DEVELOPMENT OF MOBILE APPLICATION: ENHANCED SOCIAL INTERACTION AND CONTENT DISCOVERY FOR FICT STUDENTS

By
Tong Chee Qing

A REPORT
SUBMITTED TO
Universiti Tunku Abdul Rahman
in partial fulfillment of the requirements
for the degree of
BACHELOR OF INFORMATION SYSTEMS (HONOURS)
INFORMATION SYSTEMS ENGINEERING
Faculty of Information and Communication Technology
(Kampar Campus)

FEBRUARY 2026

COPYRIGHT STATEMENT

This Final Year Project report is submitted in partial fulfillment of the requirements for the degree of Bachelor of Information Systems (Honours) Information Systems Engineering at Universiti Tunku Abdul Rahman (UTAR). This Final Year Project proposal represents the work of the author, except where due acknowledgment has been made in the text. No part of this Final Year Project proposal may be reproduced, stored, or transmitted in any form or by any means, whether electronic

ACKNOWLEDGEMENTS

I would like to express my sincere thanks and appreciation to my supervisors, Mr. Phan Koo Yuen who has given me this bright opportunity to engage in this project. Guidance and suggestion give are much appreciated, a million thanks to you

I would like to express my profound gratitude to my family that actively supported me and encouraged me during the course. Finally, I would also like to acknowledge my deepest gratitude to my friends that have given me invaluable emotional support and valuable advice and gave me a chance to achieve this project in the frame of time.

ABSTRACT

This project deals with a growing need in educational technology – integrated mobile campus apps. Mobile educational apps bring social platforms, and personalized content delivery made to improve university life. As mobile technology use grows in education, students face big problems like social platforms, and broken access to academic resources. There are three main problems for this project – weak social interaction, and poor content discovery. Most campus apps do not give a full mix social tools, and personalized content, so there is still a big gap in making one complete mobile solution for university life. Research and literature reviews were done to study mobile campus app development, social systems, and content personalization methods. The reviews of current tools like recommendation methods, cross-platform frameworks, and login systems are in the literature review section. After looking at the good and bad points of 19 academic papers and current products, a full picture of the challenges and chances for mobile educational apps became clear. The solution is to make one mobile app using cross-platform tools to help FICT students with social links and finding academic resources. The final app will use the Flutter cross-platform framework, Node.js REST API with JWT login for safe user control, MySQL for storing all data together, and machine learning for personalized content suggestions. This will make one platform that connects, involves, and informs the FICT student group.

Area of Study: Mobile Application Development, Educational Technology

Keywords: Mobile Application Development, Educational Technology, Social Interaction Platforms, Content Discovery, Flutter Framework, Personalized Learning

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

API	Application Programming Interface
EdTech	Educational Technology
LMS	Learning Management System

Chapter 1

Introduction

The quick growth of mobile technologies has changed how people interact with their surroundings and offers the chance to make everyday life better with smart apps. Some of the problems that students have in the learning setting are getting along with other students and teachers and getting to learning materials and other resources outside of school. Based on these problems, we need a central platform that can help make real social interactions easier and give university communities a more personalised way to find material. This project will focus on making a complete mobile app that can solve these unique issues that the FICT students are having. These days, community applications mostly concentrate on the social aspects of things, giving education little to no consideration. On the other hand, educational applications are a fragmented experience that doesn't satisfy the demands of every student and typically don't provide social interaction. According to a survey conducted on campus, students believe that the lack of personalised content search tools designed specifically for university settings makes it difficult for them to locate academic and extracurricular materials fast. The goal of this project is to make campus life more connected and useful for students by adding personalised content delivery systems and real-time contact services. These issues are very important according to academic research. For example, Ansari and Khan [1] found that social media sites have a big impact on collaborative learning among 360 college students, and Romero-Rodriguez et al. [2] looked at 1,544 college professors and found that mobile learning systems can be used to improve content discovery and delivery in higher education. With this kind of study, it's clear that certain mobile solutions are needed that allow for social interaction as well as smart content discovery systems and are only available to college students.

1.1 Problem Statement

People and schools are becoming more diverse, which means there is more information available. However, there aren't any good apps to collect this information, which leads to "Information Cocoons." Researchers have found that college students are the most likely to be affected by information fragmentation. When college students consume information in pieces, it becomes incomplete and useless, and they have to spend a lot of time putting it all together [4]. The project will make a group mobile app for UTAR FICT students that will help with the issues listed below:

1. Not enough information was gathered

The majority of current community apps focus on social features, with almost none geared toward the education field. This means that community software that isn't directly related to education exists. Because of this, the collection of information about things like schooling is not well organised and is not full. According to Pedro et al. [4], this breakdown between social media use and academic applications was found in 70% of computer lab sessions. Students were found to be multitasking, and they were unable to combine social and educational functions effectively while using the computer lab. Scholarly literature has shown that this kind of separation means there isn't much practical study on how these tools are actually used to help teach and learn, and there aren't many descriptions of how college students use smartphones and social media.

2. The lack of a basic integrated platform

Other community mobile app that is based on educational resources that is similarly biased, but this one can't find messages like news and events happening on campus because of how complicated the working interface, messages, and other content are. This thorough study of educational technology systems by Vlachogianni and Tselios [5] did not do very well in terms of usability, as shown by the mean score of 72.75 out of 100 on validated PSSUQ measures. According to their research, the internet platforms are probably the most popular piece of educational technology. However, users are not happy with how easy it is to use them. Also, Alsanousi et al. [6] talked about some usability issues in educational apps, like issues with AI-related functions, speed, bias, explanation, and features that don't work, all of which have a big impact on how satisfied and successful users are with the apps.

3. The absence of personalized content discovery systems in university environments

There is no such thing as useful software that could help students find more personalized content that is made just for college students. This could make it harder for students to find the right academic tools, school events, and educational content. According to Oliveira et al. [3], there was a big gap between what students needed and how much they actually used mobile apps. Each student used the apps 11,177 times during theoretical classes, which is a lot more than how much they said they used them, which shows that the system for delivering content is not working. Multiple studies have shown that students who use their phones during talks take 22–59% longer to finish reading assignments than students who don't use their phones during classes.

1.2 Motivation

Students often have trouble trying to find appropriate school services and manage their academic calendars. In the age of digitalisation, the needs could not be addressed by standard methods like email notifications and bulletin boards. The absence of a centralised, user-friendly platform for social interaction and information discovery limits student engagement and productivity. The proposed mobile application would revolutionise the students' experience by incorporating cutting-edge technology, including tailored content suggestions and captivating social elements. It will offer a single point of access to campus services, recreational and academic activities, and social networking. In addition to improving campus operations, this integration will foster a feeling of camaraderie among FICT students. The application's development has been inspired by the goal of making the campus closer together, creative, innovative, and engaging, which ends up resulting in FICT students having a more enjoyable academic experience.

1.3 Project Scope

The project's purpose is to develop a mobile app to help FICT students manage campus life. The software should perform 30% better than what kids are already using, such as various social media programs. The program will help them with three tasks: talking with one another and exchanging information. The project will provide a complete system for supporting students, meeting other students more readily, and obtaining school information in a single application.

The former is the creation of an intricate user profile and customisation base, allowing FICT students to tailor their experience based on academic and social interests. During the onboarding process, the program allows students to create extensive profiles that contain personal information as well as a preference for topics such as assignment help, campus activities, and course discussions. It is a preference management system that serves as the foundation for personal content recommendation. The backend uses a weighted recommendation algorithm that examines individual activity patterns such as post views and interaction histories. The technology constantly selects the user's For You feed to highlight the most relevant academic information and debates within the FICT community.

The second purpose is to provide an environment in which FICT students may create pieces, discuss issues, and find things they appreciate. The show will also include a forum where students may discuss campus events, school-related topics, and everyday living on campus. The postings will be divided into categories and marked so that students may easily discover what they want to read. Students will be alerted via the system when new posts or comments are made on topics to which they have subscribed.

Last goal is to implement chat tools that will allow students to communicate with one another instantaneously. The app will contain private messages for communicating with a single individual as well as group communications for use in school projects or with friends. To keep students informed, they will receive immediate messaging as new communications arrive. Enrolment in classes, grade verification, and payment will not be included in this project because the institution currently has procedures in place to handle these tasks.

1.4 Research Objectives

This initiative aims to accomplish a number of goals, including:

Objective 1: To implement a personalised method for finding and recommending material.

Sub objectives:

1. Create a smart content filtering and search engine that follows privacy rules and lets students find posts, resources, and information based on their past activity.
2. Make a smart search and content filtering engine that lets students find posts, resources, and information based on their course, interests, and past participation experience while still protecting their privacy.
3. Set up a system for managing user preferences that lets students change how they find information based on their hobbies. They can be interested in certain topics and get personalised content alerts when something fits their intellectual and social interests.

Objective 2: Create a community where FICT students may share, discuss, and find interesting posts.

Sub Objectives:

1. Set up a community forum where FICT pupils can posts, comments, and have discussions about problems pertaining to school and shared interests.
2. Implement an easy-to-use filters and search post categorization and tagging system that allows users to find posts of their interest with ease.
3. Introduce more sophisticated content interaction functions, like reposting and bookmark, which enable students to redistribute valuable information and manage their own academic resources in the community.

Objective 3: T**o establish social connection by communication integration into the system.**

Sub Objectives:

1. To make it easier for students to talk to each other, add a real-time messaging tool that lets them do so alone or with a group.
2. Provide a group chat function that may be used for academic discussions, project collaboration, and campus socialising.

3. Introduce a notification system that will inform users about new information and updates in order to keep communication on time.

1.5 Background Information

Since the first smartphone was introduced in 2007, the number of mobile apps has grown significantly. These days, mobile applications may be utilised for almost everything, including work, education, shopping, and pleasure. Around 2010, universities started using mobile technology by developing basic applications that listed their student directory and basic campus information. These early apps, however, were rudimentary and performed a single function. To do numerous duties, like engaging with peers, obtaining campus news, or using study resources, the students had to use several programs. Software that is compatible with tablets and smartphones is referred to as mobile apps. They take use of capabilities including the capacity to access the internet to rapidly get updates, the ability to use user authentication to secure personal information, and the ability to use push notifications as fast communications.

For more than 50 years, educational technology, or EdTech, has been on the rise. In the 1970s, it was limited to computer programs; in the 1990s, it progressed to online learning platforms. In the 2010s, as smartphones gained popularity, the usage of instructional technology shifted to mobile devices. Another important notion is cross-platform programming, which involves creating a single software that operates on both Android and iPhones. Another similar concept is the Application Programming Interface (API). It allows computer programs to interact with one another. In addition, a database API is used to store and retrieve student data for the apps, as well as share postings and messages. Machine learning is another significant technology that enables computers to learn and provide intelligent recommendations. For instance, it may propose a discussion subject aligned with a student's interests or provide pertinent study materials according to their registered courses. Campus administration systems have progressed from paper-based to digital formats throughout time. Universities use learning management systems (LMS), such as Moodle and Blackboard, to provide online courses.

They use the student information system for both admissions and grading. They rely on several websites for information about activities and happenings on campus.

Nevertheless, these systems typically function independently and create what experts call "information silos," where data is segregated and difficult to access in a comprehensive way. In order to keep students connected and informed about academic activities, real-time communication and notifications are essential in today's educational system. Education is an online community where students may engage with one another work together on projects and learning groups on social media.

Information discovery systems employ algorithms based on user behaviour and interests to provide options for providing information that is relevant to the user, much as Netflix suggests movies and Amazon suggests particular goods. Many essential components make up the technological core of today's mobile campus applications. Lots of student, course, and campus data are stored and organised using database management systems like MySQL so that applications may readily access them. Additionally, by restricting access to certain services to those who are authorised, authentication techniques safeguard student privacy.

Real-time (instantaneous) user communication via real-time messaging systems is made possible via WebSocket. While offering clients customised content recommendations, machine learning algorithms consider user participation. Push notification services will advise users of fresh information that may be of interest as well as any significant developments.

1.6 Contribution

This initiative contributes to the development of FICT360, a mobile application that is both user-friendly and specialised to enhance social interaction and academic content discovery for Universiti Tunku Abdul Rahman students. The application provides a unified digital ecosystem that bridges the gap between social networking and educational tools, effectively mitigating the issues of information fragmentation and "Information Cocoons" frequently encountered by students. It incorporates a personalized onboarding framework, enabling users to select specific academic and social interests—such as assignment help or course discussions—to tailor their community experience from the initial setup. A significant technical contribution is the data-driven recommendation logic, which utilizes user tag weights derived from active interaction patterns, such as post views and likes, to prioritize relevant content in the "For You" feed. Furthermore, the project implements professional-grade interaction mechanisms, including customized Reposting and Bookmarking systems, allowing students to redistribute and manage valuable academic resources seamlessly within the faculty. By utilizing WebSocket technology, the application establishes a real-time collaboration platform for private and group messaging, facilitating immediate project coordination and strengthening social bonds. This mobile solution also saves significant effort and time for students by centralizing campus updates and study resources into one accessible interface. Finally, the scalable cross-stack architecture using the Flutter framework, Node.js REST API, and MySQL demonstrates a robust foundation for building faculty-specific applications that ensure consistent performance across all mobile devices.

1.7 Report Organization

In the report for Final Year Project 2, there are seven chapters that provide a comprehensive examination of the project's various components. The Introduction (Chapter 1) provides a comprehensive overview, which encompasses background information, project contributions, motivation, project scopes, research objectives, and problem statements. In Chapter 2, the Literature Review, the available systems and applications are thoroughly examined, including popular platforms such as Microsoft Teams, Facebook, GroupMe, Campuswire, Slack, and Telegram. In the future, Chapter 3 will cover the system methodology and approach, which encompasses an overview of the Agile SDLC Methodology, System Architecture Diagram, Use Case Diagram, Use Case Descriptions, and Flowcharts. The fourth chapter is a system design that encompasses the RESTful API endpoint designs, database, and System Block Diagram. The hardware and software setup, system configurations, system operations, and issues and challenges of system implementation will be reviewed in chapter five. The System Evaluation and Discussion, which is the assessment of the mobile application to ascertain its functionality, is further elaborated upon in Chapter 6. Furthermore, the chapter has integrated an objective evaluation and the obstacles encountered by the initiative. The report's conclusion and recommendations for improving the FICT360 mobile application will be presented in the final chapter

Chapter 2 LITERATURE REVIEW

2.1 Reviewing Existing Application

2.1.1 Microsoft Teams

Figure 2.1.1.1 Microsoft Teams Mobile Application

Microsoft Teams is a comprehensive collaboration platform developed by Microsoft Corporation that serves as a workspace for real-time collaboration and communication in educational settings [1], it also a digital space that unites conversations, meetings, files, and apps in a single place; it is built on the foundation of Office 365, which schools can integrate with their existing Office apps and services [2]. The platform allows students to access course materials, participate in video conferences, submit assignments, and participate in both synchronous and asynchronous discussions via an integrated interface.

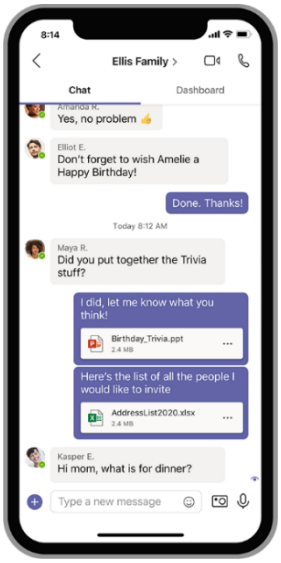


Figure 2.1.1.2 Chat System

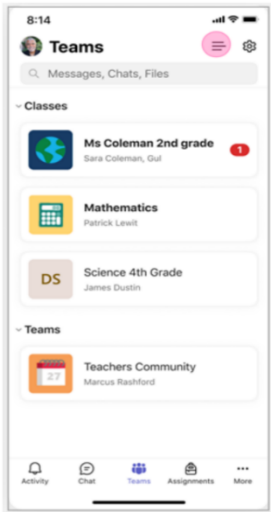


Figure 2.1.1.3 Classroom System

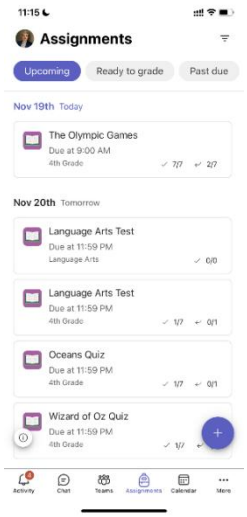


Figure 2.1.1.4 Assignment System

The classroom administration interface, depicted in Figure 2.1.1.3, allows instructors to design virtual classrooms tailored to individual courses, arrange different subjects, and establish a secure learning environment with restricted access to course materials. The assignment management system, seen in Figure 2.1.1.4, enables instructors to assign tasks to students, including due dates and grading standards. It is a comprehensive online learning environment that allows students to effectively manage their academic duties by applying categorise lists of assignments and knowing the duties they must carry out, deadlines, and the manner in which they are due.

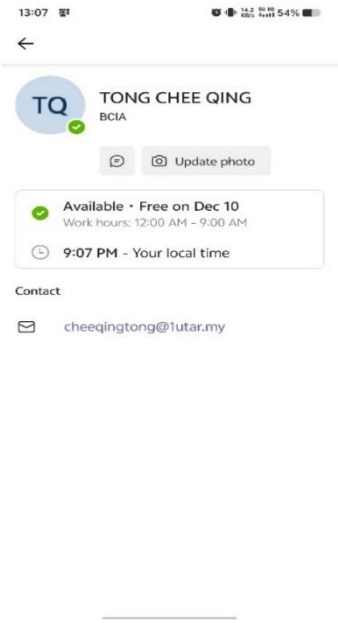


Figure 2.1.1.5 User Profile

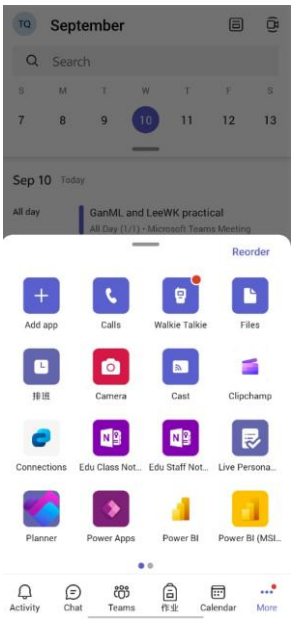


Figure 2.1.1.6 More Tab Features

Figure 2.1.1.5 shows the condensed version of the user profile, which includes the student's name, institutional affiliation (BOA), availability, and contact details, among other important user data. In order to give users a personalised experience and make it easier for them to recognise and access the appropriate user information in the learning environment, the profile allows users to upload images and personalise their avatars. Aside from that, the bottom navigation bar's huge More tab is provided as figure 2.1.1.6 functionality. This category encompasses a diverse array of applications and tools, including calendar integration, document management, communication platforms, productivity apps, and specialised educational software. The integration of a wide range of linked applications allows for the administration of numerous administrative and instructional responsibilities within the same platform interface,

providing the user with a significant amount of autonomy that extends beyond basic communication.

Strengths of Microsoft Teams

Microsoft Teams is education-specific and integrates widely. Teams improve classroom learning and collaboration by fostering skills and confidence with customised and AI-assisted tools that allow students to study independently across all grade levels [1]. Microsoft Teams facilitates group projects by providing dedicated channels for students to interact, exchange ideas, and engage in real-time work. Additionally, students can collaborate on documents using other Microsoft programs like Word, Excel, and PowerPoint without ever leaving the Teams platform [3]. Another outstanding aspect of the platform is its round-the-clock access to data, resources, lesson plans, and recorded lessons. Features like Together Mode enable students to access whatever they need at any time and place.

Weaknesses of Microsoft Teams

Despite its extensive capability, Microsoft Teams has certain shortcomings in learning contexts. There are potential drawbacks, such as the need for students to overcome technological learning curves and the potential for less face-to-face interaction [3]. The internet connection and the necessity of properly installing the equipment to get the best performance are among the technical difficulties. It also takes a long time to educate and be able to utilise the platform effectively in an academic context because it may be highly complex, which might confuse new users, especially those who are not tech-savvy.

Ways to Resolve the Weaknesses of Microsoft Teams

Reducing user learning difficulty while maintaining essential features should be the top priority in order to overcome the limitations found. Customer adoption could increase if the user interface is redesigned to be simpler, more succinct, and easier to use. To reduce performance requirements and remove phone verification, tiny security weaknesses may be inserted, which raises the platform's use restrictions and prevents new users from using it. Providing training and ongoing technical assistance to help with the learning curve, as well as showcasing the real advantages, are some

deployment tactics. In order to make education more inclusive and accessible, infrastructure development should place a high priority on establishing a consistent internet connectivity and provide technical support to resolve device compatibility issues.

2.1.2 **Facebook**

Figure 2.1.2.1 Facebook Mobile Application

Facebook is a huge social networking site that is being used as an instructional tool at higher education institutions in addition to being a personal communication tool. Facebook is the largest active social media site in the world, with 2.989 billion members as of April 2023, according to the literature, which also shows that the number of active users, including students, is expanding rapidly [4]. Facebook is a well-known social networking site with special built-in features that may be used for teaching and learning and provide pedagogical, social, and technical affordances [5]. Discussion forums, IM, and file sharing are just a few of the many built-in capabilities, the platform helps educational institutions develop closed communities for managing courses, having discussions, exchanging learning resources, and improving student-teacher engagement.

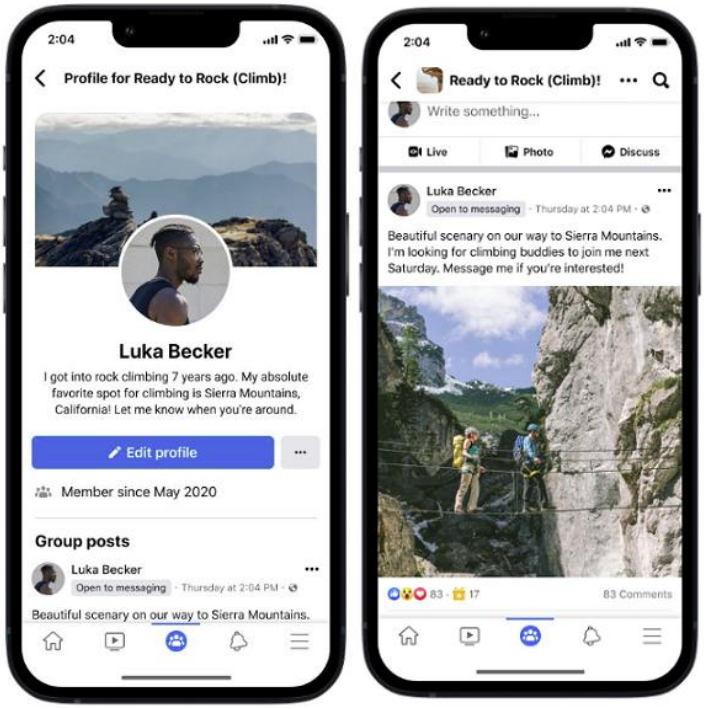


Figure 2.1.2.2 User Profile & Group Post

Facebook's community group function, seen in Figure 2.1.2.2, combines several significant educational aspects into a single interface. The platform's user profile system allows users to identify members based on their level of skill, personal background, and date of membership. Using the group posting function, members may publish photos, text updates, and geographical information to develop community connections. The Facebook group feed, popular and conversation-starting information is given priority in a dynamic learning environment based on engagement rates (shares, reactions and comments). Learning communities can use personal profiles for accountability and trust-building; while posting and content distribution systems encourage peer-to-peer knowledge sharing, question-and-answer exchanges, and multimedia resource distribution for both formal and informal educational interactions.

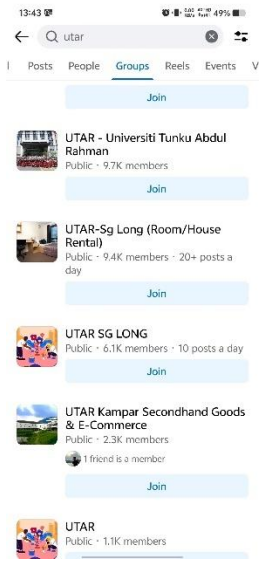


Figure 2.1.2.3 Search & Filter Function

Figure 2.1.2.4 Group System

Strengths of Facebook

Facebook's groups, events, and pages make it a useful tool for creating communities. In an effort to foster engagement and loyalty, companies and organisations may build communities around their brands, and users can follow pages (Figure 2.1.2.4) or join groups (Figure 2.1.2.3) that interest them [6]. It also supports a variety of content formats, including text postings, images, videos, live videos, and narratives. Because of this diversity, individuals and companies are able to customise the material they share to their audience's tastes. Conversely, other social media platforms do not offer the same variety of content posting possibilities, which are mostly focused on text, photos, videos, and links [6].

Weaknesses of Facebook

Among Facebook's disadvantages, privacy concerns rank highest. By its very nature, the platform's economic model, which relies heavily on data collecting and targeted advertising, puts users' privacy at risk. A breach of trust and a major ethical dilemma is the misuse of personal data [7]. Facebook's role in the spread of false information and fake news is another major problem. Sensational and contentious information, whether or whether it is genuine, is more likely to be prioritised by the algorithms employed to maximise engagement on the site [7].

Ways to Resolve the Weaknesses of Facebook

User verification techniques can be used selectively to prevent the emergence of phoney accounts and to guarantee that learning group members' identities are authentic and legitimate in order to stop the spread of false knowledge. More sophisticated privacy settings that allow users to opt out of having their information used to create a data feed and targeted advertising could be introduced in order to allay user concerns about data leakage. This would protect users' privacy without creating needless privacy issues.

2.1.3 GroupMe

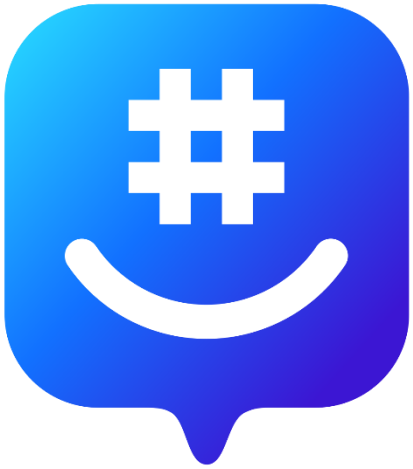


Figure 2.1.3.1 GroupMe Mobile Application

GroupMe is a group-specific mobile instant messaging app that has gained popularity in higher education [8]. With features that allow users to explore campus organisations, find classmates and receive offers, and interact with school-specific communities, it is stated to be the easy method to access all the groups in your life, big and small [8]. According to estimates, more than 70% of US institutions already use GroupMe, and the number of new campus groups created has surged by more than 200%, making it extremely well-liked in the academic community [8]. Using authorized.edu email addresses, the site's GroupMe Campus feature assists college students in finding student organisations, forming deep connections with their classmates, and learning about campus events [9].

Strengths of GroupMe

The simplicity and extensive use of GroupMe on the campuses as a tool in organizing students is also its greatest strength [9]. The platform has introduced GroupMe Campus to enable college students to create meaningful relationships with their classmates, join new student groups, and know what is happening on campus, and has functions such as message reactions, one-on-one and group calling, link previews, and an improved polling experience [9]. The platform is very successful in the creation and management of groups quickly, and therefore, organizations will easily have their activities coordinated, information shared and constant communication maintained [8]. GroupMe Campus offers groups created specifically to schools and allows learners to interact with individuals in their major, but retains privacy since no information or chats are shared with schools or other third parties [8]. The event management feature in the platform enables users to organize events easily and distribute them to group members as well as out of group members to boost campus community building [9].

Weaknesses of GroupMe

Students at Northwestern University who use GroupMe report that they are more frequently added to unwanted groups and that they receive spam messages and bots in their chatrooms. These messages and bots are especially likely to be selling concert tickets that are in high demand, especially in larger groups and under the guise of student accounts. Although it's easy to join a group, its openness poses a risk of spam and illegal access. Average members find GroupMe chatbots annoying, and organisations must manually approve new users and report incidents since it is thought that new students who have never used GroupMe might fall prey to fraud [10].

Ways to Resolve the Weaknesses of GroupMe

New group members should undergo multi-step authentication, including institutional email address verification and manual approval from group administrators. Advanced spam detection algorithms would automatically prohibit bot accounts based on message patterns and other behavioural red flags like recurrent ticket purchases.

2.1.4 Campuswire

Figure 2.1.4.1 Campuswire Mobile Application

Campuswire integrates course management, real-time communications, and Q&A forums for higher education. With features including organised sections for real-time conversations, live video lectures and office hours, and a seamless application of active learning, the platform bills itself as Q&A, announcements, and course discussion [11]. Additionally, it is a full-fledged teaching platform that gives instructors the ability to hold live video office hours, run course Q&A sessions, and include active learning into lectures—a contemporary solution to the conventional LMS discussion forums [12]. With a familiar question-and-answer format, an intuitive interface, Slack-like chat rooms, one-on-one chat, document exchange, live polling, and participation tracking, it is designed to look personal and resemble contemporary messaging apps rather than traditional LMS forums, all of which are intended to increase student engagement [13].

Strengths of Campuswire

Campuswire's ability to integrate many communication channels on a single learning platform is a major asset. Teachers say that Campuswire has an excellent forum, so students will use it voluntarily rather than being forced to, and professors will spend less time answering students' questions because they can respond to their peers. The site's forum feature is particularly well-reviewed. Because of its duplicate question avoidance system and content screening technology, the website is good at offering organised discussion forums that are well-categorized, have effective search options, and make it easy for students to access pertinent material [12]. Campuswire's engagement options include reputation points that reward student participation when messages are marked as answers or upvoted by peers. This creates a gamified learning environment that encourages peer support [13].

Weaknesses of Campuswire

Since the site does not offer roster sync or single sign-on, the students should be invited and the site should be manually configured [13]. Because it is new to the market and lacks the user base and institutional level integration of more established platforms, its complexity—which provides a wide variety of features—may be confusing to those accustomed to simpler communication tools. Furthermore, the platform's emphasis on gamification through reputation systems may foster competitive dynamics that discourage students who are not as bright academically.

Ways to Resolve the Weaknesses of Campuswire

Campuswire is recommended to provide Single Sign-On capabilities to streamline setup processes and automate roster synchronisation with key student information systems in an effort to address administrative complexity. Additionally, the platform offers reduced interface options that allow institutions to modify feature complexity based on user needs. Transform the competitive reputation system with collaborative goals that reward peer support rather than individual success, the gamification concerns would be resolved. Improved institutional relationships with present LMS providers and extensive faculty training would speed up deployment and foster user confidence.

2.1.5 Slack

Figure 2.1.5.1 Slack Mobile Application

Originally designed to help with commercial communication, Slack is a channel-based messaging system that has gained widespread use in higher education as a comprehensive tool for classroom collaboration and communication [14]. With

features like structured channels, real-time messaging, file sharing, and extensive third-party integration, the platform is described as the place where work flows and offers an environment where people, information, and tools are combined to support educational actions. By establishing virtual classroom communities that enable students to study together and participate in the learning process in a variety of ways, Slack enables instructors and students to work together, share, discuss, and engage in learning in a very meaningful way [14]. According to research, Slack can significantly improve student collaboration, communication, and classroom engagement. It has also been shown to have positive effects on active learning through features like backchanneling, which allows students to ask questions, interact with peers, and share resources in real-time during lectures [15].

Strengths of Slack

Through backchanneling, the platform may support in-class communication by giving students who might be reluctant to voice their ideas to the entire class more ways to participate, which is appropriate for a variety of requirements and preferences [15]. including Slack's multimedia features, students may generate original content that supports course subjects in addition to accessing information including images, GIFs, emoticons, videos, and embedded rich media [15]. While the personal channels provide a space for students to engage on group projects and team assignments, the platform's organisational structure in terms of channels is similar to real-life functions like lecture halls and classrooms [16].

Weaknesses of Slack

Since the website lacks certain of the FERPA compliance components needed to offer student information at educational institutions, its business-like structure might not be as suitable for educational privacy concerns [16]. Additionally, Slack promotes an informal communication style that might cause the lines between academic and informal speech to become less distinct in an educational context [16]. The platform's numerous features and notification system may contribute to this information overload and distraction, which has to be controlled to prevent students from becoming overloaded with messages and updates.

Ways to Resolve the Weaknesses of Slack

Educational institutions must establish an effective data governance architecture by signing particular FERPA compliance agreements with Slack and establishing enterprise-tier security settings that comply with educational data protection standards. Simplified notification hierarchies that only notify students to crucial academic communications, as well as optional digest summaries in general discussion channels, should be used to manage information overload.

2.1.6 Telegram

Figure 2.1.6.1 Telegram Mobile Application

Telegram is a cloud-based instant messaging program that is becoming more and more well-liked in educational groups as a means of improving material distribution, communication, and teamwork [17]. The platform is described as a mobile application that facilitates communication between users via computers and mobile phones. Its extensive features and user-friendly design have sparked interest in improving a variety of learning outcomes [17]. Telegram may be used to provide instructional information because of its big group sizes (up to 200,000 members), file sharing, chatbots, and channel-based streaming [18]. Telegram has been used to create argumentative texts, collaborative learning activities, and as an adjunct to the traditional learning process [19]. The studies show that Telegram could be used to increase student engagement in an online educational process.

Strengths of Telegram

Telegram's high message quality and multimedia support capabilities are its primary educational assets. Students may easily access instructional talks and content on many platforms and devices thanks to the Telegram cloud storage [18]. Research shows that students had favourable opinions of using Telegram for educational purposes and that the app may offer dynamic, interactive learning environments that can support both individual and group learning [17].

Weaknesses of Telegram

Students complain of technological barriers and complexity of interfaces that may pose obstacles to usefulness [17]. This informal character of the platform can result in attention-seeking behaviour and provocative behaviour by making inappropriate jokes and intolerant comments and posing an ethical issue to the educators [19]. The wastage of time may be brought about by information overload and distraction tendencies, whereas some students feel reluctant to participate in any meaningful academic discussion out of the fear of initiating a conflict or offending colleagues [17].

Ways to Resolve the Weaknesses of Telegram

In order to address behavioural issues, it should include a set of explicit communication standards that govern the content and offer many informal socialisation channels as well as serious academic discussion. Technical solutions, content filters, silent hours, and regular instruction on digital citizenship can all be provided.

2.2 Comparison for reviewing mobile applications

The current messaging and collaboration apps, such as Microsoft Teams, Facebook, GroupMe, Campuswire, Slack, and Telegram, will be examined in this section. Table 2.2.1 is a comparison table of mobile applications in several areas that will be assessed.

Criteria / Dimension	Microsoft Teams	Facebook Groups	GroupMe	Campuswire	Slack	Telegram
Instant Communication	Supported	Supported	Supported	Supported	Supported	Supported

Educational Functionality	Available	Not Available	Not Available	Available	Not Available	Not Available
Scalability (Large Groups)	High	High	Limited	Limited	High	High
Cross-platform Access	Android, iOS	Android, iOS	Android, iOS	Android, iOS	Android, iOS	Android, iOS
Offline Usability	Enabled	Not Supported	Not Supported	Not Supported	Enabled	Enabled
Content Sharing Features	Available	Available	Not Available	Available	Available	Available
Ease of Use	Less Intuitive	User-friendly	User-friendly	Less Intuitive	Less Intuitive	User-friendly
Configuration Difficulty	Complex	Simple	Simple	Moderate	Moderate	Simple

Table 2.2.1 Comparison for reviewing mobile applications

2.3 Critical Analysis of Existing Solutions

Six mobile applications were critically analysed, and the results show several underlying issues that have a direct impact on university students' experiences and learning outcomes. There are very few fully specialised apps for higher education settings, and the platforms that are now accessible are largely not education-friendly. Despite being a tool for learning, Microsoft Teams remains fundamentally an enterprise-oriented collaboration platform, adding unnecessary cognitive burden in the form of intricate interface hierarchies and function hierarchies that make it difficult for students to utilise on a daily basis. Despite being positioned as an educational platform, Campuswire's functionality is exclusive to Q&A forums; it cannot provide complete social interaction or content discovery abilities. Other applications are frequently communal or commercial general-purpose tools that aren't always created with academic standards in mind. One of the most crucial aspects is the level of complexity of the user interface that impacting student acceptance, and platforms with a variety of

ease-of-use impact students' performance. Microsoft Teams' complex feature configurations and multi-level menu systems increase the cost of learning, particularly for students with limited technical skills. Slack's interface design is more business-oriented, which is at odds with university students' usage habits. However, Campuswire's relative simplicity is offset by a functional restriction on its potential as a universal platform. These platforms typically use faulty notification systems that either transmit alerts inopportunely, causing students to miss crucial academic information, or overwhelm the user with information, causing them to ignore essential signals. These design issues lead to lower levels of participation and communication effectiveness, which directly affects how well the platforms function as learning tools.

One major flaw in the current platforms' integration of educational aspects is the lack of deep integration of the most crucial academic functions, such as the timetable for school, course organization, and learning progress tracking. Even though some of these platforms offer basic file sharing and discussions, many of these platforms' content recommendation systems are primarily based on general social media logic and are unable to successfully identify and present the content that is particularly relevant to a student in his or her academic environment and course requirements. Most systems lack customised features that adapt to university learning styles, such material categorisation by major, study group matching, and academic activity suggestions.

Furthermore, these problems indicate a research gap: no previous studies have been undertaken on a single, coherent, educationally orientated platform that would allow the reciprocal integration of academic assistance and social participation on a platform created specifically for university students.

2.4 Proposed System

This design proposes a unified mobile campus application that incorporates personalised recommendations, information acquisition, and social engagement into an easy application to solve the absence of each individual mobile application. The suggested system would provide a sophisticated social platform for FICT students to participate in group academic conversations, broadcast resources immediately, manage academic discussion threads, and filter content according to their academic preferences.

This will guarantee appropriate academic contact and minimal information dispersal among the student body. An intelligent content discovery algorithm that can handle the various academic and social content formats will be built into the program. Through the use of a machine learning algorithm that examines user behaviour, academic interests, and social connections, users may upload audiovisual content that is categorised and academic tagged. Additionally, the system automatically suggests relevant content. Customisable filtering features, including category-based discovery, interest-based suggestions, collaborative filtering, and accurate reflection of user preferences, academic calendar events, and popular campus subjects, will be made possible by the recommendation engine. A unified messaging service will be provided to facilitate social engagement by enabling direct connection inside the university community without the need for a number of outside communication providers.

Additionally, the solution is pertinent to education as it encourages cross-platform access, timely alerts, and academic-based content classification. Users may form interest-based groups, craft comprehensive biographies that showcase their passion for learning, and receive personalised material recommendations based on their campus activities and learning preferences.

Finally, by merging intelligent content discovery, sharing of knowledge within a community, as well as complete social interaction capabilities into a single educational platform, the solution has the potential to fill holes in existing applications. In addition to addressing functional and user experience shortcomings with current campus communication systems, the innovation is projected to share of knowledge within a community, and improve student participation, campus community development, and academic cooperation.

2.5 Summary

Six mobile communication platforms were compared, and the results indicate that each has distinct advantages and disadvantages in the context of education. In order to provide the FICT students with a full usability optimisation, the benefits of each program will be shown and acknowledged in this project. The combination of design will be based on the interface's ease of use while maintaining a high standard of educational knowledge. This will make it easy for FICT students to use both the social

networking features and the tools for finding academic material in a single, cohesive mobile application.

Chapter 3

System Methodology/Approach

3.1 System Design Diagram/Equation

3.1.1 Overview of Agile SDLC Methodology

This individual project adopts an **Agile development model** tailored for solo engineering efforts, emphasizing a technical iterative cycle to manage the complexity of full-stack development. Rather than relying on external user focus groups during each stage, the methodology focuses on **feature-driven iterations** and **systematic self-evaluation** to ensure the robustness of the FICT360 application. This approach allows for the dynamic adjustment of technical requirements as the backend logic and frontend UI evolve simultaneously.

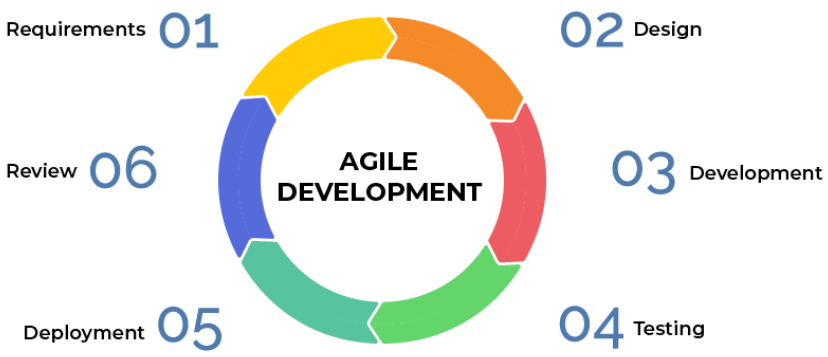


Figure 0.1.1.1 Agile Development SDLC

The development process is structured into phased technical sprints, each consisting of planning, implementation, and rigorous self-testing. During these sprints, the developer performs constant debugging and performance hardening—such as resolving media display issues and optimizing database queries—to ensure the system aligns with the primary project objectives. This iterative refinement ensures that each core module,

from authentication to the recommendation engine, is technically verified before moving to the next stage.

3.1.2 System Architecture Diagram

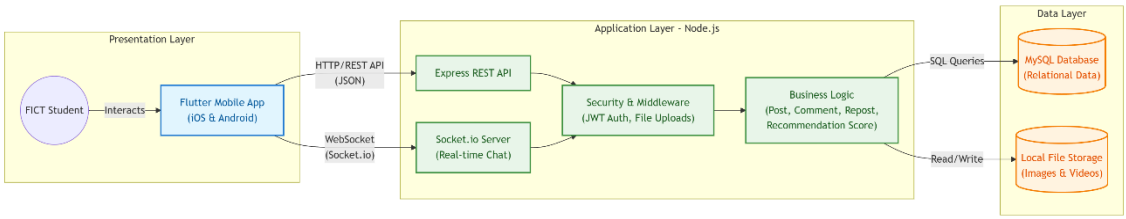


Figure 0.1.2.1 System Architecture Diagram

A solid three-tier architecture model that logically splits the system into three layers: display, application, and data form the foundation of the FICT360 application's system architecture. Scalability, high performance, and safe data management are ensured by this decoupling, which is especially made to satisfy the needs of the FICT student body. The display layer, which is at the top of this architecture, includes an Android cross-platform mobile application made with the Flutter framework. The primary interface for FICT students is this client-side interface, which renders media material and manages the local state. The client uses two different channels to communicate with the backend: real-time chat functionality, which allows private and group messages to be transmitted immediately without necessitating a page refresh, and an HTTP/REST interface that uses JSON as a standard data exchange format for user authentication and profile management.

The core processing engine of the entire system is the application layer, which is situated in the centre of the architecture and is built on a Node.js environment. Using a few unique components, the layer manages the complex connection between the data storage and the user interface. The Socket.io server is the entry point for handling requests pertaining to the application's forum features, while the Express REST API is the usual entry point for handling queries related to administering the social interaction components of the program. A multipart form data sub-layer handles JWT media file upload, while a security middleware sub-layer authenticates each student identity using JWT (JSON Web Token). In order to customise the content stream, this layer applies the weighted recommendation logic, which determines relevance ratings depending on user tag interaction. To offer a dynamic user experience, it also manages the logic

underlying sophisticated community features like bookmarking, commenting, and reposting.

The foundation of the design is the data layer, which uses two primary storage techniques to manage overall data durability. All structured data, including user credentials, recommendation system relational weights, post content, and follower connections, is stored in a MySQL relational database management system. The server has its own local file storage system in addition to the database. This system is intended to safely store unstructured media items, such student-uploaded photos and videos, which are then returned to the frontend via statical URL routes. When taken as a whole, these levels provide the university community with a safe, responsive, and well-integrated environment that successfully bridges the gap between social engagement and academic material discovery inside a single digital ecosystem.

3.1.3 Use Case Diagram and Description

Use Case Diagram

Figure 0.1.3.1 Use Case Diagram

Use Case Description

Table 0.1.3.1 User Login Use Case Description Table

Use Case ID	**UC01**
Use Case Title	User Authentication
Summary	Registered users access the system using valid credentials
Primary Actor	FICT Student
Initiating Event	The user opens the app and chooses sign in.
Preconditions	A valid account must exist and the application is installed
Main Flow	1. The user starts the application. 2. The system shows the login interface. 3. The user enters their password and e-mail. 4. The system verifies the credentials 5. The system redirects the homepage.
Sub-process	S1: Password Recovery – user requests reset, receives OTP, verifies code, and sets new password
Alternative Scenarios	4a. Invalid credentials trigger an error message 3a. Incorrect OTP prompts error and allows resend

Table 0.1.3.2 User Logout Use Case Description Table

Use Case ID	**UC02**
Use Case Title	User Logout
Summary	The user exits the application securely
Primary Actor	FICT Student

Initiating Event	The user selects logout from the sidebar
Preconditions	The user must be logged in
Main Flow	1. The user selects logout 2. The system shows confirmation 3. The user confirms or cancels 4. The system logs out or stays on page
Sub-process	None
Alternative Scenarios	3a. Cancel action keeps user on current page

Table 0.1.3.3 User Registration Use Case Description Table

Use Case ID	**UC03**
Use Case Title	User Registration
Summary	New users create an account in the system
Primary Actor	FICT Student
Initiating Event	The user selects sign-up
Preconditions	Application must be installed
Main Flow	1. User open the app 2. Registration page appears on the system. 3. The user enters password and email. 4. The system verifies the presence of accounts 5. A verification code is sent by the system 6. User inputs the code

	7. The system verifies the code. 8. User sets up their profile.
Sub-process	None
Alternative Scenarios	4a. Existing account shows error 7a. Invalid code prompts retry

Table 0.1.3.4 Create Post Use Case Description

Use Case ID	**UC04**
Use Case Title	Create Post
Summary	The user publishes a new post
Primary Actor	FICT Student
Initiating Event	The user selects create option
Preconditions	User must be logged in
Main Flow	1. User selects create 2. User enters content/media 3. System validates input 4. System saves post 5. Post appears on homepage
Sub-process	None
Alternative Scenarios	3a. Empty input triggers validation message

Table 0.1.3.5 View Post History Use Case Description

Use Case ID	**UC05**
Use Case Title	View Post History
Summary	The user reviews past posts
Primary Actor	FICT Student
Initiating Event	User selects post history
Preconditions	User must be on profile page
Main Flow	1. The user accesses their profile 2. The user chooses the past 3. The system retrieves postings 4. Records are checked by the system 5. Posts are shown by the system
Sub-process	None
Alternative Scenarios	4a. No records show empty state

Table 0.1.3.6 Edit Profile Use Case Description

Use Case ID	**UC06**
Use Case Title	Edit Profile
Summary	The user updates profile details
Primary Actor	FICT Student
Initiating Event	User selects edit profile
Preconditions	User must be logged in

Main Flow	1. The user accesses their profile 2. The user modifies information 3. The system verifies input 4. The system stores modifications 5. The system goes back to the profile
Sub-process	None
Alternative Scenarios	3a. Missing fields trigger error

Table 0.1.3.7 Create Comment Use Case Description

Use Case ID	**UC07**
Use Case Title	Create Comment
Summary	The user posts a comment
Primary Actor	FICT Student
Initiating Event	User selects comment option
Preconditions	User must view a post
Main Flow	1. The user views the post 2. The user adds a remark 3. The system verifies input 4. Comments are stored by the system 5. A comment appears
Sub-process	None

Alternative Scenarios	3a. Empty comment shows error

Table 0.1.3.8 Edit Comment Use Case Description

Use Case ID	**UC08**
Use Case Title	Edit Comment
Summary	The user modifies a comment
Primary Actor	FICT Student
Initiating Event	User selects edit option
Preconditions	User must have comment
Main Flow	1. The user views the post 2. A user modifies a comment 3. The system verifies 4. Comment on system updates 5. A revised remark was shown
Sub-process	None
Alternative Scenarios	3a. Empty input shows error

Table 0.1.3.9 Delete Comment Use Case Description

Use Case ID	**UC09**
Use Case Title	Delete Comment
Summary	The user removes a comment

Primary Actor	FICT Student
Initiating Event	User selects delete option
Preconditions	User must have comment
Main Flow	1. The user views the post 2. The user chooses to delete 3. The system verifies the action 4. The user attests 5. The comment is removed by the system
Sub-process	None
Alternative Scenarios	4a. Cancel stops deletion

Table 0.1.3.10 Like Post Use Case Description

Use Case ID	**UC10**
Use Case Title	Like Post
Summary	The user likes or removes like from a post
Primary Actor	FICT Student
Initiating Event	User clicks like button
Preconditions	User must be logged in
Main Flow	1. User selects post 2. User clicks like 3. System checks status 4. System updates status

	5. Data saved
Sub flows	None
Alternate flow	4a. Post already liked - system executes unlike action 4b. Post not liked - system executes like action

Table 0.1.3.11 Search Posts Use Case Description

Use Case ID	**UC11**
Use Case Title	Search Posts
Summary	The user searches posts
Primary Actor	FICT Student
Initiating Event	User uses search function
Preconditions	User must access explore page
Main Flow	1. User opens explore 2. User enters keywords 3. System retrieves posts 4. System validates results 5. System displays posts
Sub-process	S1: Browse by category
Alternative Scenarios	4a. No results shows message

Table 0.1.3.12 Search Users Use Case Description

Use Case ID	**UC12**

Use Case Title	Search Users
Summary	The user searches other users
Primary Actor	FICT Student
Initiating Event	User performs search
Preconditions	User must be logged in
Main Flow	1. User opens explore 2. User inputs search 3. System retrieves users 4. System validates 5. System displays results
Sub-process	S1: Follow/Unfollow
Alternative Scenarios	4a. No user found

Table 0.1.3.13 Create Private Chat Use Case Description

Use Case ID	**UC13**
Use Case Title	Private Chat
Summary	The user starts a one-to-one chat
Primary Actor	FICT Student
Initiating Event	User selects chat
Preconditions	User must be authenticated
Main Flow	1. User opens chat 2. System checks existing chat

	3. User selects new chat 4. User selects recipient 5. System creates chat
Sub-process	S1: Resume existing chat
Alternative Scenarios	3a. Existing chat opens directly

Table 0.1.3.14 Create Group Chat Use Case Description

Use Case ID	**UC14**
Use Case Title	Group Chat
Summary	The user creates a group conversation
Primary Actor	FICT Student
Initiating Event	User selects group chat
Preconditions	User must be logged in
Main Flow	1. User opens chat 2. User selects group option 3. User selects members 4. System creates group 5. Chat is displayed
Sub-process	S1: Resume existing group
Alternative Scenarios	3a. Existing group opens

Table 0.1.3.15 Repost Content Use Case Description

Use Case ID	**UC15**
Use Case Name	Repost Content
Brief Description	User redistributes an existing post into their own feed.
Actor	FICT Student
Trigger	User clicks the repost icon on a post card.
Precondition	User is logged in; the target post exists in the database.
Normal flow of events	1. User clicks the repost button on a specific post. 2. System checks if the user has already reposted this content. 3. System creates a new post entry with a reference to the original_post_id. 4. Post appears in the community feed with a "Reposted" label.
Sub flows	None
Alternate flow	2a. Content already reposted - System executes "un-repost" action (deletes the repost record).

Table 0.1.3.16 Save/Bookmark Post Use Case Description

Use Case ID	**UC16**
Use Case Name	Save / Bookmark Post
Brief Description	User saves a post for future reference.
Actor	FICT Student
Trigger	User clicks the bookmark icon on a post.
Precondition	User is logged in.
Normal flow of events	1. Click bookmark icon. 2. The saved_posts table stores user and post IDs. 3. Post goes into "Saved Posts"
Sub flows	None

Alternate flow	2a. Post already saved - System removes the post from the saved_posts table.

Table 0.1.3.17 Global Search Use Case Description

Use Case ID	**UC17**
Use Case Name	Global Search
Brief Description	User searches for both posts and other students simultaneously.
Actor	FICT Student
Trigger	User inputs a query into the global search bar on the Explore page.
Precondition	User is logged in; the target post exists in the database.
Normal flow of events	1. User enters keywords in the search bar. 2. System performs a "Global Search" SQL query matching usernames and post content. 3. System returns a combined list of matching users and posts.
Sub flows	None
Alternate flow	3a. No results found - System displays "Search failed" or empty results message.

Table 0.1.3.18 Weighted 'For You' Feed Use Case Description

Use Case ID	**UC18**
Use Case Name	Weighted 'For You' Feed
Brief Description	System delivers personalized content based on user engagement patterns.
Actor	FICT Student / System
Trigger	On Home, user selects "For You" tab.

Precondition	User has interaction history (likes/views).
Normal flow of events	1. User opens the application. 2. System calculates a recommendation_score for each post using user_tag_weights. 3. System sorts posts by the highest weight and creation date. 4. User views a personalized feed tailored to their academic and social interests.
Sub flows	None
Alternate flow	None

3.1.4 Activity Diagram

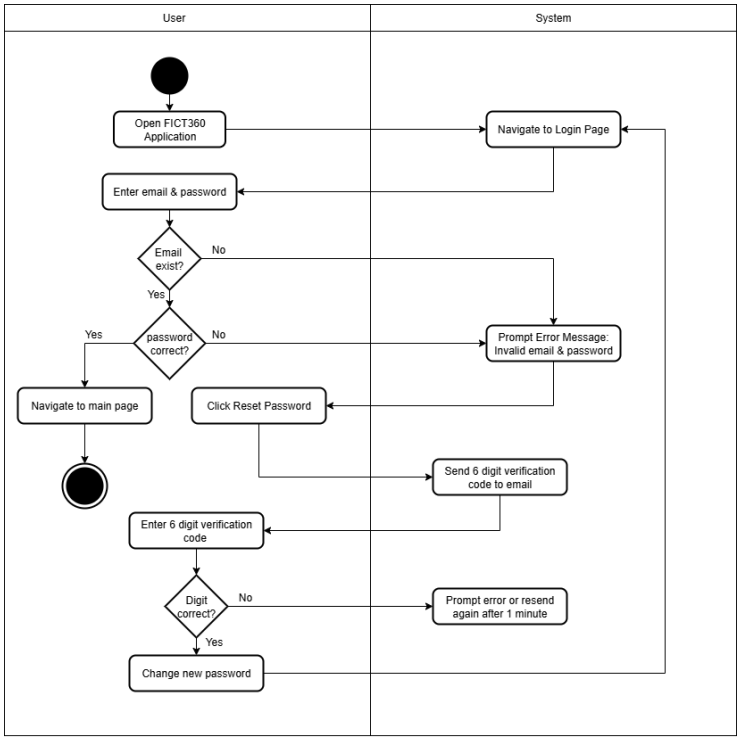


Figure 0.1.4.1 Activity Diagram for Login

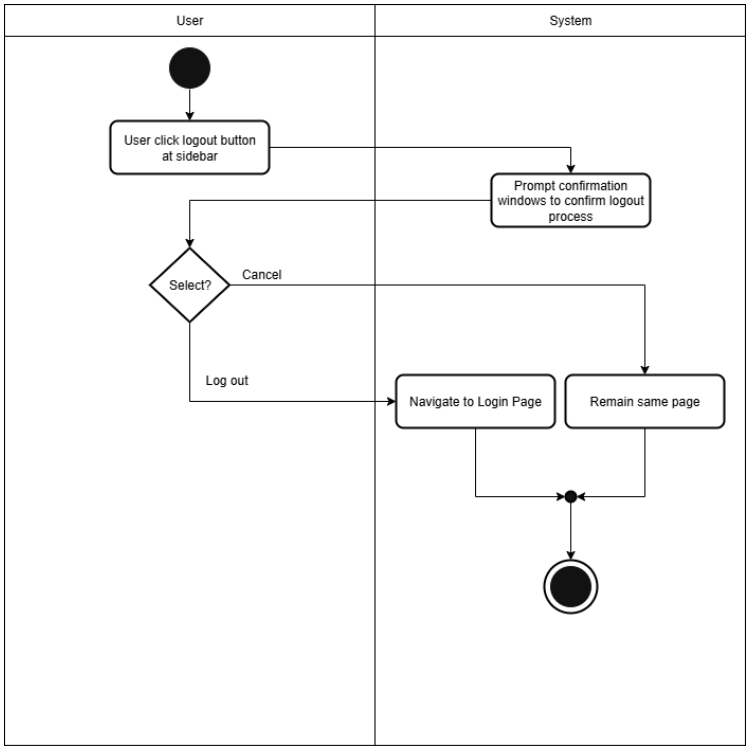


Figure 0.1.4.2 Activity Diagram for Logout

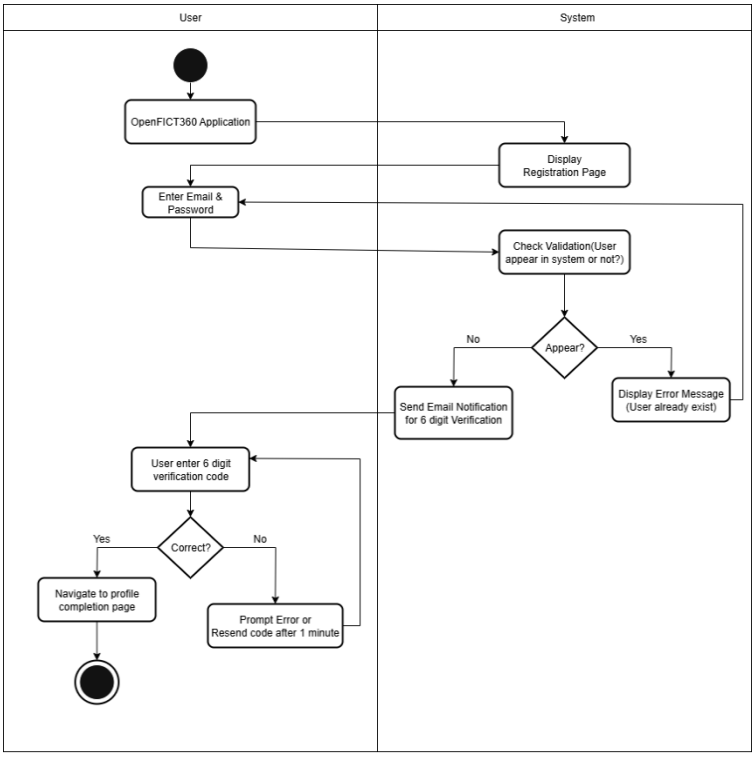


Figure 0.1.4.3 Activity Diagram for Registration

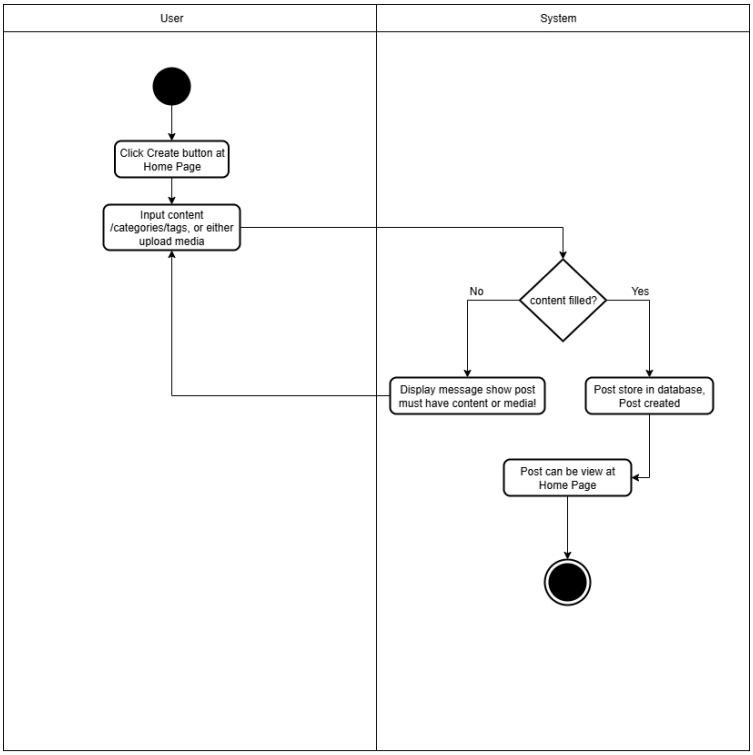


Figure 0.1.4.4 Activity Diagram for Create Post

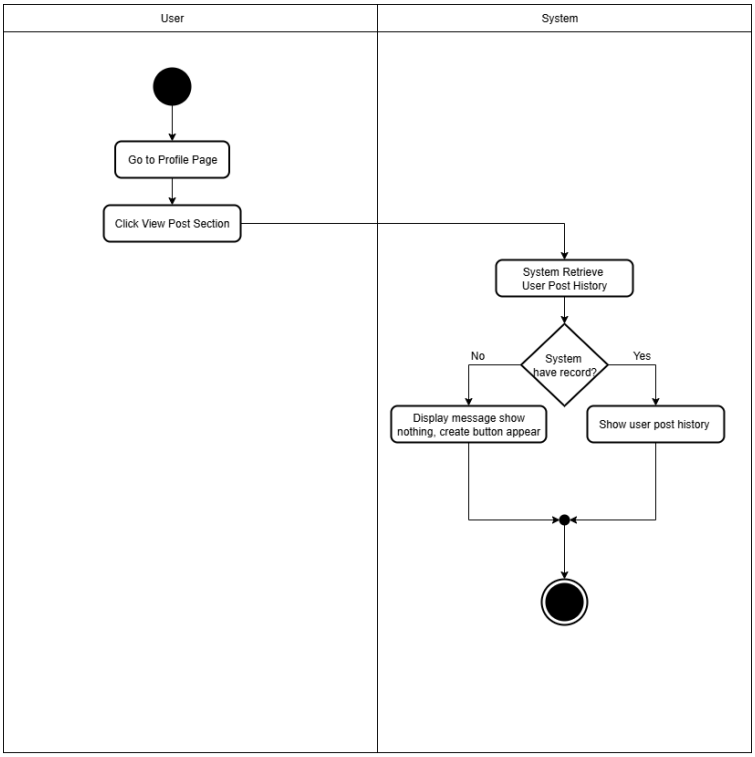


Figure 0.1.4.5 Activity Diagram for View Post History

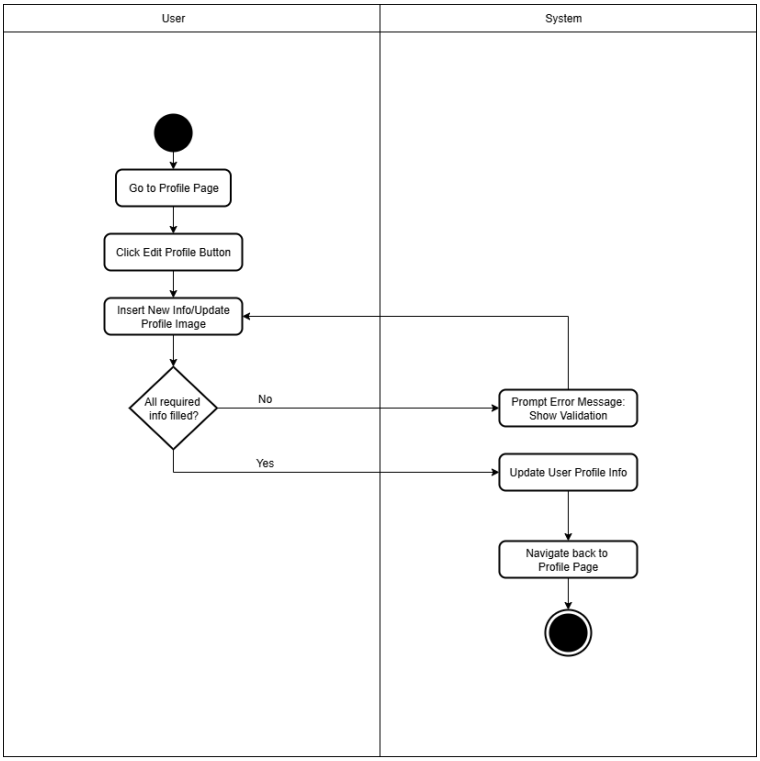


Figure 0.1.4.6 Activity Diagram for Edit Profile Info

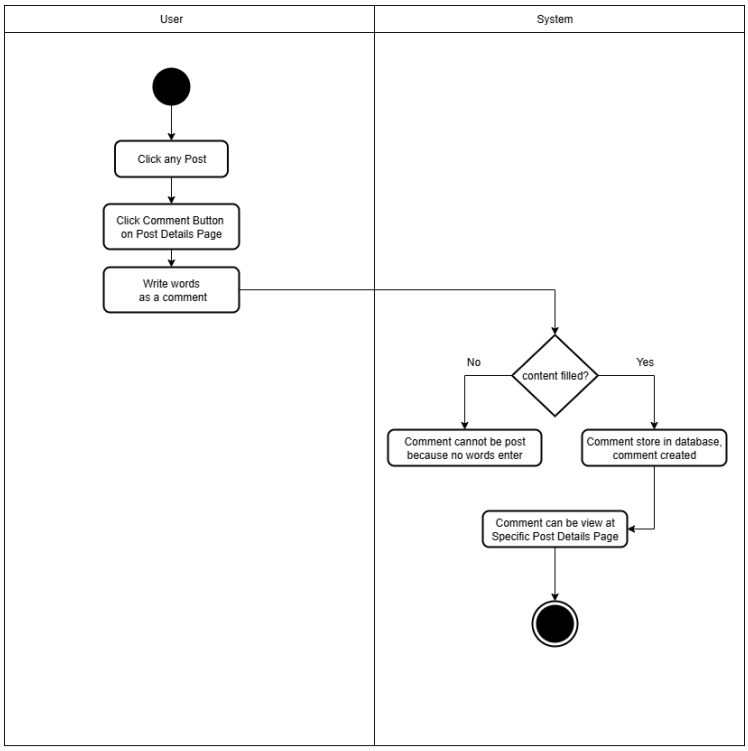


Figure 0.1.4.7 Activity Diagram for Create Comment

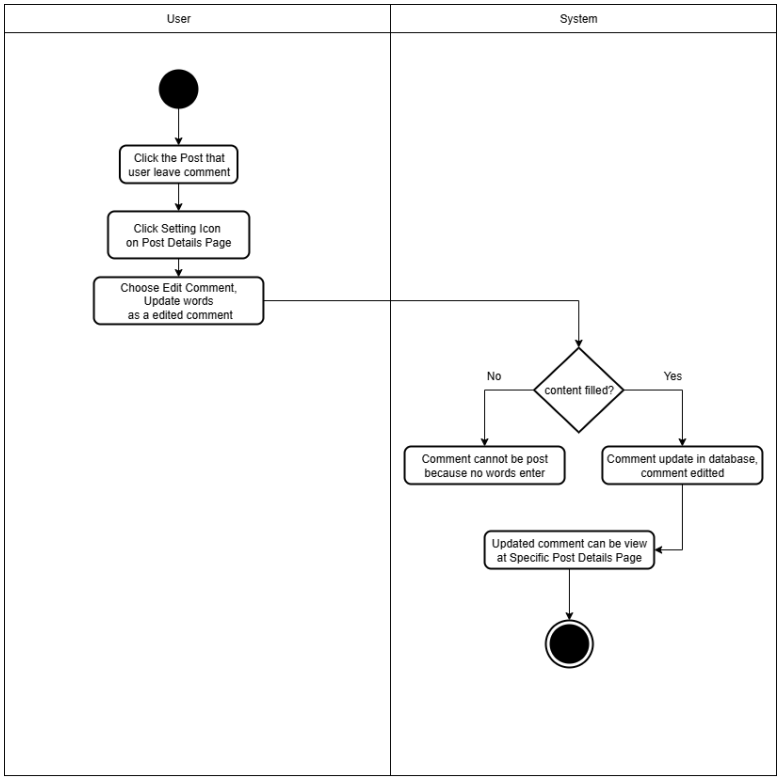


Figure 0.1.4.8 Activity Diagram for Edit Comment

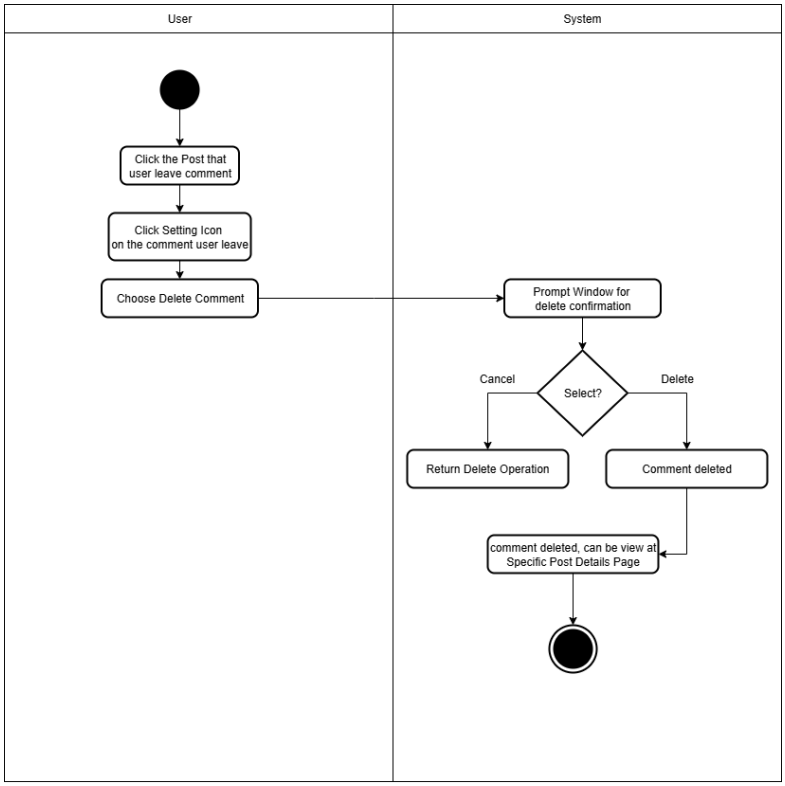


Figure 0.1.4.9 Activity Diagram for Delete Comment

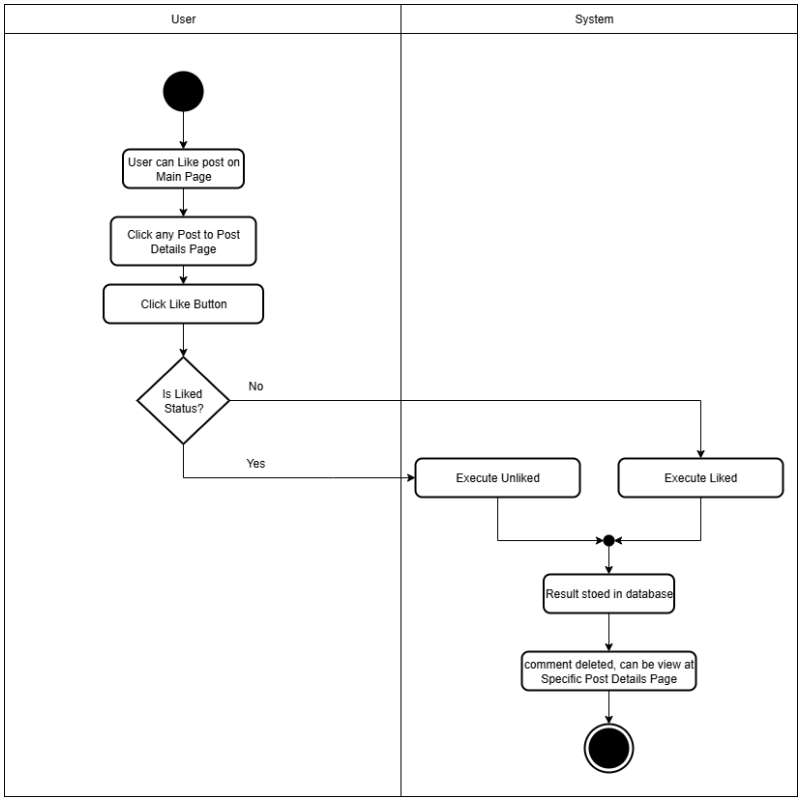


Figure 0.1.4.10 Activity Diagram for Like Posts

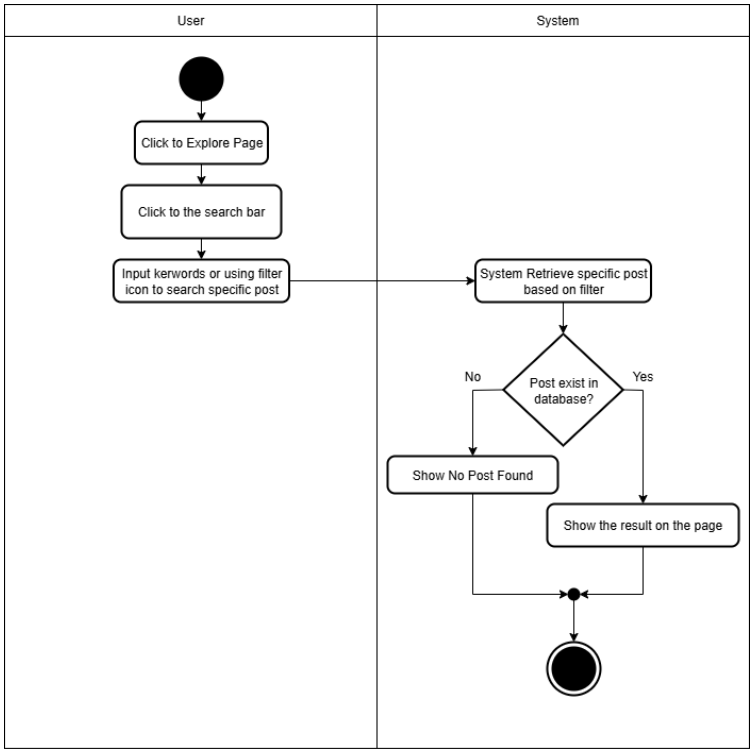


Figure 0.1.4.11 Activity Diagram for Search Posts

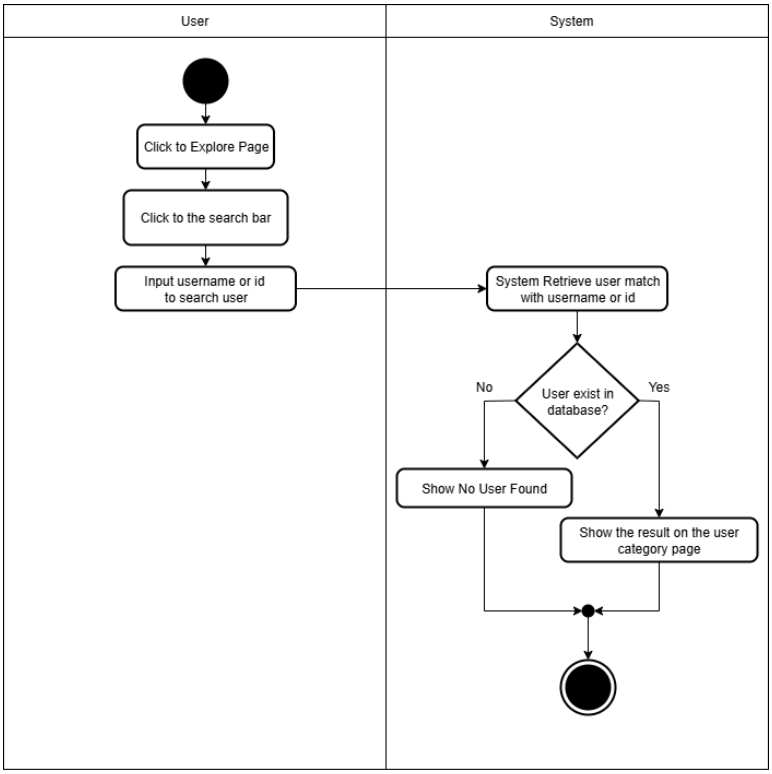


Figure 0.1.4.12 Activity Diagram for Search Users

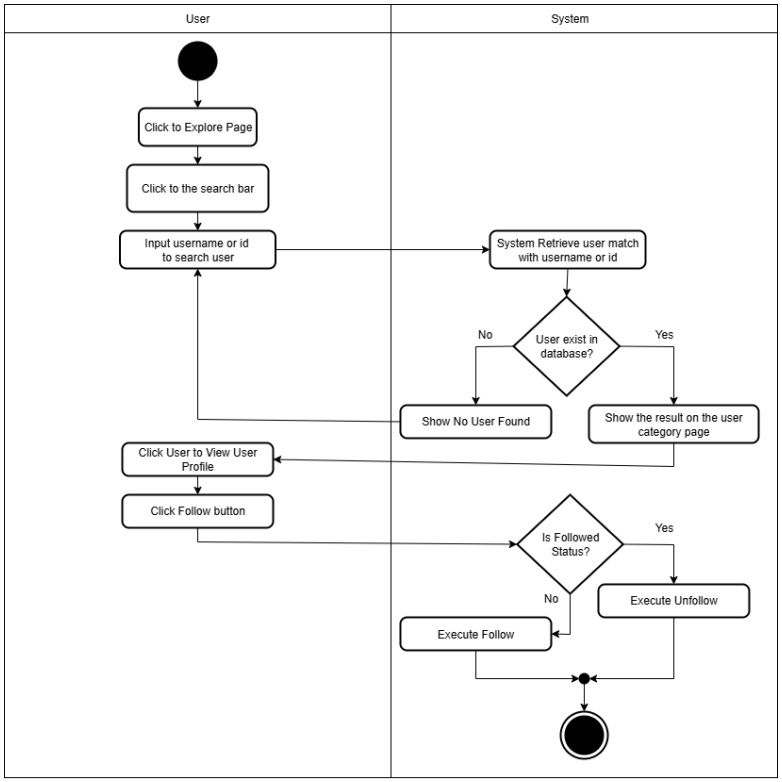


Figure 0.1.4.13 Activity Diagram for Follow/Unfollow Users

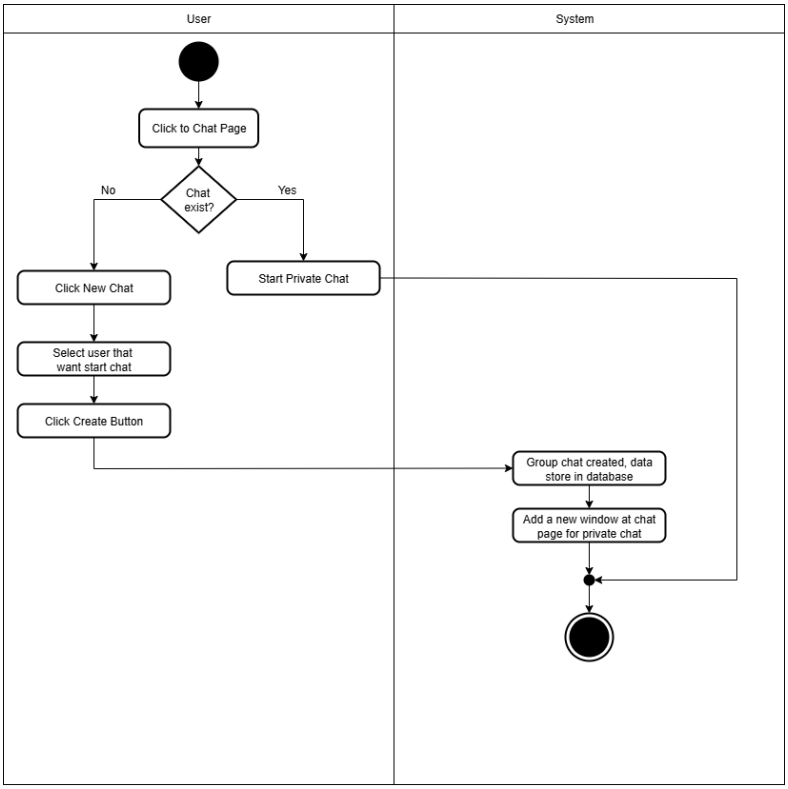


Figure 0.1.4.14 Activity Diagram for Create Private Chat

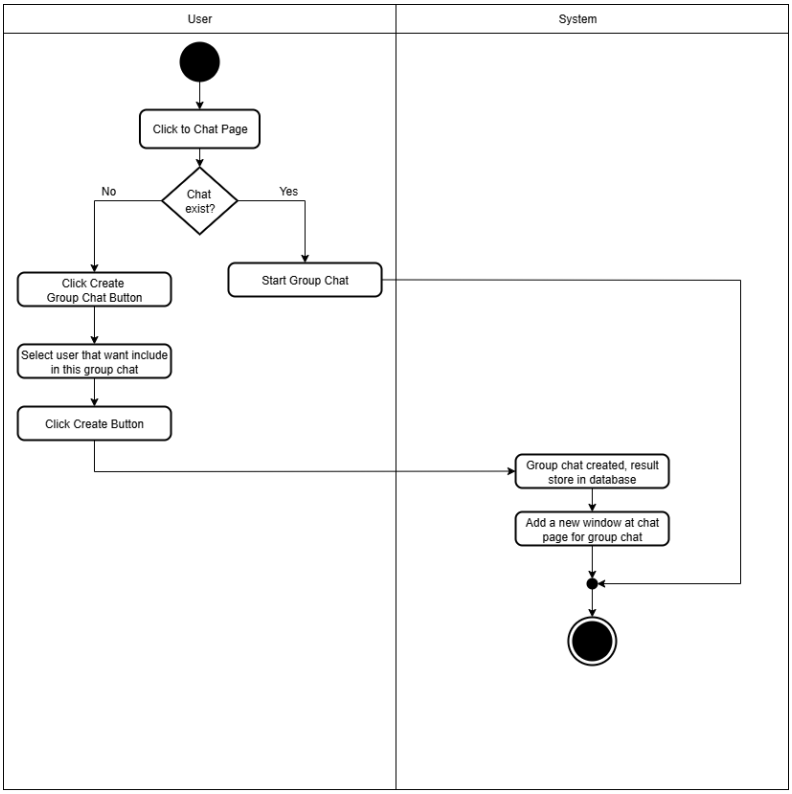


Figure 0.1.4.15 Activity Diagram for Create Group Chat

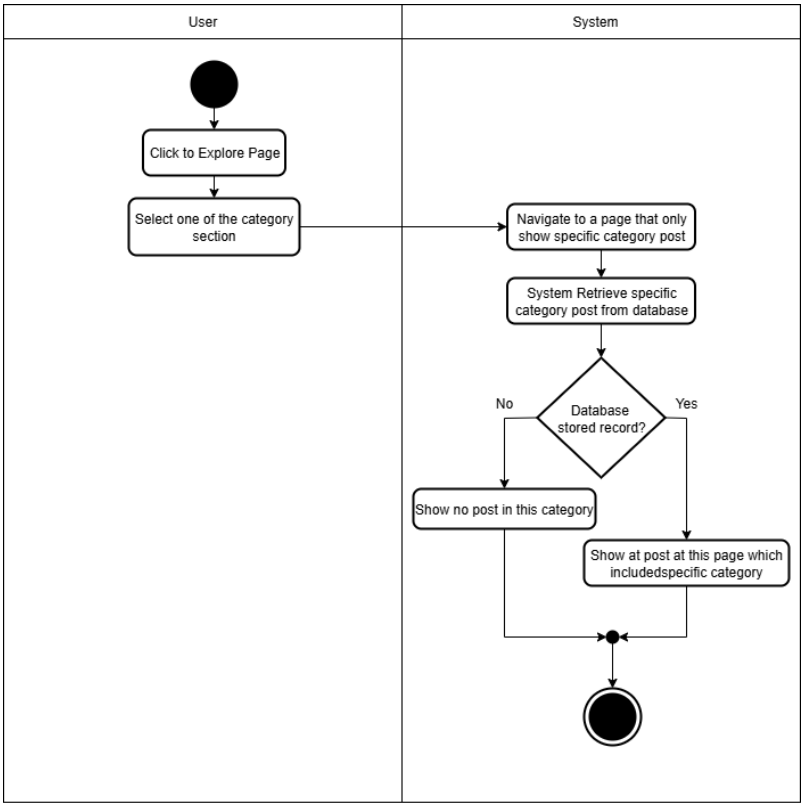


Figure 0.1.4.16 Activity Diagram for Categorized Posts

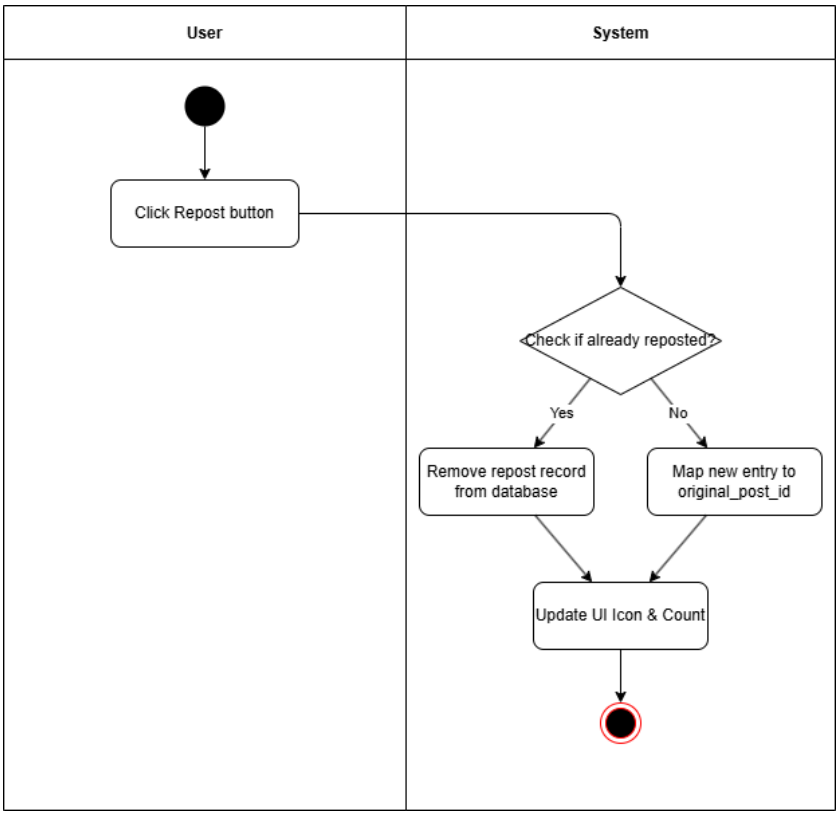


Figure 0.1.4.17 Activity Diagram for Repost Content

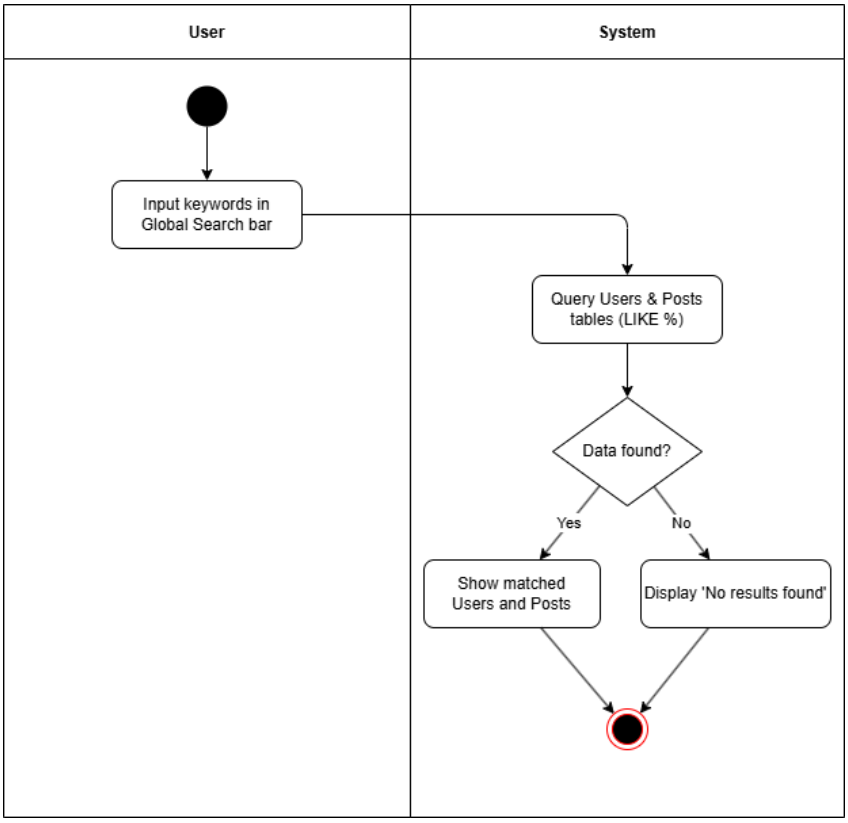


Figure 0.1.4.18 Activity Diagram for Global Search

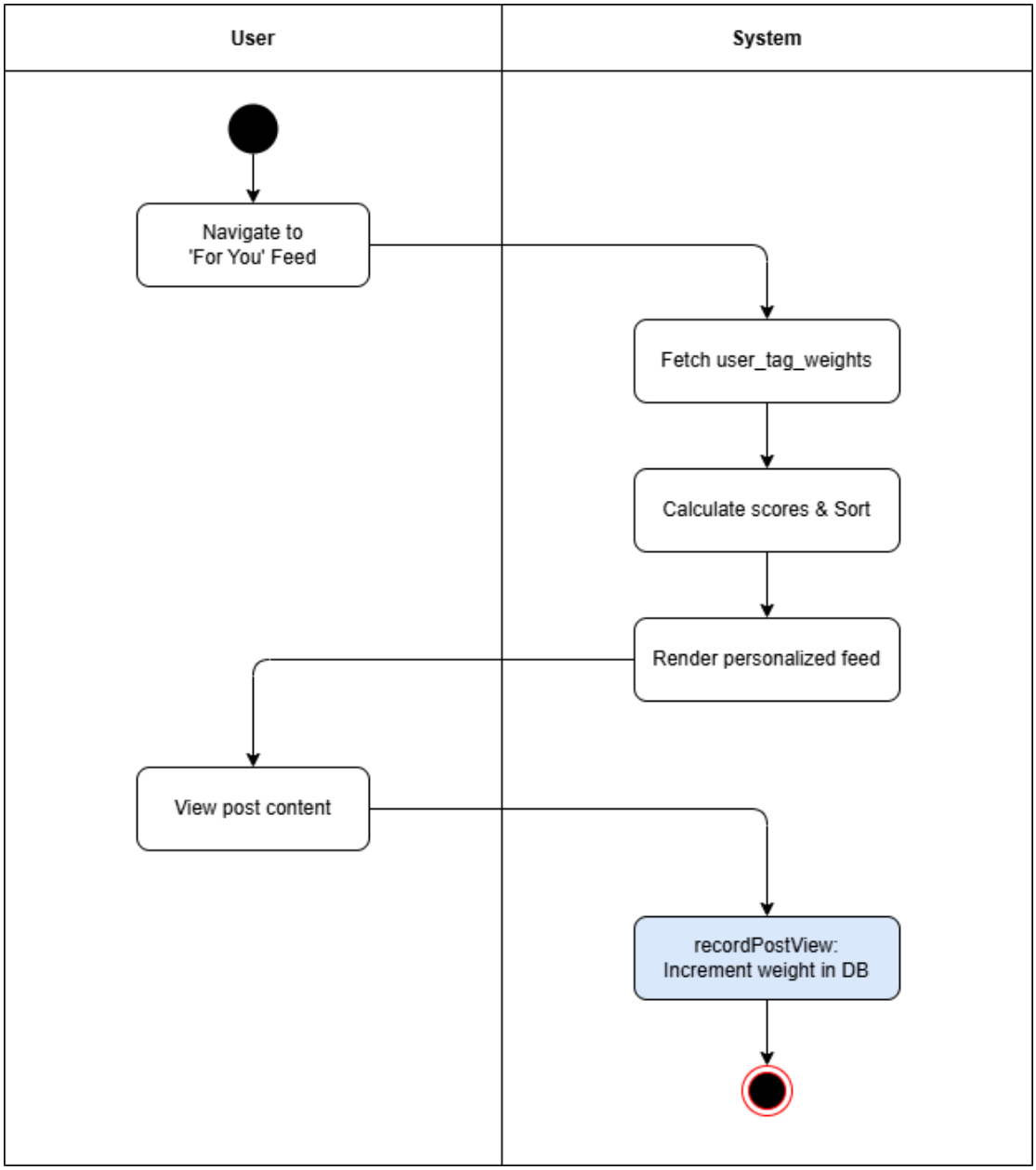


Figure 0.1.4.19 Activity Diagram for Weighted 'For You' Feed

Chapter 4

System Design

4.1 System Block Diagram

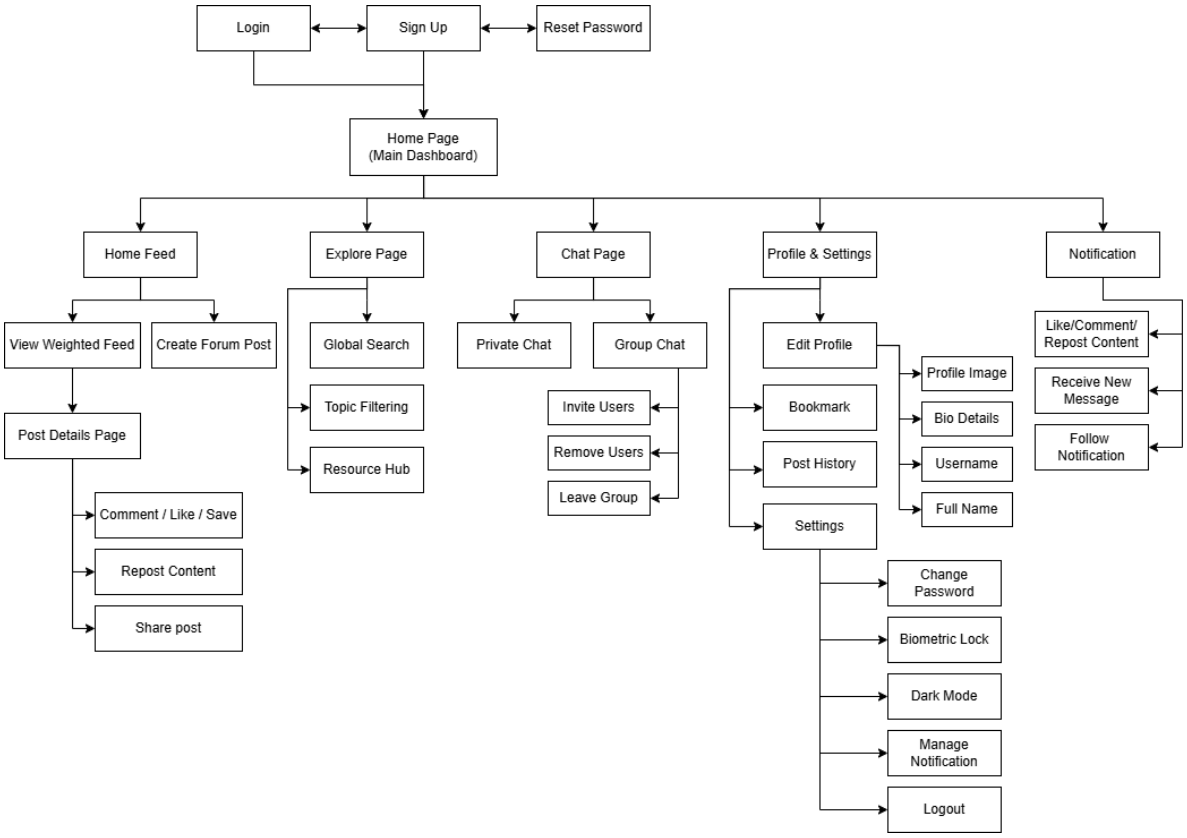


Figure 0.1.1 System Block Diagram

4.2 System Components Specifications

Understanding the user experience from the moment the mobile application opens is essential. A login page, which serves as the safe entry point to the campus community platform, is the first thing a student sees when they launch FICT360. Accessing the forum and social function requires a registered email address and password. The application provides new students with a straightforward sign-up process that makes it easy for them to construct their academic profile. They can also choose to change their password in order to get their accounts back. The technology

automatically navigate user to the main Home Page dashboard, where their individual academic and social experience begins, following a successful login or registration.

.

4.2.1 Home Page

content consumption requirements. In order to constantly prioritise information that corresponds with the student's interests, the "For You" feed employs a weighted recommendation logic that analyses user interactions (such post views and likes) with particular academic categories. On the other hand, only updates from peers the user has selected to connect with are displayed chronologically in the "Following" feed.

This page's main goals are interactivity and content development. By using the "Create Forum Post" button, users may quickly start a new topic, upload pertinent photos or videos, and classify their post using pre-established academic tags (e.g., Assignment Help, Campus Life). The program provides extensive interaction mechanisms for content that already exists. By like, commenting, or bookmarking posts for later use, users can participate. One particularly noteworthy feature is the "Repost" option, which enables students to share important academic materials with their own followers, encouraging faculty members to share information. Additionally, users can update or remove their postings straight from the feed, giving them complete control over their own content.

4.2.2 Explore Page

The Explore Page is designed to break down "Information Cocoons" by facilitating efficient content discovery across the entire platform. Upon entering this page, users are presented with a series of quick-action category buttons representing popular academic topics. These topic filters allow students to rapidly sort through the database and locate specific discussions relevant to their current coursework or interests, such as programming help or upcoming campus events.

Beyond predefined categories, the Explore Page incorporates a robust Global Search function. This unified search bar allows students to input custom keywords to simultaneously query both the post content database and the user directory. This feature proves invaluable when a student is looking for highly specific academic materials or trying to locate a particular peer or study group within the faculty. The integration of

this search mechanism ensures that valuable information and connections are always easily accessible.

4.2.3 Chat Page

The Chat Page provides a dedicated environment for real-time social and academic communication, powered by WebSocket technology. This section is crucial for transforming the platform from a simple forum into a collaborative community. Users can initiate private, one-on-one conversations to discuss specific assignments, share resources, or simply build social connections with fellow students.

Recognizing the collaborative nature of university life, the system also supports robust Group Chat functionalities. Users can create dedicated group channels for specific course projects or study groups. The group management interface allows the creator to invite other active community members to join the discussion, remove participants if necessary, or leave groups they no longer need. This real-time messaging capability significantly reduces the friction of project coordination and helps students maintain active social relationships on campus.

4.2.4 Profile Page

Within the FICT360 environment, the Profile Page serves as the user's individual academic identity. Here, students can view how their profile appears to others, displaying their chosen username, full name, profile image, and a brief biography detailing their major or academic interests. This page also serves as a personal archive, allowing users to review their entire post history to manage past discussions or track their academic contributions over the semester.

Additionally, this section houses the user's "Saved Posts" collection. By navigating to the Bookmark tab, students can quickly retrieve the valuable resources, assignment tips, or important announcements they had previously saved from the Home or Explore feeds, effectively turning the profile into a personalized academic knowledge base.

4.2.5 Notification Page

The Notification Page ensures that students remain actively engaged and informed about interactions related to their content. It serves as a centralized hub for all incoming alerts. Users receive immediate updates when someone likes their post, leaves

a new comment, or reposts their content. It also notifies users of new followers and alerts them when they receive a new direct message or group chat invitation. This timely feedback loop encourages continuous participation and ensures that students do not miss important academic discussions or social outreach.

4.2.6 Settings Page

Students have complete control over their application choices and account security through the Settings Page. This interface makes it simple for users to edit their biography and profile picture, among other details. The page offers choices to modify their account password for security management.

The Settings page provides visual and privacy customisations to improve the user experience, like adjusting notification preferences to prevent distractions during lectures and enabling "Dark Mode" for comfortable late-night study sessions. Additionally, it offers a biometric lock feature for compatible devices, giving the student's private communications and information an additional degree of security. Lastly, users can safely terminate their active session with a simple logout feature.

4.3 System Data & Interfaces Design

4.3.1 Database Design

For the back-end storage of FICT360, the database tables were structured to minimize duplicate data, aligning closely with Third Normal Form (3NF) standards. Since the application's features evolved frequently during the development phase, managing database updates manually became inefficient. To solve this, I configured the Node.js server to handle database migrations automatically. Whenever the server starts, it checks the current database structure and safely appends any new columns to the existing tables without wiping the previous records. The core tables, their primary fields, and how they connect to each other are outlined below.

Table 0.3.1.1 bookmark Table

Column Name	Type	Default	Description
id	int	AUTO_INCREMENT	Primary key
user_id	int	NOT NULL	Foreign key -> users.id
post_id	int	NOT NULL	Foreign key -> posts.id
folder_name	varchar(100)	'General'	Folder name
notes	text	NULL	Optional notes
is_private	tinyint(1)	1	Private flag
created_at	timestamp	CURRENT_TIMESTAMP	Creation timestamp
updated_at	timestamp	CURRENT_TIMESTAMP	Update timestamp

Table 0.3.1.2 categories Table

Column Name	Type	Default	Description
id	int	AUTO_INCREMENT	Primary key
name	varchar(100)	NOT NULL	Category name
description	text	NULL	Category description

Column Name	Type	Default	Description
parent_id	int	NULL	Foreign key -> categories.id (parent)
color_code	varchar(7)	'#6B7280'	Display color code
icon	varchar(50)	'folder'	Icon identifier
is_active	tinyint(1)	1	Active flag
sort_order	int	0	Display order
post_count	int	0	Number of posts
created_at	timestamp	CURRENT_TIMESTAMP	Creation timestamp
updated_at	timestamp	CURRENT_TIMESTAMP	Update timestamp

Table 0.3.1.3 comments Table

Column Name	Type	Default	Description
id	int	AUTO_INCREMENT	Primary key
post_id	int	NOT NULL	Foreign key -> posts.id
user_id	int	NOT NULL	Foreign key -> users.id
parent_id	int	NULL	Foreign key -> comments.id (reply)
content	text	NOT NULL	Comment content
created_at	timestamp	CURRENT_TIMESTAMP	Creation timestamp
updated_at	timestamp	CURRENT_TIMESTAMP	Update timestamp
likes_count	int	0	Number of likes

Table 0.3.1.4 content_views Table

Column Name	Type	Default	Description
id	int	AUTO_INCREMENT	Primary key

Column Name	Type	Default	Description
post_id	int	NOT NULL	Foreign key -> posts.id
user_id	int	NULL	Foreign key -> users.id
view_duration	int	0	View duration in seconds
referrer_source	varchar(255)	NULL	Traffic source
device_type	enum	NULL	desktop / mobile / tablet
is_unique_view	tinyint(1)	1	Unique view flag
user_ip	varchar(45)	NULL	User IP address
created_at	timestamp	CURRENT_TIMESTAMP	Creation timestamp

Table 0.3.1.5 conversation_participants Table

Column Name	Type	Default	Description
conversation_id	int	NOT NULL	PK + Foreign key -> conversations.id
user_id	int	NOT NULL	PK + Foreign key -> users.id
joined_at	timestamp	CURRENT_TIMESTAMP	Join timestamp
is_muted	tinyint(1)	0	Mute flag
status	enum	'active'	active / left / removed

Table 0.3.1.6 conversations Table

Column Name	Type	Default	Description
id	int	AUTO_INCREMENT	Primary key
type	enum	'private'	private / group
name	varchar(255)	NULL	Group name
admin_id	int	NULL	Foreign key -> users.id (admin)

Column Name	Type	Default	Description
last_message_id	int	NULL	ID of latest message
last_message_at	timestamp	NULL	Timestamp of latest message
created_at	timestamp	CURRENT_TIMESTAMP	Creation timestamp
group_image	varchar(255)	NULL	Group avatar path

Table 0.3.1.7 email_verifications Table

Column Name	Type	Default	Description
id	int	AUTO_INCREMENT	Primary key
email	varchar(255)	NOT NULL	Email address
verification_code	varchar(6)	NOT NULL	Verification code
expires_at	timestamp	NOT NULL	Expiry timestamp
created_at	timestamp	CURRENT_TIMESTAMP	Creation timestamp
is_verified	tinyint(1)	0	Verified flag
attempts	int	0	Number of attempts
verification_type	varchar(50)	'email_verification'	Verification type

Table 0.3.1.8 follows Table

Column Name	Type	Default	Description
id	int	AUTO_INCREMENT	Primary key
follower_id	int	NOT NULL	Foreign key -> users.id (follower)
following_id	int	NOT NULL	Foreign key -> users.id (followed)

Column Name	Type	Default	Description
created_at	timestamp	CURRENT_TIMESTAMP	Creation timestamp
status	enum	'accepted'	pending / accepted

Table 0.3.1.9 likes Table

Column Name	Type	Default	Description
id	int	AUTO_INCREMENT	Primary key
post_id	int	NULL	Foreign key -> posts.id
user_id	int	NOT NULL	Foreign key -> users.id
created_at	timestamp	CURRENT_TIMESTAMP	Creation timestamp
comment_id	int	NULL	Foreign key -> comments.id

Table 0.3.1.10 messages Table

Column Name	Type	Default	Description
id	int	AUTO_INCREMENT	Primary key
conversation_id	int	NOT NULL	Foreign key -> conversations.id
sender_id	int	NOT NULL	Foreign key -> users.id (sender)
content	text	NOT NULL	Message content
post_id	int	NULL	Foreign key -> posts.id (shared post)
is_read	tinyint(1)	0	Read flag
created_at	timestamp	CURRENT_TIMESTAMP	Creation timestamp
message_type	enum	NULL	text/image/video/audio/document/system

Column Name	Type	Default	Description
media_url	varchar(255)	NULL	Media file path
file_name	varchar(255)	NULL	File name
file_size	int	NULL	File size in bytes

Table 0.3.1.11 notifications Table

Column Name	Type	Default	Description
id	int	AUTO_INCREMENT	Primary key
user_id	int	NOT NULL	Foreign key -> users.id (recipient)
type	enum	NOT NULL	like/comment/follow/message/general/follow_request
related_user_id	int	NULL	Foreign key -> users.id (actor)
related_post_id	int	NULL	Foreign key -> posts.id
related_comment_id	int	NULL	Foreign key -> comments.id
related_message_id	int	NULL	Foreign key -> messages.id
content	text	NULL	Notification message
is_read	tinyint(1)	0	Read flag
read_at	timestamp	NULL	Read timestamp
created_at	timestamp	CURRENT_TIMESTAMP	Creation timestamp

Table 0.3.1.12 post_media Table

Column Name	Type	Default	Description
id	int	AUTO_INCREMENT	Primary key

Column Name	Type	Default	Description
post_id	int	NOT NULL	Foreign key -> posts.id
media_url	varchar(500)	NOT NULL	Media file path
media_type	enum	'image'	image / video
file_name	varchar(255)	NULL	File name
file_size	int	NULL	File size in bytes
mime_type	varchar(100)	NULL	MIME type
display_order	int	0	Display order
created_at	timestamp	CURRENT_TIMESTAMP	Creation timestamp
updated_at	timestamp	CURRENT_TIMESTAMP	Update timestamp

Table 0.3.1.13 post_tags Table

Column Name	Type	Default	Description
id	int	AUTO_INCREMENT	Primary key
post_id	int	NOT NULL	Foreign key -> posts.id
tag_id	int	NOT NULL	Foreign key -> tags.id
created_at	timestamp	CURRENT_TIMESTAMP	Creation timestamp

Table 0.3.1.14 posts Table

Column Name	Type	Default	Description
id	int	AUTO_INCREMENT	Primary key
user_id	int	NOT NULL	Foreign key -> users.id
content	text	NOT NULL	Post content
category	varchar(50)	'general'	Post category
created_at	timestamp	CURRENT_TIMESTAMP	Creation timestamp

Column Name	Type	Default	Description
updated_at	timestamp	CURRENT_TIMESTAMP	Update timestamp
likes_count	int	0	Number of likes
original_post_id	int	NULL	Foreign key -> posts.id (quote/repost)

Table 0.3.1.15 saved_posts table

Column Name	Type	Default	Description
user_id	int	NOT NULL	PK + Foreign key -> users.id
post_id	int	NOT NULL	PK + Foreign key -> posts.id
created_at	timestamp	CURRENT_TIMESTAMP	Creation timestamp

Table 0.3.1.16 search_logs Table

Column Name	Type	Default	Description
id	int	AUTO_INCREMENT	Primary key
user_id	int	NULL	Foreign key -> users.id
search_query	varchar(255)	NOT NULL	Search keyword
results_count	int	0	Number of results returned
clicked_result_id	int	NULL	Foreign key -> posts.id (clicked)
search_filters	json	NULL	Applied search filters
user_ip	varchar(45)	NULL	User IP address
user_agent	text	NULL	User agent string
created_at	timestamp	CURRENT_TIMESTAMP	Creation timestamp

Table 0.3.1.17 study_resources Table

Column Name	Type	Default	Description
id	int	AUTO_INCREMENT	Primary key
user_id	int	NOT NULL	Foreign key -> users.id
title	varchar(255)	NOT NULL	Resource title
course_code	varchar(50)	NOT NULL	Course code
category	varchar(50)	NOT NULL	Resource category
file_url	varchar(255)	NOT NULL	File path
file_type	varchar(20)	NOT NULL	File type
file_size	int	NOT NULL	File size in bytes
created_at	timestamp	CURRENT_TIMESTAMP	Creation timestamp
updated_at	timestamp	CURRENT_TIMESTAMP	Update timestamp

Table 0.3.1.18 tags Table

Column Name	Type	Default	Description
id	int	AUTO_INCREMENT	Primary key
name	varchar(50)	NOT NULL	Tag name (unique)
slug	varchar(60)	NOT NULL	URL slug (unique)
description	text	NULL	Tag description
usage_count	int	0	Usage count
color	varchar(7)	'#3B82F6'	Display color code
is_featured	tinyint(1)	0	Featured flag
created_at	timestamp	CURRENT_TIMESTAMP	Creation timestamp
updated_at	timestamp	CURRENT_TIMESTAMP	Update timestamp
category_id	int	NULL	Foreign key -> categories.id

Table 0.3.1.19 trending_topics Table

Column Name	Type	Default	Description
id	int	AUTO_INCREMENT	Primary key
topic_name	varchar(100)	NOT NULL	Topic name
topic_type	enum	'keyword'	tag / category / keyword
mention_count	int	0	Mention count
trend_score	decimal(8,4)	0.0000	Trend score
trend_date	date	NOT NULL	Trend date
created_at	timestamp	CURRENT_TIMESTAMP	Creation timestamp
updated_at	timestamp	CURRENT_TIMESTAMP	Update timestamp

Table 0.3.1.20 user_tag_weights Table

Column Name	Type	Default	Description
user_id	int	NOT NULL	PK + Foreign key -> users.id
tag_id	int	NOT NULL	PK + Foreign key -> tags.id
weight	int	1	Interest weight
updated_at	timestamp	CURRENT_TIMESTAMP	Update timestamp

Table 0.3.1.21 users Table

Column Name	Type	Default	Description
id	int	AUTO_INCREMENT	Primary key
email	varchar(100)	NOT NULL	Email address (unique)
username	varchar(50)	NULL	Username (unique)
full_name	varchar(100)	NULL	Full name
preferred_topics	varchar(255)	NULL	Preferred topics

Column Name	Type	Default	Description
profile_image	varchar(255)	NULL	Profile image path
bio	varchar(255)	NULL	User biography
profile_completed	tinyint(1)	0	Profile completion flag
password	varchar(255)	NOT NULL	Hashed password
created_at	timestamp	CURRENT_TIMESTAMP	Creation timestamp
updated_at	timestamp	CURRENT_TIMESTAMP	Update timestamp
is_private	tinyint(1)	0	Private account flag
fcm_token	varchar(512)	NULL	FCM push notification token

Database Design Considerations

The system's database design is structured to prioritize data normalization, strong relationships, data integrity, performance, and security, all of which are essential for supporting advanced application features. The users table serves as the central hub, linking directly to key modules like posts, comments, follows, and messages. We implemented one-to-many and many-to-many relationships using foreign keys and junction tables (such as post_tags and conversation_participants) to keep the structure flexible and scalable.

To maintain strict data integrity, we applied constraints like NOT NULL, UNIQUE, and ENUM types where appropriate. Foreign key references are used extensively to prevent orphaned or invalid records, and standardized timestamp fields are included across tables to simplify tracking and auditing.

For performance optimization, frequently queried fields—such as user_id and post_id—are indexed. We also selectively denormalized certain attributes, like likes_count, to reduce the need for complex, heavy queries and to improve overall response times.

Security was another major consideration during the design phase. Sensitive information, particularly user passwords, is stored securely as hashes. The verification mechanisms also include built-in safeguards, such as expiry timestamps and attempt limits, to protect against abuse.

Finally, the database architecture is built to support advanced functionality. For example, the user_tag_weights table enables personalized recommendations, trending_topics facilitates trend analysis, and the combination of the conversations and messages tables provides a scalable messaging system. Ultimately, this design ensures the system is robust, efficient, secure, and ready for future expansion.

4.3.2 RESTful API Endpoint Design

A standardised and modular set of RESTful APIs was created and put into use to support the FICT360 platform's backend architecture. The service closely adheres to REST principles and was developed using Node.js and the Express framework. For efficient resource management, it uses common HTTP methods like GET, POST, PUT, and DELETE.

The endpoints are arranged logically around the application's primary functionality. User identification, profile and privacy control, social interactions (such as posts, comments, and followers), real-time messaging, sharing of study resources, and system alerts are all included in these functional areas.

During development, security and data handling were also important areas of concern. To stop unwanted access, authentication middleware is used on the majority of core routes. In the meantime, users can exchange documents and media with ease because file upload handlers are integrated directly into the routes that need them. Overall, this decoupled design makes it much simpler to maintain and grow the system in the future while maintaining the security of data exchange between the client and the database.

Table 0.3.2.1 System API Endpoints by Functional Category

Category	**Method**	**Endpoint**	**Description**
Authentication	POST	/api/auth/send-verification	Send email verification code
	POST	/api/auth/verify-email	Verify email address using code
	POST	/api/auth/register	Register a new user account
	POST	/api/auth/login	User login
	POST	/api/auth/forgot-password	Send forgot password verification code
	POST	/api/auth/verify-forgot-password	Verify the forgot password code
	POST	/api/auth/reset-password	Reset to a new password
User Profile	GET	/api/users/me	Get current logged-in user profile

	PUT	/api/users/complete-profile	Complete user profile setup
	PUT	/api/users/update-profile	Update profile information (with profile image upload)
	PUT	/api/users/privacy	Toggle private account setting
	PUT	/api/users/change-password	Change user password
	GET	/api/users/:id/profile	Get another user's profile
Follow & Relationship	POST	/api/users/:userId/follow	Follow a specific user
	DELETE	/api/users/:userId/unfollow	Unfollow a specific user
	POST	/api/users/respond-request	Respond to a pending follow request
	GET	/api/users/:id/followers	Get a user's followers list
	GET	/api/users/:id/following	Get a user's following list
Post & Content	GET	/api/posts/test	Debug route to validate database connectivity and tables
	GET	/api/posts/	Get all posts (feed)
	GET	/api/posts/search	Global search for posts
	GET	/api/posts/:id	Get a specific post by ID
	GET	/api/posts/user/:userId	Get posts by a specific user
	POST	/api/posts/	Create a new post (with media upload)
	PUT	/api/posts/:id	Update an existing post
	DELETE	/api/posts/:id	Delete a post
	POST	/api/posts/:id/like	Toggle like status on a post
	POST	/api/posts/:id/view	Record a view on a post
	GET	/api/posts/saved/me	Get current user's saved posts
	POST	/api/posts/:id/save	Toggle save/bookmark status on a post

	POST	/api/posts/:id/repost	Quote or repost an existing post
Comment	GET	/api/posts/:postId/comments	Get all comments for a specific post
	GET	/api/comments/:commentId	Get a specific comment details
	GET	/api/comments/:commentId/replies	Get nested replies for a specific comment
	GET	/api/users/:userId/comments	Get all comments made by a specific user
	POST	/api/posts/:postId/comments	Create a new comment on a post
	PUT	/api/comments/:commentId	Update an existing comment
	DELETE	/api/comments/:commentId	Delete a comment
	POST	/api/comments/:commentId/like	Toggle like status on a comment
Chat & Messaging	GET	/api/chat/mutual-followers	Get a list of mutual followers for chatting
	GET	/api/chat/conversations	Get the user's active conversation list
	POST	/api/chat/groups	Create a new group chat
	GET	/api/chat/groups/:id/messages/search	Search message history within a specific group
	GET	/api/chat/groups/:id	Get details of a specific group
	PUT	/api/chat/groups/:id/name	Update group chat name
	PUT	/api/chat/groups/:id/mute	Toggle group chat mute status
	POST	/api/chat/groups/:id/members	Add new members to a group chat
	PUT	/api/chat/groups/:id/avatar	Update group chat avatar (with image upload)
	DELETE	/api/chat/groups/:id/members/:memberId	Remove a member from the group chat
	DELETE	/api/chat/groups/:id/leave	Leave a group chat
	GET	/api/chat/:userId/messages	Get message history for a direct or group conversation
	POST	/api/chat/:userId/messages	Send a text message

	POST	/api/chat/:userId/messages/media	Send a media message (with file upload)
	GET	/api/chat/:targetId/messages/search	General search for messages within a conversation
Study Resources	GET	/api/resources/categories	Get all resource categories
	GET	/api/resources/categories/:categoryId/tags	Get tags associated with a specific category
	GET	/api/resources/tags	Get all resource tags
	GET	/api/resources/	Get all study resources (supports searching)
	POST	/api/resources/upload	Upload a new study resource document
Notification	GET	/api/notifications/	Get user notifications
	PUT	/api/notifications/mark-read	Mark all notifications as read
	POST	/api/notifications/update-token	Update FCM device token for push notifications

Chapter 5

System Implementation

5.1 Hardware Setup

An Android mobile device and a personal computer make up the majority of the the hardware involved in this project. The development computer is the primary workstation. Reference research, database design, and mobile frontend and backend server code are all done with it. The entire full-stack development process will be more effective and efficient with a dedicated laptop.

An Android smartphone is used for deployment and practical evaluation of the FICT360 mobile application. To make sure that the program is really user pleasant in a real-world context, it is crucial to test the UI's responsiveness on an actual mobile device. To guarantee reliable test results, the smartphone's operating system is kept up to date. Furthermore, the device's RAM and internal storage capacity easily above the system's basic needs, enabling the application—which includes features for real-time conversation and media rendering—to run without any unexpected crashes or performance bottlenecks.

Table 0.1.1 Specifications of laptop

Component	**Specifications**
Model	Legion 5 Pro 16IRX10
Processor	Intel® Core™ i9-14900HX
Operating System	Microsoft Windows 11
Graphic	8-core GPU
Memory	32GB
Storage	1TB SSD

Table 0.1.2 Specifications of Android Device

Component	**Specifications**
Model	VIVO IQOO 12
Chipset	Qualcomm SM8650-AB Snapdragon 8 Gen 3 (4 nm)
Operating System	Android 14
RAM	16GB
Storage	512GB
Resolution (pixels)	1230 x 2800

Table 0.1.3 Specifications of simulator

Component	**Specifications**
Model	Medium Phone API 36.0
Processor	x86_64, arm64-v8a
Operating System	Android 16.0
RAM	2GB
Storage	6GB
Resolution (pixels)	1080 x 2400

5.2 Software Setup

The software environment configured for developing the FICT360 platform comprises several specialized tools, frameworks, and programming languages. These

technologies were specifically selected based on their performance, scalability, and suitability for building a real-time campus community application.

Table 0.2.1 Requirement Tools to Use

Description	Requirement
Language	Dart Programming Language
Software	Visual Studio 2022, Android Studio, Flutter, Android Software Development Kits (SDK), MySQL
Database	MySQL

Visual Studio Code / Android Studio

An Integrated Development Environment (IDE) is essential for efficient coding. Visual Studio Code (and Android Studio for device emulation) provides a comprehensive workspace that helps in designing, writing, and debugging the application all in one place. These IDEs offer built-in tools for syntax highlighting, error checking, and terminal access, which significantly streamline both the frontend and backend development processes.

Flutter

Flutter is an open-source UI software development kit created by Google. It is utilized in this project to build the cross-platform mobile application using a single codebase. Flutter allows for the rapid creation of highly customized and visually appealing user interfaces. A key advantage during this project's development was its "Hot Reload" feature, which provides real-time previews of UI changes, drastically speeding up the coding and testing cycle.

Dart

Dart is the foundational programming language used alongside the Flutter framework. Developed by Google, it is an object-oriented language designed for building fast and scalable applications. It is relatively straightforward to learn while

offering modern features that increase developer productivity. In this project, Dart handles all the client-side business logic, state management, and API data fetching.

Node.js & Express.js

For the server-side architecture, Node.js is utilized as the backend runtime environment. Combined with the Express.js framework, it provides a robust and scalable foundation for building the RESTful APIs required by the application. This backend setup handles critical operations such as user authentication (JWT), executing the weighted recommendation logic, and processing media file uploads.

MySQL

MySQL serves as the primary Relational Database Management System (RDBMS) for this project. It is responsible for securely storing and managing all structured data generated by the platform. This includes user profiles, post metadata, relational weights for the recommendation system, and chat histories, ensuring data integrity and fast query retrieval.

Socket.io

To achieve the instant messaging requirements of the application, Socket.io is integrated into both the frontend and backend. This library enables real-time, bi-directional, and event-based communication between the mobile client and the server, making features like private messaging and group chats possible without the need for the user to manually refresh the page.

5.3 Settings and Configuration

Setting up Developer Mode on Android Device

To test the FICT360 application on a physical device, the developer mode must be activated.

1. On your Android device, tap the “Settings” app.

2. Tap "About phone" after scrolling all the way down. Until you receive a notification stating you are now a developer, repeatedly tap the build number.

3.Return to the Settings menu and select "Developer options."

4. Activate "Wireless debugging" or "USB debugging." In order to install apps and do real-time testing, this enables the device to connect to the IDE (Visual Studio Code or Android Studio).

5. You will see a warning prompt, tap “Allow” for allowing the laptop and the mobile device to connect to each other**.**

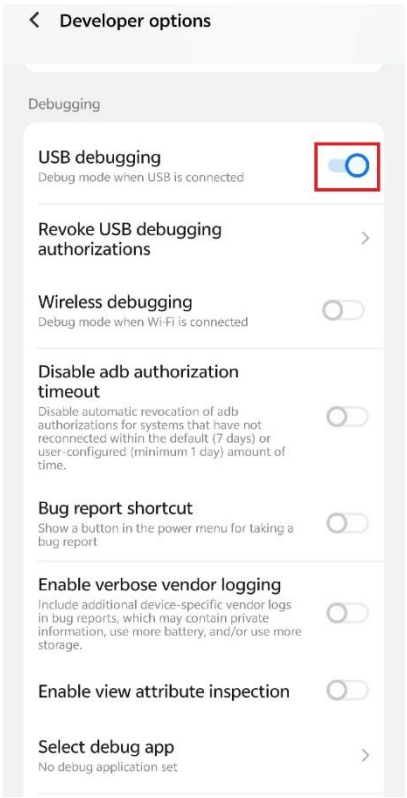


Figure 0.3.1 Screenshot of USB Debugging

Setting up the Backend Server (Node.js & Socket.io)

The backend server requires specific configurations to handle APIs and real-time messaging.

1. Open the backend project folder using Visual Studio Code.
2. Open the terminal and execute the command npm install. This command automatically downloads and installs all required dependencies listed in the package.json file, such as express, mysql2, jsonwebtoken, and socket.io.

```
PS C:\projects\backend> npm install

up to date, audited 323 packages in 3s

41 packages are looking for funding
  run `npm fund` for details
```

Figure 0.3.2 Screenshot of Command "npm install"

3. Create a .env (environment variables) file in the root directory. This file is used to securely store sensitive configuration details.
4. Configure the variables inside the .env file, including the server PORT, JWT Secret Key, and MySQL database connection credentials (Host, User, Password, Database Name).

```
.env

1   PORT=3000
2   DB_HOST=localhost
3   DB_USER=root
4   DB_PASSWORD=Utar@fict360
5   DB_NAME=fict360_db
6   JWT_SECRET=FICT360_MyFirstApp_2025_SuperSecretKey_12345!@#
7   JWT_EXPIRE=7d
8   EMAIL_USER=cheeqingtong2004@gmail.com
9   EMAIL_PASSWORD=kchdbcpdehdmuazj
```

Figure 0.3.3 Screenshot of .env file

5. Start the server by running the command node server.js or npm start in the terminal. A message will appear indicating the server and Socket.io are running successfully.

```
PS C:\projects\backend> npm start

> backend@1.0.0 start
> node server.js

Database connected successfully
Email service ready
```

Figure 0.3.4 Screenshot of "npm start"

Setting up the Database (MySQL)

The relational database must be configured before the application can store or retrieve any data.

1. Start the "MySQL" services from the control panel. (Task Manager -> Services -> MySQL80 -> Click “Start”)
2. Open the MySQL administration tool (MySQL Workbench).
3. Create a new, empty database named fict360_db.
4. Write the SQL query to create required tables.

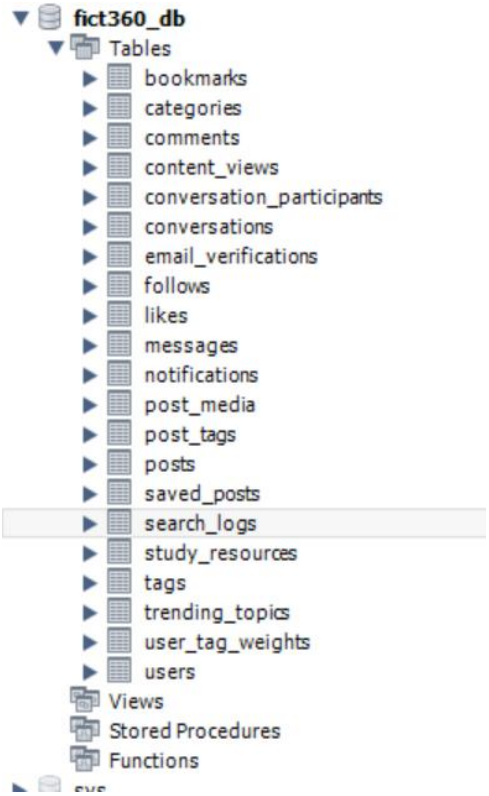


Figure 0.3.5 Screenshot of Database "fict360_db"

Push Notification Setup (Firebase Cloud Messaging)

To enable real-time notifications for the application, **Firebase Cloud Messaging (FCM)** was integrated.

1. Go to the Firebase Console, create a new project named “fict360”, and generate new private key with JSON format, renamed as “firebase-service-account.json” and put it in the backend folder.

2. Install the Firebase Command Line Interface on the backend terminal using “npm install -g firebase-tools” and log in via "firebase login".

3. Run the command “dart pub global activate flutterfire_cli” followed by “flutterfire configure” in the Flutter project root directory. This automatically registers the Android apps and generates the “firebase_options.dart” file.

4. Add “firebase_core” and “firebase_messaging” dependencies to pubspec.yaml, then call Firebase.initializeApp() in the main() function of the Flutter app.

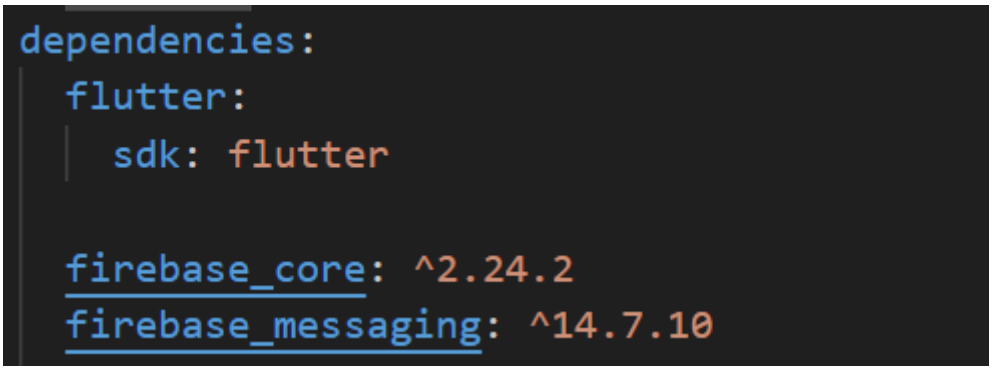


Figure 0.3.6 Screenshot of Firebase Package

```
void main() async {
  WidgetsFlutterBinding.ensureInitialized();

  await Firebase.initializeApp(
    options: DefaultFirebaseOptions.currentPlatform,
  );

  FirebaseMessaging.onBackgroundMessage(_firebaseMessagingBackgroundHandler);

  runApp(const MyApp());
}
```

Figure 0.3.7 Screenshot of Firebase.initializeApp()

Configuring the Frontend Project (Flutter)

The mobile client must be configured to connect properly with the local backend server.

1. Open the Flutter frontend project folder in Visual Studio Code.
2. Open the pubspec.yaml file to verify the installed packages. Run flutter pub get in the terminal to download essential packages.

```yaml
dependencies:
  flutter:
    sdk: flutter

  firebase_core: ^2.24.2
  firebase_messaging: ^14.7.10
  local_auth: ^2.1.3
  cupertino_icons: ^1.0.8
  socket_io_client: ^3.1.4
  video_thumbnail: ^0.5.6
  visibility_detector: ^0.4.0+2
  file_picker: ^11.0.2
  url_launcher: ^6.3.2
  http: ^1.1.0
  image_picker: ^1.0.5
  shared_preferences: ^2.2.2
  video_player: ^2.8.1
  video_compress: ^3.1.2
  path: ^1.8.3
  mime: ^1.0.4
  audioplayers: ^6.6.0
  record: ^6.2.0
  path_provider: ^2.1.5
  permission_handler: ^12.0.1
  flutter_launcher_icons: ^0.14.4

dev_dependencies:
  flutter_test:
    sdk: flutter
  flutter_lints: ^5.0.0
```

Figure 0.3.8 Screenshot of Flutter Essential Packages

3. Navigate to the API configuration file (api_config.dart). Modify the BASE_URL and SOCKET_URL to match the IPv4 address of the laptop hosting the Node.js server. This ensures the mobile device can communicate with the local server over the same Wi-Fi network.

lib > constants > api_config.dart > ApiConfig

```dart
// lib/constants/api_config.dart

class ApiConfig {

  static const String _hostIp = "192.168.123.9";
  static const String _port = "3000";

  static const String baseUrl = "http://$_hostIp:$_port/api";

  static const String socketUrl = "http://$_hostIp:$_port";

  static const String imageUrl = "http://$_hostIp:$_port/uploads/";
}
```

Figure 0.3.9 Screenshot of "api_config.dart"

4. Select the connected Android device in the IDE and press "Run" to build the APK and launch the FICT360 application on the phone.

```
PS C:\projects\fict360> flutter run
Connected devices:
V2307A (mobile)             • 10ADCC2JAN00229 • android-arm64  • Android 15 (API 35)
sdk gphone64 x86 64 (mobile) • emulator-5554   • android-x64    • Android 16 (API 36) (emulator)
Windows (desktop)           • windows         • windows-x64    • Microsoft Windows [Version 10.0.26200.8246]
Chrome (web)                • chrome          • web-javascript • Google Chrome 147.0.7727.138
Edge (web)                  • edge            • web-javascript • Microsoft Edge 147.0.3912.98
[1]: V2307A (10ADCC2JAN00229)
[2]: sdk gphone64 x86 64 (emulator-5554)
[3]: Windows (windows)
[4]: Chrome (chrome)
[5]: Edge (edge)
Please choose one (or "q" to quit):
```

Figure 0.3.10 Screenshot of "flutter run" Select Device

5.4 System Operation

"FICT360" is the name of the program, and Figure 5.4.1's logo depicts a networked circle. The primary purpose of the moniker "FICT360" is to draw attention to an all-encompassing digital community for FICT students. This mobile application primarily enables real-time communication, resource sharing, and the discovery of personalised academic information.

Figure 0.4.1 Logo of "FICT360" Mobile Application

Users must first authenticate by entering their registered email address and password on the Login page in order to use the mobile application

Figure 0.4.2 Login Page

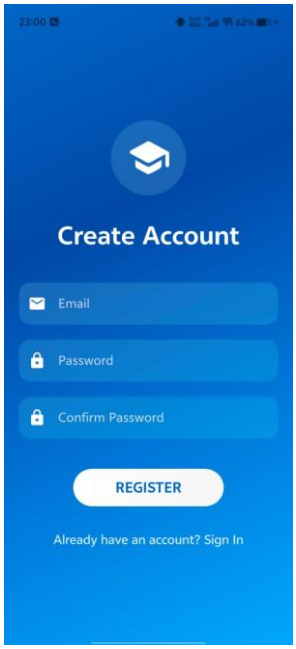


Figure 0.4.3 Register Page

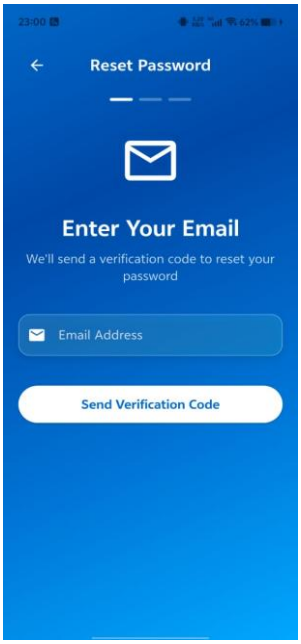


Figure 0.4.4 Reset Password Page

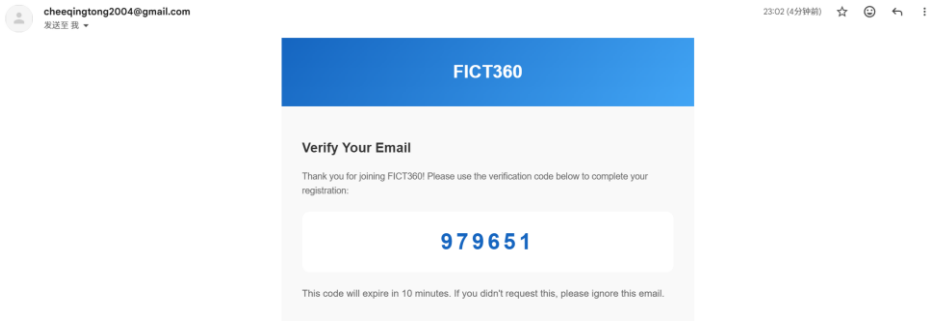


Figure 0.4.5 Email with 6-digit Verification Code (For Register)

The program allows new users to safely create an account. Prospective users are required to create a password and provide a working email address on the Sign-Up page. After you submit, the system will create a six-digit verification number and send it to the address you provided. To confirm their identification, the user must input this code into the program. The user can then access the application after completing the account registration process and verifying the six-digit verification code.

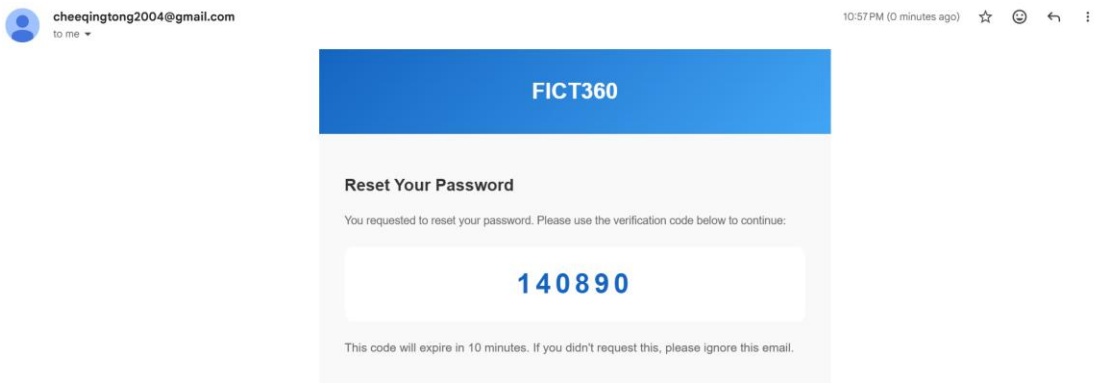


Figure 0.11 Email with 6-digit Verification Code (For Reset Password)

A user can begin the process of recovering their account if they forget their password. The user must first input the email address linked to their account. A six-digit verification number will then be sent to that email address by the system. The user is sent to the Reset Password page, where they may create and verify a new password, if the received code is correctly entered and verified. The user will then be instantly sent by the system to the login page after saving their new password.

5.4.1 Home Page

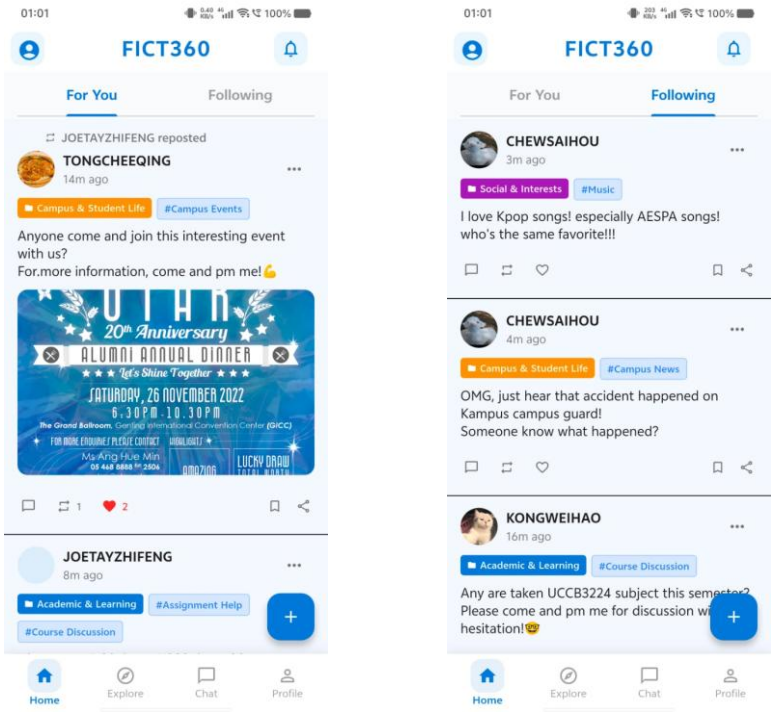


Figure 5.4.1.1 "For You"/"Following" Home Page

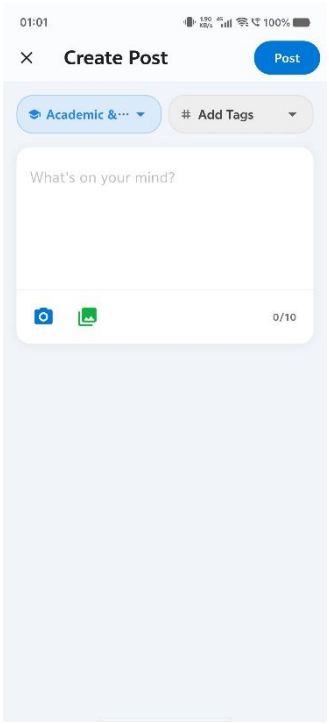


Figure 0.4.1.2 Create Post Page

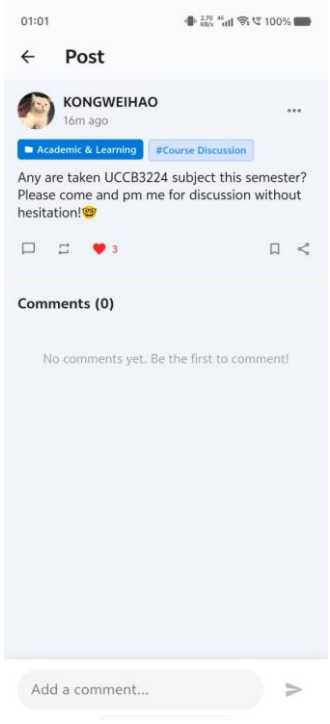


Figure 0.4.1.3 Post Details Screen

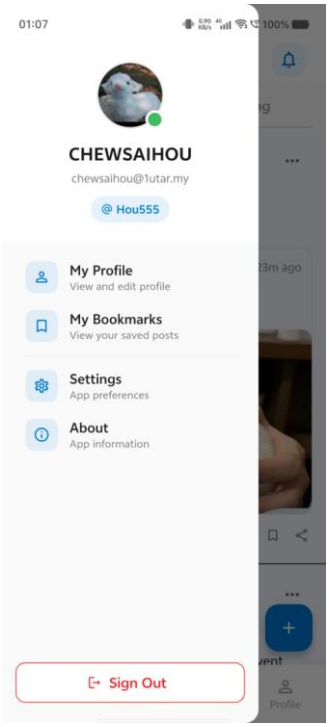


Figure 0.4.1.4 Onboarding Page

As seen in Figure 5.4.7, visitors who successfully log in are taken to the main home page, which has a centralised feed separated into "For You" and "Following" tabs. Users may peruse a variety of postings from academic conversations to campus activities that are tagged with particular categories and hashtags inside this feed. By offering user-friendly icons for commenting, reposting, like, bookmarking, and sharing the material, each post card facilitates immediate interaction. As shown in Figure 5.4.8, users may share their own updates by tapping the floating '+' icon to open the Create Post page. Here, they may choose a suitable category, add media files, enter text, and give tags using the system. Additionally, clicking on any particular post displays the detailed view shown in Figure 5.4.9, where viewers may read the entire text and use the bottom comment box to take part in debates. Lastly, as seen in Figure 5.4.10, the user can click on the onboarding screen to take immediate action.

5.4.2 Explore Page

Figure 0.4.2.1 Explore Page

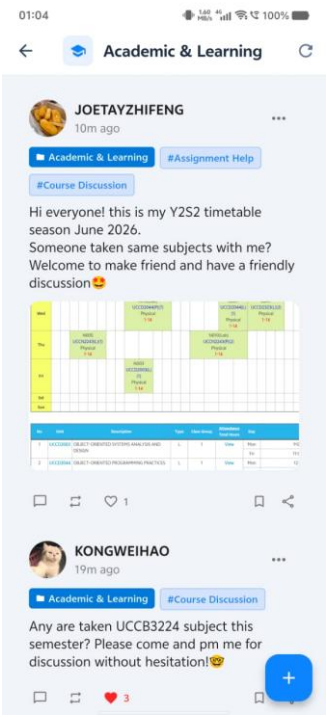


Figure 0.4.2.2 Browse Category Page

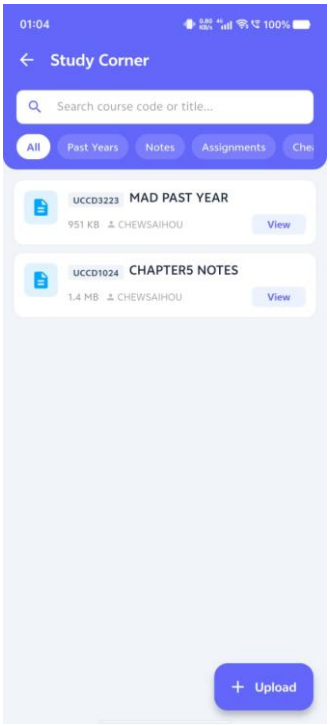


Figure 0.4.2.3 Resource Hub Page

To help users navigate the diverse content within the application, the Explore page serves as a centralized discovery hub, as shown in Figure 5.4.11. At the top, a versatile search bar allows students to quickly look up specific posts, topics, or peers. The page is primarily organized into a "Browse Categories" section, featuring distinct areas such as Academic & Learning, Campus & Student Life, and Social & Interests. Tapping on any of these category cards directs the user to a curated feed containing only relevant discussions, as demonstrated in Figure 5.4.12. Additionally, the bottom section of the Explore page introduces a dedicated "Study Corner" tailored for academic collaboration. As illustrated in Figure 5.4.13, this resource-sharing space allows students to search by course code or title, and filter uploaded documents—such as past year papers and lecture notes—using intuitive horizontal tabs. Students can either view existing study materials shared by others or contribute their own academic resources by tapping the floating upload button at the bottom right.

5.4.3 Search Page

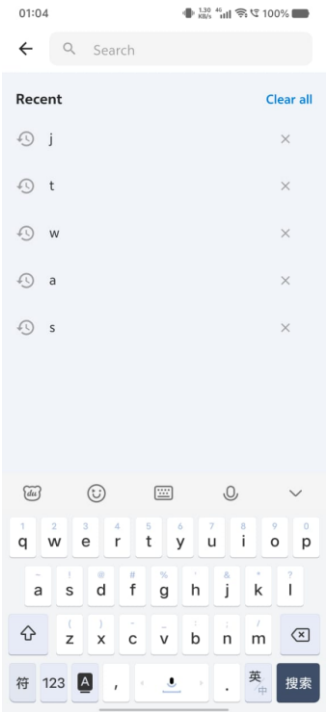


Figure 0.4.3.1 Search History Page

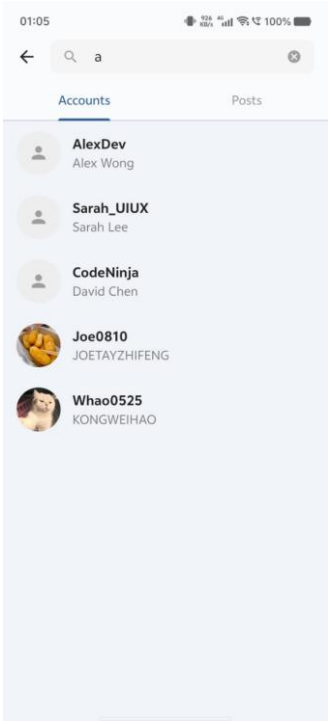


Figure 0.4.3.2 Search Result (Accounts) Page

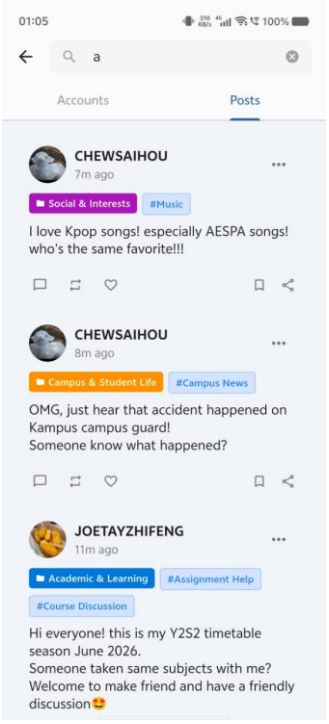


Figure 0.4.3.3 Search Result (Posts) Page

The search feature, which is built right into the Explore page, may be used by touching the search box at the top of the screen. As seen in Figure 5.4.14, the program initially displays the user's recent search history, along with a "Clear all" option for improved convenience and privacy control. The system dynamically gathers pertinent results and arranges them into two separate tabs: "Accounts" and "Posts" once a user enters a certain keyword. The Accounts tab makes it simple for students to locate and establish connections with particular peers by filtering the database to show matched user profiles, as shown in Figure 5.4.15. As an alternative, users may browse through specific conversations, announcements, and shared material that contain the searched phrase by selecting the Posts tab, which is shown in Figure 5.4.16. Users may easily switch between seeking particular information and locating individuals thanks to this dual-tab technique.

5.4.4 Chat Page (Private Chat/Group Chat)

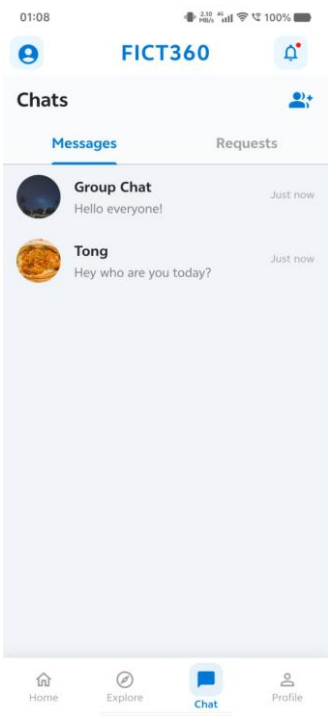


Figure 0.4.4.1 Chat List Page

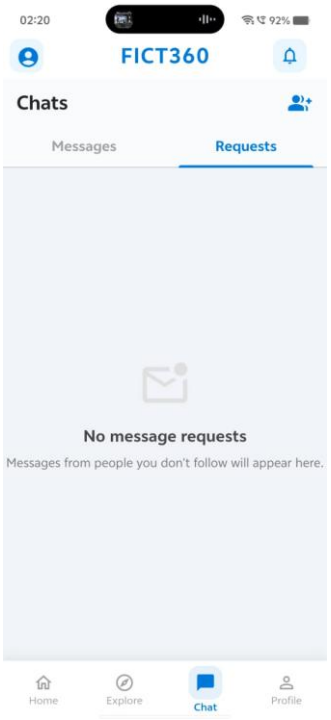


Figure 0.4.4.2 Chat Request Page

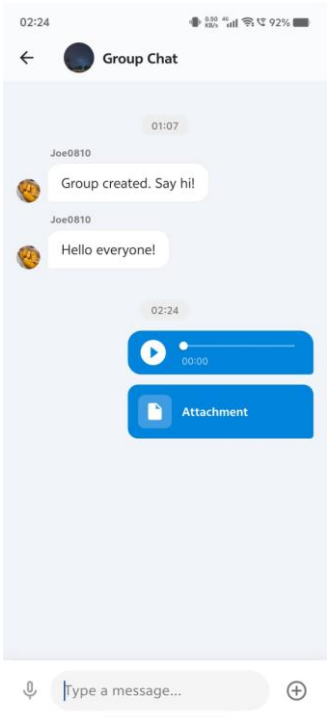


Figure 0.4.4.3 Chat Room Page

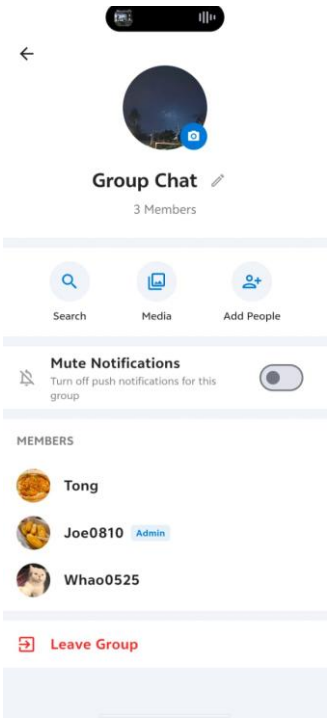


Figure 0.4.4.4 Group Details Page

For both private and group discussions, the Chat interface acts as the main core for communication. The main directory is cleanly divided into two tabs, "Messages" for current conversations and "Requests" for incoming messages from unfollowed accounts, as seen in Figures 5.4.17 and 5.4.18, guaranteeing a clutter-free inbox. In order to improve chat management, users may mute incoming alerts, pin the chat to the top, and remove the chat history by long-pressing any conversation thread on this list.

Once inside a chat room, as depicted in Figure 5.4.19, the application supports a variety of multimedia communication formats. Beyond standard text messaging, users can seamlessly send real-time voice recordings and document attachments. For group chats, the system includes a dedicated management screen accessible by tapping the chat header (Figure 5.4.20). This settings page empowers users to customize the group's name and profile picture, search through past messages, review shared media, invite new participants, toggle push notifications, and monitor member roles, such as group administrators.

5.4.5 Profile Page

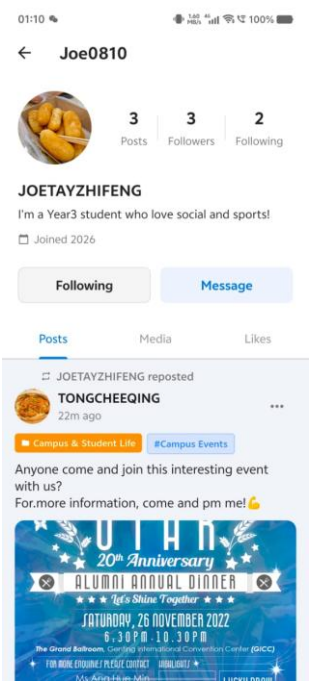


Figure 0.4.5.1 Other User Profile Page

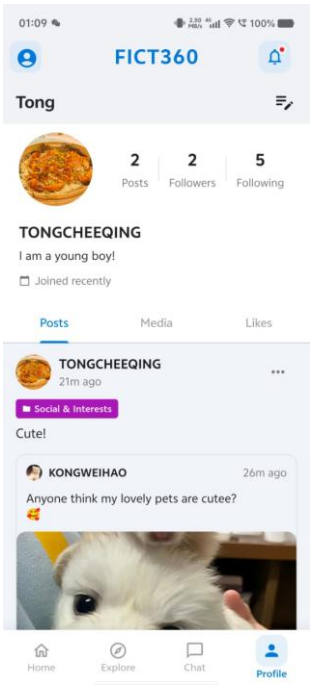


Figure 0.4.5.2 Profile Page

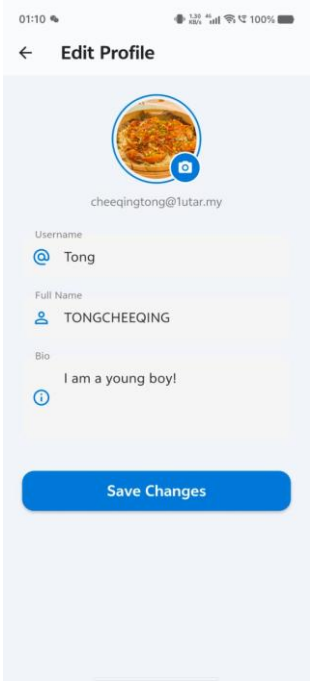


Figure 0.4.5.3 Edit Profile Page

The Profile page functions as a customised online environment where users may manage social relationships, create their own identities, and examine past exchanges. A user's profile photo, username, biography, and platform join date are shown in the main view of their own profile, as seen in Figure 5.4.21. Three important statistical metrics—the total number of published posts, accumulated followers, and the number of accounts the user is presently following—are prominently displayed at the top to offer instant insight into their community activity.

To maintain the accuracy of their personal information, users can access the Edit Profile screen (Figure 5.4.22). This dedicated interface features straightforward text input fields and image upload mechanisms. It grants users the flexibility to seamlessly update their profile picture, modify their unique username, adjust their full name, and rewrite their biography to reflect current interests. Tapping the “Save Changes” button ensures the new details are immediately updated across the application.

When visiting to the profile of another peer, the program dynamically modifies its layout. When reading someone else's page, key interaction buttons take the place of editing tools, as seen in Figure 5.4.23. The two users' relationship status is the only factor that determines whether these buttons are shown. The interface strictly displays a single "Follow" button to promote a new connection if the user has not yet followed the viewed account. In contrast, the interface extends to provide a direct "Message" button and a "Following" sign if the user is already following the account, allowing for easy private contact.

Furthermore, to keep the profile page organized, a standardized horizontal tab system is implemented at the bottom half of the screen. These tabs—categorized into Posts, Media, and Likes—act as intuitive filters. By tapping them, users can effortlessly toggle between viewing chronological text-based publications, browsing multimedia uploads, or exploring content they have previously liked.

5.4.6 Settings Page

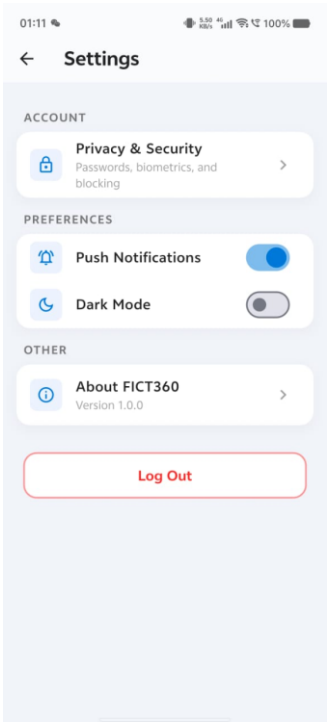


Figure 0.4.6.1 Settings Page

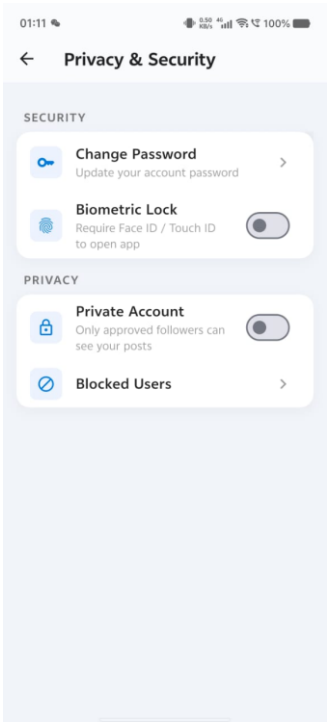


Figure 0.4.6.2 Settings (Privacy & Security) Page

Users have centralised control over their security settings, application preferences, and account management through the Settings page. The primary interface is separated into logical sections for easy navigation, as seen in Figure 5.4.24. Users may easily adjust the frequency of their alerts by toggling push notifications under the "Preferences" section. In a similar vein, a dedicated "Dark Mode" setting enables immediate visual customisation by modifying the application's UI to lessen eye strain in dimly lit areas. The application version details and a prominently displayed "Log Out" button for safe session termination are located at the bottom of the screen.

To access deeper account controls, users can tap the "Privacy & Security" menu, which navigates them to a detailed configuration screen illustrated in Figure 5.4.25. This secondary page is crucial for maintaining account integrity and is subdivided into Security and Privacy controls. Within the Security section, users are given the option to update their existing password or enable a "Biometric Lock". This specific feature

leverages native device hardware, such as Face ID or Touch ID, adding an essential layer of physical authentication before the application can be opened.

Users have tight control over their digital footprint and social boundaries thanks to the Privacy area. Users may create a more secure and regulated sharing environment by limiting the visibility of their published posts to just authorised followers by turning on the "Private Account" toggle. In order to provide users complete control over their interactions on the platform, the "Blocked Users" directory also offers a special area where users may examine, manage, and unblock previously banned accounts.

5.4.7 Other Page (Notification/Bookmark)

Figure 0.4.7.1 Notification Page

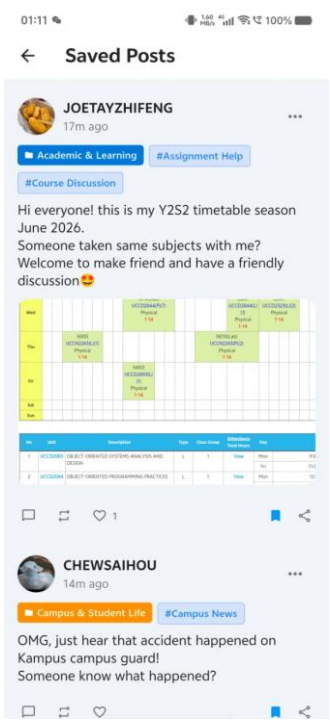


Figure 0.4.7.2 Bookmark Page

The Notifications screen (Figure 5.4.26) provides a centralized chronological feed of recent social interactions relevant to the user. This interface displays real-time alerts for incoming messages, post likes, and new followers, utilizing blue dot indicators to clearly highlight unread activities.

The Saved Posts shown as Figure 5.4.27 interface functions as a personal archive allowing users to easily retrieve content, they have previously bookmarked. It organizes

these selected items into a continuous scrolling feed, ensuring that important academic discussions and campus announcements remain readily accessible for future reference.

5.5 Implementation Issues and Challenges

1.Managing Complex App State and Optimistic UI Updates

Building a dynamic social platform required managing a lot of moving parts across different screens. Whenever a user liked a post, saved an item, or followed another user, the interface needed to reflect these changes instantly. Initially, waiting for the backend API to respond before updating the UI caused a noticeable lag. To solve this, I had to learn and implement "optimistic UI updates," where the app visually updates the interface immediately while handling the actual database request in the background. Managing these local states without causing data inconsistencies across the feed and profile pages proved to be a significant learning curve in Flutter.

2. Real-Time Chat Integration via WebSockets

Transitioning from standard REST APIs to real-time bidirectional communication was one of the biggest technical hurdles. Implementing the private and group chat features required setting up Socket.io. I faced several challenges in managing the persistent connections, especially when handling scenarios where the device temporarily lost internet access or when the app was pushed to the background. Ensuring that users joined the correct chat rooms, syncing the message history accurately, and preventing duplicate messages from rendering required extensive debugging and a deep dive into socket event lifecycles.

3.Handling Multimedia Uploads and Playback

Managing device memory and speed became a major concern because the program lets users exchange a wide range of material, including documents, music files, films up to 100MB, and photos. Complex challenges included creating dynamic video thumbnails and implementing a unique video player that didn't stall the main thread of the application. Additionally, managing device permissions for file storage and

microphones across various Android versions necessitated the creation of platform-channel code and cautious error handling to keep the program from crashing.

4.Relational Database Constraints and Testing

Highly complicated foreign key connections were required while designing the MySQL database architecture for a platform that included postings, tags, categories, user interactions, and chat logs. I regularly had to clear test data throughout the development and testing stages. However, because the tables are interrelated, straight deletion frequently resulted in foreign key constraint issues (e.g., removing a post when comments or saved post entries still remained). Strict database connection design is crucial, as I discovered when I had to learn how to correctly handle database rollbacks and temporarily suspend constraint checks to truncate tables.

5.Security and Dynamic Access Control

Securing user data went beyond just hashing passwords during registration. I had to implement a robust token-based authentication system to protect the API routes. A specific challenge was handling dynamic access control, particularly in group chats. When a user was removed from a group or left voluntarily, the app needed to immediately update their local permissions so they could view past messages but no longer send new ones. Balancing these security checks between the Node.js backend and the Flutter frontend required careful logic design to prevent unauthorized access.

5.6 Conclusion Remark

I have outlined the crucial actions done to turn this mobile application's conceptual design into a working reality throughout this chapter. Every stage was essential to shaping the finished result, from configuring the system connections to setting up the local development environments and database architecture. A clear visual depiction of the user interfaces and essential features that have been effectively integrated may be found in the system operating screenshots shown in the previous sections.

As mentioned in the concerns and challenges section, the development process was not without difficulties, but in the end, these technological challenges led to a more reliable program. My exploration of sophisticated programming approaches was

prompted by tackling the challenges of real-time Socket.io connections, handling complicated multimedia states in Flutter, and resolving database relationship issues. The practical experience and knowledge acquired from overcoming these implementation obstacles will surely provide a solid basis for any upkeep, optimisation, or feature development of the program in the future.

Chapter 6

System Evaluation and Discussion

6.1 System Testing and Performance Metrics

The last steps to confirm the stability and dependability of the FICT360 mobile application are system testing and performance assessment. Verifying that the platform satisfies all predetermined design criteria and can withstand real-world usage without crashing was the primary goal of this phase. Since data mismatches or state management problems sometimes arise at these connection points when many operations occur simultaneously, special attention was paid to the interaction between the Flutter user interface and the database backend.

This application was built to offer university students a comprehensive digital community, featuring tools for creating multimedia posts, sharing study materials, exploring categorized feeds, and engaging in real-time private or group chats. Given the heavy reliance on continuous data fetching and complex user interface states, rigorous testing was necessary to iron out any functional bugs. The primary approach employed was functional and integration testing, driven by a detailed checklist that tracked every major user action and the corresponding system response.

This checklist contained specific test cases tailored to the app's core mechanics, ranging from verifying the instant delivery of Socket.io messages to ensuring that large image and document uploads did not freeze the application. Throughout the testing period, the outcomes of these test cases were carefully recorded. Whenever a defect was identified—such as a delayed UI update, a database query error, or a layout breaking on smaller screens—the code was immediately revised and retested. By systematically documenting and resolving these issues, the application was heavily optimized to guarantee a stable, secure, and smooth social networking experience for its end users.

6.2 Testing Setup and Result

Table 0.2.1 Login Page Testing

No	Test Phase / Item	Data	Expected Result	Actual Result	Pass/Fail
1	Valid Email and Password	Email: cheeqingtong@1utar.my Password: 000000	User redirect to home page.	User redirect to home page.	Pass
2	Invalid Email Format	Email: cheeqingtong Password: 000000	System displays error message: "Invalid email format."	System displays error message: "Invalid email format."	Pass
3	Valid Email but Incorrect Password	Email: cheeqingtong@1utar.my Password: wrongpass123	System displays error message: "Incorrect password."	System displays error message: "Incorrect password."	Pass
4	Unregistered Email	Email: unknown@1utar.my Password: 123456	System displays error message: "User not found."	System displays error message: "User not found."	Pass
5	Empty Email Field	Email: [Empty] Password: 000000	System prompts "Email is required."	System prompts "Email is required."	Pass
6	Empty Password Field	Email: cheeqingtong@1utar.my Password: [Empty]	System prompts "Password is required."	System prompts "Password is required."	Pass
7	Leading/Trailing Spaces in Email	Email: cheeqingtong@1utar.my Password: 000000	System trims spaces and logs in successfully. User redirected to home page.	System trims spaces and logs in successfully. User redirected to home page.	Pass

8	Password Visibility Toggle	Email: test@1utar.my Password: 123456	Clicking the "eye" icon shows/hides the password text.	Clicking the "eye" icon shows/hides the password text successfully.	Pass
9	Forgot Password Link	N/A	Clicking the link redirects the user to the Password Recovery page.	User redirected to Password Recovery page.	Pass

Table 0.2.2 Registration Page Testing

No	**Test Phase / Item**	**Data / Action**	**Expected Result**	**Actual Result**	**Pass/Fail**
Step 1: Initial Registration					
1.1	Valid Initial Input	Email: newuser@1utar.my Password: 123456 Confirm: 123456	System accepts input, sends a 6-digit OTP to the email, and navigates to the OTP Verification screen.	System sent OTP and navigated to Verification screen.	Pass
1.2	Invalid Email Format	Email: newuser.1utar.my Password: 123456 Confirm: 123456	System displays error: "Invalid email format."	System displays error: "Invalid email format."	Pass
1.3	Email Already Registered	Email: existing@1utar.my Password:	System displays error: "Email is	System displays error: "Email is	Pass

		123456 Confirm: 123456	already registered."	already registered."	
1.4	Passwords Do Not Match	Email: test@1utar.my Password: 123456 Confirm: 654321	System displays error: "Passwords do not match."	System displays error: "Passwords do not match."	Pass
Step 2: OTP Verification					
2.1	Correct 6-Digit OTP	Input: 123456 (Matching sent code)	Verification successful. System navigates to Profile Completion page.	Verification successful, navigated to Profile Completion.	Pass
2.2	Incorrect OTP	Input: 000000 (Mismatch)	System displays error: "Invalid verification code. Please try again."	System displays error: "Invalid verification code."	Pass
2.3	Incomplete OTP	Input: 1234 (Only 4 digits)	System prompts: "Please enter all 6 digits."	System prompts: "Please enter all 6 digits."	Pass
2.4	Resend OTP Function	Action: Click "Resend Code" button	System sends a new 6-digit code and restarts countdown timer.	System sent new code and timer restarted.	Pass
Step 3: Profile Completion					
3.1	Valid Profile Details	Username: AlexDev Full Name: Alex Wong	Profile updated successfully. User	Profile updated, user	Pass

			navigated to Home page.	navigated to Home page.	
3.2	Empty Mandatory Fields	Username: [Empty] Full Name: Alex Wong	System prompts: "Username is required."	System prompts: "Username is required."	Pass

Table 0.2.3 Reset Password Page Testing

No	**Test Phase / Item**	**Data / Action**	**Expected Result**	**Actual Result**	**Pass/Fail**
Step 1: Request Password Reset					
1.1	Valid Registered Email	Email: cheeqingtong@1utar.my	System sends a 6-digit OTP and navigates to OTP Verification screen.	System sent OTP and navigated to Verification screen.	Pass
1.2	Invalid Email Format	Email: cheeqing.1utar.my	System displays error message: "Invalid email format."	System displays error message: "Invalid email format."	Pass
1.3	Unregistered Email	Email: unknown@1utar.my	System displays error message: "Email not found."	System displays error message: "Email not found."	Pass
1.4	Empty Email Field	Email: [Empty]	System prompts: "Email is required."	System prompts: "Email is required."	Pass
Step 2: OTP Verification					
2.1	Correct 6-Digit OTP	Input: 123456	Verification successful, navigates to	Verification successful, navigated to	Pass

		(Matching sent code)	Create New Password screen.	Create New Password screen.	
2.2	Incorrect OTP	Input: 000000 (Mismatch)	System displays error message: "Invalid verification code."	System displays error message: "Invalid verification code."	Pass
2.3	Incomplete OTP	Input: 1234 (Only 4 digits)	System prompts: "Please enter all 6 digits."	System prompts: "Please enter all 6 digits."	Pass
2.4	Resend OTP Function	Action: Click "Resend Code" button	System sends new 6-digit code and restarts countdown timer.	System sent new code and timer restarted.	Pass
Step 3: Create New Password					
3.1	Valid New Password & Confirm	New Password: 654321 Confirm Password: 654321	System displays success message "Password reset successfully" and navigates to Login page.	System displayed success and navigated to Login page.	Pass
3.2	Passwords Do Not Match	New Password: 654321 Confirm Password: 111111	System displays error message: "Passwords do not match."	System displays error message: "Passwords do not match."	Pass
3.3	Password Too Short	New Password: 123 Confirm Password: 123	System displays error message: "Password must be at	System displays error message: "Password must be at	Pass

			least 6 characters."	least 6 characters."	
3.4	Empty Password Fields	New Password: [Empty] Confirm Password: [Empty]	System prompts: "Password is required."	System prompts: "Password is required."	Pass

Table 0.2.4 Create Post Testing

No	Test Item	Data / Action	Expected Result	Actual Result	Pass/Fail
1	Valid Text Only Post	Content: Test Post Category: Academic	System creates post and redirects to Home page.	System creates post and redirects to Home page.	Pass
2	Valid Post With Image	Content: Check this out Image: 1 attached	Post is created successfully with the attached image.	Post created successfully with the attached image.	Pass
3	Empty Content Field	Content: [Empty] Category: Campus	System displays error message: "Content is required."	System displays error message: "Content is required."	Pass
4	Exceed Media Limit	Content: Gallery Images: 11 attached	System only selects the first 10 attached images.	System selects the first 10 attached images.	Pass
5	Cancel Post Creation	Content: Draft Action: Click "Back" button	System prompts "Discard post?". Upon confirmation, post is not saved.	System prompts "Discard post?". Post is not saved.	Pass

Table 0.2.5 Browse Categories Page Testing

No	Test Item	Data / Action	Expected Result	Actual Result	Pass/Fail
1	Browse by Specific Category	Action: Click on a specific category tag (e.g., "Academic" or "Campus Life")	System filters the feed and only displays posts that are tagged with the	Feed successfully filtered and displayed posts matching the	Pass

			selected category.	selected category.	

Table 0.2.6 Resource Hub Page Testing

No	Test Item	Data / Action	Expected Result	Actual Result	Pass/Fail
1	Access Resource Hub	Action: Click the "Resource Hub" entry point	System fetches and displays a categorized list of available study materials.	Resource list loaded successfully from the database.	Pass
2	Download/View Document	Action: Click on a specific document (e.g., "Chapter_1_Notes.pdf")	System opens it in a google browser.	Document opened successfully.	Pass
3	Upload Valid Study Material	Action: Upload a 2MB PDF file and fill in the title	File is uploaded successfully, and the new resource appears in the Hub.	File uploaded and displayed in the Hub.	Pass
4	Filter Resources	Action: Apply a filter for a specific module or year	System updates the Resource Hub list to show only matching materials.	System correctly filtered and displayed materials.	Pass

Table 0.2.7 Global Search Page Testing

No	Test Item	Data / Action	Expected Result	Actual Result	Pass/Fail
1	Valid Keyword Search	Keyword: "UCCD3223"	System fetches and displays posts/resources containing the keyword.	System displayed relevant search results.	Pass

2	No Results Found	Keyword: "xyz123abc"	System displays a friendly "No results found" message.	System displayed "No results found" message.	Pass
3	Case-Insensitive Search	Keyword: "jAvA" (instead of "Java")	System ignores case and returns the correct results for "Java".	System returned correct results regardless of case.	Pass
4	Empty Search Submission	Action: Click the Search icon with an empty input field	System ignores the action	System ignored the action, no search triggered.	Pass
5	Clear Search History	Action: Click the "Clear" or "X" button in the search bar	Search input is cleared, and the screen returns to the default Explore view.	Input cleared and screen returned to default view.	Pass

Table 0.2.8 Private Chat Page Testing

No	**Test Item**	**Data / Action**	**Expected Result**	**Actual Result**	**Pass/Fail**
1	Send Real-time Text Message	Sender sends text: "Hi, do you have the notes?"	Message appears immediately on sender's screen and receiver receives it in real-time.	Message sent and received in real-time.	Pass
2	Send Empty Message	Click Send with no text	Send disabled or ignored	System ignored action	Pass
3	Send Image Attachment	Send valid image	Image uploaded and shown	Image uploaded and displayed	Pass
4	Retrieve Chat History	Open chat and scroll	Load previous messages	Messages loaded	Pass
5	Offline Message Delivery	Send while receiver offline	Saved and shown on login	Loaded on login	Pass

Table 0.2.9 Group Chat Page Testing

No	Test Item	Data / Action	Expected Result	Actual Result	Pass/Fail
1	Create New Group Chat	Name: Assignment Group, Members: A, B	Group created and redirect	Success	Pass
2	Broadcast Text Message	Send Hello team	All members receive	Received by all	Pass
3	Add New Member	Add user	User added with system msg	Added	Pass
4	Leave Group	Click leave	User removed	Removed	Pass
5	Group Chat History Loading	Scroll up	Load older messages	Loaded correctly	Pass

Table 0.2.10 Profile/Other Profile Page Testing

No	Test Item	Data / Action	Expected Result	Actual Result	Pass/Fail
1	View User Posts Activity	Action: Navigate to another user's profile page	System fetches and displays all posts created by that specific user in chronological order.	User's post activity displayed correctly.	Pass
2	Follow a User	Action: Click the "Follow" button on another user's profile	Button changes to "Unfollow" (or "Following"). The target user's follower count increases by 1.	User followed successfully and count updated.	Pass
3	Unfollow a User	Action: Click the "Unfollow" button on a	Button reverts to "Follow". The target user's follower	User unfollowed successfully	Pass

		currently followed user	count decreases by 1.	and count updated.	
4	Message Button Visibility & Action	Action: Follow a user, then click the newly visible "Message" button	System navigates the user directly to a private chat dialog with that specific user.	Message button appeared after following, successfully navigated to private chat.	Pass
5	View Followers List	Action: Click on the "Followers" count/text	System displays a list of all users who are currently following this profile.	Followers list loaded and displayed accurately.	Pass
6	View Following List	Action: Click on the "Following" count/text	System displays a list of all users that this profile is currently following.	Following list loaded and displayed accurately.	Pass

Table 0.2.11 Edit Profile Page Testing

No	**Test Item**	**Data / Action**	**Expected Result**	**Actual Result**	**Pass/Fail**
1	Update All Fields	Profile Image: new_avatar.jpg Username: AlexDev Full Name: Alex Wong Bio: "Software Engineering student."	Profile is updated successfully and changes are reflected immediately.	Profile updated successfully and changes reflected immediately.	Pass
2	Update Mandatory Fields Only	Username: AlexDev Full Name: Alex Wong Bio: [Empty] Image: No change	Profile is updated successfully without requiring a bio.	Profile updated successfully without requiring a bio.	Pass
3	Update with Duplicate Username	Username: ExistingName Full Name: Alex Wong	Profile is updated successfully (system allows duplicate usernames).	Profile updated successfully.	Pass

4	Update with Duplicate Full Name	Username: AlexDev Full Name: SameNameAsOthers	Profile is updated successfully (system allows duplicate full names).	Profile updated successfully.	Pass
5	Empty Username Field	Username: [Empty] Full Name: Alex Wong	System prevents saving and displays error: "Username is required."	System prevented saving and displayed error: "Username is required."	Pass
6	Empty Full Name Field	Username: AlexDev Full Name: [Empty]	System prevents saving and displays error: "Full name is required."	System prevented saving and displayed error: "Full name is required."	Pass
7	Cancel Profile Update	Action: Modify text fields and click the "Back" or "Cancel" button	Changes are discarded and previous profile information is retained.	Changes discarded and previous profile information retained.	Pass

Table 0.2.12 Internal Notification Page Testing

No	Test Item	Data / Action	Expected Result	Actual Result	Pass/Fail
1	New Follower Notification	Action: User A navigates to User B's profile and clicks "Follow".	User B receives an internal notification: "User A started following you." The unread badge count increases by 1.	Notification received successfully and unread count updated.	Pass
2	Post Comment Notification	Action: User A writes a comment on User B's post.	User B receives an internal notification: "User A commented on	Notification received successfully and unread count updated.	Pass

			your post." The unread badge count increases by 1.		
3	Post Like Notification	Action: User A clicks the "Like" button on User B's post.	User B receives an internal notification: "User A liked your post." The unread badge count increases by 1.	Notification received successfully and unread count updated.	Pass
4	New Private Message Notification	Action: User A sends a direct message to User B.	User B receives a chat notification (e.g., banner or chat tab badge) indicating a new message from User A.	Chat notification received and displayed accurately.	Pass
5	Notification Redirection (Post)	Action: User B clicks on the "Like" or "Comment" notification from the notification list.	System navigates User B directly to the specific post that was liked or commented on.	System successfully navigated to the corresponding post.	Pass
6	Notification Redirection (Profile)	Action: User B clicks on the "New Follower" notification.	System navigates User B directly to User A's profile page.	System successfully navigated to the follower's profile.	Pass

Table 0.2.13 Settings Page Testing

No	Test Item	Data / Action	Expected Result	Actual Result	Pass/Fail
1	Reset Password Entry	Action: Click on "Reset Password" option	System navigates user to the Reset Password verification screen.	Navigated to Reset Password screen successfully.	Pass
2	Toggle Biometric Lock (ON)	Action: Switch "Biometric Lock" to ON	System prompts for fingerprint/Face ID to confirm. Once enabled, next app launch requires	Biometric prompted and lock enabled successfully.	Pass

			biometric authentication.		
3	Toggle Private Account	Action: Switch "Private Account" to ON	Profile is set to private. Non-followers can no longer view the user's posts or following list.	Account set to private and visibility restricted correctly.	Pass
4	Manage Blocked Users	Action: Open "Blocked Users", select a user, and click "Unblock"	The selected user is removed from the blocked list and can resume viewing/interacting with the profile.	User unblocked and removed from the list successfully.	Pass
5	Toggle Push Notifications	Action: Switch "Push Notifications" to OFF	System stops sending OS-level background notifications (alerts/banners) to the user's device.	Push notifications successfully disabled at the system level.	Pass
6	Toggle Dark Mode	Action: Switch theme from Light to "Dark Mode"	The application's UI theme immediately updates to a dark color scheme across all screens without needing a restart.	UI theme updated to Dark Mode immediately.	Pass
7	View About FICT360	Action: Click on "About FICT360"	System displays the application version, developer information, and Terms of Service/Privacy Policy.	App information and policies displayed correctly.	Pass
8	Logout Functionality	Action: Click "Logout" and confirm the prompt	User session is securely terminated, local cache is cleared, and user is redirected to the Login page.	Session terminated and redirected to Login page.	Pass

6.3 Project Challenges

There were a number of logistical and technological challenges in turning FICT360 from a conceptual idea into a fully working mobile application. As a lone engineer in charge of the mobile front-end and back-end infrastructure, I came into issues with everything from file storage speed optimisation to real-time data synchronisation. Implementing the real-time private and group chat features was the biggest technological challenge. Instant messaging necessitated the integration of WebSockets to maintain permanent connections between the client and the server because standard HTTP requests were insufficient.

It was challenging to manage this permanent connection, especially when dealing with edge circumstances like unexpected network outages, suspending background apps, and making sure offline messages were appropriately queued and delivered upon reconnection. It took a lot of troubleshooting and a thorough reorganisation of the chat event listeners to fix these problems.

Managing multimedia and document uploads for the Study Corner and dynamic post feeds was another significant challenge. At first, enabling users to upload big PDF files and high-resolution photos resulted in perceptible slowness during data retrieval and rapidly burnt up server capacity. I had to apply stringent file size restrictions and format validations directly on the client side in order to fix this. Additionally, I made sure that users could upload and download academic materials without the user interface stalling by optimising the backend routing to process multipart form data more effectively.

On the front-end development side, maintaining a smooth user experience while rendering data-heavy screens was a persistent challenge. The application relies heavily on dynamic feeds, notification badges, and live comment updates. During early testing, frequent widget rebuilds in the Flutter framework caused the application to stutter during scrolling. Overcoming this required a deeper understanding of state management. I had to refine the architecture so that only specific components—such as a single 'like' button or an unread message counter—rebuilt upon state changes, rather than refreshing the entire screen.

Beyond the purely technical aspects, managing the project timeline alongside other academic responsibilities was a constant challenge. Balancing the intensive coding and testing phases of FICT360 with concurrent university coursework, midterm exams, and assignment deadlines required strict adherence to my initial Gantt chart. As development progressed, feature creep became a noticeable risk. I had to make difficult decisions to prioritize core academic and communication functionalities over secondary features to make sure the project was completed on schedule and met all fundamental objectives.

6.4 Objective Evaluation

One main purpose behind this effort involved creating a mobile tool to support student interaction and material exchange at universities. Achievement came by finishing FICT360, a functional digital space built around those needs. A set of straightforward utilities now serves individuals within academic settings. Categorized threads appear for viewing, while study files go into a collective section known as Study Corner - interaction follows via saved posts, reactions, or written notes. Beyond public areas, messaging happens live, whether one-on-one or in groups, making it simpler to arrange tasks, examine subject matter, or track activities happening across college grounds - all inside one system.

Early in the project, examination of established social and learning platforms guided both technical planning and interface decisions. Building on insights from Chapter 2, commonly adopted messaging tools were assessed for performance in higher education contexts. Their shortcomings informed key aspects of the app's development path. Merging informal interaction spaces with organized course material exchange emerged naturally from this analysis.

In total, every planned goal of the project was met without exception. Not only does the completed mobile app allow messaging, but it also supports broader aspects of academic living. Because features like organizing learning materials, instant peer connections, and tailored recommendations are built in, engagement feels natural and useful. Early analysis of current tools guided decisions so functionality matched what

students truly require. One result stands out: an accessible platform shaped by real usage patterns rather than assumptions.

Chapter 7

Conclusion and Recommendation

7.1 Conclusion

This project grew out of a real need: university students struggle to connect and share resources in a meaningful way. Campus life is busy, often chaotic, and it's easy for people to drift apart outside of lectures. FICT360 steps in by giving students a space all their own—a digital commons that stretches far beyond the classroom walls. With features like organized discussion boards and straightforward file-sharing, it actively encourages students to help each other out and stay engaged with campus life.

I set out to build a tool that makes day-to-day university life easier and more social. The finished platform does exactly that. It comes packed with an Explore feed to help students stay up-to-date, a Study Corner for focused collaboration, and real-time messaging to keep conversations flowing. It's not just a digital bulletin board. Students can ask for help with assignments, swap study materials, or simply meet new people, all in the same place.

Before this, students had to juggle scattered WhatsApp and Telegram groups to find past papers or set up study groups. This patchwork always led to repeated questions and lost files—frankly, a mess. FICT360 cleans up that chaos by offering a central hub where resources and announcements are organized and easy to search.

As education has shifted to digital-first, student interactions now lean heavily on their phones. By breaking topics down into clear categories—Academic & Learning, Campus Life, and more—FICT360 makes sure vital coursework doesn't get buried under unrelated chatter. Students dig up what they need fast, whether it's notes from last week or details about the next big event.

Looking back, getting the app up and running wasn't easy. Integrating live chat with WebSockets, handling large file uploads—those were real hurdles. But the finished result works, and it fills a genuine gap. As digital learning keeps evolving, platforms like FICT360 show just how much students gain from a campus community they can actually rely on.

7.2 Recommendation

Though FICT360 works well now and fulfils core goals, a few upgrades might boost how it supports users. Connecting the tool to the university's main API - or its online learning system - could be especially helpful down the line. With that link in place, updates like class times, test dates, or school notices would flow straight into each person's dashboard without delay. That kind of setup means less jumping from one site to another just to stay informed.

A different hands-on improvement focuses on how content is found within the platform. Right now, finding study resources requires active searching by the user. By using an automated suggestion tool that draws from a learner's current classes or earlier searches, useful material like lecture summaries or homework threads might appear on their personalized feed without extra effort. One option gaining quiet traction includes a points-based motivation layer - students who often share well-rated documents earn visible tokens of recognition, which quietly pushes more peers to add value into the collective library.

One way to improve how people interact with the app is by including a tool that shows documents inside it. Without needing to save files like PDFs or Word docs onto their devices, learners could simply open them right away. This kind of feature speeds up access while also reducing clutter on personal gadgets. Another step forward might involve upgrading messaging features so spoken conversation becomes possible. With live audio or visual connection options added, classmates working together from different places gain better ways to collaborate. A third update worth considering is allowing logins through each person's school-issued email account. Using such authentication helps simplify signing up and ensures only enrolled individuals enter restricted areas.

REFERENCES

[1] Microsoft Corporation, "Microsoft Teams for Schools and Students," *Microsoft Education*, 2024. [Online]. Available: https://www.microsoft.com/en-us/education/products/teams. [Accessed: Dec. 26, 2024].

[2] Microsoft Corporation, "Set up Teams for Education - M365 Education," *Microsoft Learn*, 2024. [Online]. Available: https://learn.microsoft.com/en-us/microsoft-365/education/deploy/set-up-teams-for-education. [Accessed: Dec. 26, 2024].

[3] LearningMole, "Microsoft Teams Education: Comprehensive Guide for Schools," *LearningMole*, Dec. 2024. [Online]. Available: https://learningmole.com/microsoft-teams-education/. [Accessed: Dec. 26, 2024].

[4] V. N. Hoi and H. L. Hang, "Student Engagement in the Facebook Learning Environment: A Person-Centred Study," *Journal of Educational Computing Research*, p. 073563312110301, Jul. 2021, doi: https://doi.org/10.1177/07356331211030158.

[5] A. Pandey, "List Of Top Educational Facebook Groups," *Classplus Growth Blog*, Jul. 13, 2022. https://classplusapp.com/growth/list-of-top-educational-facebook-groups/ (accessed Sep. 10, 2025).

[6] P. Gratton, "Facebook's Advantage Over Other Social Media," Investopedia, Mar. 28, 2024. https://www.investopedia.com/articles/company-insights/070216/what-facebooks-advantage-over-other-social-media-fb.asp

[7] D. Raymond, "Top 10 Cons & Disadvantages of Facebook," Articles for Project Managers - The Project Management Network, Jan. 09, 2024. https://projectmanagers.net/top-10-cons-disadvantages-of-facebook/

[8] "GroupMe | Group text messaging with GroupMe," *GroupMe*, 2018. https://groupme.com/

[9] “What’s New in GroupMe: Back to School - GroupMe,” *Groupme.com*, 2024. https://groupme.com/blog/what-s-new-in-groupme-back-to-school (accessed Sep. 10, 2025).

[10] assistants and M. Wong, “GroupMe chatbots raise frustrations,” *The Daily Northwestern*, Nov. 07, 2024. https://dailynorthwestern.com/2024/11/06/lateststories/groupme-chatbots-raise-frustrations-for-student-organizations (accessed Sep. 10, 2025).

[11] “Campuswire,” *Campuswire.com*, 2025. https://campuswire.com (accessed Sep. 10, 2025).

[12] B. A. Smith, “Campuswire FAQs,” *Medium*, Feb. 10, 2019. https://medium.com/campuswire/campuswire-faqs-c6993d93cb29 (accessed Sep. 10, 2025).

[13] “Campuswire, Course Communication Tool,” *Uillinois.edu*, 2020. https://answers.uillinois.edu/illinois/page.php?id=104844 (accessed Sep. 10, 2025).

[14] Slack, “Set up a Slack workspace for your college or university course,” *Slack*. https://slack.com/resources/using-slack/set-up-a-slack-workspace-for-your-college-or-university-course (accessed Jan. 07, 2024).

[15] A. Wright, “Enhancing Active Learning with Slack - MIT Sloan Teaching & Learning Technologies,” *MIT Sloan Teaching & Learning Technologies*, Aug. 28, 2024. https://mitsloanedtech.mit.edu/2024/08/28/enhancing-active-learning-with-slack/

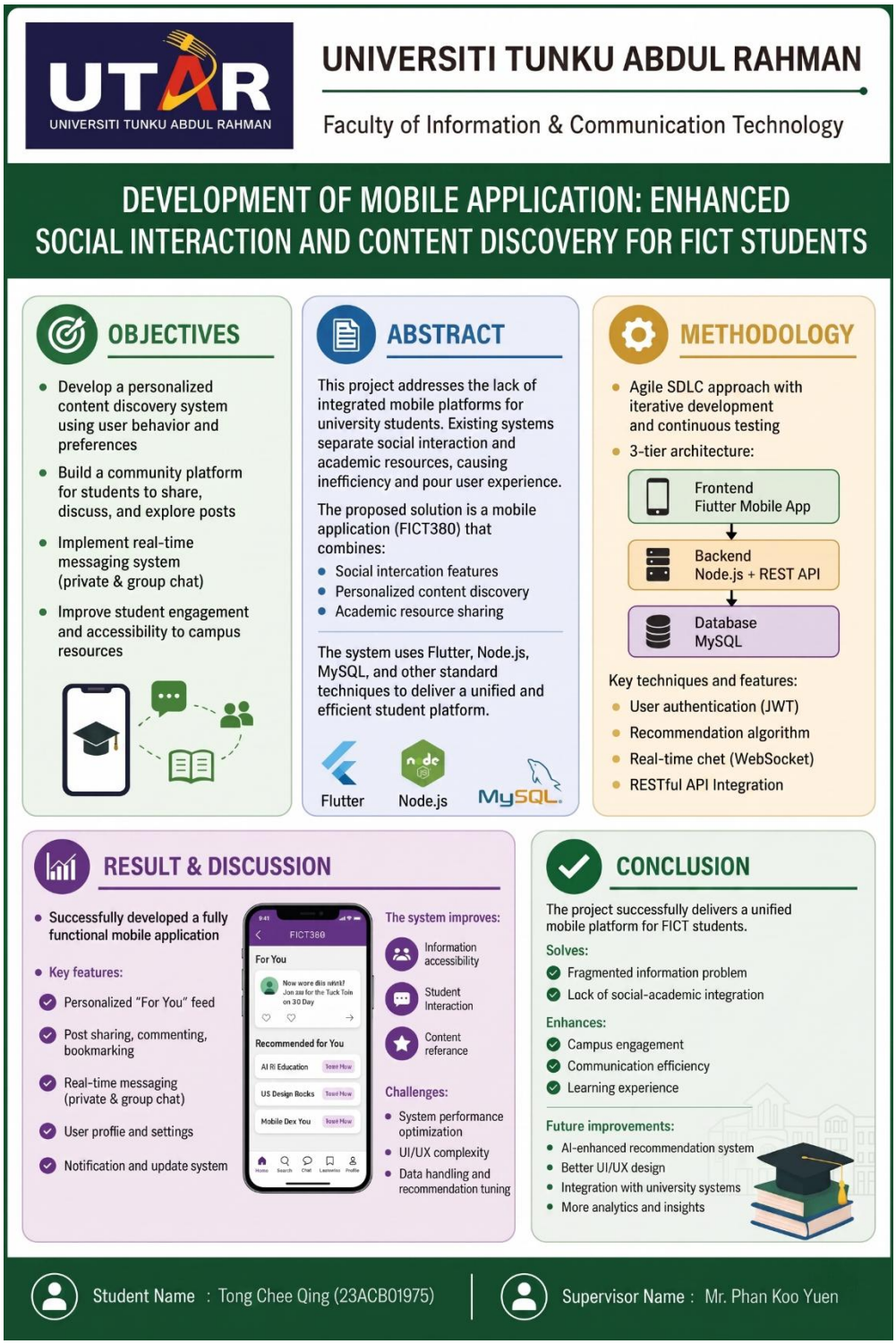
[16] Slack, “Distance learning thrives in Slack,” *Slack*, 2020. https://slack.com/blog/collaboration/distance-learning-in-slack/ (accessed Sep. 10, 2025).

[17] N. Md Yusof and A. Abdullah, “The efficacy of Telegram Messenger as a tool for enhancing argumentative writing among students in open and distance learning,” *Asian Association of Open Universities Journal*, Dec. 2024, doi: https://doi.org/10.1108/aaouj-07-2022-0091.

[18] Z. Zhao, X. Wang, S. M. Ismail, Md. K. Hasan, and A. Hashemifardnia, "Social media and academic success: Impacts of using telegram on foreign language motivation, foreign language anxiety, and attitude toward learning among EFL learners," *Frontiers in Psychology*, vol. 13, Sep. 2022, doi: https://doi.org/10.3389/fpsyg.2022.996577.

[19] R. Adesope, Nwaizugbu Yinka, and Nkeiruka Queendarline, "Telegram as a social media tool for teaching and learning in tertiary institutions," Jul. 01, 2018. https://www.researchgate.net/publication/370376455_Telegram_as_a_social_media_tool_for_teaching_and_learning_in_tertiary_institutions

POSTER


UTAR
UNIVERSITI TUNKU ABDUL RAHMAN
UNIVERSITI TUNKU ABDUL RAHMAN
Faculty of Information & Communication Technology
DEVELOPMENT OF MOBILE APPLICATION: ENHANCED SOCIAL INTERACTION AND CONTENT DISCOVERY FOR FICT STUDENTS
OBJECTIVES
Develop a personalized content discovery system using user behavior and preferences
Build a community platform for students to share, discuss, and explore posts
Implement real-time messaging system (private & group chat)
Improve student engagement and accessibility to campus resources
ABSTRACT
This project addresses the lack of integrated mobile platforms for university students. Existing systems separate social interaction and academic resources, causing inefficiency and pour user experience.
The proposed solution is a mobile application (FICT380) that combines:
Social intercation features
Personalized content discovery
Academic resource sharing
The system uses Flutter, Node.js, MySQL, and other standard techniques to deliver a unified and efficient student platform.
Flutter
Node.js
MySQL
METHODOLOGY
Agile SDLC approach with iterative development and continuous testing
3-tier architecture:
Frontend
Fiutter Mobile App
Backend
Node.js + REST API
Database
MySQL
Key techniques and features:
User authentication (JWT)
Recommendation algorithm
Real-time chet (WebSocket)
RESTful API Integration
RESULT & DISCUSSION
Successfully developed a fully functional mobile application
Key features:
Personalized "For You" feed
Post sharing, commenting, bookmarking
Real-time messaging (private & group chat)
User proffie and settings
Notification and update system
FICT380
For You
Recommended for You
AI RI Education
US Design Rocks
Mobile Dex You
The system improves:
Information accessibility
Student Interaction
Content referance
Challenges:
System performance optimization
UI/UX complexity
Data handling and recommendation tuning
CONCLUSION
The project successfully delivers a unified mobile platform for FICT students.
Solves:
Fragmented information problem
Lack of social-academic integration
Enhances:
Campus engagement
Communication efficiency
Learning experience
Future improvements:
AI-enhanced recommendation system
Better UI/UX design
Integration with university systems
More analytics and insights
Student Name : Tong Chee Qing (23ACB01975)
Supervisor Name : Mr. Phan Koo Yuen