(v -> {
           dialog.dismiss();
       });

       dialog.show();
   }
   public void selected(String title,TextView textView){
       txt_get.setText("Recieve / Withdraw");
       txt_send.setText("Send");


txt_get.setTextColor(getResources().getColor(R.color.black,
null));

txt_send.setTextColor(getResources().getColor(R.color.black,
```

```
null));

        SpannableString spannableString = new
SpannableString(title);
        spannableString.setSpan(new UnderlineSpan(), 0,
title.length(), 0);
        textView.setText(spannableString);

textView.setTextColor(getResources().getColor(R.color.blue1,
null));
    }
    private void showDialog(String title, String message) {
        Dialog dialog = new Dialog(View_Donation.this);
        dialog.setContentView(R.layout.create_dialog);

Objects.requireNonNull(dialog.getWindow()).setLayout(ViewGrou
p.LayoutParams.WRAP_CONTENT,
ViewGroup.LayoutParams.WRAP_CONTENT);

dialog.getWindow().setBackgroundDrawable(getDrawable(R.drawab
le.dialog));


        TextView t = dialog.findViewById(R.id.tittle);
        TextView d = dialog.findViewById(R.id.detail);
        Button btnC = dialog.findViewById(R.id.confirm);

        t.setText(title);
        d.setText(message);
        btnC.setText("Close");
        btnC.setOnClickListener(v -> {
            dialog.dismiss();
        });

        dialog.show();
    }
    private Boolean checkEmtpy(Dialog dialog){
        boolean check = true;
        TextInputLayout l_account =
dialog.findViewById(R.id.l_bank_number);
        TextInputLayout l_name =
dialog.findViewById(R.id.l_bank_name);
        TextInputLayout l_amount =
dialog.findViewById(R.id.amount);

        if(account == null){
            check = false;

l_account.setHelperTextColor(ColorStateList.valueOf(RED));
            l_account.setHelperText("Please fill in Account
Bank");
        } else if (account.isEmpty()) {
            check = false;

l_account.setHelperTextColor(ColorStateList.valueOf(RED));
            l_account.setHelperText("Please fill in Account
Bank");
        }else{

l_account.setHelperTextColor(ColorStateList.valueOf(BLACK));
            l_account.setHelperText("");
```

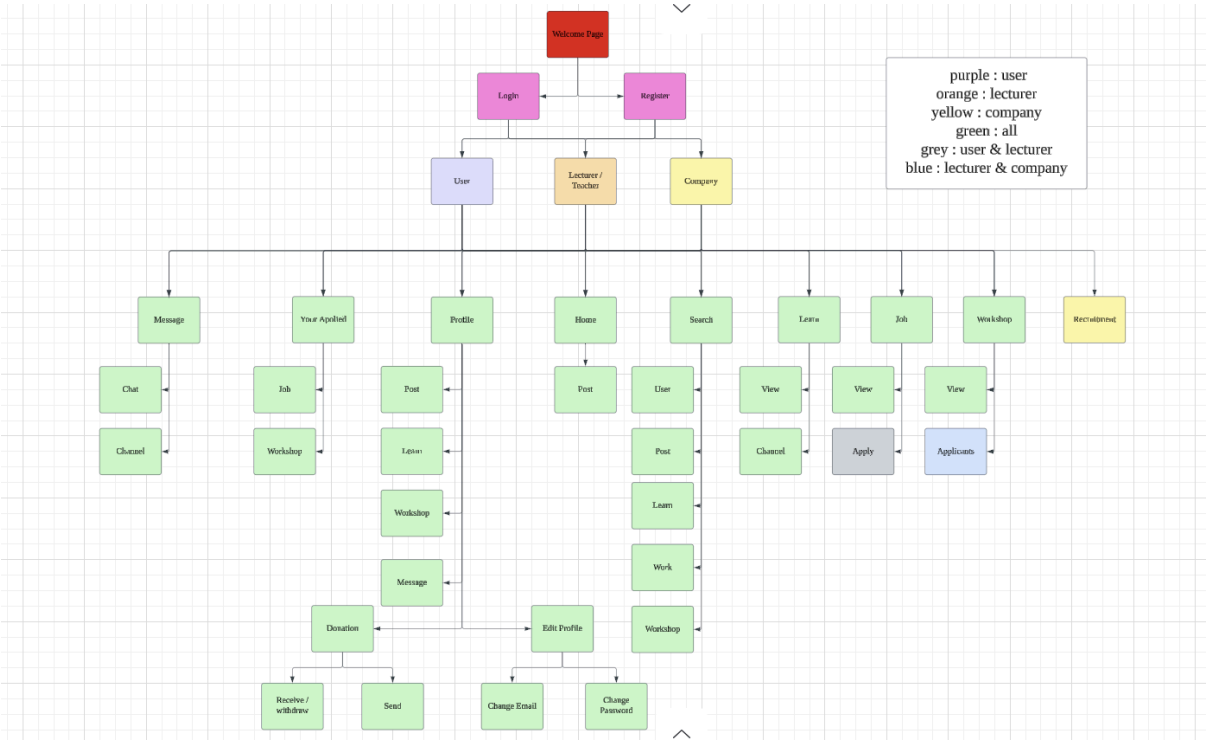
```
        }

        if(bank_name == null){
            check = false;

l_name.setHelperTextColor(ColorStateList.valueOf(RED));
            l_name.setHelperText("Please fill in Bank Name");
        } else if (bank_name.isEmpty()) {
            check = false;

l_name.setHelperTextColor(ColorStateList.valueOf(RED));
            l_name.setHelperText("Please fill in Bank Name");
        }else{

l_name.setHelperTextColor(ColorStateList.valueOf(BLACK));
            l_name.setHelperText("");
        }

        if(amount_w == null){
            check = false;

l_amount.setHelperTextColor(ColorStateList.valueOf(RED));
            l_amount.setHelperText("Please fill in Amount
(RM)");

        } else if (amount_w.isEmpty()) {
            check = false;

l_amount.setHelperTextColor(ColorStateList.valueOf(RED));
            l_amount.setHelperText("Please fill in Amount
(RM)");
        }else{
            Double amount = Double.valueOf(amount_w);
            if(amount > total){
                check = false;

l_amount.setHelperTextColor(ColorStateList.valueOf(RED));
                l_amount.setHelperText("Total Amount is : RM
"+ total +"\nCannot withdraw more than Total Amount");
            }else{

l_amount.setHelperTextColor(ColorStateList.valueOf(BLACK));
                l_amount.setHelperText("");
            }
        }
        return check;
    }

    private void readReceive(){
        total = 0;
        receiveList.clear();
        firestore.collection("Donation")
                .whereEqualTo("To",ic)
                .addSnapshotListener(eventListener);

    }
    private final EventListener<QuerySnapshot> eventListener =
(value, error) -> {
        if (error != null){
            return;
        }
```

```
        if (value != null) {
            for (DocumentChange documentChange :
value.getDocumentChanges()) {
                if (documentChange.getType() ==
DocumentChange.Type.ADDED) {
                    Receive receive = new Receive();
                    receive.donationID =
documentChange.getDocument().getString("id");
                    receive.userID =
documentChange.getDocument().getString("Send");
                    if("Withdraw".equals(receive.userID)){
                        receive.status =
documentChange.getDocument().getLong("status").intValue();

                        if(receive.status == 1 ||
receive.status == 2){
                            String amount =
documentChange.getDocument().getString("Amount");

                            BigDecimal decimalAmount = new
BigDecimal(amount);
                            receive.amount = "-" +
decimalAmount.setScale(2).toString();
                            total = total +
Double.parseDouble(receive.amount);
                        }else{
                            String amount =
documentChange.getDocument().getString("Amount");

                            BigDecimal decimalAmount = new
BigDecimal(amount);
                            receive.amount = "-" +
decimalAmount.setScale(2).toString();
                        }
                    }else{
                        String amount =
documentChange.getDocument().getString("Amount");

                        BigDecimal decimalAmount = new
BigDecimal(amount);
                        receive.amount =
decimalAmount.setScale(2).toString();

                        total = total +
Double.parseDouble(receive.amount);
                    }
                    Date timestamp =
documentChange.getDocument().getDate("timestamp");


                    SimpleDateFormat sdf = new
SimpleDateFormat("dd MMM yyyy, hh:mm a",
Locale.getDefault());
                    receive.date =  sdf.format(timestamp);

                    receiveList.add(receive);

                } else if (documentChange.getType() ==
DocumentChange.Type.MODIFIED) {
                    for (int i = 0; i < receiveList.size();
i++) {
```

```
                        String id =
documentChange.getDocument().getString("id");
                        if
(receiveList.get(i).donationID.equals(id)){
                            receiveList.get(i).status =
documentChange.getDocument().getLong("status").intValue();
                            total = total -
Double.parseDouble(receiveList.get(i).amount);
                            if(receiveList.get(i).status == 1
|| receiveList.get(i).status == 2){
                                total = total +
Double.parseDouble(receiveList.get(i).amount);
                            }
                            break;
                        }
                    }
                }
            }
            Collections.sort(receiveList,(obj1, obj2)->
obj2.date.compareTo(obj1.date));
            receiveAdapter.notifyDataSetChanged();
            TextView txt_total = findViewById(R.id.total);
            String s_total = String.valueOf(total);
            txt_total.setText("RM " + s_total);
        }
    };

    private void sendReceive(){
        sendList.clear();
        firestore.collection("Donation")
                .whereEqualTo("Send",ic)
                .addSnapshotListener(sendListener);

    }

    private final EventListener<QuerySnapshot> sendListener =
(value, error) -> {
        if (error != null){
            return;
        }
        if (value != null) {
            for (DocumentChange documentChange :
value.getDocumentChanges()) {
                if (documentChange.getType() ==
DocumentChange.Type.ADDED) {
                    Receive receive = new Receive();
                    receive.donationID =
documentChange.getDocument().getString("id");
                    receive.toID =
documentChange.getDocument().getString("To");
                    receive.userID =
documentChange.getDocument().getString("Send");
                    String amount =
documentChange.getDocument().getString("Amount");

                    BigDecimal decimalAmount = new
BigDecimal(amount);
                    receive.amount =
decimalAmount.setScale(2).toString();
                    Date timestamp =
documentChange.getDocument().getDate("timestamp");
```

```
                    SimpleDateFormat sdf = new
SimpleDateFormat("dd MMM yyyy, hh:mm a",
Locale.getDefault());
                    receive.date =  sdf.format(timestamp);

                    sendList.add(receive);

                }
            }
            Collections.sort(sendList,(obj1, obj2)->
obj2.date.compareTo(obj1.date));
            sendAdapter.notifyDataSetChanged();
        }
    };
}
```

6.3 App Flow



https://lucid.app/lucidchart/5cb19953-a537-4c89-8ed0-39abcbeab3ad/edit?viewport_loc=-1494%2C404%2C4806%2C1955%2C0_0&invitationId=inv_5b15a585-6ff6-4e0f-9dbb-914200052187

7.0 Testing

In the testing phase, we implemented a comprehensive strategy to rigorously validate the fulfilment of both functional and non-functional requirements. Our approach encompassed the following key methodologies:

1) **Unit Testing**

   Each individual component and function of the application underwent meticulous unit testing. This focused approach allowed us to isolate and assess the functionality of specific units, ensuring they performed as intended in isolation.

2) **Integration Testing**

   Following successful unit tests, we proceeded with integration testing, evaluating the seamless interaction and collaboration between different modules. This phase ensured that integrated components functioned harmoniously and that potential issues arising from their combination were identified and addressed.

7.1 Functional Requirement

The first approach we adopted involves functional testing , ensuring that each feature of the application aligns with the expected behaviour. Certain tests involve negative testing to ensure that the application can gracefully handle invalid input or unexpected user behaviour.

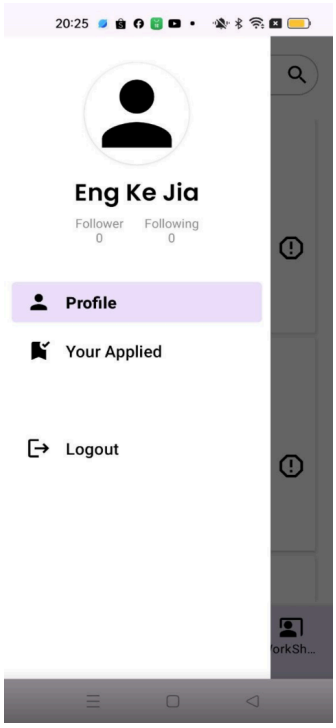
7.1.1 General

1) **User Authentication Module**

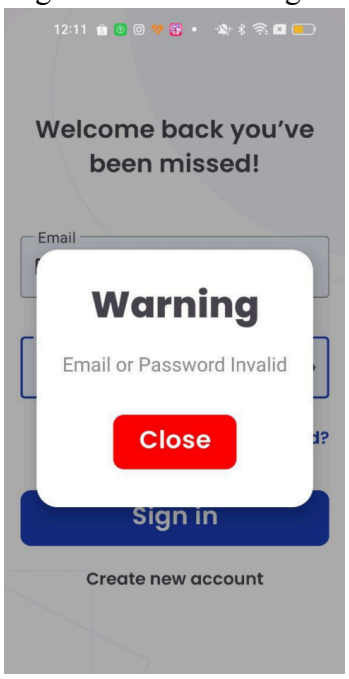
a. Log in

i. Positive Testing

Input	Expected Output	Actual Output
Log in with correct credentials Users will be able to log in to their own accounts.	Users will be able to log in to their own accounts.	Users will be able to log in to their own accounts.

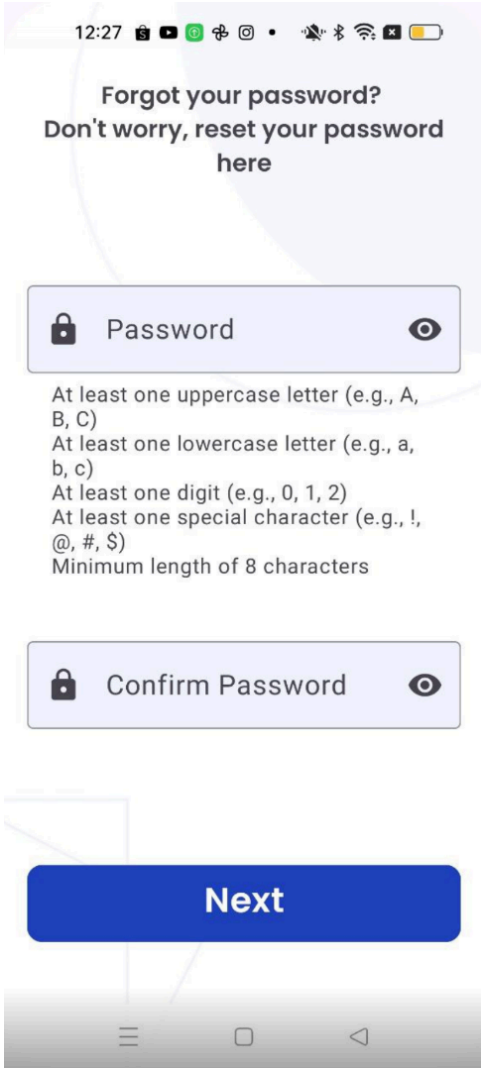
ii. Negative Testing

Input	Expected Output	Actual Output
Log in with wrong credentials. 	A warning message prompts out and the user will remain in the 'Log In' page.	A warning message prompts out and the user will remain in the 'Log In' page.

b. Log out

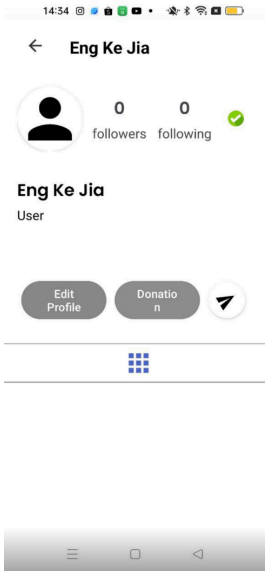
Input	Expected Output	Actual Output
User press on the Log out button	Users will be directed to the 'LogIn' page.	Users will be directed to the 'LogIn' page.

c. Forget Password

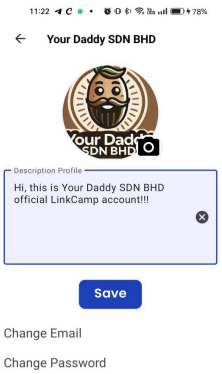
Input	Expected Output	Actual Output
Users press on the 'forget the password' and enter their credentials. 	Users will receive otp from email and reset password.	Users will receive otp from email and reset password.

2) User Profile Module

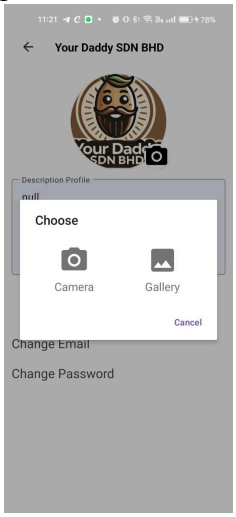
a. Profile page

Input	Expected Output	Actual Output
Users press on 'Profile Page' at the side navigation bar. 	Users will be directed to the 'Profile Page'	Users will be directed to the 'Profile Page'

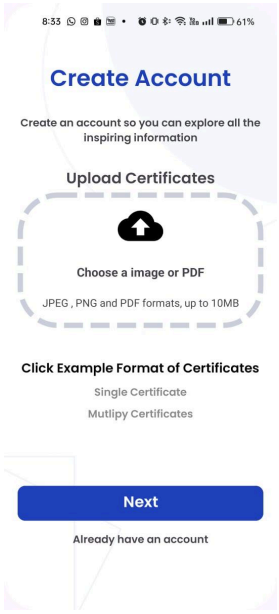
b. Edit profile

Input	Expected Output	Actual Output
Users edit profile description 	New profile description will be displayed	New profile description will be displayed

c. Upload profile pictures

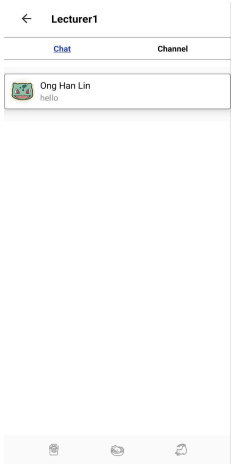
Input	Expected Output	Actual Output
Users upload their new profile picture 	New profile pictures will be displayed	New profile pictures will be displayed

d. Upload education certificates

Input	Expected Output	Actual Output
Users upload their education certificates 	New education certificates will be displayed	New education certificates will be displayed

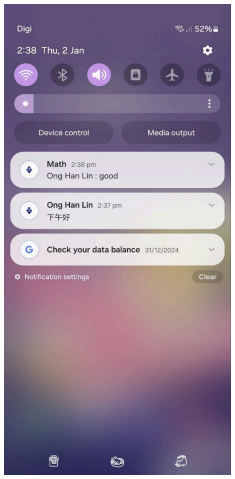
3) Messaging Module

a. Message details

Input	Expected Output	Actual Output
The user sends a message to another user. 	The message contains details such as sending time.	The message contains details such as sending time.

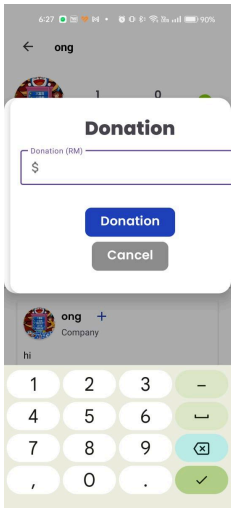
4) Notification Module

a. Receive announcement notification

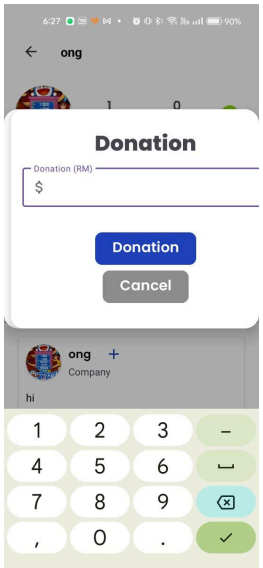
Input	Expected Output	Actual Output
After users have enabled the notification function, they will be receiving notifications from LinkCamp app. 	Notification will be displayed on the notification panel.	Notification will be displayed on the notification panel.

5) Payment Module

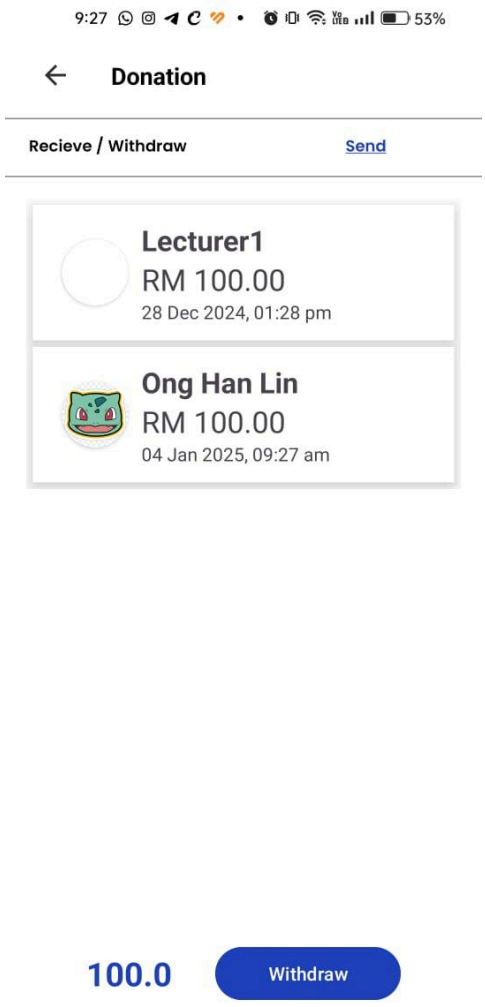
a. Make donation

Input	Expected Output	Actual Output
Users can make donations to another user. 	Other users will receive the donation from the user.	Other users will receive the donation from the user.

b. Receive donation

Input	Expected Output	Actual Output
Other users can make donations to users. 	User will receive the donation from another user.	User will receive the donation from another user.

c. Donation list

Input	Expected Output	Actual Output
Users make donations to another user and receive donations from others. 	Users can view the list of donations made or received.	Users can view the list of donations made or received.

7.1.2 User role

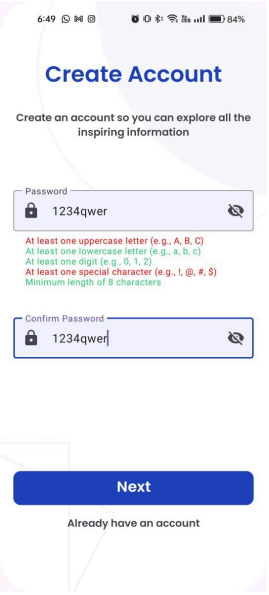
1) User Authentication Module

a. Register

i. Positive Testing

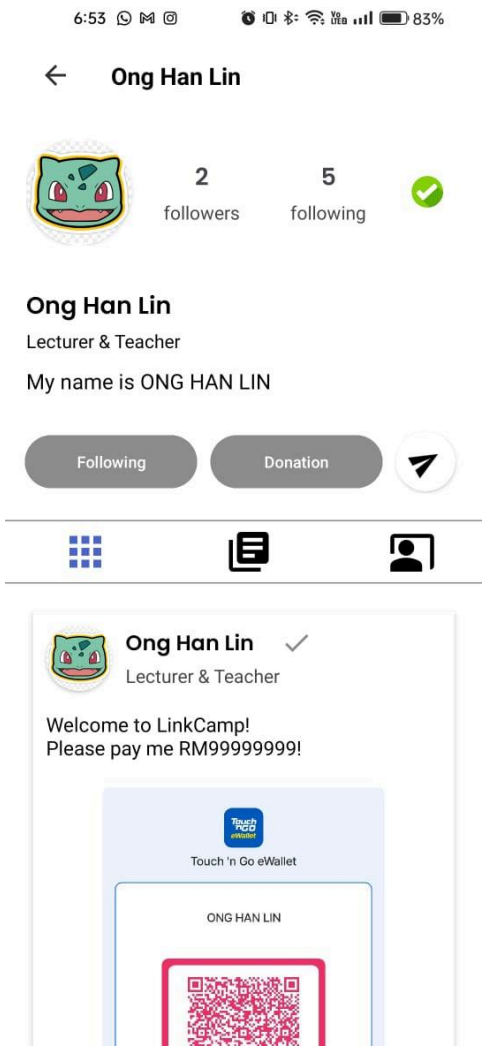
Input	Expected Output	Actual Output
Sign up with an identification number, name, email, password and upload identification card. 	Users will be directed to the "Log In" fragment.	Users will be directed to the "Log In" fragment.

ii. Negative Testing

Input	Expected Output	Actual Output
Passwords do not contain at least 8 characters and include 1 uppercase letter, 1 lowercase and 1 digit. 	Warning icons will appear at the fields that are not complete or incorrect.	Warning icons will appear at the fields that are not complete or incorrect.

2) User Profile Module

a. Review others' profile

Input	Expected Output	Actual Output
User can click and review lecture's and company's profile 	The details of lecturers and company will be displayed	The details of lecturers and company will be displayed

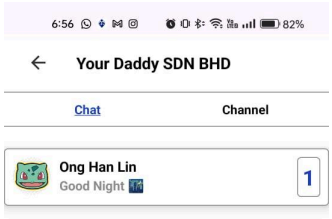
3) Messaging Module

a. Send messages

Input	Expected Output	Actual Output

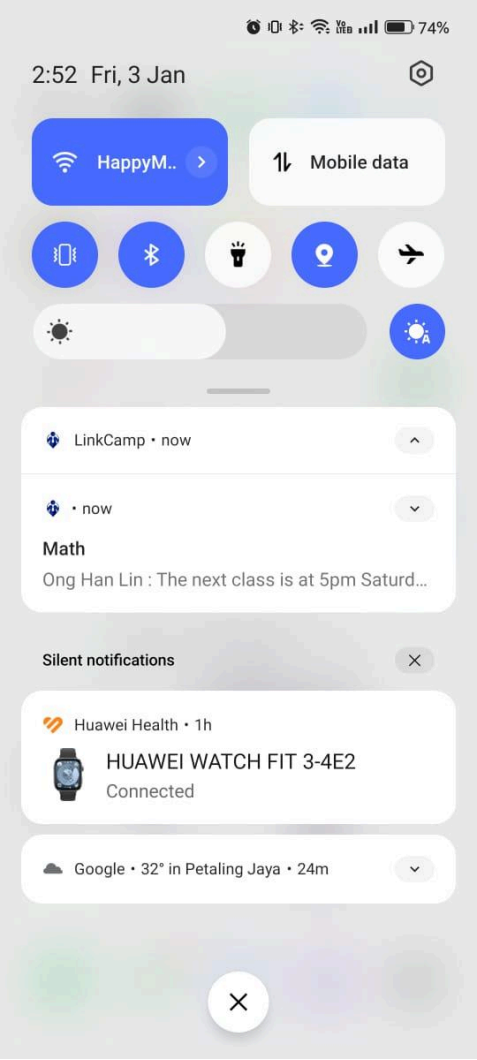
Users can send messages to others. 	The receivers can receive messages instantly.	The receivers can receive messages instantly.

b. Receive messages

Input	Expected Output	Actual Output
Others send messages to the user. 	The user can receive a message instantly.	The user can receive a message instantly.

4) Notification Module

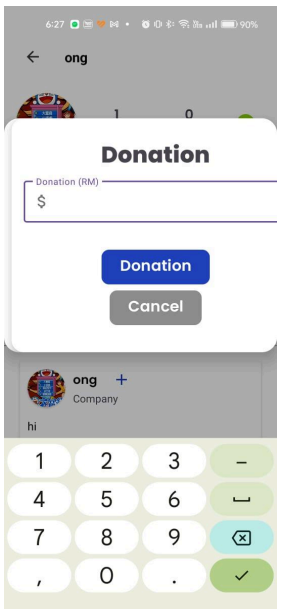
a. Update

Input	Expected Output	Actual Output
Lecture will publish the notification about class participation 	Notification will be displayed on the notification panel.	Notification will be displayed on the notification panel.

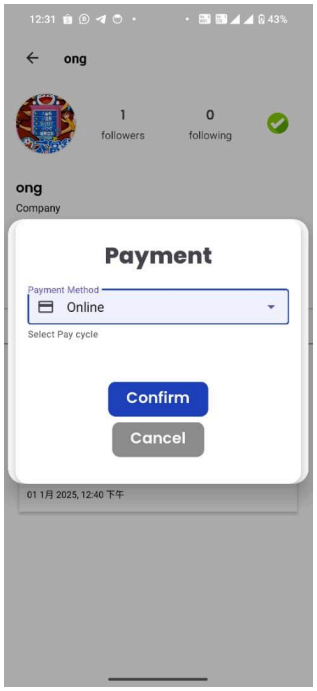
5) Payment Module

a. Donations

Input	Expected Output	Actual Output

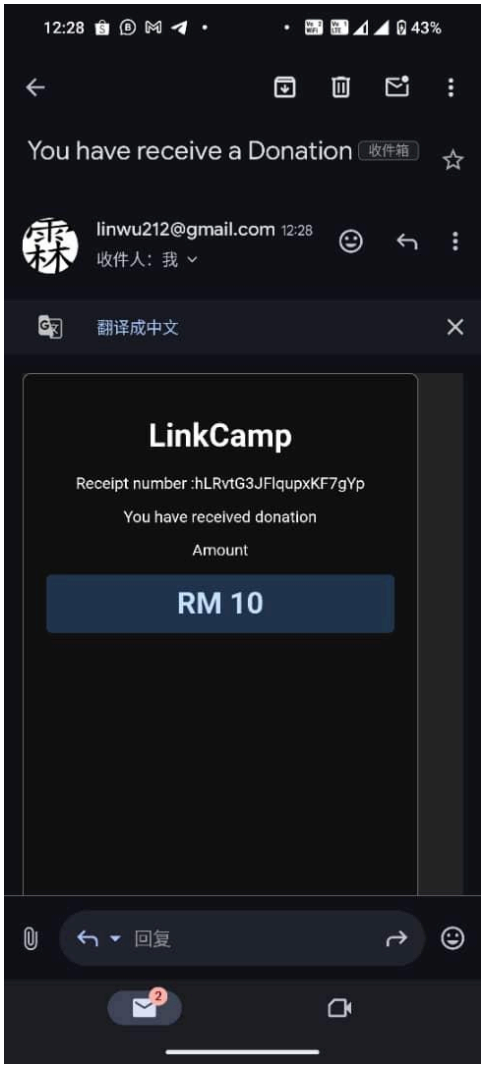
Users can make donations to other users. 	Others will receive the donations from the user.	Others will receive the donations from the user.

b. Multiple payment methods

Input	Expected Output	Actual Output
Users choose their preferred payment method to make purchases. 	User successfully purchases the items.	User successfully purchases the items.

c. Receipt

i. Positive testing

Input	Expected Output	Actual Output
Users successfully make donations to others. 	Receipts of donation are generated.	Receipts of donation are generated.

ii. Negative testing

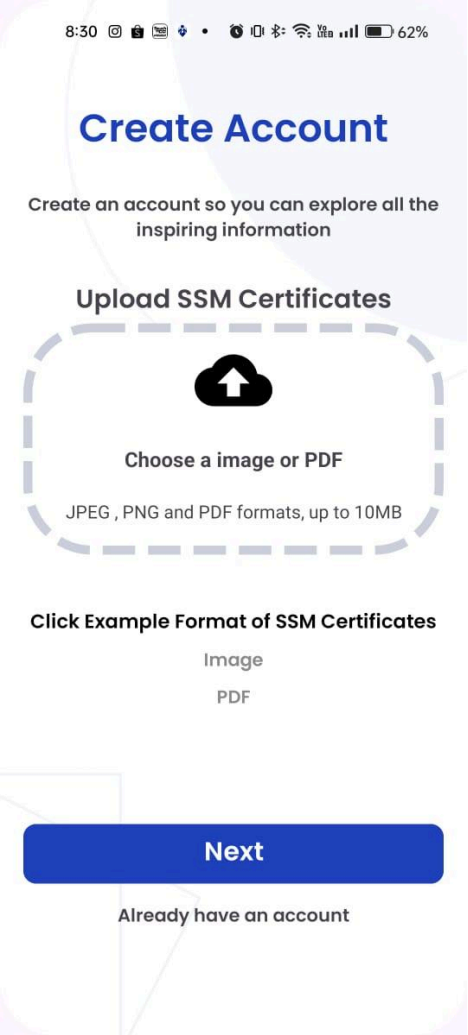
Input	Expected Output	Actual Output
Users unsuccessfully make donations to others.	No receipts of donation are generated.	No receipts of donation are generated.

7.1.3 Company Role

1) User Authentication Module

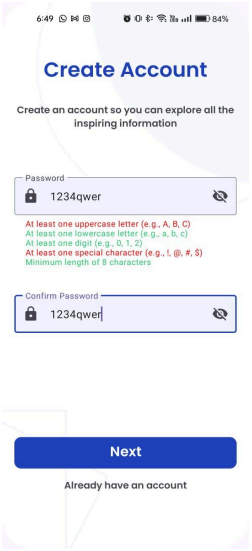
a. Register

i. Positive testing

Input	Expected Output	Actual Output
Sign up with a company registration number, company name, email, password and upload SSM Certification. 	Users will be directed to the "Log In" fragment.	Users will be directed to the "Log In" fragment.

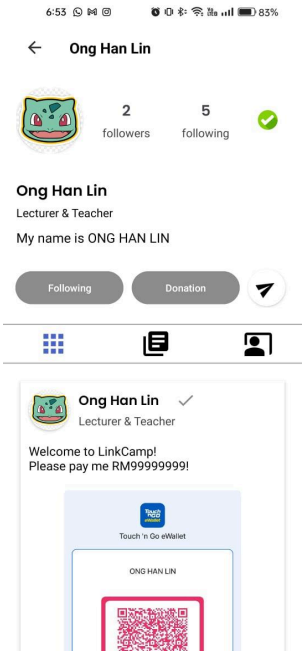
ii. Negative Testing

Input	Expected Output	Actual Output

Passwords do not contain at least 8 characters and include 1 uppercase letter, 1 lowercase and 1 digit. 	Warning icons will appear at the fields that are not complete or incorrect.	Warning icons will appear at the fields that are not complete or incorrect.

2) User Profile Module

a. Review end-user profiles

Input	Expected Output	Actual Output
Company users click on end-user profiles. 	Company users can access and review end-user profiles.	Company users can access and review end-user profiles.

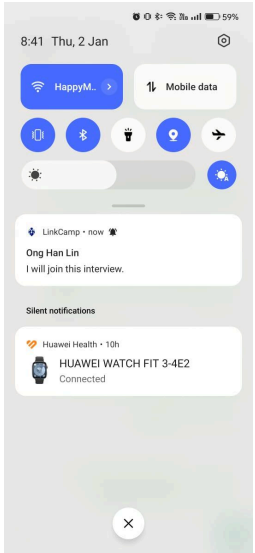
3) Messaging Module

a. Receive message

Input	Expected Output	Actual Output
Users can message company users. 	Company users received messages from the user.	Company users received messages from the user.

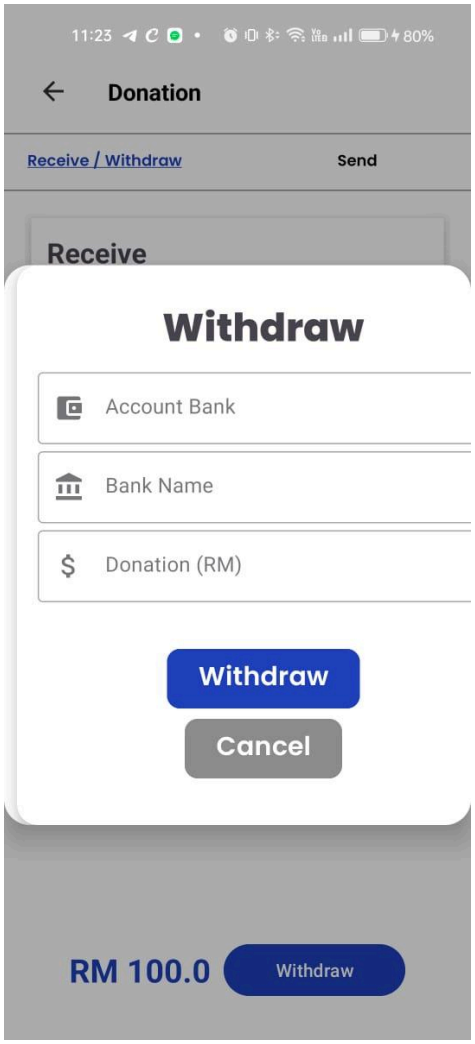
4) Notification Module

a. Receive notification

Input	Expected Output	Actual Output
Company users receive interview confirmation from users. 	Notification will be displayed on the company user's notification panel.	Notification will be displayed on the company user's notification panel.

5) Payment Module

a. Withdraw donation

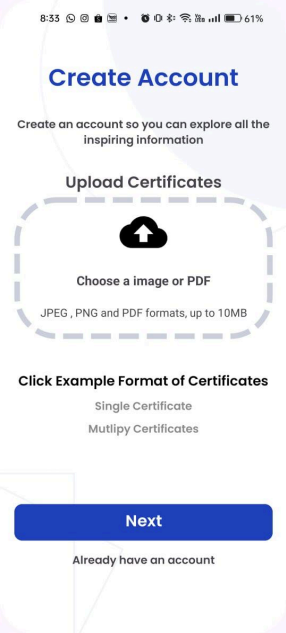
Input	Expected Output	Actual Output
Users can withdraw the donation from other users. 	The donation is successfully withdrawn.	The donation is successfully withdrawn.

7.1.4 - Lecturer Role

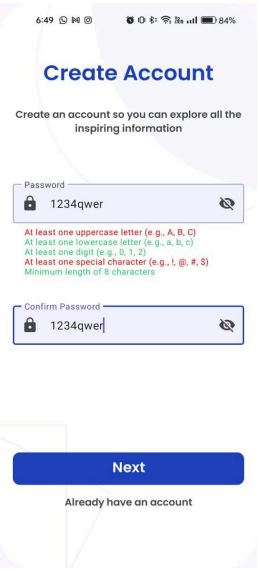
1) User Authentication Module

a. Register

i. Positive testing

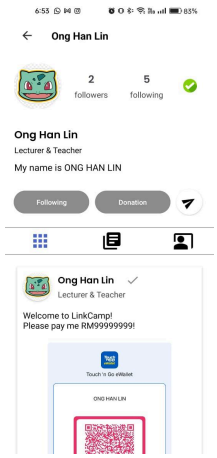
Input	Expected Output	Actual Output
Sign up with an identification number, name, email, password and upload education certificates. 	Users will be directed to the "Log In" fragment.	Users will be directed to the "Log In" fragment.

ii. Negative Testing

Input	Expected Output	Actual Output
Passwords do not contain at least 8 characters and include 1 uppercase letter, 1 lowercase and 1 digit. 	Warning icons will appear at the fields that are not complete or incorrect.	Warning icons will appear at the fields that are not complete or incorrect.

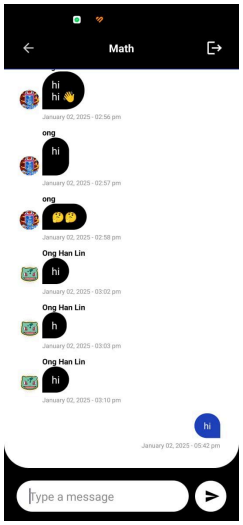
2) User Profile Module

a. Review user profiles

Input	Expected Output	Actual Output
Lecturers can click on the end user's profile. 	Lecturers can review and access end-user profiles.	Lecturers can review and access end-user profiles.

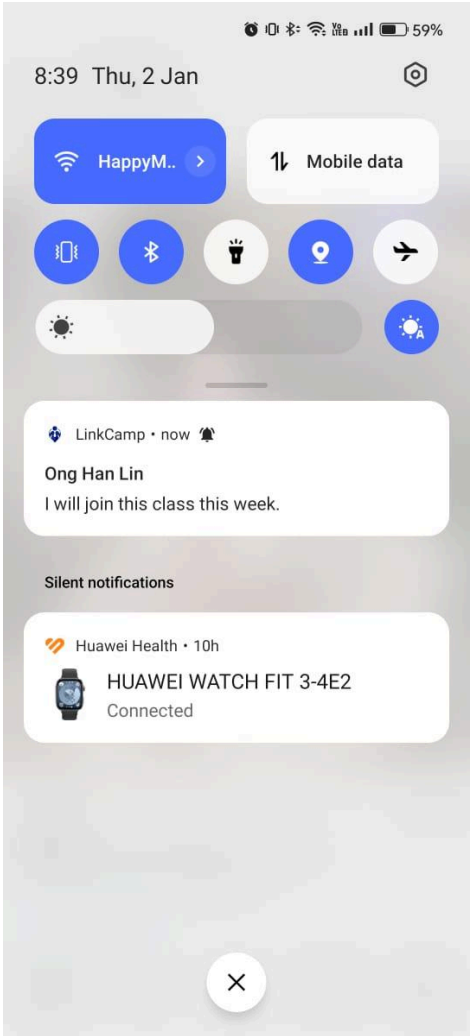
3) Messaging Module

a. Broadcast channels

Input	Expected Output	Actual Output
Lecturers create broadcast channels and facilitate classroom announcements. 	End-users who join the classroom will receive the message.	End-users who join the classroom will receive the message.

4) Notification Module

a. Receive notification

Input	Expected Output	Actual Output
Lecturer users can receive confirmation from other users. 	Lecturer will receive the notification when students join their classes.	Lecturer will receive the notification when students join their classes.

7.2 Non-functional Requirement

In order to ensure that all the non-functional requirements are achieved, we also conducted testing on each non-functional requirement.

1) Performance

a. Fast and responsive experience

Input	Expected Output	Actual Output
Enter the main screen under normal network conditions.	The main screen loading in under 2 seconds under normal network conditions.	The main screen loading in under 2 seconds under normal network conditions.

b. Handle concurrent users

Input	Expected Output	Actual Output
Three users are using the app concurrently.	The app handles concurrent users without compromising functionality.	The app handles concurrent users without compromising functionality.

2) Security

a. Database access restrictions testing

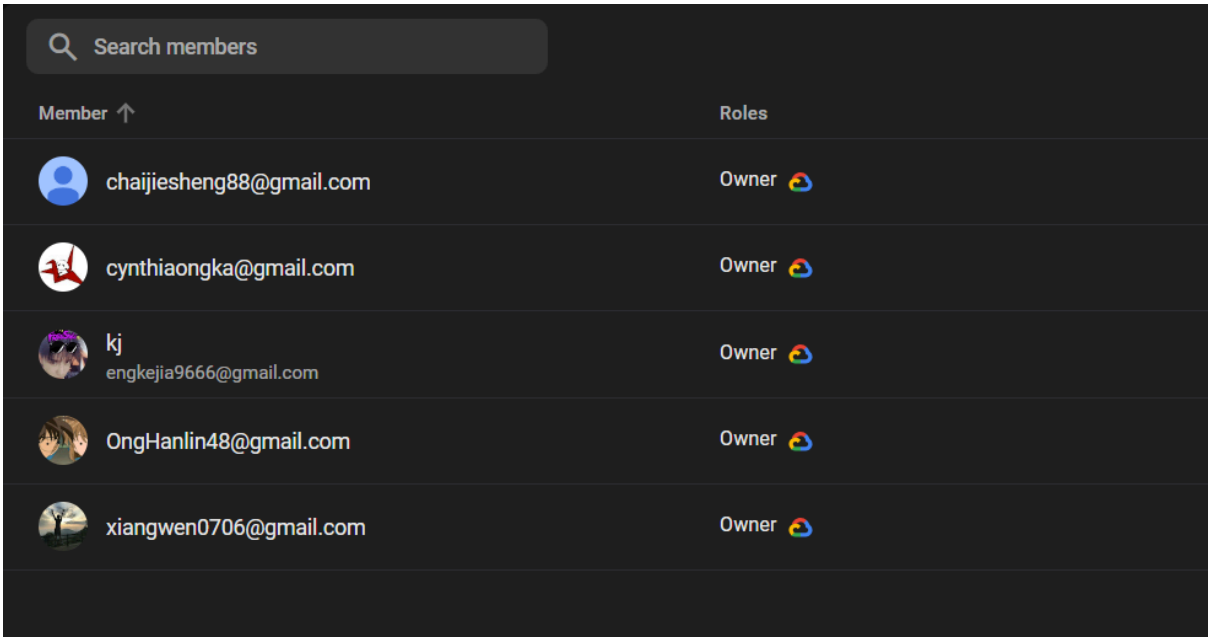
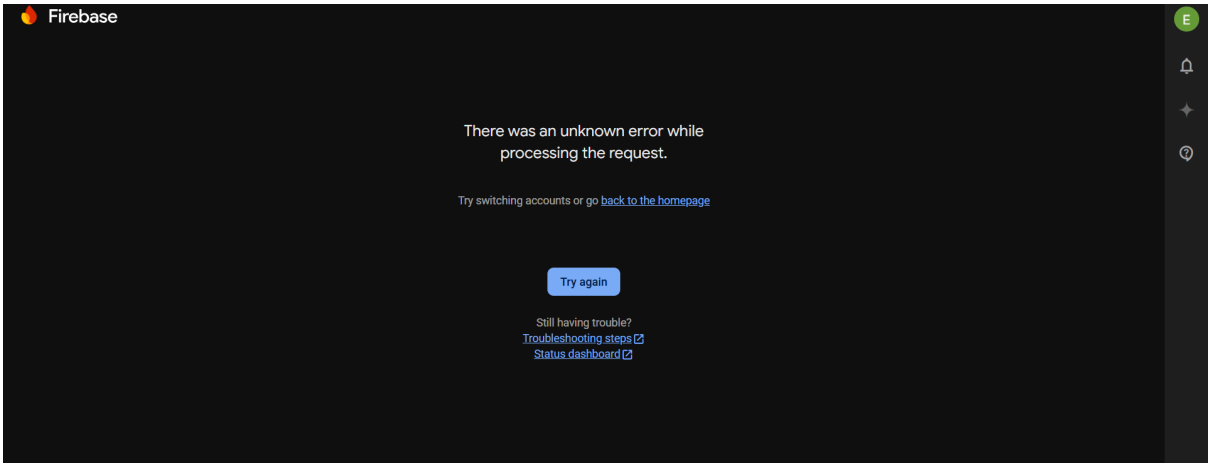


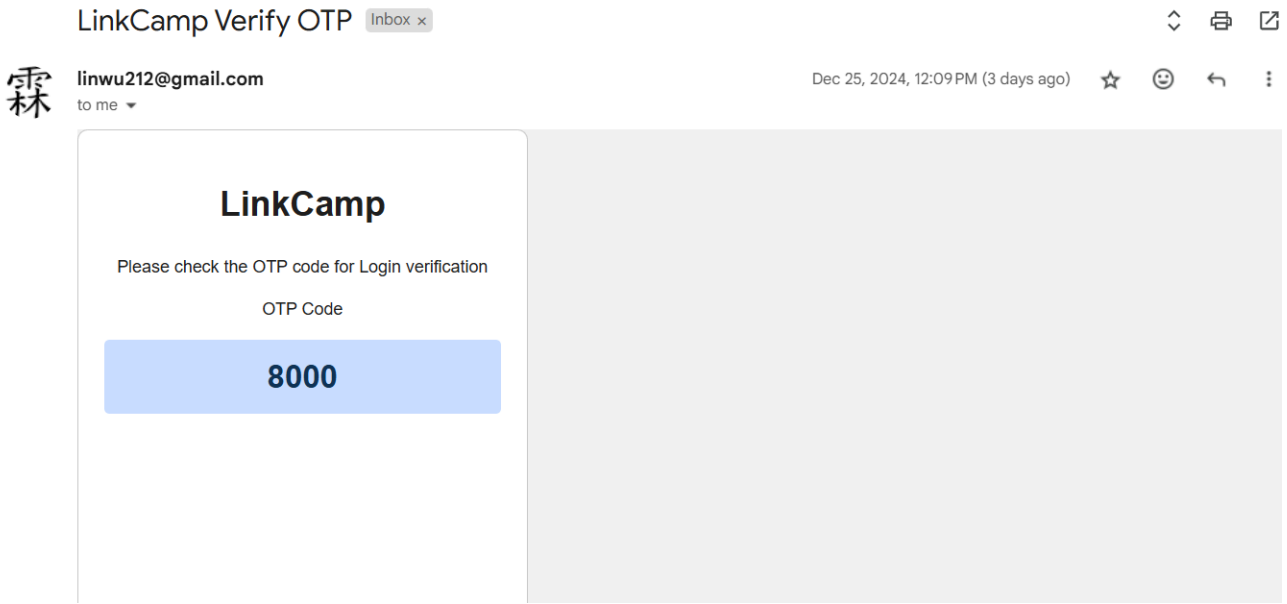
Figure shows the list of users that have access to the Firebase project which consists only of our team members. To ensure that no other unauthorized party can access the database, we conduct the following test.

Input	Expected Output	Actual Output
Access the Firebase project using another unauthorized account.	Unable to access the database.	Unable to access the database.

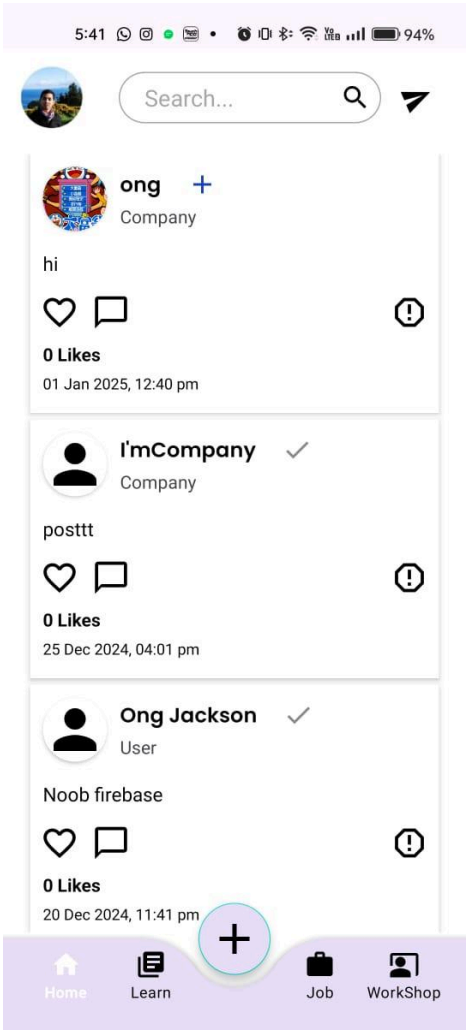


b. Two-factor authentication

When a user requests login access, it will prompt the user to enter an OTP. The authentication service sends the OTP as a token to the user's email.

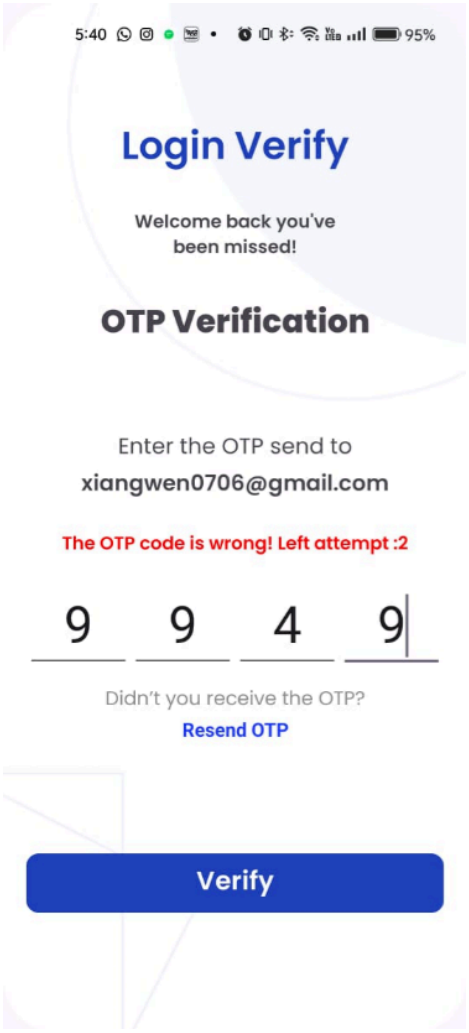


i. Positive Testing

Input	Expected Output	Actual Output
Users enter the correct OTP that was sent to their email. 	Users will be able to log in to their own accounts.	Users will be able to log in to their own accounts.

ii. Negative Testing

Input	Expected Output	Actual Output
Users enter the wrong OTP.	Users unable to login their own accounts.	Users unable to login their own accounts.

 5:40 95% Login Verify Welcome back you've been missed! OTP Verification Enter the OTP send to xiangwen0706@gmail.com The OTP code is wrong! Left attempt :2 9 9 4 9 Didn't you receive the OTP? Resend OTP Verify		

8.0 References

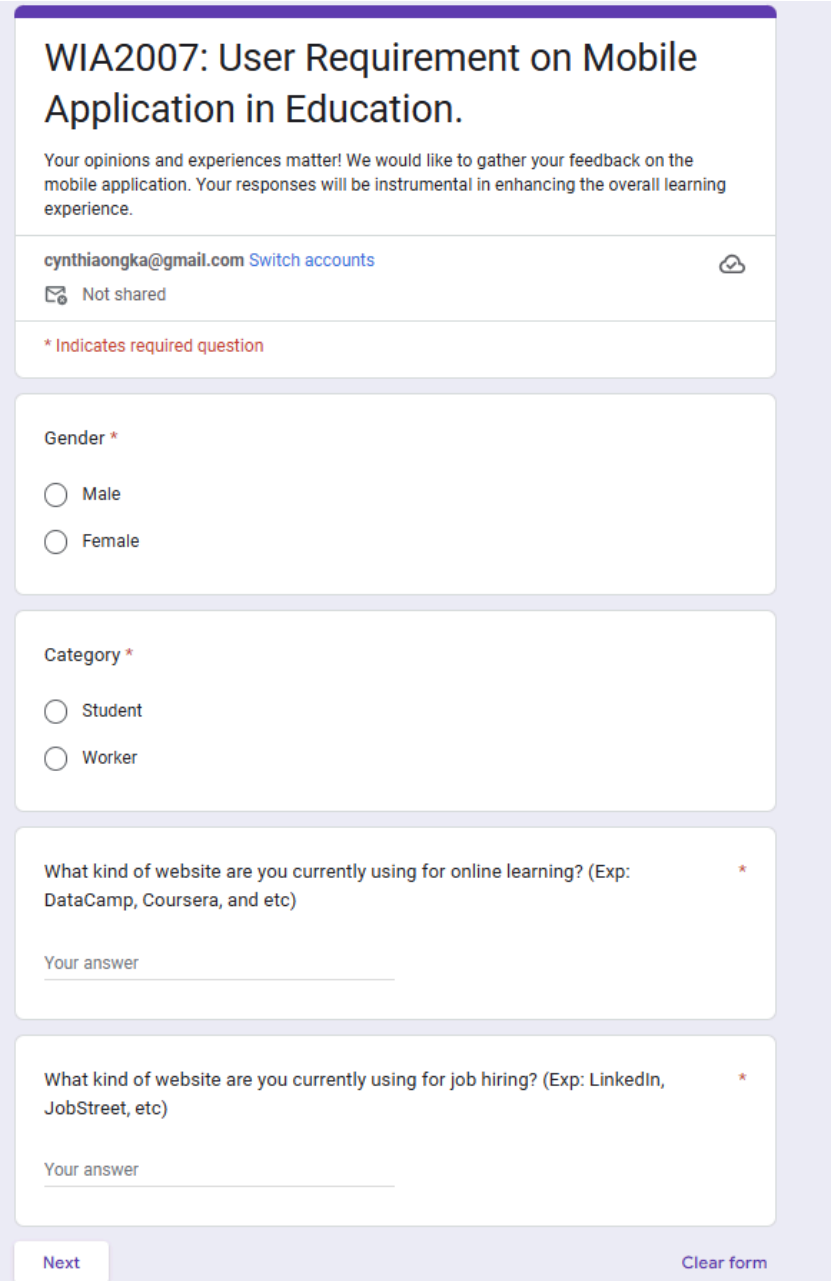
The following sources informed the development of the project:

1. *Strengths and Weaknesses of Online Learning | University of Illinois Springfield*. (n.d.). https://www.uis.edu/ion/resources/tutorials/overview/strengths-weaknesses
2. *Pros and Cons of top job search Websites | JohnLeonard*. (n.d.). https://johnleonard.com/blog/pros-and-cons-of-top-job-search-websites/
3. *Sustainable Development*. (n.d.). https://sdgs.un.org/

9.0 Appendix

9.1 Google Form

The appendix includes a Google Form used during the project for data collection and feedback purposes



WIA2007: User Requirement on Mobile Application in Education.

Your opinions and experiences matter! We would like to gather your feedback on the mobile application. Your responses will be instrumental in enhancing the overall learning experience.

cynthiaongka@gmail.com Switch accounts

Not shared

* Indicates required question

Gender *

Male

Female

Category *

Student

Worker

What kind of website are you currently using for online learning? (Exp: DataCamp, Coursera, and etc) *

Your answer

What kind of website are you currently using for job hiring? (Exp: LinkedIn, JobStreet, etc) *

Your answer

Next

Clear form

User Requirement

We are planning to create a mobile application. We want to understand your requirement mobile application. Your honest responses will help us identify areas for improvement.

What kind of log-in method you are prefer? *

- ◯ Password
- ◯ Link with Google Account
- ◯ QR code
- ◯ Other: _____

What kind of feature do you need the most to improve your learning experience? *

- ☐ Communication channels with lecture
- ☐ Rate class material function
- ☐ Quiz system
- ☐ Virtual lab simulation
- ☐ Timetable
- ☐ Free learning material

What is the weakness of job hiring system you are facing? *

- ☐ Long and inefficient hiring process
- ☐ Skill mismatch
- ☐ Geographical limitations and lack of flecibility
- ☐ Lack of Emphasis on soft skill
- ☐ Ineffective matching of jobs and candidates
- ☐ Free Training

Back Next Clear form

User Requirement Evaluation

We are excited to introduce a idea of mobile application that combines the functions of online learning and job hiring. Please share your thoughts on the proposed changes.

Rate the importance of the login system. *

	1	2	3	4	5	
Not Important	○	○	○	○	○	Very Important

We are planning to develop a new mobile application for your skills enhancement and job hiring. Do you think this is a positive step? *

- ○ Yes
- ○ No
- ○ Maybe

How satisfied are you with your current online learning platform and job hiring platform? *

	1	2	3	4	5	
Very Unsatisfied	○	○	○	○	○	Very Satisfied

Back Next Clear form

The End.

Thank you for taking the time to share your insights. Your feedback is invaluable in shaping a better learning and job hiring environment for everyone!

Back **Submit** Clear form

9.2 Proof of Contribution