BetterU: Productivity and Well-Being Application

By

Lim Jun Wei

FACULTY OF COMPUTING AND
INFORMATION TECHNOLOGY

TUNKU ABDUL RAHMAN UNIVERSITY OF
MANAGEMENT AND TECHNOLOGY
KUALA LUMPUR

ACADEMIC YEAR
2025/26

BetterU: Productivity and Well-Being App: Goals Assistance (shared), Time Usage Tracker, Anonymous Community, Personal Chat, Report Module

By

Lim Jun Wei

Supervisor: Ms Yeoh Kar Peng

A project report submitted to the

Faculty of Computing and Information Technology

in partial fulfillment of the requirement for the

Bachelor of Information Technology (Honours)

Faculty of Computing and Information Technology

Tunku Abdul Rahman University of Management and Technology

Kuala Lumpur

Declaration

The project submitted herewith is a result of my own efforts in totality and in every aspect of the project works. All information that has been obtained from other sources had been fully acknowledged. I understand that any plagiarism, cheating or collusion or any sorts constitutes a breach of TAR University rules and regulations and would be subjected to disciplinary actions.

Lim Jun Wei

Bachelor of Information Technology (Honours) in Software Systems Development

ID: 24WMR09078

Abstract

BetterU mobile application is a system designed to address issues of social isolation, cognitive overload and poor task management commonly faced by students and office workers. By providing solutions to these issues, the productivity and well-being of users can be effectively improved. BetterU consists of five key modules, namely Goal Assistance, Time Usage Tracker, Anonymous Community, Personal Chat and Report modules. The combination of these modules offers users an intelligent task scheduling system, a secure communication platform and personalized insights.

The literature review has identified the target market and technologies used in BetterU, including Flutter and Dart for cross-platform development, Python with FastAPI for backend services, Google Firebase for cloud storage, and AI algorithms such as Vosk, Whisper and spaCy for natural language processing, task categorization and scheduling optimization. In addition, a feasibility study was conducted to evaluate the project's technical, economic and operational feasibility.

The project applies the Incremental development model to ensure continuous refinement through requirement gathering, functional specification and supervisor feedback. From the requirement analysis perspective, both functional and non-functional requirements were collected, and various system design diagrams were prepared, including sequence, activity, entity relationship, class, data dictionary, report design and deployment architecture diagrams.

Implementation and testing were carried out for all major modules, including speech-to-text processing, NLP-based task extraction, NFC-enabled task tracking and community communication features. Integration testing was applied to validate the correctness, reliability and usability of the system.

Finally, the project highlights its achievements and contributions by demonstrating how BetterU successfully integrates AI-driven task management with anonymous social engagement. Limitations, improvement opportunities and lessons learned are also discussed. Overall, BetterU demonstrates a practical, scalable and user-friendly solution to improve time management, productivity and emotional well-being among students and office workers.

Acknowledgement

I would like to express my sincere gratitude to everyone who has supported me throughout the development of the BetterU project. Your guidance, encouragement and motivation were crucial for helping me to accomplish this project successfully.

Foremost, I am deeply thankful to my lovely supervisor, Ms Yeoh Kar Peng. Under her continuous guidance, valuable suggestions and support throughout the project, I am able to successfully achieve high quality works throughout the whole phases of the project including project planning, system development and implementation and evaluation. Meanwhile, she also helped me to continuously improve the BetterU project on the right track.

Apart from that, I would also like to extend my appreciation to the lecturers and faculty members of Faculty of Computing and Information Technology at Tunku Abdul Rahman University of Management and Technology (TARUMT) for providing the crucial knowledge and foundation necessary for this project. Via their impartation of knowledge and detailed advice, I am able to plan the BetterU project and develop the mobile application system efficiently and effectively.

Moreover, a heartfelt thanks to my parents, Lim Chin Keong and Liow Mee Ling for constant support and encouragement throughout this project. Their unwavering love, patience and support have allowed me to pick myself up at all times to overcome the difficulties and challenges faced during the project.

Not only that, I am also especially grateful to my teammate, Chia Ming Yi for her strong collaboration, commitment and valuable contributions throughout this project journey. Via teamwork with her and shared dedication, we are able to achieve the successful completion of this project. At the same time, her active participation and excellent collaboration has supported me to formulate the solution for solving all the problems faced during the project.

Lastly, my sincere thanks go to my fellow classmate, Ong Yi Xin who always provides her continuous encouragement and moral support throughout this project. She always offers valuable feedback which motivated me to improve the BetterU project and features. She even helps me to overcome all the challenges during this project journey.

I am truly thankful for the collective support, encouragement and guidance that made the successful completion of this project possible.

Table of Contents

Chapter 1

Introduction

1 Introduction

BetterU: Productivity and Well-Being Application is a mobile application system designed to help users formulate a "strategy" for smartly allocating their time on different types of tasks and retaining their personal space for social interactions. In overall, the scope of the project includes the features of managing users' tasks, tracking their time usage, encouraging them to engage in anonymous social interactions, communicating privately and receiving personalized productivity and social engagement insights.

In order to help users to achieve productivity while maintaining appropriate social interactions, BetterU mobile application comes with different core modules such as goal assistance, time usage tracker, anonymous community, personal chat and report modules. Via the collaboration and implementation of these modules, the system can continuously analyze users' behavior data and build an excellent "timetable" for them to ensure accomplishment of tasks efficiently. Meanwhile, it can also reserve sufficient time for facilitating users' social interactions. Thus, both productivity and well-being of users can be easily improved and maintained via using BetterU mobile application.

However, there are also some functions not included in this BetterU project. Direct mental health assessments and therapeutic recommendations are not offered in BetterU mobile application. Users are not able to directly undergo an accurate mental health assessment and inquire for advice. Moreover, BetterU mobile application does not include the real-identity authentication for users. Meanwhile, BetterU mobile application also does not integrate with third-party social platforms such as Facebook, Instagram, WeChat and others. It does not support the contact synchronization from third-party social platform applications.

1.1 Objectives

BetterU is a mobile application that helps users to efficiently create a "timetable" dedicated to them for reasonably allocating their time on different tasks while retaining their personal space. In order to achieve this goal, the modules "Goals Assistance" and "Time Usage Tracker" are implemented to arrange various tasks based on users' past behavior, deadlines and importance while dynamically monitoring and analyzing users' activity during app screen time. Apart from that, BetterU will also implement the module "Anonymous Community" and "Personal Chat" for offering a secure platform which allows users to ask and share any topic anonymously without concerns about judgment and denounce.

Goal Assistance Module (Shared)

Every day, students and office workers have to face infinite tasks and workloads until they are unable to manage or accomplish them in an ideal way within the time limitation. Thus, the goal assistance is designed to help those users to overcome this issue by accepting task item input from the users via speech-to-text extraction feature. This module can effectively analyze users' task list and generate an ideal and balanced schedule based on the deadlines and importance. Meanwhile, goal assistance also allows users to update their task progress and dynamically makes adjustments on the schedule based on current task progress and missed tasks. Via the smart advisor and the smart reminder feature, the users will be able to receive personalized suggestions on tasks prioritization for improving work strategies.

Time Usage Tracker Module

In order to increase the accuracy and relevance of task management and improvement suggestions for specific users and tasks, time usage tracker module can continuously monitor and gather users' habits based on their activity during app screen time. After gathering the useful information such as the time spent on various applications by the users, starting time and end time of app usage, it will analyze the data and provide detailed views for enabling users to know how and where their time was spent on. According to the analyzed results, it will also provide an appropriate recommendation for optimizing users' habits for better productivity and striving for a more freedom lifestyle. Via the integration with NFC feature, it also allows users to clock in and out of specific tasks using NFC stickers for more advanced real-world task tracking.

Anonymous Community Module

Due to the consideration of insecure situations when users deal with social media platforms, BetterU provides an anonymous community module for providing a safe space to users to freely share their thoughts. The main difference between this and the common social media platform is it allows users to anonymously share contents without worrying about being recognized and judged. It also allows users to join topic-based group chats, share notes, diary posts and seek advice without revealing their real identity.

Personal Chat Module

For providing a more interactive communication among users, a personal chat module can offer a private conversation for one-to-one chat to users. In the anonymous community or groups, users can send a private conversation request to the users who have the common understanding among them. After accepting the private conversation request, they can undergo direct messaging between each other. In order to protect users' privacy but maintain the users' freedom, a personal chat module allows users to choose whether to use anonymous

identity to communicate with others. In the personal chat, they can send text messages, images, videos or even share notes to maximize the interaction.

Report Module

The report module can regularly analyze and evaluate the users' productivity on completing the tasks. Meanwhile, it also summarizes the real-world and application time usage of the users and presents it in visualized forms via graphs and charts. From the aspect of social interaction, the report module will summarize a social interaction report which contains number of posts shared by users, number of comments and likes given with common discussion topics. It also analyzes users' active conversations number and frequency of seeking help in connection and support reports. All these reports can effectively help users to realize about their "work and life" progress and improvement that can be taken for increasing work productivity and social interaction.

1.2 Problems

1.2.1 Social Isolation and Fear of Seeking Help

Problem Statement:

- Insufficient time for communication with others
    - Students and office workers are struggling with infinite school work and workloads day and night, this has led to insufficient time and chance for them to get in touch with people and things beyond the school and work tasks. Eventually, they will be isolated from society where this phenomenon is known as social isolation. It is a situation where the people are lacking friends or close co-workers, and they often feel lonely, depressed, lack self-esteem and anxiety. (Tulane University, 2020)

    - 63% of Malaysian workers surveyed revealed that they have not been spending enough time with their family due to long working hours. Nearly 70% of people spend 2 to 5 hours working beyond their official work hours every day. Most of the reason is they are getting a lot of unreasonable deadlines and a huge workload. (SEEK Limited, 2023)

- Busy schedules and unaffordable workloads
    - Malaysia ranks as the second worst country globally for work-life balance out of 60 nations, scoring only 27.51 out of 100 in the Global Work-Life Balance Index 2024. ([Malaysia] Second Worst Country for Work-Life Balance in

New Ranking, 2015) This poor ranking indicates that long working hours and minimal personal time for many workers.

    - The lack of structured planning and poor work-life boundaries will lead to burnout and difficulty in managing heavy workloads among Malaysian employees. ("Work Ends at 6, Not 10": How Malaysians from All Generations Are Drawing the Line in 2025 | Malaysian Career Advice by Hiredly, 2025) There are many Malaysian employees and students who do not organize their work schedules in a rational manner, they always spend unnecessary time on unproductive works and tasks.

- Uncomfortable social environment
    - Social isolation always happens due to an unfriendly workplace. There are many teleworkers or employees working remotely in Malaysia reported that reduced casual interaction with supervisors and colleagues, lack of emotional and social support and an environment that is not conducive to social connection. (Go to GoGuardian App, 2025) This perceived isolation has led to fear of employees seeking help from supervisors, colleagues or other people.

    - According to Universiti Sains Malaysia, 84.7% of undergraduate students experienced social anxiety. (Aleeya & Binti Roslan, 2023) The main reason is students have perceived extremely low social support which causes an uncomfortable or unsupportive social environment. This would seriously affect students' fear of social interaction and seeking help.

- Resistance of using social media platforms as fear of judgment
    - Some users are reluctant to share images and content or even engage on social media due to fear of judgment and scopophobia (fear of being watched or scrutinized). (Mir & Shardeo, 2024) The judgment and scrutinization will significantly impact self-esteem and increase feelings of inadequacy among users.

BetterU Solution:

- Anonymous Community Module
    - BetterU allows users to post content, comment posts and interact without revealing their identity. This can effectively reduce social pressure and eliminates the fear of being judged.

- Users are able to join their favourite communities around topics which can help them to find relevant conversations, achieve common understanding and even find a soulmate.

- BetterU encourages users to express their inner thoughts, seek emotional support and share small wins or struggles, which can help to reduce the feelings of isolation and builds a sense of belonging.

- Personal Chat Module
  - BetterU enables users to initial personal chats with others who share similar experiences, emotional struggles or goals after mutual acceptance.

  - Users have the option to choose whether to continue using anonymous names or reveal their real identity. This can maximize user freedom and protect user privacy.

  - Users are allowed to send messages in various forms such as text, images and videos for enriching communication and build empathy in conversations.

- Report Module
  - BetterU provides social engagement analytics which can track the number of posts, comments, likes and other relevant useful information from the user usage. This could easily help users to view their gradual reconnection with the community.

  - It can act as a connection and support metrics for acknowledging users about their emotional engagement and support patterns based on the frequency and types of help sought.

1.2.2 Cognitive Overload and Difficulty in Task Management

Problem Statement:

- Cognitive overload among students and workers
  - Based on the Cognitive Load Theory, students may easily experience increased cognitive load when dealing with multiple tasks simultaneously. (Yi Lin Tan et al., 2022) By adding various assignment deadlines and CGPA result considerations, students may perceive more cognitive overload and are unable to manage the tasks well. This could seriously affect the learning and academic performance.

- Most of the malaysian workers will face a lot of cognitive load when working at a company or communicating with various customers. This will significantly create emotion exhaustion, anxiety and social media cognitive fatigue. The workers' mind will eventually be overwhelmed by excessive information and communication demands, they will feel fatigue until unable to manage their tasks in a good way.

- Difficulty in prioritization and task clarity
  - There are many Malaysian students who face difficulties in time management, procrastination and prioritization, this phenomenon will negatively affect academic performance. (Peng et al., 2017) Without a proper time allocation plan based on task importance and deadline, students will accidentally engage in unnecessary activities. They may even get used to last-minute preparations, leading to piling up of tasks and failure of completing assignments before the deadline.

  - Every day, workers will have to deal with infinite various tasks assigned by their supervisors. They struggle to decide where to begin, which task to do first and how to plan tasks logically. Meanwhile, the overwhelmed working memory will also make it hard to track and manage tasks efficiently.

- Existing tools are too complex for personal use
  - Although there are many professional task management tools in the market, most of them are designed for team collaboration and include advanced features that can overwhelm individual users. (The, 2025) For individual users, they would desire a task management tool which is simple and easy to manage personal tasks instead of unnecessary and complicated features.

- Low adoption due to complexity and effort
  - When dealing with task management, there is a significant portion of users who are unwilling to manually input tasks due to time and effort constraints.

  - Sometimes, the task management tools can also become overwhelming due to the poor task clarity entered by users. (The, 2024) Users may accidentally miss some important information when manually jotting down all the task items using own-typing text. They might feel lost or confused when reviewing their own task lists, which hampers productivity and consistent use.

BetterU Solution:

- Goal Assistance Module (Shared)
    - BetterU has offered users the speech-based task input where users can input tasks using voice and it will automatically convert speech-to-text. This could effectively eliminate the time spent on manual typing.
    - Once the tasks are inputted, BetterU can automatically analyze the urgency and importance of each task, then generate a balanced schedule designated for specific users. This can minimize the cognitive load of users when deciding what to do and when.
    - Users are allowed to update their task progress anytime by labelling the status such as pending, completed and cancelled. BetterU will dynamically reorganize the schedule in real-time based on the task progress for ensuring continuous progress even if plans change.
    - Smart advisor features in this module will give personalized tips on how to optimize the task order or manage focus based on historical behavior. Thus, users will not feel lost or overwhelmed when dealing with many tasks.
- Time Usage Tracker Module
    - BetterU can continuously track app screen time and digital activity by users so they can easily know how much time they have spent on productive versus unproductive activities.
    - Via integration with NFC stickers, users can clock in or out of specific tasks by tapping their phone. This can make the task tracking seamless and ease for tracking real-world activities.
    - BetterU can gather useful information from users' activity to analyze user behavior. It will be used for providing custom suggestions to reduce the time drains and refocus attention on higher-priority tasks.
- Report Module
    - BetterU will also regularly summarize and generate reports to visualize the completion rate, average time per task and missed deadlines using graphs and charts. Thus, users can review their task progress and obtain insights on improving task management.

- BetterU will also do analysis on time distribution to show users how their time is spent across various categories like work, study and leisure. This can help users to identify the imbalances or wasted time.

1.3 Advantages and Contributions

1.3.1 Goal Assistance Module (Shared)

Advantages:

- Cognitive load when making decisions on managing tasks in terms of importance and time can be reduced since goal assistance modules can dynamically help users to organize all the tasks based on their behaviors.
- BetterU can efficiently simplify complex and overwhelming tasks into smaller and manageable steps so users only need to execute their tasks based on the instructions given by the app.
- BetterU can minimize the effort in task recording via speech-based input. Users can easily and immediately record down all the important tasks any time.

Functional Contributions:

- BetterU allows users to input tasks via speech recognition which can prevent the fatigue and human mistakes of manual typing.
- BetterU automatically analyzes users' tasks and generates a smart and balanced schedule based on the urgency and deadlines.
- BetterU allows users to update their task progress and it will automatically adjust the schedule based on task completion and real-time changes.
- BetterU can regularly provide smart suggestions and reminders for maintaining users' motivation and improving productivity.

1.3.2 Time Usage Tracker Module

Advantages:

- BetterU is able to increase users' self-awareness of time usage patterns via acknowledging users about the time usage using graphs and charts.

- BetterU helps users to reflect and improve their time allocation for better productivity by providing useful suggestions based on their current performance on time spending.
- A good habit of users can be built via tracking users' app activities and providing useful insights for enhancing their productivity.

Functional Contributions:

- BetterU can dynamically monitor and analyze screen time and app usage for highlighting unproductive patterns.
- BetterU supports the integration with NFC stickers to realize the clock in and out of tasks by tapping the phone on the sticker. This can further promote the seamless tracking of users' tasks.
- BetterU offers data-driven suggestions to adjust habits and build healthier time management behavior.

1.3.3 Anonymous Community Module

Advantages:

- BetterU encourages users to share thoughts and struggles without fear of judgment by enabling them to anonymously post contents in the community.
- The social anxiety and feelings of isolation can be reduced efficiently by increasing engagement and communication with other peoples.
- BetterU can help for building a supportive and stigma-free environment to motivate users to express themselves.

Functional Contributions:

- BetterU provides a safe and anonymous space for users to participate in topic-based group chats, share diaries and seek advice.
- Anonymous community removes identity-related pressures for promoting honest expression and deeper emotional connection.
- Anonymous community enables users to feel heard and understood in a secure environment.

1.3.4 Personal Chat Module

Advantages:

- BetterU can effectively strengthen one-to-one emotional support connections.
- BetterU enables private and secure communication between two users while preserving anonymity if desired.
- BetterU can prevent users' fear of public disclosure in help-seeking.

Functional Contributions:

- BetterU enables private messaging between users who bond in the anonymous community once both users have accepted the request.
- Personal chat allows users to decide whether to communicate in anonymous or identified form to match users' comfort levels.
- Personal chat supports text, images, videos and posts sharing for enhancing user experience and self-expression.

1.3.5 Report Module

Advantages:

- BetterU promotes self-awareness through visual summaries of tasks, time usage and social interactions so users can gradually improve their mental health and strategies when dealing with task management. Users can also improve their self-confidence when social interaction.
- BetterU encourages reflection, personal growth and continuous improvement by regularly providing useful feedback and improvement suggestions to users.

Functional Contributions:

- BetterU summarizes task performance by showing the number of completed or pending tasks, completion rate and missed deadlines using intuitive graphs.
- BetterU analyzes time usage data and offers suggestions for time distribution across various activities such as work, study and leisure.
- BetterU generates connection and interaction reports that show the number of posts, private chats, like and active conversations for enabling users to review their social engagement performance in community.

1.4 Project Plan

1.4.1 Project Scope

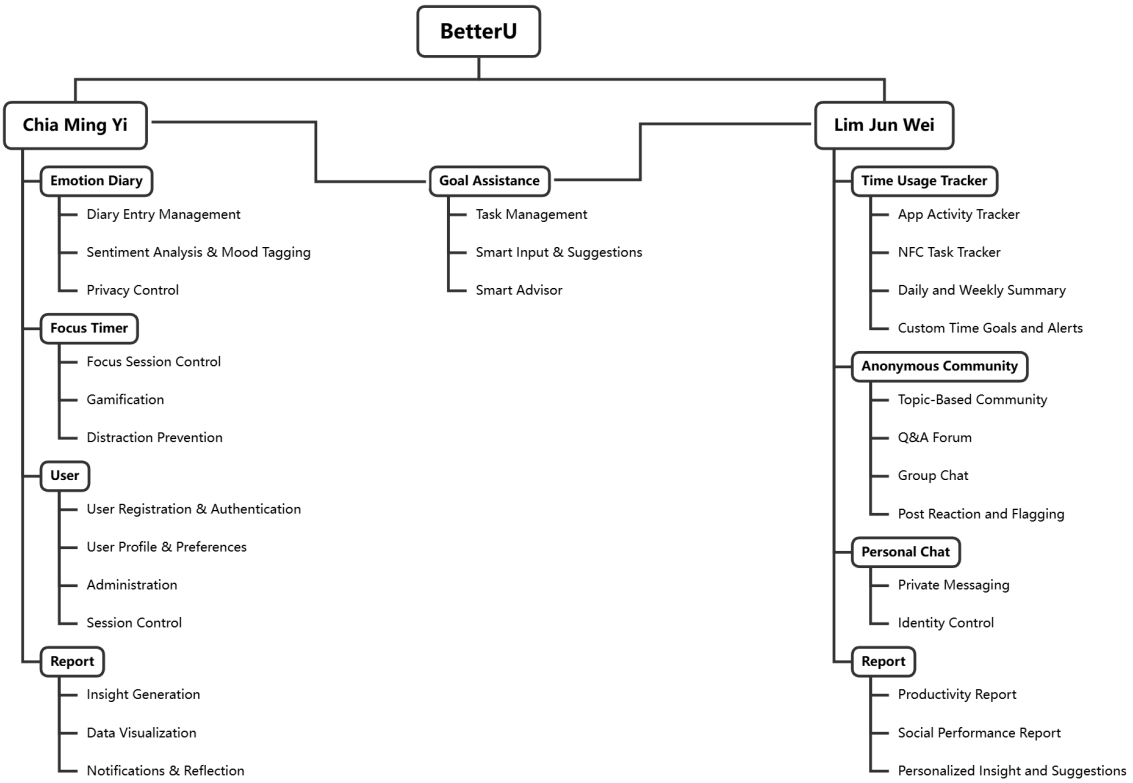


Figure 1.1: Hierarchy Chart of BetterU

Functional Features

1.0 Goal Assistance Module (Shared)

1.1 Task Management

1.1.1 The system shall allow users to create, edit, delete, and mark tasks as completed.

1.1.2 The system shall support optional fields in task creation, including description, link, image attachment, and sub-tasks.

1.1.3 The system shall allow users to classify tasks into predefined or user-customized categories.

1.1.4 The system shall support task types: One-time or Repeat (with frequencies: daily, weekly, monthly, yearly).

1.1.5 The system shall allow users to set task priority levels from P1 (highest) to P4 (default, least important).

1.1.6 The system shall allow users to specify task timing: create date, occur date, and due date with preset options (Today, Tomorrow, Weekday, Weekend, Specific Date, or No Due Date).

1.1.7 The system shall allow enabling push notifications as reminders, triggered at the occurrence time or set intervals before (e.g., 10 minutes before).

1.1.8 The system shall support progress tracking based on sub-task completion (displayed as a percentage).

1.1.9 The system shall allow users to sort and filter tasks by category, priority, type, and due date.

1.1.10 The system shall support a calendar view and list view for task visualization.

1.2 Smart Input & Suggestions

1.2.1 The system shall extract task goals from natural language inputs using NLP (e.g., "Remind me to submit a project report tomorrow at 3 PM").

1.2.2 The system shall auto-suggest suitable time slots based on availability, habits, and workload.

1.2.3 The system shall recommend task categories and priority based on task keywords and past behaviors.

1.2.4 The system shall provide predefined templates for recurring task types (e.g., study, exercise, work).

1.3 Smart Advisor

1.3.1 The system shall analyze users' past task completion habits to suggest priority adjustments (e.g., start earlier, complete before usual fatigue hours).

1.3.2 The system shall detect schedule conflicts and suggest alternative time slots dynamically.

1.3.3 The system shall provide emotional and productivity-based reminders, such as taking breaks or relaxation suggestions.

1.3.4 The system shall learn long-term behavior trends to optimize future task planning (e.g., avoid late-night tasks, balance workload).

1.3.5 The system shall allow users to enable/disable Smart Advisor and choose whether to apply, ignore, or manually adjust its suggestions.

2.0 Time Usage Tracker Module

2.1 App Activity Tracker

2.1.1 The system shall continuously monitor the screen time usage per app once permitted by the user.

2.1.2 The system shall collect the data when users start and stop using the apps.

2.1.3 The system shall analyze the app usage duration based on the collected data of time spending on different apps.

2.1.4 The system shall categorize apps into productivity, study and leisure.

2.1.5 The system shall display a dashboard which generates visual graphs of app activity with time spending based on users' daily and weekly app usage.

2.1.6 The system shall sort the app usage based on the categories, time spent on apps and apps' name.

2.2 NFC Task Tracker

2.2.1 The system shall allow users to write and read the task info into and from the NFC tag.

2.2.2 The system shall detect NFC tag scans to clock in or out of specific tasks.

2.2.3 The system shall update the existing task progress based on the NFC tag scanned.

2.2.4 The system shall link NFC sessions to goals and reports.

2.2.5 The system shall notify users if an NFC tag is invalid or not linked to any task.

2.3 Daily and Weekly Summary

2.3.1 The system shall display the summaries of tasks completed and time spent on each task.

2.3.2 The system shall allow users to filter the task summaries based on category, priority and time created.

2.3.3 The system shall generate a visual breakdown of daily and weekly app activity in categories.

2.3.4 The system shall highlight the inconsistencies and unusual time usage patterns.

2.3.5 The system shall provide appropriate suggestions on app usage based on inconsistent time usage patterns detected.

2.3.6 The system shall allow users to export the daily or weekly summaries as PDF files.

2.3.7 The system shall allow users to view historical summaries based on selected date ranges.

2.4 Custom Time Goals or Alerts

2.4.1 The system shall allow users to define time limits or productivity goals for specific app categories.

2.4.2 The system shall send notifications when a time goal is exceeded or if a break is needed.

2.4.3 The system shall provide suggestions for scheduling and task balancing when irregular patterns are detected.

3.0 Anonymous Community Module

3.1 Topic-Based Community

3.1.1 The system shall allow users to join anonymous communities based on shared topics of interest.

3.1.2 The system shall allow users to create posts without revealing their identities.

3.1.3 The system shall allow users to edit the post title, description and attachments before posting into the community.

3.1.4 The system shall allow users to subscribe to various topics to receive updates and engage with the content.

3.1.5 The system shall allow users to report the communities for moderation review.

3.1.6 The system shall allow users to search for posts based on the post title and date range.

3.2 Q&A Forum

3.2.1 The system shall allow users to post questions anonymously.

3.2.2 The system shall allow users to answer questions posted in the forum via text and images.

3.2.3 The system shall allow users to mark answers with upvote or downvote.

3.2.4 The system shall send notifications to the original poster when their question receives new answers.

3.2.5 The system shall highlight the most helpful answer for each question.

3.2.6 The system shall categorize the posts with tags to make them searchable and easily organized.

3.2.7 The system shall allow users to search forum posts based on post title, labeled tags and created date.

3.2.8 The system shall allow users to sort answers based on the amount of upvotes and the created date.

3.3 Group Chat

3.3.1 The system shall allow users to engage in real-time group chats using anonymous display names.

3.3.2 The system shall allow users to send messages in text and emoji forms.

3.3.3 The system shall allow users to insert image attachments when sending messages.

3.3.4 The system shall send notifications to the joined users when there are new messages in joined group chat.

3.3.5 The system shall allow users to select whether to receive or mute notifications of messages in joined group chat.

3.3.6 The system shall allow users to leave the group chat anytime.

3.3.7 The system shall automatically block the message sending when there are illegal terms in the text messages.

3.4 Post Reaction and Flagging

3.4.1 The system shall allow users to react to posts with likes or emoticons.

3.4.2 The system shall allow users to remove or update the previous reaction to the posts.

3.4.3 The system shall automatically block or drop the posts when there are illegal terms in the posts content.

3.4.4 The system shall allow users to manually flag the inappropriate or harmful posts for administrator review.

3.4.5 The system shall block or drop the inappropriate or harmful posts if verified by administrator review.

4.0 Personal Chat Module

4.1 Private Messaging

4.1.1 The system shall allow users to engage in one-on-one conversations via text messages, image sharing and video sending.

4.1.2 The system shall allow users to reply to specific messages sent or received.

4.1.3 The system shall send notifications to the users when receiving new messages.

4.1.4 The system shall allow users to see the last online status of other users.

4.1.5 The system shall automatically block the message sending when there are illegal terms in the text messages.

4.1.6 The system shall allow users to report the specific user for administrator review.

4.1.7 The system shall allow users to block the specific user from sending messages to him or her.

4.1.8 The system shall notify the user if a message is blocked due to privacy violation.

4.2 Identity Control

4.2.1 The system shall allow users to select the option whether to chat anonymously or reveal their identity during private messaging.

4.2.2 The system shall automatically assign a random name to the user who has selected anonymous chat.

4.2.3 The system shall prevent anonymous users from impersonating real usernames.

4.2.4 The system shall prevent anonymous users from viewing detailed profile information of others.

4.2.5 The system shall limit the number of daily messages allowed in anonymous chats.

5.0 Report Module

5.1 Productivity Report

5.1.1 The system shall generate reports summarizing all completed, ongoing and missed tasks over a selected time period.

5.1.2 The system shall analyze and display the task completion rate and overall performance trend.

5.1.3 The system shall allow users to export reports as PDF format.

5.1.4 The system shall provide visualizations, graphs and charts when displaying the summaries of tasks statistics.

5.1.5 The system shall allow users to filter the reports based on task categories, priority level and date range.

5.2 Social Performance Report

5.2.1 The system shall generate reports which reflect users' community engagement, including number of posts made, comments, likes given and conversations started.

5.2.2 The system shall track help-seeking and support-offering behaviors to measure social connectivity.

5.2.3 The system shall categorize social interactions based on different types such as supportive and informative.

5.2.4 The system shall summarize which communities or topics the user has engaged with the most.

5.3 Personalized Insight and Suggestions

5.3.1 The system shall provide suggestions to users for improving work productivity based on users' activity data.

5.3.2 The system shall assess users' social engagement progress and provide insights to improve social engagement.

5.3.3 The system shall provide the scores based on the support-seeking and support-giving patterns to evaluate user potential social roles.

Non-Functional Features

Product Requirements

1.0 Usability

1.1 The system shall provide a user-friendly interface that is intuitive for novice users without requiring external assistance.

1.2 The system shall provide simple navigation and offer onboarding guidance for first-time users.

1.3 The system shall allow each module to be accessible within a few clicks.

1.4 The system shall ensure the appropriate and consistent font size, icons and buttons for mobile devices.

2.0 Efficiency

2.1 The system shall minimize manual input effort by offering speech-to-text task entry and NFC tagging.

2.2 The system shall ensure a fast retrieval of data when maintaining the user interface smoothness.

3.0 Performance

3.1 The system shall load each interface within 3 seconds on every device.

3.2 The system shall send notifications and reminders in real-time.

3.3 The system shall be responsive when executing concurrent tasks.

4.0 Portability

4.1 The system shall function on Android platforms with various versions.

4.2 The system shall run smoothly on both low-end and high-end mobile devices.

Process / Organizational Requirements

1.0 Delivery

1.1 The system shall follow Incremental methodology to introduce new modules or enhanced functionality in each increment.

2.0 Implementation

2.1 The system shall be developed using Dart programming language with Flutter framework.

2.2 The system shall use Google Firebase to store system configuration and users' data.

2.3 The system shall use Git and GitHub for system version control.

External Requirements

1.0 Interoperability

1.1 The system shall support integration with third-party APIs such as calendar and mental health resources.

1.2 The system shall export the database data in standard JSON format.

1.3 The system NFC functionality shall comply with standard tag types such as NTAG215 or Mifare.

2.0 Privacy

2.1 The system shall securely store and encrypt all users' personal data such as chat content.

2.2 The system shall not disclose any users' sensitive information to outside parties.

2.3 The system shall acknowledge users about the information being collected and provide options to users to opt in or out of data sharing.

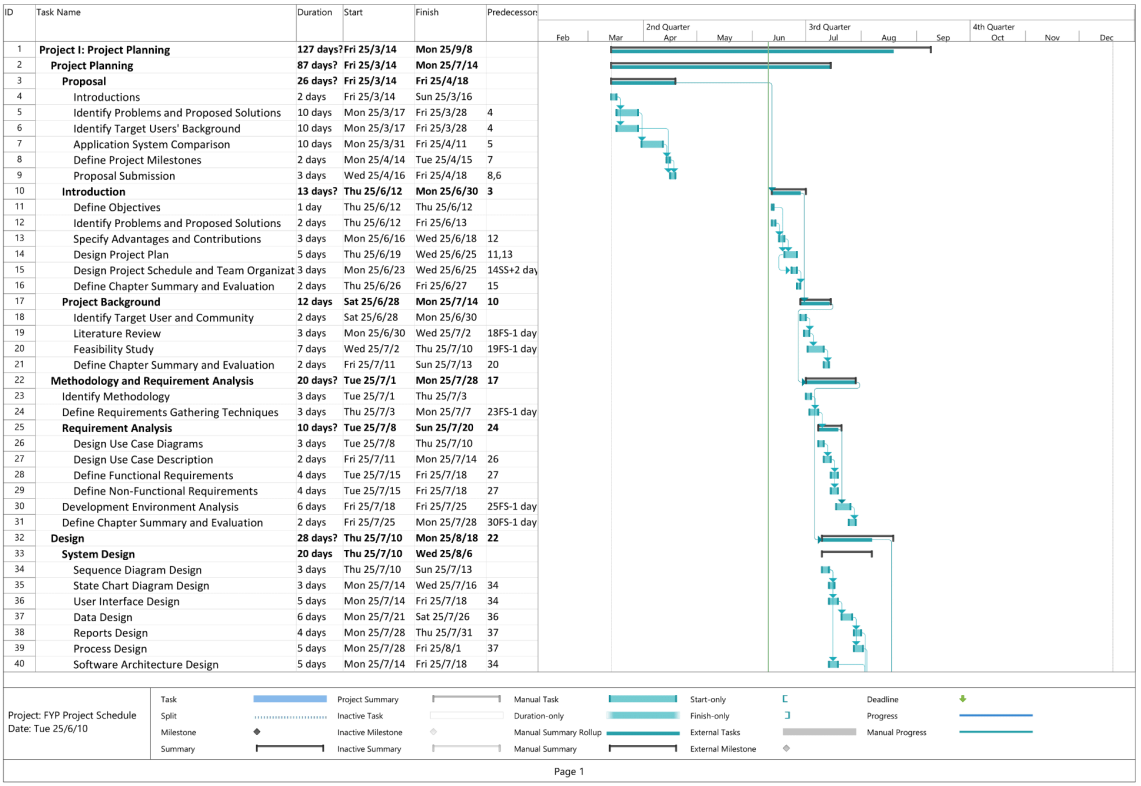
3.0 Safety

3.1 The system should filter or block inappropriate content in chat using AI-based detection.

3.2 The system should allow users to report and flag violated contents for moderation review.

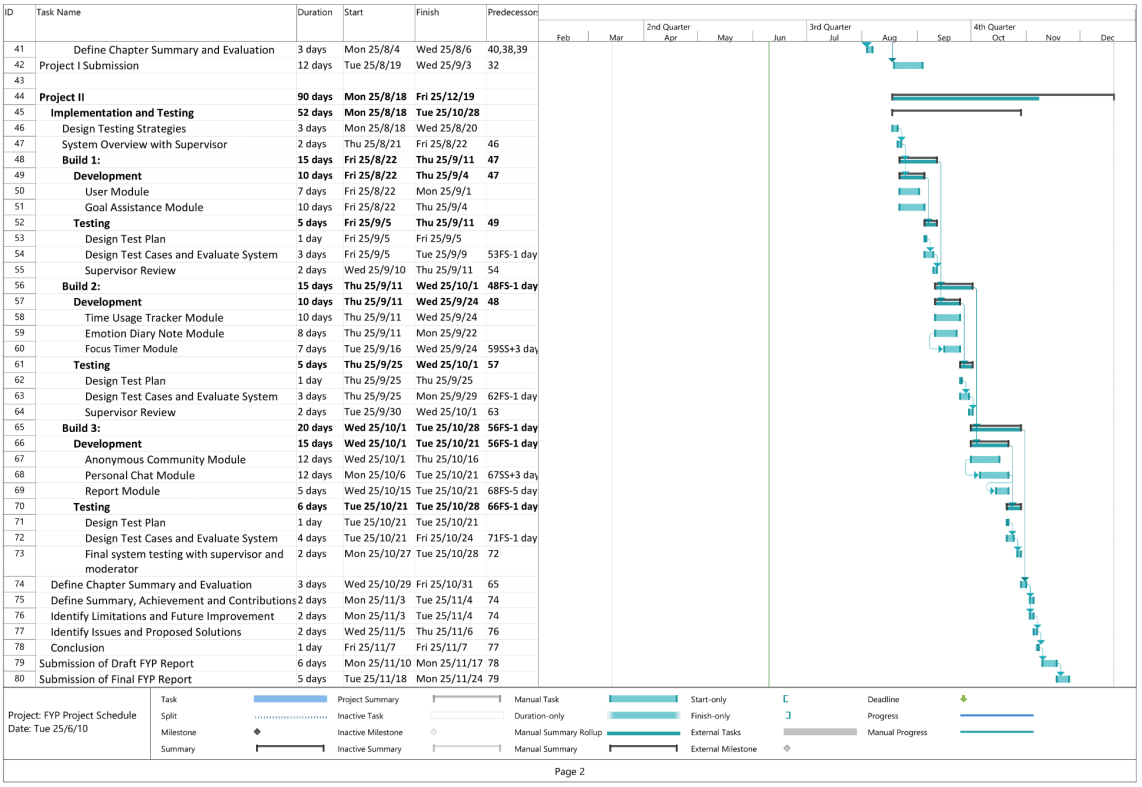
1.4.2 Project Schedule

Gantt Chart



ID	Task Name	Duration	Start	Finish	Predecessors
1	**Project I: Project Planning**	**127 days?**	**Fri 25/3/14**	**Mon 25/9/8**	
2	**Project Planning**	**87 days?**	**Fri 25/3/14**	**Mon 25/7/14**	
3	**Proposal**	**26 days?**	**Fri 25/3/14**	**Fri 25/4/18**	
4	Introductions	2 days	Fri 25/3/14	Sun 25/3/16	
5	Identify Problems and Proposed Solutions	10 days	Mon 25/3/17	Fri 25/3/28	4
6	Identify Target Users' Background	10 days	Mon 25/3/17	Fri 25/3/28	4
7	Application System Comparison	10 days	Mon 25/3/31	Fri 25/4/11	5
8	Define Project Milestones	2 days	Mon 25/4/14	Tue 25/4/15	7
9	Proposal Submission	3 days	Wed 25/4/16	Fri 25/4/18	8,6
10	**Introduction**	**13 days?**	**Thu 25/6/12**	**Mon 25/6/30**	**3**
11	Define Objectives	1 day	Thu 25/6/12	Thu 25/6/12	
12	Identify Problems and Proposed Solutions	2 days	Thu 25/6/12	Fri 25/6/13	
13	Specify Advantages and Contributions	3 days	Mon 25/6/16	Wed 25/6/18	12
14	Design Project Plan	5 days	Thu 25/6/19	Wed 25/6/25	11,13
15	Design Project Schedule and Team Organizat	3 days	Mon 25/6/23	Wed 25/6/25	14SS+2 day
16	Define Chapter Summary and Evaluation	2 days	Thu 25/6/26	Fri 25/6/27	15
17	**Project Background**	**12 days**	**Sat 25/6/28**	**Mon 25/7/14**	**10**
18	Identify Target User and Community	2 days	Sat 25/6/28	Mon 25/6/30	
19	Literature Review	3 days	Mon 25/6/30	Wed 25/7/2	18FS-1 day
20	Feasibility Study	7 days	Wed 25/7/2	Thu 25/7/10	19FS-1 day
21	Define Chapter Summary and Evaluation	2 days	Fri 25/7/11	Sun 25/7/13	20
22	**Methodology and Requirement Analysis**	**20 days?**	**Tue 25/7/1**	**Mon 25/7/28**	**17**
23	Identify Methodology	3 days	Tue 25/7/1	Thu 25/7/3	
24	Define Requirements Gathering Techniques	3 days	Thu 25/7/3	Mon 25/7/7	23FS-1 day
25	**Requirement Analysis**	**10 days?**	**Tue 25/7/8**	**Sun 25/7/20**	**24**
26	Design Use Case Diagrams	3 days	Tue 25/7/8	Thu 25/7/10	
27	Design Use Case Description	2 days	Fri 25/7/11	Mon 25/7/14	26
28	Define Functional Requirements	4 days	Tue 25/7/15	Fri 25/7/18	27
29	Define Non-Functional Requirements	4 days	Tue 25/7/15	Fri 25/7/18	27
30	Development Environment Analysis	6 days	Fri 25/7/18	Fri 25/7/25	25FS-1 day
31	Define Chapter Summary and Evaluation	2 days	Fri 25/7/25	Mon 25/7/28	30FS-1 day
32	**Design**	**28 days?**	**Thu 25/7/10**	**Mon 25/8/18**	**22**
33	**System Design**	**20 days**	**Thu 25/7/10**	**Wed 25/8/6**	
34	Sequence Diagram Design	3 days	Thu 25/7/10	Sun 25/7/13	
35	State Chart Diagram Design	3 days	Mon 25/7/14	Wed 25/7/16	34
36	User Interface Design	5 days	Mon 25/7/14	Fri 25/7/18	34
37	Data Design	6 days	Mon 25/7/21	Sat 25/7/26	36
38	Reports Design	4 days	Mon 25/7/28	Thu 25/7/31	37
39	Process Design	5 days	Mon 25/7/28	Fri 25/8/1	37
40	Software Architecture Design	5 days	Mon 25/7/14	Fri 25/7/18	34

Figure 2.1.1: Gantt Chart of BetterU Project Schedule - Page 1



ID	Task Name	Duration	Start	Finish	Predecessors
41	Define Chapter Summary and Evaluation	3 days	Mon 25/8/4	Wed 25/8/6	40,38,39
42	Project I Submission	12 days	Tue 25/8/19	Wed 25/9/3	32
43					
44	**Project II**	**90 days**	**Mon 25/8/18**	**Fri 25/12/19**	
45	**Implementation and Testing**	**52 days**	**Mon 25/8/18**	**Tue 25/10/28**	
46	Design Testing Strategies	3 days	Mon 25/8/18	Wed 25/8/20	
47	System Overview with Supervisor	2 days	Thu 25/8/21	Fri 25/8/22	46
48	**Build 1:**	**15 days**	**Fri 25/8/22**	**Thu 25/9/11**	**47**
49	**Development**	**10 days**	**Fri 25/8/22**	**Thu 25/9/4**	**47**
50	User Module	7 days	Fri 25/8/22	Mon 25/9/1	
51	Goal Assistance Module	10 days	Fri 25/8/22	Thu 25/9/4	
52	**Testing**	**5 days**	**Fri 25/9/5**	**Thu 25/9/11**	**49**
53	Design Test Plan	1 day	Fri 25/9/5	Fri 25/9/5	
54	Design Test Cases and Evaluate System	3 days	Fri 25/9/5	Tue 25/9/9	53FS-1 day
55	Supervisor Review	2 days	Wed 25/9/10	Thu 25/9/11	54
56	**Build 2:**	**15 days**	**Thu 25/9/11**	**Wed 25/10/1**	**48FS-1 day**
57	**Development**	**10 days**	**Thu 25/9/11**	**Wed 25/9/24**	**48**
58	Time Usage Tracker Module	10 days	Thu 25/9/11	Wed 25/9/24	
59	Emotion Diary Note Module	8 days	Thu 25/9/11	Mon 25/9/22	
60	Focus Timer Module	7 days	Tue 25/9/16	Wed 25/9/24	59SS+3 day
61	**Testing**	**5 days**	**Thu 25/9/25**	**Wed 25/10/1**	**57**
62	Design Test Plan	1 day	Thu 25/9/25	Thu 25/9/25	
63	Design Test Cases and Evaluate System	3 days	Thu 25/9/25	Mon 25/9/29	62FS-1 day
64	Supervisor Review	2 days	Tue 25/9/30	Wed 25/10/1	63
65	**Build 3:**	**20 days**	**Wed 25/10/1**	**Tue 25/10/28**	**56FS-1 day**
66	**Development**	**15 days**	**Wed 25/10/1**	**Tue 25/10/21**	**56FS-1 day**
67	Anonymous Community Module	12 days	Wed 25/10/1	Thu 25/10/16	
68	Personal Chat Module	12 days	Mon 25/10/6	Tue 25/10/21	67SS+3 day
69	Report Module	5 days	Wed 25/10/15	Tue 25/10/21	68FS-5 day
70	**Testing**	**6 days**	**Tue 25/10/21**	**Tue 25/10/28**	**66FS-1 day**
71	Design Test Plan	1 day	Tue 25/10/21	Tue 25/10/21	
72	Design Test Cases and Evaluate System	4 days	Tue 25/10/21	Fri 25/10/24	71FS-1 day
73	Final system testing with supervisor and moderator	2 days	Mon 25/10/27	Tue 25/10/28	72
74	Define Chapter Summary and Evaluation	3 days	Wed 25/10/29	Fri 25/10/31	65
75	Define Summary, Achievement and Contributions	2 days	Mon 25/11/3	Tue 25/11/4	74
76	Identify Limitations and Future Improvement	2 days	Mon 25/11/3	Tue 25/11/4	74
77	Identify Issues and Proposed Solutions	2 days	Wed 25/11/5	Thu 25/11/6	76
78	Conclusion	1 day	Fri 25/11/7	Fri 25/11/7	77
79	Submission of Draft FYP Report	6 days	Mon 25/11/10	Mon 25/11/17	78
80	Submission of Final FYP Report	5 days	Tue 25/11/18	Mon 25/11/24	79

Figure 2.1.2: Gantt Chart of BetterU Project Schedule - Page 2

Incremental Model Diagram

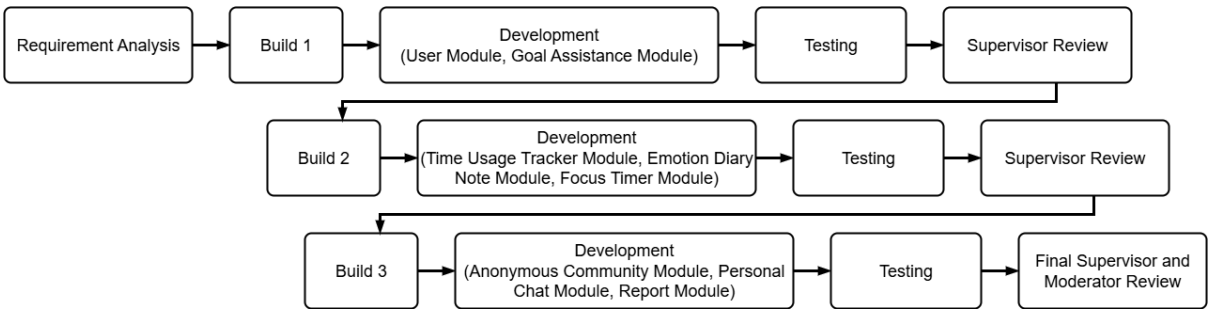


Figure 2.2: Incremental Model DIagram of BetterU

1.5 Project Team and Organization

Module in charge		Member Name
Emotion Diary Note	• Diary Entry Management • Sentiment Analysis & Mood Tagging • Privacy Control	Chia Ming Yi
Focus Timer	• Focus Session Control • Gamification • Distraction Prevention	
User	• User Registration & Authentication • User Profile & Preferences • Administration • Session Control	
Report	• Insight Generation • Data Visualization • Notifications & Reflection	
Time Usage Tracker	• App Activity Tracker • NFC Task Tracker • Daily and Weekly Summary • Custom Time Goals and Alerts	Lim Jun Wei
Anonymous Community	• Topic-Based Community • Q&A Forum • Group Chat • Post Reaction & Flagging	
Personal Chat	• Private Messaging • Identity Control	
Report	• Productivity Report • Social Performance Report • Personalized Insight and Suggestions	
Goal Assistance (Shared)	• Task Management • Smart Input & Suggestions • Smart Advisor	Chia Ming Yi Lim Jun Wei

Table 1.1: Organization Table for BetterU

1.6 Chapter Summary and Evaluation

The main objective of BetterU project is to develop a task management and well-being application for intelligently and logically allocating users' time for executing different tasks. Meanwhile, it also provides a secure platform for users to engage with society without judgment. Thus, it can help users to improve their productivity when maintaining emotional well-being and time management.

The problem statement highlights the issue of social isolation due to having busy work schedules or school work. This has led to insufficient time for engaging with the outside world. At the same time, the uncomfortable social environment and fear of judgment in social platforms have caused people to resist seeking help from others no matter physically or via online social media platforms. Apart from that, the cognitive overload and ambiguity of task prioritization also make students and workers unable to arrange their tasks properly, they might spend a lot of time on unnecessary tasks. This would significantly drop the productivity and efficiency of time management. Not only that, the high complexity and low adoption of current task management tools make people reluctant to use them for managing their personal tasks. Most of the features only required by team and organization, they are meaningless for personal users to manage individual tasks.

In order to overcome these challenges faced by current people, BetterU has provided a solution via implementing various modules such as Goal Assistance, Time Usage Tracker, Anonymous Community, Personal Chat and Report modules. Via integration with the modules, BetterU can effectively arrange users' task schedules by tracking and analyzing users' behavior. The combination of Anonymous Community and Personal Chat can also motivate users to improve social interaction and self-confidence to seek help or even provide support to others.

In the project plan, BetterU solution has adopted the Incremental Model as the process model. After the requirement gathering and analysis, BetterU will mainly focus on development of User and Goal Assistance modules as fundamental modules during the first build. During the second build, BetterU will integrate with Time Usage Tracker and Focus Timer as an extended integration module for improving the relevance and accuracy of planning users' task schedules, and enhancing productivity of users based on their behavior. In the third build, BetterU will focus on Anonymous Community, Personal Chat and Report modules for providing a secure and comfortable social engagement to users, and generating various reports with suggestions for users. In every build, BetterU will undergo a series of testing and supervisor review for collecting feedback and continuous improvement via feedback integration. Thus, BetterU can always maintain a high quality of services and functionality.

During project team and organization, the development of BetterU mobile application is undertaken by a two member team consisting of Lim Jun Wei and Chia Ming Yi. Both members collaboratively contribute to the design, development and testing phases of the project. Lim Jun Wei is responsible for the development of Goal Assistance (shared), Time Usage Tracker, Anonymous Community, Personal Chat and Report modules. Meanwhile, Chia Ming Yi is responsible for development of Goal Assistance (shared), User, Emotion Diary Note, Focus Timer and Report modules.

Chapter 2

Literature Review

2 Literature Review

This chapter will present a comprehensive literature review for supporting the development of BetterU mobile application. In the section of project background, the target market of the BetterU mobile application will be identified and the issues faced by them will be discussed. Not only that, the technology and development tools applied in the system will also be explored such as programming languages, development frameworks, backend and API integration, AI algorithms, database solutions and UI/UX design platforms. Each tool section will highlight the contribution to the BetterU mobile application features. Eventually, the feasibility study will be carried out to evaluate the project's technical, economic, operational and schedule feasibility.

2.1 Project Background

2.1.1 Target Market of the Proposed System

In order to address the current issues faced by students and office workers, BetterU mobile application is designed for them to provide an appropriate solution to improve their productivity and mental well-being. BetterU targets users who often juggle multiple responsibilities and require a unified platform to manage goals, track time usage, maintain emotional health and stay connected to the society.

1. Students

Demographic Analysis

BetterU is especially designed for students who are between the ages of **13 - 25 years old** as most of the students within this age range are **owning a smartphone** and **engage the most with the mobile applications**. The age group who will use mobile phone applications will also gradually expand following the evolution of technology. Meanwhile, they also **own a desktop or laptop** for completing assigned homeworks. Usually, the signs of learning stress in students will gradually become apparent and serious when they are pursuing studies in **secondary school, pre-university, colleges or universities** because the complexity and amount of the school work and examinations are getting higher and higher. Within this education level range, students are **keenly needing a suitable task management tool** for organizing their tasks clearly and efficiently. Meanwhile, the strength and potential of task management applications can be represented significantly for helping students to solve their issues. BetterU mobile application is focusing on the students who are living in **urban and semi-urban areas** since a **stable and strong Internet connection** is available in these areas and it may be required by the operation of BetterU mobile application.

- **Age Range**: 13 - 25 years old
- **Education level**: Secondary school, pre-university, colleges and university students

- **Occupation**: Full-time students
- **Location**: Primarily urban and semi-urban areas where mobile app usage with Internet connection is prevalent
- **Device Access**: Regular access to smartphones and laptops or desktops which supported by common operating system such as Android

Psychographic Analysis

From the aspect of students' lifestyle, they frequently deal with large amounts of **schoolwork** and participate in a lot of **co-curricular activities**, especially for students who are pursuing studies in high ranked schools or universities. Some of the students even need to deal with **part-time work** during weekends to earn money to continue their studies. This indicates that their **schedules are extremely squeezed** and **cause a lot of stress**. At the same time, this would significantly **exploit students' time for socialization** as well. In order to ensure academic success and effective time management, they would require a **good planning of work schedules** for handling or accomplishing all the tasks within the deadline in a well-planned way. Apart from academic success, they will also need to **maintain their emotional balance and connection with peers or others** for ensuring their self-confidence and mental health. However, most of the students would face the issue of **procrastination** and **huge effort to organize tasks**, this will lead to the failure of completing tasks and increase the stress due to increment of unorganized tasks. Due to a high sensitivity of students towards the surrounding environment, they will probably **lose focus** or be **distracted by incidents at the surroundings** when dealing with tasks.

- **Lifestyle**: Busy student lives with schoolwork, co-curricular activities, exams and part-time work
- **Goals**: Academic success, improved time management, emotional balance and peer connection
- **Values**: Self-improvement, productivity, efficiency, mental health awareness
- **Challenges**: Struggling with procrastination, lack of organization, stress from multitasking and difficulty staying focused

Behavioral Analysis

Nowadays students have a **heavy dependence on mobile devices for studies and productivity**. These can help them to effectively search for online learning resources in text, image or video forms and schedule their tasks reasonably via installed task management applications. Besides, they also access the **social media application** on their devices for communicating with families, classmates or other Internet users. This helps for building their social engagement skills and releasing their stress during school. They usually access mobile applications on mobile devices before schools for **obtaining latest information from social media platforms** or **getting reminders from task management applications**. After school, they will **access entertainment applications** for relaxing themselves or continue **conversation with friends** for sharing their life experiences. Meanwhile, they will also access their task management applications to **check for their pending tasks or assignments** that are required to be completed now. After finishing all the planned tasks, they will go back with the

social media platforms for **social engagement or watching video and latest news as entertainment**. When it comes to mobile applications preferences of students, they may seek **user-friendly, flexible and personalized mobile applications**. These characteristics allow them to easily adapt and adopt a new mobile application via effective graphical design while deciding what they want in the applications via customization and personalization features. When dealing with task management applications, students would desire to get **task management strategies and features which are definitely suitable and simple for them to use**. However, most of the task management applications are not completely designed for them, they usually come with a lot of **additional features which are meaningless** to them. The **high complexity of existing applications** has also led to unwillingness of students to spend efforts for adopting them.

- **Tech Usage**: Heavy smartphone usage for study, productivity, communication and entertainment
- **Tech Habit**: Before schools, after schools or after completing school tasks and assignments
- **App Preferences**: User-friendly, flexible, personalized and simple to use
- **Pain Points**: Existing task management applications are too complex, unnecessary features in the applications affect the learning ability of students

2. Office Workers

Demographic Analysis

Another target market of BetterU mobile application is office workers who are between the ages of **22 to 45 years old**. Most of the office workers at this age range are taking the occupation of **full-time office workers, administrative staff or junior to mid-level professionals**. Meanwhile, the education level of office workers would likely be at **college graduates or higher level**. At this range, many office workers are dealing with **office tasks and meetings** physically in office or remote working places every day. Since the office tasks have **high demand on Internet connectivity**, the targeted office workers will be clustered in primarily **urban and semi-urban areas** with good Internet infrastructure. They often have **access to smartphones, laptops or desktop computers** for managing their office tasks via online. In order to build well-structured schedules, they would also utilize task management applications to maintain their productivity and ensure all tasks are completed before deadlines.

- **Age Range**: 22 - 45 years old
- **Occupation**: Full-time office workers, administrative staff, junior to mid-level professionals
- **Education level**: College graduates or higher
- **Location**: Primarily urban and semi-urban areas with good Internet infrastructure
- **Device Access**: Regular access to smartphones, laptops and desktop computers with stable Internet connections

Psychographic Analysis

Every day, office workers have to deal with a **high load of office tasks with urgent deadlines**, whether in the office or at home. Sometimes, they are even forced to **work overtime** to finish their tasks in the office. So, it is definitely undeniable that office workers will spend a lot of time staying in front of the smartphone and computer's screen for work. Indirectly, they will also **lack time for communication with families and friends**, they are only able to communicate with workplace colleagues regarding working tasks. In order to excellently complete the task, they also have to maintain a good mentality by consuming huge amounts of refreshment food or drinks such as coffees when dealing with tasks. To prevent the continuous mental fatigue, they desire to achieve work-life balance using different ways while ensuring an excellent work efficiency. Not only that, they also desire to **design strategies for reducing stress** during work but still can **promise their gradual career growth**. Thus, they would need to plan perfect task schedules for time and task management so that it can **improve their productivity**. Meanwhile, their **mental well-being can be indirectly enhanced** since working overtime can be avoided and chance has been seized to social interacting with families and friends. However, most office workers **do not have the time or effort for formulating detailed plans** before they start doing their work. Without a detailed task organization, they will be easily getting **confused and losing focus** when deciding which tasks need to be handled first. Even with a carefully tailored plan, it is **difficult for them to balance their work with their personal responsibilities**.

- **Lifestyle**: Busy schedules, long screen time, high mental load, insufficient time for social interaction with families and friends
- **Goals**: Work-life balance, efficiency, reduced stress and career growth
- **Values**: Time management, productivity and mental well-being
- **Challenges**: Insufficient effort and time for planning, lack of focus, poor task organization, difficulty balancing personal and work responsibilities

Behavioral Analysis

Office workers will **frequently use productivity tools** such as Google Calendar, Notion and Slack for **managing their office tasks**. Those tools can act as a reminder for reminding them of tasks pending to be completed or note taking tools for jotting down important notes regarding tasks. From the aspect of communication, office workers in different companies may **use different communication applications** such as Zoom and Microsoft Teams to execute **online meetings**. On usual days, they will **check their smartphones for any office notifications and announcements** before going to work in the morning. In the office, they may **access a calendar app via smartphones or computers to check for pending tasks** to be done today, then **set the reminders or timers for tracking task progress**. During working hours, they will **use online communication applications for assigning or getting tasks** from supervisors and updating progress to supervisors or colleagues. After working hours, they will **access the calendar apps again for recording the pending tasks to be handled** in future days. From the aspect of app preferences, office workers would tend to desire a mobile application with **minimalist design** so all the features in the mobile application can be easily understood and accessed. A **responsive and real-time tracking application** is also crucial for them to efficiently access any features and dynamically update their task progress.

However, the current task management applications in markets are **difficult to synchronize the users' real time activity progress** and **lack of personalization** designated for task management. These issues have made the task management applications not actually "manage the tasks" in a smart manner, it even makes the task management worse as users need to **spend more effort and time to manually configure and sync the task management**.

- **Tech Usage**: Frequent use of productivity apps and communication tools
- **Habits**: Planning their day in the morning, checking notifications and announcements before working, using reminders or timers to stay on track
- **App Preferences**: Minimalist design, responsive, real-time tracking and synchronization
- **Pain Points**: Lack of personalized task management and inflexible task management

2.1.2 Technology and Development Tools Used

Development Platform and Framework

BetterU is developed using the **Flutter** framework with **Dart** programming language. The main reason for choosing Flutter is because It allows the app to be deployed on Android with a single codebase. Meanwhile, Flutter enables a rapid development cycle with ready-made widgets to realize the BetterU solution ideas.

Database

Google Firebase is used by BetterU mobile application for executing real-time data storage and syncing. It is also designed for cross-platform development where we can use the same database to build the BetterU app for Android in different versions. Meanwhile, it is also a free platform which can be easily implemented into the BetterU mobile application. Thus, the application of Google Firebase can effectively reduce the development time and costs.

AI or ML Features

Speech to text

Vosk and **Whisper** are used by BetterU mobile application for realizing the real-time and high accuracy transcription based on voice input. It allows users to easily input all the task items within the application without manually hand-typing text inside. At the same time, they also support language detection which can automatically detect users' speech language and convert them into the corresponding text precisely. (Radford et al., 2023)

Task Extraction

BetterU mobile application also implements **spaCy** for tokenizing and splitting the task inputs by users via speech or manual typing. This helps BetterU to detect and extract the date and time phrases from the task items, then automatically determine the possible date and time from the tasks. Meanwhile, it also helps for cleaning the task description to make the task item more readable and "straight to the point".

Categorization and Prioritization

BetterU mobile application will apply the transformer-based **SetFit classification models** to automatically determine the task category and priority level from transcribed text input by the users. (Tunstall et al., 2022) These lightweight NLP models are trained on a labelled synthetic task dataset and learn to classify tasks such as study, work, personal, exercise and entertainment. Meanwhile, it will also estimate the priority level of the tasks.

Smart Advisor

The Smart Advisor feature will use the **Alternating Least Squares (ALS) collaborative filtering model** to learn the user-task interaction patterns. Via the analysis on a synthetic dataset of historical interactions, the model can identify which types of tasks will likely be preferred by each user and complete successfully.

Matrix factorization will also be used to reveal the hidden relationships. For example, the users who frequently complete "study" tasks are more likely to engage with related task types. Thus, the system will then recommend the tasks which align with the user's productivity habits and behavioral patterns.

Besides, a **rule-based scoring** fallback will be used when there is insufficient interaction data collected. The purpose of this is to ensure robust recommendations even in cold-start scenarios.

Time Optimization

A **hybrid scheduling engine** will be implemented by BetterU mobile application to optimize the user's daily plan. The system will first predict whether a task is likely to be delayed using a Random Forest classifier trained on behavioral and temporal features. Once there are tasks with high delay probability found, they will be passed to a rule-based scheduling engine. The engine will suggest a more suitable timeslot for the user to decide whether to apply this timeslot suggestion.

The scheduling engine will consider the practical constraints such as the working hours, busy windows, existing task blocks, rest buffers and task priority. This can effectively ensure a conflict-free and realistic scheduling without requiring complex optimization solvers.

Task Rescheduling

BetterU implements a **Random Forest classifier** to analyze the behavioral features such as past task completion, delay history, task priority, time-of-day patterns and workload pressure. This model can predict whether a task is at a risk of being delayed or missed.

If the predicted delay probability has exceeded the defined threshold, the **Task Rescheduling Engine** will start generating a new optimal schedule for that particular task. The engine will use rule-based constraints to avoid conflicts, respect user availability, maintain rest buffers and also prioritize the essential tasks.

UI Design Tool

Figma will be used for designing the graphical user interface of BetterU mobile application. It supports collaborative features which allows multiple users to work on the same design at the same time. This would potentially improve the productivity for multiple people for designing different modules' user interfaces. It is also a free UI design tool which helps BetterU solution to minimize the designing cost. Since it is a cloud-based platform, users can access it from anywhere and anytime using any smartphone, tablets or laptops. At the same time, it also allows users to create interactive components which can change state upon user interactions. Eventually, a more dynamic experience can be presented during prototyping of BetterU mobile application.

2.2 Literature Review

2.2.1 Programming Languages and Frameworks

Dart Programming Language

Dart is a client-optimized and object-oriented language developed by Google for creating a fast and scalable application on any platform. (Dart, n.d.) It helps BetterU to build the logic and code to realize all the planned features of the application. Via integration with Flutter, it allows developers to integrate the code logic into the interactive UI when adapting to different platforms.

Powerful Performance

Dart utilizes the Ahead-of-Time (AOT) compilation to compile the code to native machine code. (Nuraly, 2020) This can effectively increase the app startup times and maintain a smooth runtime performance. Thus, the mobile devices can easily run the mobile apps built with Dart efficiently with seamless user experiences. Meanwhile, it also supports Just-In-Time (JIT) compilation during development. So, developers can immediately view the real impact on the mobile apps after changing the code without restarting the apps. (Valeriu Crudu, 2024) It can increase the speed and productivity to develop, debug and test the applications.

Cross-Platform Development

Dart is designed to work seamlessly with Flutter which enables developers to write a single codebase that runs on both Android and Windows platforms. (Eleftheria Drosopoulou, 2024) These features allow BetterU to develop a mobile application using one type of code but able to adapt to different platforms. This unified approach can save a lot of time and effort for developing and maintaining the performance and appearance of mobile apps.

Object-Oriented and Familiar Syntax

Dart is an object-oriented language with syntax similar to Java and C++ so the BetterU developers can easily learn and transition from any object-oriented language to Dart. (Valeriu Crudu, 2024) This potentially reduces the learning curve for developers. Not only that, it can also help catch the potential errors early in the development process which helps to build a more reliable code since Dart is a strongly typed language where meaning variables have defined types. (Eleftheria Drosopoulou, 2024) For example, it can support all OOPs concepts such as classes, inheritance, interfaces and optional typing features. (Alias & Swathiga, 2021)

Python Programming Language

Python is a versatile programming language that is highly suitable for mobile application development such as BetterU mobile application. (Manish, 2021) It mainly serves for building the backend logic and AI features in BetterU mobile app development. It is also an easy-to-learn programming language which allows developers to quickly develop powerful functionalities on the mobile app. (Python for Mobile App Development in 2023 – Kivy vs. BeeWare, n.d.)

Powerful AI and NLP Libraries

Python comes with rich and popular AI and NLP libraries such as spaCy, Scikit-learn and SetFit. These libraries can help BetterU mobile applications to implement most of the AI features such as speech-to-text, task extraction and task categorization. (Raschka et al., 2020) Meanwhile, these libraries are also useful for assisting BetterU to develop its own designated model via applying different algorithms.

Backend Development

Python is also useful for developing the backend logic for BetterU mobile applications via integrating with FastAPI. Python can easily implement the AI-powered logic and services into the mobile application (Flutter + Dart) through building APIs using FastAPI for connecting mobile frontend (Flutter + Dart) and backend services. (FastAPI, n.d.) Besides, python also applies simple and readable syntax which allows a fast development process and high accessibility of AI models. (Saabith, Fareez, & Vinothraj, 2019) Thus, rapid prototyping can be achieved to present BetterU mobile application's features, especially AI features.

Clean Separation of Concerns

Python is mainly used for developing AI features and backend logic in BetterU mobile application development. So, a clean separation of concerns can be achieved where Dart with Flutter Framework will be responsible in handling most of the mobile frontend user interfaces and simple logic handling while Python with FastAPI will be responsible in handling backend, AI and ML logic and API serving. Thus, it could make the whole system easier to manage, debug and test since each part of the application is responsible for only several specific types of tasks.

Flutter Framework

Flutter is an open-source UI software development kit which is developed by Google for building high-performance, natively compiled applications across multiple platforms from a single codebase. (Flutter, 2024) For example, mobile phone operating systems Android and laptop operating systems Windows. It helps in developing mobile applications by integrating with codebase in Dart language.

Cross-Platform Development Efficiency

Flutter enables developers to develop a Flutter application across multiple platforms using a single codebase. The developed application can run seamlessly on different platforms such as Android, iOS, web and even desktop. Meanwhile, the application can always maintain a consistent behavior and appearance across different devices and operating systems. (Why Choose Flutter? 6 Top Reasons to Use Flutter for Mobile App Development | Monterail Blog, 2025) This strength can effectively reduce the effort and time required for BetterU to develop the same applications in different platforms using different codes. It can also minimize the maintenance effort of BetterU when dealing with different platforms and devices since all of them are relying on the same codebase.

Abundant and Customizable UI

Flutter comes with rich and customizable widgets which allows BetterU to create visually interactive and intuitive user interfaces. (Bhagat, 2022) Thus, BetterU developers can easily build an effective user interface from scratch and engage users. The high customization of the user interface also allows BetterU to create its own designated components and UI design. Meanwhile, it also supports high-performance animations in the user interface to offer a smooth and lag-free experience to users while managing tasks and maintaining well-being.

Hot Reload for Fast Development

The hot reload feature in Flutter enables developers to make real-time code changes and immediately review the code result without having to restart the whole application. (Bhagat, 2022) This can enhance the productivity of BetterU mobile application development since the time for restarting the whole application once any code changes are made can be eliminated. It is also especially useful for front-end developers in BetterU to execute any minor adjustments on the user interface layout and component styles. The impact can be directly observed after adjustments are made.

2.2.2 Development Tools and IDE

Android Studio

Android Studio is an official Integrated Development Environment (IDE) provided by Google, it is designated for developing the user interface of mobile applications. The reason for choosing Android Studio as IDE for developing BetterU mobile applications is it provides many useful features for frontend development such as Flutter Plugin Integration, Built-in Emulator and Intelligent Code Editor. Under this comprehensive environment, BetterU can easily build a cross-platform mobile application frontend without switching between different IDE.

Flutter Plugin Integration

Since Android Studio has offered the plugin for Flutter development, it can perfectly fit with the mobile application built with Flutter with Dart programming language via installing the Flutter plugin. Android Studio can automatically provide the basic project structure with necessary files and configurations for the Flutter project. (Android Studio & Flutter: The Complete Guide | Cellular Insider, 2024) At the same time, it can also support the behavior of Flutter and Dart in which it can achieve hot reload, the implementation of code changes can immediately be shown on the UI preview section without restarting the application. (Android Studio & Flutter: The Complete Guide | Cellular Insider, 2024) These functionalities can effectively help BetterU to develop the mobile application in a faster and efficient way.

Built-in Emulator

In Android Studio, users are allowed to create and manage multiple Android Virtual Devices (AVD) to simulate different Android devices such as Phone, Tablet, Wear OS, Desktop and other common devices. (Create and Manage Virtual Devices | Android Developers, 2019) Meanwhile, it also allows users to define the hardware profile of devices such as System images used with different API levels. These flexible configurations allow BetterU to easily present and evaluate the effect of code implementation on different devices with different scales or versions. Thus, developers can directly test out and debug all the app features on different virtual devices flexibly without requiring the physical devices.

Intelligent Code Editor

Android Studio is able to provide the intelligent code completion for Dart and Flutter widget editing apart from supporting the Flutter and Dart code implementation. (Android Studio & Flutter: The Complete Guide | Cellular Insider, 2024) It provides the debugging tools such as breakpoints, variable inspection and even step-through debugging for streamline frontend development. This feature is developer-friendly for applying testing and debugging the mobile app frontend logic issues. It helps BetterU mobile applications to quickly accomplish the code and find out the potential problems within the code implementation.

Visual Studio Code (VS Code)

Visual Studio Code (VS Code) is code editor tool chosen for BetterU mobile application backend development. It is mainly used for developing the backend logic and services behind the mobile application using Python programming language. Visual Studio Code has also offered a lot of efficient plugins and environment for creating the backend functionalities which allows BetterU for connecting the backend code with the Flutter frontend code.

Integrated Terminal

Apart from offering a code workplace for developers to type codes, it also integrates with the terminal which can significantly expand the functionalities. The integrated terminal allows BetterU developers to directly execute actions such as targeting specific directory, creating new environments for testing app features, install and upgrade the AI or ML packages and libraries and so on. (Integrated Terminal in Visual Studio Code, n.d.) Via an integrated terminal, there is no need to open a Windows' terminal separately for executing the same actions. Not only that, VS Code also allows users to create multiple terminal tabs for executing different commands at the same time.

API Development

VS Code will also be used for developing the Python backend framework such as FastAPI for BetterU mobile application. It supports real time error checking on the Python code after the installation of the Python extension. Thus, it helps BetterU to spend less time to develop the API as a bridge to link between Flutter mobile application and the Python backend features. especially for the data handling and AI-powered features.

GitHub Desktop

GitHub Desktop is an open-source software which allows developers to work with repositories and manage the code changes via a user interface. (GeeksforGeeks, 2024) The main purpose for BetterU mobile application to use it is it can help for tracking all the code changes, solving the conflicts in code and act as a cloud backup. It is especially useful when there is a collaboration team with multiple developers working on the code at the same time.

Branch and Commit Management

GitHub Desktop has simplified the staging changes and written descriptive commit messages for ensuring a better collaboration and history tracking. (GeeksforGeeks, 2024) It also supports multiple branch creation, switching and merging within the same repository. Thus, BetterU can easily manage separate different features and fix the bugs independently before combining them into one. Every code change uploaded will be treated as a commit with a

descriptive message, it will ease the developers for reviewing back the code changes made before or even roll back the mistakes.

Conflict Resolution

Since there are more than one BetterU developer working on the same code, code conflict must be unavoidable. Thus, GitHub Desktop has provided sufficient tools to help solve the merged conflict for keeping the project stable. (GeeksforGeeks, 2024) It lets developers decide whether to overwrite it with the latest code or current code change or manually modifying the code.

Improved Collaboration

When anyone has made any changes to the code and committed it, other developers can immediately sync the code via pulling the changes. (GeeksforGeeks, 2024) So, this allows BetterU developers to always keep the code updated and minimize the risk of code clashing and missing among each other. When there is any code change needed to be committed, they can also immediately push the changes to the GitHub repository and allow other developers to pull the latest code into their current local project.

2.2.3 Backend and API Integration

FastAPI

FastAPI is a modern, fast (high-performance), web framework for building APIs with Python based on standard Python type hints. (FastAPI, 2023) It is highly suitable for being implemented into BetterU mobile application for acting as a bridge between mobile frontend (built with Flutter framework + Dart language) and backend logic (built with Python language). For example, BetterU mobile application heavily relies on different AI-driven features such as task extraction, speech recognition and task categorization. So, FastAPI can easily expose those Python-based AI features as web API endpoints. The Flutter app can easily communicate with FastAPI over HTTP/HTTPS using the http package in Flutter and JSON responses from FastAPI. (Espindola, 2025)

2.2.4 AI Tools and Algorithms

Vosk

Vosk is an offline speech recognition toolkit which supports multiple languages. (Puddot, 2025) It assists BetterU to achieve the speech-to-text conversion features during task inputting without requiring manual typing. It is mainly used for achieving the real-time voice to text transcribing based on what users spoke.

Whisper

Whisper is also being used by BetterU mobile application development. It is another speech recognition tool for accurate transcription of voice notes or conversations. (OpenAI, 2022) BetterU mobile application combines Whisper and Vosk to achieve a transcription which is real-time but accurate when detecting the language of voice input by users whether it is English or Mandarin. (Ashraff, 2025) This tool can effectively help users to reduce their efforts on input tasks into the BetterU mobile application.

spaCy

spaCy is a powerful Natural Language Processing (NLP) library used for linguistic analysis. (spaCy, 2015) It is helpful for BetterU mobile applications to process the transcribed text from Vosk and Whisper and extract the task description, deadline date and time. Meanwhile, it also helps BetterU mobile application to perform tokenization, part-of-speech tagging and syntactic parsing for understanding the sentence structure. (spaCy, 2015) Thus, BetterU mobile application can smartly extract the important points from the voice input by users, users do not need to assign the deadline manually by themselves.

dateparser

dateparser is a Python library that can interpret natural language date or time expressions. (Using DateDataParser — DateParser 0.5.1 Documentation, 2015) BetterU mobile app will integrate it with the spaCy library to help convert the phrases like "next Friday", "at 3 pm" and "tomorrow" into structured datetime objects. The combination of dateparser and spaCy can help BetterU mobile app to extract and normalize dates and time from user inputs in a more precise manner. https://dateparser.readthedocs.io/en/v0.5.1/usage.html

Scikit-learn

Scikit-learn is a versatile machine learning library for Python to execute feature extraction and model training for simpler ML components. (scikit-learn, 2019) BetterU uses it for building a predictive model to predict different task's category and priority based on what task item has been entered. For example, different tasks can belong to different categories (Study, Work, Entertainment, Exercise or Personal) and priority (P1, P2, P3 or P4).

SetFit

SetFit is a tool that helps in fine-tuning sentence transformers, making it easier to build classification systems. (YOUSSEF CHAMRAH, 2024) Thus, it is highly suitable for BetterU to fine-tune the task categorization model with a small labeled dataset via collaboration with Scikit-learn. This tool is especially useful for the Goal Assistance module to recognize and classify different tasks' categories and priorities.

2.2.5 Database and Data Storage

Google Firebase

Google Firebase is a comprehensive backend-as-a-service (BaaS) platform commonly used for mobile app development, including BetterU mobile application development. (Tamplin, 2016) It is also a very reliable database platform for providing scalable, secure and real-time database and data storage solutions. This is suitable for BetterU mobile app to store and retrieve its productivity and well-being data.

Realtime Database and Firestore

Google Firebase implements a cloud-hosted NoSQL database that stores data as a large JSON tree. (Firebase, 2019) Thus, it is extremely flexible for BetterU to create and manage the database with fast retrieval. Since it is designed for real-time synchronization of data across connected clients, it can always keep the BetterU mobile app updated instantly without manual refresh. (Richman, 2023) For example, it is ideal for BetterU to store and retrieve the

task data, user information and preferences. Any changes on the attributes' values will be immediately updated on the Google Firebase database. Meanwhile, Google Firebase supports a low latency and efficient real-time data syncing. (Firebase, 2019) Thus, it is applicable for BetterU to store the data which is frequently updated such as posts and messages data in Anonymous Community and Personal Chat.

Authentication and Security

Google Firebase has implemented an easy-to-use authentication service which supports multiple sign-in methods such as email or password, Google, Facebook and other common third-party providers. (Firebase vs Firestore - Which Is More Reliable for Real-Time Applications?, 2024) It can also help BetterU mobile application development to enforce security rules seamlessly on Realtime Database and Firestore. So, only identified and authorized users can access the specific data within the database.

Cloud Storage

Firebase Cloud Storage is a scalable object storage service designed for storing large files such as images, audio and videos. (Firebase, 2019) This is because its ability of data handling for large amounts of unstructured data is more powerful than the traditional RDBMS which always struggle with unstructured data such as videos, images and audio. (Khawas & Shah, 2018) Meanwhile, it can also offer secure file uploads and downloads directly from mobile clients. Thus, it is ideal for BetterU to store large amounts of media especially when dealing with posts and media files uploaded in Anonymous Community and Personal Chat.

2.2.6 UI and UX Design

Figma

Figma is a web-based design tool mainly used for designing user interface (UI) and user experience (UX) of mobile applications, including BetterU mobile application. It provides abundant design tools and components which allows BetterU to design interactive prototypes or mockups before code implementation on Flutter. Not only that, it also provides a collaborative platform for multiple designers to work together on the same page at the same time.

Seamless Collaboration

Figma allows real-time collaboration where multiple BetterU designers can work simultaneously on the same design file, with each user's cursor visible in real time and uniquely identified. (Figma: The Ultimate Design Collaboration Platform | Real-Time Workflow, Prototyping, and Whiteboarding | Jimmy Viquez, 2025) This means that multiple users can edit together to improve the speed and productivity of building the prototype. Meanwhile, it also provides version history which allows for tracking changes and even reverting to previous versions. (The Design Project, n.d.) Thus, it can promise the manageability and safety of a design file when editing the design file.

Consistent and Scalable Design System

Figma supports the creation of design systems through reusable components and variants. (The Design Project, n.d.) This can help the BetterU mobile application to maintain the

consistency of standardized UI elements across all features such as buttons, icons and input fields. It also enables BetterU designers to update a component once and immediately propagate the updates throughout the entire application. This can effectively reduce the manual effort and human errors when designing BetterU mobile application user interface.

Rapid Prototyping

Figma provides powerful prototyping tools that allow BetterU designers to create interactive, responsive and clickable prototypes without writing the code. (The Design Project, n.d.) It helps BetterU to simulate the user flows and animations within a short period of time before implementing Dart code. It is very efficient when it comes to executing early usability testing and validation of design concepts before development begins. (The Design Project, n.d.) Meanwhile, it can also minimize the misunderstanding and rework on development code since the prototype can quickly let reviewers or supervisors glance through the overall design.

2.3 Feasibility Study

2.3.1 Technical Feasibility

BetterU mobile application is technically feasible since it utilizes the popular and widely supported development tools and platforms. From the aspect of hardware, BetterU mobile application is **compatible with standard mobile devices** such as smartphones with Android and portable devices such as laptops with Windows operating system since it is developed with Flutter which supports cross-platform application. For design and development, a **Windows laptop or desktop computer can easily support the user interfaces design, Flutter frontend and Python backend code development**. With the additional GPU, it can help BetterU to develop, train and test the AI models in a more efficient way. During the testing phase, an **Android device can be used as a physical device to test all the features** via the installed BetterU mobile application.

From the aspect of software, there are some development software and platforms required for developing the BetterU mobile app. **Android Studio is required for developing the frontend logic of BetterU mobile app using Dart language with Flutter**. It can also provide virtual devices which simulate the physical device behaviors and patterns, enabling testers and developers to debug and review the code impact efficiently. Meanwhile, **Visual Studio Code is mainly used for developing the backend features and training AI models** using Python programming language. It will also be used for **building the FastAPI for linking Flutter app with Python backend services**. Within Visual Studio Code, there are also **abundant AI and NLP open-source libraries available** to be installed and utilized for training the AI models and developing the Python backend. For example, spaCy, Vosk, Whisper and SetFit. Not only that, **Figma will be used for designing the user interface** of BetterU mobile application and prototyping. So, it can easily show the overall user experience and workflow before implementing the code.

2.3.2 Economical Feasibility

The BetterU mobile application project is economically feasible. The main reason is all the **tools and software used for developing this mobile application are free and open-source**. For code development, **Flutter, Dart, Python and FastAPI are all open-source software and tools**. Meanwhile, **Google Firebase offers a free cloud service** which includes real-time database, authentication and cloud storage. It can perfectly fit and sufficient for BetterU mobile application to store and retrieve the application data and user information. On the other hand, **GitHub Desktop and GitHub repositories are also available at no cost**. They allow BetterU mobile application developers to easily collaborate when working on the code and keep syncing with the latest code. GitHub also enables developers to achieve code version control and track the code changes for maintaining efficient code management. When dealing with AI features development, there are **free and open-source libraries available** for BetterU developers to implement and develop their own designated AI models and services. For example, Vosk and spaCy are free to use for transcribing the voice input to text smartly and extract the important information from the task text transcribed. In overall, there are **no direct financial costs involved in the BetterU project**. Thus, the BetterU mobile application project is sustainable and affordable for development and implementation.

2.3.3 Operational Feasibility

BetterU mobile application is operationally feasible and it is accessible across the commonly used platforms. Since it is developed using Flutter, it can be **flexibly run on mobile devices with Android operating system**. The mobile app can support the Android device with API level 21 and above. During development and testing, the **app can smoothly run on Android devices with API level 21 and above**. From the aspect of backend services, they will be integrated into the Flutter app using FastAPI. Thus, it allows applications to **access the backend functionalities including AI features via API calls**. For the database storage, **Google Firebase will be applied for providing a real-time database and authentication support**. The application data and personal information will be stored and retrieved back and forth with this cloud and real-time database securely.

2.3.4 Schedule Feasibility

In order to ensure that the BetterU mobile application project is able to be completed within the time range provided, it has been separated into Project 1 and 2. **Project 1** is mainly focusing on identifying and analyzing the objectives, problems and solutions for the BetterU mobile application. Meanwhile, it is also used to **create the plan or strategies for analyzing requirements and designing the system using various diagrams** such as sequence diagram, state chart diagram, software architecture diagram and other relevant design documents. Project 1 will **start working on 14/3/2025** while the **deadline is assigned on 8/9/2025**.

- Proposal (14/3/2025 - 18/4/2025)
- Chapter 1: Introduction (12/6/2025 - 30/6/2025)
- Chapter 2: Project Background (28/6/2025 - 14/7/2025)
- Chapter 3: Methodology and Requirement Analysis (1/7/2025 - 28/7/2025)
- Chapter 4: System Design (10/7/2025 - 18/8/2025)

When it comes to **Project 2**, it will mainly focus on **designing the testing strategies, developing various planned modules using code and supervisor review sessions**. Since the BetterU mobile application project is applying the incremental model, Project 2 will be separated into 3 builds. Within each build, there will be **development, testing and review phases focusing on different modules**. Thus, it can efficiently and effectively ensure the smooth progress for all modules development and testing while avoiding overburdened situations within a phase. Project 2 will **start working on 18/8/2025** while the **deadline is assigned on 19/12/2025**.

- Design Testing Strategies (18/8/2025 - 20/8/2025)
- Chapter 5: Implementation and Testing
    - Build 1: User Module, Goal Assistance Module (22/8/2025 - 11/9/2025)
    - Build 2: Time Usage Tracker Module, Emotion Diary Note Module, Focus Timer Module (11/9/2025 - 1/10/2025)

    - Build 3: Anonymous Community Module, Personal Chat Module, Report Module (1/10/2025 - 28/10/2025)

- Chapter 6: Discussions and Conclusions (29/10/2025 - 7/11/2025)

2.4 Chapter Summary and Evaluation

In overview, BetterU mobile application is mainly targeting two primary user groups which are students and office workers. The main reason is both of them are facing similar productivity challenges due to busy schedules, multitasking and mental fatigue. Via analyzing each user group's patterns in aspects of demographic, psychographic and behavioral, the result has indicated that BetterU is the crucial solution for solving both user groups' issues by improving their productivity and connectivity with society.

Based on the literature review, there are some core technologies that have been highlighted and selected by BetterU for development purposes. From the aspect of programming languages and frameworks, Dart and Flutter are chosen due to the excellent cross-platform capabilities and developer-friendly features like hot reload with Just-In-Time (JIT) compilation. Meanwhile, BetterU also selected Python as the programming language for developing backend and AI functionalities. On the other hand, Android Studio, VS Code, Github Desktop have been chosen as development tools and IDE. They offer an excellent development environment which enables developers to collaborate with each other to write and manage the code. For the AI tools, Vosk, Whisper, spaCy, SetFit and Scikit-learn have been chosen for developing advanced features such as speech recognition, natural language task extraction and intelligent task categorization. Apart from that, Google Firebase was selected for its real-time database, authentication and cloud storage support. At the same time, Figma was chosen as the main UI and UX design tools due to its abundance of reusable components, collaborative and prototyping capabilities.

Lastly, the feasibility study has concluded that the BetterU mobile application is viable from a technical, economic, operational and schedule perspective. The current existence of development hardwares and softwares has directly supported the technical feasibility such as Windows laptop with Windows operating system with installed Visual Studio Code. Besides, the availability of open-source tools and cloud-based services can effectively ensure cost-efficiency during the whole development process. Not only that, the operational feasibility is also easily reached since the BetterU mobile application is supported by Flutter, FastAPI and Google Firebase cloud storage. The BetterU project flow is strictly based on the incremental development model which can support systematic progress and reduces implementation risk.

Chapter 3

Methodology and Requirements Analysis

3 Methodology and Requirements Analysis

Chapter 3 will focus on the overall methodology and requirements analysis involved in developing the BetterU mobile application. It will describe and explain the chosen development methodology and how it was implemented within the whole project. Moreover, it highlights the fact-gathering techniques used to identify the system requirements which includes functional and non-functional requirements. On the other hand, this chapter also includes the diagrams used to visualize the system flow such as use case diagram, use case description and functional requirements table. Lastly, it will describe the development environment, including the tools, frameworks and external components that support the implementation of the application.

3.1 Methodology

3.1.1 Selection of Development Methodology

BetterU mobile application has selected the **Incremental Development Model** as the backbone of the whole project development workflow. The main reason is this model enables the BetterU mobile application to be developed and improved in stages which aligns well with the **modular structure behavior and feature complexity of the project**. The BetterU mobile application is divided into many modules such as Goal Assistance, Time Usage Tracker, Anonymous Community, Personal Chat and Report modules.

In order to prevent the overloading of module development at once, the incremental model allows the BetterU mobile application to be broken into **multiple functional builds which focus on different parts of modules tasks**. It does not require all features to be finalized before development begins like the traditional Waterfall model. This approach allows the BetterU mobile application to be developed module by module, so each feature can be tested, improved, and refined before moving on to the next stage. Meanwhile, this structure can also effectively **minimize the risk and challenges for the whole project development**. Each increment undergoes its own cycle of planning, development, testing and review with specified focusing modules such as Goal Assistance and Anonymous Community modules. The **continuous feedback and integration in each increment can help BetterU to identify the issues and improvement required**. This allows timely modifications or enhancements to address issues identified during each cycle. This can also ensure a consistent flow for enhancing the existing functionalities or adding on new features into the system.

Since the development of the BetterU mobile application will be conducted in multiple builds, it **enables developers to arrange the sequence of module development based on the priorities**. The core modules will be developed at the earlier stages as it will be linked with other supportive modules in future. Meanwhile, the supportive modules will be arranged at the later stages for integration with the core modules functionalities or acting as extended features for the core modules.

- Functional Build 1: User, Goal Assistance modules
- Functional Build 2: Time Usage Tracker, Emotion Diary Note, Focus Timer modules
- Functional Build 3: Anonymous Community, Personal Chat, Report modules

3.1.2 Application of Incremental Model in BetterU Development

Requirement Analysis Phase

The requirement analysis phase is the baseline for the entire BetterU mobile application development process. At the beginning of this phase, the development team will **identify the real-world problems faced by people** regarding the topic of task management, productivity and well-being via online research and anonymous observation. Subsequently, they will **propose potential solutions for addressing these problems effectively**. Not only that, the development team will also **identify the target users of the system** by analyzing their background and behaviors for consolidating the development direction of the BetterU mobile application. A **comparison between proposed solutions (BetterU mobile application) and other existing application systems (Google Keep and Confluence)** in the market will also be made. These help for identifying the strengths and weaknesses of proposed solutions.

After the research of real-world issues, target users and existing solutions, the development team will proceed to the steps of **defining the system and user requirements** in BetterU mobile application. It ensures that the final developed system is able to offer the services which satisfy all the requirements with a high quality within specific time and budget constraints. Before gathering requirements, the appropriate **fact-finding techniques must be identified** clearly for effectively collecting the useful information for future system designing. For example, online research and observation. After collecting all the requirements, a **requirement analysis will be conducted by designing the Use Case Diagrams and Use Case Descriptions** and **defining the functional requirements and non-functional requirements**. So, all the user needs can be translated into technical understanding which helps for guiding the whole development process of BetterU mobile application and avoiding ambiguity in later stages.

System Design Phase

Before proceeding to the development of modules, a system design is crucial for transforming the requirements gathered in previous phases into a concrete and structured plan that indicates how the BetterU mobile app will be built. Based on the project schedules proposed for BetterU mobile application, several diagrams will be designed such as **Sequence Diagram**, **State Chart Diagram**, **User Interface Design**, **Data Design**, **Reports Design**, **Process Design** and **Software Architecture Diagram**. These diagrams will act as a structured blueprint which visually shows the workflow and interaction during the development process.

Software Architecture Diagram

The software architecture diagram mainly shows the **high-level structure of the BetterU mobile application with various interactions between each other**. In our case, the architecture will adopt a client-server architecture which separates the frontend interface from

backend processing and data management. Thus, the core components like frontend, backend services and database will be involved in this diagram.

From the aspect of frontend, BetterU mobile application interfaces will be developed using Flutter which allows for cross-platform mobile application. The **backend will be implemented using FastAPI with Python for handling the business logic, API requests and AI-related tasks such as task extraction and categorization**. For the data storage, **Google Firebase will be applied to manage the user authentication and real-time database** such as user preferences data and task items. In overall, the frontend will access the data via calling the FastAPI and enable the Python code to fetch the data form the Google Firebase.

Process Design

Process design will outline the **internal workflows and logical operations of BetterU mobile application** based on different functionalities in different modules. It gives a detailed representation of how data and control flow through different modules when accepting and responding to user actions and system events. Thus, developers will be able to develop the corresponding features based on the workflow overview provided by the process design such as task input, tracking the app activity and interacting with someone in personal chat.

In order to provide a clear representation of data and control flow for each feature, each key feature will be designed in one activity diagram. Thus, every logical step and action sequence is visible, especially when handling conditions and loops. The activity diagrams are planned using **Draw.io** to visualize the logical steps and interactions across modules. There are several key features will be mainly focused when designing the activity diagram:

- Goal Assistance Module
    - Task Entry and Extraction Flow
    - Smart Reminder Scheduling
    - Task Progress Update
- Time Usage Tracker Module
    - Start and Stop Time Tracking
    - NFC-Based Activity Logging
    - Screen Usage Monitoring
- Anonymous Community Module
    - Anonymous Post Submission
    - Post Viewing and Commenting
- Personal Chat Module
    - Sending a Personal Message

    - Emergency Alert Flow
- Report Module
    - Generate Daily or Weekly Report
    - Suggest Improvements Based on Usage

Sequence Diagram Design

Sequence diagram design represents the **flow of interactions between different components of the BetterU mobile application from time to time**. It mainly focuses on how the system parts such as frontend interface, backend server, database and external services interact with each other when being triggered by user actions. This helps for visualizing the order and timing of messages exchanged between these components, making it easier to understand the system's dynamic behavior during runtime.

In BetterU, the **sequence diagrams** can help to visualize the communications between the Flutter frontend, FastAPI backend, Firebase database and other tools such as NLP and AI models. This allows for clearly ensuring the correct flow of API calls, database updates and data responses during runtime.

Each diagram will include the key actors and lifelines for representing the system components with arrows showing what message exchanged between in chronological order. Some examples of the key scenarios can be modeled using sequence diagrams:

- Task entry and extraction
    - User submits task (via text or voice) → NLP processing → datetime extraction → task saved
- Reminder triggering
    - Scheduled reminder triggers notification at the right time
- Anonymous community posting
    - User submits post → system validates → added to feed
- Personal message sending
    - User sends message → message stored → recipient receives it
- Report generation
    - User opens report → system queries tasks logs → data visualized

State Chart Diagram Design

During the development of various features, many objects in the BetterU mobile application will transition between different states based on user actions or system events. Without proper planning and standardization, managing these state transitions could become chaotic and error-prone.

To address this, state chart diagrams are used to **model the different states of key objects and components**, as well as their transitions under various scenarios. These diagrams provide a clear understanding of how an object moves from one state to another, helping developers design consistent and reliable system logic for updating and tracking object states.

In the BetterU mobile application, state chart diagrams are especially useful for modeling behavior in features such as task progress tracking, reminder handling, and chat messaging. The following key objects and their possible states will be represented:

- Task (Goal Assistance Module):
    - Pending → In Progress → Completed → Overdue
- Reminder (Goal Assistance Module):
    - Scheduled → Triggered → Snoozed
- Post (Anonymous Community Module):
    - Draft → Pending Moderation → Published → Edited → Deleted
- Message (Personal Chat Module):
    - Composing → Sent → Delivered → Read → Failed

User Interface Design

User Interface is one of the crucial components in the BetterU mobile application since it is the only part directly visible and interacted with by users. Therefore, BetterU must **ensure that the interface is clean, intuitive and user-friendly**. This allows users to easily understand the layout of each screen and navigate the app's features without any confusion.

In order to design an excellent user interface, **Figma** will be used to design the user interfaces before proceeding to the Flutter frontend development stages. Every key screen of the BetterU mobile application will be designed in Figma with the support of reusable and responsive components. Via the continuous feedback and enhancement made on the high-fidelity prototypes created, the **user interface will be able to cater to the needs of students and office workers**. Each major module will be planned with specific screens with the consistent standards, including:

- Goal Assistance Module: Task creation screen, smart suggestions, task list, and task editing interface
- Time Usage Tracker: Usage history view and NFC tap tracking screen
- Anonymous Community: Post feed, post submission screen, and comment section
- Personal Chat: Chat interface and trusted contact list
- Report Module: Weekly productivity and social performance report and personalized suggestions display

Data Design

Data design is applied in BetterU development to **define how the information is structured, stored and accessed within the mobile application**. BetterU requires efficient and scalable data storage planning since it will be handling different types of object data such as user preferences, tasks items, reminders, posts, messages and other crucial information. Thus, a Class Diagram, Entity Relationship Diagram (ERD) and data dictionary will be designed in advance before development. BetterU mobile application uses **Google Firebase as the primary backend database storage**. The characteristics of flexible and NoSQL document-based structure allows the mobile application to store various types of data when supporting real-time synchronization across devices. The data will be organized into collections while each collection will represent a category of data. Examples of key collections and their document structures include:

- Tasks
    - Attributes: task_id (PK), task_cat_id (FK), member_id (FK), title, description, priority, planned_start_time, planned_end_time, status
    - Associated with the TaskCategory and Member by task_cat_id and member_id
- CommunityPost
    - Attributes: post_id (PK), community_id (FK), member_id (FK), content, like_num, created_datetime, status
    - Associated with Community and Member by community_id and member_id
- ChatMessage
    - Attributes: message_id (PK), chat_room_id (FK), sender_id (FK), reply_to_id (FK), content, sent_datetime
    - Associated with ChatRoom and ChatRoomMember by chat_room_id, sender_id and reply_to_id
- NFCTag
    - Attributes: nfc_tag_id (PK), member_id (FK), task_id (FK), subtask_id (FK), tag_code, label, purpose, created_datetime, status
    - Associated with Member, Task and Subtask by member_id, task_id and subtask_id

Reports Design

Reports design focuses on **generating visual summaries of user behavior and progress to support self-reflection and well-being improvement in the context of BetterU mobile application**. Thus, the reports will be generated based on the Google Firebase data collected by various modules such as goal assistance module and time usage tracker. In order to standardize all reports' design, each report will include:

- Clear title

- Report period or issue date
- Filter options to view reports by month, date range or category
- Summary section which shows the key statistics such as total tasks completed, time spent on activities
- Detailed records such as the list of tasks (completed, overdue or in progress) and time usage breakdown
- Graphical elements such as bar charts, line graphs and pie charts

Not only that, 3 types of reports which are **summary reports**, **detailed reports** and **exception reports** will be generated. The summary reports act as a condensed overview of key metrics which are filtered by date or month. The detailed reports will show the comprehensive listing of entries such as tasks and time usage history. The exception reports will highlight the key insights from the information such as missed deadlines, high time usage on certain tasks or activities and low social performance.

Functional Build 1: User & Goal Assistance

At the starting point of functional build 1, the **testing strategy will be firstly outlined** to ensure that the early builds can meet the quality standards. Various types of testing will be conducted such as unit testing and integration testing. Meanwhile, an appropriate test case design will also need to be prepared in advance. After that, a **system overview session with the supervisor will be conducted** for validating the design and clarifying the expectations for Build 1.

During the development phase of Build 1, the project will **focus on the user module and goal assistance module**. The main reason is that both modules are the core modules in BetterU mobile application. Since users must log in before accessing the useful functionalities in the mobile application, the user module was developed first to handle user registration, login, session management and account preferences. On the other hand, the goal assistance module also takes the same precedence and importance for allowing users to implement task input via text or voice, task extraction using NLP, task categorization and other basic task management features such as task editing, deleting and status updating.

After the development, the **test plan and test cases will be created** to evaluate the functionality and accuracy of existing functionalities such as task extraction and user registration. This can ensure that all the existing features developed can work properly and logically without any significant issues. Once the completion of development and testing, the **build will be reviewed by the supervisor to assess functionality**, adherence to requirements and areas for improvement. The **feedback will be collected and act as the adjustment guidance during the next build**. Via this build 1, the basic system and core task management can be delivered successfully. This is also a foundation for time usage tracker, focus timer and report generation features in future builds.

Functional Build 2: Time Usage Tracker, Emotion Diary Note, Focus Timer

In functional build 2, the development will focus on 3 key modules which are **Time Usage Tracker**, **Emotion Diary Note** and **Focus Timer** modules. The Time Usage Tracker Module monitors how users spend their time by tracking their app usage. Meanwhile, it also offers the NFC tag interactions for task clock-in and clock-in and clock-out features by integrating with the goal assistance module. It can also integrate with the focus timer module to achieve

starting or stopping focus timer via NFC connectivity. Apart from that, the Emotion Diary Note module will also be developed during this build, it enables users to log their daily emotions, write reflective notes and receive the mood scores using the built-in NLP sentiment analysis. This helps for tracking the emotional patterns over time for self-awareness and well-being. Moreover, the Focus Timer module supports the work sessions by providing a gamified countdown timer with start, pause and stop functionality. It also includes the reward mechanisms or progress feedback to encourage productive behaviors. After the development of modules, **all the modules will be undergoing a series of test plans and test cases** to ensure the functionality and integration between modules. Not only that, the **build will also be presented to the supervisor for evaluation for collecting feedback regarding system performance, UI behavior and overall integration**.

Functional Build 3: Anonymous Community, Personal Chat, Report

Functional build 3 is the final functional build which focuses on implementing the social communication and reporting features in BetterU mobile application. The **Anonymous Community**, **Personal Chat** and **Report** modules will be involved in this build. Anonymous Community module will allow users to post and respond anonymously within a community and forum. Users can submit the posts, view others' posts and comment while maintaining their privacy. For the Personal Chat module, it allows the user to conduct a secure one-to-one messaging with his or her selected trusted contacts. For the report module, it will generate the regular summaries based on user activity, task progress, time usage and social performance weekly and monthly. The reports will include the visual elements such as charts, completion rates and personalized suggestions to encourage the improvement of users.

3.2 Requirements Gathering Techniques

3.2.1 Observation

In order to better understand and explore real challenges faced by the potential users, informal observations were carried out within the Author's daily environment. For example, close friends and family members. Thus, the appropriate functional and non-functional requirements can be extracted and implemented into BetterU mobile application. Not only that, this can make sure that the BetterU mobile application is able to really offer an effective solution for solving the problems faced by the potential users.

Poor Task Arrangement and Time Mismanagement

From the Author's observation, many of his friends tend to struggle with arranging their tasks properly. This often leads to missed deadlines or a significant drop in the quality of their work. Even when they are aware that they need to plan their tasks, they do not seem to have a strong sense of time management. Some tasks that only require a short period are given too much time while more complex tasks are rushed or delayed. This often results in procrastination and an overwhelming workload near the deadline.

Trouble with Task Recording Habits

The Author's elder brother and father always lack consistent habits when it comes to recording their tasks including work and house work. They always think that the process of organizing and managing tasks are so troublesome. Instead of using a proper app or tool, they tend to jot down their tasks on random pieces of paper or in messy, unstructured note apps on their phones. After a period of time, they either forget where they wrote the notes or cannot understand their own writing when they finally find it. This significantly causes confusion and missed responsibilities.

Distraction from Mobile Apps

There is another common issue the Author noticed among the Author's friends is how easily they get distracted by entertainment apps on their phones while trying to complete tasks. Without external reminders or team supervision, they often fall into the habit of scrolling through social media or playing games. As a result, the progress is slowed down and a lot of valuable time is wasted. These distractions not only affect their productivity but also build up unnecessary stress as deadlines approach.

Lack of Time for Social Life

The Author's elder brother's daily routine shows how work-life balance is becoming a challenge for working adults. He leaves for the office early in the morning and only returns at night. By the time he finishes dinner, he usually just scrolls through social media briefly before heading to bed. Sometimes, he even shares his troubles with the Author about his serious lack of time for catching up with his old friends or maintaining his social connections. Over time, this lack of social interaction may negatively affect his mental well-being.

Fear of Reaching Out for Help

There were also moments when the Author's brother faced challenges at work but hesitated to ask for help. As someone who is more introverted, he feels uncomfortable asking questions on public forums or even approaching colleagues in person. He is afraid of being judged or criticized since his name or identity is visible. This hesitation can delay his problem-solving and increase his stress, especially when deadlines are difficult or tight deadlines.

3.2.2 Online Research

For exploring the issues and requirements of users in various and deeper horizons, the online research is conducted for gathering and analyzing the reviews and issues faced by potential users when dealing with task management tools. Via this technique, BetterU can more comprehensively understand user needs, common challenges and useful features found in popular task management applications. Meanwhile, BetterU can implement those useful features for fulfilling users' requirements and satisfaction.

Most apps are built for teams, not individuals

There are many popular task management applications such as Asana, Notion and Trello that are packed with features. However, they are often designed for team collaboration instead of individual use. Based on the user comparisons and app reviews, these tools can feel overly complex or overwhelming for personal daily planning. (GetApp, n.d.; Tech.co, n.d.) Several users commented that they just wanted a simple system to track their tasks but still had to deal with team functions like project timelines and shared dashboards. (Software Advice, n.d) For students or solo office workers, this complexity makes the tools less practical and appropriate for personal use.

Too much manual effort involved

One of the most common complaints seen in blog articles and app reviews is the amount of manual work needed to use productivity apps effectively. The users must enter tasks one by one, then define the due dates, set reminders and organize their task lists regularly. Many users had mentioned that the time spent managing the app sometimes outweighed the benefit of using it. (Smartsheet, 2023; FlowWright, n.d.) The research from productivity platforms also showed that workers spend a large portion of their week which is up to 25% on repetitive manual tasks that could be automated. (Dev.to, 2023) However, most current applications still lack the intelligence to automate or simplify these processes.

Difficulty in prioritizing and planning

Most of the applications are unable to assist users in deciding which task to prioritize or how much to assign to it. The reviews and feedback from forums has stated that users often feel overwhelmed because apps leave all the decision-making to them. (NNG Group, n.d.; IJSRD, 2024) There is a user who has stated that although their app could set reminders, it did not really help them to plan when or how long to work on each item, making it feel more like a static list than a helper planner. (Reddit, 2024)

Lack of a safe and anonymous support space

Many users also face emotional struggles when using productivity tools. Some feel discouraged from asking for help because they worry about being judged or exposed. On

platforms like Reddit or discussion boards, users have to sign in or use their real usernames. This may reduce their comfort level in sharing personal concerns. (Pew Research Center, 2013; Mozilla Foundation, 2023) Even mental health apps have been criticized for poor privacy protection which makes users hesitate to open up in those spaces. This indicates a clear need for anonymous environments where users can discuss problems safely without exposing their identity. (Avast, 2023)

3.2.3 Summary of Fact Gathering Results

Fact-Finding Technique	Key Observations or Findings	Implications for BetterU Functionalities
Observation	Many users struggle with poor time management, leading to procrastination and low-quality work.	Include smart task scheduling and reminders to improve time awareness and reduce last-minute rush.
	Users find it troublesome to manually record tasks and often rely on scattered notes.	Offer a quick task input interface, including voice command or minimal-typing input options.
	Users easily get distracted by entertainment apps and lose focus while working.	Implement productivity-focused features like scheduled focus sessions timer and time usage tracker.
	Working adults lack time for social interaction due to long work hours and routine fatigue.	Introduce the anonymous community with social suggestions for self-care and break management.
	Some users are introverted and afraid to seek help publicly due to fear of being judged.	Provide an anonymous community or in-app support forum where users can express concerns safely.
Online Research	Most task apps are designed for teams, not for individuals. Thus, they feel too complex for personal use.	Design a clean and individual-focused interface with simplified task views and user-friendly layout.
	Too much manual effort is needed to manage tasks which makes users feel like the tool is a burden.	Integrate the automation features such as auto-categorization and auto-scheduling of tasks.
	Users find it hard to prioritize or estimate time for tasks.	Add smart prioritization logic and AI-based suggestions for estimated time and daily schedules.

	Users want a safe space to express stress or concerns without exposing identity.	Build a secure and anonymous community and personal chat which allows users to share their thoughts comfortably.

Table 3.2.1: Summary of Fact Gathering Results

3.3 Requirements Analysis

3.3.1 Use Case Diagram

System Overview Use Case Diagram

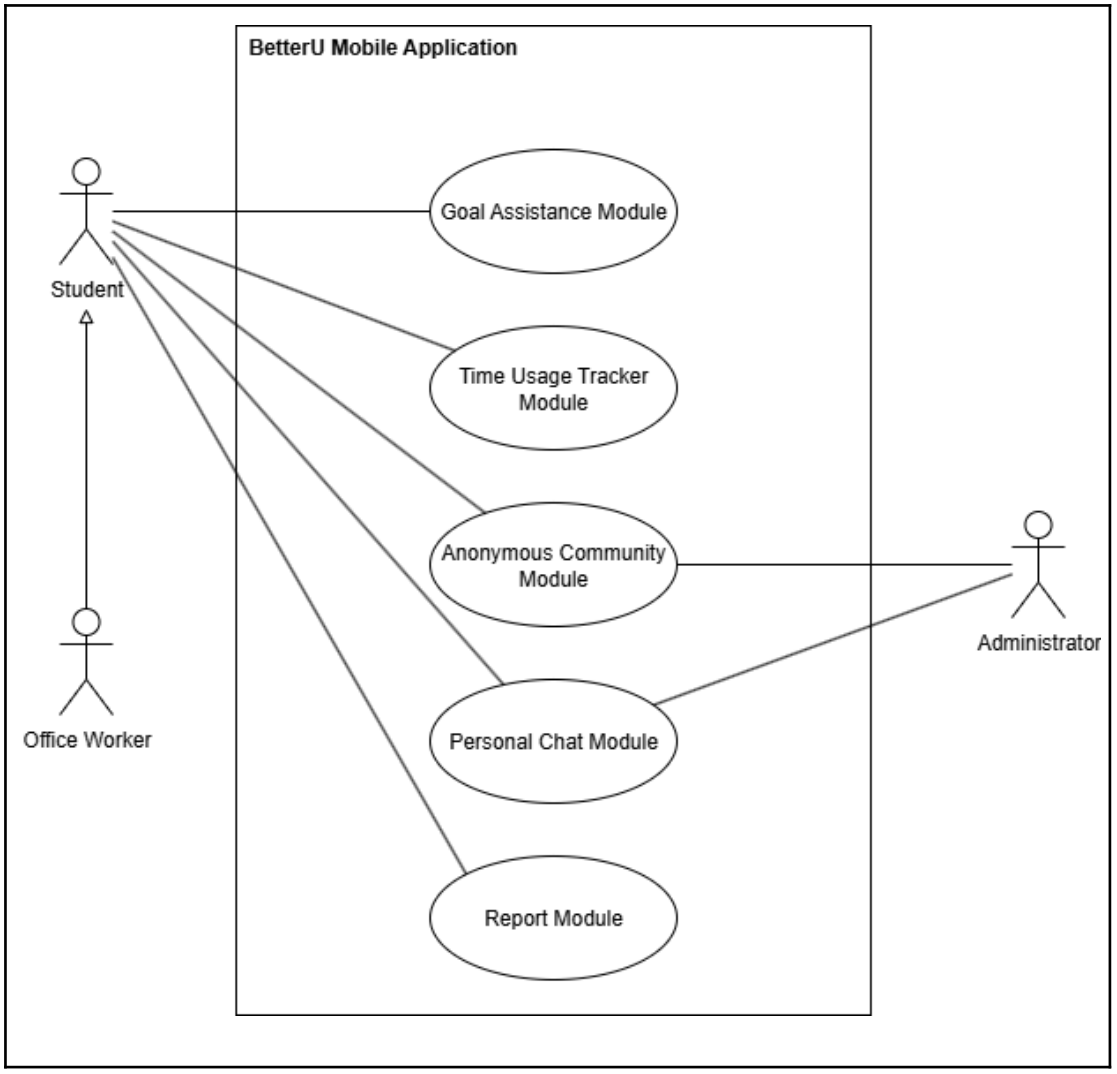


Figure 3.3.1: System Overview Use Case Diagram

Goal Assistance Module

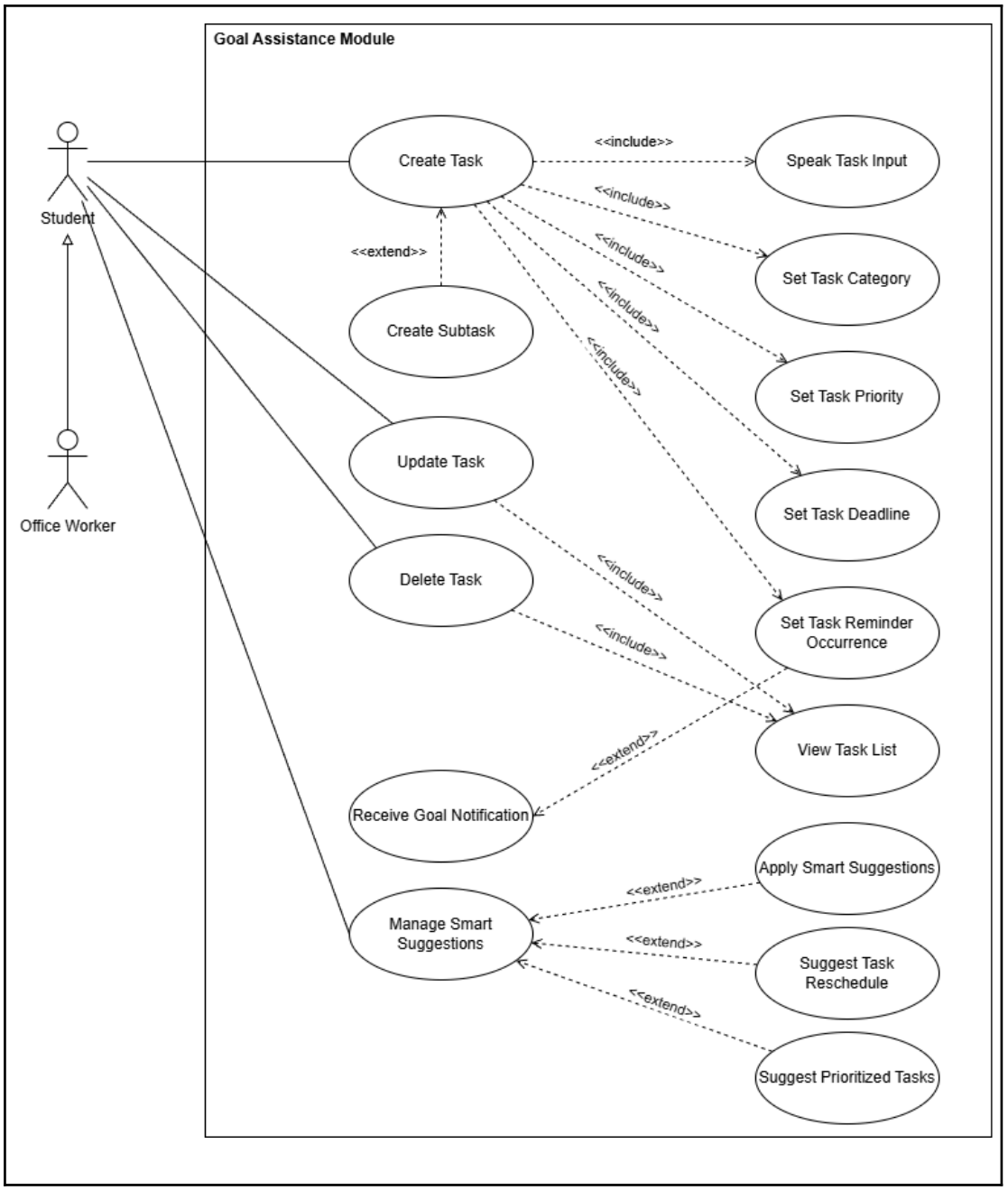


Figure 3.3.2: Goal Assistance Module Use Case Diagram

Time Usage Tracker Module

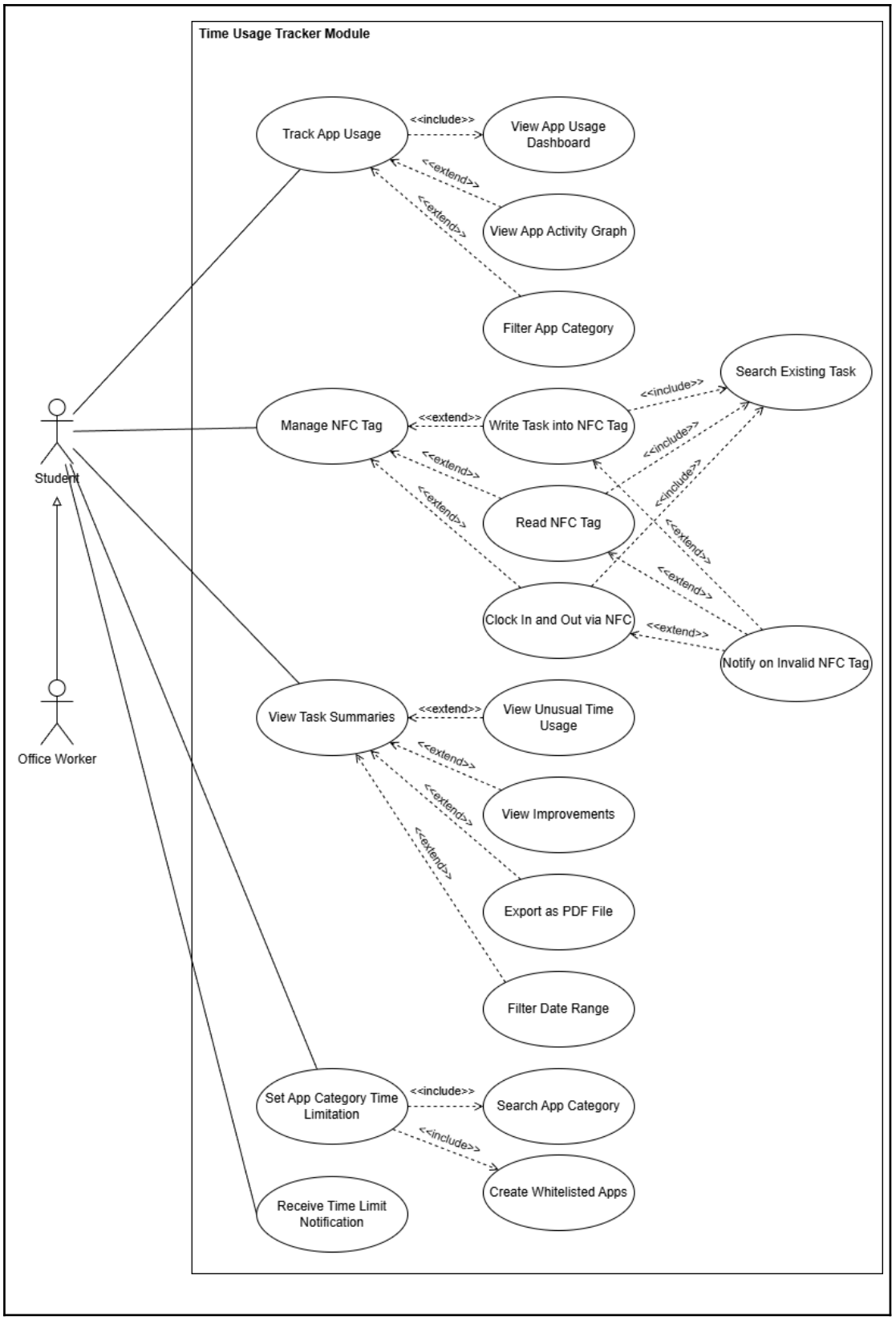


Figure 3.3.3: Time Usage Tracker Module Use Case Diagram

Anonymous Community Module

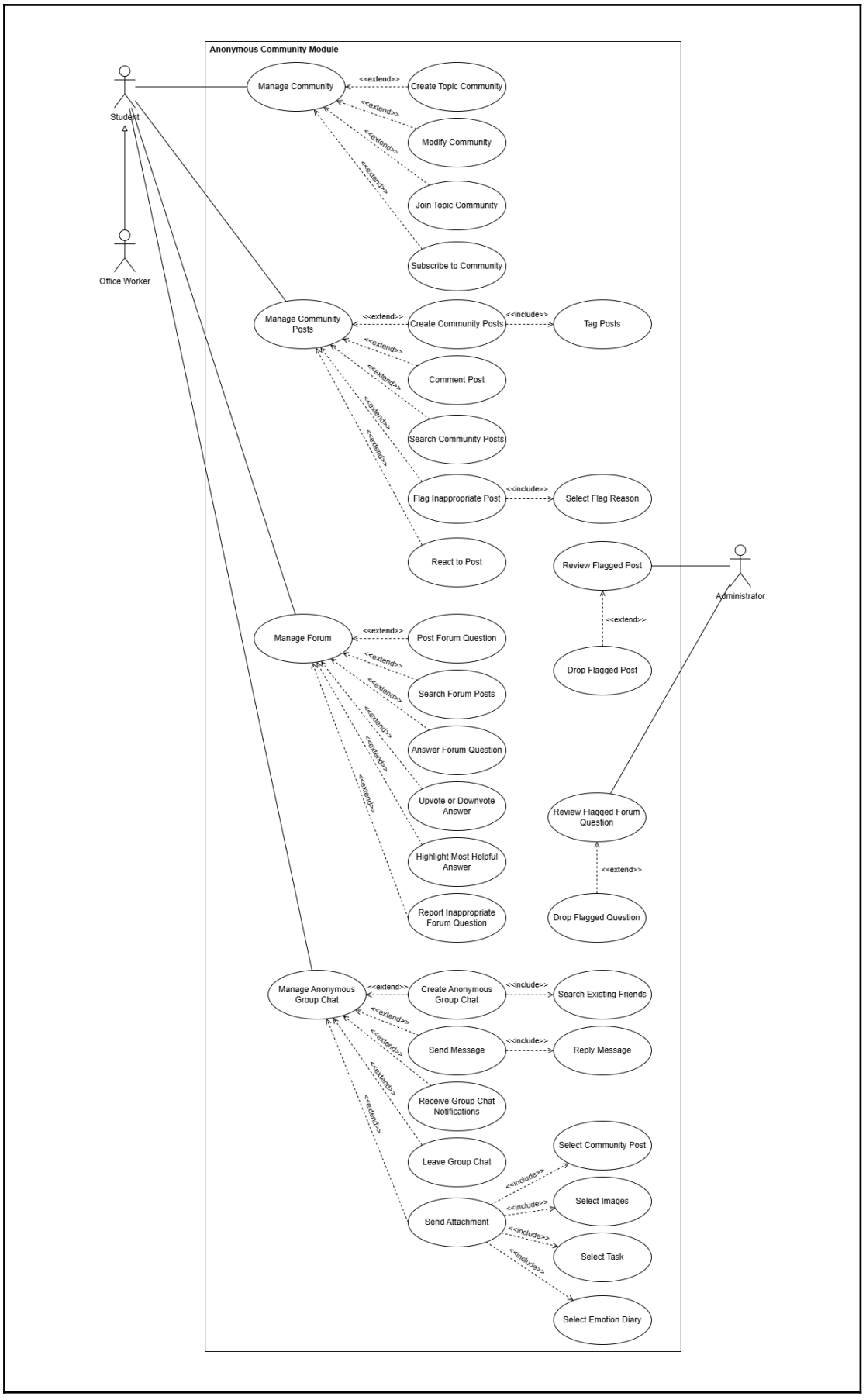


Figure 3.3.4: Anonymous Community Module Use Case Diagram

Personal Chat Module

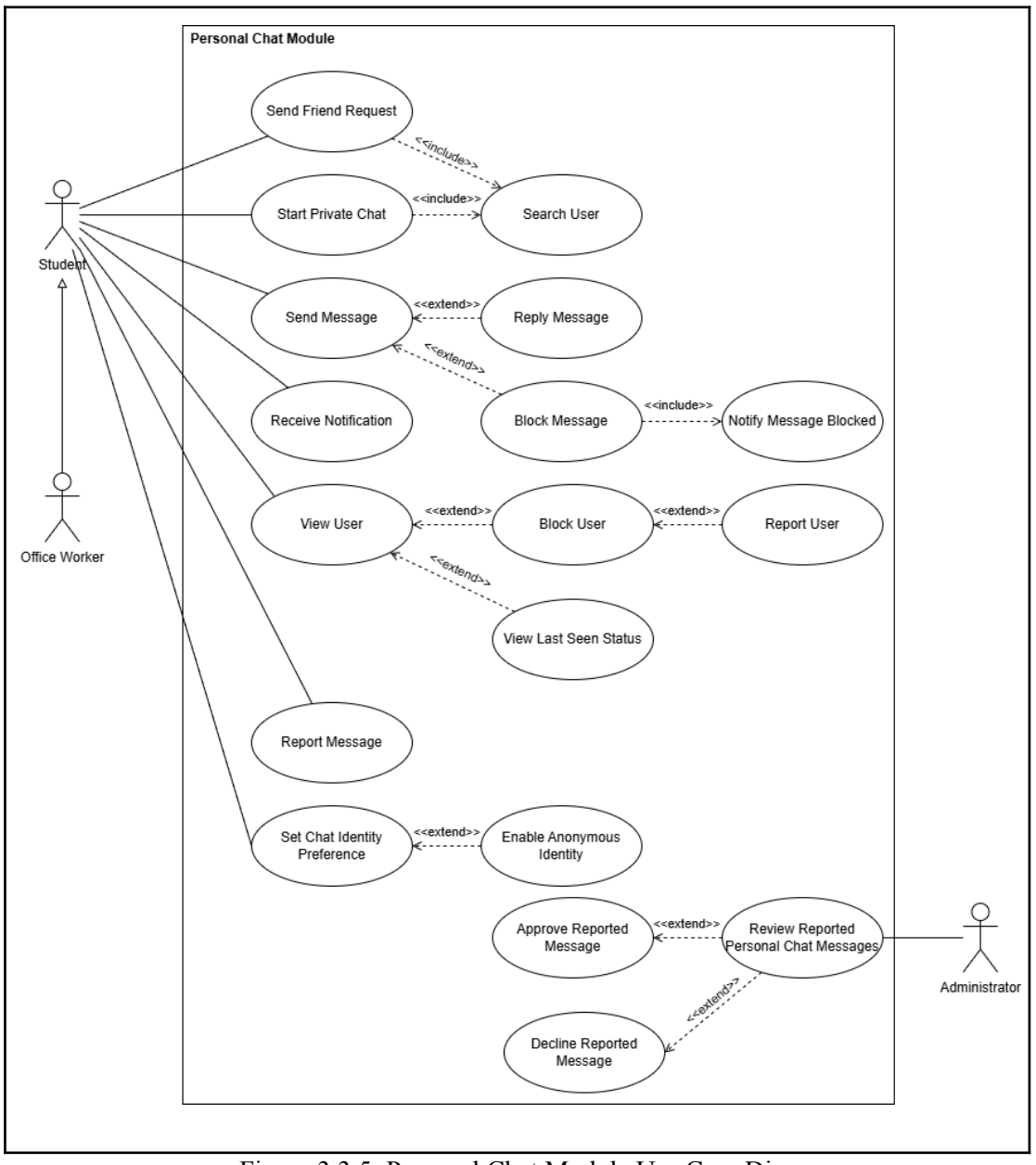


Figure 3.3.5: Personal Chat Module Use Case Diagram

Report Module

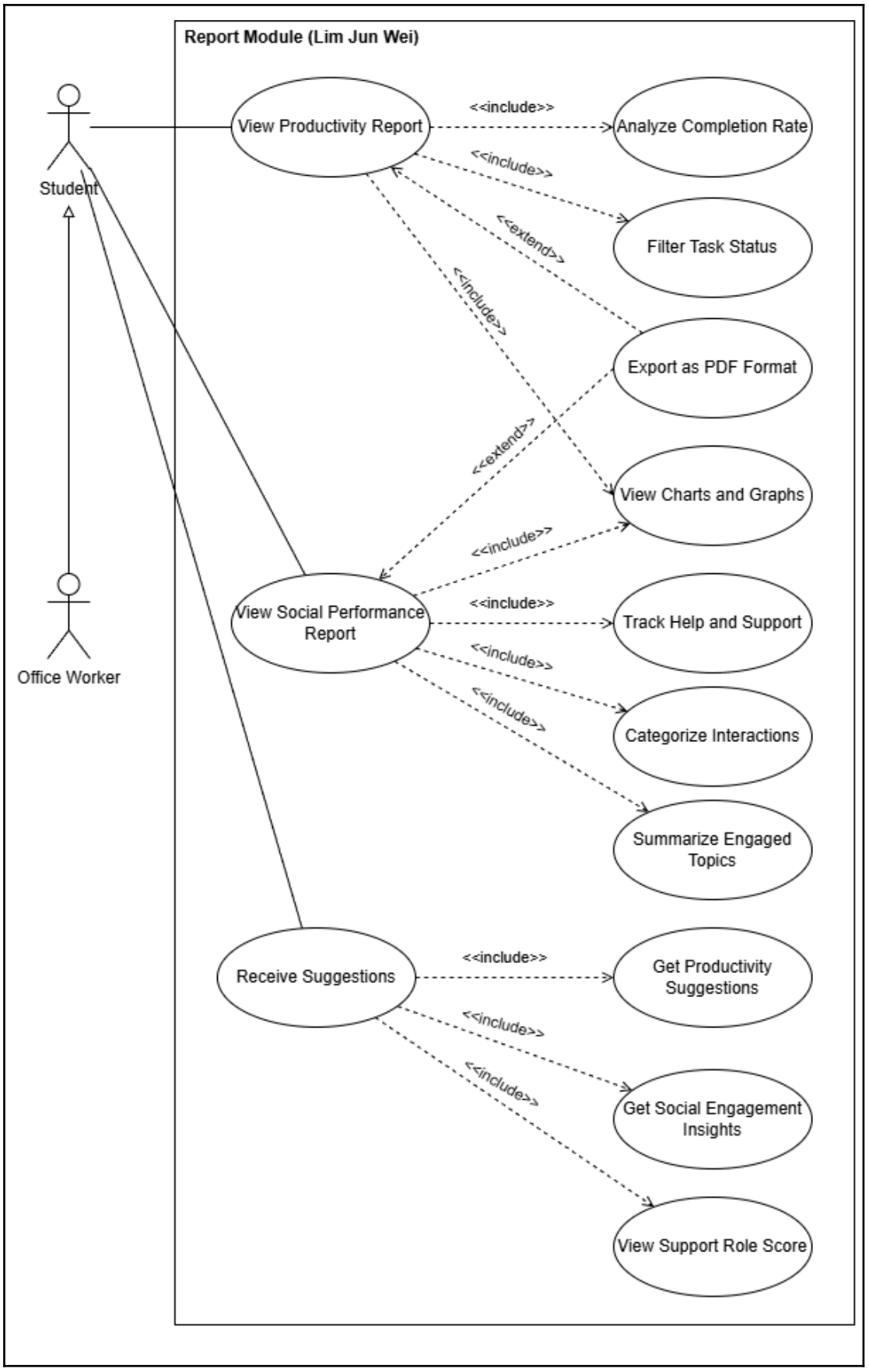


Figure 3.3.6: Report Module Use Case Diagram

3.3.2 Use Case Description

Goal Assistance Module

Create Task

Name of Use Case: Create Task	
Brief Description: The user can create a new task with subtasks by entering task details such as title, description, deadline, category and priority level. The system validates the input, stores the task and displays it in the task list.	
Actor: Student, office worker	
Precondition: The user must be logged into the system and the role is member.	
Actor Action	**System Response**
1. The user selects the "Task" tab.	2. The system displays the task list view.
3. The user selects the "Create Task" button from the task menu.	4. The system displays the task creation form.
5. The user enters task title, description, deadline, priority level, categories and task reminder occurrence via text input and voice input.	6. The system validates the input fields.
7. The user adds one or more subtasks with individual titles and due dates.	8. The system adds each subtask to the main task structure.
9. The user clicks the "Create" button.	10. The system saves the task along with any linked subtasks to the database.
	11. The system displays a confirmation message and shows the task in the task list.
Alternative Flow: A1. Step 6. If the user has left the task title field empty and attempts to save the task, the system will highlight the missing fields and display an error message "You must fill in the task title" which indicates that the required field must be filled. If the user has chosen an invalid date or deadline in the past, the system will display an error message "The date should not be before today's date." to prompt the user to correct	

the deadline. A3. Step 7. If the user chooses not to add any subtasks, the system will proceed to save the task without any subtasks.
Postcondition: A new task is successfully created and added to the user's task list. The user can directly view the new added task in the task list view.

Table 3.3.1: Goal Assistance Module - Create Task Use Case Description

Update Task

Name of Use Case: Update Task	
Brief Description: The user can modify the details of an existing task such as its title, description, deadline, category, priority levels and subtasks.	
Actor: Student, office worker	
Precondition: The user must be logged into the system and the role is member. The user must have created at least one task in the task list.	
Actor Action	**System Response**
1. The user selects the "Task" tab.	2. The system displays the task list view.
3. The user selects the existing task item from the task list.	4. The system displays the task details in editable fields.
5. The user modifies the task details such as title, description, due date and subtasks.	6. The system validates the modified input.
7. The user presses the "Save" button.	8. The system saves the updated task to the database.
	9. The system displays the updated task in the task list.
Alternative Flow: A1. Step 6. If the user has left the task title field empty and attempts to save the task, the system will highlight the missing fields and display an error message "You must fill in the task title" which indicates that the required field must be filled. If the user has chosen an invalid date or deadline in the past, the system will display an error message "The date should not be before today's date." to prompt the user to correct the deadline.	
Postcondition: A task is successfully updated and displayed to the user's task list. The user	

can directly view the updated task in the task list view.

Table 3.3.2: Goal Assistance Module - Update Task Use Case Description

Delete Task

Name of Use Case: Delete Task	
Brief Description: The user can remove an existing task and its associated subtasks if any from the task list.	
Actor: Student, office worker	
Precondition: The user must be logged into the system and the role is member. The user must have created at least one task in the task list.	
Actor Action	**System Response**
1. The user selects the "Task" tab.	2. The system displays the task list view.
3. The user long presses on the existing task item from the task list.	4. The system pops out an options bar.
5. The user selects the "Delete" option from the options bar.	6. The system prompts the user with a confirmation dialog "Are you sure to delete this task? The task will be removed along with its associated subtasks.".
7. The user selects the "Confirm" button in the confirmation dialog.	8. The system removes the task and associated data, then updates the task list view.
Alternative Flow: A1. Step 7. If the user selects on the "Cancel" button in the confirmation dialog, the system will retain the task and return back to the task list view.	
Postcondition: The selected task and its associated data are permanently deleted from the system.	

Table 3.3.3: Goal Assistance Module - Delete Task Use Case Description

Receive Goal Notifications

Name of Use Case: Receive Goal Notifications
Brief Description: The system sends reminder notifications to the user when a task's scheduled occurrence time is reached.

Actor: Student, office worker	
Precondition: The user must be logged into the system and the role is member. The user has already created a task and has set a reminder occurrence time.	
Actor Action	**System Response**
	1. The system periodically checks tasks for upcoming reminder times.
	2. The system detects a task with a scheduled reminder.
	3. The system pushes notification to the user about the task.
4. The user taps the task reminder notification.	5. The system navigates the user to the task detail interface.
Alternative Flow: A1. Step 4. If the user ignores the task reminder notification, the system will not navigate the user to the task detail interface.	
Postcondition: The user receives a task reminder and can access the task's detail view through the notification.	

Table 3.3.4: Goal Assistance Module - Receive Goal Notifications Use Case Description

Manage Smart Suggestions

Name of Use Case: Manage Smart Suggestion	
Brief Description: When a user creates or updates a task, the system will evaluate it. If optimization is needed, the system provides smart suggestions such as task rescheduling or task prioritization to improve scheduling efficiency and productivity. The user may choose to apply or ignore the suggestion.	
Actor: Student, office worker	
Precondition: The user must be logged into the system and the role is member. The user has created or updated a task and enabled the smart suggestion analysis.	
Actor Action	**System Response**
1. The user creates or updates a task.	2. The system displays successful operation dialog once the task creation or updation is successful.

	3. The system analyzes the task for optimization needs.
	4. If an optimization is needed, the system generates one or more smart suggestions such as suggest task reschedule and task prioritization
	5. The system displays a suggestion in the smart task advisor view.
6. The user taps on "Apply" on the suggestion pop-out.	7. The system applies the suggested changes, updates the task and collapses the suggestions.
	8. The system updates the task list view.
Alternative Flow: A1. Step 4. If there is no optimization needed, the system will not display the suggestion pop-up at the bottom of the task list view. A2. Step 5. If the user taps on the "Ignore" button on the suggestion pop-out, the system will only collapse the suggestions. The users can apply the suggestion in the future via expanding the collapsed suggestions at the bottom of the task list view if needed.	
Postcondition: The smart suggestion is either applied or ignored, its status is collapsed and hidden in the smart task advisor view. The task is updated accordingly if applied.	

Table 3.3.5: Goal Assistance Module - Manage Smart Suggestions Use Case Description

Time Usage Tracker Module

Track App Usage

Name of Use Case: Track App Usage	
Brief Description: The system can monitor and log the duration and frequency of application usage on the user's device for productivity analysis. The user can access the collected data in a dashboard with optional graph views and filtering by app category.	
Actor: Student, office worker	
Precondition: The user must be logged into the system and the role is member. The user has granted the required permissions for usage access. The time usage tracker module is enabled in system settings. Meanwhile, the application usage data is recorded and stored such as app name, duration and timestamps.	
Actor Action	**System Response**

	1. The system automatically starts tracking app usage in the background.
	2. The system collects the data such as app name, app launch time, total usage time and frequency.
	3. The system stores data securely in the database.
4. The user opens the time usage tracker dashboard.	5. The system displays the breakdown of app usage by date, duration and category.
	6. The system displays the usage statistics through visual elements such as charts and timeline.
	7. The system highlights the most used apps and total screen time.
Alternative Flow: A1. Step 4. If the user has selected the filtering options on app category and date range, the system will display the breakdown of app usage based on the selected app category and date range. A2. Step 5. If there is insufficient data gathered, the system will show an empty time usage tracker dashboard with the message "Insufficient app usage data are gathered right now.".	
Postcondition: The application usage data is recorded and stored so the user can view usage history and trends.	

Table 3.3.6: Time Usage Tracker Module - Track App Usage Use Case Description

Manage NFC Tag

Name of Use Case: Manage NFC Tag
Brief Description: The users can manage the interactions with NFC tags for task management purposes. They can write a task into an NFC tag, read an existing tag to retrieve task data or use NFC to clock in and out of a task. Each action will involve task lookup and tag validation.
Actor: Student, office worker
Precondition: The user must be logged into the system and the role is member. The user's device must support NFC and have it enabled. The user must have created valid tasks into the system database.

Actor Action	System Response
1. The user opens the NFC management interface in the app.	2. The system displays the NFC management view.
3. The user brings their device near an NFC tag.	4. The system detects the NFC tag and confirms readiness.
	5. The system displays the read NFC Tag information with action buttons such as Write Task and Clock In/Out.
6. The user selects an action button.	7. The system initiates the selected action and transitions to the corresponding extended use case.
Alternative Flow: A1. Step 4. If the system has failed to read the NFC tag, the system will display failed operation dialog and prompt users whether to try again the NFC tag reading process. A2. Step 6. If the user selects the action "write task", the system will navigate the user to a task selection list view. The user can select one of the existing tasks to write into the task. After selecting the task, the system will display the confirmation dialog to prompt user confirmation. After confirmation from the user, the system will ask the user to attach the NFC tag near the device to detect the NFC tag and write into the tag. If the writing process is successful, the system will display a success operation dialog. If not, the system will display the failed operation dialog and prompt the user whether to select "Try again". If the user selects the action "clock in" or "clock out", the system will ask the user to attach the NFC tag near the device to detect the NFC tag and update the clock in or out information into the tag. If the writing process is successful, the system will display a success operation dialog. If not, the system will display the failed operation dialog and prompt the user whether to select "Try again".	
Postcondition: Task information is successfully written to or read from or associated with an NFC tag. The system provides feedback on the success or failure of each NFC interaction.	

Table 3.3.7: Time Usage Tracker Module - Manage NFC Tag Use Case Description

View Task Summaries

Name of Use Case: View Task Summaries
Brief Description: The user can view a summary of tasks completed and time spent, broken down into daily and weekly views. The user can filter, analyze patterns, export summaries and view historical trends. It extends several supporting use cases for filtering,

highlighting inconsistencies, suggesting improvements and exporting reports.	
Actor: Student, office worker	
Precondition: The user must be logged into the system and the role is member. The user has previously completed or logged tasks and app usage data.	
Actor Action	**System Response**
1. The user opens the Task Summary from the Time Usage Tracker interface.	2. The system navigates the user to the task summary view.
	3. The system loads the daily and weekly task summaries.
4. The user selects the summary type (daily or weekly).	5. The system displays the summaries of tasks completed and time spent per task.
6. The user applies the filter on task categories, priority and date range.	7. The system displays the filtered summary data with charts and visual breakdowns.
	8. The system displays the tips or suggestions based on usage patterns
9. The user selects the "Export" button.	10. The system exports the current displaying report into a PDF file and displays the PDF view to the user.
Alternative Flow: A1. Step 7. If there are no tasks available in the selected filter range, the system will display a message "No tasks match your selected filters. Try adjusting the filter options.".	
Postcondition: Task summaries are displayed with visual breakdowns, potential anomalies and suggestions. The optional export into PDF file is available.	

Table 3.3.8: Time Usage Tracker Module - View Task Summaries Use Case Description

Set App Category Time Limitation

Name of Use Case: Set App Category Time Limitation
Brief Description: The user can define the time limits or productivity goals for a specific app category, the system allows the user to choose which app category to apply the limits on.

Actor: Student, office worker	
Precondition: The user must be logged into the system and the role is member. Meanwhile, the app usage data has been categorized.	
Actor Action	**System Response**
1. The user selects the Time Usage Tracker interface.	2. The system navigates the users to the Time Usage Tracker view.
3. The user selects the setting button at the top right corner of the view.	4. The system displays the app category time limitation rule list.
5. The user selects the "Add new rule" button.	6. The system displays the create rule form view with available app category and time limitation per day.
7. The user chooses the one specific app category and selects the time limitation per day.	
8. The user selects the "Save" button.	9. The system validates the create rule form input.
	10. The system displays the successful operation dialog and updates the app category time limitation rule list.
11. The user selects "Create Whitelisted App".	12. The system displays the whitelisted app creation list that can be exempted from time limitations.
13. The user selects one or more apps to whitelist and confirms.	14. The system links it to the chosen app category.
	15. The system displays a successful operation dialog and updates the app category time limitation rule list.
Alternative Flow: A1. Step 4. If the app usage data has not been categorized, the system will display the empty app category time limitation rule list with disabled "Add new rule" button. A2. Step 9. If the user has left either app category or option time limitation per day option empty, the system will highlight the required field to select and display the error message to ask the user to select an option.	

A3. Step 12. If no apps can be whitelisted, the system displays an empty whitelisting list and informs the user accordingly. A4. Step 13. The user may dismiss the form, then the system returns to Step 4 without adding any whitelisted apps.
Postcondition: A time limit is set for the selected app category. Any whitelisted apps are linked to the rule and remain exempt from the limit. The notifications will be triggered if the non-whitelisted apps exceed the time limit..

Table 3.3.9: Time Usage Tracker Module - Set App Category Time Limitation Use Case Description

Receive Time Limit Notification

Name of Use Case: Receive Time Limit Notification	
Brief Description: The system will notify the user when their app usage exceeds the configured time limit. The user can interact with the notification to view the details in the time usage tracker dashboard.	
Actor: Student, office worker	
Precondition: The user must be logged into the system and the role is member. The user has also set the time limits for specific app categories.	
Actor Action	**System Response**
1. The user uses apps normally.	2. The system tracks the usage duration in the background.
	3. The system detects the time limit is exceeded for a specific app category.
	4. The system pushes a notification to the user.
5. The user taps the notification.	6. The system redirects the user to the usage dashboard highlighting the app category that hit the limit.
Alternative Flow: A1. Step 5. If the user ignores the notification, no redirection will happen.	
Postcondition: The user is informed about the time limit breach, he or she can view the details of the application category that exceeds the defined time limitation.	

Table 3.3.10: Time Usage Tracker Module - Receive Time Limit Notification Use Case Description

Anonymous Community Module

Manage Community

Name of Use Case: Manage Community	
Brief Description: The user can manage their interactions with anonymous topic-based communities, modifying existing community details, joining topic-based groups and subscribing to specific community.	
Actor: Student, office worker	
Precondition: The user must be logged into the system and the role is member.	
Actor Action	**System Response**
1. The user taps on the “Community” tab.	2. The system navigates the user to the Community interface.
	3. The system displays a list of available communities and user options.
4. The user selects to create, modify, join or subscribe to a community topic.	5. The system performs the corresponding action and updates the community data accordingly.

Alternative Flow:

A1. Step 4. If the user taps on the "Create Community" button at the top right corner of the Community interface, the system will navigate the user to the community registration form page. The user fills in the community details such as community title, description, profile image and maximum allowed members. The user will click on the "Submit" button. The system evaluates the community registration form field input. If all the required input fields are filled correctly, the system will add the community registration record into the database for enabling the administrator to review before being approved. If not, the system will highlight the required form field and display the error messages for correction. After successful submission, the system will display the successful operation dialog.

If the user taps on the "My Community" button and selects the "Edit" button on the specific created community, the system will display the community detail editable form field. The user modifies the community details such as community name, community description or profile images. The user will tap on the "Save" button. The system evaluates the updated fields. If all the input fields are updated correctly, the system will update the community details and update the community list. the system will highlight the required form field and display the error messages for correction. After successfully updating, the system will display the successful operation dialog.

If the user taps on the "Join" button on the specific community in the community list, the system will update the user's joined communities record into the database. The system will

navigate the user into the joined community view. If the user taps on the "Subscribe" button within the joined community view, the system will update the user's community subscription record into the database. The system will push the notifications to the user if there are any new posts in the community in future.
Postcondition: The user has created a new community, modified an existing community, joined a topic community or subscribed to receive the updates from specific topics.

Table 3.3.11: Anonymous Community Module - Manage Community Use Case Description

Manage Community Posts

Name of Use Case: Manage Community Posts	
Brief Description: The user can manage their engagement with posts in anonymous topic-based communities. This includes creating and tagging posts, commenting on others' posts, searching for community posts, flagging inappropriate content and reacting with emojis or likes.	
Actor: Student, office worker	
Precondition: The user must be logged into the system and be a member of at least one joined community.	
Actor Action	**System Response**
1. The user taps on the specific joined community from the community list view.	2. The system navigates the user into the joined community view.
3. The user selects to perform a post-related action such as create, comment, search, react or flag.	4. The system navigates to the appropriate interface or performs the selected action.
Alternative Flow: A1. Step 3. If the user selects the "New Post" button, the system navigates the user to the post creation form with fields for title, description, optional image and tags. The user fills in the post content and selects relevant tags. The user taps the "Post" button. The system validates all the required fields. If all required fields are valid and no illegal terms in the content, the system saves the post to the database and updates the post feed. The system displays the successful operation dialog to the user. If the validation fails, the system highlights missing or problematic fields and shows the error messages. If the user taps on a post and selects "Comment", the system displays the comment input field. The user enters their comment with optional images and taps the "Submit" button. The system validates the input. If the required field is filled in correctly without illegal terms, the system adds the comment below the post. The system will show the successful	

operation dialog. If the user taps on the "Search" icon within the community, the system displays the search bar with filters such as title, tags and date. The user enters search criteria and taps "Search". The system displays the matching posts. If there is no post based on the search filter, the system displays "No matching posts". A2. Step 4. The user taps on the "Flag" icon under a post. The system displays a list of flagging reasons such as Hate Speech, Harassment, Spam and Misinformation. The user selects a reason and taps the "Submit Flag" button. The system records the report and notifies the admin for review. The system displays the successful operation dialog to the user. If the user taps the "React" icon under a post, the system displays the available emoticons or like buttons. The user selects a reaction. The system updates the reaction and reflects it in the UI. If the user reacts again, the previous reaction is updated or removed accordingly.
Postcondition: The user has successfully created and tagged a new community post, commented on one or more posts, searched for relevant community content, flagged a post for moderation with a selected reason or reacted to posts using likes or emojis.

Table 3.3.12: Anonymous Community Module - Manage Community Posts Use Case Description

Manage Forum

Name of Use Case: Manage Forum	
Brief Description: The user can interact with the question-and-answer forum by posting questions, answering others' queries, searching existing forum discussions, voting on answers and receiving notifications about the forum activity. He or she can also mark the most helpful answer to their own questions to guide others.	
Actor: Student, office worker	
Precondition: The user must be logged into the system with a Member role and have access to the forum feature.	
Actor Action	**System Response**
1. The user taps on the "Q&A Forum" tab.	2. The system navigates the user to the Q&A Forum view.
	3. The system displays a list of recent or popular forum questions.
4. The user selects an action to perform such as posting a question, searching, answering and upvoting or downvoting answers.	5. The system opens the corresponding interface and performs the selected action.

Alternative Flow: A1. Step 4. If the user taps on the "Ask a Question" button in the Q&A Forum view, the system displays a form with fields for title, detail description and optional tags or images. The user fills up all the fields and taps on the "Submit" button. The system validates whether all the required form fields are filled up correctly. If all the required form fields are filled up correctly, the system will save the question and update the forum list. The system displays the successful operation dialog. If there are empty required form fields or illegal terms in the content, the system will highlight the problematic form fields and display the error messages. If the user taps on the "Search" button in the Q&A Forum view, the system will display a search bar with filter options such as tags, popularity and date. The user enters the keywords and selects the filters, then taps on the "Search" icon button. The system returns a list of matching questions and answers. If there are no matching questions, the system returns the message "No matching questions found.". If the user taps on a question and selects "Answer", the system provides a text input area with optional images or links. The user fills up all the required input fields and taps on the "Submit" button. The system validates whether the required fields are filled up and the content does not include illegal terms. If the answer has fulfilled all the validations, the system posts an answer under the question. The system displays a successful operation dialog. If not, the system highlights the problematic form fields and displays the error messages. If the user taps on a question and scrolls down the question view, the system displays a list of answers regarding the selected question. The user taps on the upvote or downvote icon under a specific answer. The system records the vote and updates the total score. If the user changes his or her vote later, the system updates the result accordingly and highlights the most helpful answer based on the score. The system automatically sends notifications to the user when someone has answered his or her questions or reacts to his or her answers via vote or comment. The user can tap on the forum notification. The system navigates the user to the question answering view with upvote and downvote score and comments replying to the message.
Postcondition: The user has successfully posted a question, searched or browsed forum threads, submitted or interacted with answers and received relevant notifications based on forum engagement.

Table 3.3.13: Anonymous Community Module - Manage Forum Use Case Description

Review Flagged Post

Name of Use Case: Review Flagged Post
Brief Description: The administrator reviews posts flagged by users in the anonymous community. After evaluation, the admin may decide to take no action or drop flagged post if the content violates guidelines.

Actor: Administrator	
Precondition: The administrator must be logged into the system with administrator role and there are posts flagged for review in the system.	
Actor Action	**System Response**
1. The administrator opens the moderation dashboard.	2. The system displays a list of flagged posts with details such as reason, reporter and timestamp.
3. The administrator selects a flagged post to review.	4. The system displays the full content of flagged posts.
5. The administrator reviews the content.	
6. The administrator selects the "Drop Post" button.	7. The system performs the action to drop the flagged post and notifies the post author.
Alternative Flow: A1. Step 6. If the administrator did not find the violation in the post, he or she will dismiss the flag. The system records the review action and updates the post's status as "Cleared".	
Postcondition: The flagged post has been reviewed and either cleared or removed based on content policy evaluation.	

Table 3.3.14: Anonymous Community Module - Review Flagged Post Use Case Description

Review Flagged Forum Question

Name of Use Case: Review Flagged Forum Question	
Brief Description: The administrator can review flagged forum questions to determine whether content violated community guidelines, contains inappropriate material, or requires moderation actions such as hiding the question, issuing warnings, or removing the content.	
Actor: Administrator	
Precondition: The administrator must be logged into the system with administrator role and there must be at least one forum question has been flagged by users	
Actor Action	**System Response**
8. The administrator opens the admin panel.	9. The system displays a list of flagged forum questions with

	details such as reason, reporter and timestamp.
10. The administrator selects a flagged forum question to review.	11. The system displays the full content of flagged posts.
12. The administrator reviews the content.	
13. The administrator selects the "Drop Post" button.	14. The system performs the action to drop the flagged post and notifies the post author.
Alternative Flow: A1. Step 6. If the administrator did not find the violation in the forum questions, he or she will dismiss the flag. The system records the review action and updates the post's status as "Cleared".	
Postcondition: The flagged forum question has been reviewed and either cleared or removed based on content policy evaluation.	

Table 3.3.15: Anonymous Community Module - Review Flagged Forum Question Use Case Description

Manage Anonymous Group Chat

Name of Use Case: Manage Anonymous Group Chat	
Brief Description: The user can manage anonymous group chat sessions, including creating new group chats with friends, sending and replying to messages, sending attachments such as images, tasks, community posts and emotion diary entries, receiving chat notifications and leaving the chat when needed.	
Actor: Student, office worker	
Precondition: The user must be logged into the system with a Member role.	
Actor Action	**System Response**
1. The user taps on the "Community" tab.	2. The system navigates the user to the Community view.
	3. The system displays a list of available communities and user options.
4. The user taps on the "Anonymous Group" button.	5. The system navigates the user to the Anonymous Group view.

	6. The system displays a list of joined and available group chats.
7. The user selects one of the actions such as creating, sending or managing a group chat.	8. The system performs the corresponding action and updates the group chat data accordingly.

Alternative Flow:

A1. Step 7. If the user taps on "New Group Chat", the system displays a new group chat creation input form with the various fields such as group chat title, group chat profile image, group chat description and member options. The user fills up all the required fields and selects the members to invite. The user taps on the "Create" button. The system validates whether all the required form inputs are filled correctly. If yes, the system saves the new group chat into the database, updates the list of group chats and navigates users into the newly created group chat room. If not, the system highlights the problematic input fields and displays the error messages.

If the user taps on the "Modify" button on the owned group chat from the group chat list, the system will display the group chat details modification input form. The user modifies the existing group chat details and taps on "Save" button. The system validates whether all the required form inputs are filled correctly. If yes, the system updates the group chat details in the group chat list. If not, the system highlights the problematic input fields and displays the error messages.

If the user taps on "Join" on the available group chat (pending invitation), the system adds the user into the group chat. The system navigates the user to the joined group chat room.

If the user taps on the specific group chat, the system navigates the user into the group chat room. The system displays the chat history. The user types a message or replies to a specific previous message and taps on the "Send" button. The system sends the message, displays it in the chat and delivers it to all group members in real-time.

If the user taps on "Leave Chat" button in the specific group chat setting, the system prompts for confirmation. If the user taps on "Yes", the system removes the user from the group and stops future messages and notifications. If the user taps on "No", the system hides the confirmation pop-out dialog only.

A2. Step 8. When new messages are received in joined chats, the system sends push notifications to the user. The user can tap on the notification. The system navigates the user to the group chat room view and displays the chat history.

Inside the group chat room, the user taps the Attachment "+" button. The system displays an attachment menu with the options such as send image, send task, send community post and send emotion diary entry. The user selects one of the attachment types.

Postcondition: The user has successfully created or managed an anonymous group chat, including sending or replying to messages, receiving relevant notifications or leaving the chat group.

Table 3.3.16: Anonymous Community Module - Manage Anonymous Group Chat Use Case

Description

Personal Chat Module

Send Friend Request

Name of Use Case: Send Friend Request	
Brief Description: The user can send a friend request to another user. Before sending the request, the system allows the user to search for specific users by name, username or other identifiers. Once a user is found, the requester may view the profile summary and send the friend request.	
Actor: Student, office worker	
Precondition: The user must be logged into the system as a Member role and the target user has not already been added as a friend.	
Actor Action	**System Response**
1. The user taps on the "Personal Chat" tab.	2. The system displays the list of recent personal chats, search bar and add friend options.
3. The user taps on the "add friend" button.	4. The system queries the database and returns the list of matching users.
5. The user selects a target user from the search results.	6. The system displays the friend request identity options such as "Anonymous identity" and "Use real identity"..
7. The user selects the identity option.	8. The system sends the friend request and updates the friend request state for both users.
	9. The system shows a confirmation message indicating the friend request was sent successfully.
Alternative Flow: A1. Step 4. If the target user is already a friend, the system prevents sending duplicate requests and displays an appropriate message. If the user has already sent a pending request to the same target user, the system informs the user that the request is still awaiting approval. If no users match the search keyword, the system displays "No users found". A2. Step 7. If the user has chosen the "Use anonymous identity" option, the system will	

send the anonymous friend request to the targeted user with anonymous name and hidden avatar image. If the user has chosen the "Use real identity" option, the system will send the real friend request to the targeted user showing the real name and avatar image.
Postcondition: A friend request is successfully sent and stored in the system. The target user will see the request and may choose to accept or reject it.

Table 3.3.17: Personal Chat Module - Send Friend Request Use Case Description

Start Private Chat

Name of Use Case: Start Private Chat	
Brief Description: The user can initiate a private one-on-one chat by first searching for another user and then starting a chat session.	
Actor: Student, office worker	
Precondition: The user must be logged into the system as a Member role and own an existing friend.	
Actor Action	**System Response**
10. The user taps on the "Personal Chat" tab.	11. The system displays the list of recent personal chats and a search bar.
12. The user enters a name or keyword into the search bar.	13. The system queries the database and returns the list of matching friends.
14. The user selects a user from the search results.	15. The system checks for an existing chat session.
	16. The system navigates the user to the private chat interface.
	17. The system displays the chat history.
Alternative Flow: A1. Step 4. If there is no matching friend found, the system displays the message "No matching user found". A2. Step 8. If the chat session does not exist, the system creates a new private chat session without chat history.	
Postcondition: The user has either successfully initiated a new private chat session or continued an existing one with another user.	

Table 3.3.18: Personal Chat Module - Start Private Chat Use Case Description

Send Message

<table>
<tr><td colspan="2">Name of Use Case: Send Message</td></tr>
<tr><td colspan="2">Brief Description: The user can send a message in a private chat which includes replying to specific messages and handling blocked message scenarios.</td></tr>
<tr><td colspan="2">Actor: Student, office worker</td></tr>
<tr><td colspan="2">Precondition: The user must be logged into a chat session and not blocked by the recipient in private chat.</td></tr>
<tr><th>Actor Action</th><th>System Response</th></tr>
<tr><td>1. The user selects a user in the private chat list view.</td><td>2. The system navigates the user to the private chat session.</td></tr>
<tr><td></td><td>3. The system displays the chat interface and previous messages.</td></tr>
<tr><td>4. The user selects the "Reply" option on the message he or she wants to reply to.</td><td>5. The system adds the replying message stack above the message text input field.</td></tr>
<tr><td>6. The user types a message into the input field.</td><td></td></tr>
<tr><td>7. The user taps on the "Send" button.</td><td>8. The system validates the message.</td></tr>
<tr><td></td><td>9. The system sends the message to the intended recipient.</td></tr>
<tr><td></td><td>10. The system updates the chat history shown in the chat interface.</td></tr>
<tr><td></td><td>11. The system notifies the recipient regarding the new message sent.</td></tr>
<tr><td colspan="2">Alternative Flow:

A1. Step 3. If there is no previous message in the private chat session, the system will display the empty private chat interface.

A2. Step 4. If the user does not select the "Reply" option on the message, the system will not add the reply message stack above the message text input field.

A3. Step 8. If there is invalid or illegal message, the system will display the error message and prevent the message from being sent.</td></tr>
<tr><td colspan="2">Postcondition: The message is sent, displayed in the chat interface or blocked with proper error message if it violates the rules.</td></tr>
</table>

Table 3.3.19: Personal Chat Module - Send Message Use Case Description

Receive Notification

Name of Use Case: Receive Notification	
Brief Description: The system will deliver the real-time notifications to the users when new messages are received in personal chats. The notification helps users to stay informed even when they are not actively viewing the chat.	
Actor: Student, office worker	
Precondition: The user must be logged into the system as a Member role and have personal chat notifications enabled in their settings.	
Actor Action	**System Response**
1. The user is logged in and the system is running in the foreground or background.	2. The system listens for new messages in real-time.
3. Another user sends a message to this user in a personal chat.	4. The system detects the new message event and checks notification settings.
	5. The system pushes the notification to the user.
6. The user receives a notification banner.	7. The system displays the sender's display name and part of the message content in the notification.
8. The user taps on the notification.	9. The system navigates the user to the private chat interface for the respective sender.
Alternative Flow: A1. Step 5. When the recipient is offline, the system will queue the notification on the server. Once the user comes back online, the system pushes the missed message notifications.	
Postcondition: The user is notified of the new personal chat message in real-time or upon coming back online.	

Table 3.3.20: Personal Chat Module - Receive Notification Use Case Description

View User

Name of Use Case: View User

Brief Description: The user is allowed to view another user's profile and may choose to block them, report them or view their last seen status based on privacy permissions.	
Actor: Student, office worker	
Precondition: The user must be logged into the system as a Member role.	
Actor Action	**System Response**
1. The user taps on another user's profile.	2. The system displays the user's profile which includes username, profile image and last seen status with user options.
3. The user views profile details and available options.	
4. The user chooses an action such as blocking the user, reporting them or checking last seen status.	5. The system performs the selected action by the user.
Alternative Flow: A1. Step 2. If the other user has disabled the visibility of last seen status, the system will hide the last seen timestamp. A2. Step 4. If the user taps on the "Block" button, the system prompts for confirmation. If the user taps on the "Yes" button, the system updates the block list and disables the communication with that user. The system displays the successful operation dialog for confirming block action. If the user taps on the "No" button, the system will hide the confirmation dialog. If the user chooses to report while blocking, the system displays a list of reasons such as harassment, spam and offensive content. The user selects a reason and submits the report. The system stores the report and notifies administrators for review.	
Postcondition: The user has successfully viewed the user profile and optionally blocked, reported or viewed the last seen information.	

Table 3.3.21: Personal Chat Module - View User Use Case Description

Set Chat Identity Preference

Name of Use Case: Set Chat Identity Preference
Brief Description: The user can configure their preferred identity mode whether to use their real user name or anonymous for future one-to-one chat. This includes the option for enabling the anonymous identity feature.
Actor: Student, office worker

Precondition: The user must log into the system with a Member role and navigate to specific user chat settings.	
Actor Action	**System Response**
1. The user taps on "Chat Settings" within the other user's profile view.	2. The system displays other user's profile information with chat identity options.
3. The user selects his or her preferred identity mode as Anonymous	4. The system activates the selected mode and updates the user's preference in the database.
	5. The system hides the username and profile image of the user and replaces it with an Anonymous random name and image.
	6. The system will restrict the specific selected user to view the anonymous name and profile image only instead of the real username.
Alternative Flow: A1. Step 3. If the user selects his or her preferred identity mode as Using Real Username, the system will expose the real username and profile image to the specific selected user.	
Postcondition: The user's chat identity preference is updated and the selected user can view the anonymous or real name or profile image based on selected preference.	

Table 3.3.22: Personal Chat Module - Set Chat Identity Preference Use Case Description

Review Reported Personal Chat Messages

Name of Use Case: Review Reported Personal Chat Messages	
Brief Description: The administrator reviews personal chat messages that have been reported by users for inappropriate, harmful or abusive content. After assessment, the administrator may choose to approve (confirm the violation and take moderation action) or decline (dismiss the report).	
Actor: Administrator	
Precondition: The administrator must be logged into the system with administrator role and there must be at least one personal chat message that has been reported by users.	
Actor Action	**System Response**

1. Administrator opens the admin panel.	2. The system displays the dashboard with a list of reported personal chat messages.
3. The administrator selects a reported message to review.	4. The system displays the message details including sender, receiver, timestamp, message content, attached media (if any), and reporter comments.
5. The administrator assesses the message for violations.	6. The system presents moderation options: Approve (take action) or Decline (dismiss report).
7. The administrator selects a moderation option.	1. The system updates the admin review log and pushes notification to the creator of the reported messages.
Alternative Flow: A1. Step 6. If the administrator determines there is no violation, the administrator selects Decline, and the system dismisses the report and records it for audit purposes.	
Postcondition: The reported personal chat message is reviewed, and the system either logs a moderation action (if approved) or dismisses the report (if declined).	

Table 3.3.23: Personal Chat Module - Review Reported Personal Chat Message Use Case Description

Report Module

View Productivity Report

Name of Use Case: View Productivity Report	
Brief Description: The user can view a detailed report summarizing their task performance, including completed, ongoing and missed tasks. The report provides options to analyze task completion rate, filter by status, export as PDF and visualize data through charts and graphs.	
Actor: Student, office worker	
Precondition: The user must be logged into the system with a Member role and must have task activity history.	
Actor Action	**System Response**

1. The user taps on the "Reports" tab and selects "Productivity Report".	2. The system loads and displays the productivity report interface with task data.
3. The user applied filters such as date range, task category and status.	4. The system filters and updates the report view accordingly.
	5. The system automatically calculates and displays the task completion rate and performance trend.
	6. The system shows key metrics like percentage of tasks completed, missed or ongoing.
7. The user views the visual representation of the statistics.	8. The system displays charts and graphs for better understanding.
Alternative Flow: A1. Step 7. If the user taps on the "Export as PDF" button, the system formats the current view into a PDF document. The user is prompted to save or download the file.	
Postcondition: The user has viewed their filtered productivity report, analyzed their performance and optionally exported the data for personal tracking.	

Table 3.3.24: Report Module - View Productivity Report Use Case Description

View Social Performance Report

Name of Use Case: View Social Performance Report	
Brief Description: The user can view an automatically generated report that summarizes their social engagement activities, including support-seeking and support-giving behavior, types of interactions and active topic participation. The report presents the data visually using charts and graphs, and can be exported as PDF.	
Actor: Student, office worker	
Precondition: The user must be logged into the system as a Member role and have prior social interaction history.	
Actor Action	**System Response**
1. The user taps on the "Reports" tab and selects "Social Performance Report".	2. The system loads the social interaction data from the database and displays the social performance report.

	3. The system analyzes completion rate and support activity.
	4. The system categorizes interaction types.
	5. The system identifies most engaged communities and topics.
	6. The system displays all insights using charts and graphs.
7. The user views the report.	
Alternative Flow: A1. Step 7. If the user taps on the "Export as PDF" button, the system formats the current view into a PDF document. The user is prompted to save or download the file.	
Postcondition: The user has successfully viewed a complete social performance summary and optionally exported the report.	

Table 3.3.25: Report Module - View Social Performance Report Use Case Description

Receive Suggestions

Name of Use Case: Receive Suggestions	
Brief Description: The system automatically provides personalized suggestions to help the user improve productivity, increase social engagement and better understand their support role performance. These insights are generated based on user behavior and activity patterns.	
Actor: Student, office worker	
Precondition: The user must be logged into the system as a Member role and have sufficient activity data recorded in the system.	
Actor Action	**System Response**
1. The user taps on the "Suggestions" section within the "Reports" tab.	2. The system analyzes task and time management behavior.
	3. The system reviews the user's community and chat engagement.
	4. The system evaluates user's supportive interactions.

	5. The system presents the suggestions and support score in an interactive and readable format.
6. The user views a list of tailored suggestions and scores.	
Alternative Flow: A1. Step 2. If the user has insufficient activity data, the system displays a message which shows that the suggestions cannot yet be generated and encourages further usage of features.	
Postcondition: The user has reviewed system-generated suggestions for productivity, social engagement and support performance.	

Table 3.3.26: Report Module - Receive Suggestions Use Case Description

3.3.3 Functional Requirements

1.0 Goal Assistance Module (Shared)

1.1 Task Management

1.1.1 The system shall allow users to create, edit, delete, and mark tasks as completed.

1.1.2 The system shall support optional fields in task creation, including description, link, image attachment, and sub-tasks.

1.1.3 The system shall allow users to classify tasks into categories (Personal, Exercise, Entertainment, Study, Working).

1.1.4 The system shall allow users to set task priority levels from P1 (lowest) to P5 (default, most important).

1.1.5 The system shall allow users to specify task timing: start date time and end date time.

1.1.6 The system shall allow enabling push notifications as reminders, triggered at the occurrence time or set intervals before (e.g., 10 minutes before).

1.1.7 The system shall support progress tracking based on sub-task completion (displayed as a percentage).

1.1.8 The system shall allow users to sort and filter tasks by category, priority, type, and due date.

1.1.9 The system shall support a calendar view and list view for task visualization.

1.2 Smart Input & Suggestions

1.2.1 The system shall extract task goals from natural language inputs using NLP (e.g., "Remind me to submit a project report tomorrow at 3 PM").

1.2.2 The system shall auto-suggest suitable time slots based on availability, habits, and workload which allows the users to decide whether to apply the reschedule.

1.2.3 The system shall recommend task categories and priority based on extracted task information from speech input.

1.2.4 The system shall provide predefined templates for recurring task types (e.g., study, exercise, work).

1.3 Smart Advisor

1.3.1 The system shall analyze users' past task completion habits to suggest priority adjustments (e.g., start earlier, complete before usual fatigue hours).

1.3.2 The system shall detect schedule conflicts and suggest alternative time slots dynamically.

1.3.3 The system shall provide emotional and productivity-based reminders, such as taking breaks or relaxation suggestions.

1.3.4 The system shall learn long-term behavior trends to optimize future task planning (e.g., avoid late-night tasks, balance workload).

1.3.5 The system shall allow users to choose whether to apply or ignore the smart advisor suggestions.

2.0 Time Usage Tracker Module

2.1 App Activity Tracker

2.1.1 The system shall continuously monitor the screen time usage per app once permitted by the user.

2.1.2 The system shall collect the data when users start and stop using the apps.

2.1.3 The system shall analyze the app usage duration based on the collected data of time spending on different apps.

2.1.4 The system shall categorize apps into productivity, study, entertainment and others.

2.1.5 The system shall display a dashboard which generates visual graphs of app activity with time spending based on users' daily and weekly app usage.

2.1.6 The system shall sort the app usage based on the categories, time spent on apps and apps' name.

2.2 NFC Task Tracker

2.2.1 The system shall allow users to write and read the task info into and from the NFC tag.

2.2.2 The system shall detect NFC tag scans to clock in or out of specific tasks.

2.2.3 The system shall update the existing task progress based on the NFC tag scanned.

2.2.4 The system shall link NFC sessions to goals and reports.

2.2.5 The system shall notify users if an NFC tag is invalid or not linked to any task.

2.3 Daily and Weekly Summary

2.3.1 The system shall display the summaries of tasks completed and time spent on each task.

2.3.2 The system shall allow users to filter the task summaries based on category, priority and time created.

2.3.3 The system shall generate a visual breakdown of daily and weekly app activity in categories.

2.3.4 The system shall highlight the inconsistencies and unusual time usage patterns.

2.3.5 The system shall provide appropriate suggestions on app usage based on inconsistent time usage patterns detected.

2.3.6 The system shall allow users to export the daily or weekly summaries as PDF files.

2.3.7 The system shall allow users to view historical summaries based on selected date ranges.

2.4 Custom Time Goals or Alerts

2.4.1 The system shall allow users to define time limits or productivity goals for specific app categories.

2.4.2 The system shall send notifications when a time goal is exceeded or if a break is needed.

2.4.3 The system shall provide suggestions for scheduling and task balancing when irregular patterns are detected.

3.0 Anonymous Community Module

3.1 Topic-Based Community

3.1.1 The system shall allow users to join anonymous communities based on shared topics of interest.

3.1.2 The system shall allow users to create posts without revealing their real identities.

3.1.3 The system shall allow users to edit the post title, tag, description and attachments before posting into the community.

3.1.4 The system shall allow users to subscribe to various communities to receive updates and engage with the content.

3.1.5 The system shall allow users to report the communities for moderation review.

3.1.6 The system shall allow users to search for posts based on the post title and date range.

3.2 Q&A Forum

3.2.1 The system shall allow users to post questions anonymously.

3.2.2 The system shall allow users to answer questions posted in the forum via text and images.

3.2.3 The system shall allow users to mark answers with upvote or downvote.

3.2.4 The system shall highlight the most helpful answer for each question.

3.2.5 The system shall categorize the posts with tags to make them searchable and easily organized.

3.2.6 The system shall allow users to search forum posts based on post title, labeled tags and created date.

3.2.7 The system shall allow users to sort answers based on the amount of upvotes and the created date.

3.3 Group Chat

3.3.1 The system shall allow users to engage in real-time group chats using anonymous display names.

3.3.2 The system shall allow users to send messages in text and emoji forms.

3.3.3 The system shall allow users to insert image attachments when sending messages.

3.3.4 The system shall send notifications to the joined users when there are new messages in joined group chat.

3.3.5 The system shall allow the group chat creator to remove the group chat members.

3.3.6 The system shall automatically block the message sending when there are illegal terms in the text messages.

3.4 Post Reaction and Flagging

3.4.1 The system shall allow users to react to posts with likes or emoticons.

3.4.2 The system shall allow users to remove or update the previous reaction to the posts.

3.4.3 The system shall automatically block or drop the posts when there are illegal terms in the posts content.

3.4.4 The system shall allow users to manually flag the inappropriate or harmful posts for administrator review.

3.4.5 The system shall block or drop the inappropriate or harmful posts if verified by administrator review.

4.0 Personal Chat Module

4.1 Private Messaging

4.1.1 The system shall allow users to engage in one-on-one conversations via text messages, image sharing, community post sharing and emotion diary sharing.

4.1.2 The system shall allow users to reply to specific messages sent or received.

4.1.3 The system shall send notifications to the users when receiving new messages.

4.1.4 The system shall allow users to see the last online status of other users.

4.1.5 The system shall automatically block the message sending when there are illegal terms in the text messages.

4.1.6 The system shall allow users to report the specific user message for administrator review.

4.1.7 The system shall allow users to block the specific user from sending messages to him or her.

4.1.8 The system shall notify the user if a message is blocked due to privacy violation.

4.2 Identity Control

4.2.1 The system shall allow users to select the option whether to chat anonymously or reveal their identity during private messaging.

4.2.2 The system shall automatically assign a random name to the user who has selected anonymous chat.

4.2.3 The system shall prevent anonymous users from impersonating real usernames.

4.2.4 The system shall prevent anonymous users from viewing detailed profile information of others.

4.2.5 The system shall allow the user to switch the identity mode in the personal chat from real identity to anonymous identity and vice versa.

5.0 Report Module

5.1 Productivity Report

5.1.1 The system shall generate reports summarizing all completed, ongoing and missed tasks over a selected time period.

5.1.2 The system shall analyze and display the task completion rate and overall performance trend.

5.1.3 The system shall allow users to export reports as PDF format.

5.1.4 The system shall provide visualizations, graphs and charts when displaying the summaries of tasks statistics.

5.1.5 The system shall allow users to filter the reports based on task categories, priority level and date range.

5.2 Social Performance Report

5.2.1 The system shall generate reports which reflect users' community engagement, including number of posts made, comments, likes given and conversations started.

5.2.2 The system shall track help-seeking and support-offering behaviors to measure social connectivity.

5.2.3 The system shall categorize social interactions based on different types such as supportive and informative.

5.2.4 The system shall summarize which communities or topics the user has engaged with the most.

5.3 Personalized Insight and Suggestions

5.3.1 The system shall provide suggestions to users for improving work productivity based on users' activity data.

5.3.2 The system shall assess users' social engagement progress and provide insights to improve social engagement.

5.3.3 The system shall provide the scores based on the support-seeking and support-giving patterns to evaluate user potential social roles.

3.3.4 Non-Functional Requirements

Product Requirements

1.0 Usability

1.1 The system shall provide a user-friendly interface that is intuitive for novice users without requiring external assistance.

1.2 The system shall provide simple navigation and offer onboarding guidance for first-time users.

1.3 The system shall allow each module to be accessible within a few clicks.

1.4 The system shall ensure the appropriate and consistent font size, icons and buttons for mobile devices.

2.0 Efficiency

2.1 The system shall minimize manual input effort by offering speech-to-text task entry and NFC tagging.

2.2 The system shall ensure a fast retrieval of data when maintaining the user interface smoothness.

3.0 Performance

3.1 The system shall load each interface within 3 seconds on every device.

3.2 The system shall send notifications and reminders in real-time.

3.3 The system shall be responsive when executing concurrent tasks.

4.0 Portability

4.1 The system shall function on supported Android platforms.

4.2 The system shall run smoothly on both low-end and high-end mobile devices.

Process / Organizational Requirements

1.0 Delivery

1.1 The system shall follow Incremental methodology to introduce new modules or enhanced functionality in each increment.

2.0 Implementation

2.1 The system shall be developed using Dart programming language with Flutter framework.

2.2 The system shall use Google Firebase to store system configuration and users' data.

2.3 The system shall use Git and GitHub for system version control.

External Requirements

1.0 Interoperability

1.1 The system shall support integration with third-party APIs such as calendar and mental health resources.

1.2 The system shall export the database data in standard JSON format.

1.3 The system NFC functionality shall comply with standard tag types such as NTAG215 or Mifare.

2.0 Privacy

2.1 The system shall securely store and encrypt all users' personal data such as chat content.

2.2 The system shall not disclose any users' sensitive information to outside parties.

2.3 The system shall acknowledge users about the information being collected and provide options to users to opt in or out of data sharing.

3.0 Safety

3.1 The system should filter or block inappropriate content in chat using AI-based detection.

3.2 The system should allow users to report and flag violated contents for moderation review.

3.4 Development Environment

3.4.1 Hardware and Device Specification

Development Machine

The development of the BetterU mobile application will be conducted using a gaming laptop for simulating the real-world usage scenarios. The gaming laptop will mainly be used for developing the Flutter frontend codes and Python backend codes. This machine will also be used for creating virtual devices for debugging purposes. Not only that, this machine will also be used for designing the UI and UX of BetterU mobile application and developing AI features since it can provide sufficient resources and performance on CPU and GPU.

Aspect	Specification
Device Model	ASUS TUF Gaming A15 FA507NU
Processor	AMD Ryzen 7 7735HS with Radeon Graphics
RAM	16 GB
Storage	512 GB
Graphics	NVIDIA® GeForce RTX™ 4050 Laptop GPU
Operating System	Windows 11 Home

Table 3.4.1: Development Machine Specification

Mobile Device for Testing

From the testing machine, an Android mobile device is chosen for running the BetterU mobile application. It can be used as a real-device testing during iterative development for ensuring the performance, layout responsiveness and functionality of the app under realistic mobile conditions. When applying the real testing device, it can provide more comprehensive features such as camera and NFC functionalities. These can effectively help for testing the BetterU mobile application in a more comprehensive way such as NFC sensing for writing in and out from NFC tag.

Aspect	Specification
Device Model	OPPO Reno 13 Pro 5G
Processor	Mediatek Dimensity 8350
RAM	12 GB
Storage	512 GB
Operating System	ColorOS 15.0 (Android 15)
NFC Support	Yes

Table 3.4.2: Mobile Device for Testing Specification

Virtual Machine

When there is no real physical mobile device for testing, the Android Studio can still offer a Virtual Studio Emulator for creating an Android virtual machine. Thus, all the Flutter code can be directly tested within the virtual machine without physical access to any smartphone. It is especially useful for quick debugging and UI adjustments when providing the flexibility when real hardware is not available.

Aspect	Specification
Device Name	Medium Phone
API Level	35
Resolution (px)	1080 x 2400
Density	420 dpi
ABI List	x86_64
Size on Disk	6.4 GB

Table 3.4.3: Virtual Machine Specification

NFC Tags

During the development of NFC functionalities in BetterU mobile application, NFC tags are also required for acting as a medium to store the data such as users' task for clocking in and out. Thus, the app functions can be directly implemented on the existing physical NFC tags such as NFC-based task tracking features. These NFC tags will have a wide compatibility with the Android devices and BetterU mobile application to detect and update task progress.

Aspect	Specification
Tag Type	NTAG213
Frequency	13.56 MHz
Memory	144 bytes of usable NDEF memory
Features	Read and write capability with supported password encryption
Form Factor	Adhesive sticker tags
Compatibility	Fully compatible with Android devices that support NFC.

Table 3.4.4: NFC Tags Specification

3.4.2 Software Tools and Platforms

Android Studio

Android Studio acts as the primary integrated development environment (IDE) for building and testing the BetterU mobile application. The main strength of this IDE is it can provide a robust environment with tools such as Android Emulator, layout editor and APK analyzer.

(Android Developers, n.d.) It also supports different versions of APIs with different Android devices with various layouts. Plus, it has high compatibility and performance when perfectly integrating with Flutter plugins. All the Flutter code can be directly compiled and running on its virtual Android devices for displaying the smooth user interface.

Aspect	Specification
Version Used	Android Studio Meerkat Feature Drop | 2024.3.2 Patch 1
System Requirement	● Operating System: Latest 64-bit version of Windows ● RAM: 8 GB ● CPU: Virtualization support Required (Intel VT-x or AMD-V) ● Disk space: 32 GB
Installed Plugins	● Dart (version 243.27824.5) ● Flutter (version 86.0.1) ● Git and GitHub

Table 3.4.5: Android Studio Specification

Visual Studio Code

Visual Studio Code is also utilized alongside Android Studio for backend development using Python and FastAPI. It provides a lightweight performance and extensive extension marketplace. In BetterU development, it mainly focuses on developing the Python backend code, AI features and integration with Flutter code using FastAPI. On the other hand, it also supports syntax highlighting, code linting, Git integration and terminal access in a unified interface. (Microsoft, n.d.) So, all the operations needed on different platforms can be easily streamlined within one application.

Aspect	Specification
Version Used	1.102
System Requirement	● Processor: 1.6 GHz ● RAM: 1 GB ● Operating System: Windows 10 and 11 (64-bit) ● Disk: 500 MB minimum
Key Features	● Python extension support ● Git integration ● Integrated terminal
Use Case	● Backend API development ● Python environment management

Table 3.4.6: Visual Studio Code Specification

Github Desktop

Github Desktop is one of the most popular and important tools when it comes to the code collaboration within a development team in BetterU mobile application. It helps developers to easily manage the version control process when providing the graphical user interface for Git

operations. For example, commit, merge, push and pull requests. It enables developers to keep track of the code progress from time to time. When there is any update from any team member, all other team members can directly synchronize their local code with the updated code in the GitHub repository. Meanwhile, it also acts as a free cloud-based code backup for BetterU mobile application developers. Thus, all the code can still be retrieved properly even when there are unexpected issues happening on any team members' development machine. (GitHub, n.d.)

Aspect	Specification
Version Used	3.5.2
System Requirement	● OS: Windows 10 or 11 ● RAM: 2 GB minimum ● Disk: 200 MB
Key Features	● Visual Git interface ● Branch comparison and merging ● Integrated GitHub sync
Use Case	● Version control ● Issue tracking ● Collaborative codebase management

Table 3.4.7: Github Desktop Specification

3.4.3 Programming Languages and Frameworks

Dart with Flutter

During the development of BetterU mobile application, the Dart programming language and Flutter framework were primarily used for the front-end mobile application development. Dart is a general-purpose language developed by Google which is optimized for building the user interfaces on various platforms such as Android and Windows. It is well-suited for Flutter due to its just-in-time (JIT) and ahead-of-time (AOT) compilation. This can effectively improve both development speed and runtime performance. On the other hand, the Flutter framework which is also developed by Google which supports flexible UI designs and ensures the compatibility across Android platforms with a single codebase. (Flutter, 2024)

Python with FastAPI

On the server side, the BetterU mobile application will use Python as the backend programming language. This is because it provides a high simplicity, wide community support and high availability of machine learning and natural language processing (NLP) libraries for developers for free. For example, spaCy, scikit-learn and setFit are applied within the BetterU mobile application. Meanwhile, the FastAPI framework will also be used to implement the backend REST API service. It provides a high-performance framework for building the APIs with Python type hints. So, the Flutter code can easily integrate with the Python backend code to perform all the backend services. (Tiangolo, 2024)

Criteria	Dart with Flutter	Python with FastAPI

Primary Use	Front-end mobile app development (UI and UX)	Back-end RESTful API development
Programming Language	Dart	Python
Framework	Flutter	FastAPI
Compilation	JIT for development and AOT for production	Interpreted and supports async execution
Community and Ecosystem	Strong and growing Flutter community	Growing FastAPI ecosystem, large Python community
Platform Support	Cross-platform (Android, iOS, Web and Desktop)	Server-side which is deployable on any OS with Python support

Table 3.4.8: Comparison between Dart with Flutter and Python with FastAPI

3.4.4 Database and Storage Services

Google Firebase

Google Firebase serves as the primary cloud-based and non-relational database used in BetterU mobile applications. Meanwhile, it also supports Firebase Authentication, Firestore and Firebase Cloud Messaging for allowing BetterU mobile application to seamlessly store, retrieve and update all the real-time data. Firebase Realtime Database and Cloud Firestore can provide secure storage and retrieval of structured data such as the task records, summaries, NFC tag associations and user settings. Via Google Firebase, the security of the database data can also be ensured via user authentication and cross-device synchronization features provided. (Firebase, n.d.) It also has a high compatibility to fit with the Flutter framework and security features to provide the services for mobile-first development including BetterU mobile application.

Aspect	**Specification**
Platform Type	Cloud-based Backend-as-a-Service (BaaS)
Key Features	● Firebase Authentication ● Cloud Firestore ● Firebase Cloud Messaging ● Firebase Hosting
Plan	Spark Plan (Free tier limits): ● 50K document reads per day ● 1 GB storage ● 10K monthly cloud functions invocations ● 5 GB hosting bandwidth
Use Case	● Cloud storage ● User authentication ● Real-time syncing

Table 3.4.9: Google Firebase Specification

3.4.5 Application Architecture and Deployment Environment

BetterU mobile application is designed to use a **client-server architecture** where the frontend is developed using Dart with Flutter and the backend is powered by Python with FastAPI. This architecture can help for separating the user interface and the server-side logic. Thus, both layers can be developed and maintained independently.

The **Flutter-based mobile frontend** mainly focuses on delivering interactive and responsive user interfaces across different screen sizes and platforms. In order to communicate with the Python backend code, it will use **RESTful APIs supported by FastAPI** to implement the backend features in BetterU mobile app. On the server side, FastAPI will also handle data processing, user authentication logic and manage the integration with cloud services such as **Google Firebase**.

After the development phase for each increment, the BetterU mobile app will be **deployed and tested on physical Android devices with API level 21 and above**. The device will also support the NFC tag scanning since it is part of the mobile app features. Via this modular deployment approach, all the **modules can be developed one by one via continuous incremental development**. Meanwhile, the mobile app can also be debugged efficiently for future improvements and maintenance.

3.4.6 UI/UX Design Tools

When it comes to user interface design, Figma will be primarily used as the UI/UX design tool for BetterU mobile application before the development stage. It allows the mobile app designers to create the wireframes, mockups and interactive prototypes. Varieties of reusable and interactive components also help designers to create responsive and user-friendly interfaces. (Figma, 2024) In BetterU mobile application, it is typically useful for providing a seamless sharing, version control and feedback throughout the design process. Thus, all the team members can collaboratively design the user interface of the BetterU mobile app when synchronizing the latest updates simultaneously.

Aspect	**Specification**
Platform Type	Web-based (Cloud) with desktop app support (Windows)
Key Features	● UI/UX design ● Prototyping ● Wireframing ● Collaboration
System Requirement	● Operating System: Windows 8.1 or later ● Graphics: Windows (Nvidia or AMD) ● Minimum Browser Version: ○ Chrome 99 or later ○ Microsoft Edge 121 or later
Pricing Tier Used	Free (Education Plan)

Table 3.4.10: Figma Specification

3.4.7 External Libraries, APIs and Plugins

Speech Recognition and Transcription

Vosk API and OpenAI Whisper are used for real-time and offline speech-to-text conversion tasks in BetterU mobile application goal assistance module. The Vosk supports a lightweight, on-device speech recognition which is compatible with Android and Python environments. (Vosk, 2024) At the same time, Whisper is also used for generating highly accurate transcripts of longer audio recordings in multiple languages. (OpenAI, 2024) Via the combination of both libraries, the accuracy of the speech transcription can be improved effectively to extract the task input from the users.

Natural Language Processing

In order to extract the task insights, the libraries like spaCy and dateparser will be used in BetterU mobile application. spaCy supports tokenization, entity recognition and dependency parsing. (Explosion AI, 2024) It can help BetterU mobile app to understand and extract the key points from users' task input such as task item. Meanwhile, the dateparser library is also used for extracting and normalizing human-readable dates from free-text inputs. (Scrapinghub, 2024) It acts as an assisting tool for extracting the specific date and time from users' task input.

Machine Learning and Text Classification

Scikit-learn library is used for implementing traditional ML models to detect the task categories in the goal assistance module. (Pedregosa et al., 2011) It can effectively detect and categorize a user's task input without the user manually selecting the task categories. Moreover, the SetFit library is also integrated for few-shot text classification using Sentence Transformers. (Zimmer, 2023) Without requiring large datasets, it can still help BetterU to fine-tune the task classification model to accurately classify the task categories and priority level.

Backend API Interface

FastAPI framework acts as a primary RESTful API backend for the BetterU mobile application. It helps for offering the scalable communication between the mobile frontend and the backend services in the mobile app. Besides, it also offers the automatic data validation, interactive Swagger documentation and asynchronous support for concurrent requests. It is highly suitable for prototyping and deploying the machine learning and NLP models used in BetterU mobile applications since it has a high performance and developer-friendly syntax. (Tiangolo, 2024)

3.5 Chapter Summary and Evaluation

This chapter has described how BetterU mobile application was planned, analyzed and technically prepared. The software process model applied for this project is Incremental Development Model. The main reason is it can fit well with the behavior of BetterU since each feature can be built and tested one by one. This characteristic can effectively allow the flexibility of development and make it easier to gather the feedback from supervisors early. Thus, the improvements can be made in future increments without waiting until the end.

In order to collect useful information before developing a BetterU mobile application, there are two fact-finding techniques applied which are observation and online research. Observation will mainly focus on observing the daily behavior of people around me such as friends and family members. Meanwhile, the online research was conducted via exploring the trusted resources from the Internet. The main goal of this research is to obtain more information to understand common issues faced in task planning and personal productivity. So, it can make sure that BetterU mobile application is fulfilling the real-world needs, especially for target users which are students and office workers.

Apart from that, this chapter also discusses the detailed requirement analysis starting from the use case diagram, use case descriptions and listings of functional and non-functional requirements. These diagrams and listings help BetterU project to clearly define what the app should do and how it should behave when dealing with different scenarios. Thus, the future development process can be conducted in a more manageable way.

Eventually, the development environment and tools used in developing BetterU mobile application are explained. It includes the devices, programming tools and software frameworks such as Flutter and FastAPI. It also discusses the crucial backend components such as database and UI/UX design tools like Figma. Meanwhile, it highlights the core external libraries and APIs used for supporting attractive features such as speech recognition, natural language processing and machine learning. These tools can help BetterU mobile app to provide a more convenient and intelligent solution for understanding user input and deliver the intelligent suggestions.

Chapter 4

System Design

4 System Design

This chapter will describe and explain the detailed system design of the BetterU mobile application for specifying the structural and behavioral characteristics of the system. It includes various design models and specifications for illustrating how the system works and interacts with the users such as students, office workers and administrators. For example, sequence diagram, state chart diagram and activity diagram will describe the dynamic processes and user interactions. Meanwhile, the user interface design will show the overview of the visual layout of BetterU mobile application. Furthermore, the data design will cover the class diagram, entity relationship diagram and data dictionary which define the logical structure and the flow of information. The report designs will describe what components, data and granularity requirements are needed in each report such as productivity report and social performance report. Besides, the process design (activity diagram) and software architecture design will demonstrate how the system's components are organized and deployed. Lastly, the AI algorithms that will be applied within the mobile application will be highlighted with sample training dataset provided for offering the intelligent features such as task categorization and rescheduling prediction.

4.1 Sequence Diagram

Goal Assistance Module

Create Task

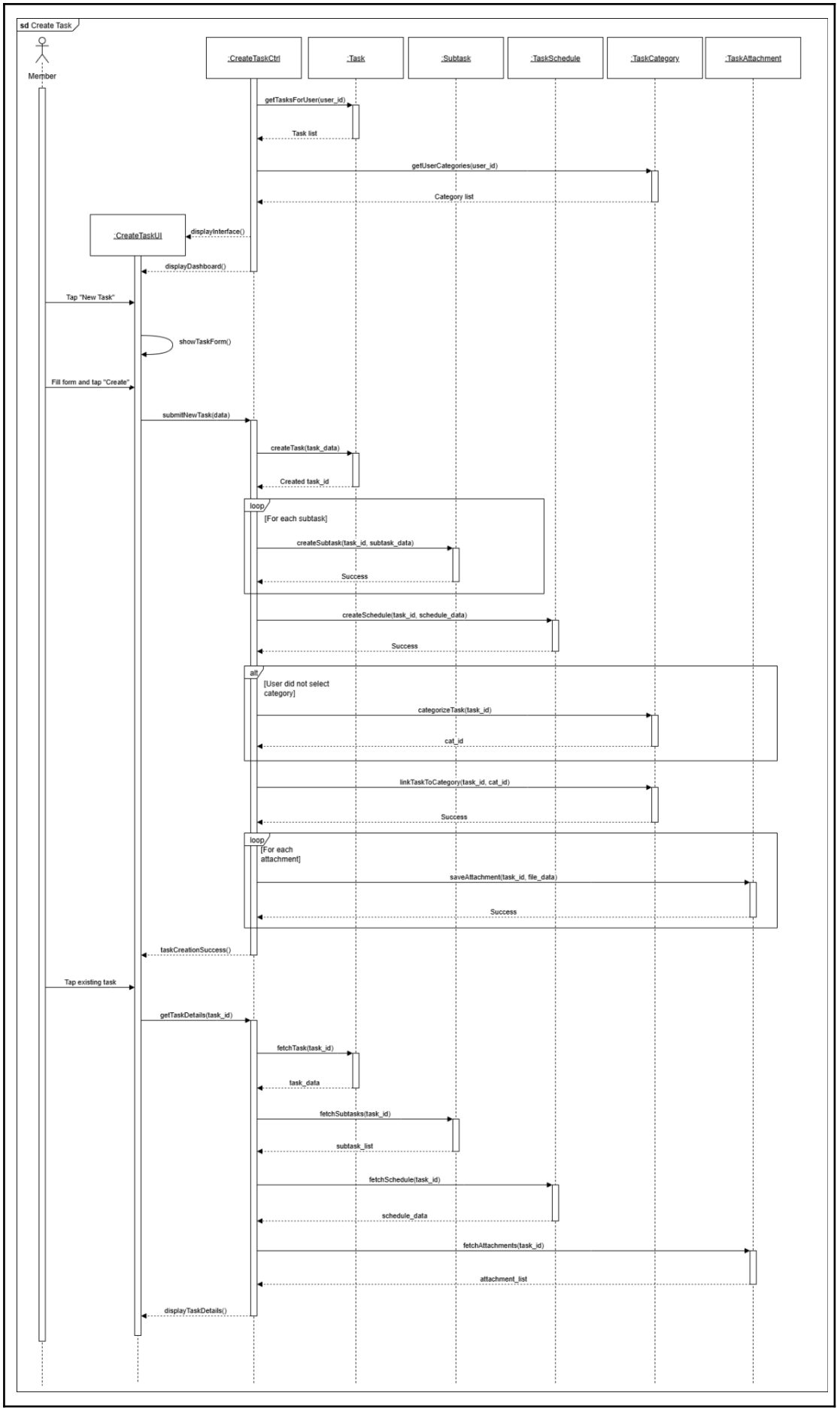

sd Create Task
Member
:CreateTaskCtrl
:Task
:Subtask
:TaskSchedule
:TaskCategory
:TaskAttachment
:CreateTaskUI
getTasksForUser(user_id)
Task list
getUserCategories(user_id)
Category list
displayInterface()
displayDashboard()
Tap "New Task"
showTaskForm()
Fill form and tap "Create"
submitNewTask(data)
createTask(task_data)
Created task_id
loop
[For each subtask]
createSubtask(task_id, subtask_data)
Success
createSchedule(task_id, schedule_data)
Success
alt
[User did not select category]
categorizeTask(task_id)
cat_id
linkTaskToCategory(task_id, cat_id)
Success
loop
[For each attachment]
saveAttachment(task_id, file_data)
Success
taskCreationSuccess()
Tap existing task
getTaskDetails(task_id)
fetchTask(task_id)
task_data
fetchSubtasks(task_id)
subtask_list
fetchSchedule(task_id)
schedule_data
fetchAttachments(task_id)
attachment_list
displayTaskDetails()

Diagram 4.1.1: Sequence Diagram - Create Task (Goal Assistance Module)

Update Task

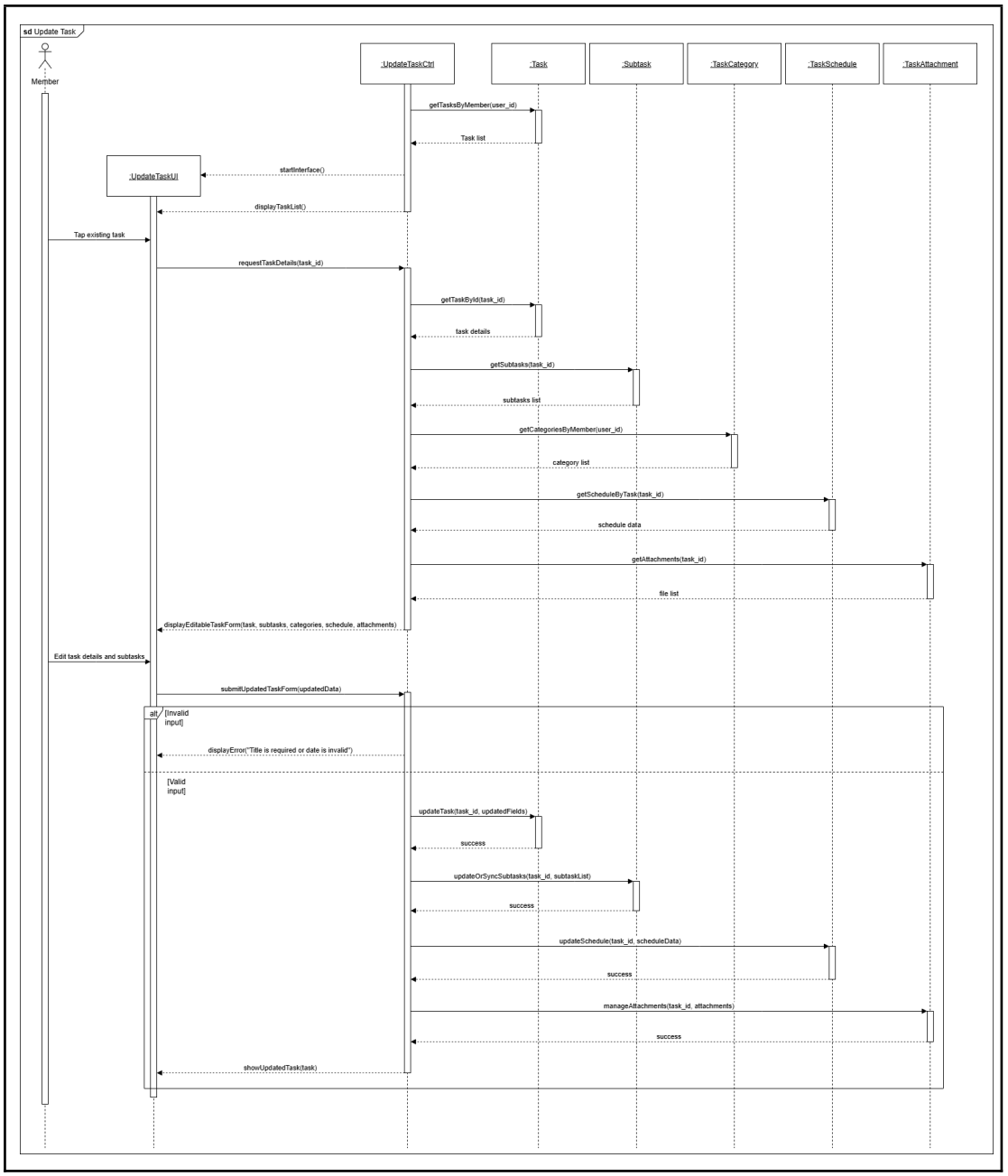


Diagram 4.1.2: Sequence Diagram - Update Task (Goal Assistance Module)

Delete Task

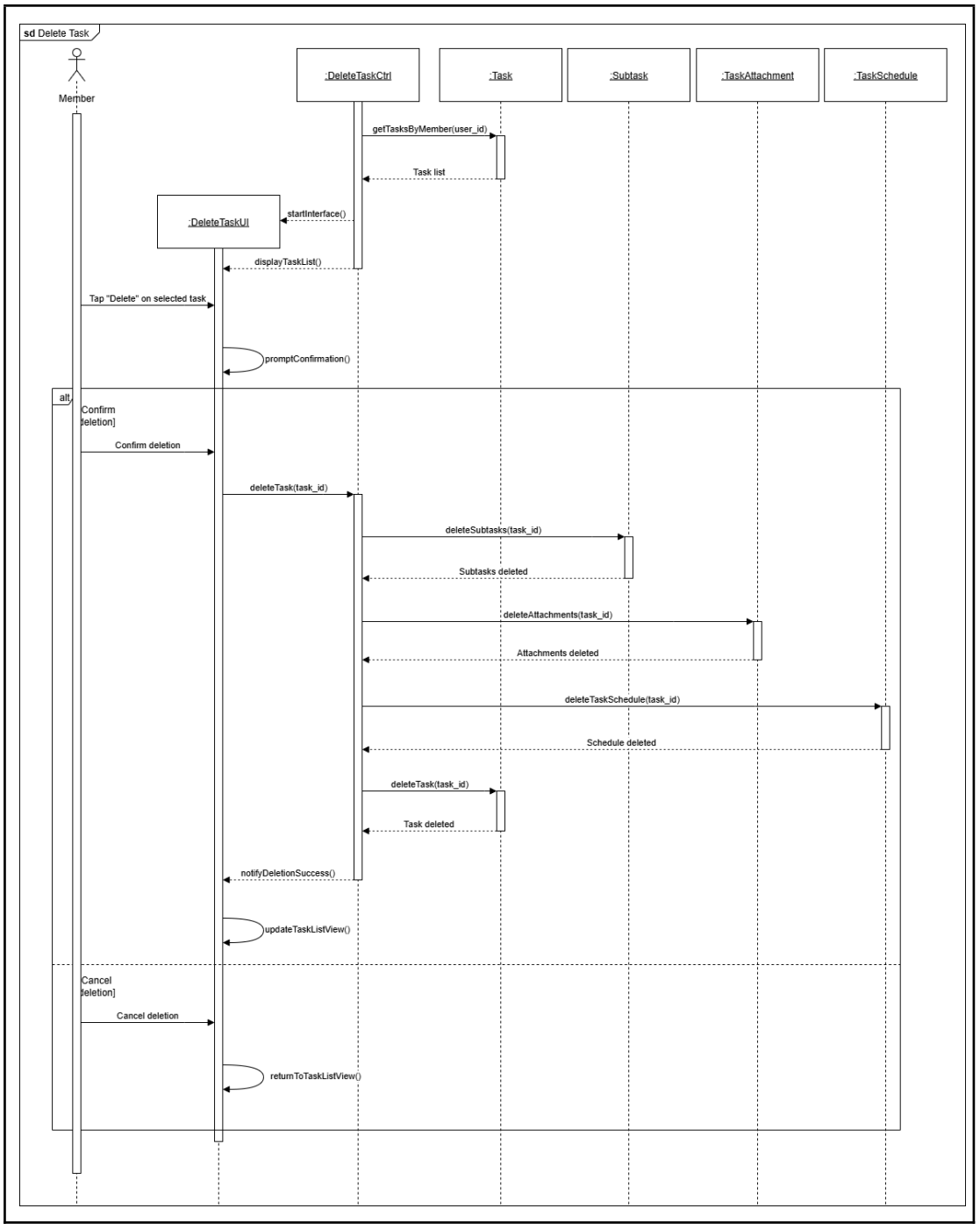


Diagram 4.1.3: Sequence Diagram - Delete Task (Goal Assistance Module)

Receive Goal Notification

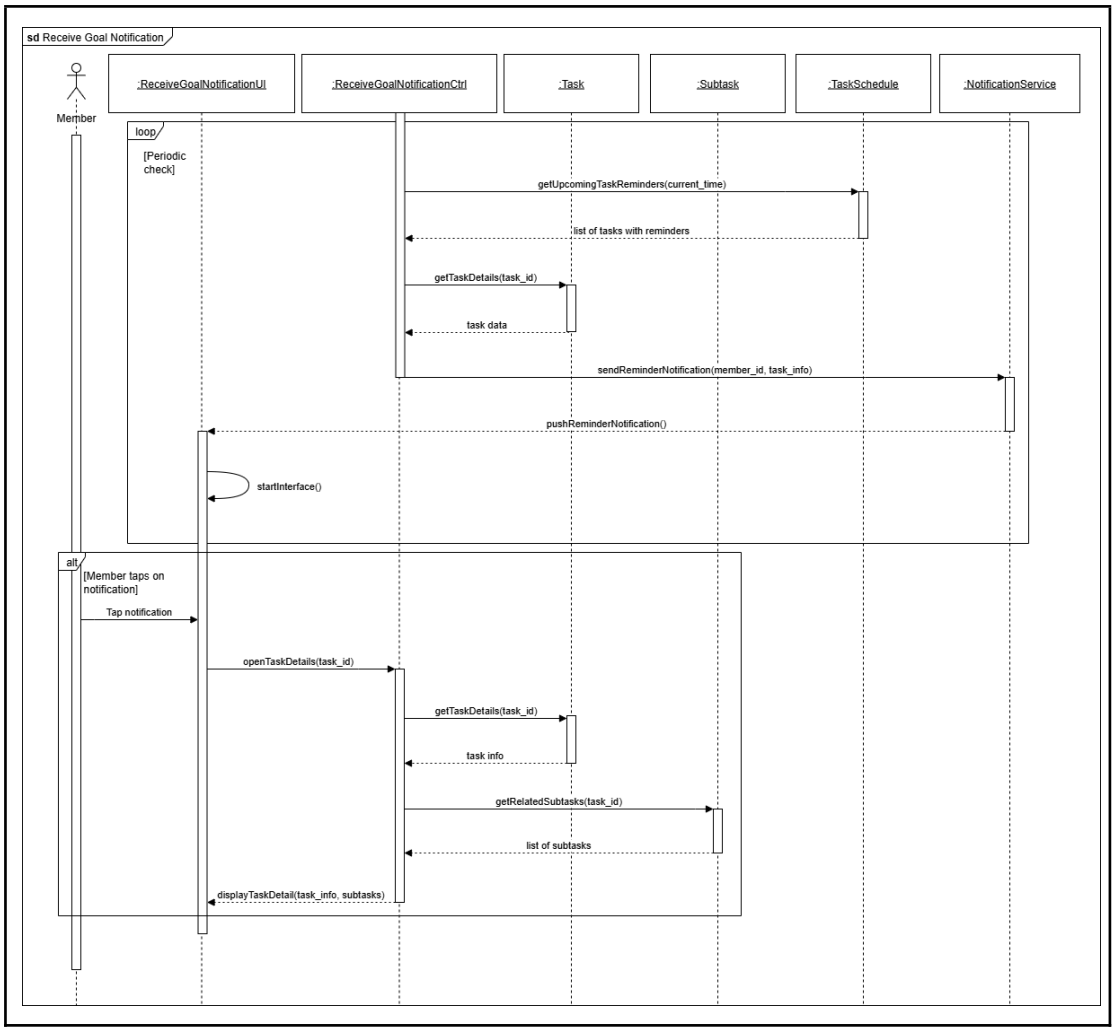


Diagram 4.1.4: Sequence Diagram - Receive Goal Notification (Goal Assistance Module)

Manage Smart Suggestions

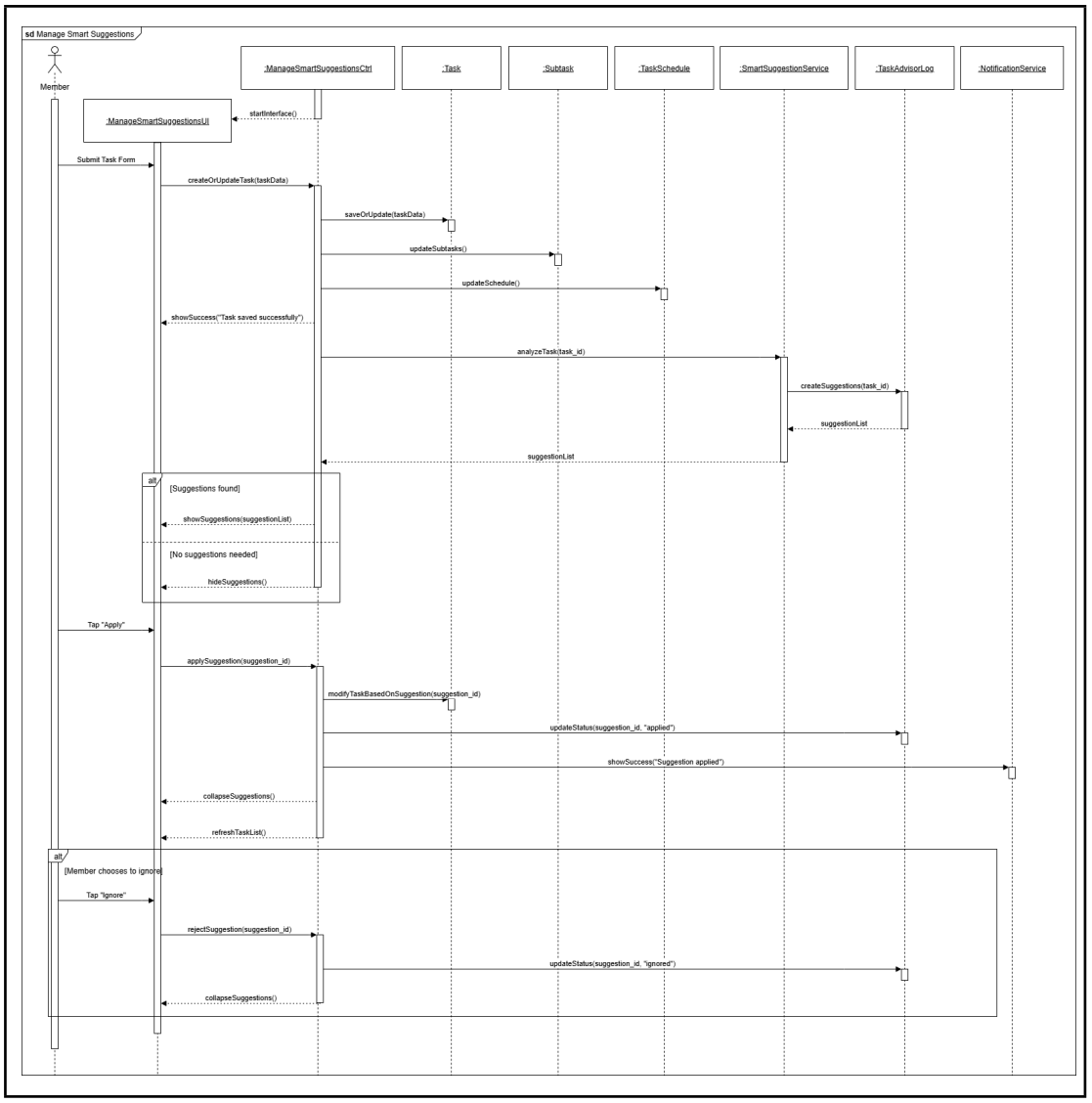


Diagram 4.1.5: Sequence Diagram - Manage Smart Suggestions (Goal Assistance Module)

Time Usage Tracker Module

Track App Usage

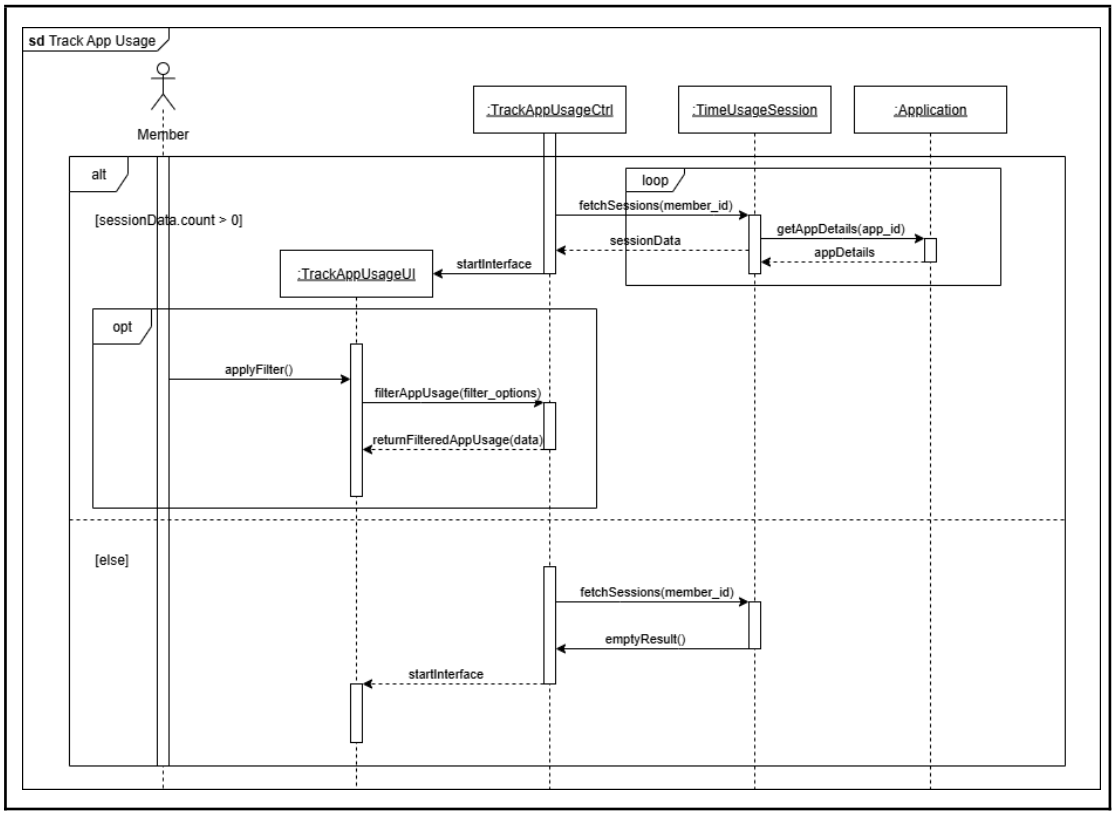


Diagram 4.1.6: Sequence Diagram - Track App Usage (Time Usage Tracker Module)

Manage NFC Tag

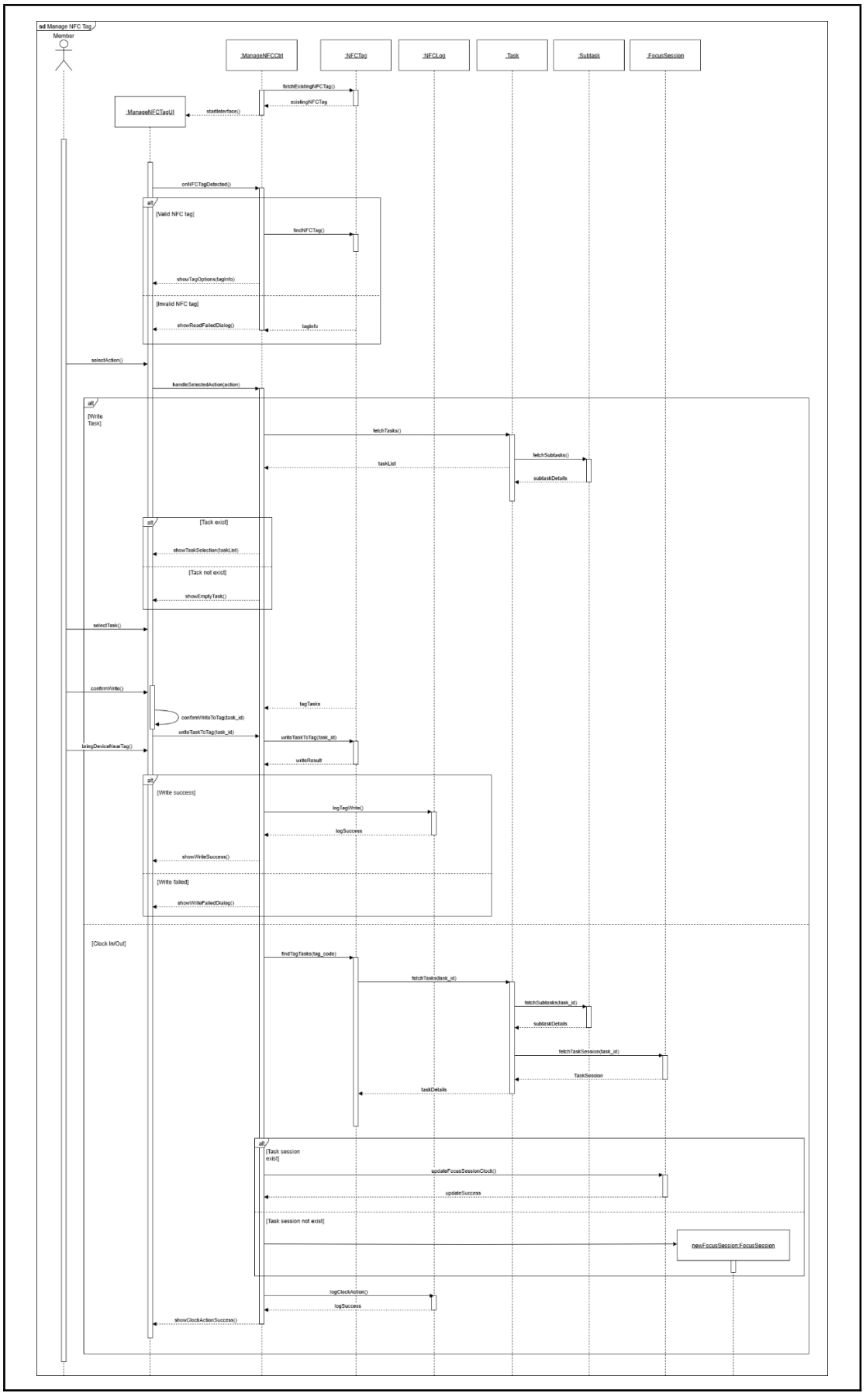


Diagram 4.1.7: Sequence Diagram - Manage NFC Tag (Time Usage Tracker Module)

View Task Summaries

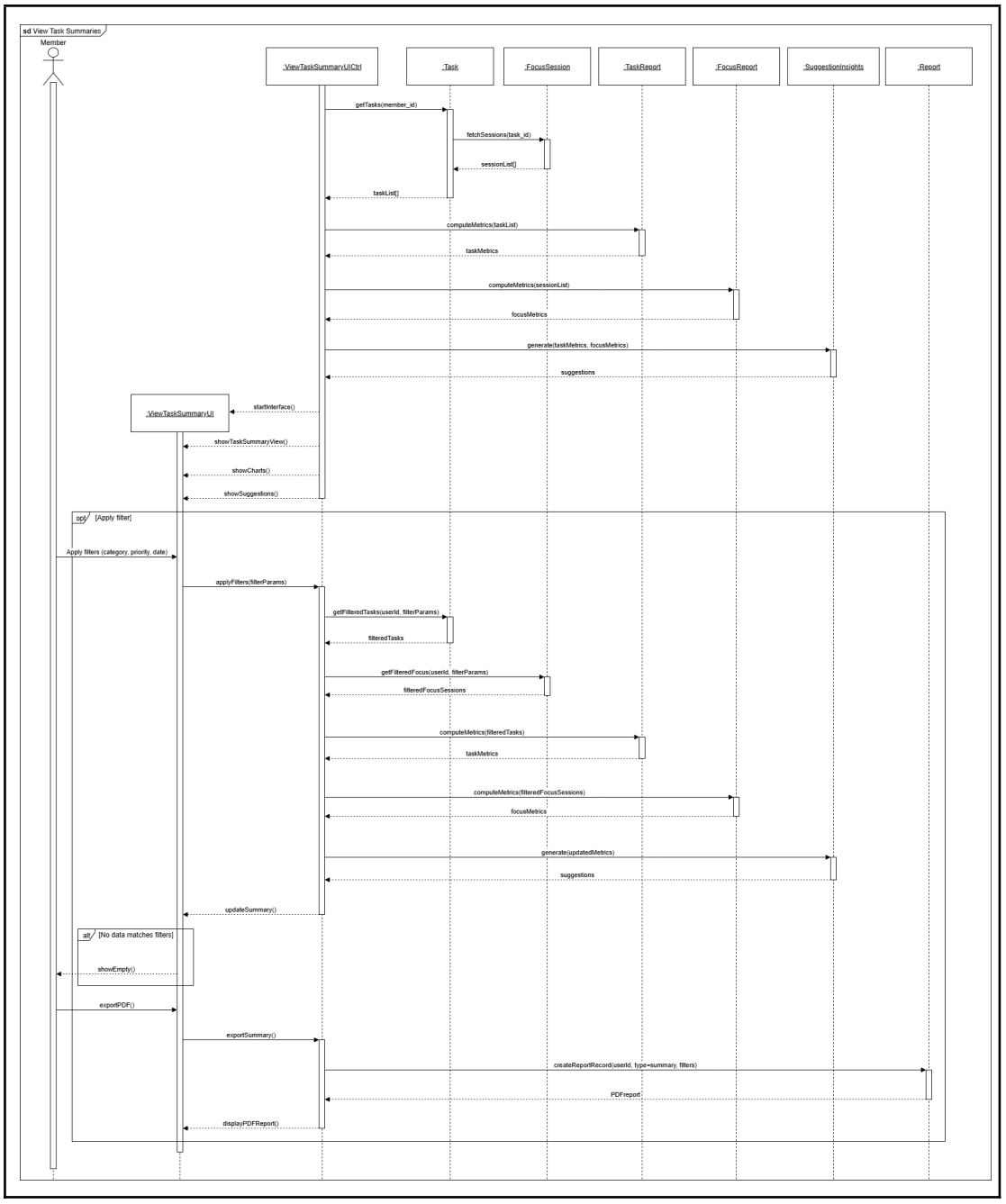
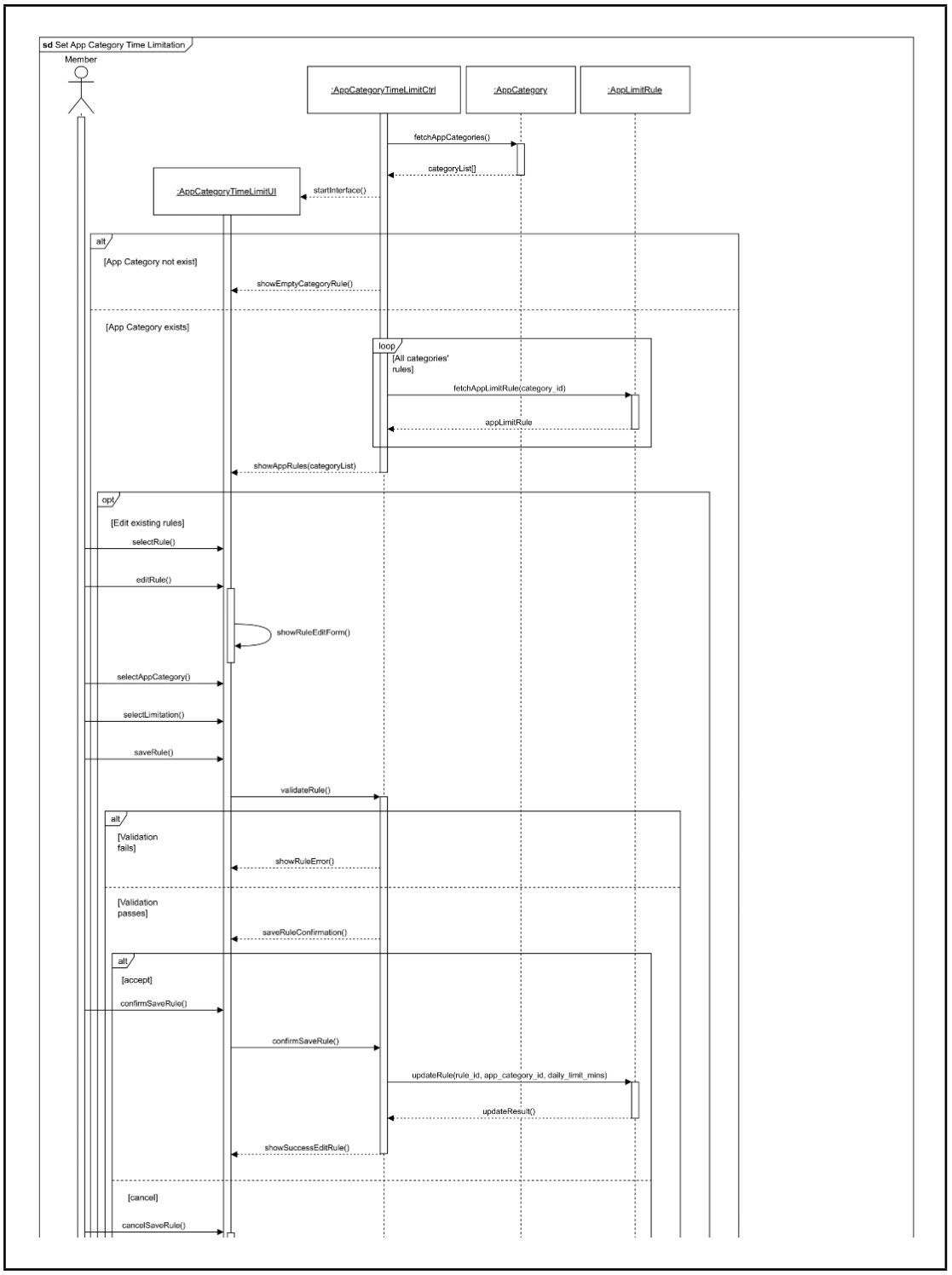


Diagram 4.1.8: Sequence Diagram - View Task Summaries (Time Usage Tracker Module)

Set App Category Time Limitation



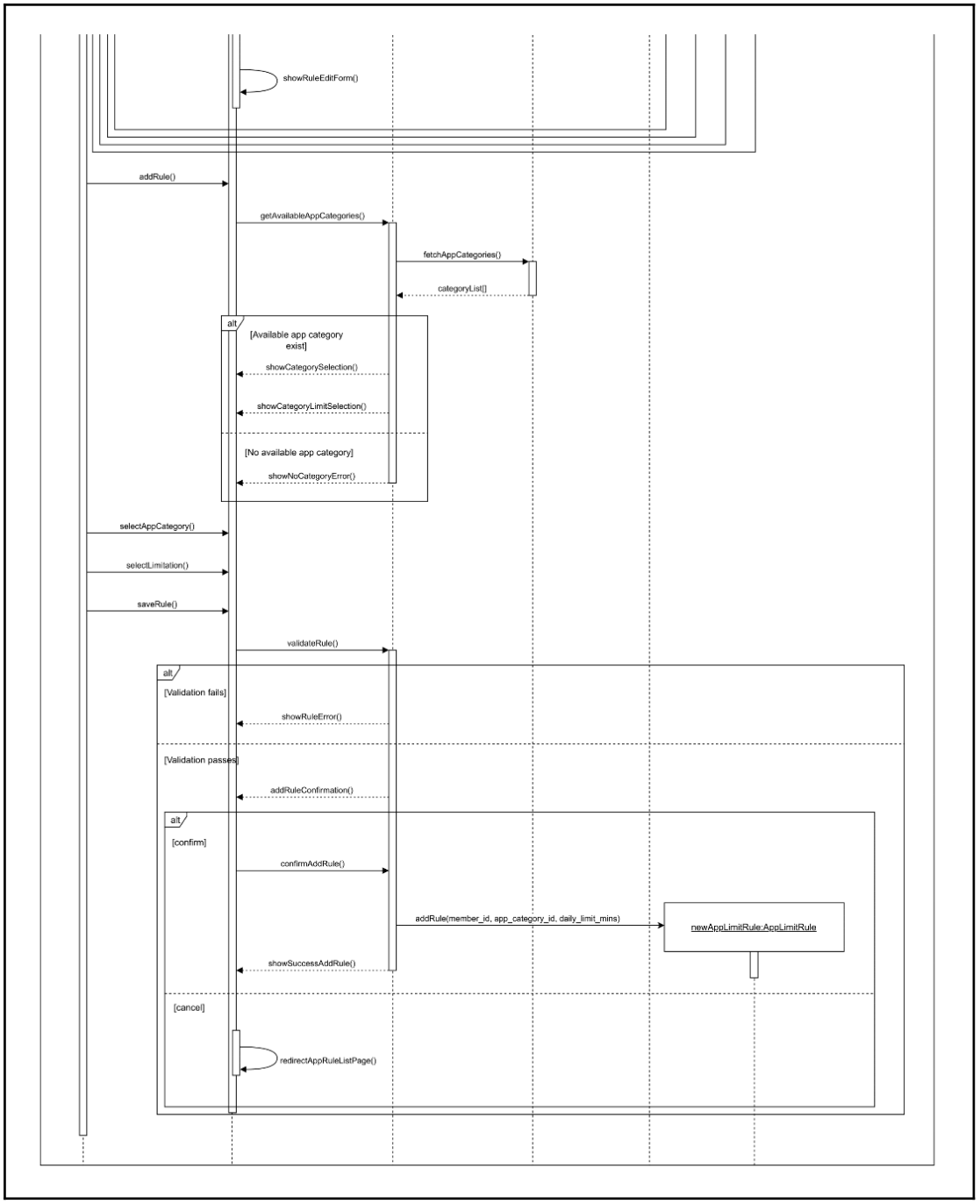


Diagram 4.1.9: Sequence Diagram - Set App Category Time Notification (Time Usage Tracker Module)

Receive Time Limit Notification

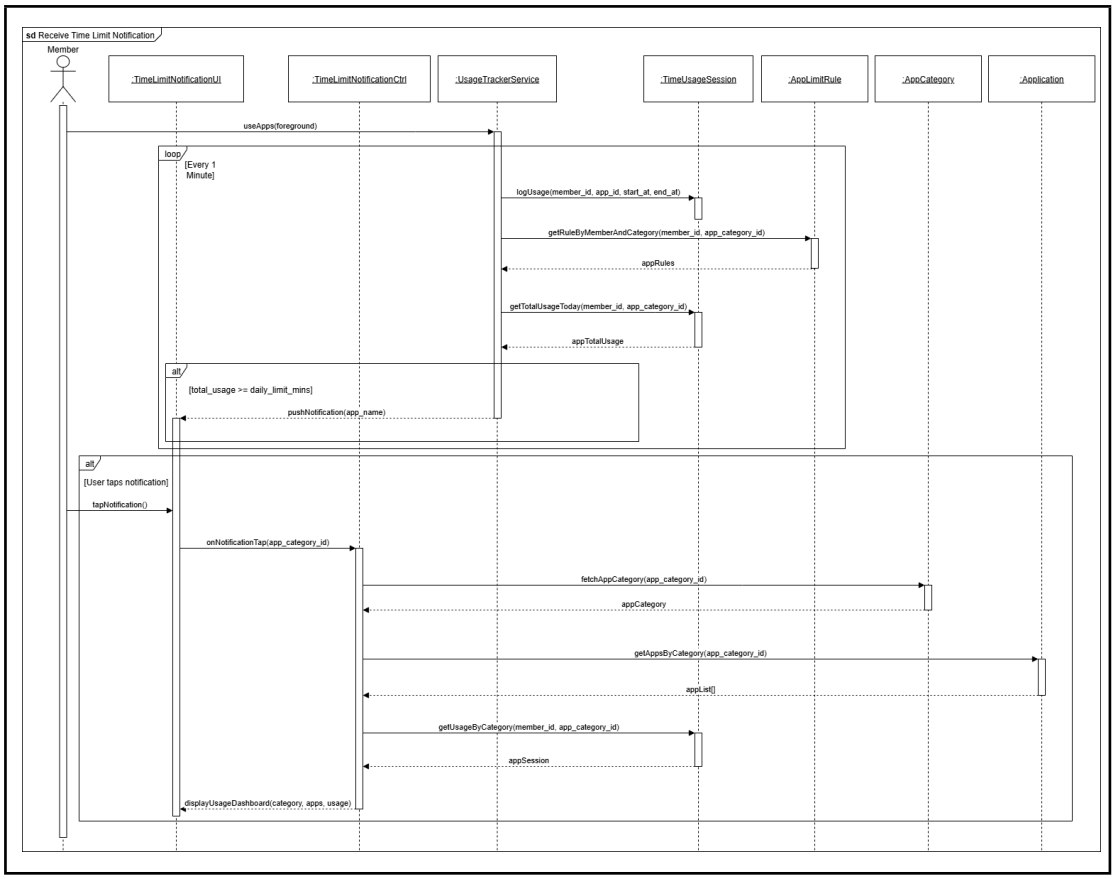
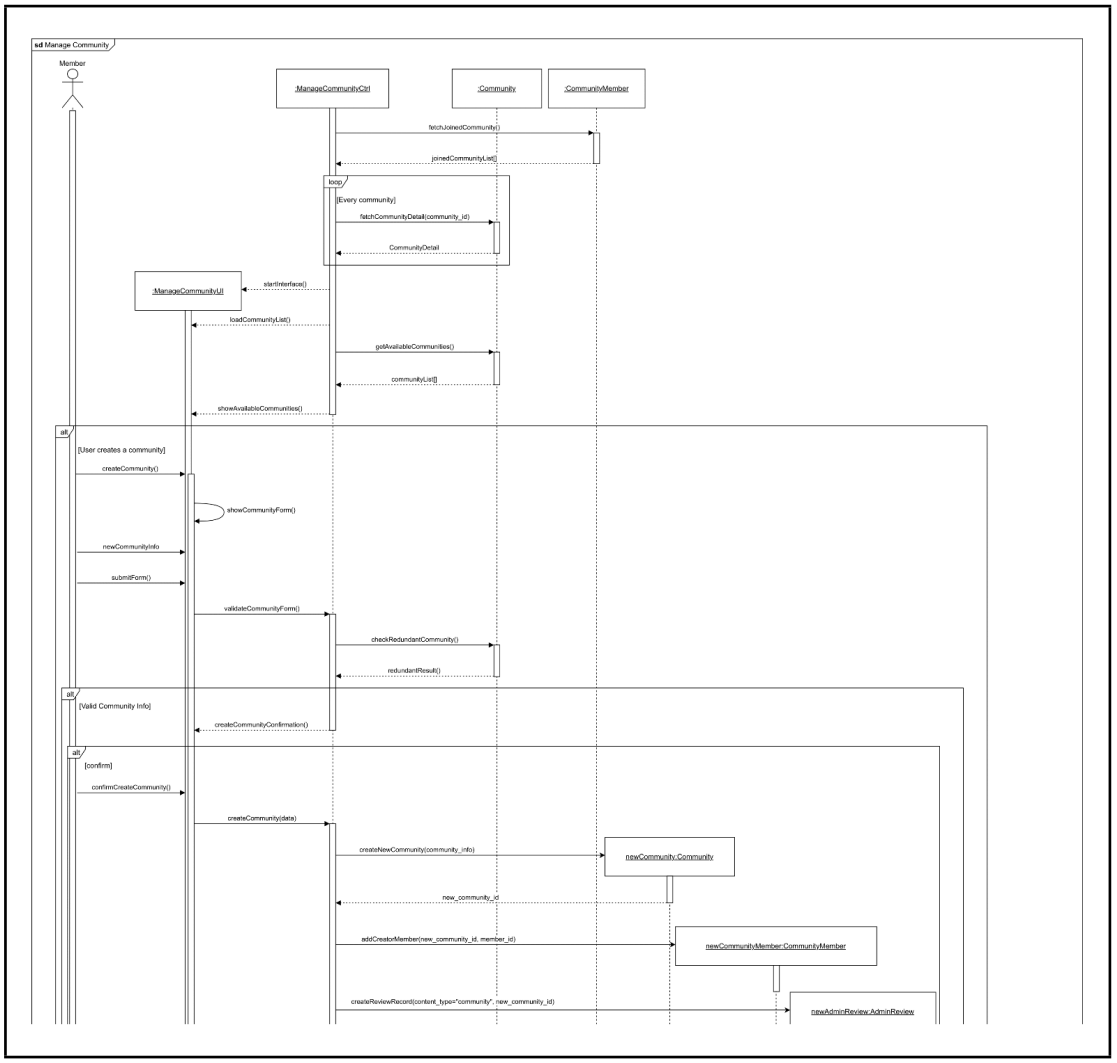


Diagram 4.1.10: Sequence Diagram - Receive Time Limit Notification (Time Usage Tracker Module)

Anonymous Community Module

Manage Community



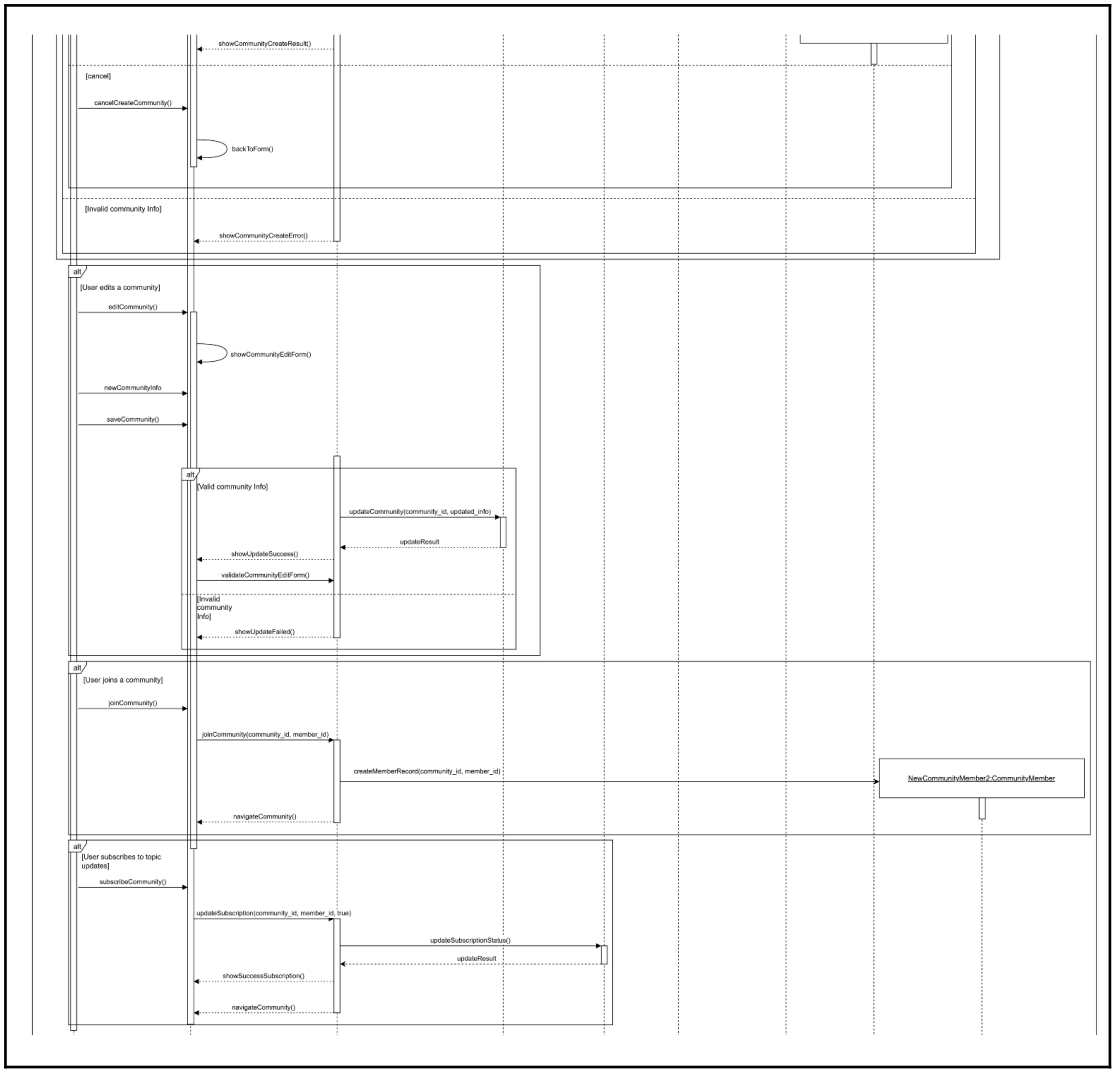
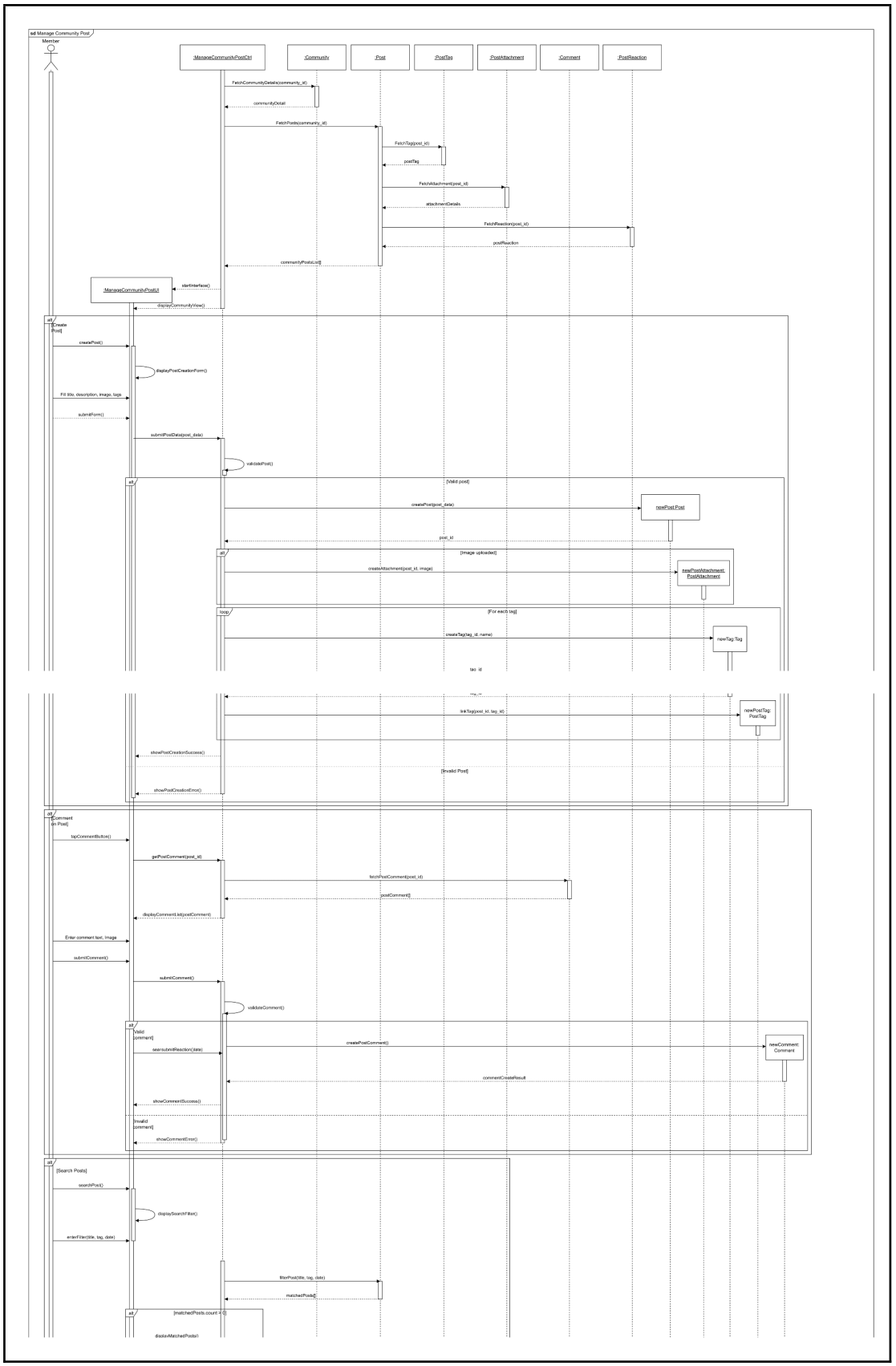


Diagram 4.1.11: Sequence Diagram - Manage Community (Anonymous Community Module)

Manage Community Post



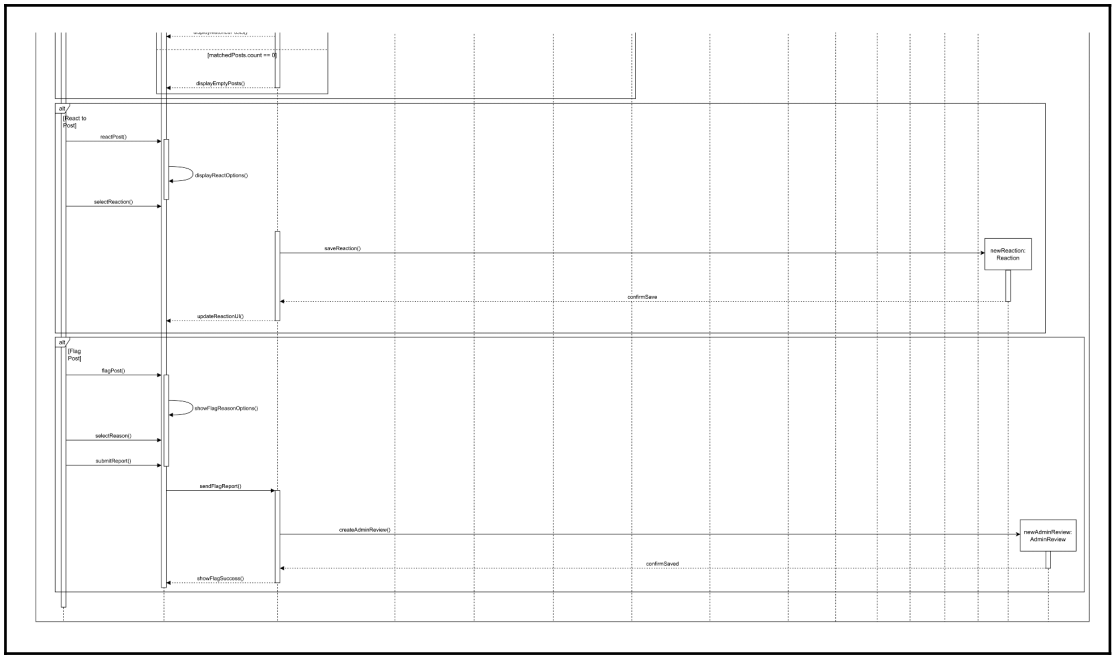
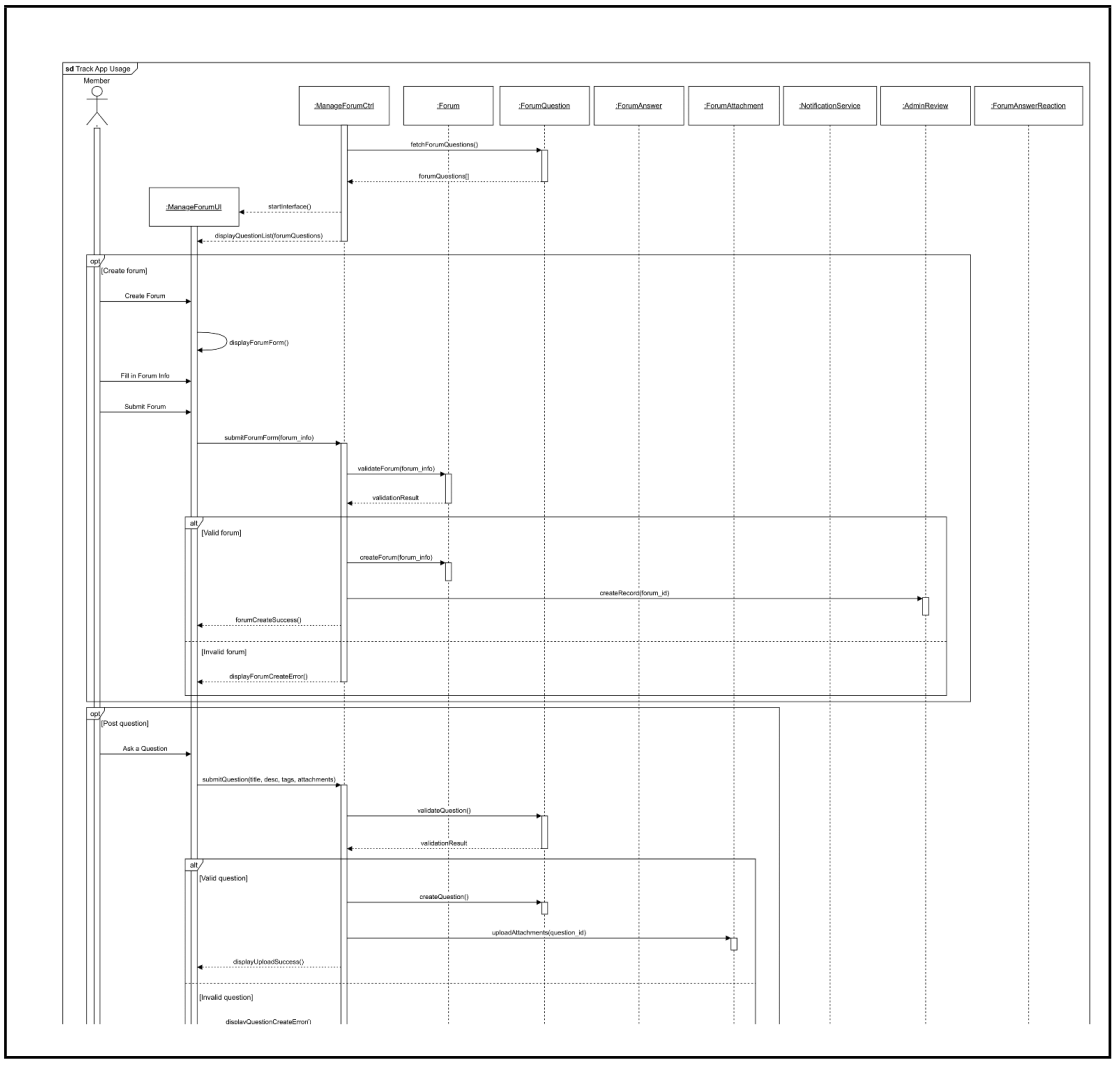


Diagram 4.1.12: Sequence Diagram - Manage Community Post (Anonymous Community Module)

Manage Forum



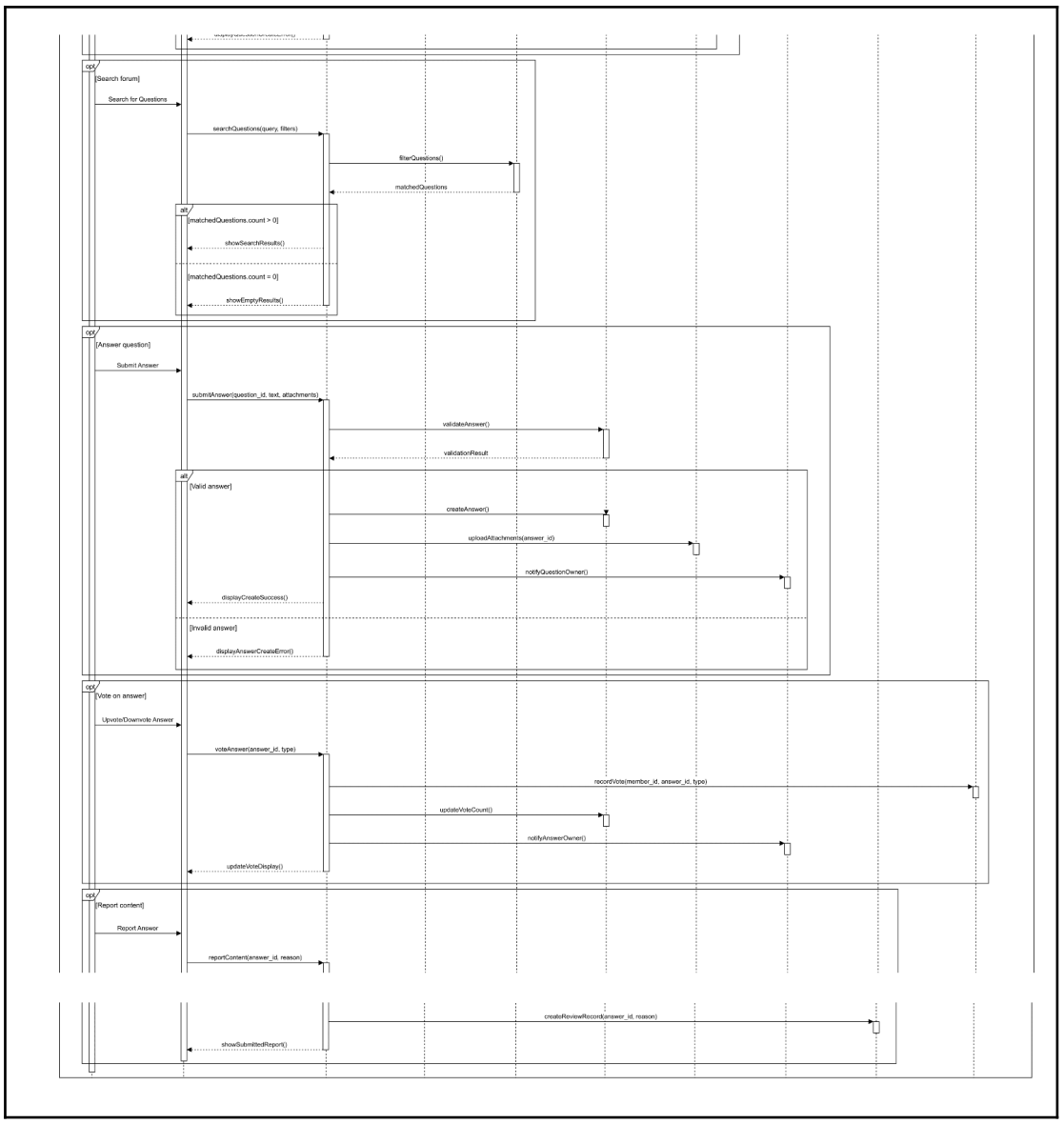
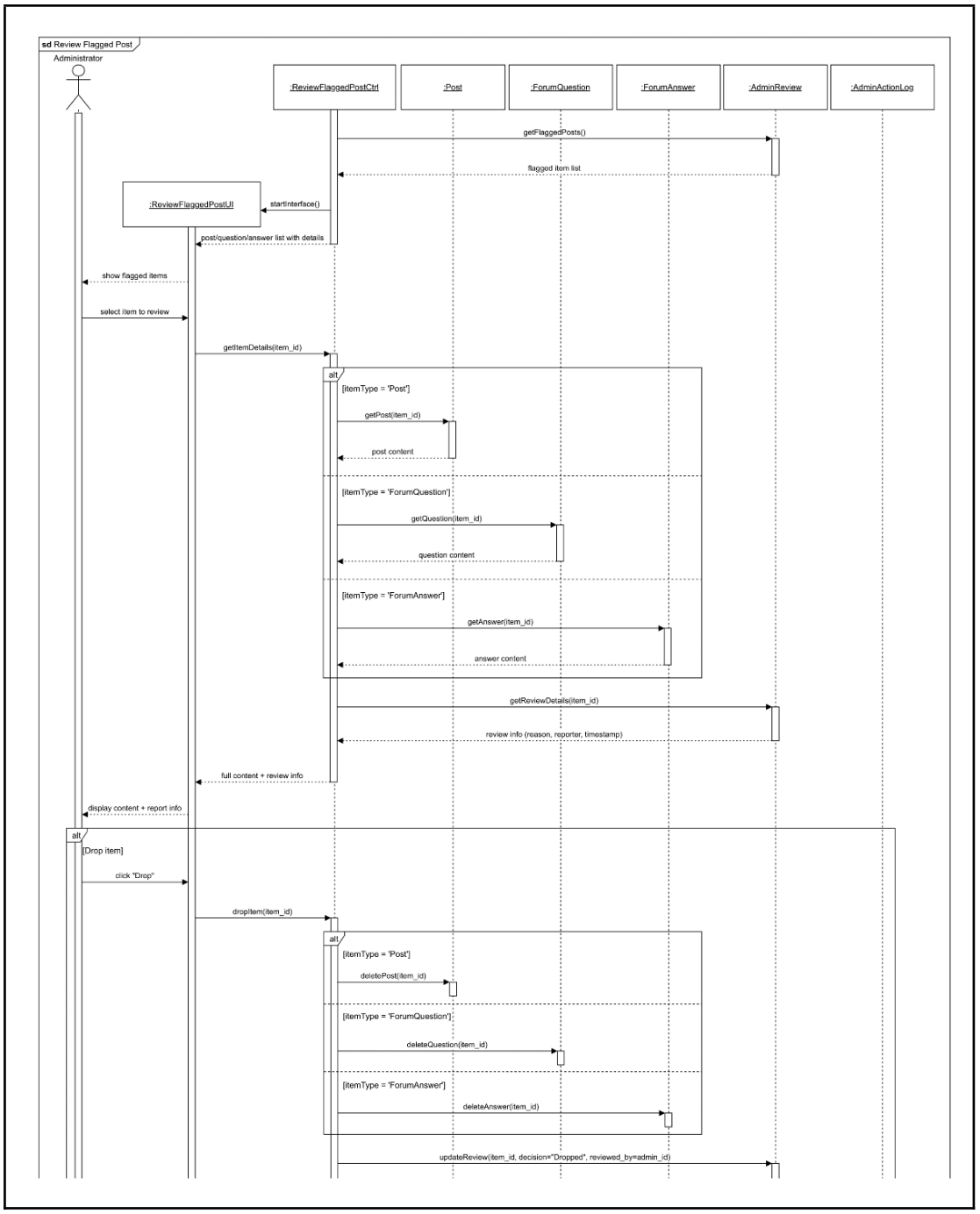


Diagram 4.1.13: Sequence Diagram - Manage Forum (Anonymous Community Module)

Review Flagged Post



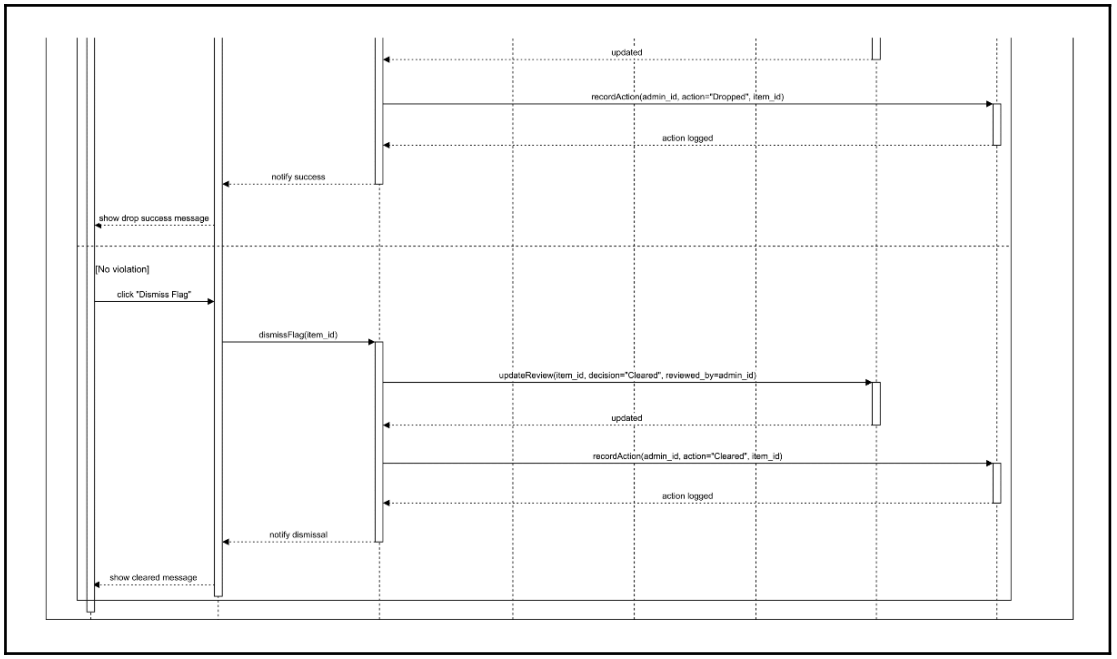


Diagram 4.1.14: Sequence Diagram - Review Flagged Post (Anonymous Community Module)

Review Flagged Forum Question

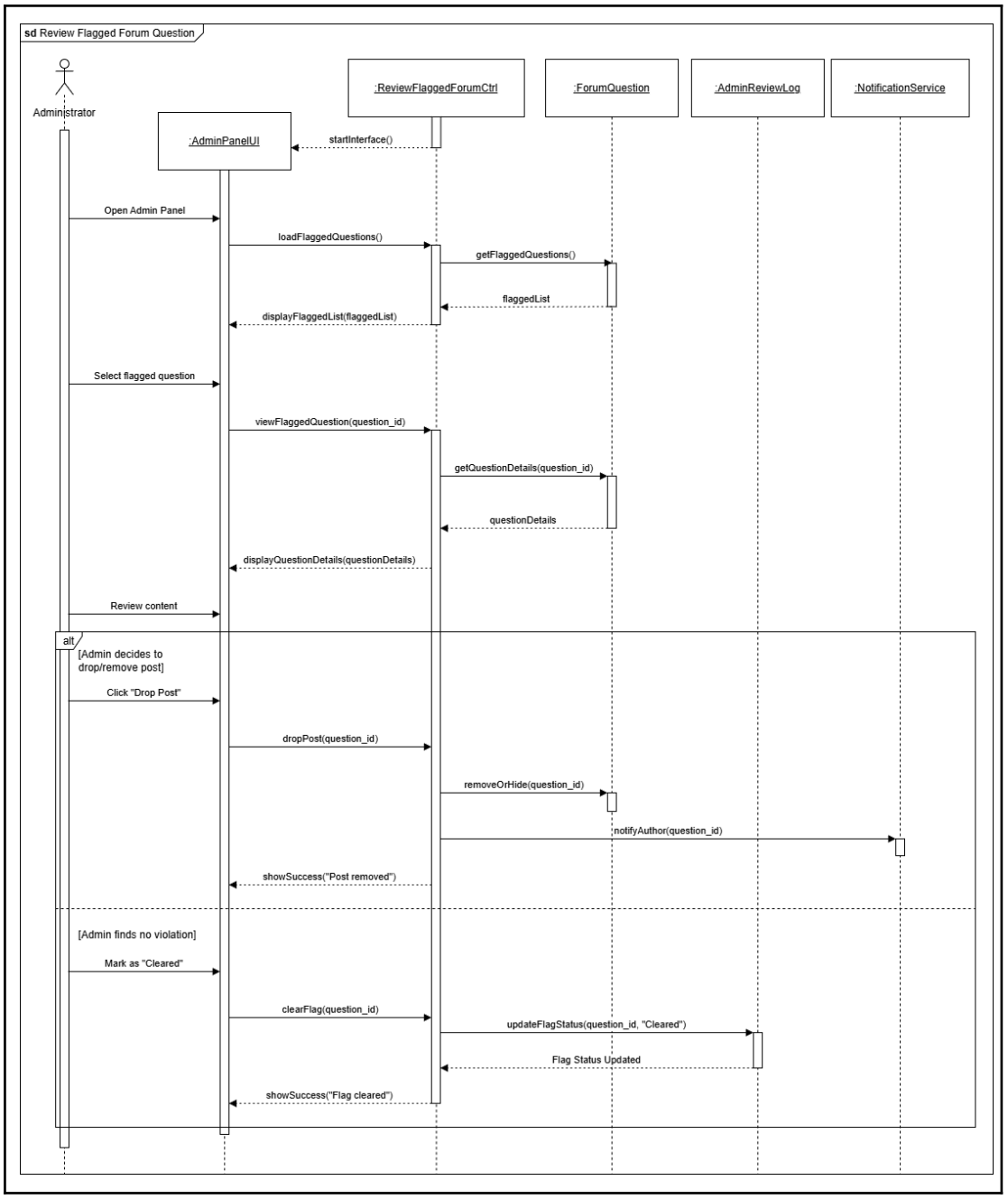
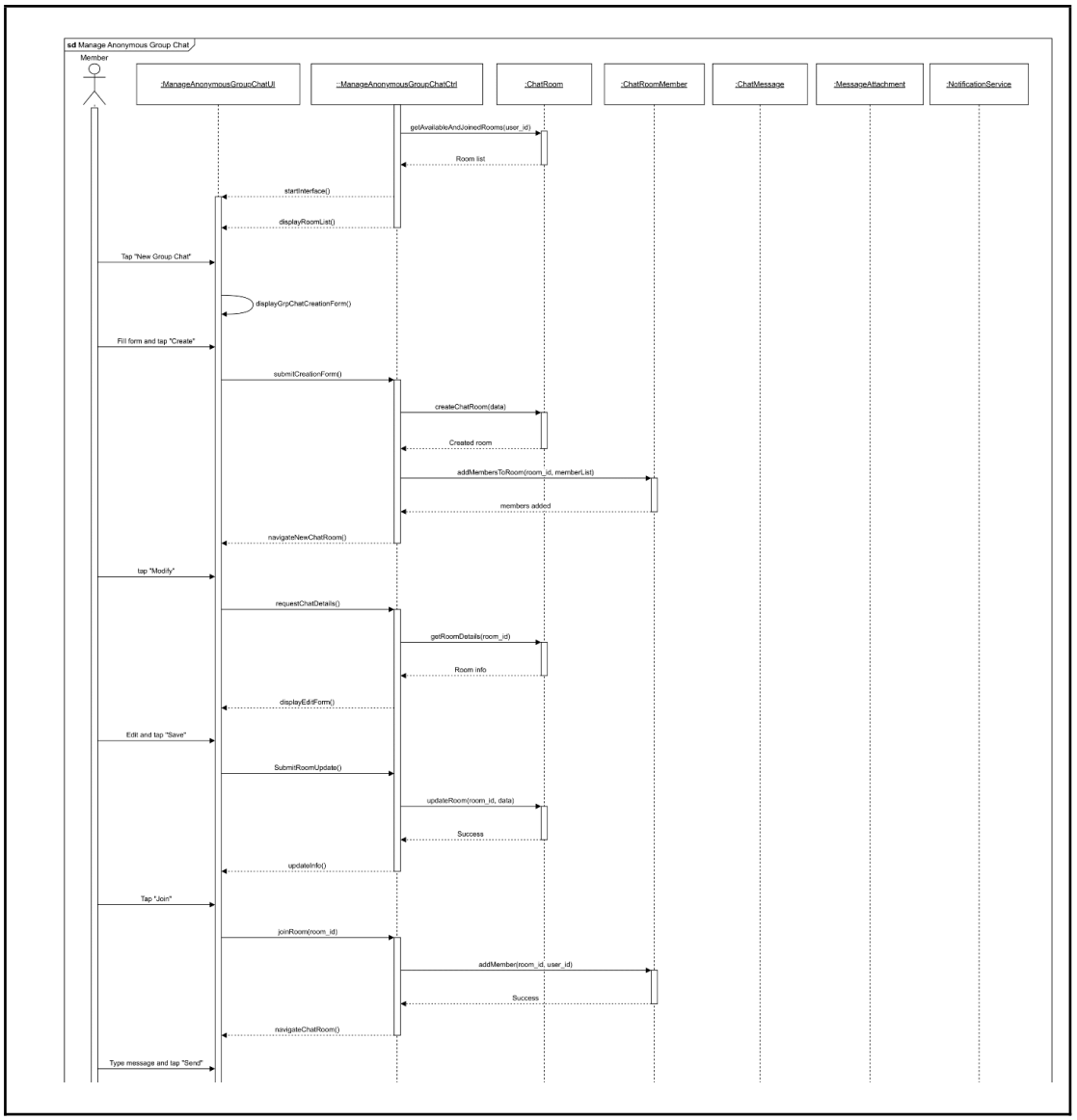


Diagram 4.1.15: Sequence Diagram - Review Flagged Forum Question (Anonymous Community Module)

Manage Anonymous Group Chat



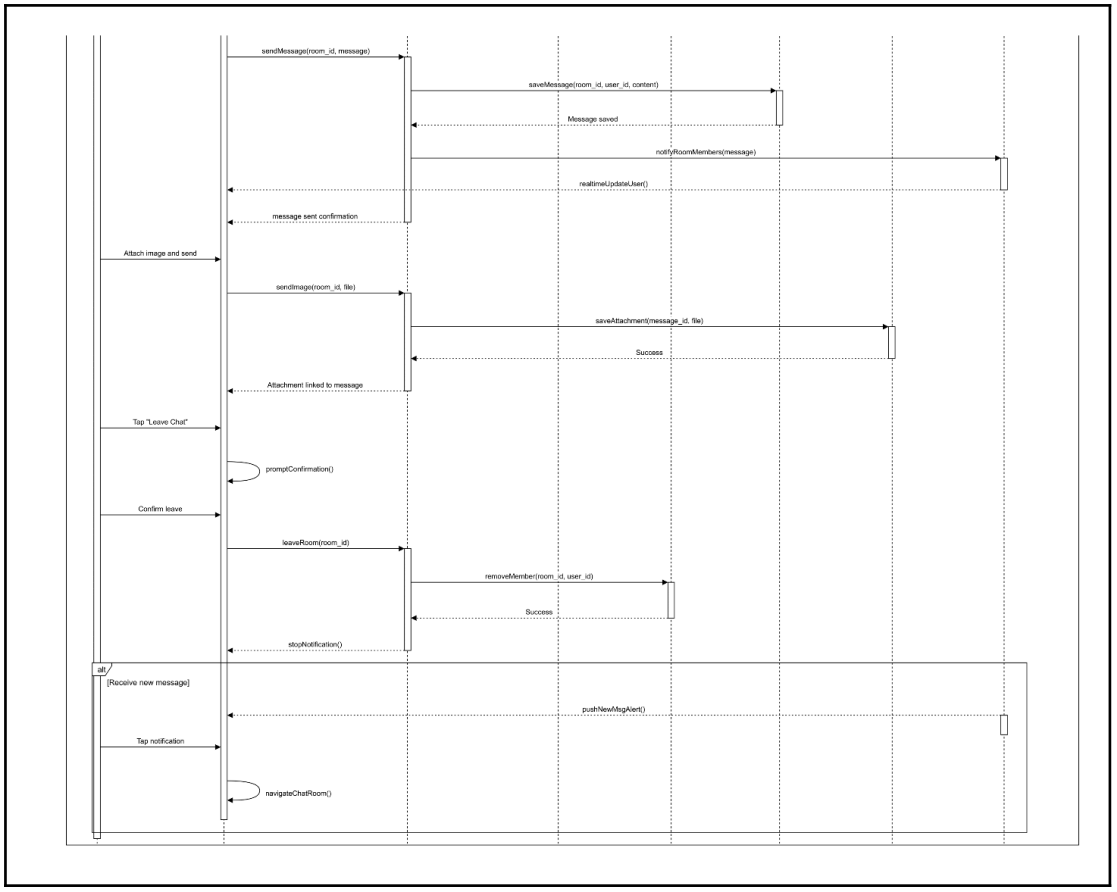


Diagram 4.1.16: Sequence Diagram - Manage Anonymous Group Chat (Anonymous Community Module)

Personal Chat Module

Send Friend Request

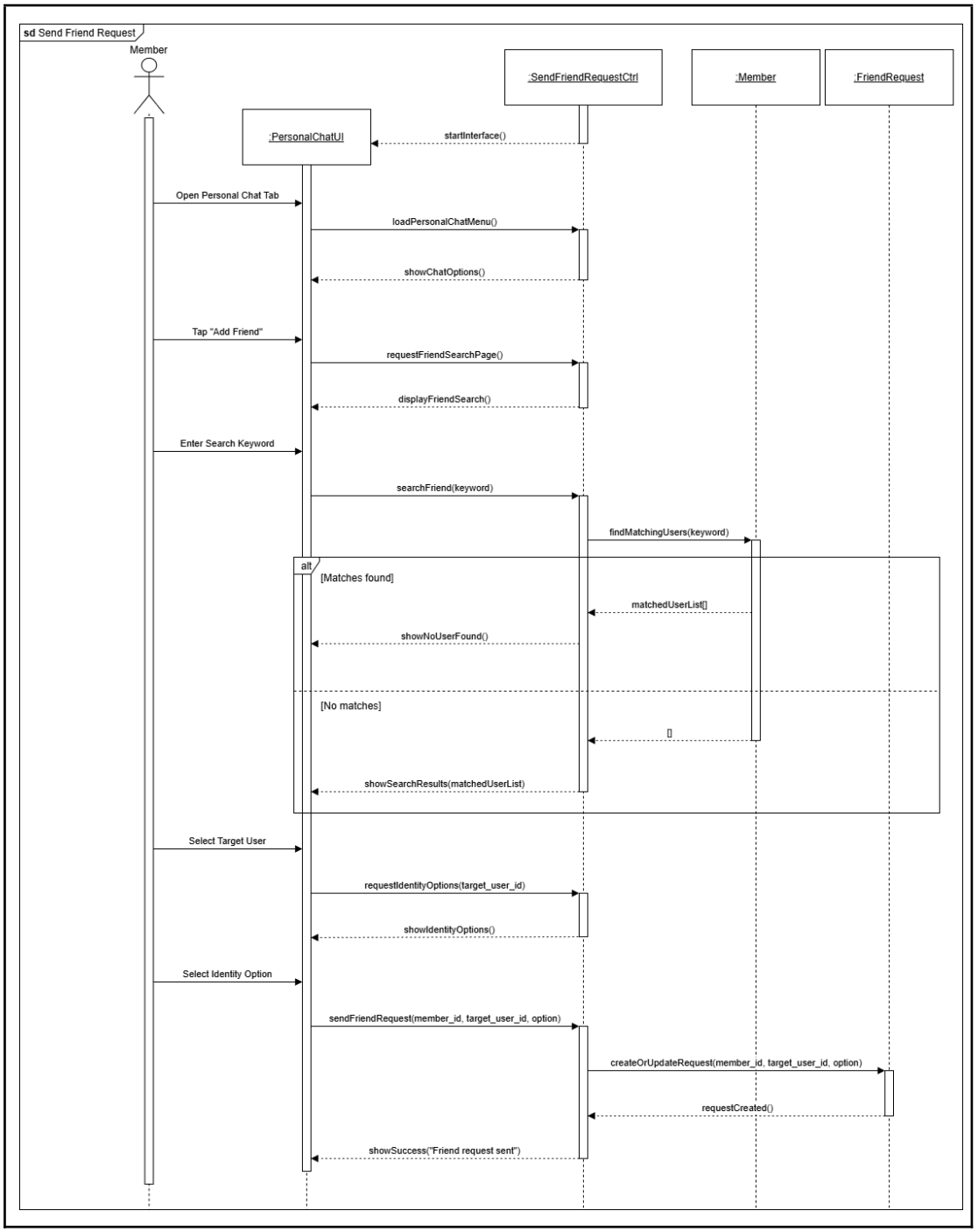
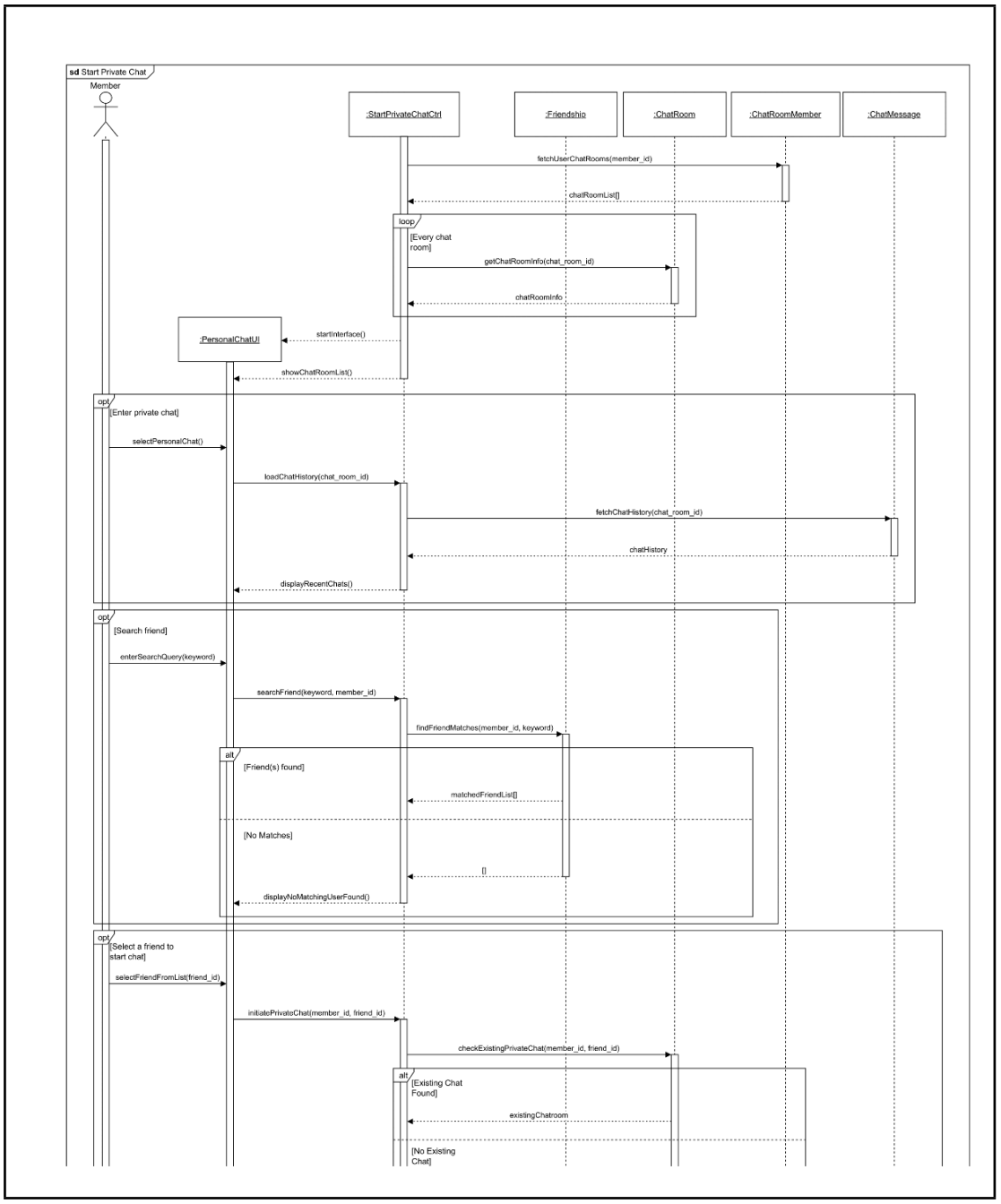


Diagram 4.1.17: Sequence Diagram - Send Friend Request (Personal Chat Module)

Start Private Chat



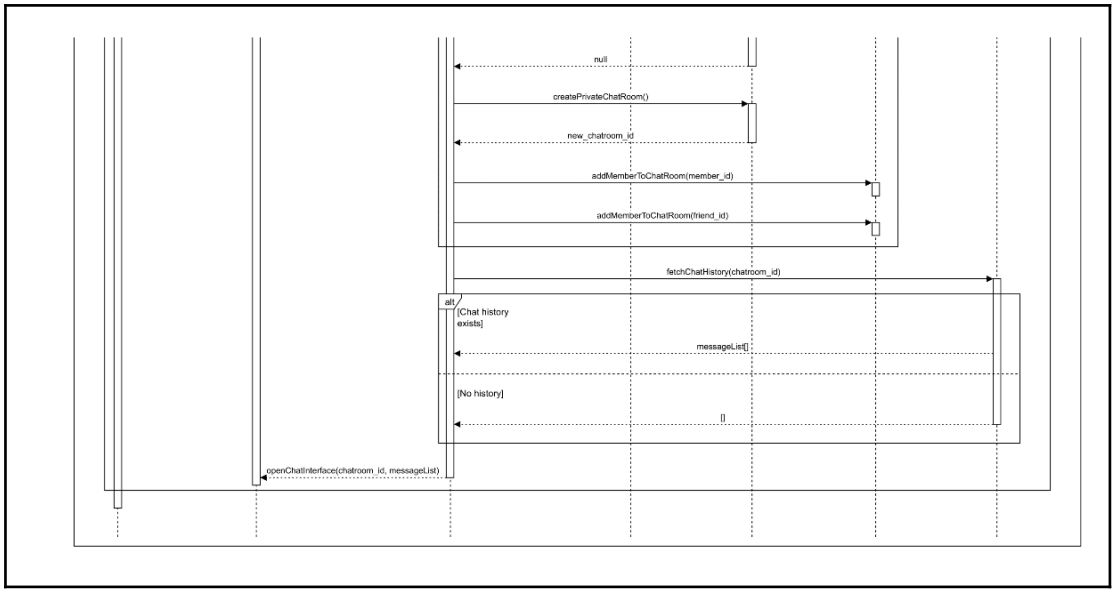


Diagram 4.1.18: Sequence Diagram - Start Private Chat (Personal Chat Module)

Send Message

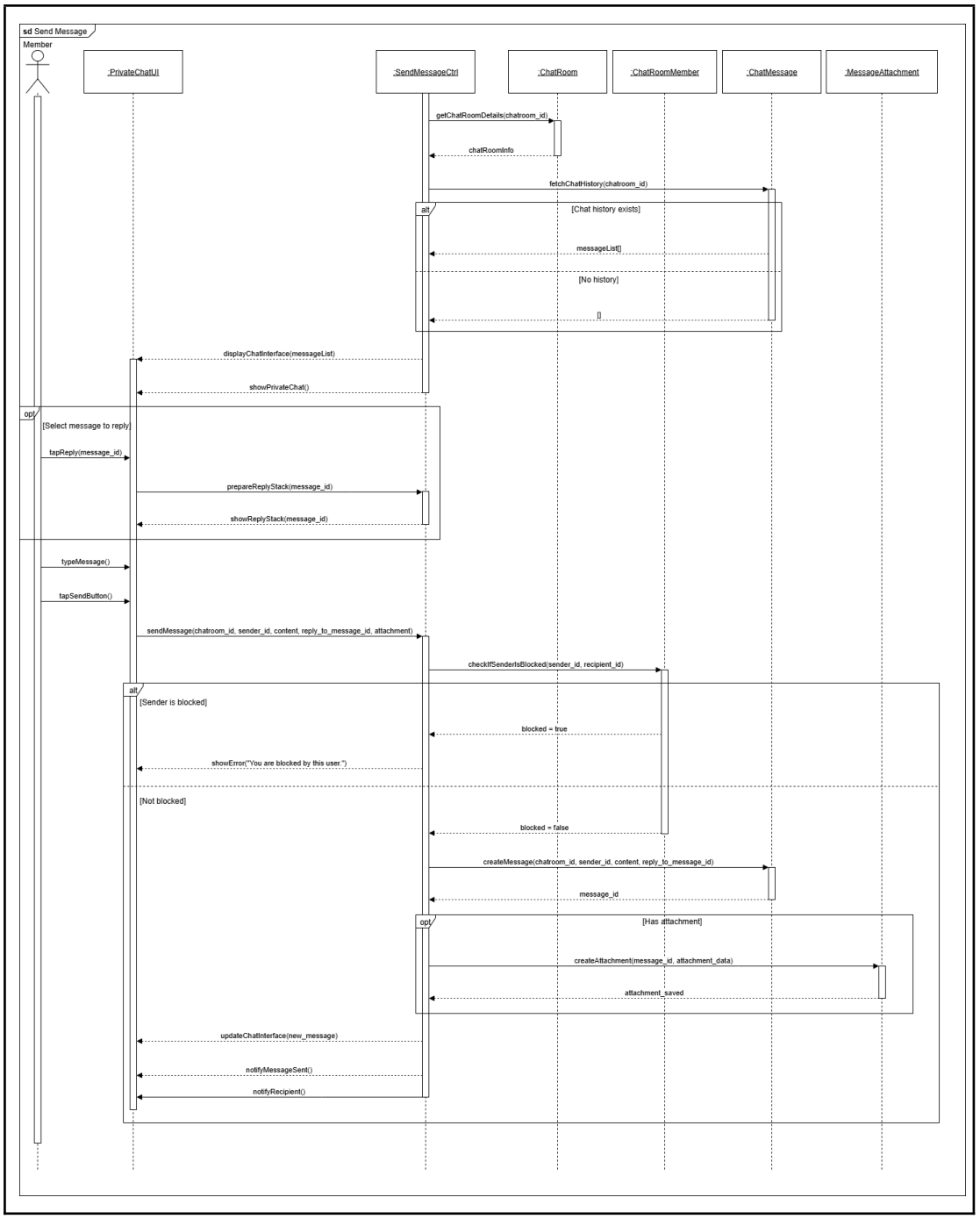


Diagram 4.1.19: Sequence Diagram - Send Message (Personal Chat Module)

Receive Notification

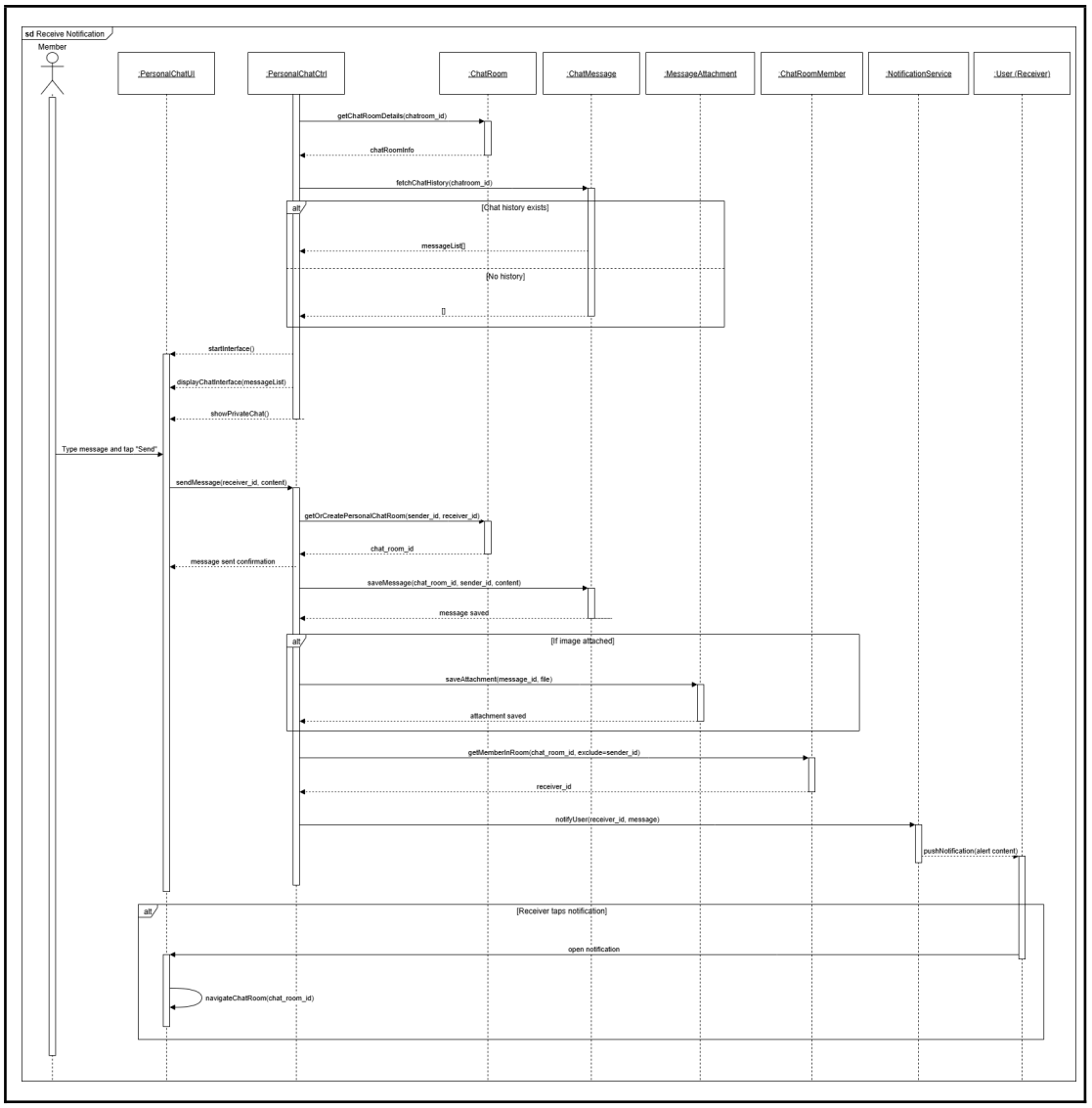


Diagram 4.1.20: Sequence Diagram - Receive Notification (Personal Chat Module)

View User

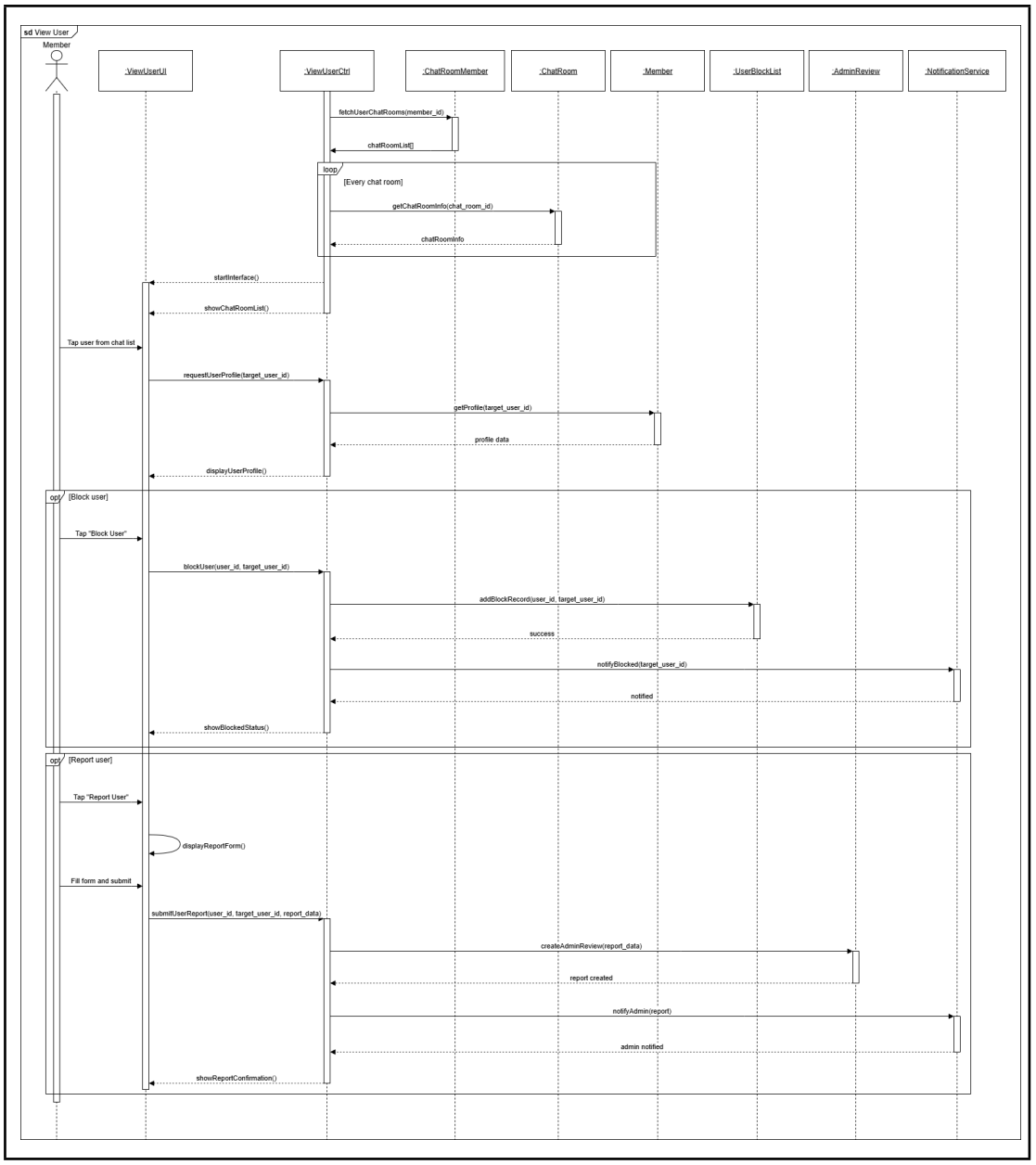


Diagram 4.1.21: Sequence Diagram - View User (Personal Chat Module)

Set Chat Identity Preference

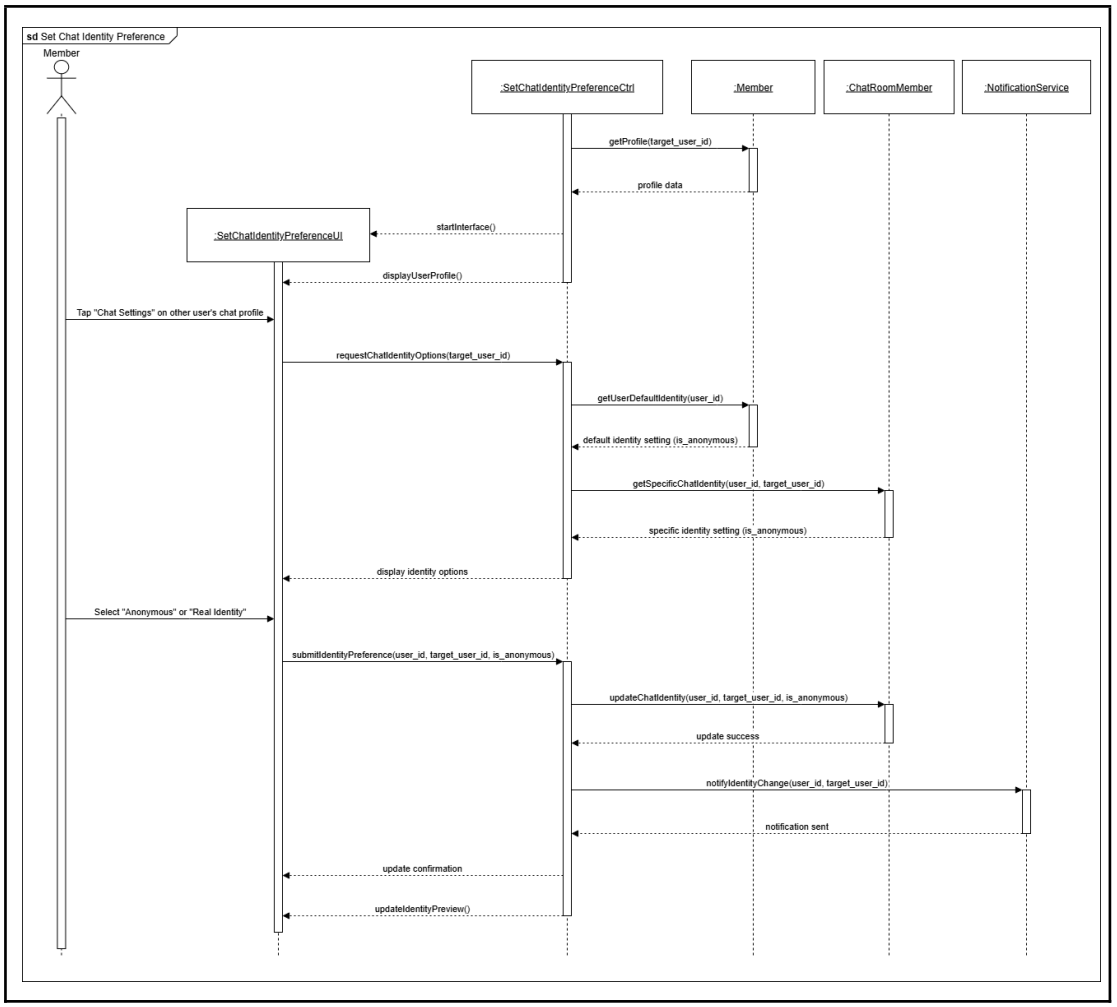


Diagram 4.1.22: Sequence Diagram - Set Chat Identity Preference (Personal Chat Module)

Review User

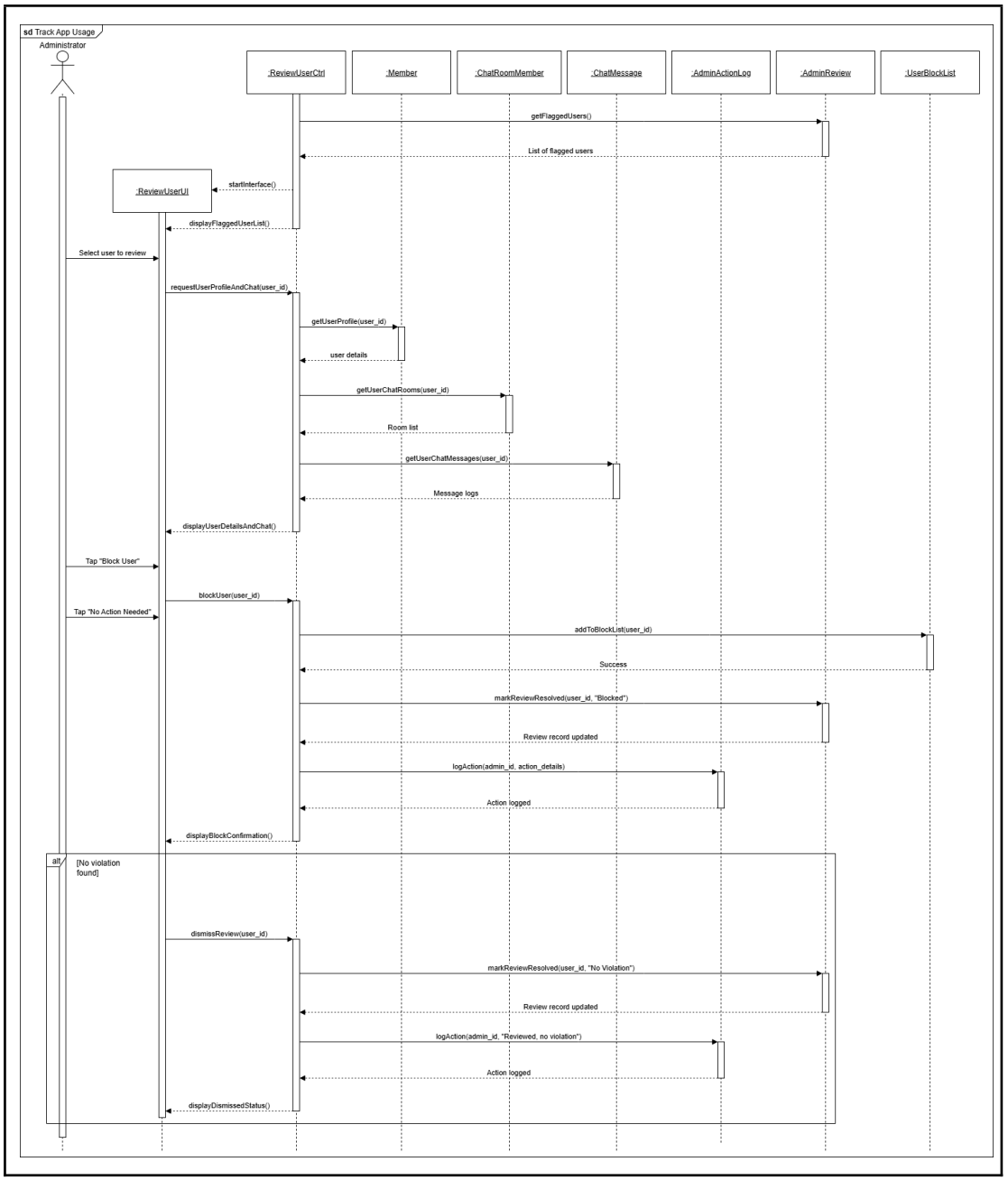


Diagram 4.1.23: Sequence Diagram - Review User (Personal Chat Module)

Review Reported Personal Chat Messages

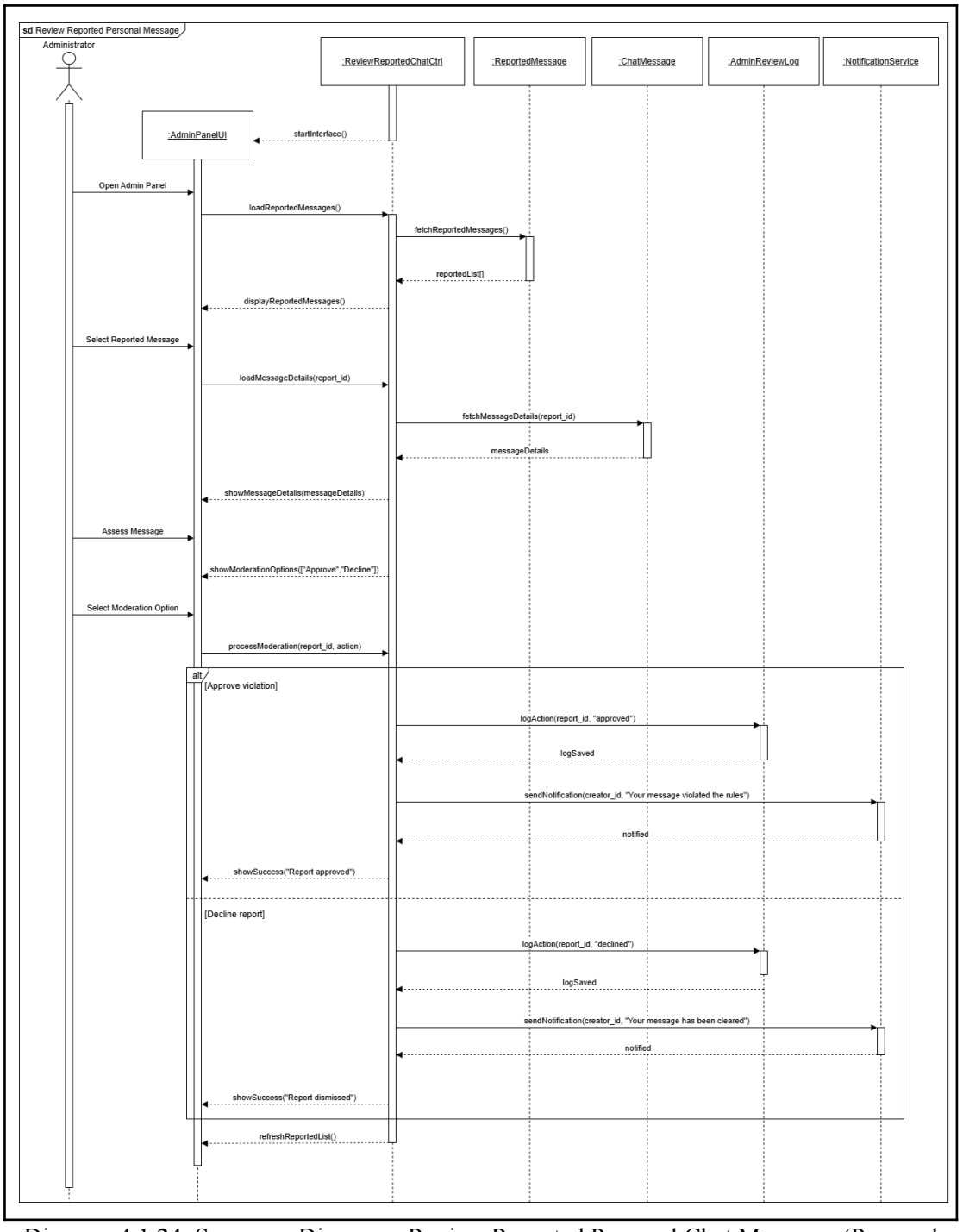


Diagram 4.1.24: Sequence Diagram - Review Reported Personal Chat Messages (Personal Chat Module)

Report Module

View Productivity Report

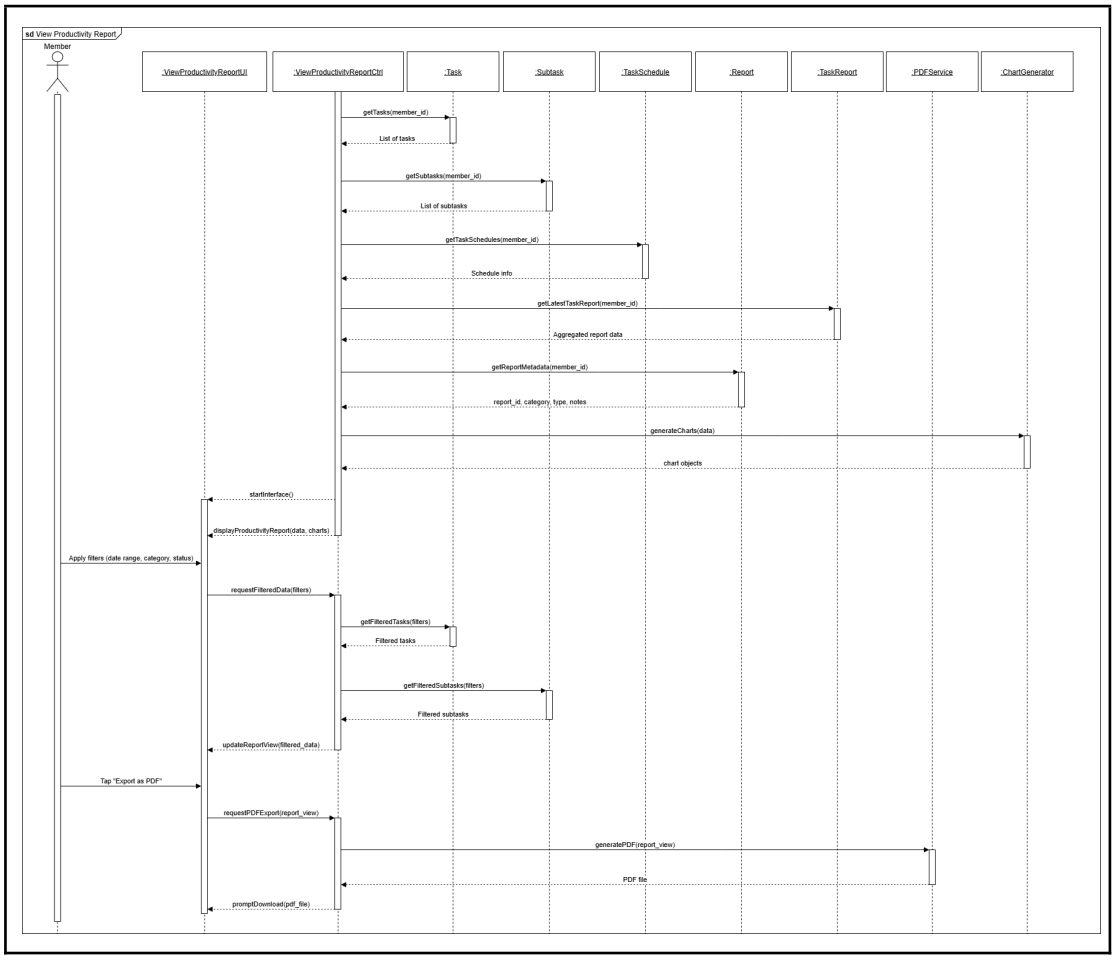


Diagram 4.1.25: Sequence Diagram - View Productivity Report (Report Module)

View Social Performance Report

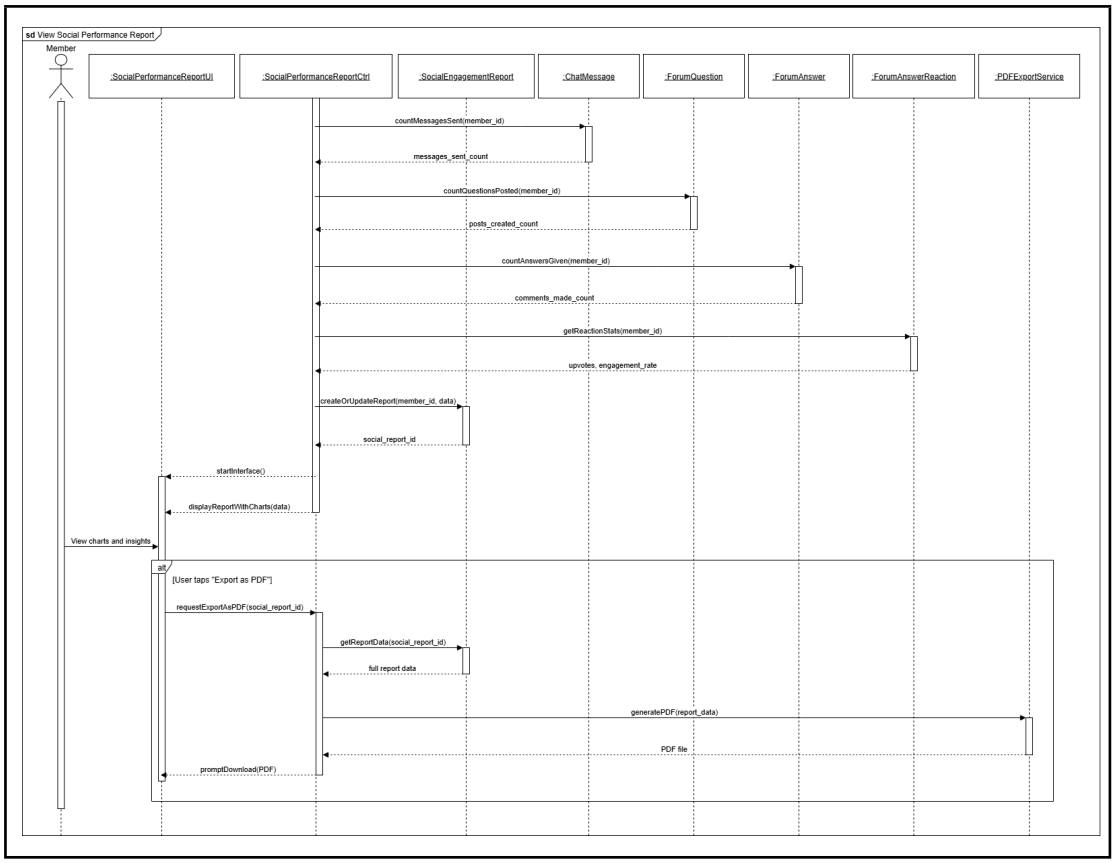


Diagram 4.1.26: Sequence Diagram - View Social Performance Report (Report Module)

Receive Suggestion

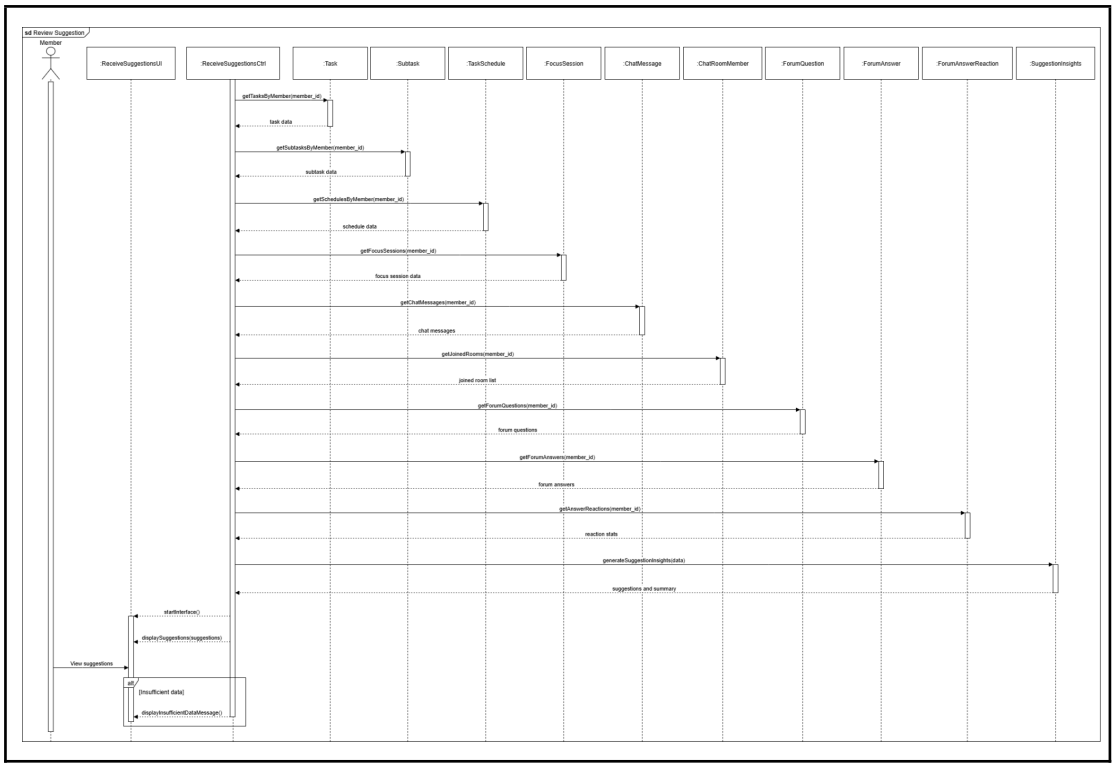


Diagram 4.1.27: Sequence Diagram - Receive Suggestion (Report Module)

4.2 State Chart Diagram

Task

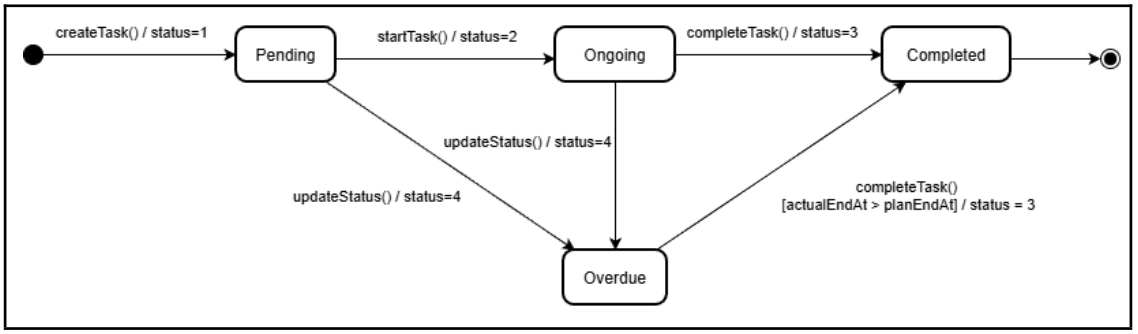


Diagram 4.2.1: State Chart Diagram - Task

TaskAdvisorLog

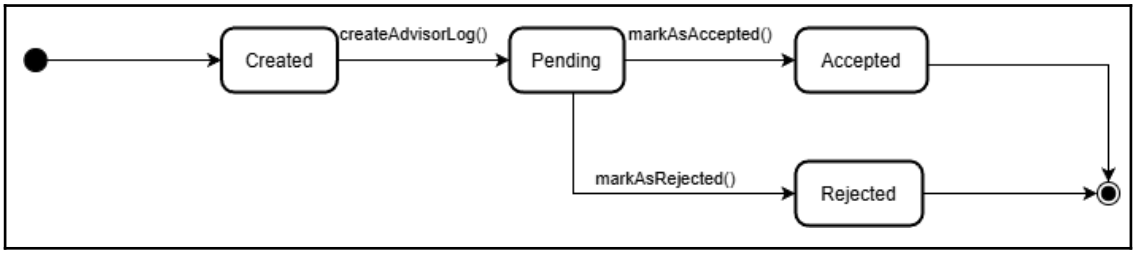


Diagram 4.2.2: State Chart Diagram - TaskAdvisorLog

NFCTag

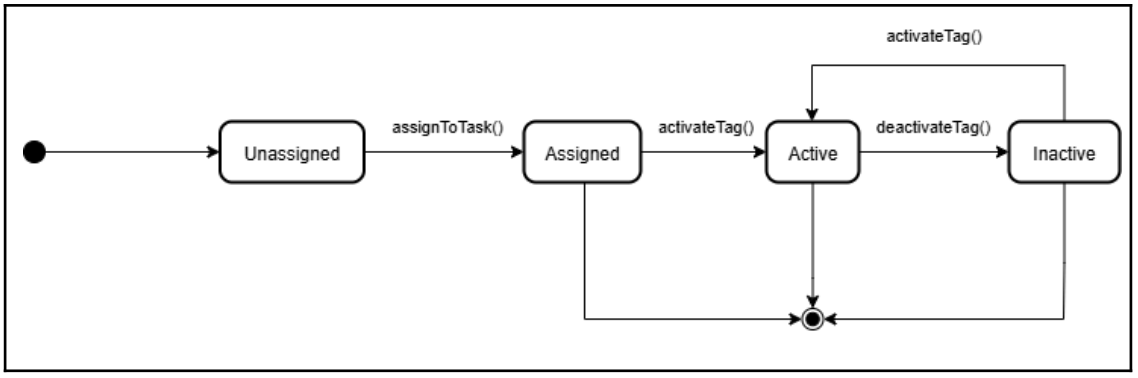


Diagram 4.2.3: State Chart Diagram - NFCTag

Community

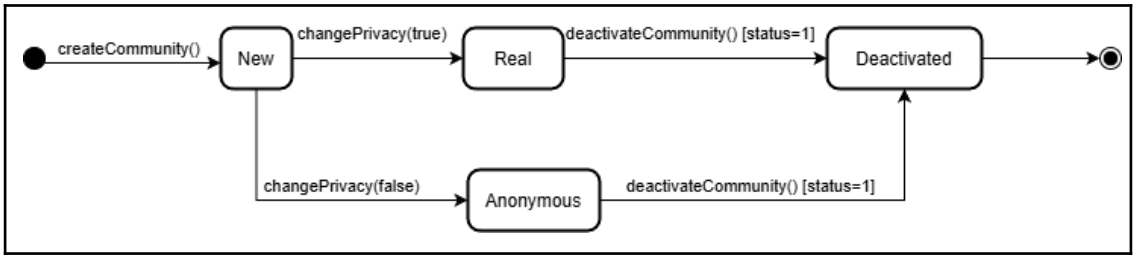


Diagram 4.2.4: State Chart Diagram - Community

Post

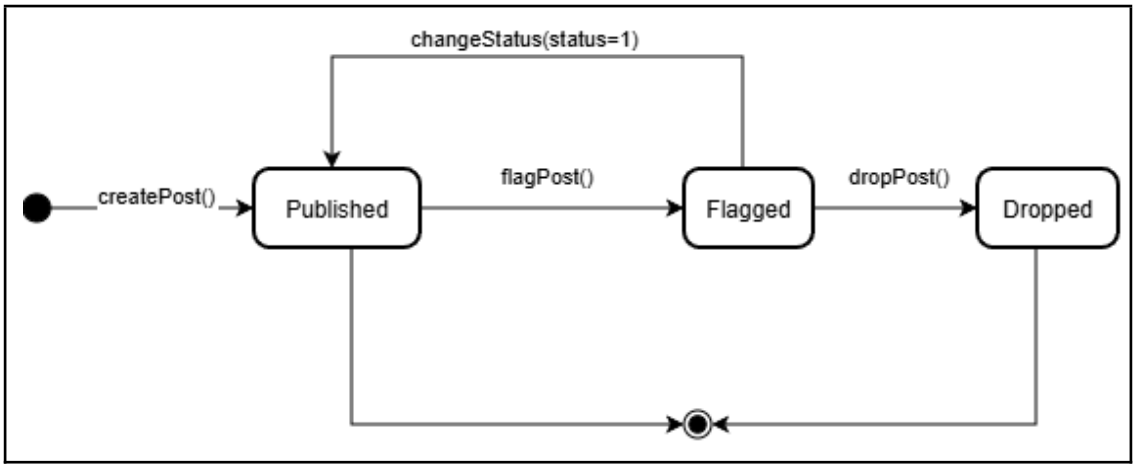


Diagram 4.2.5: State Chart Diagram - Post

ForumQuestion

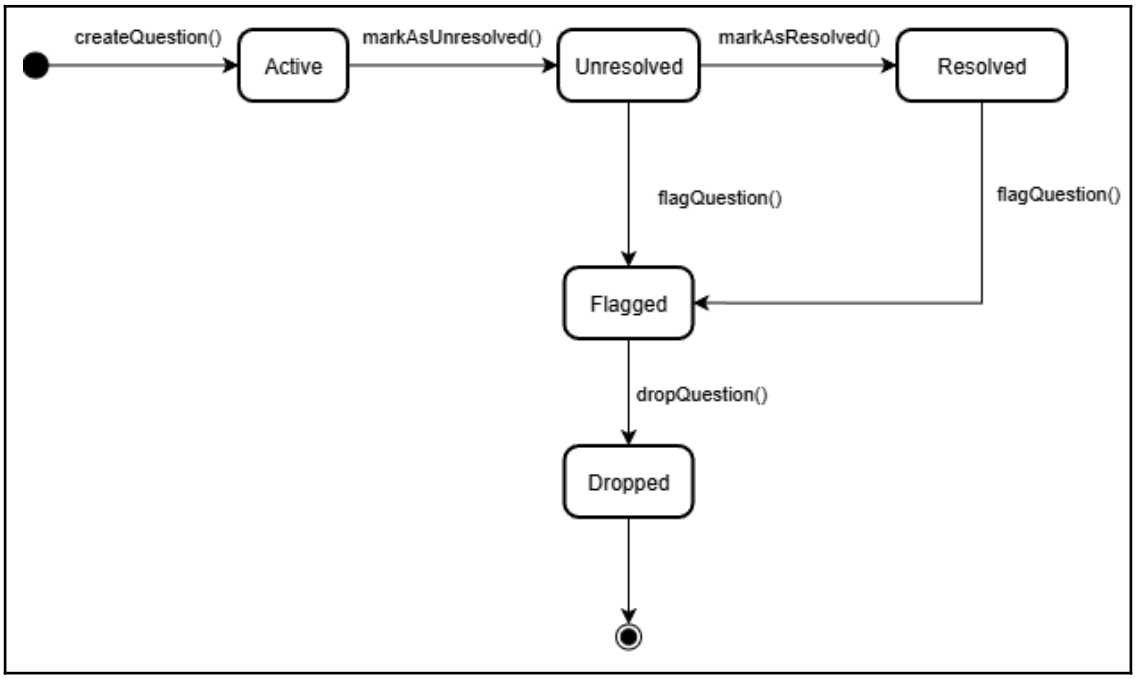


Diagram 4.2.6: State Chart Diagram - ForumQuestion

ForumAnswer

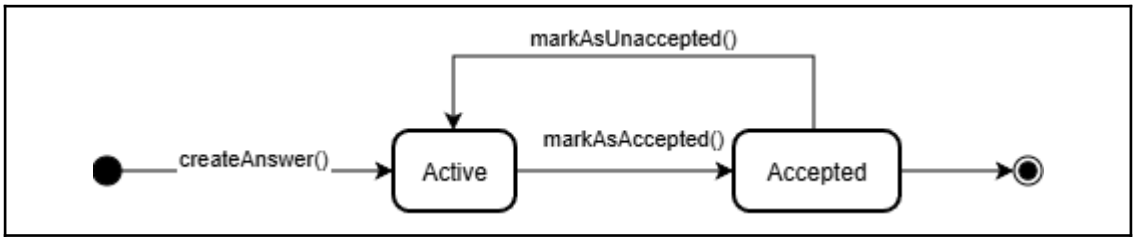


Diagram 4.2.7: State Chart Diagram - ForumAnswer

AdminModeration

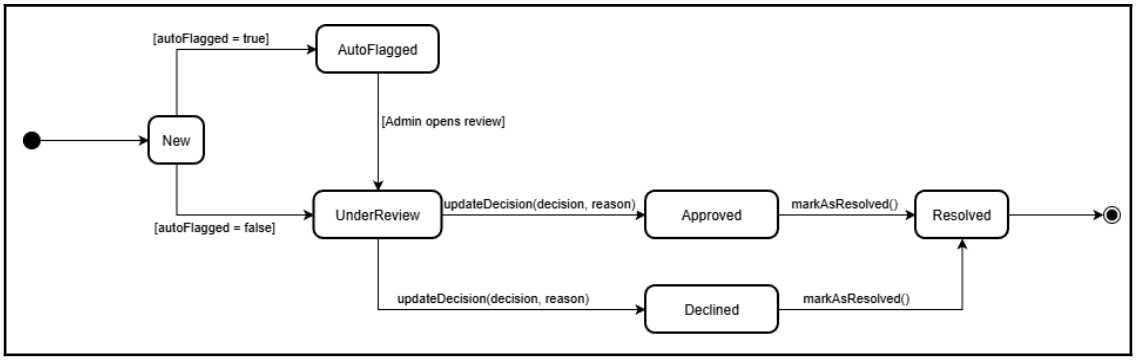


Diagram 4.2.8: State Chart Diagram - AdminModeration

Friendship

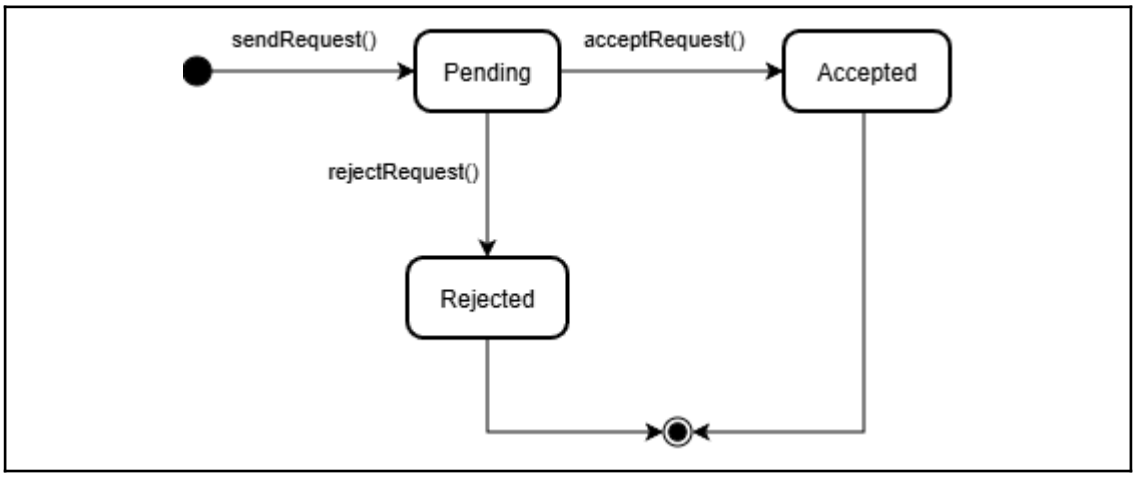
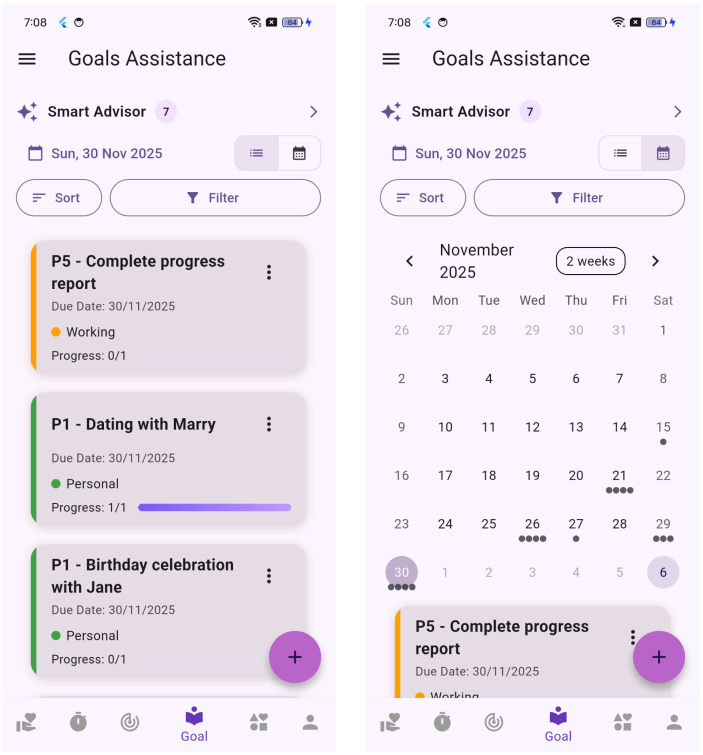


Diagram 4.2.9: State Chart Diagram - Friendship

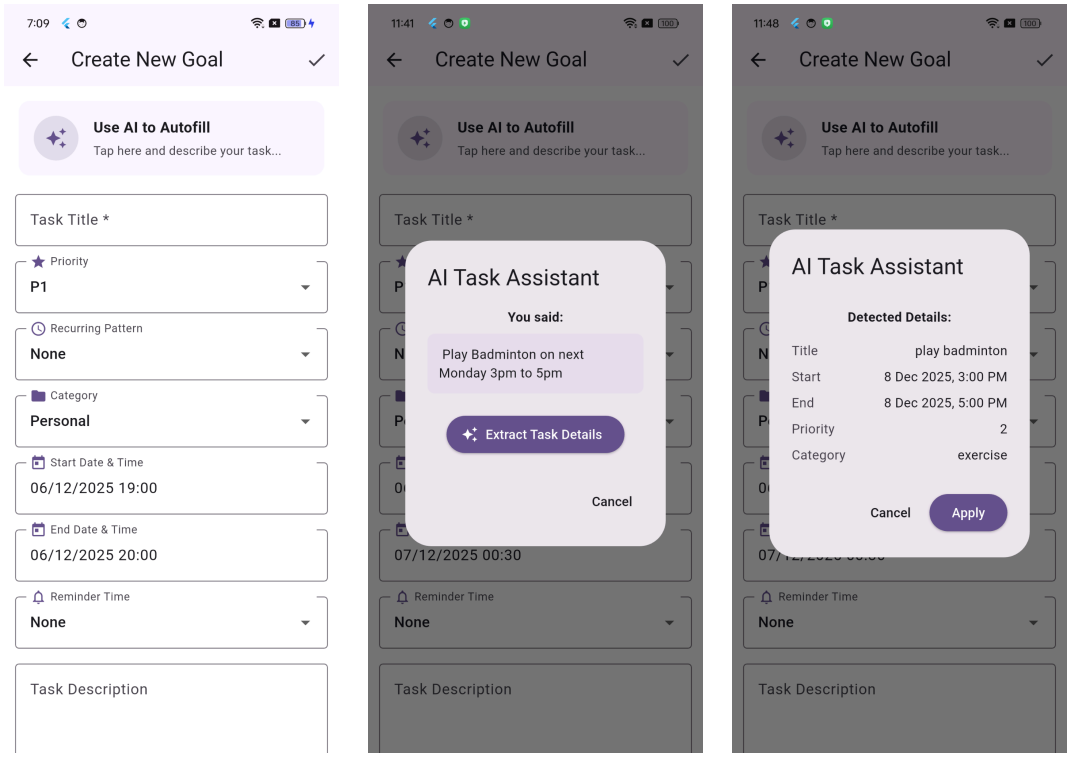
4.3 User Interface Design

Goal Assistance Module



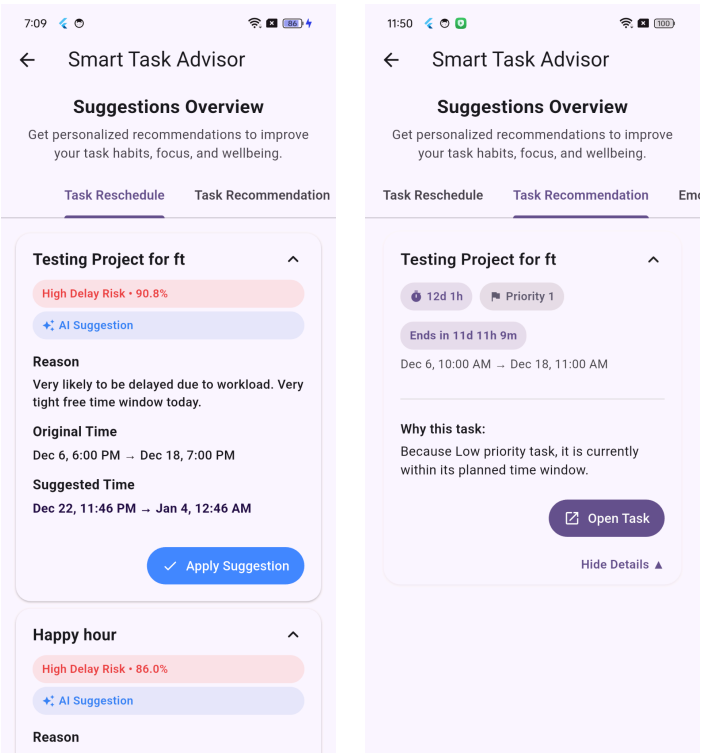
Goal Listing

- The Card View shows each goal as a clean with its title, category, due date and progress. This helps the users to quickly scan and manage their daily tasks.
- The Calendar View shows all goals based on their dates. This allows the users to check when the tasks are due and plan their schedule more easily.



Goal Creation View

- The first screen is the Goal Creation Form where users can manually fill in details such as title, priority, category, start and end date time, subtasks and other relevant info.
- The second and third screen show the AI Task Assistant which enables users to speak their task, automatically transcribe it into text, and then extract important details like title, start datetime, end date time, priority and category. Thus, users can apply these extracted details directly to the form immediately.

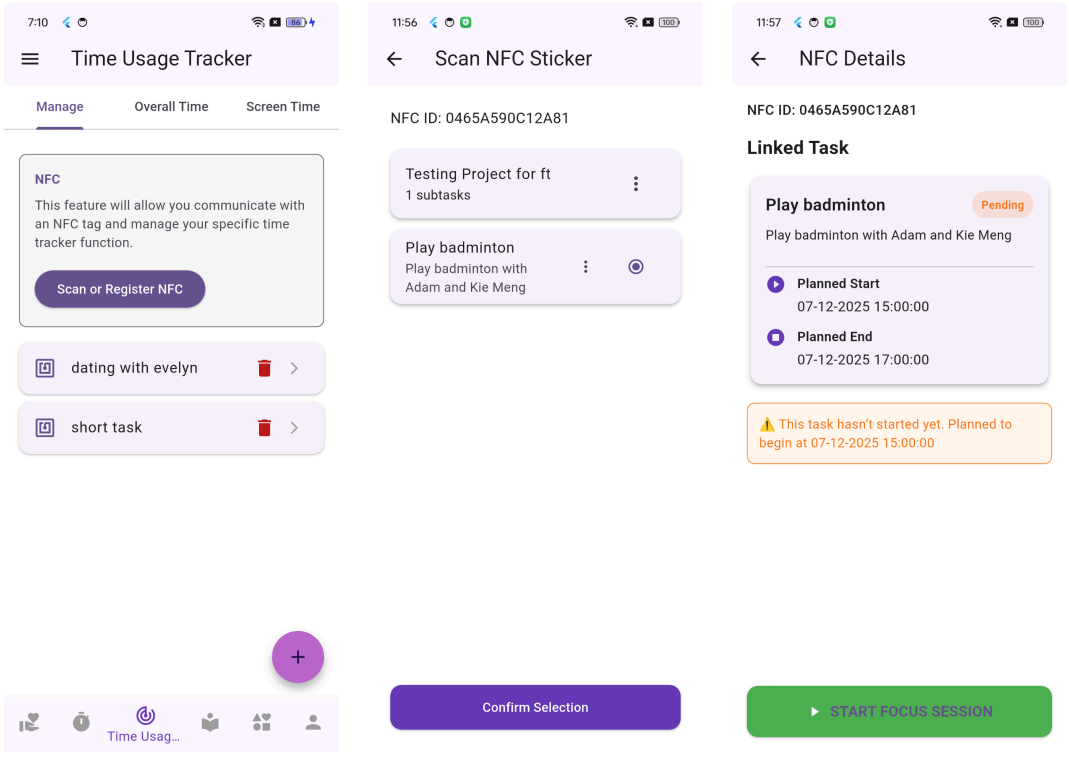


Task Reschedule & Recommendation

- The Task Reschedule view analyzes each task and identifies those that are at risk of being delayed. It explains the reason for the delay risk, shows the original schedule and recommends a new date and time. Users can choose whether to apply the suggested schedule.
- The Task Recommendation view recommends which tasks the user should handle first. The system highlights the priority level, time window and gives a clear reason for why the task is recommended at that moment.

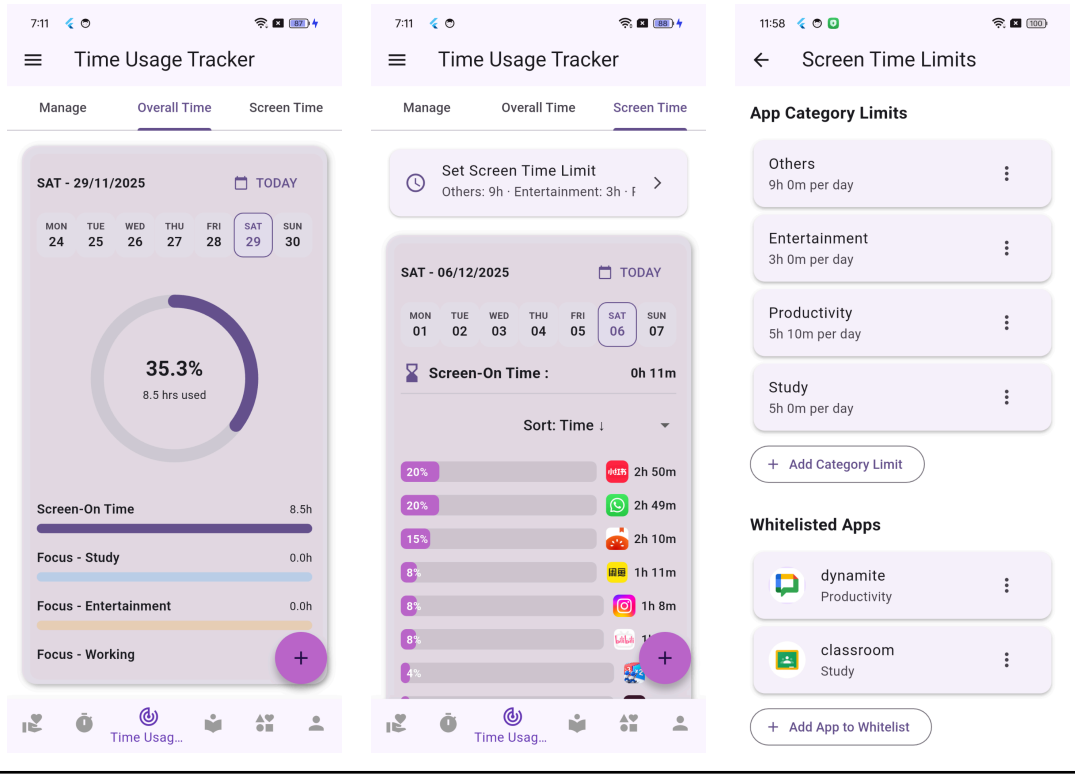
Table 4.3.1: User Interface Design - Goal Assistance Module

Time Usage Tracker Module



NFC Management

- The first screen displays all the tasks or subtasks that are already linked to an NFC sticker. Users can scan a new sticker or register a new one using the "Scan or Register NFC" button.
- After scanning an NFC sticker, users can choose which existing task or subtask they want to link to in the second screen.
- The third screen shows the linked task details, including planned start and end time. The user can start a focus session using the bottom "Start Focus Session" button once the task is valid to begin.

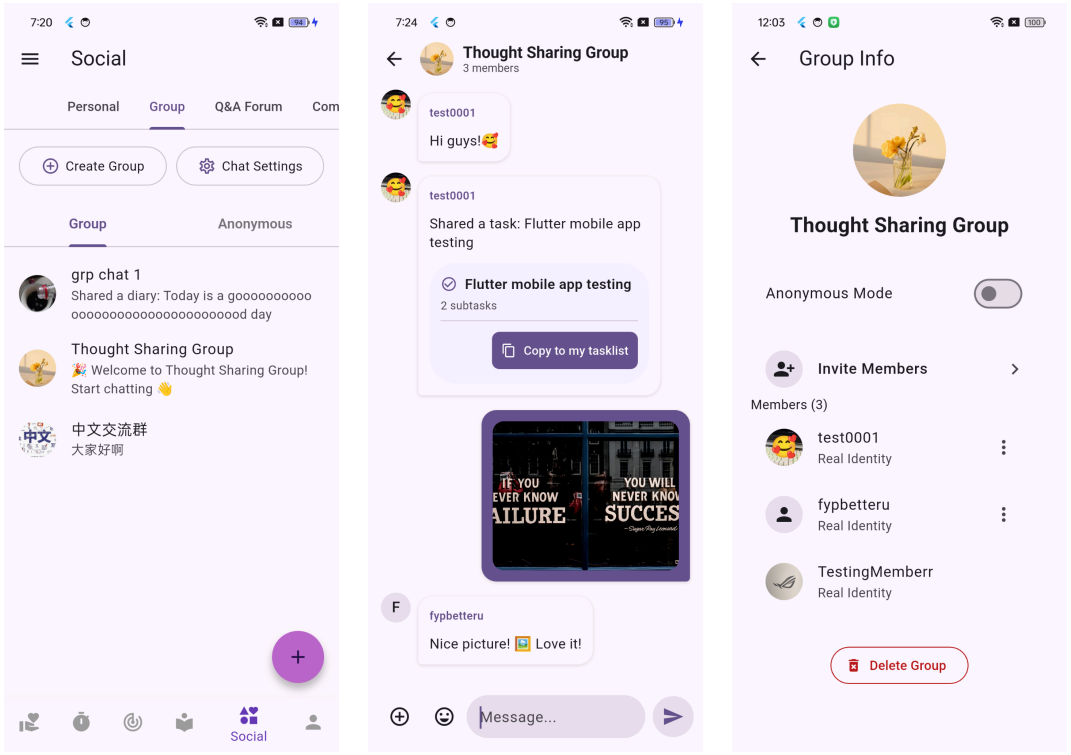


Overall Time Usage & App Category Limit

- The first and second screen shows the daily screen-on time and focus session duration while also showing the visual breakdown usage by individual apps.
- The third screen allows users to set the usage limits for app categories and whitelist apps to exclude them from the limits.

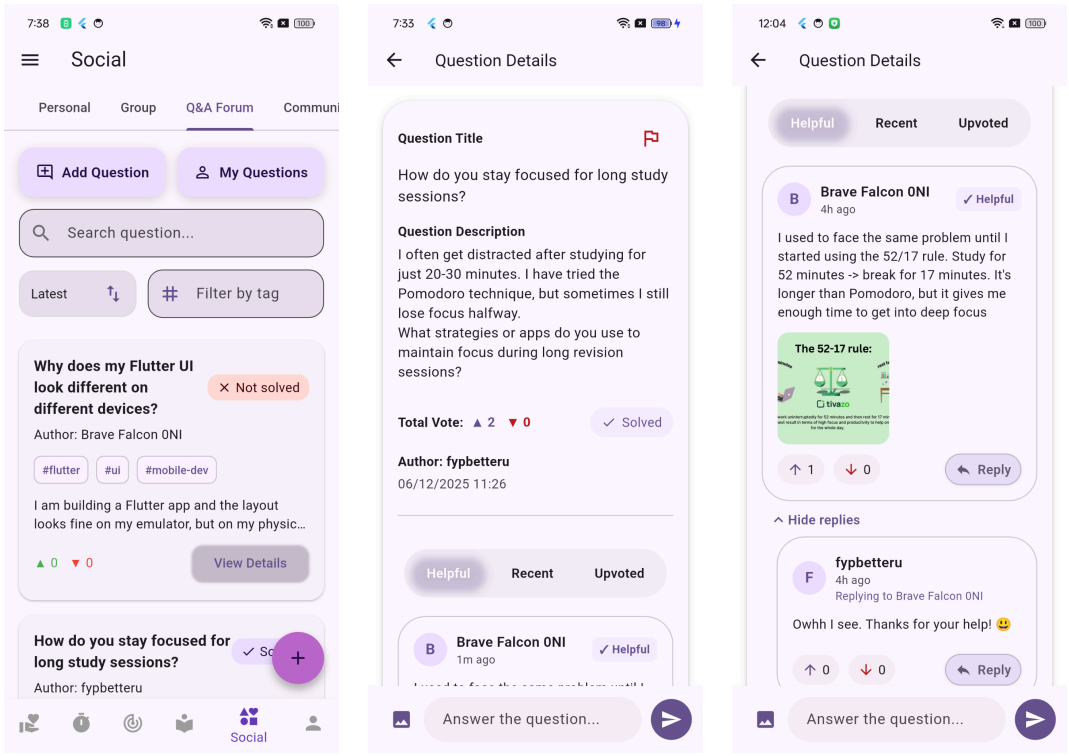
Table 4.3.2: User Interface Design - Time Usage Tracker Module

Anonymous Community Module



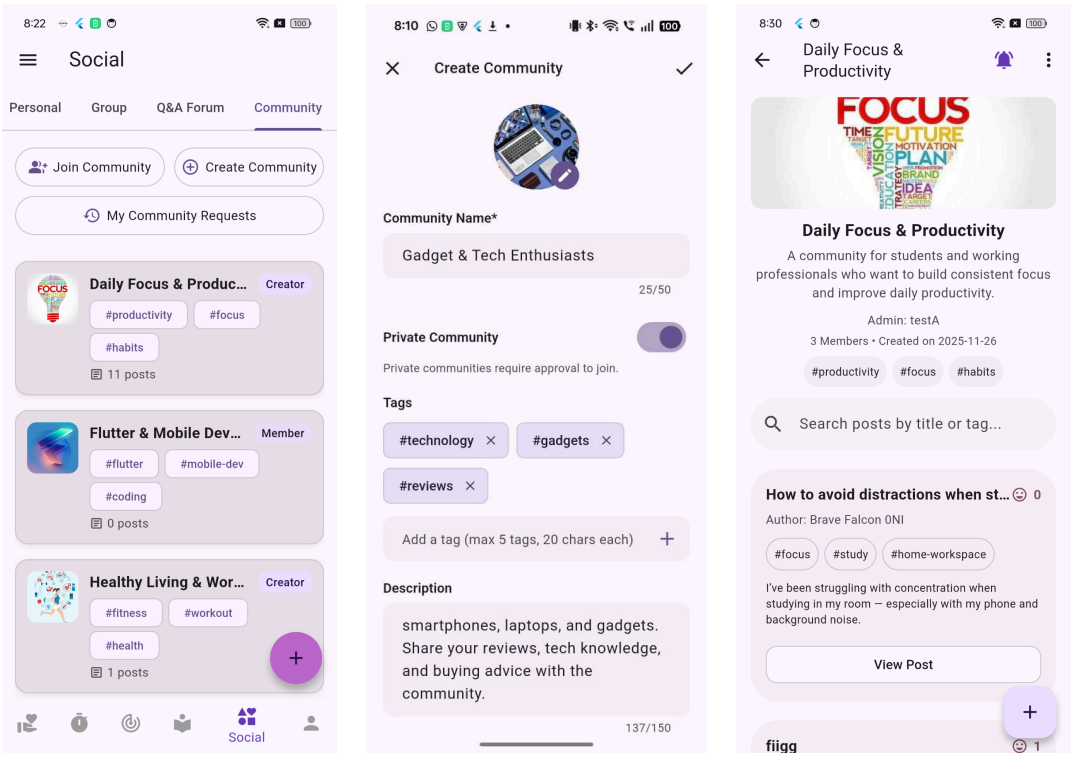
Group Chat

- The first screen displays all group chats the user has joined, including ongoing conversations and newly created groups. Users can create new group or adjust chat settings
- The second screen shows the full chat history and allows users to send messages, emojis, images and also share tasks, posts or diaries within the group. Messages from other members are also displayed here.
- The third screen provides management options for the group creator, including switching Anonymous Mode, inviting members, removing members or deleting the group.



Forum

- The first screen displays all posted questions with tags, vote counts, author name and solve status. Users can add new questions or filter questions by tags or sorting options.
- The second and third screen shows the full question content including its title, description, total votes, author information and posted date. Users can view all answers, vote on helpful responses and check the reply threads. The forum also supports answering with text, images and replying to existing answers.

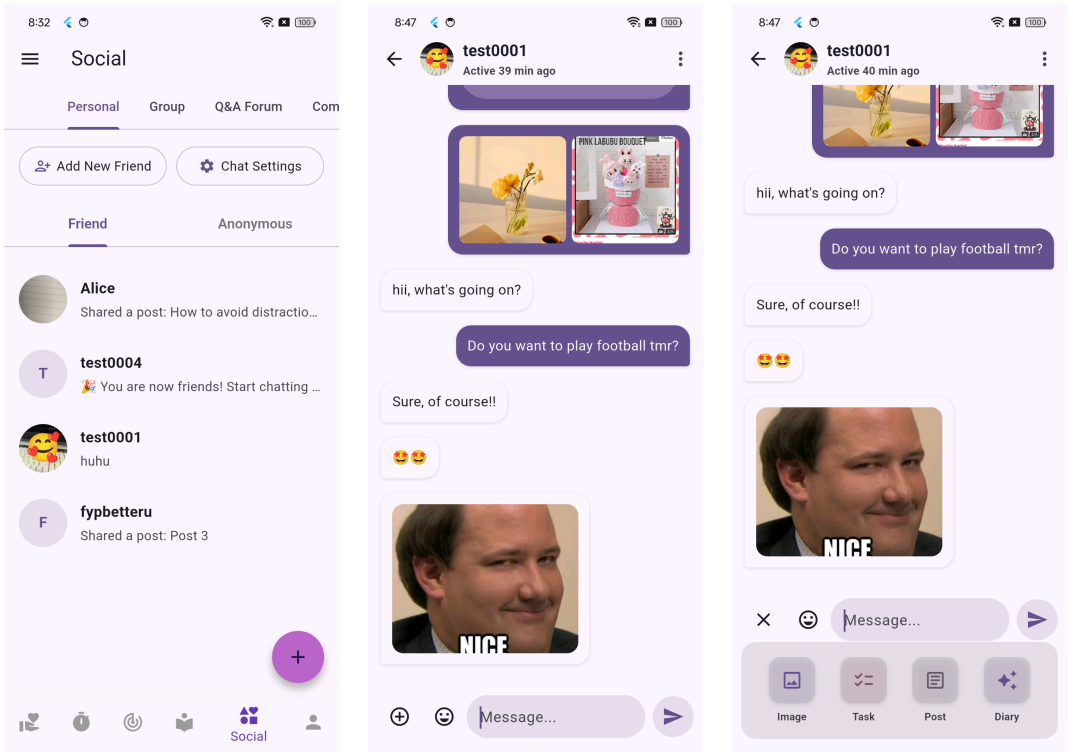


Community

- The first screen shows all communities the user has joined, including each community's name, tags, and the user's role. Users can also join or create new communities and view their pending requests.
- The second screen is the community creation form, where users upload an avatar, enter the community name, select privacy, add tags, and write a description before submitting it for admin review.
- The third screen shows the selected community's details, such as its description, admin, member count, and tags. Users can browse and search posts, subscribe, and interact within the community.

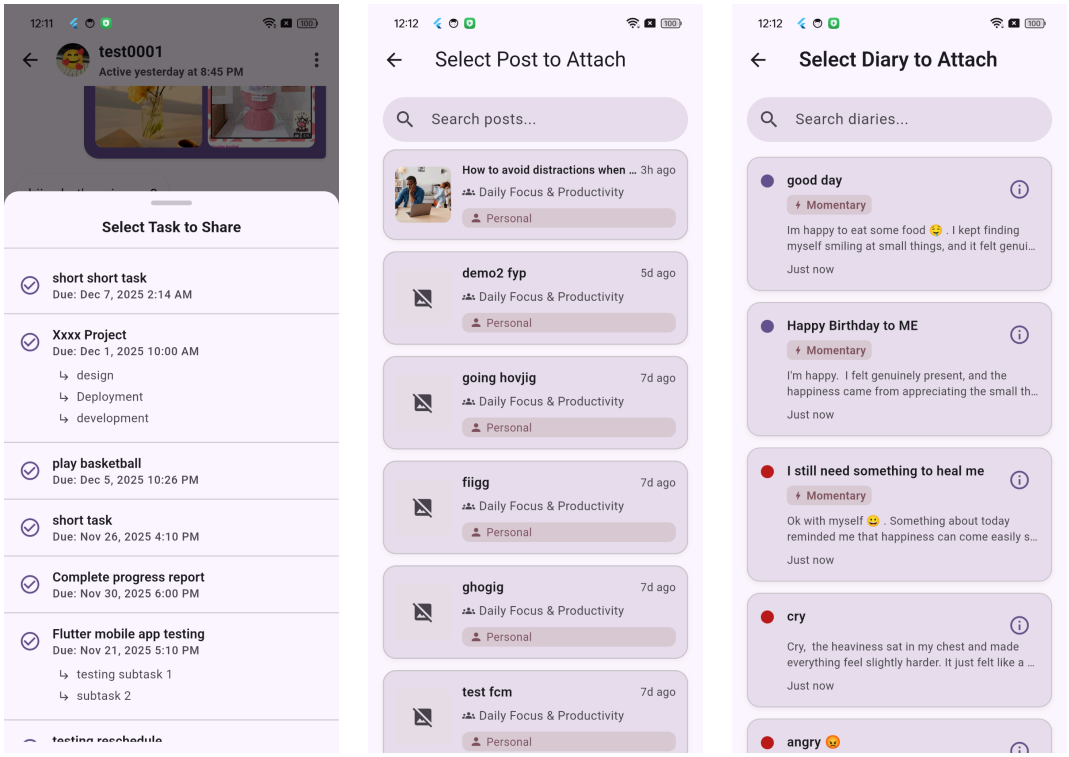
Table 4.3.3: User Interface Design - Anonymous Community Module

Personal Chat Module



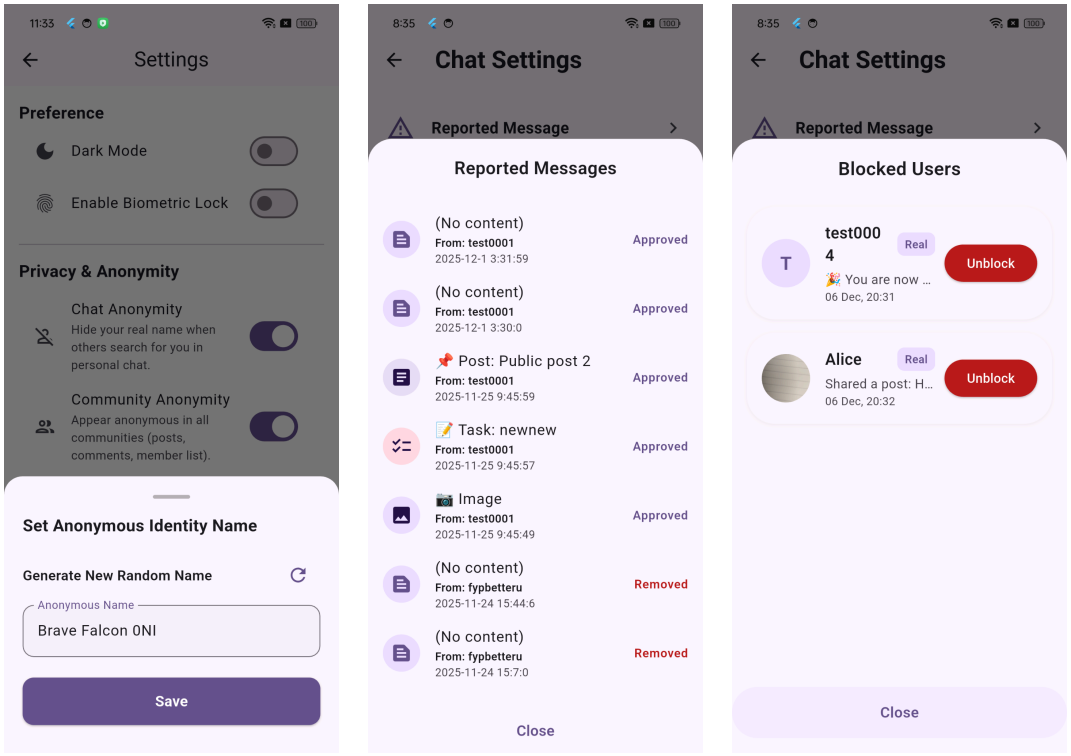
Personal Chat View

- The first screen displays the user's friend list, along with options to add new friends or access chat settings.
- The chat view (second and third screens) shows the conversation history and allows users to send messages or attach images, tasks, posts, and diary entries.



Attachment Picker

- The attachment picker provides three options for selecting content to share in chats. The task picker lets users choose from their task list or subtasks. The post picker displays all community posts available for sharing. The diary picker lists personal diary entries that users can attach to the conversation.

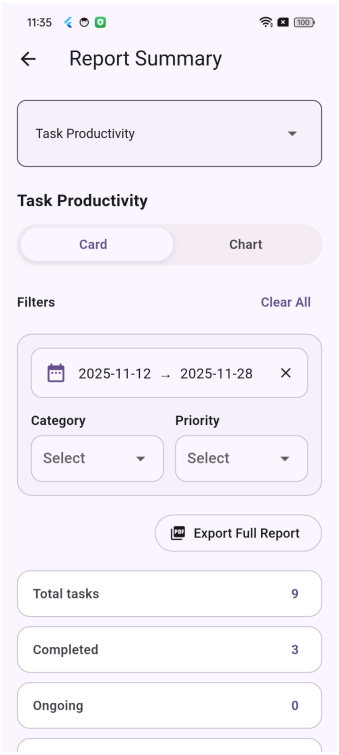


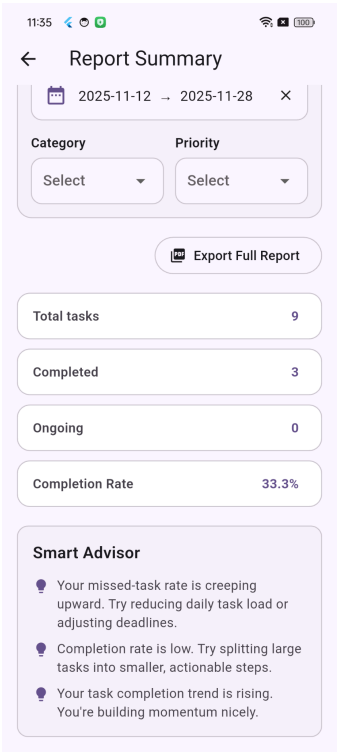
Personal Chat Settings

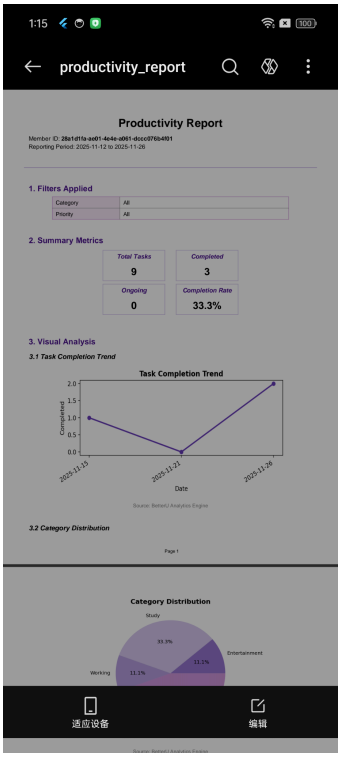
- The first screen allows users to set or generate a new anonymous identity name for private chatting.
- The second screen displays all reported chat messages along with their review status.
- The third screen shows the list of blocked users, where users can review or unblock them.

Table 4.3.4: User Interface Design - Personal Chat Module

Report Module

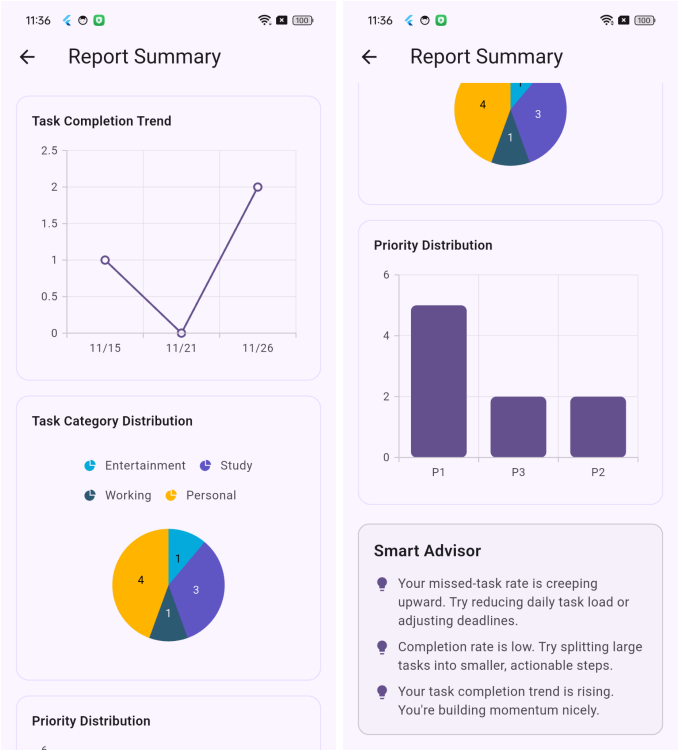






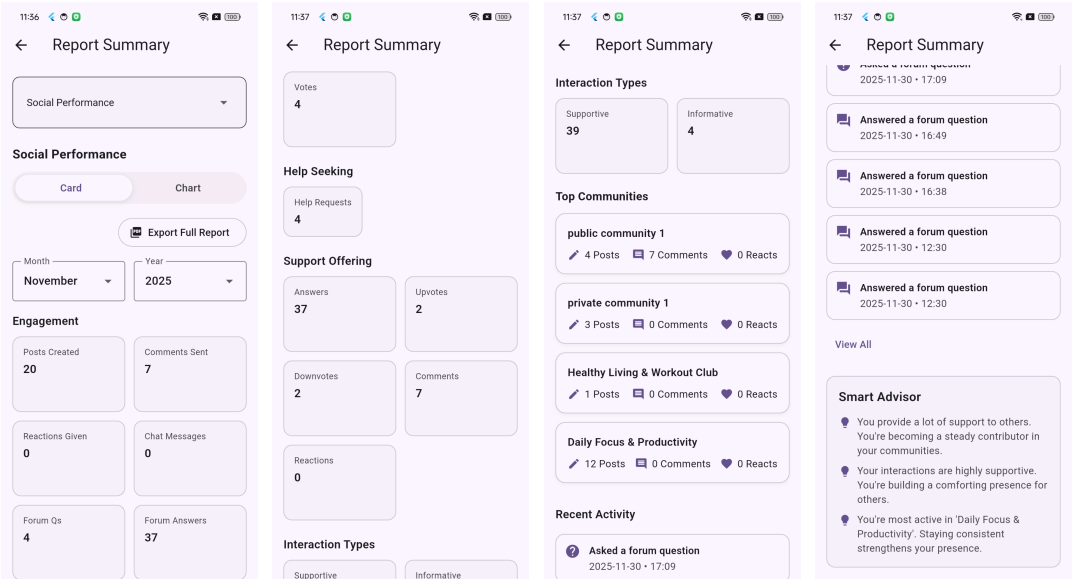
Task Productivity Report (Card View)

- This screen summarizes the user's task productivity based on the selected date range, category, and priority. It shows key statistics such as total tasks, completed tasks, ongoing tasks and the overall completion rate.
- A Smart Advisor section provides brief insights to help improve productivity, and users can export the full report with one tap.



Task Productivity Report (Chart View)

- This view visualizes the user's productivity using charts, including task completion trends over time, task category distribution and priority distribution.
- These charts help users quickly understand their progress and work patterns while the Smart Advisor provides brief insights based on the data.

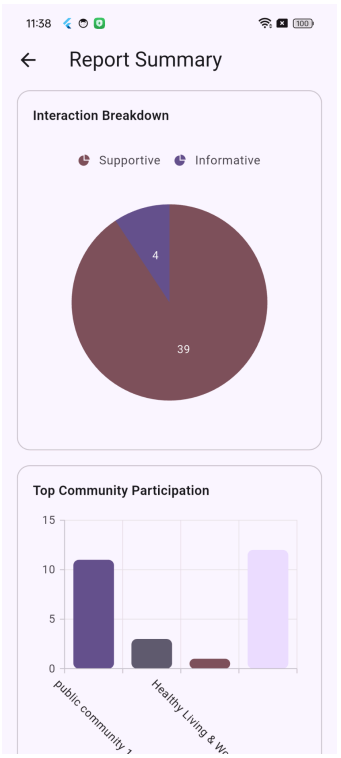


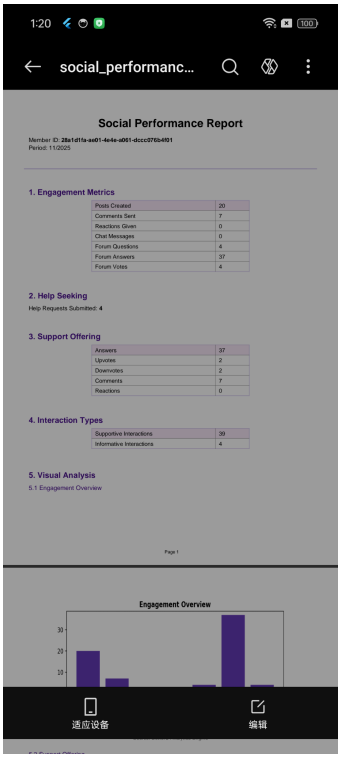
Social Performance Report (Card View)

- This view summarizes the user's social engagement within the app for the selected

month and year.

- It displays key activity metrics such as posts created, comments, answers, help requests, upvotes, and reactions.
- It also highlights interaction types, top communities the user participated in and recent activities.
- A Smart Advisor provides simple insights based on the user's contribution patterns and users can export the full report if needed.





Social Performance Report (Card View)

- The chart view visualizes the user's social activity through graphs such as engagement overview, interaction breakdown, support offering and community participation.
- These charts help users quickly understand their contribution patterns. Users can also export a full PDF report which summarizes all metrics and includes visual charts for easy review and sharing.

Table 4.3.5: User Interface Design - Report Module

4.4 Data Design

4.4.1 Class Diagram

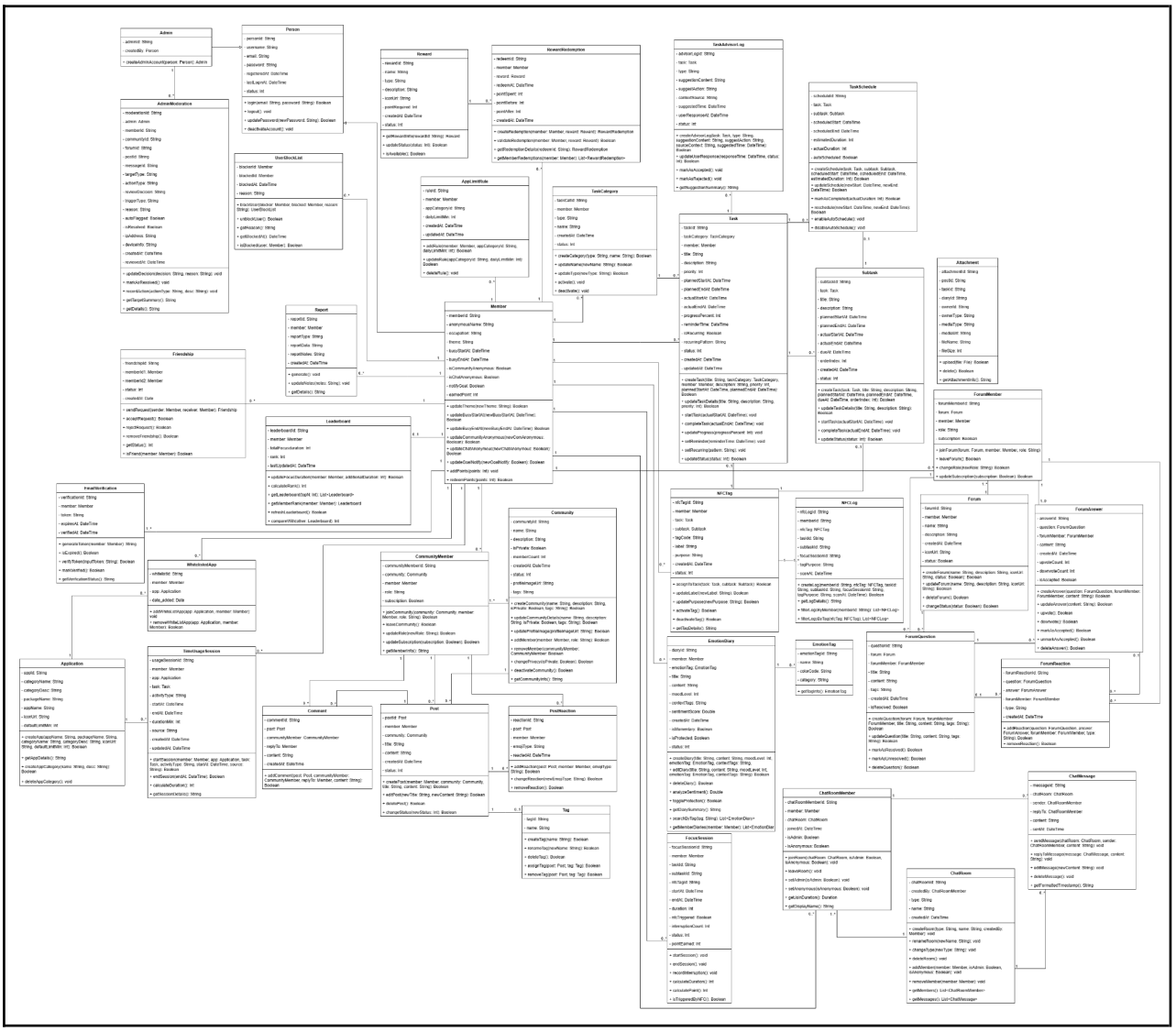


Diagram 4.4.1: Class Diagram of BetterU Mobile Application

4.4.2 Entity Relationship Diagram

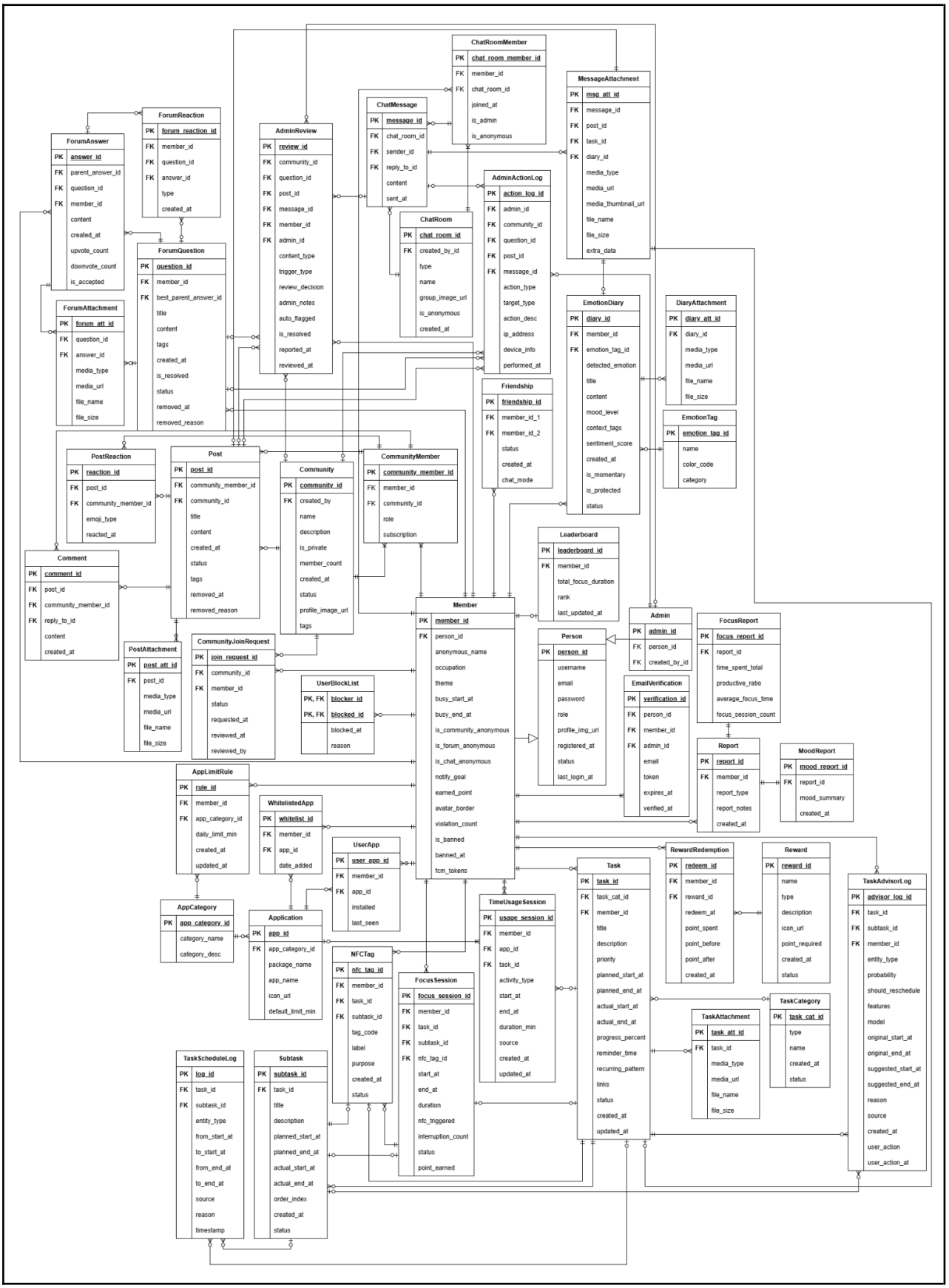


Diagram 4.4.2: Entity Relationship Diagram of BetterU Mobile Application

4.4.3 Data Dictionary

Person

Field Name	Data Type	Size/Constraint	Key	Default	Description
person_id	String	UUID (36)	PK	Auto-generated	Unique identifier for each person (primary key).
username	String	VARCHAR(50), Unique, Not Null	-	-	Display name chosen by the user for login and identification.
email	String	VARCHAR(100), Unique, Not Null	-	-	User's email address, used for login and communication.
password	String	VARCHAR(255), Not Null (hashed)	-	-	User's encrypted/hashed password for authentication.
role	String	VARCHAR(20), Not Null	-	-	Defines user's system role such as member or admin
profile_img_url	String	VARCHAR(255), Nullable	-	NULL	URL path pointing to the user's profile image stored
registered_at	DateTime	Not Null	-	Current timestamp	The date and time when the user registered.
last_login_at	DateTime	Nullable	-	NULL	Timestamp of the admin's

					most recent successful login.
status	Int	Enum: {0=Inactive, 1=Active}	-	1 (Active)	Account status indicator.

Table 4.4.1: Data Dictionary - Person Entity

Member

Field Name	Data Type	Size/Constraint	Key	Default	Description
member_id	String	UUID (36)	PK	Auto-generated	Unique identifier for each member (primary key).
person_id	String	UUID (36), Not Null	FK	-	References Person.person_id (foreign key).
anonymous_name	String	VARCHAR(50), Nullable	-	NULL	Pseudonym used in community/forum features if anonymity is enabled.
occupation	String	VARCHAR(100), Nullable	-	NULL	Member's occupation or role (optional).
theme	String	Enum: {Light, Dark, System}	-	System	Preferred UI theme for the app interface.
busy_start_at	Time	Nullable	-	NULL	Start time of daily busy period (used for scheduling/notifications).

busy_end_at	Time	Nullable	-	NULL	End time of daily busy period.
is_community_anonymous	Boolean	-	-	false	If true, the member appears anonymous in community posts.
is_forum_anonymous	Boolean	-	-	false	If true, the member appears anonymous when posting in the forum.
is_chat_anonymous	Boolean	-	-	false	If true, the member appears anonymous in chats.
notify_goal	Boolean	-	-	true	Whether the member receives goal-related notifications.
earned_point	Integer	>=0	-	0	Accumulated reward points earned through app activities.
avatar_border	String	VARCHAR(50), Nullable	-	NULL	The selected decorative border style for the member's avatar.
violation_count	Integer	Not Null, >= 0	-	0	Total number of approved violations committed by the member.

is_banned	Boolean	Not Null	-	False	Indicates whether the member is banned due to repeated violations.
banned_at	DateTime	Nullable	-	NULL	Timestamp of when the member was banned.
fcm_tokens	Array	Array of VARCHAR(255), Nullable	-	Empty list ([])	List of Firebase Cloud Messaging (FCM) tokens for the member's devices for pushing notifications.

Table 4.4.2: Data Dictionary - Member Entity

Admin

Field Name	Data Type	Size/Constraint	Key	Default	Description
admin_id	String	UUID (36)	PK	Auto-generated	Unique identifier for each admin (primary key).
person_id	String	UUID (36), Not Null	FK	-	References Person.person_id (foreign key). Links the admin role to a person.
created_by_id	String	UUID (36), Nullable	FK (self-ref)	NULL	References Admin.admin_id who created this admin account (for tracking delegation).

Table 4.4.3: Data Dictionary - Admin Entity

EmailVerification

Field Name	Data Type	Size/Constraint	Key	Default	Description
verification_id	String	UUID (36)	PK	Auto-generated	Unique identifier for each email verification record.
person_id	String	UUID (36), Not Null	FK	-	References Person.person_id as a universal identifier for the individual whose email is being verified.
member_id	String	UUID (36), Not Null	FK	-	References Member.member_id (the member whose email is being verified).
admin_id	String	UUID (36), Nullable	FK	-	References Admin.admin_id when the email verification is initiated or approved by an administrator.
email	String	VARCHAR (255), Not Null	-	-	The email address undergoing verification.
token	String	VARCHAR(255), Unique, Not Null	-	-	Randomly generated secure token used for email verification.

expires_at	DateTime	Not Null	-	-	Timestamp indicating when the verification token becomes invalid.
verified_at	DateTime	Nullable	-	NULL	Timestamp of when the email was successfully verified.

Table 4.4.4: Data Dictionary - EmailVerification Entity

Friendship

Field Name	Data Type	Size/Constraint	Key	Default	Description
friendship_id	String	UUID (36)	PK	Auto-generated	Unique identifier for each friendship record.
member_id_1	String	UUID (36), Not Null	FK	-	References Member.member_id. Represents one side of the friendship.
member_id_2	String	UUID (36), Not Null	FK	-	References Member.member_id. Represents the other side of the friendship.
status	String	Enum: {0=Blocked, 1=Pending, 2=Accepted, 3=Declined}	-	1	Current status of the friendship request.
created_at	DateTime	Not Null	-	Current timestamp	The date and time when the friendship

					request was initiated.
chat_mode	String	Enum: {"real", "anonymous"}, Not Null	-	real	Preferred chat identity mode selected by the requester.

Table 4.4.5: Data Dictionary - Friendship Entity

UserBlockList

Field Name	Data Type	Size/Constraint	Key	Default	Description
blocker_id	String	UUID (36), Not Null	PK, FK	-	References Member.member_id. The member who initiates the block.
blocked_id	String	UUID (36), Not Null	PK, FK	-	References Member.member_id. The member being blocked.
blocked_at	DateTime	Not Null	-	Current timestamp	The date and time when the block action occurred.
reason	String	VARCHAR(255), Nullable	-	NULL	Optional description or reason provided for blocking.

Table 4.4.6: Data Dictionary - UserBlockedList Entity

TaskCategory

Field Name	Data Type	Size/Constraint	Key	Default	Description
task_cat_id	String	UUID (36)	PK	Auto-generated	Unique identifier for each task category.

type	String	Enum: {System, User}	-	User	Indicates whether the category was system pre-created (System) or created by a member (User).
name	String	VARCHAR(100), Not Null	-	-	Name of the category (e.g., Work, Study, Fitness, etc.).
created_at	DateTime	Not Null	-	Current timestamp	The date and time when the category was created.
status	Integer	Enum: {0=Inactive, 1=Active}	-	Active	Current status of the task category.

Table 4.4.7: Data Dictionary - TaskCategory Entity

Task

Field Name	Data Type	Size/Constraint	Key	Default	Description
task_id	String	UUID (36)	PK	Auto-generated	Unique identifier for each task.
task_cat_id	String	UUID (36), Nullable	FK	NULL	References TaskCategory.task_cat_id. Defines which category the task belongs to.
member_id	String	UUID (36), Not Null	FK	—	References Member.member_id. The member who created/owns the task.

title	String	VARCHAR(150), Not Null	—	—	Title or short name of the task.
description	Text	Nullable	—	NULL	Detailed description of the task.
priority	Integer	Enum: {1=High, 2=Medium, 3=Low}	—	Medium	Importance level of the task.
planned_start_at	DateTime	Nullable	—	NULL	Planned start date/time of the task.
planned_end_at	DateTime	Nullable	—	NULL	Planned deadline/end date of the task.
actual_start_at	DateTime	Nullable	—	NULL	Actual start time of the task (when user begins).
actual_end_at	DateTime	Nullable	—	NULL	Actual completion time of the task.
progress_percent	Integer	Range 0–100	—	0	Percentage progress of the task.
reminder_time	DateTime	Nullable	—	NULL	Date/time when a reminder notification should be sent.
recurring_pattern	String	Nullable (e.g., Daily, Weekly, Monthly)	—	NULL	Pattern definition for recurrence (if is_recurring = true).
links	Array of String	Nullable	-	NULL	List of related links or URLs associated

					with the task.
status	Integer	Enum: {0=Cancelled, 1=Pending, 2=Ongoing, 3=Completed, 4=Overdue}	—	Pending	Current lifecycle state of the task.
created_at	DateTime	Not Null	—	Current timestamp	When the task was created.
updated_at	DateTime	Not Null	—	Current timestamp (auto-update)	When the task record was last modified.

Table 4.4.8: Data Dictionary - Task Entity

Subtask

Field Name	Data Type	Size/Constraint	Key	Default	Description
subtask_id	String	UUID (36)	PK	Auto-generated	Unique identifier for each subtask.
task_id	String	UUID (36), Not Null	FK	-	References Task.task_id. Defines the parent task this subtask belongs to.
title	String	VARCHAR(150), Not Null	-	-	Title or short name of the subtask.
description	Text	Nullable	-	NULL	Detailed description of the subtask.
planned_start_at	DateTime	Nullable	-	NULL	Planned start date/time of the subtask.
planned_end_at	DateTime	Nullable	-	NULL	Planned deadline/end

					date of the subtask.
actual_start_at	DateTime	Nullable	-	NULL	Actual start time of the subtask.
actual_end_at	DateTime	Nullable	-	NULL	Actual completion time of the subtask.
order_index	Integer	>= 0	-	0	Defines the display or execution order of subtasks within a task.
created_at	DateTime	Not Null	-	Current timestamp	Timestamp when the subtask was created.
status	Integer	Enum: {0=Pending, 1=Ongoing, 2=Completed, 3=Overdue}	-	Pending	Current lifecycle state of the subtask.

Table 4.4.9: Data Dictionary - Subtask Entity

TaskAttachment

Field Name	Data Type	Size/Constraint	Key	Default	Description
task_att_id	String	UUID (36)	PK	Auto-generated	Unique identifier for each task attachment.
task_id	String	UUID (36), Not Null	FK	-	References Task.task_id. Defines the task this attachment belongs to.
media_type	String	Enum: {Image, Video}	-	-	Type of media

					attached to the task.
media_url	String	VARCHAR(255), Not Null	-	-	URL where the attachment is stored.
file_name	String	VARCHAR(150), Not Null	-	-	Filename of the uploaded attachment.
file_size	Integer	Size in KB, >=0	-	0	Size of the attachment file.

Table 4.4.10: Data Dictionary - TaskAttachment Entity

TaskScheduleLog

Field Name	Data Type	Size/Constraint	Key	Default	Description
log_id	String	UUID (36)	PK	Auto-generated	Unique identifier for each schedule change log entry.
task_id	String	UUID (36), Not Null	FK	-	References Task.task_id. Identifies the task that this log entry belongs to.
subtask_id	String	UUID (36), Not Null	FK	-	References Subtask.subtask_id when the change belongs to a subtask; NULL otherwise.
entity_type	String	Enum: {"task", "subtask"}, Not Null	-	-	Indicates whether the schedule changed applies to a task or a subtask

from_start_at	DateTime	Not Null	-	-	Previous scheduled start time before the change occurred.
to_start_at	DateTime	Not Null	-	-	Updated scheduled start time after the change.
from_end_at	DateTime	Not Null	-	-	Previous scheduled start time before the change.
to_end_at	DateTime	Not Null	-	-	Updated scheduled end time after the change.
source	String	Enum: {"user", "system_ai"}, Not Null	-	user	Indicates who triggered the schedule change (user or AI system).
reason	String	Nullable (Text)	-	NULL	Optional explanation or justification for the schedule adjustment.
timestamp	DateTime	Not Null	-	Current timestamp	The date and time when the rescheduling event occurred.

Table 4.4.11: Data Dictionary - TaskScheduleLog Entity

TaskAdvisorLog

Field Name	Data Type	Size/Constraint	Key	Default	Description

advisor_log_id	String	UUID (36)	PK	Auto-generated	Unique identifier for each advisor log entry.
task_id	String	UUID (36), Not Null	FK	-	References Task.task_id. The task that the suggestion is related to.
subtask_id	String	UUID (36), Not Null	FK	-	References Subtask.subtask_id when the suggestion is for a subtask.
member_id	String	UUID (36), Not Null	FK	-	References Member.member_id. Represents the user receiving the advisor suggestion
entity_type	String	Enum: {"task", "subtask"}, Not Null	-	task	Indicates whether the advisor suggestion relates to a task or a subtask
should_reschedule	Boolean	Not Null	-	-	Indicates whether the AI predicts that the task's schedule should be adjusted.
features	JSON	Not Null	-	{}	Dictionary of input features used by the AI model to generate the suggestion.

model	String	VARCHAR (100), Not Null	-	-	Name or identifier of the AI model that generated the suggestion.
original_start_at	DateTime	Nullable	-	NULL	Previously scheduled start time before the advisor-generated update.
original_end_at	DateTime	Nullable	-	NULL	Previously scheduled end time before the advisor-generated update.
suggested_start_at	DateTime	Nullable	-	NULL	AI-suggested start time after analyzing schedule conflicts or productivity patterns.
suggested_end_at	DateTime	Nullable	-	NULL	AI-suggested end time for the task or subtask
reason	String	Nullable (Text)	-	NULL	Explanation of why the AI suggested the change.
source	String	Enum: {"ai_optimizer", "manual", "calendar_conflict"}, Not Null	-	ai_optimizer	Origin of the suggestion.
created_at	DateTime	Not Null	-	Current timestamp	Timestamp when the advisor log entry was created.

user_action	String	Enum: {"pending", "accepted", "rejected", "ignored"}, Not Null	-	pending	Indicates how the user responded to the advisor's suggestion.
user_action_at	DateTime	Nullable	-	NULL	Timestamp of the user's response to the suggestion.

Table 4.4.12: Data Dictionary - TaskAdvisorLog Entity

AppCategory

Field Name	Data Type	Size/Constraint	Key	Default	Description
app_category_id	String	UUID (36)	PK	Auto-generated	Unique identifier for each app category.
category_name	String	100, Not Null, Unique	-	-	Name of the category (e.g., Productivity, Education, Health).
category_desc	Text	Nullable	-	NULL	Description of the category, explaining its purpose or scope.

Table 4.4.13: Data Dictionary - AppCategory Entity

AppLimitRule

Field Name	Data Type	Size/Constraint	Key	Default	Description
rule_id	String	UUID (36)	PK	Auto-generated	Unique identifier for each app usage limit rule.
member_id	String	UUID (36)	FK	-	References the Member

					who owns this rule.
app_category_id	String	UUID (36)	FK	-	References the AppCategory this limit applies to.
daily_limit_min	Integer	>= 0	-	0	Maximum daily allowed usage (in minutes) for apps in this category.
created_at	DateTime	Not Null	-	Current time	Timestamp when the rule was created.
updated_at	DateTime	Not Null	-	Auto-updated	Timestamp when the rule was last modified.

Table 4.4.14: Data Dictionary - AppLimitRule Entity

Application

Field Name	Data Type	Size/Constraint	Key	Default	Description
app_id	String	UUID (36)	PK	Auto-generated	Unique identifier for each application record.
app_category_id	String	UUID (36)	FK	-	References AppCategory that this application belongs to.
package_name	String	150	Unique	-	System package identifier (e.g., com.whatsapp, org.mozilla.firefox).

app_name	String	100	-	-	Display name of the application (e.g., WhatsApp, Chrome).
icon_url	String	255	-	-	URL or file path to the app's icon.
default_limit_min	Integer	≥ 0	-	0	Default recommended daily limit (in minutes).

Table 4.4.15: Data Dictionary - Application Entity

UserApp

Field Name	Data Type	Size/Constraint	Key	Default	Description
user_app_id	String	UUID (36)	PK	Auto-generated	Unique identifier for each user-application record.
member_id	String	UUID (36), Not Null	FK	-	References Member.member_id. Identifies the member who installed or uses the application.
app_id	String	UUID (36), Not Null	FK		References Application.app_id. Indicates which third-party app or integrated service is being tracked.
installed	Boolean	Not Null	-	True	Indicates whether the app is currently

					installed by the member.
last_seen	DateTime	Not Null	-	Current timestamp	Timestamp of the most recent time the user interacted with or opened the app.

Table 4.4.16: Data Dictionary - UserApp Entity

TimeUsageSession

Field Name	Data Type	Size/Constraint	Key	Default	Description
usage_session_id	String	UUID (36)	PK	Auto-generated	Unique identifier for each usage session.
member_id	String	UUID (36), Not Null	FK	-	References Member.member_id. The member performing the activity.
app_id	String	UUID (36), Nullable	FK	NULL	References Application.app_id. Optional, used if session involves app usage.
task_id	String	UUID (36), Nullable	FK	NULL	References Task.task_id. Optional, used if session involves task focus.
activity_type	Enum	{AppUsage, TaskFocus, Other}	-	-	Type of activity being tracked in this session.

start_at	DateTime	Not Null	-	-	Start timestamp of the session.
end_at	DateTime	Nullable	-	NULL	End timestamp of the session. Can be NULL if session is ongoing.
duration_min	Integer	>= 0	-	0	Duration of the session in minutes. Can be auto-calculated from end_at – start_at.
source	String	VARCHAR(50)	-	Device/App	Source of the activity data (e.g., MobileApp, Task, NFCTrigger).
created_at	DateTime	Not Null	-	Current timestamp	Timestamp when the session record was created.
updated_at	DateTime	Not Null	-	Auto-updated	Timestamp when the session record was last updated.

Table 4.4.17: Data Dictionary - TimeUsageSession Entity

WhitelistedApp

Field Name	Data Type	Size/Constraint	Key	Default	Description
whitelist_id	String	UUID (36)	PK	Auto-generated	Unique identifier for each whitelist entry.

member_id	String	UUID (36), Not Null	FK	-	References Member.member_id. The member who set the whitelist.
app_id	String	UUID (36), Not Null	FK	-	References Application.app_id. The whitelisted application.
date_added	DateTime	Not Null	-	Current timestamp	Date and time when the app was whitelisted.

Table 4.4.18: Data Dictionary - WhiteListedApp Entity

NFCTag

Field Name	Data Type	Size/Constraint	Key	Default	Description
nfc_tag_id	String	UUID (36)	PK	Auto-generated	Unique identifier for each NFC tag record.
member_id	String	UUID (36), Not Null	FK	-	References Member.member_id. The owner of the NFC tag.
task_id	String	UUID (36), Nullable	FK	-	References Task.task_id. Task linked to this NFC tag (optional if tied directly to a subtask).
subtask_id	String	UUID (36), Nullable	FK	-	References Subtask.subtask_id. Subtask linked to this NFC tag.

tag_code	String	255, Unique, Not Null	-	-	The physical NFC tag's unique code/UID read from the chip.
label	String	100	-	-	User-friendly label given to the tag (e.g., "Study Desk Tag").
purpose	String	50	-	-	Describes the tag's intended use (e.g., Focus, TaskStart, TaskEnd, Reminder).
created_at	DateTime	Not Null	-	Current timestamp	The time the NFC tag was registered in the system.
status	Integer	Enum: {0=Inactive, 1=Active}	-	1	Current availability/usage state of the tag.

Table 4.4.19: Data Dictionary - NFCTag Entity

Community

Field Name	Data Type	Size/Constraint	Key	Default	Description
community_id	String	UUID (36)	PK	Auto-generated	Unique identifier for the community.
created_by	String	UUID (36), Not Null	FK	-	References Member.member_id. Identifies the member who created the community.
name	String	150, Not Null, Unique	-	-	Display name of the community.

description	Text	-	-	-	Brief overview or details about the community's purpose.
is_private	Boolean	Not Null	-	false	Defines if the community is private (invite/join request only) or public.
member_count	Integer	>= 1, Not Null	-	1	Total number of members in the community (starts at 1 for the creator).
created_at	DateTime	Not Null	-	Current timestamp	Timestamp of when the community was created.
status	Integer	Enum: {0 = inactive, 1 = active}	-	1	Current status of the community.
profile_image_url	String	255	-	NULL	URL for the community's profile picture or banner.
tags	String	-	-	NULL	List of tags/keywords associated with the community.

Table 4.4.20: Data Dictionary - Community Entity

CommunityJoinRequest

Field Name	Data Type	Size/Constraint	Key	Default	Description
join_request_id	String	UUID (36)	PK	Auto-generated	Unique identifier for

					each community join request.
community_id	String	UUID (36), Not Null	FK	-	References Community.community_id. The community the member wants to join.
member_id	String	UUID (36), Not Null	FK	-	References Member.member_id. The member who submitted the join request.
reviewed_by	String	UUID (36), Not Null	FK	NULL	References Member.member_id. The admin who reviewed the join request.
status	Integer	Enum: {1=Pending, 2=Approved, 3=Rejected}, Not Null	-	1	Current review status of the join request.
requested_at	DateTime	Not Null	-	Current timestamp	Timestamp when the join request was submitted.
reviewed_at	DateTime	Nullable	-	NULL	Timestamp when the request was reviewed (approved or rejected).

Table 4.4.21: Data Dictionary - CommunityJoinRequest Entity

CommunityMember

Field Name	Data Type	Size/Constraint	Key	Default	Description
community_member_id	String	UUID (36)	PK	Auto-generated	Unique identifier for each community membership record.
member_id	String	UUID (36), Not Null	FK	-	References Member.member_id. The member who joined the community.
community_id	String	UUID (36), Not Null	FK	-	References Community.community_id. The community this member belongs to.
role	Integer	Enum: {1=member, 2=creator}, Not Null	-	-	Role of the member in the community, referencing their authority level.
subscription	Boolean	Not Null	-	true	Whether the member is subscribed to community updates/notifications.

Table 4.4.22: Data Dictionary - CommunityMember Entity

Post

Field Name	Data Type	Size/Constraint	Key	Default	Description
post_id	String	UUID (36)	PK	Auto-generated	Unique identifier for the post.

community_member_id	String	UUID (36), Not Null	FK	-	References CommunityMember.community_member_id. Identifies the member who created the post.
community_id	String	UUID (36), Not Null	FK	-	References Community.community_id. Identifies the community where the post belongs.
title	String	200, Not Null	-	-	Title or headline of the post.
content	Text	Not Null	-	-	Main body/content of the post.
created_at	DateTime	Not Null	-	Current timestamp	Timestamp when the post was created.
status	Integer	Enum: {0 = inactive, 1 = active}, Not Null	-	1	Current status of the post.
tags	Array of String	Nullable	-	Empty list ([])	List of tag names used for search and filtering.
removed_at	DateTime	Nullable	-	NULL	Timestamp indicating when the post was soft-deleted or removed by an admin.
removed_reason	String	Nullable (Text)	-	NULL	Explanation provided by an admin stating why

					the post was removed.

Table 4.4.23: Data Dictionary - Post Entity

PostAttachment

Field Name	Data Type	Size/Constraint	Key	Default	Description
post_att_id	String	UUID (36)	PK	Auto-generated	Unique identifier for each post attachment.
post_id	String	UUID (36), Not Null	FK	-	References Post.post_id. The post that this attachment belongs to.
media_type	String	50, Not Null	-	-	Type of the media (e.g., image, video).
media_url	String	255, Not Null	-	-	URL to the uploaded media file.
file_name	String	150, Not Null	-	-	File name of the attachment.
file_size	Integer	Size in KB, Not Null	-	-	Size of the file in bytes.

Table 4.4.24: Data Dictionary - PostAttachment Entity

PostReaction

Field Name	Data Type	Size/Constraint	Key	Default	Description
reaction_id	String	UUID (36)	PK	Auto-generated	Unique identifier for each reaction.
post_id	String	UUID (36), Not Null	FK	-	References Post.post_id. The post that

					received the reaction.
community_member_id	String	UUID (36), Not Null	FK	-	References CommunityMember.community_member_id. The member who reacted to the post.
emoji_type	String	50, Not Null	-	-	Type of reaction/emoji used (e.g., like, love, laugh, sad, angry).
reacted_at	DateTime	Not Null	-	Current timestamp	Timestamp when the reaction was made.

Table 4.4.25: Data Dictionary - PostReaction Entity

Comment

Field Name	Data Type	Size/Constraint	Key	Default	Description
comment_id	String	UUID (36)	PK	Auto-generated	Unique identifier for each comment.
post_id	String	UUID (36), Not Null	FK	-	References Post.post_id. The post that this comment belongs to.
community_member_id	String	UUID (36), Not Null	FK	-	References CommunityMember.community_member_id. The member who made the comment.

reply_to_id	String	UUID (36), Nullable	FK	NULL	References Comment.comment_id. If the comment is a reply, this points to the parent comment.
content	Text	Not Null	-	-	Text content of the comment.
created_at	DateTime	Not Null	-	Current timestamp	Timestamp when the comment was created.

Table 4.4.26: Data Dictionary - Comment Entity

ForumQuestion

Field Name	Data Type	Size/Constraint	Key	Default	Description
question_id	String	UUID (36)	PK	Auto-generated	Unique identifier for each forum question.
member_id	String	UUID (36), Not Null	FK	-	References Member.member_id. Identifies the member who posted the forum question.
best_parent_answer_id	String	UUID (36), Nullable	FK	NULL	References ForumAnswer.answer_id. Stores the ID of the accepted/best answer for this question.
title	String	255, Not Null	-	-	Title of the forum question.

content	Text	Not Null	-	-	Detailed description of the forum question.
tags	String	255, Nullable	-	-	Comma-separated tags for categorization of the question.
created_at	DateTime	Not Null	-	Current timestamp	When the question was created.
is_resolved	Boolean	Not Null	-	false	Indicates whether the question has been resolved.
status	Integer	Enum: {0=Inactive, 1=Active}, Not Null	-	1	Indicates the current lifecycle state of the forum question.
removed_at	DateTime	Nullable	-	NULL	Timestamp when the question was removed or soft-deleted by a moderator.
removed_reason	String	Nullable (Text)	-	NULL	Reason provided by a moderator for removing the question.

Table 4.4.27: Data Dictionary - ForumQuestion Entity

ForumAnswer

Field Name	Data Type	Size/Constraint	Key	Default	Description
answer_id	String	UUID (36)	PK	Auto-generated	Unique identifier for each forum answer.

parent_answer_id	String	UUID (36), Not Null	FK	NULL	References ForumAnswer.answer_id when this answer is a reply to another answer.
question_id	String	UUID (36), Not Null	FK	-	References ForumQuestion.question_id. The question this answer belongs to.
member_id	String	UUID (36), Not Null	FK	-	References Member.member_id. Identifies the member who posted the forum answer.
content	Text	Not Null	-	-	The textual content of the forum answer.
created_at	DateTime	Not Null	-	Current timestamp	When the answer was created.
upvote_count	Integer	Not Null	-	0	Number of upvotes the answer has received.
downvote_count	Integer	Not Null	-	0	Number of downvotes the answer has received.
is_accepted	Boolean	Not Null	-	false	Indicates whether the answer has been accepted as the solution.

Table 4.4.28: Data Dictionary - ForumAnswer Entity

ForumAttachment

Field Name	Data Type	Size/Constraint	Key	Default	Description
forum_att_id	String	UUID (36)	PK	Auto-generated	Unique identifier for the forum attachment.
question_id	String	UUID (36), Nullable	FK	-	References ForumQuestion.question_id. Required if the attachment belongs to a question.
answer_id	String	UUID (36), Nullable	FK	-	References ForumAnswer.answer_id. Required if the attachment belongs to an answer.
media_type	String	Enum: {image, video}, Not Null	-	-	The type of media file attached.
media_url	String	255, Not Null	-	-	URL of the attachment.
file_name	String	255	-	NULL	File name of the uploaded attachment.
file_size	Integer	≥ 0	-	NULL	File size in bytes (for validation or limits).

Table 4.4.29: Data Dictionary - ForumAttachment Entity

ForumReaction

Field Name	Data Type	Size/Constraint	Key	Default	Description
forum_reaction_id	String	UUID (36)	PK	Auto-generated	Unique identifier for the reaction.

question_id	String	UUID (36), Nullable	FK	-	References ForumQuestion.question_id.
answer_id	String	UUID (36), Nullable	FK	-	References ForumAnswer.answer_id.
member_id	String	UUID (36), Not Null	FK	-	References Member.member_id. Identifies the member who voted to this forum question.
type	Integer	Enum: {0=Downvote, 1=Upvote}	-	-	Reaction type.
created_at	DateTime	Not Null	-	CURRENT_TIMESTAMP	Timestamp when the reaction was added.

Table 4.4.30: Data Dictionary - ForumReaction Entity

ChatRoom

Field Name	Data Type	Size/Constraint	Key	Default	Description
chat_room_id	String	UUID (36)	PK	Auto-generated	Unique identifier for the chat room.
created_by_id	String	UUID (36), Not Null	FK	-	References ChatRoomMember.chat_room_member_id (the creator participant).
type	Integer	Enum: {1=Direct, 2=Group}	-	-	Defines the type of chat room.

name	String	255, Nullable	-	NULL	Room name (only required for group/community chats).
group_image_url	String	VARCHAR (255), Nullable	-	NULL	URL of the group chat's image/avatar. Only applicable for group chat rooms.
is_anonymous	Boolean	Not Null	-	false	Indicates whether this chat room hides member identities.
created_at	DateTime	Not Null	-	CURRENT_TIMESTAMP	Timestamp when the chat room was created.

Table 4.4.31: Data Dictionary - ChatRoom Entity

ChatRoomMember

Field Name	Data Type	Size/Constraint	Key	Default	Description
chat_room_member_id	String	UUID (36)	PK	Auto-generated	Unique identifier for a chat room participant.
member_id	String	UUID (36), Not Null	FK	-	References Member.member_id (the actual user).
chat_room_id	String	UUID (36), Not Null	FK	-	References ChatRoom.chat_room_id.
joined_at	DateTime	Not Null	-	CURRENT_TIMESTAMP	Timestamp when the member joined the chat room.

is_admin	Boolean	Not Null	-	FALSE	Indicates if the member has admin privileges in the chat room.
is_anonymous	Boolean	Not Null	-	FALSE	Defines whether the member is visible by identity or hidden (e.g., anonymous posting/chat).

Table 4.4.32: Data Dictionary - ChatRoomMember Entity

ChatMessage

Field Name	Data Type	Size/Constraint	Key	Default	Description
message_id	String	UUID (36)	PK	Auto-generated	Unique identifier for each chat message.
chat_room_id	String	UUID (36), Not Null	FK	-	References ChatRoom.chat_room_id, links the message to a chat room.
sender_id	String	UUID (36), Not Null	FK	-	References ChatRoomMember.chat_room_member_id, ensures only valid room members can send messages.
reply_to_id	String	UUID (36), Nullable	FK (self-reference)	-	References ChatMessage.message_id, allows threaded replies.

content	Text	Not Null	-	-	Message body text (supports plain text, markdown, or rich text depending on design).
sent_at	DateTime	Not Null	-	CURRENT_TIMESTAMP	Timestamp when the message was sent.

Table 4.4.33: Data Dictionary - ChatMessage Entity

MessageAttachment

Field Name	Data Type	Size/Constraint	Key	Default	Description
msg_att_id	String	UUID (36)	PK	Auto-generated	Unique identifier for each attachment.
message_id	String	UUID (36), Not Null	FK	-	References ChatMessage.message_id, links the attachment to a specific message.
post_id	String	UUID (36), Nullable	FK	-	References Post.post_id, links the attachment to a specific community post.
task_id	String	UUID (36), Nullable	FK	-	References Task.task_id, links the attachment to a specific personal task.
diary_id	String	UUID (36), Nullable	FK	-	References EmotionDiary.diary_id, links the

					attachment to a specific emotion diary note.
media_type	Enum	{image, video}	-	-	Type of media uploaded.
media_url	String	255, Not Null	-	-	Storage location (cloud URL or local path).
media_thumbnail_url	String	VARCHAR (255), Nullable	-	NULL	URL of the thumbnail image generated for media files.
file_name	String	255, Nullable	-	-	Original name of the file uploaded.
file_size	Integer	≥ 0	-	NULL	File size in bytes (for validation or limits).
extra_data	JSON	Nullable	-	NULL	Optional structured data used for non-media attachments.

Table 4.4.34: Data Dictionary - MessageAttachment Entity

FocusSession

Field Name	Data Type	Size/Constraint	Key	Default	Description
focus_session_id	String	UUID (36)	PK	Auto-generated	Unique identifier for each focus session.
member_id	String	UUID (36), Not Null	FK	-	References Member.member_id, identifies the user.

task_id	String	UUID (36), Nullable	FK	-	References Task.task_id, links to a main task.
subtask_id	String	UUID (36), Nullable	FK	-	References Subtask.subtask_id, links to a subtask (optional).
nfc_tag_id	String	UUID (36), Nullable	FK	-	References NFCTag.nfc_tag_id, indicates if session was tied to a tag.
start_at	DateTime	Not Null	-	CURRENT_TIMESTAMP	When the session started.
end_at	DateTime	Nullable	-	NULL	When the session ended (NULL if ongoing).
duration	Integer	In minutes	-	-	Total focus time, derived as end_at - start_at.
nfc_triggered	Boolean	{true, false}	-	false	Whether session started via NFC tag.
interruption_count	Integer	Default 0	-	0	Number of interruptions recorded.
status	Integer	Enum: {0=cancelled, 1=active, 2=ongoing, 3=completed}	-	0	Session lifecycle state.
point_earned	Integer	Default 0	-	0	Reward points earned for completing the session.

Table 4.4.35: Data Dictionary - FocusSession Entity

Reward

Field Name	Data Type	Size/Constraint	Key	Default	Description
reward_id	String	UUID (36)	PK	Auto-generated	Unique identifier for each reward.
name	String	150 chars, Not Null	-	-	Display name of the reward.
type	String	50 chars, Not Null	-	-	Category of reward (e.g., voucher, badge, coupon, gift).
description	Text	Optional	-	NULL	Detailed explanation of the reward.
icon_url	String	255 chars, Nullable	-	NULL	URL for reward icon image.
point_required	Integer	Not Null	-	-	Number of points a member must redeem to claim this reward.
created_at	DateTime	Not Null	-	CURRENT_TIMESTAMP	Date and time the reward was created.
status	Integer	Enum: {0=inactive, 1=active}	-	1	Indicates reward availability.

Table 4.4.36: Data Dictionary - Reward Entity

RewardRedemption

Field Name	Data Type	Size/Constraint	Key	Default	Description
redeem_id	String	UUID (36)	PK	Auto-generated	Unique identifier for

					each reward redemption record.
member_id	String	UUID (36), Not Null	FK	-	References Member.member_id; identifies the member who redeemed the reward.
reward_id	String	UUID (36), Not Null	FK	-	References Reward.reward_id; specifies which reward was redeemed.
redeem_at	DateTime	Not Null	-	CURRENT_TIMESTAMP	Date and time when the reward was redeemed.
point_spent	Integer	Not Null	-	0	Number of points spent for this redemption.
point_before	Integer	Not Null	-	0	Member's point balance before redemption.
point_after	Integer	Not Null	-	0	Member's point balance after redemption.
created_at	DateTime	Not Null	-	CURRENT_TIMESTAMP	Timestamp of when the redemption record was created in the system.

Table 4.4.37: Data Dictionary - RewardRedemption Entity

Leaderboard

Field Name	Data Type	Size/Constraint	Key	Default	Description
leaderboard_id	String	UUID (36)	PK	Auto-generated	Unique identifier for each leaderboard record.
member_id	String	UUID (36), Not Null	FK	-	References Member.member_id; identifies the member participating in the leaderboard.
total_focus_duration	Integer	Not Null	-	0	Total accumulated focus duration (in minutes) by the member.
rank	Integer	Not Null	-	-	Current ranking position of the member based on focus duration.
last_updated_at	DateTime	Not Null	-	CURRENT_TIMESTAMP	Date and time when the leaderboard record was last updated.

Table 4.4.38: Data Dictionary - Leaderboard Entity

EmotionDiary

Field Name	Data Type	Size/Constraint	Key	Default	Description
diary_id	String	UUID (36)	PK	Auto-generated	Unique identifier for each diary entry.
member_id	String	UUID (36), Not Null	FK	-	References Member.me

					mber_id; the member who created the diary entry.
emotion_tag _id	String	UUID (36), Nullable	FK	NULL	References EmotionTag. emotion_tag _id; identifies the tagged emotion for this diary entry.
title	String	50 chars, Not Null	-	-	Title of the diary entry.
content	Text	Optional	-	NULL	Main content or description of the diary entry.
mood_level	Integer	Range: 1–5, Not Null	-	5	Numerical scale representing the intensity of mood (e.g., 1 = very low, 5 = very high).
context_tags	String	255 chars, Nullable	-	NULL	Comma-separated tags or keywords describing context (e.g., “work, family, exam”).
sentiment_score	Float	Range: -1.0 to 1.0, Nullable	-	NULL	Sentiment analysis score derived from content (-1 = negative, 0 = neutral, 1 = positive).
created_at	DateTime	Not Null	-	CURRENT_TIMESTAMP	Date and time the

					diary entry was created.
is_momentary	Boolean	{true, false}	-	false	Indicates whether the diary entry was a quick, momentary note.
is_protected	Boolean	{true, false}	-	false	Marks the diary entry as private/protected (not shared).
status	Integer	Enum: {0 = inactive, 1 = active}	-	1	Status of the diary entry.

Table 4.4.39: Data Dictionary - EmotionDiary Entity

EmotionTag

Field Name	Data Type	Size/Constraint	Key	Default	Description
emotion_tag_id	String	UUID (36)	PK	Auto-generated	Unique identifier for each emotion tag.
name	String	100 chars, Not Null	-	-	Display name of the emotion (e.g., Happy, Sad, Angry).
color_code	String	7 chars, Not Null	-	-	Hexadecimal color code (e.g., #FFD700) associated with the emotion for UI visualization.
category	String	50 chars, Nullable	-	NULL	Emotion category/group (e.g., Positive,

					Negative, Neutral).

Table 4.4.40: Data Dictionary - EmotionTag Entity

DiaryAttachment

Field Name	Data Type	Size/Constraint	Key	Default	Description
diary_att_id	String	UUID (36)	PK	Auto-generated	Unique identifier for each diary attachment.
diary_id	String	UUID (36), Not Null	FK	-	References the EmotionDiary entry this attachment belongs to.
media_type	String	{image, video]	-	-	Type of media
media_url	String	255 chars, Nullable	-	NULL	Storage URL or file path to the uploaded media file.
file_name	String	150 chars, Nullable	-	NULL	Original name of the uploaded file.
file_size	Integer	Size in KB, Nullable	-	NULL	File size of the attachment.

Table 4.4.41: Data Dictionary - DiaryAttachment Entity

AdminReview

Field Name	Data Type	Size/Constraint	Key	Default	Description
review_id	String	UUID (36)	PK	Auto-generated	Unique identifier for each admin review record.

community_id	String	UUID (36), Nullable	FK	NULL	References the Community where the flagged content was posted (if applicable).
question_id	String	UUID (36), Nullable	FK	NULL	References ForumQuestion.question_id. Identifies the forum question that triggered the admin review.
post_id	String	UUID (36), Nullable	FK	NULL	References the Community Post that triggered the review.
message_id	String	UUID (36), Nullable	FK	NULL	References the Message that was flagged for review.
member_id	String	UUID (36), Nullable	FK	NULL	References the Member who created the flagged content.
admin_id	String	UUID (36), Not Null	FK	-	References the Admin who performed the review action.
content_type	String	50 chars, Not Null	-	-	Type of content under review (e.g., post, message, forum_answer).

trigger_type	String	50 chars, Not Null	-	-	Reason for triggering review (e.g., report, auto-flag, manual-check).
review_decision	Integer	Enum: {0=pending, 1=approved, 2=removed}	-	0	Decision outcome of the review.
admin_notes	Text	Optional	-	NULL	Notes or remarks provided by the reviewing admin.
auto_flagged	Boolean	{false, true}	-	-	Indicates if the content was auto-flagged by the system.
is_resolved	Boolean	{false, true}	-	false	Whether the review case has been resolved.
reported_at	DateTime	Not Null	-	CURRENT_TIMESTAMP	Date and time when the content was reported or flagged.
reviewed_at	DateTime	Nullable	-	NULL	Date and time when the admin reviewed the case.

Table 4.4.42: Data Dictionary - AdminReview Entity

AdminActionLog

Field Name	Data Type	Size/Constraint	Key	Default	Description
action_log_id	String	UUID (36)	PK	Auto-generated	Unique identifier for each admin

					action log entry.
admin_id	String	UUID (36), Not Null	FK	-	References the Admin who performed the action.
community_id	String	UUID (36), Nullable	FK	NULL	References the Community where the action occurred (if applicable).
question_id	String	UUID (36), Nullable	FK	NULL	References ForumQuestion.question_id. Identifies the forum question involved in the admin action.
post_id	String	UUID (36), Nullable	FK	NULL	References the Post involved in the action (if applicable).
message_id	String	UUID (36), Nullable	FK	NULL	References the Message affected by the action (if applicable).
action_type	String	50 chars, Not Null	-	-	Type of action performed (e.g., delete, warn, suspend, approve).
target_type	String	50 chars, Not Null	-	-	Type of content targeted (e.g., post, message, forum, member).

action_desc	Text	Optional	-	NULL	Detailed description or notes of the action taken.
ip_address	String	45 chars (IPv4/IPv6)	-	NULL	IP address from which the admin performed the action.
device_info	String	255 chars	-	NULL	Device details of the admin during the action.
performed_at	DateTime	Not Null	-	CURRENT_TIMESTAMP	Date and time when the action was carried out.

Table 4.4.43: Data Dictionary - AdminActionLog Entity

Report

Field Name	Data Type	Size/Constraint	Key	Default	Description
report_id	String	UUID (36)	PK	Auto-generated	Unique identifier for each report entry.
member_id	String	UUID (36), Not Null	FK	-	References the Member for whom the report is generated.
report_type	String	Enum: {daily, weekly, monthly}, Not Null	-	-	Specifies the reporting frequency/type.
report_notes	Text	Optional	-	NULL	Additional comments, remarks, or context about the report.

created_at	DateTime	Not Null	-	CURRENT_TIMESTAMP	Date and time when the report was generated.

Table 4.4.44: Data Dictionary - Report Entity

FocusReport

Field Name	Data Type	Size/Constraint	Key	Default	Description
focus_report_id	String	UUID (36)	PK	Auto-generated	Unique identifier for each focus report.
report_id	String	UUID (36), Not Null	FK	-	References the Report to which this focus report belongs.
time_spent_total	Integer	Not Null, ≥0	-	0	Total time (in minutes or seconds) spent in focus sessions within the report period.
productive_ratio	Decimal	(5,2), Not Null	-	0.00	Ratio of productive vs. total focus time (expressed as percentage or decimal fraction).
average_focus_time	Integer	Not Null, ≥0	-	0	Average duration of focus sessions (in minutes or seconds).
focus_session_count	Integer	Not Null, ≥0	-	0	Total number of focus

					sessions recorded in the report period.

Table 4.4.45: Data Dictionary - FocusReport Entity

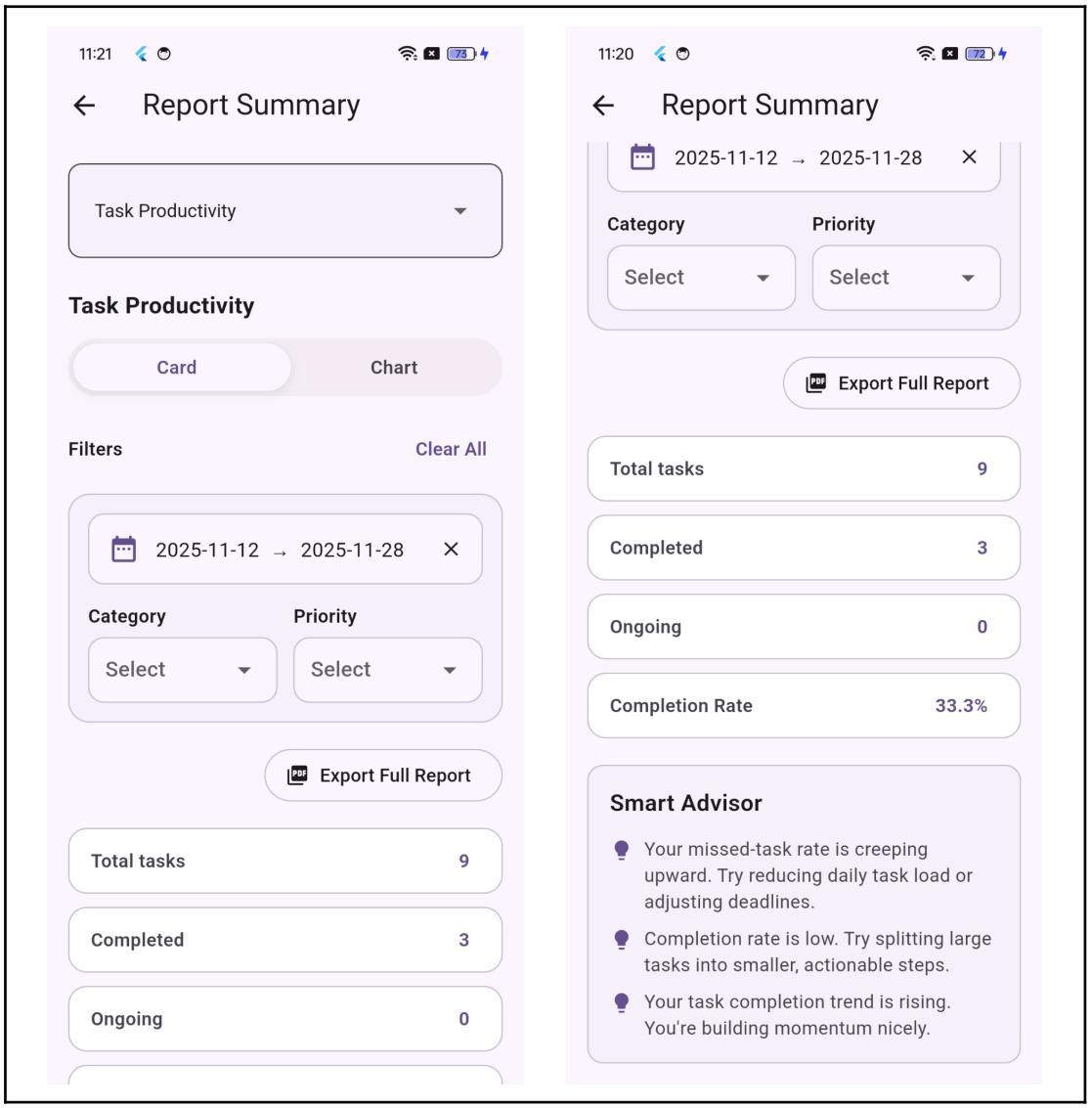
MoodReport

Field Name	Data Type	Size/Constraint	Key	Default	Description
mood_report_id	String	UUID (36)	PK	Auto-generated	Unique identifier for each mood report.
report_id	String	UUID (36), Not Null	FK	-	References the associated report record.
mood_summary	Text	Nullable	-	NULL	Summary of mood patterns or aggregated mood insights for the reporting period.
created_at	DateTime	Not Null	-	CURRENT_TIMESTAMP	Date and time the mood report was generated.

Table 4.4.46: Data Dictionary - MoodReport Entity

4.5 Reports Design

4.5.1 Productivity Report



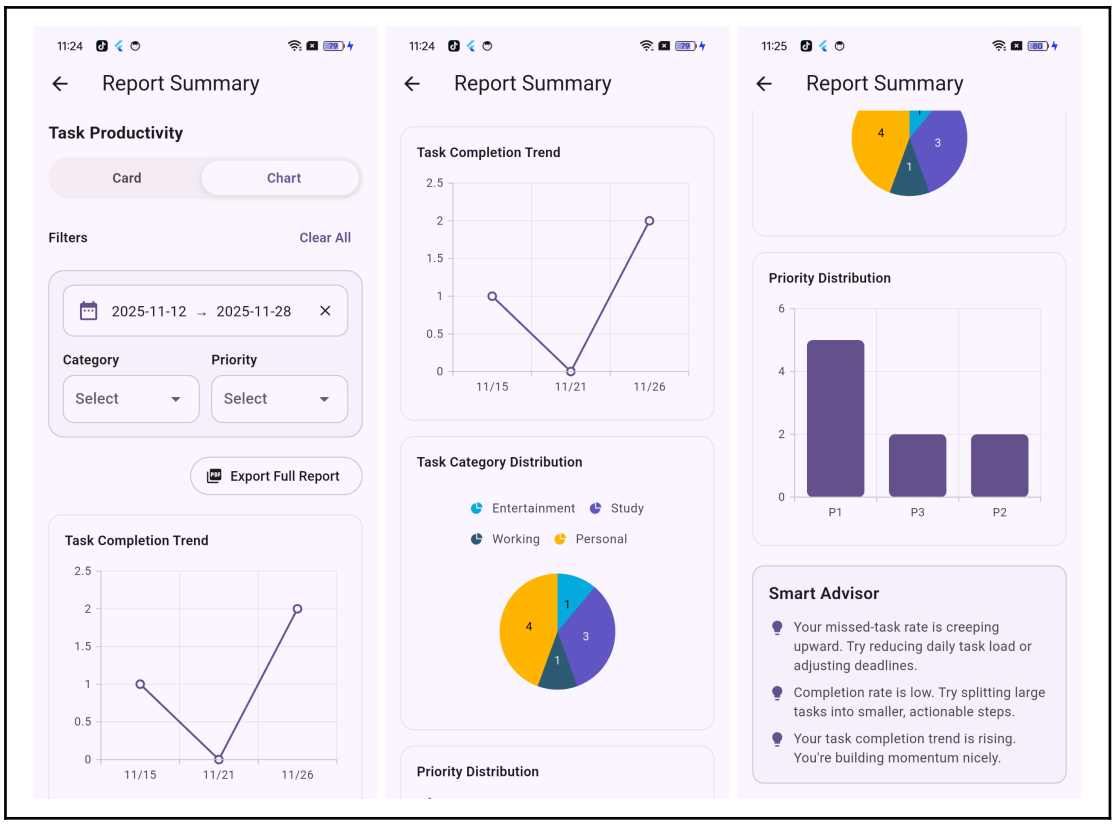


Diagram 4.5.1: Report Diagram - Productivity Report

4.5.2 Social Performance Report

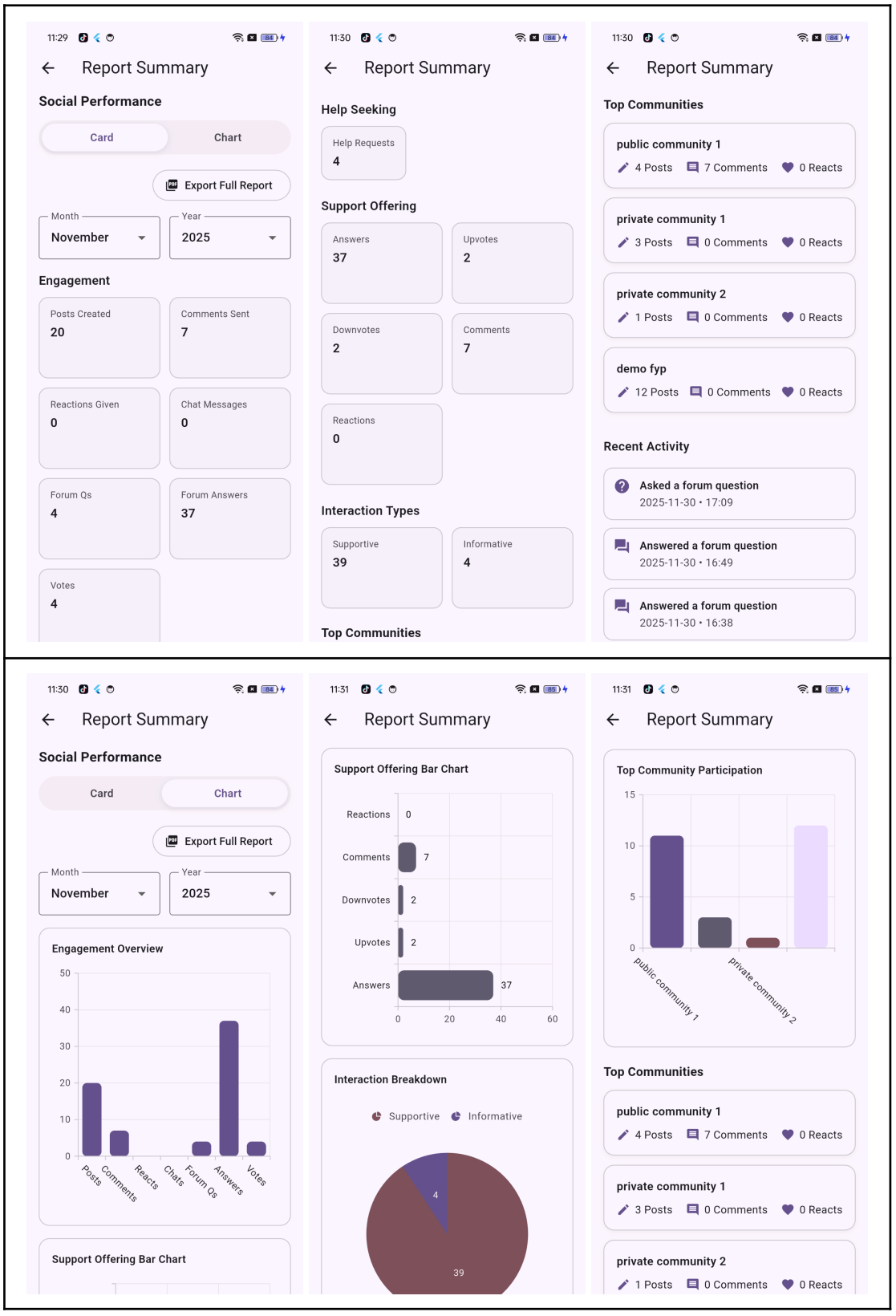


Diagram 4.5.2: Report Diagram - Social Performance Report

4.6 Process Design

Goal Assistance Module

Create Task and Subtask

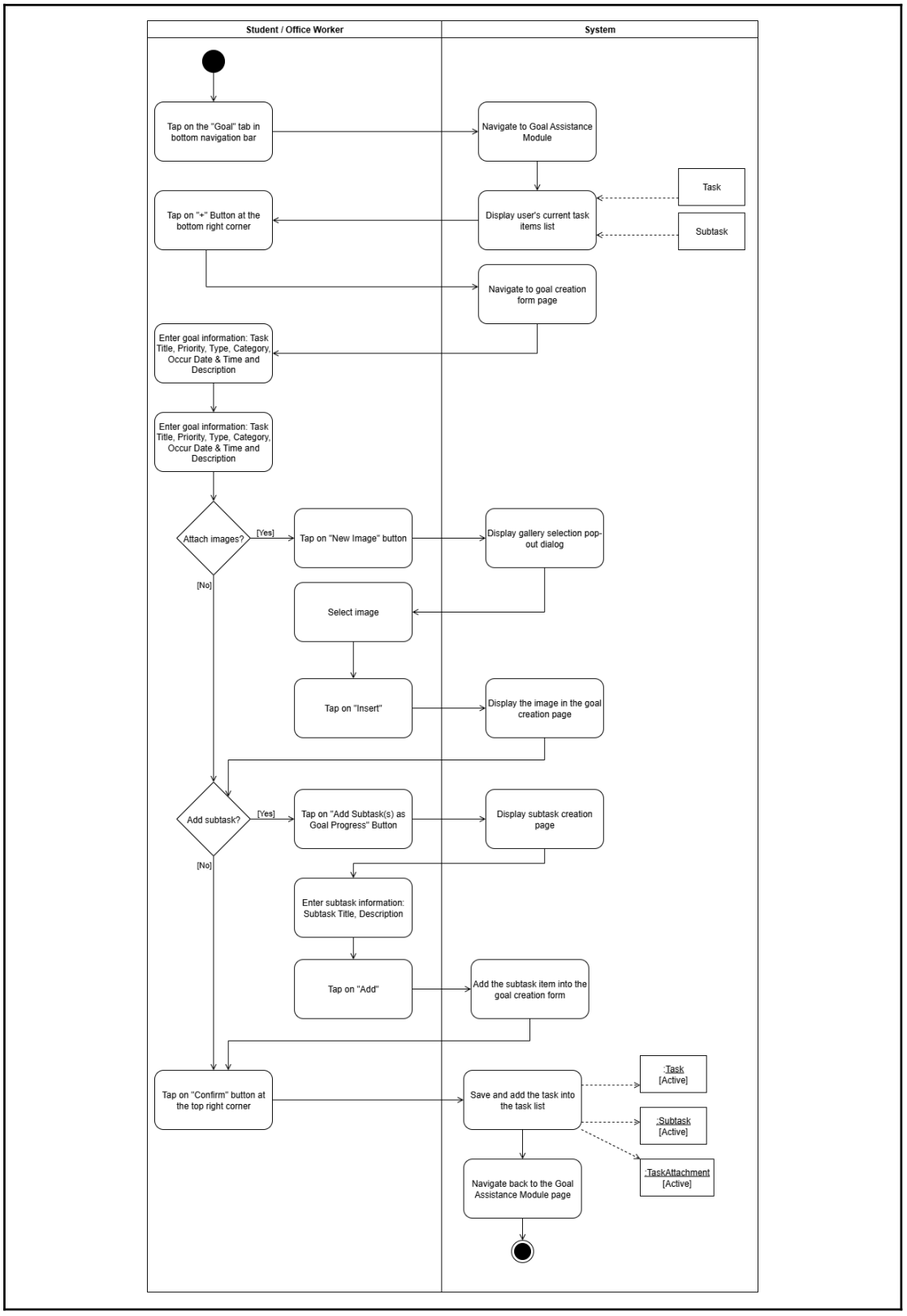


Diagram 4.6.1: Activity Diagram - Create Task and Subtask (Goal Assistance Module)

View Goal Notification

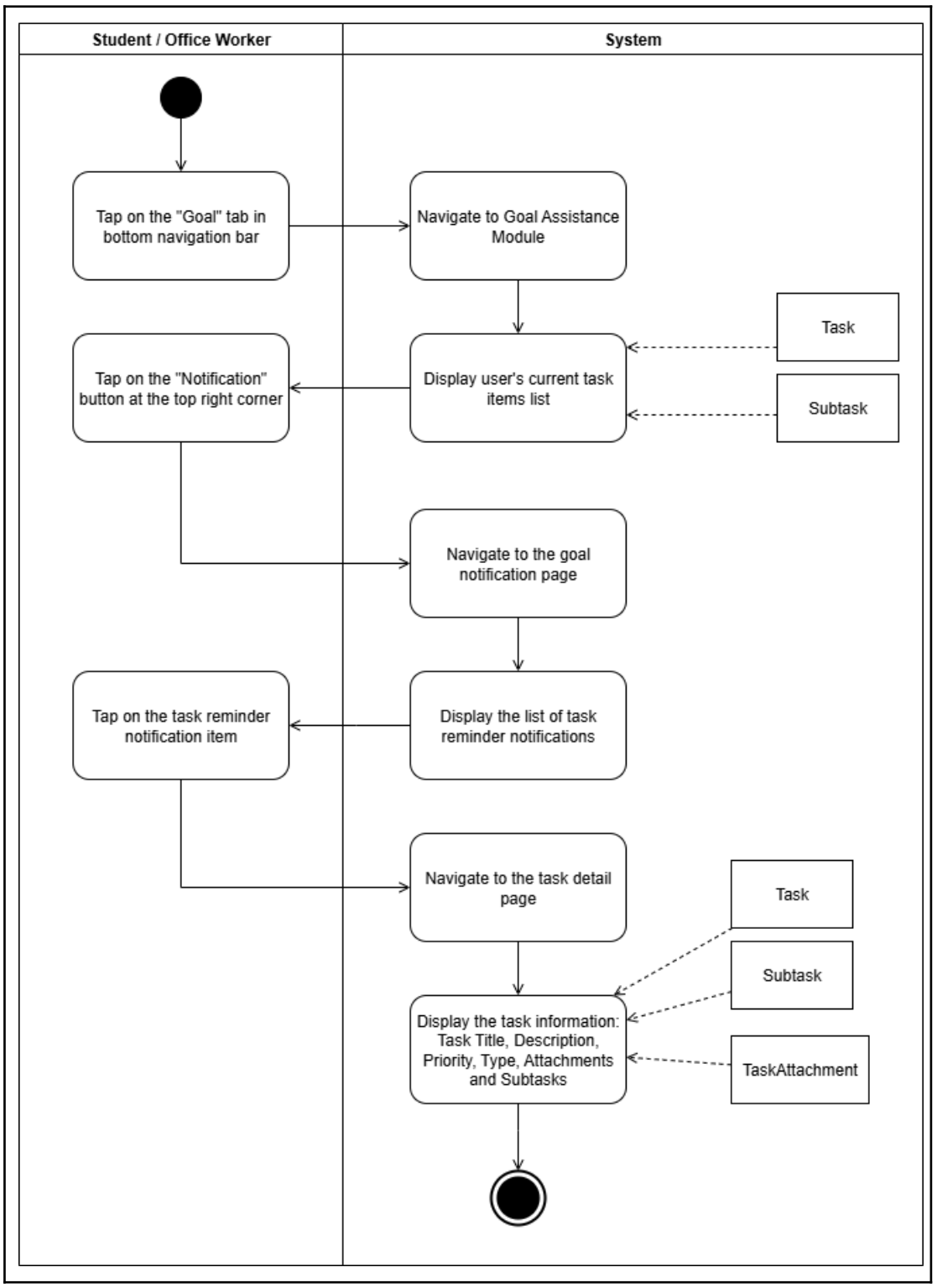


Diagram 4.6.2: Activity Diagram - View Goal Notification (Goal Assistance Module)

Manage Smart Advisor

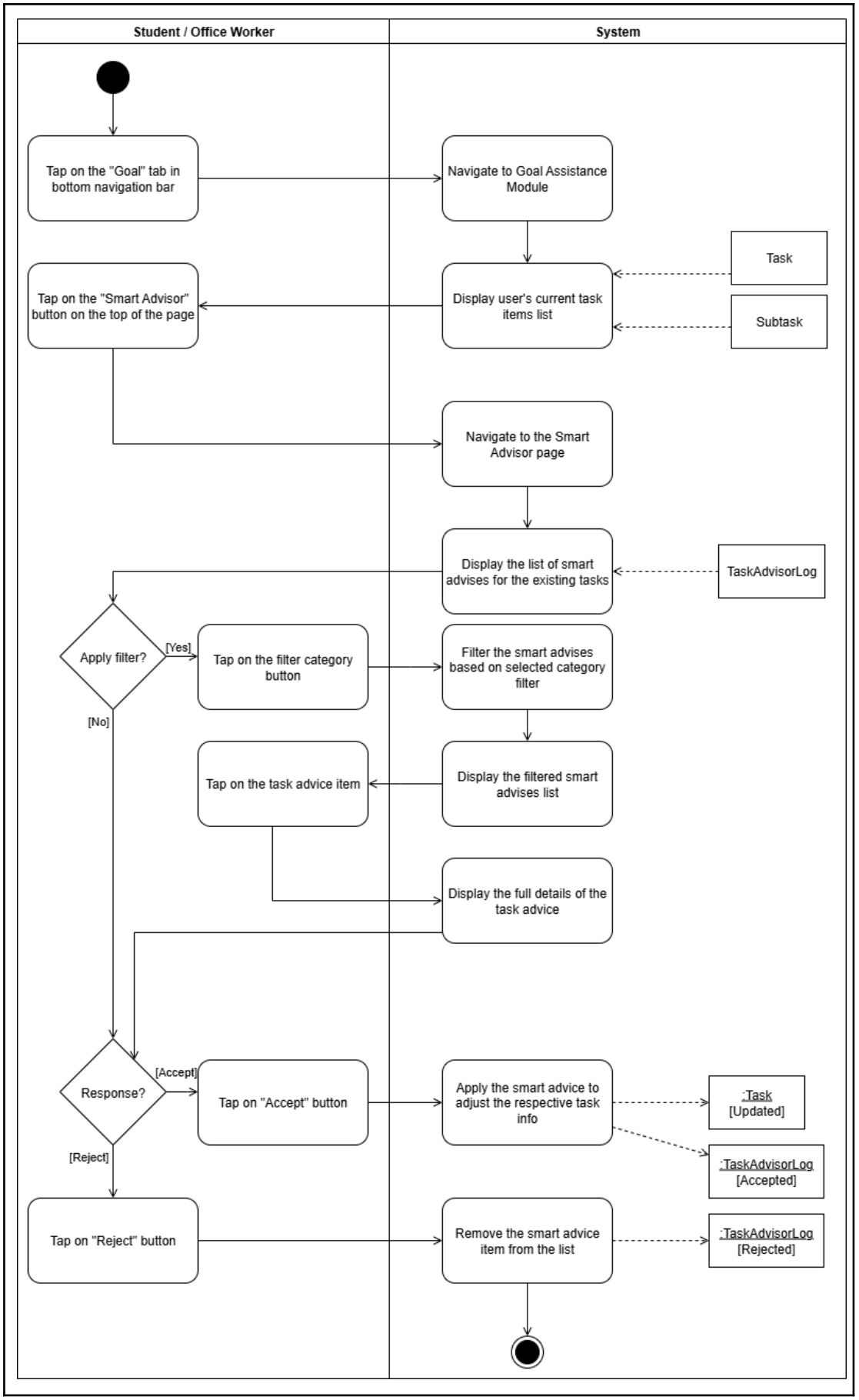


Diagram 4.6.3: Activity Diagram - Manage Smart Advisor (Goal Assistance Module)

Time Usage Tracker Module

View Time Usage Dashboard

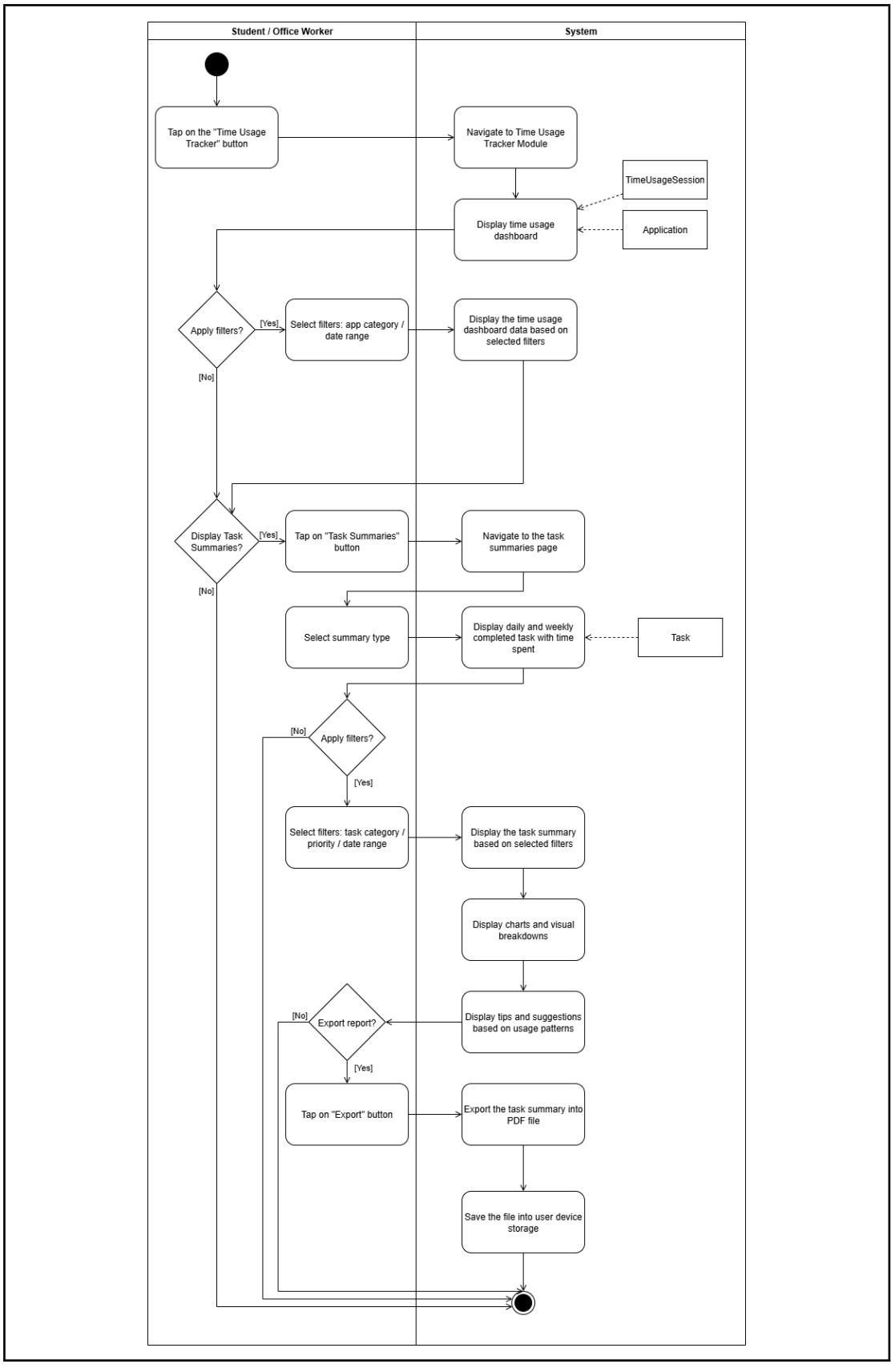


Diagram 4.6.4: Activity Diagram - View Time Usage Dashboard (Time Usage Tracker Module)

Manage NFC

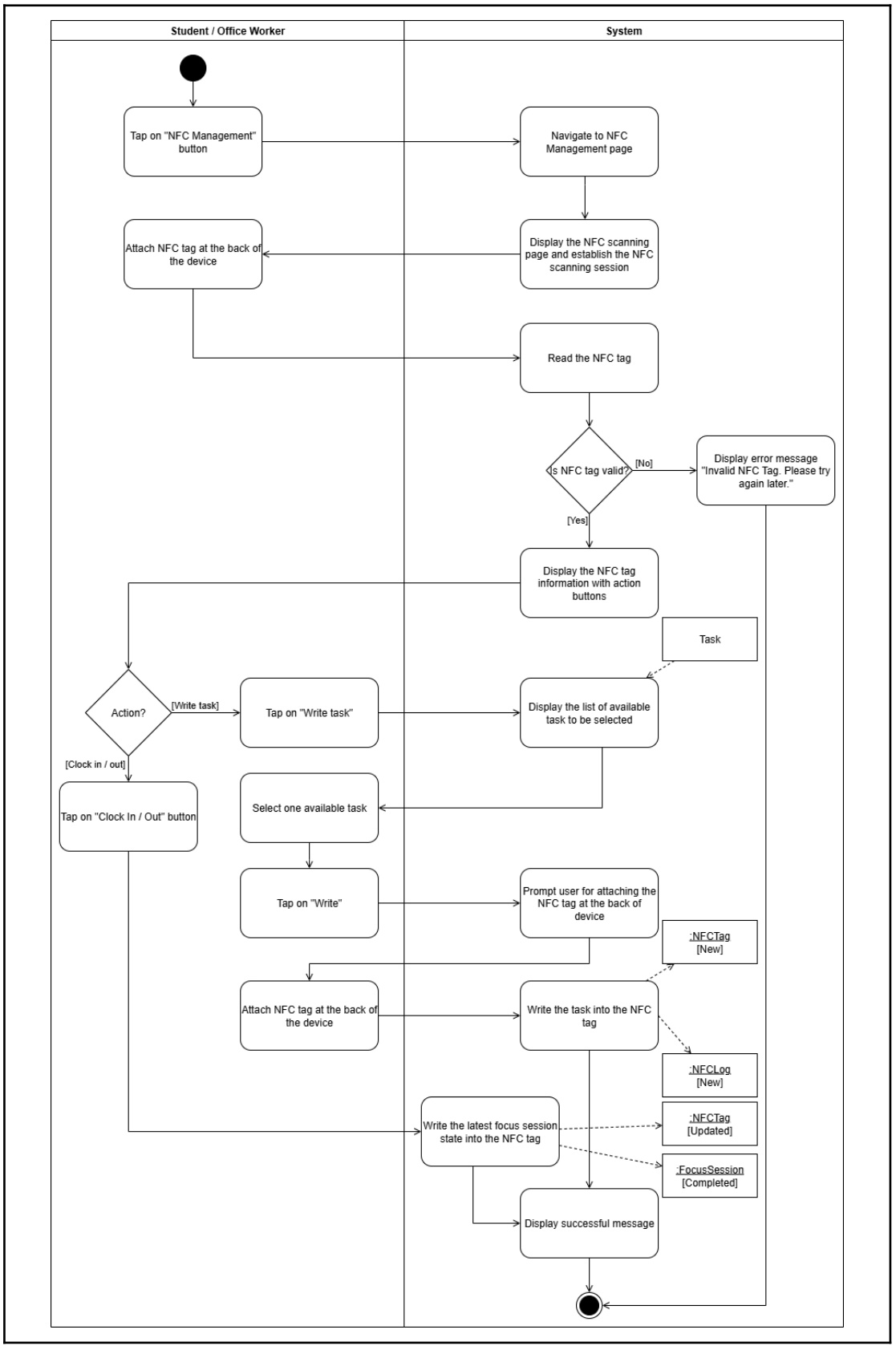


Diagram 4.6.5: Activity Diagram - Manage NFC (Time Usage Tracker Module)

Update Time Usage Tracker Settings

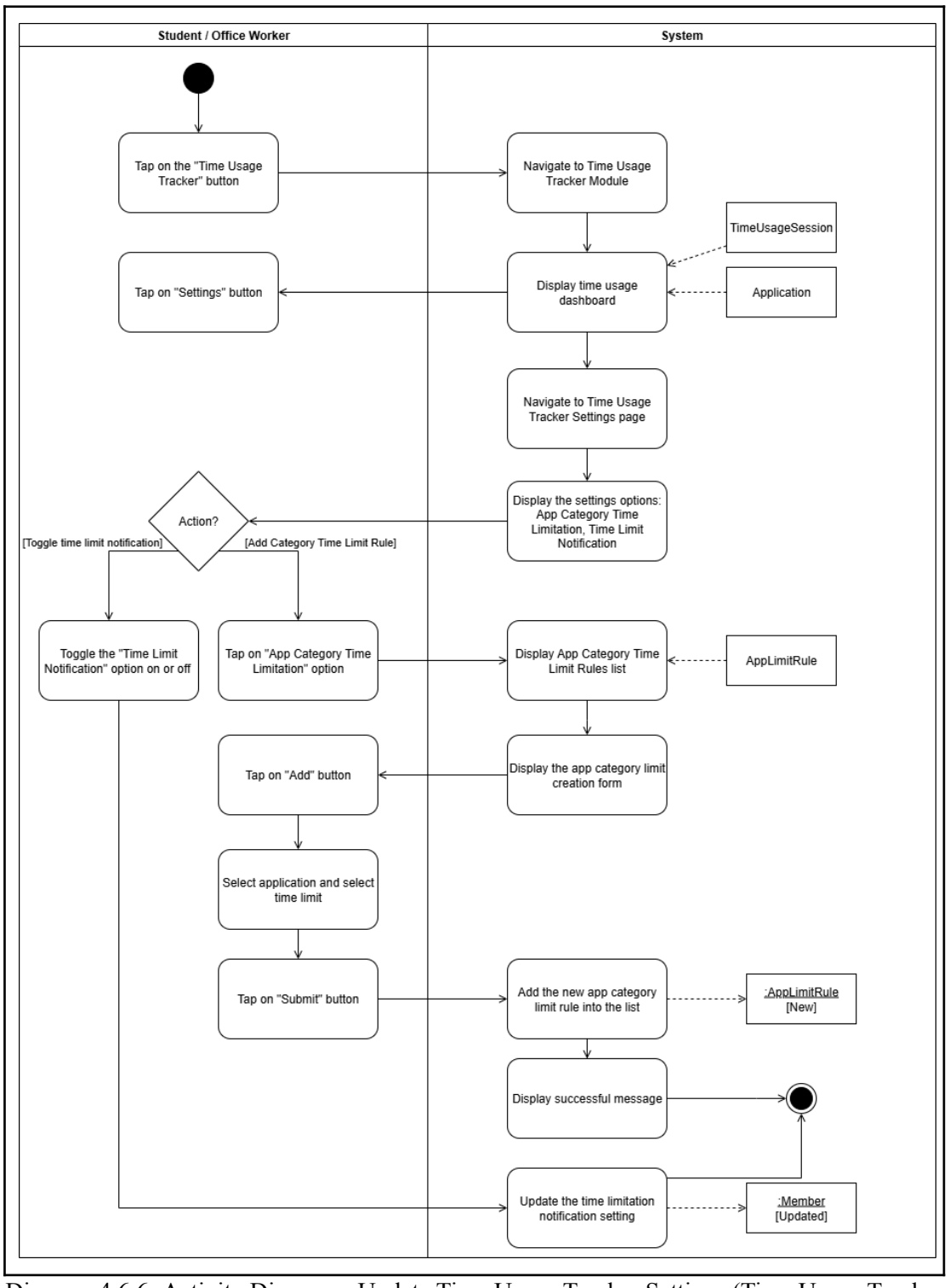


Diagram 4.6.6: Activity Diagram - Update Time Usage Tracker Settings (Time Usage Tracker Module)

Anonymous Community Module

Create Community and Share Post

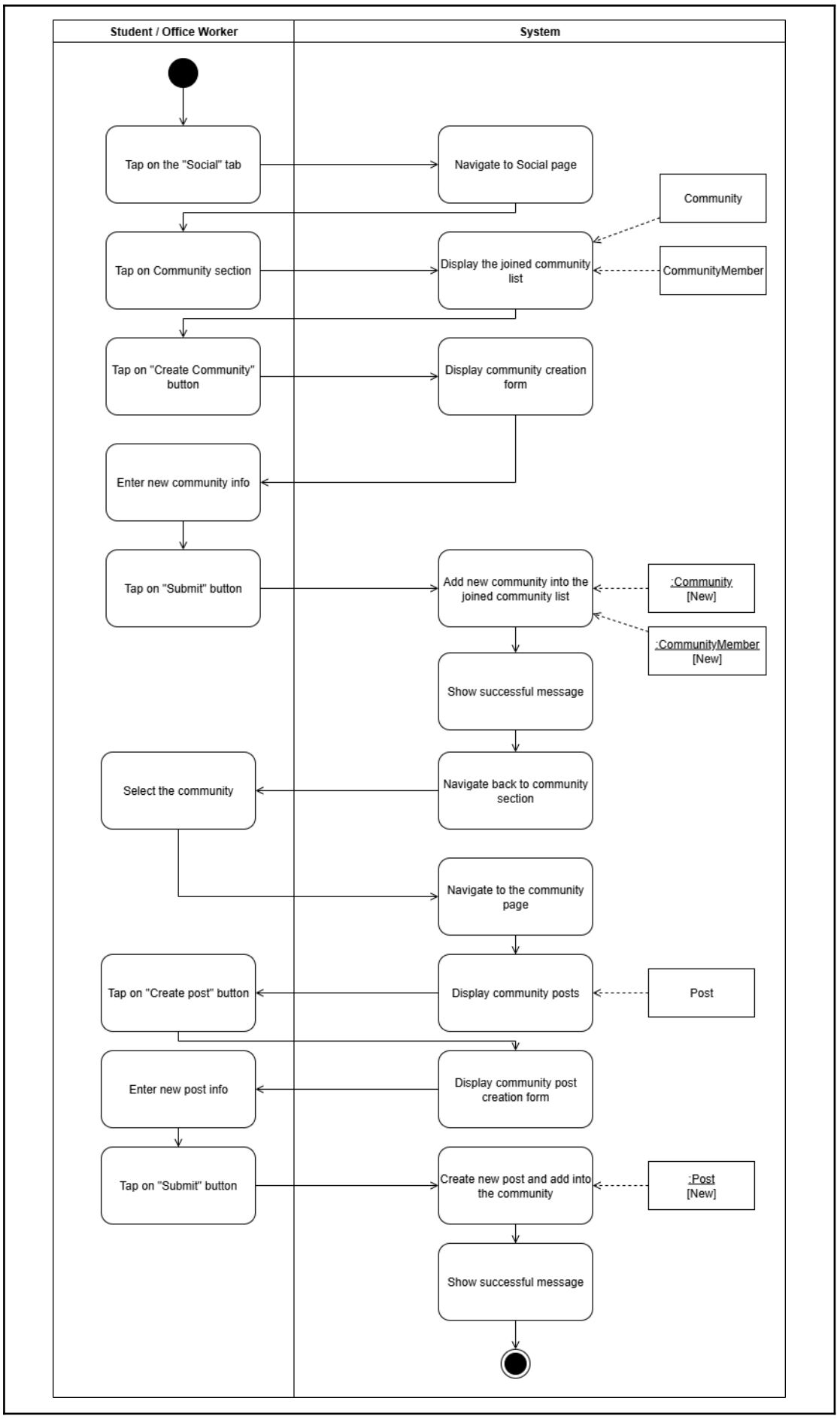

Student / Office Worker
System
Tap on the "Social" tab
Navigate to Social page
Community
Tap on Community section
Display the joined community list
CommunityMember
Tap on "Create Community" button
Display community creation form
Enter new community info
Tap on "Submit" button
Add new community into the joined community list
:Community [New]
:CommunityMember [New]
Show successful message
Select the community
Navigate back to community section
Navigate to the community page
Tap on "Create post" button
Display community posts
Post
Enter new post info
Display community post creation form
Tap on "Submit" button
Create new post and add into the community
:Post [New]
Show successful message

Diagram 4.6.7: Activity Diagram - Create Community and Share Post (Anonymous Community Module)

Create Forum and Post Forum Question

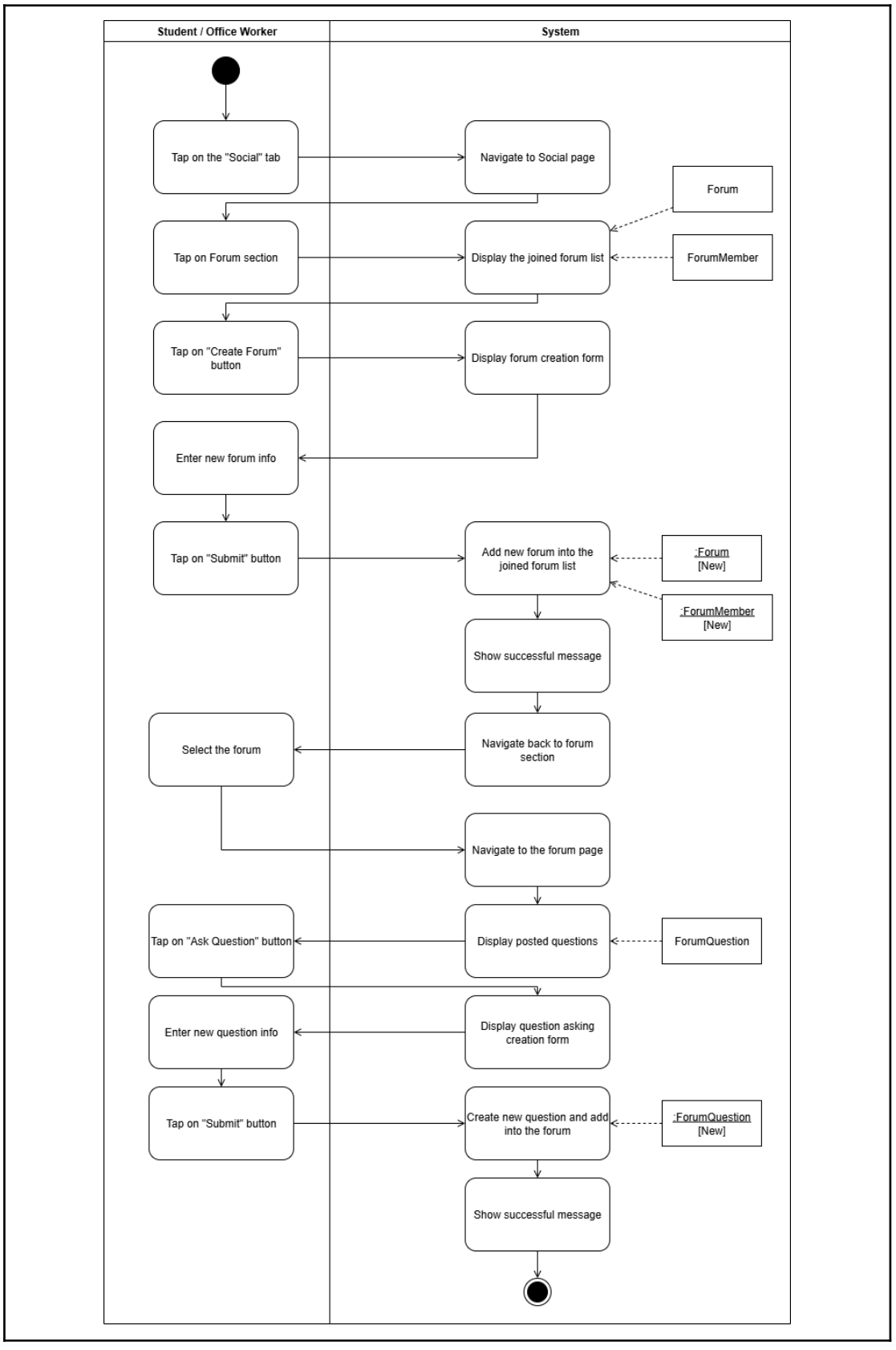


Diagram 4.6.8: Activity Diagram - Create Forum and Post Forum Question (Anonymous Community Module)

React and Answer Forum Question

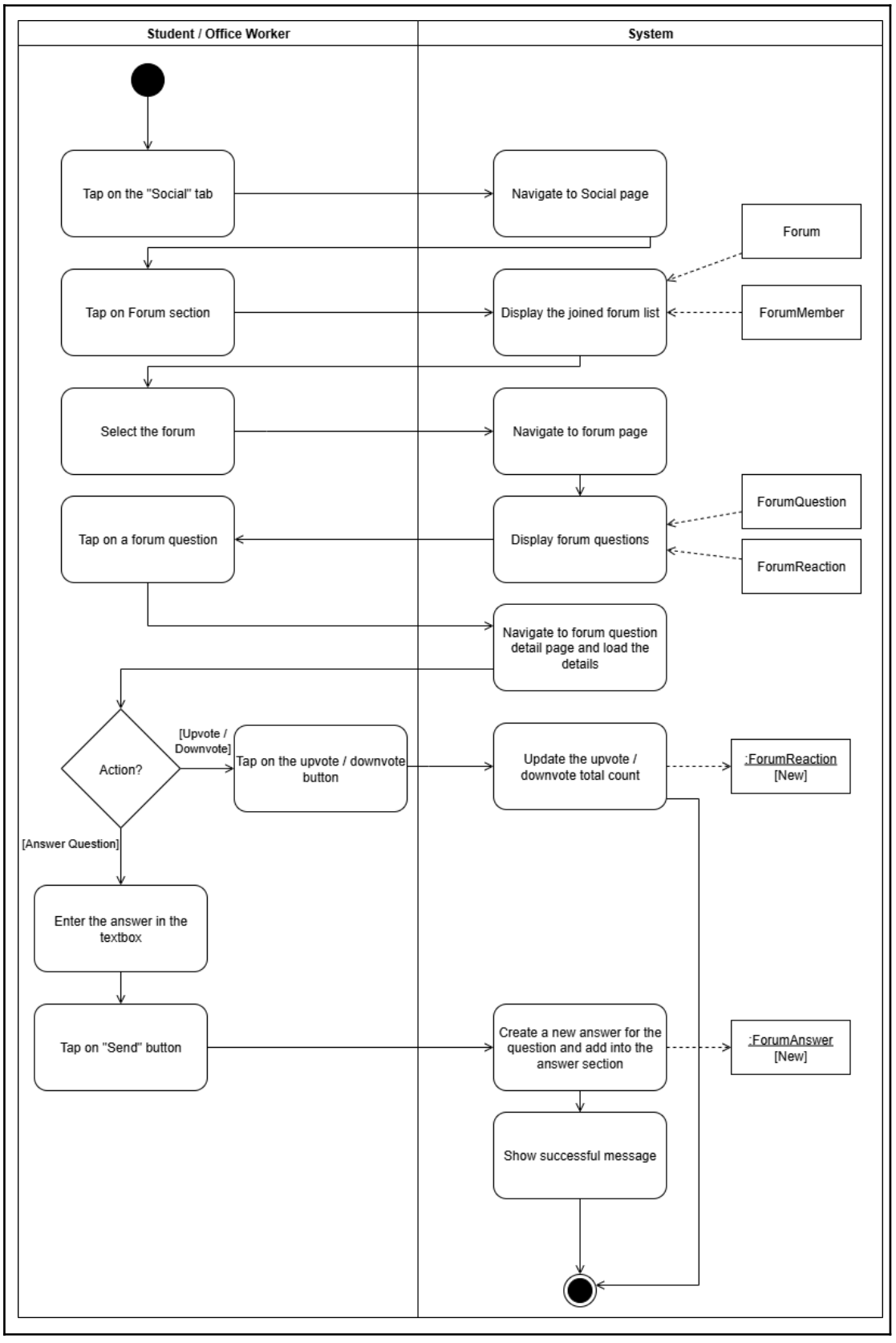


Diagram 4.6.9: Activity Diagram - React and Answer Forum Question (Anonymous Community Module)

Start Group Chat

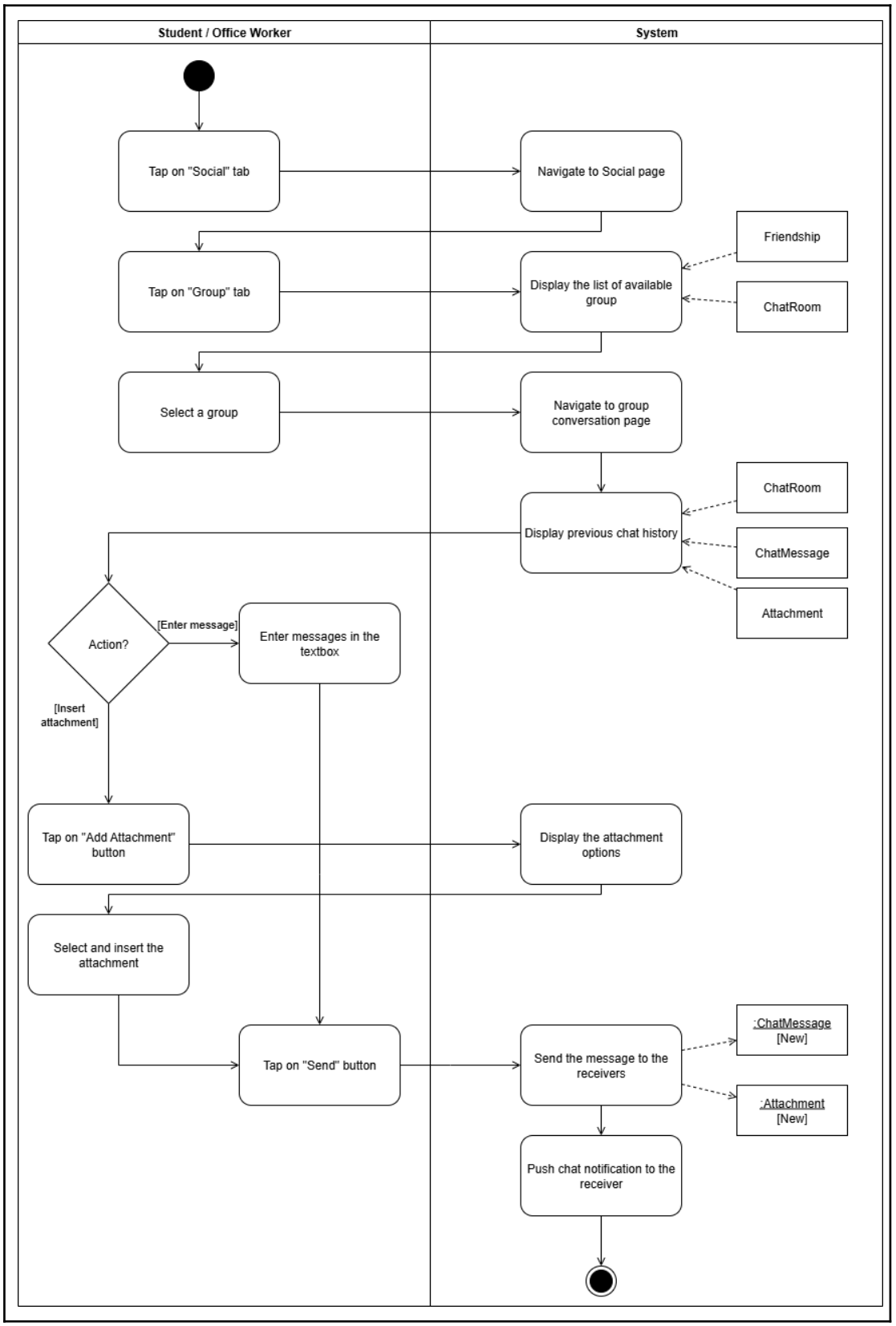


Diagram 4.6.10: Activity Diagram - Start Group Chat (Anonymous Community Module)

Review Flagged Post

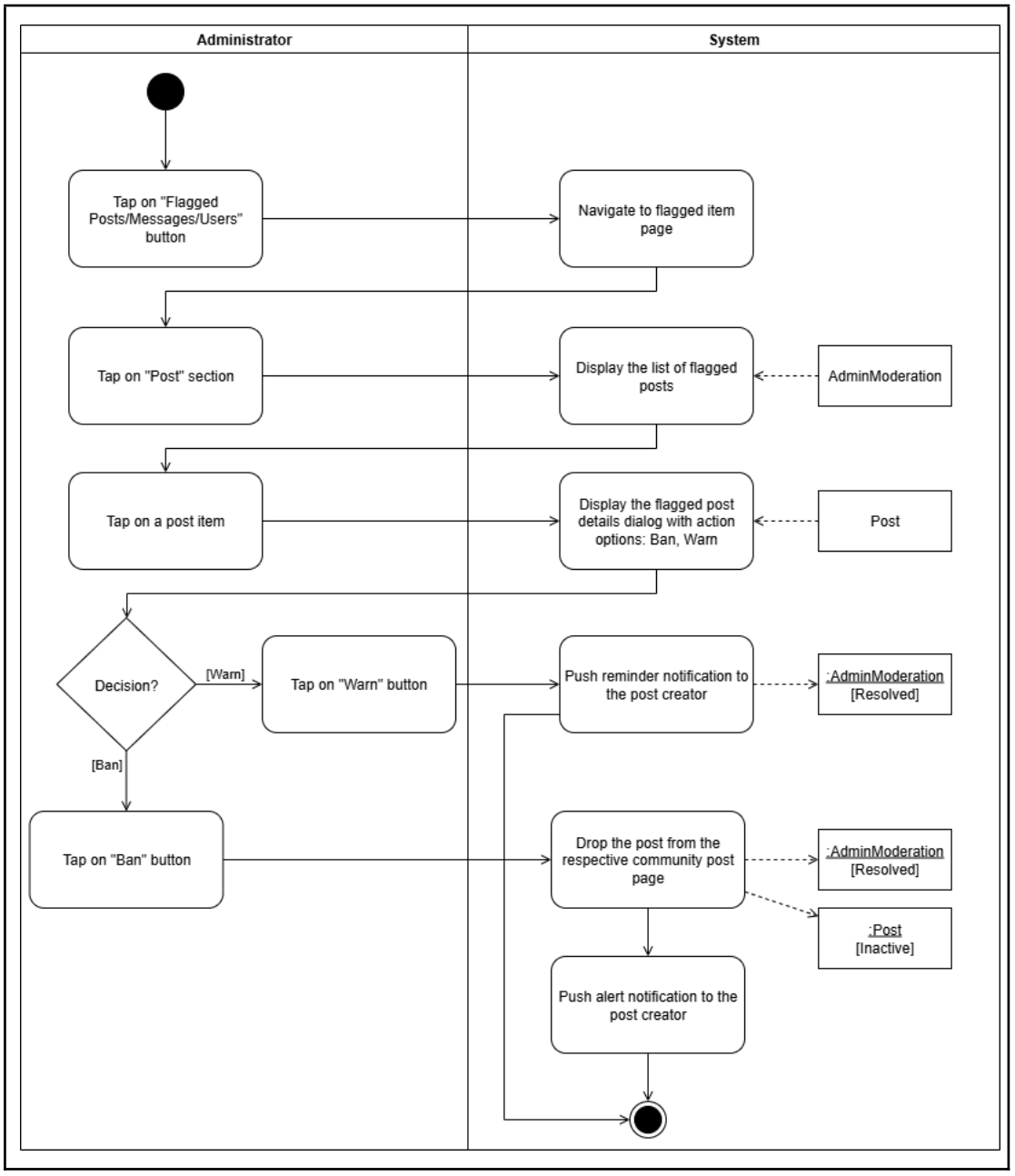


Diagram 4.6.11: Activity Diagram - Review Flagged Post (Anonymous Community Module)

Personal Chat Module

Personal Chat Workflow

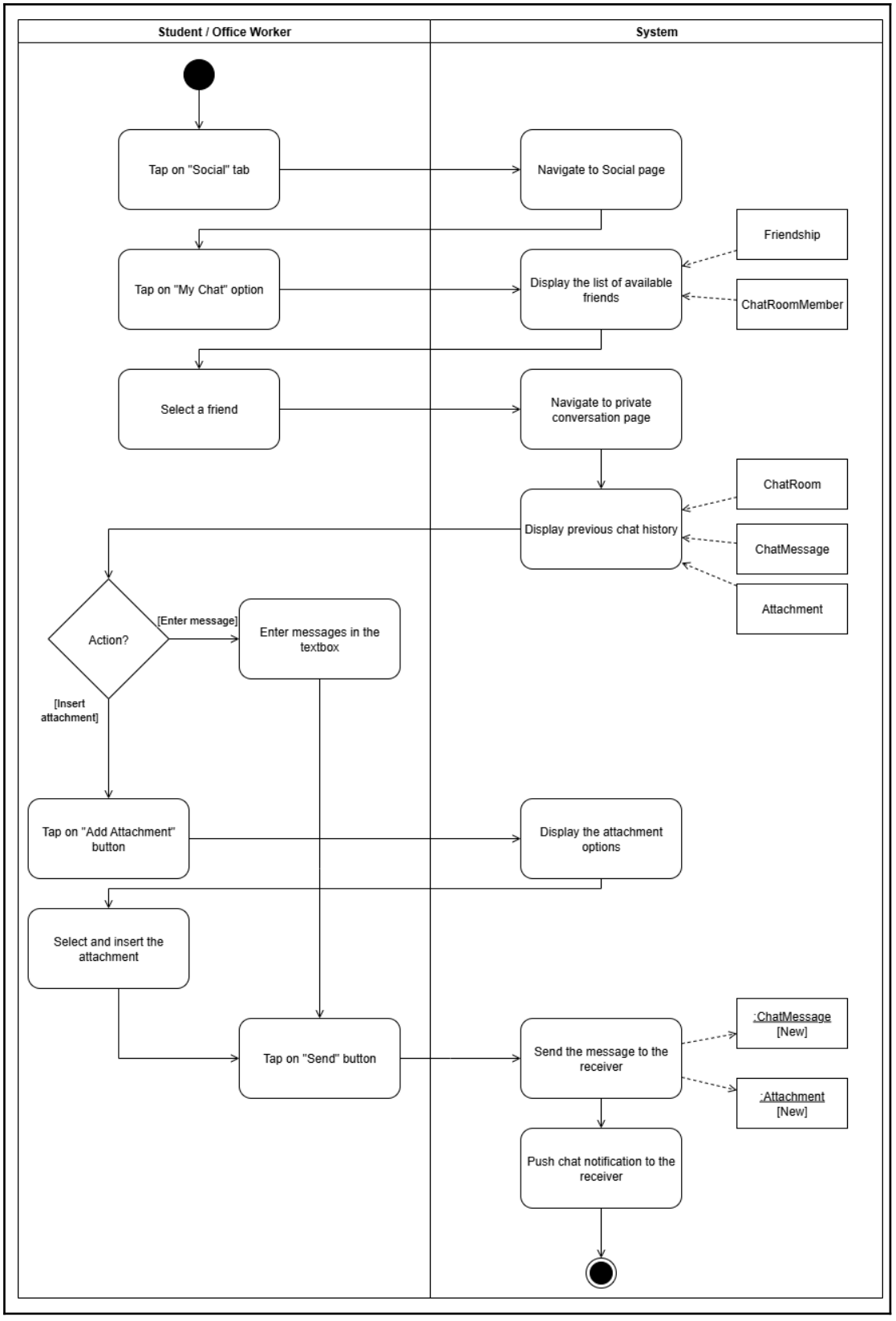


Diagram 4.6.12: Activity Diagram - Personal Chat Workflow (Personal Chat Module)

Review Reported Personal Chat Message

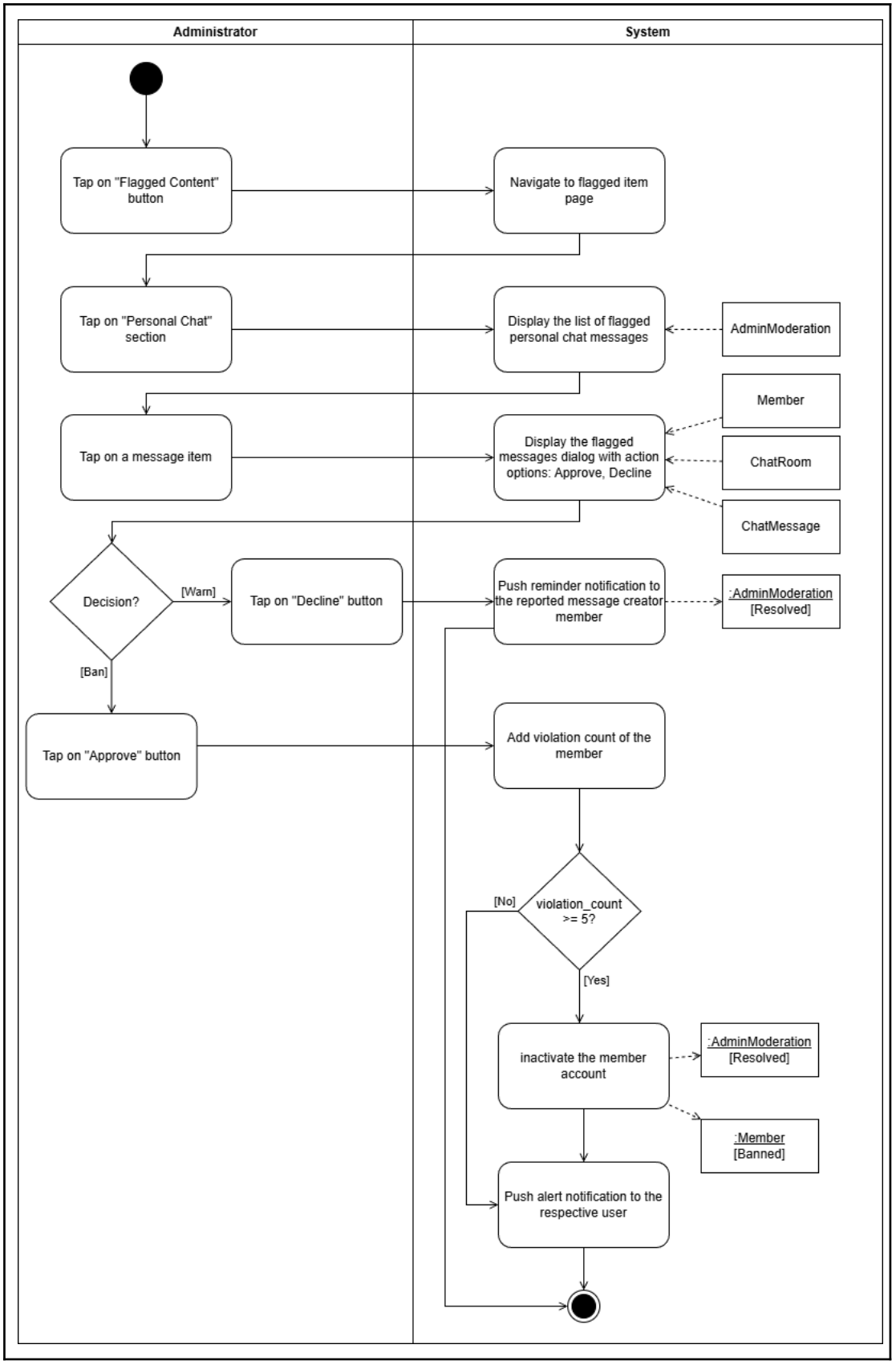


Diagram 4.6.13: Activity Diagram - Review Reported Personal Chat Message (Personal Chat Module)

Report Module

View Productivity Report

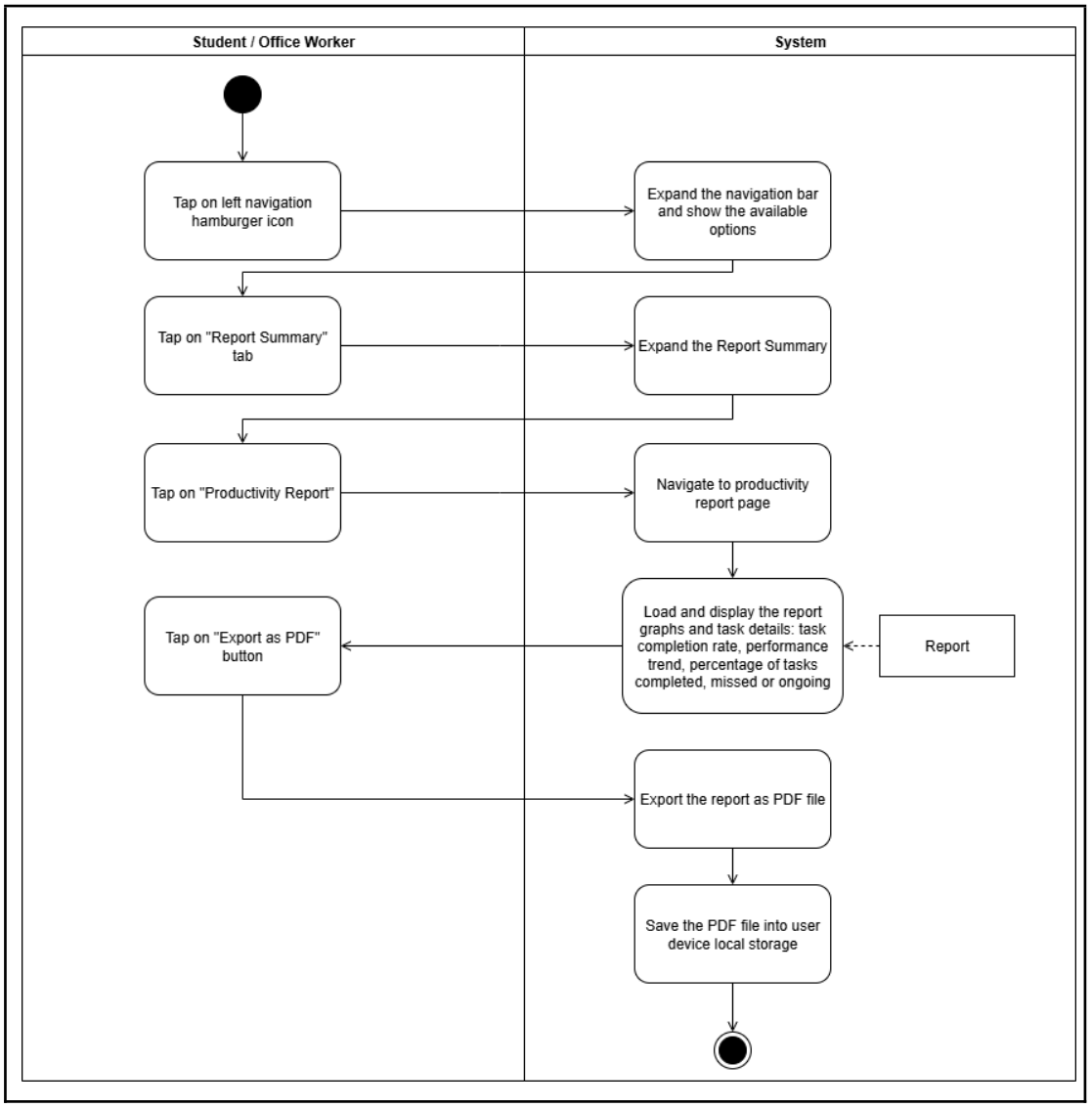


Diagram 4.6.14: Activity Diagram - View Productivity Report (Report Module)

View Social Performance Report

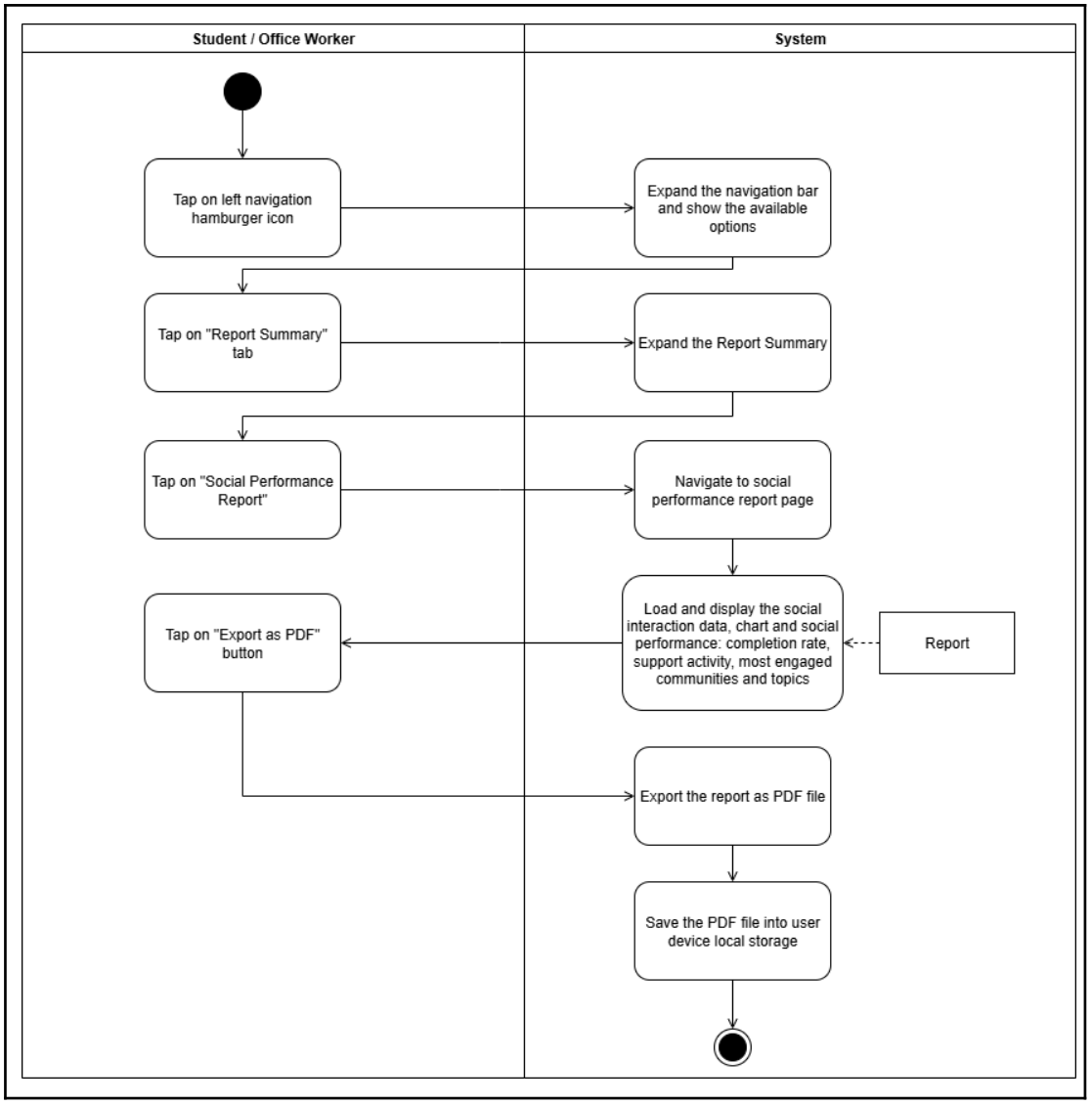


Diagram 4.6.15: Activity Diagram - View Social Performance Report (Report Module)

4.7 Software Architecture Design

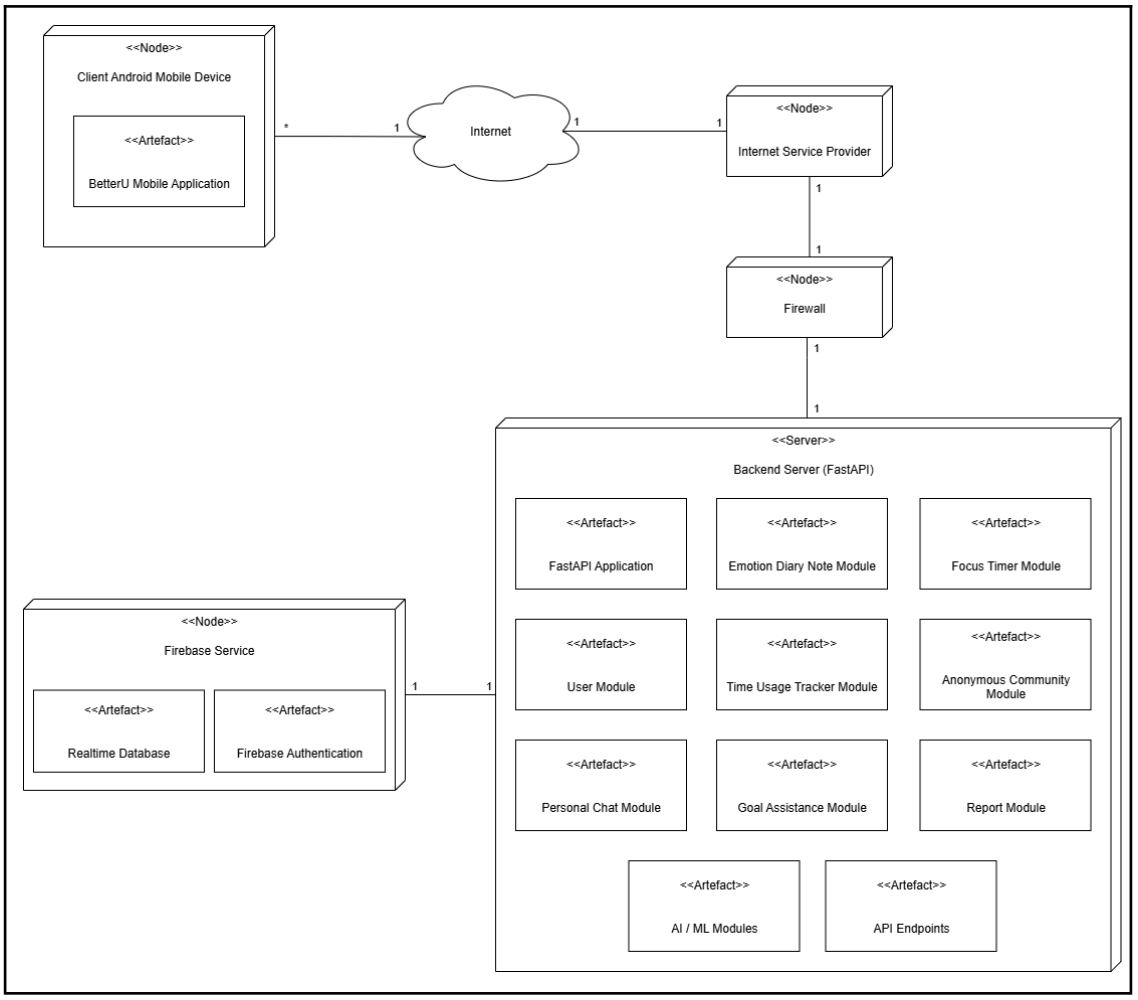


Diagram 4.7: Deployment Diagram of BetterU Mobile Application

4.8 AI Algorithms

Task Extraction from Natural Language

- **Algorithm**: Hybrid NLP Pipeline (spaCy + dateparser + SetFit)
- **Application**: This algorithm can automatically convert the free-form user text input such as "Finish report on next Monday at 9 AM" into a structured task format used by the BetterU goal assistance module task creation. It can process the natural language using spaCy to identify the verbs, actions, objects and linguistic dependencies. Meanwhile, it extracts the time expressions using dateparser and custom datetime normalization logic. Eventually, the extracted task phrase is classified using transformer-base SetFit models to determine the task category and priority level.
- **Synthetic Dataset**:
    - **Tasks dataset**: 600 tasks data records with task name, task type, priority and estimated duration.
- **Sample dataset content**:
    - Tasks Dataset

task_id	task_name	task_type	priority	estimated_duration	created_at
T001	Morning Jog	exercise	3	30	1/6/2024 7:00
T002	Read a Book	personal	2	45	1/6/2024 9:00
T003	Complete Assignment	study	5	120	1/6/2024 10:00
T004	Team Meeting	working	4	60	1/6/2024 11:00
T005	Watch Documentary	entertainment	2	90	1/6/2024 20:00
T006	Evening Walk	exercise	2	25	1/6/2024 18:00
T007	Write Journal	personal	1	20	1/6/2024 21:00
T008	Prepare Presentation	working	5	90	2/6/2024 9:00

T009	Study Math	study	4	60	2/6/2024 14:00
T010	Listen to Podcast	entertainment	1	30	2/6/2024 19:00

Table 4.8.1: Task Extraction from Natural Language - Tasks Dataset

Smart Task Advisor

- **Algorithm**: Collaborative Filtering Model - Alternating Least Squares (ALS) Matrix Factorization
- **Application**: The Smart Task Advisor will use the ALS-based recommendation model to learn the user-task interaction patterns and predict which tasks a user is most likely to complete next. The model analyzes the user's historical task performance (interaction scores) and generates the personalized task recommendations. This will be able to help the system identify tasks that match the user's habits, productivity patterns and preferred task categories.
- **Synthetic Dataset**:
  - **Users dataset**: 300 users with various ages, occupation, preferences and productivity styles.
  - **Tasks dataset**: 600 tasks data records with task name, task type, priority and estimated duration.
  - **Interaction dataset**: 6000+ interactions data records which records the user-task interaction such as completed, rescheduled, in-progress and skipped. Besides, the interaction score is labelled for each interaction record.
- **Sample dataset content**:
  - Users Dataset

user_id	**age**	**occupation**	**preferences**	**productivity_styles**	**task_preferences**
user_001	22	student	study;fitness	Morning-person	short-tasks
user_002	28	office worker	health;personal	Night-owl	high-priority-first
user_003	19	student	study;entertainment	Balanced	long-tasks
user_004	24	office worker	fitness;working	Morning-person	flexible

user_005	21	student	study;health	Night-owl	short-tasks
user_006	27	office worker	personal;entertainment	Balanced	high-priority-first
user_007	23	student	study;fitness	Morning-person	long-tasks
user_008	26	office worker	working;health	Night-owl	flexible
user_009	20	student	study;personal	Balanced	short-tasks
user_010	25	office worker	fitness;entertainment	Morning-person	high-priority-first

Table 4.8.2: Smart Task Advisor - Users Dataset

- Tasks Dataset

task_id	task_name	task_type	priority	estimated_duration	created_at
T001	Morning Jog	exercise	3	30	1/6/2024 7:00
T002	Read a Book	personal	2	45	1/6/2024 9:00
T003	Complete Assignment	study	5	120	1/6/2024 10:00
T004	Team Meeting	working	4	60	1/6/2024 11:00
T005	Watch Documentary	entertainment	2	90	1/6/2024 20:00
T006	Evening Walk	exercise	2	25	1/6/2024 18:00
T007	Write Journal	personal	1	20	1/6/2024 21:00
T008	Prepare Presentation	working	5	90	2/6/2024 9:00

T009	Study Math	study	4	60	2/6/2024 14:00
T010	Listen to Podcast	entertainment	1	30	2/6/2024 19:00

Table 4.8.3: Smart Task Advisor - Tasks Dataset

- Interaction Dataset

interaction_id	**user_id**	**task_id**	**interaction_type**	**interaction_score**	**timestamp**
int_000001	user_001	T143	completed	5	13/7/2024 6:05
int_000002	user_001	T262	rescheduled	3	25/7/2024 7:02
int_000003	user_001	T547	rescheduled	3	26/9/2024 10:38
int_000004	user_001	T063	completed	5	20/6/2024 9:45
int_000005	user_001	T043	completed	5	13/6/2024 8:11
int_000006	user_001	T198	completed	5	9/8/2024 7:03
int_000007	user_001	T252	completed	5	24/7/2024 10:40
int_000008	user_001	T024	skipped	1	15/6/2024 8:41
int_000009	user_001	T530	completed	5	9/10/2024 8:12
int_000010	user_001	T246	completed	5	25/7/2024 6:38

Table 4.8.4: Smart Task Advisor - Interaction Dataset

Task Rescheduling Prediction

- **Algorithm**: Random Forest Classifier (with SMOTE balancing)
- **Application**: The algorithm uses a machine learning pipeline to estimate the probability of a task that will be delayed based on the user behavior patterns, fatigue signals, calendar pressure, time-of-day and historical task performance. It helps the system to predict whether the task is at risk of delay. If the probability has exceeded a

threshold such as 0.7, it will trigger the Task Rescheduling Engine to propose a more optimal timeslot.

- **Synthetic Dataset**:
    - Interactions with reschedules features: This dataset inherits the Interactions dataset and expands the attributes such as previous_task_outcome, scheduled_start_time, actual_start_time, delay_minutes and other scheduling event related attributes.
- **Interaction with Rescheduling Dataset in Text Format**:

```
interaction_id,user_id,task_id,interaction_type,interaction_score,timestamp,previous_task_outcome,scheduled_start_time,actual_start_time,delay_minutes,delay_status,reschedule_count,task_priority,user_engagement_score,day_of_week,time_of_day,task_duration_estimate,completion_rate_past_week,moving_avg_delay_7d
int_000005,user_001,T043,completed,5,2024-06-13 08:11:55,None,2024-06-13 08:11:55,2024-06-13 08:11:55,0,On-time,0,High,1.0,Thursday,Morning,40,0.93,7.1
int_003170,user_001,T034,completed,5,2024-06-14 06:48:47,Completed,2024-06-14 06:48:47,2024-06-14 06:48:47,0,On-time,0,Low,1.0,Friday,Morning,15,0.98,2.5
int_000008,user_001,T024,skipped,1,2024-06-15 08:41:23,Completed,2024-06-15 08:41:23,2024-06-15 08:41:23,0,Missed,0,Low,1.0,Saturday,Morning,60,0.85,6.1
int_000004,user_001,T063,completed,5,2024-06-20 09:45:06,Delayed,2024-06-20 09:44:06,2024-06-20 09:45:06,1,On-time,0,High,1.0,Thursday,Morning,30,0.97,2.4
int_003475,user_001,T091,completed,5,2024-06-21 06:33:55,Completed,2024-06-21 06:32:55,2024-06-21 06:33:55,1,On-time,0,Low,1.0,Friday,Morning,20,0.72,8.9
int_003584,user_001,T109,skipped,1,2024-06-25 06:18:18,Completed,2024-06-25 06:16:18,2024-06-25 06:18:18,2,Missed,0,Low,1.0,Tuesday,Morning,90,0.66,3.3
int_003495,user_001,T094,completed,5,2024-07-07 08:43:26,Delayed,2024-07-07 08:43:26,2024-07-07 08:43:26,0,On-time,0,Low,1.0,Sunday,Morning,15,0.86,3.8
int_000001,user_001,T143,completed,5,2024-07-13 06:05:16,Completed,2024-07-13 05:55:16,2024-07-13 06:05:16,10,Delayed,0,High,1.0,Saturday,Morning,25,0.95,3.5
int_003897,user_001,T169,postponed,2,2024-07-19 11:50:00,Completed,2024-07-19 11:50:00,2024-07-19 11:50:00,0,On-time,1,Low,1.0,Friday,Morning,90,0.79,2.7
int_000007,user_001,T252,completed,5,2024-07-24 10:40:56,Delayed,2024-07-24 10:30:56,2024-07-24 10:40:56,10,Delayed,0,High,1.0,Wednesday,Morning,60,0.63,1.4
```

Table 4.8.5: Task Rescheduling Prediction - Interaction with Rescheduling Dataset

Time Optimization and Adaptive Scheduling

- **Algorithm**: Hybrid ML-Assisted Rule-Based Scheduling Engine
- **Application**: This algorithm will optimize the user's daily schedule by first predicting which tasks are likely to be delayed using a Random Forest classifier. After the prediction, the tasks which have a high delay probability will be passed into a rule-based scheduling engine. It will respect the user-specific constraints such as

working hours, busy windows, existing task blocks, required break buffers and individual task priorities when suggesting the new optimal timeslot to the users.

- **Synthetic Dataset**:
    - Schedule history: This dataset simulates the history and outcomes for each user's task assignment. It contains user id, task id, scheduled_start, scheduled_end, outcome and productivity score.
- **Sample dataset content**:
    - Schedule History

user_id	task_id	scheduled_start	scheduled_end	actual_start	actual_end	outcome	productivity_score
user_001	T043	13/6/2024 8:11	13/6/2024 8:51	13/6/2024 8:11	13/6/2024 8:51	completed	0.93
user_001	T034	14/6/2024 6:48	14/6/2024 7:03	14/6/2024 6:48	14/6/2024 7:03	completed	0.98
user_001	T024	15/6/2024 8:41	15/6/2024 9:41			missed	0.85
user_001	T063	20/6/2024 9:44	20/6/2024 10:14	20/6/2024 9:45	20/6/2024 10:15	completed	0.97
user_001	T091	21/6/2024 6:32	21/6/2024 6:52	21/6/2024 6:33	21/6/2024 6:53	completed	0.72
user_001	T109	25/6/2024 6:16	25/6/2024 7:46			missed	0.66
user_001	T094	7/7/2024 8:43	7/7/2024 8:58	7/7/2024 8:43	7/7/2024 8:58	completed	0.86
user_001	T143	13/7/2024 5:55	13/7/2024 6:20	13/7/2024 6:05	13/7/2024 6:30	completed	0.95
user_001	T169	19/7/2024 11:50	19/7/2024 13:20	19/7/2024 11:50	19/7/2024 13:20	delayed	0.79

user_001	T252	24/7/2024 10:30	24/7/2024 11:30	24/7/2024 10:40	24/7/2024 11:40	completed	0.63

Table 4.8.6: Time Optimization and Adaptive Scheduling - Schedule History Dataset

Toxicity Detection and Chat Moderation

- **Algorithm**: Hybrid Moderation Model (Rule-Based + BERT Classifier)
- **Application**: This algorithm will detect the toxic, offensive or harmful messages in the personal and group chat modules. It will use a multilingual rule-based filter to identify the explicitly profanity across English, Malay and Chinese. If there is no strong profanity found, a BERT-based machine learning model will be applied to perform contextual toxicity analysis to detect insults, threats, hate speech or obscene expressions even when bad words are not present.
- **Lexicon Datasets Used:**
    - English profanity dictionary: This file contains a list of English profanity, sexual content terms, hate expressions, insults, slurs and other offensive vocabulary. It is used by the rule-based filter to quickly detect explicit toxic content in English messages.
    - Malay profanity dictionary: This dataset contains the Malay language swear terms, insults, culturally offensive expressions and sensitive slang which are commonly used in Malaysia and Indonesia. It allows the system to filter the Malay toxic content even when the message uses local or region-specific profanity.
    - Chinese profanity dictionary: It is a large phrase-based lexicon of Simplified and Traditional Chinese profanity, insults, sexual terms and abusive expressions. The system will perform a substring matching for higher accuracy since Chinese toxicity often appears as full phrases instead of tokenized words.

4.9 Chapter Summary and Evaluation

This chapter has covered the main system design of the BetterU mobile application with support of useful explanation, diagrams and sample datasets. It mainly described both structural and behavioral aspects of the mobile application. For example, the sequence diagram, state chart diagram, activity diagram and user interface diagram has provided the users with a brief overview of the system processes and user interactions. Meanwhile, the data design was also documented through the class diagram, entity relationship diagram and data dictionary. This can ensure a clear understanding of the system entities and relationships offered. From the aspect of report design, the productivity report and social performance report were planned with detailed UI components and data granularity. Moreover, the deployment diagram and process design were also involved for providing an architectural view of how the application components are organized. Eventually, the AI algorithms that will be integrated into the BetterU mobile application are also demonstrated since they are used for providing intelligent and personalized features. In overall, the design has provided a comprehensive blueprint of BetterU mobile application requirements including functional and non0-functional requirements. Via the well-defined system design, the modularity, scalability and maintainability of the system can be promised.

Chapter 5

Implementation and Testing

5 Implementation and Testing

This chapter shows the implementation and testing of the BetterU mobile application. It describes how each module was developed using the selected technologies. This includes the integration of backend services, AI components and the mobile interface logic without mentioning the complex and chronological development process. The chapter also states all the test cases designed and executed to verify the module interactions and system behaviours. Each test case will be organized in a structured table for demonstrating the validation workflow of the implemented features against the project requirements.

5.1 Implementation and Coding

5.1.1 Speech-to-Text Processing

The speech-to-text feature is used in the Goal Assistance module to allow users to create tasks using voice input. For example, when a user says "Play badminton on next Wednesday from 3 PM to 5 PM", the system will convert the speech into text and later send it to the task extraction module to identify the activity, date and time details.

In order to achieve both real-time responsiveness and high transcription accuracy, the system applied a hybrid speech recognition pipeline:

- The offline **Vosk** model processes streaming audio chunks during recording in order to provide low-latency and real-time text previews.
- After the user finishes speaking, the online **Whisper** model will reprocess the full audio to generate a more accurate final transcription.

Vosk Warm-Up

```
vosk_model = None
vosk_ready = False
_warmup_started = False

def warmup_vosk():
    global vosk_model, vosk_ready, _warmup_started
    if _warmup_started:
        return
    _warmup_started = True

    print("[VOSK] Background warm-up started...")
    try:
```

```
        vosk_model = vosk.Model("vosk-model-small-en-us-0.15")
        vosk_ready = True
        print("[VOSK] Model ready in background!")
    except Exception as e:
        print("[VOSK] Failed to load model:", e)
```

Table 5.1.1.1: Speech-To-Text Processing Code Implementation - Vosk Warm-Up

Partial Transcription Pipeline

```
def transcribe_partial(audio_path: str, session_id: str,
sample_rate: int = 16000):
    # read_audio_frames returns (bytes, sr)
    data_bytes, sr = read_audio_frames(audio_path)

    # Normalize and ensure correct byte-order & length for VAD
    norm_bytes = normalize_audio_pcm16(data_bytes, in_rate=sr,
out_rate=sr)

    # ensure frames are multiples of frame_length for VAD
    frame_duration = 30
    frame_length = int(sr * frame_duration / 1000) * 2
    if len(norm_bytes) % frame_length != 0:
        pad_len = frame_length - (len(norm_bytes) % frame_length)
        norm_bytes = norm_bytes + (b"\x00" * pad_len)

    # create/get recognizer with actual sr
    rec = get_or_create_recognizer(session_id, sr)

    # VAD check on normalized bytes
    if not is_speech(norm_bytes, sr):
        return {"text": ""}

    # Feed the normalized bytes to vosk
    if rec.AcceptWaveform(norm_bytes):
        res = json.loads(rec.Result())
        text = res.get("text", "")
    else:
        res = json.loads(rec.PartialResult())
        text = res.get("partial", "")

    print(f"Vosk partial transcription for {session_id}: {text}")
    return {"text": text}
```

Table 5.1.1.2: Speech-To-Text Processing Code Implementation - Partial Transcription Pipeline

Whisper Language Detection

```
def detect_language(audio_path: str) -> str:
    audio = whisper.load_audio(audio_path)
    audio = whisper.pad_or_trim(audio)
    mel = whisper.log_mel_spectrogram(audio).to(model.device)

    _, probs = model.detect_language(mel)
    detected_lang = max(probs, key=probs.get)
    print(f"Detected language: {detected_lang}")

    if detected_lang in ["en", "zh"]:
        return detected_lang
    return "en"  # default to English
```

Table 5.1.1.3: Speech-To-Text Processing Code Implementation - Whisper Language Detection

Finalization Logic

```
def finalize_session(session_id: str):
    # Find recognizer matching this session id prefix
    recog_key = None
    for k in list(recognizers.keys()):
        if k.startswith(f"{session_id}-"):
            recog_key = k
            break

    final_text = ""
    if recog_key:
        final_text =
json.loads(recognizers[recog_key].FinalResult()).get("text", "")
        del recognizers[recog_key]
        print(f"Vosk finalized for {session_id}: {final_text}")
    else:
        print(f"No recognizer found for session {session_id}")

    session_audio_path = os.path.join(tempfile.gettempdir(),
f"{session_id}.wav")

    if not os.path.exists(session_audio_path):
        return {"final_text": final_text}

    print(f"Found audio for session {session_id}, proceeding with
Whisper transcription.")

    try:
        lang = detect_language(session_audio_path)
```

```
        whisper_text = transcribe_with_whisper(session_audio_path,
language=lang)
        print(f"Whisper transcription: {whisper_text}")
        return {"final_text": final_text, "refined_text":
whisper_text, "language": lang}
    except Exception as e:
        print("Whisper transcription error:", e)
        return {"final_text": final_text, "refined_text": "",
"language": "en"}
```

Table 5.1.1.4: Speech-To-Text Processing Code Implementation - Finalization Logic

5.1.2 Task Extraction

The task extraction feature will work together with the speech-to-text service to automatically understand and structure the user's task instructions. After the speech has been transcribed into text, this task extraction service will process the sentence using the combination of spaCy NLP, rule-based parsing and machine-learning classification to extract meaningful task details.

The system identifies key information such as:

- Task activity
- Date and time expressions
- Start and end datetime
- Task category
- Task priority

Task Phrase Extraction

```
def extract_task_phrase_from_doc(doc, time_token_idxs) -> str:
    TIME_CONNECTORS = {"from", "until", "between", "and",
"around", "about"}
    ABSOLUTE_EXCLUDE = {"am", "pm"} | TIME_CONNECTORS

    # Ending verbs to remove from task phrase
    ENDING_VERBS = {"end", "finish", "stop", "complete",
"conclude", "wrap"}

    root = None
    for tok in doc:
        if tok.head == tok:
            root = tok
            break
```

```
    if root is None:
        return doc.text.strip()

    task_tokens = set()
    task_tokens.add(root)

    def add(token):
        for t in token.subtree:

            # skip time tokens
            if t.i in time_token_idxs:
                continue
            # skip AM/PM and connectors entirely
            if t.lower_ in ABSOLUTE_EXCLUDE:
                continue
            # skip "to" ONLY if it is actually part of a time
expression
            if t.lower_ == "to" and (t.i-1 in time_token_idxs or
t.i+1 in time_token_idxs):
                continue
            # skip numbers that are part of time
            if t.pos_ == "NUM" and t.i in time_token_idxs:
                continue
            # skip "for" in "for X hours"
            if t.lower_ == "for" and re.search(r"\bfor \d+",
doc.text.lower()):
                continue
            # skip ending verbs ("end", "finish", etc.)
            if t.lemma_.lower() in ENDING_VERBS:
                continue

            task_tokens.add(t)

    # add main verb + complements
    add(root)

    # add coordinated verbs but NOT ending verbs
    for child in root.children:
        if child.dep_ in ("conj", "xcomp", "ccomp") and child.i
not in time_token_idxs:
            if child.lemma_.lower() not in ENDING_VERBS:
                add(child)

    # assemble phrase
    sorted_tokens = sorted(task_tokens, key=lambda t: t.i)
```

```
    phrase = " ".join(tok.text for tok in sorted_tokens)
    phrase = re.sub(r"\s+", " ", phrase).strip()

    # Remove trailing single junk words like "am", "finish", "end"
    trailing_junk = {"am", "pm"} | ENDING_VERBS
    words = phrase.split()

    if len(words) > 1 and words[-1].lower() in trailing_junk:
        words = words[:-1]

    phrase = " ".join(words).strip()

    return phrase
```

Table 5.1.2.1: Task Extraction Code Implementation - Task Phrase Extraction

Datetime Phrase Extraction

```
def extract_datetime_phrase(doc, time_token_idxs) -> str:
    # Special handling for "between X and Y"
    text = doc.text.lower()
    if "between" in text and "and" in text:
        return re.search(r"between (.+?) and (.+)", doc.text,
flags=re.I).group(0)

    # expand by including "from", "to", "until", "between", "and"
    connectors = {"from", "to", "until", "between", "and"}

    expanded = set(time_token_idxs)
    for i in list(time_token_idxs):
        # expand left
        if i > 0 and doc[i - 1].lower_ in connectors:
            expanded.add(i - 1)
        # expand right
        if i < len(doc) - 1 and doc[i + 1].lower_ in connectors:
            expanded.add(i + 1)

    time_token_idxs = expanded

    if not time_token_idxs:
        return ""

    sorted_ids = sorted(time_token_idxs)

    spans = []
    current = [sorted_ids[0]]
```

```
    for idx in sorted_ids[1:]:
        if idx == current[-1] + 1:
            current.append(idx)
        else:
            spans.append(current)
            current = [idx]
    spans.append(current)

    parts = []
    for span in spans:
        tokens = [doc[i].text for i in span]
        parts.append(" ".join(tokens))

    # keep exact structure
    phrase = " ".join(parts)
    return phrase.strip(" ,.")
```

Table 5.1.2.2: Task Extraction Code Implementation - Datetime Phrase Extraction

Main Structured Task Extraction Function

```
def extract_structured_task(clause: str, base_date: datetime |
None = None):
    if base_date is None:
        base_date = datetime.now()

    doc = nlp(clause)

    # mark all time-related tokens
    time_token_idxs = _get_time_token_indices(doc)

    # build task phrase and datetime phrase
    task_phrase = extract_task_phrase_from_doc(doc,
time_token_idxs)
    datetime_phrase = extract_datetime_phrase(doc,
time_token_idxs)

    from setfit import SetFitModel

    category_model =
SetFitModel.from_pretrained("app/core/task_extraction/model_task_c
ategory")
    priority_model =
SetFitModel.from_pretrained("app/core/task_extraction/model_task_p
riority")
```

```
    def classify_task(task_phrase):
        raw_cat = category_model(task_phrase)
        raw_pri = priority_model(task_phrase)

        # Convert numpy -> Python string
        category = str(raw_cat)

        # Convert tensor -> Python int
        if hasattr(raw_pri, "item"):
            priority = raw_pri.item()
        else:
            priority = int(raw_pri)

        return category, priority

    task_category, task_priority = classify_task(task_phrase)

    # parse datetime
    datetime_info = extract_datetime_info(datetime_phrase,
base_date)

    return {
        "task": task_phrase,
        "category": task_category,
        "priority": task_priority,
        "start": datetime_info["start"],
        "end": datetime_info["end"],
        "date": datetime_info["date"],
        "raw": datetime_info["raw"],
    }
```

Table 5.1.2.3: Task Extraction Code Implementation - Main Structured Task Extraction Function

5.2 Testing

5.2.1 Testing Strategies

Integration testing was selected as the main testing strategy for the BetterU mobile application because each of the modules is built from multiple interconnected components that must operate together as a unified process. Instead of testing the individual functions separately, this integration testing will focus on verifying the internal components within each module exchanged data correctly, triggered the correct logic and produced the expected outcomes when combined.

For the **Goal Assistance Module**, the integration testing was conducted across the complete task-processing pipeline. The **real-time transcription** generated by Vosk will need to be successfully refined by the Whisper model. Then, the resulting text will need to be correctly interpreted by the **NLP task extraction engine**. The structured task details were then validated for ensuring each of them can be applied into the task creation form without loss of information. The integration testing was also applied to verify that the **task editing** can correctly trigger the task delay prediction, rescheduling suggestions and task recommendations update.

In **Time Usage Tracker Module**, the integration testing will target how the **Android's UsageStatsManager is accessed through Flutter plugins** and how the provided usage data was correctly processed and displayed on the user interface. From the aspect of NFC task registration, this testing was also applied for verifying the **task or subtask deletion** will update the **NFC task registration**. The rule-based integrations were also tested. For example, the app category usage limit rules and whitelisted application settings are validated to make sure that the FCM notifications were triggered based on the settings.

When it comes to **Anonymous Community Module**, the integration testing can ensure that the **WebSocket-based group chat interaction** workflow was tested by examining the transmission of messages and notifications. This can make sure that the UI is updated in real time across multiple devices. Apart from that, the **cross-module attachments** such as tasks and community posts were validated to ensure they could be correctly retrieved and displayed in the attachment picker within the group chat view. The **admin content moderation interface** was also tested to verify that flagged content lists were correctly populated based on the backend data such as community post flagging, forum question flagging and group chat message flagging.

Eventually, the integration testing also targeted the **Personal Chat Module** where it verified that **friend request approvals or rejections** can correctly update the available friend list in

personal chat view. The **real-time messaging through WebSocket connections** was also tested to confirm that the sent messages were immediately broadcast to the intended recipients and reflected on the chat UI. Similarly, the integration tests were performed to validate the **attachment support from tasks and community posts** as well as the retrieval and display of flagged messages within the admin site.

In a nutshell, integration testing is the most appropriate strategy for this BetterU mobile application testing. The main reason is it not only examined the correctness of individual functions, instead, it also checked the flow and interactions between various functions across different modules. This can effectively ensure that each module functions reliably as a complete unit within a broader application environment.

5.2.2 Test Plan

Test Objectives

The objective of this test plan is to verify that each of the modules can operate correctly when its internal components are integrated and interacting with related subsystems. This testing is focusing on ensuring that all the workflows can function smoothly and produce the expected outputs under the real usage conditions. For example, the modules involved are Goal Assistance Module, Time Usage Tracker Module, Anonymous Community Module and Personal Chat Module.

Testing Process

Integration Testing

Integration testing was applied in the test plan for the modules. This is because each module consists of various components that must collaborate together as a complete workflow. The focus point of this testing is to validate the interactions between components such as speech-to-text integration with task extraction, NFC logic responding to task updates, mobile usage data retrieved through Flutter plugins synchronizing with the Flutter UI and WebSocket communication updating the interface in real time. This can make sure that the combined behaviors of all components are able to operate reliably and consistently while reflecting the user's actual conditions when dealing with the BetterU mobile application.

Test Plan Sequence

1. Goal Assistance Module

   Goal Assistance Module will be tested first because it acts as the core module which provides the core functionalities within the BetterU mobile application such as speech-to-text, task extraction and task creation workflows. Not only that, most of the

other modules also depend indirectly on the task data and task status updates generated here. For example, Time Usage Tracker Module's NBFC task registration and Chat's task attachment list.

2. Time Usage Tracker Module

   Time Usage Tracker Module will be tested after Goal Assistance, it consumes task or subtask information and includes the integrations with Android native services through Flutter plugins that access the UsageStatsManager API. Meanwhile, the testing will also involve the validation of NFC task behavior and daily app usage tracking.

3. Anonymous Community Module

   Anonymous Community Module is tested next as it relies on the anonymous identity settings and supports attachments from tasks and community posts. The WebSocket real-time updates and admin content moderation workflows are also verified here.

4. Personal Chat Module

   Personal Chat Module is tested last because it depends on the friend list updates, attachment integrations and WebSocket communication. During the testing of this module, it will ensure the real-time messaging and admin flagging features operate correctly.

Tested Items

- Goal Assistance Module:
  - Integration of Vosk partial transcription with Whisper final transcription
  - Whisper transcription output forwarded into the NLP task extraction pipeline
  - Extracted task details mapped into the task creation form
  - Task editing triggering task delay prediction update
  - Task editing triggering task rescheduling suggestions
  - Task editing triggering task recommendation generation
  - Updated task information reflected in attachment selection lists used by other modules
- Time Usage Tracker Module
  - Task or subtask deletion affecting NFC task registration updates
  - Task or subtask association reflected in NFC task status
  - Integration between Flutter and Android native services to obtain real device usage statistics
  - Daily app usage information displayed correctly in the dashboard

    - App category usage limit rule creation or update triggering notifications
    - Whitelisted app creation or deletion affecting notification filtering
- Anonymous Community Module
    - Anonymous identity settings applied to display anonymous names and avatars
    - WebSocket connection broadcasting messages to update the group chat interface in real time
    - Task and post attachments available in the attachment picker in group chat interface
    - Flagged posts or comments displayed correctly in the admin moderation panel
- Personal Chat Module
    - Friend request approval or rejection updating the friend list
    - WebSocket connection enabling real-time message sending and UI updates
    - Task and post attachments available in the attachment picker and sendable in chat
    - Flagged chat messages displayed correctly in the admin moderation panel

Testing Environment

Hardware and Devices

Mobile Phone	OPPO Reno 13 Pro	• Operating System: Android 15, ColorOS 15.0 • Storage: 512 GB • RAM: 16 GB • Processor: MediaTek Dimensity 8350 • Connectivity: Wi-Fi 802.11 a/b/g/n/ac/6e • NFC Support: Yes
	Realme GT Neo 3	• Operating System: Android 14, realme UI 5.0 • Storage: 128 GB • RAM: 8 GB • Processor: MediaTek Dimensity 8100 • Connectivity: Wi-Fi 802.11 a/b/g/n/ac/6 • NFC Support: Yes
Laptop	ASUS TUF Gaming A15 FA507NU	• Operating System: Windows 11 (64-bit) • Storage: 1.38 TB

		• RAM: 16 GB DDR5 (4800 MT/s) • Processor: AMD Ryzen 7 7735HS (3.20 GHz) • GPU: NVIDIA GeForce RTX 4050 Laptop GPU • Connectivity: Wi-Fi 6 (802.11ax), Ethernet • NFC Support: No

Table 5.2.2.1: Test Plan - Testing Environment Hardware Devices

Operating System

- Android 15, ColorOS 15.0
- Android 14, realme UI 5.0
- Windows 11 (64-bit)

Backend and Services

- FastAPI backend running in development environments
- Google Firestore for storing all the BetterU mobile application's data
- Firebase Cloud Messaging for push notifications on user's mobile devices
- WebSocket server used for group and personal chat messaging

Languages and Framework Environment

- Flutter 3.35.6 (channel stable)
- Dart 3.9.2
- Python 3.10.0

Network Environment

- Laptop connected to home mesh router via wired Ethernet
- Mobile devices (OPPO Reno 13 Pro & Realme GT Neo 3) connected to Wi-Fi on the same mesh network

Testing Constraints

- Speech-to-text results may vary due to background noise and microphone quality.
- App usage tracking accuracy depends on Android OS restrictions and manufacturer-specific limitations.
- NFC behaviour may vary based on device hardware supported tag model, sensitivity and tag placement.
- Real-time WebSocket features depend on network stability and router performance.

5.2.3 Test Cases and Results

Goal Assistance Module

Tester's Name	Lim Jun Wei	Date Tested	5-Dec-2025	Test Case (Pass/Fail/Not Executed)	Pass

S #	Prerequisites:
1	User is logged into the BetterU mobile app as member
2	Device microphone permission granted
3	Mobile device connected to same Wi-Fi network with the laptop connected network
4	Stable Internet connection

S #	Test Data
1	Speech input sentence = “Play badminton next Wednesday from 3 PM to 5 PM.”
2	Reminder time = 30 minutes before task start
3	Task description = “Weekly badminton practice with friends.”
4	Attachment (link) = "https://example.com/badminton-rules"
5	Attachment (image) = Any test image file
6	Subtask 1 title = “Prepare sports equipment”; Start: 2:00 PM; End: 2:30 PM
7	Subtask 2 title = “Warm-up session”; Start: 2:30 PM; End: 2:50 PM
8	Edited task time = Change to 4 PM - 6 PM
9	Edited category = Exercise
10	Edited priority = P3

Test Scenario	Creates a new task using the AI Task Extraction feature by recording a speech input, obtains the extracted task details, applies them into the task form, completes the remaining fields, and submits the task. Then, checks the Smart Advisor page for rescheduling suggestions and recommendations, edits the task to trigger updated predictions, and finally verifies that the created task is available in the Personal Chat attachment picker.

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Open the Goal Assistance module.	Goal Assistance dashboard is displayed successfully.	Goal Assistance dashboard was displayed successfully.	Pass
2	Tap the "+" button to create a new goal.	Create New Goal page is displayed with goal creation form.	Create New Goal page loaded with the goal creation form.	Pass
3	Tap on the "AI Task Extraction" card.	AI Task Extraction dialog is loaded.	AI Task Extraction dialog opened successfully.	Pass
4	Tap "Start Recording" to begin speech input.	Vosk begins showing partial real-time transcription.	Vosk began showing partial real-time transcription.	Pass
5	Speak the test sentence: "Play badminton next Wednesday from 3 PM to 5 PM."	Partial transcription updates dynamically during speech.	Partial transcription updated dynamically during speech input.	Pass
6	Tap "Stop Recording".	Whisper processes audio and returns final	Whisper processed the audio and returned a final accurate	Pass

		accurate transcription.	transcription.	
7	Tap "Extract Task Details".	NLP extracts task title, category, priority, start date/time and end date/time.	NLP successfully extracted title, category, priority, start time, and end time.	Pass
8	Tap "Apply" to insert extracted task details into the form.	Task creation form fields are automatically populated with extracted values.	Task form fields were populated automatically with extracted values.	Pass
9	Complete additional fields such as reminder time, description, links and attachments.	All entered details are accpeted and validated by the form.	All additional details (reminder, description, links, attachments) were accepted and displayed correctly.	Pass
10	Add subtasks with subtask title, description, start date/time and end date/time.	Subtasks are successfully added.	Subtasks were added successfully with valid date and time ranges.	Pass
11	Submit the task using the submit button.	Task is created successfully and appears in the task list with all details intact.	Task was created and appeared in the task list with complete details.	Pass
12	Open the Smart Advisor page and check the Reschedule tab.	Task rescheduling suggestions are displayed based on task details.	Task rescheduling suggestions were displayed based on the task details.	Pass
13	Open the Task Recommendation tab.	Recommended tasks to complete first are displayed.	Recommended tasks to complete first were displayed successfully.	Pass
14	Edit the task by modifying the date, time, category or priority.	Delay prediction, rescheduling suggestions and recommendations update automatically.	Delay prediction, rescheduling suggestions, and recommendations updated according to edited task information.	Pass
15	Navigate to the Personal Chat module and select a conversation.	Chat interface loads successfully.	Chat interface loaded without issues.	Pass
16	Open the attachment picker and select the "Task" option.	The newly created task appears in the task selection list.	The newly created task appeared correctly in the task attachment selection list.	Pass

Table 5.2.3.1: Test Cases - Goal Assistance Module

Time Usage Tracker Module

Tester's Name	Lim Jun Wei	**Date Tested**	5-Dec-2025	**Test Case (Pass/Fail/Not Executed)**	Pass

S #	**Prerequisites:**	**S #**	**Test Data**
1	User is logged into the BetterU mobile app as member	1	NFC Tag Code = 04579E90C12A81
2	NFC permission enabled on the	2	Task for NFC Registration = "Play Badminton"

	mobile device			
3	Usage Access permission granted on Android		3	Subtask for NFC Registration = "Prepare sports equipment"
3	Mobile device connected to same Wi-Fi network with the laptop connected network		4	Daily App Category Limit = Entertainment Apps - 10 minutes
4	Stable Internet connection		5	Whitelisted App Name = xhs
5	At least one test task and subtask exist on NFC registration			
6	Test device contains apps with measurable usage time			
7	Firebase Cloud Messaging (FCM) token is stored in the Google Firestore login member account			

Test Scenario	Registers an NFC tag to a selected task or subtask, attempts to delete the associated item to verify that the system prevents deletion while the NFC link is active, and confirms that the NFC task list remains unchanged. Then, views the daily screen-on time and app usage data, updates an app category usage limit to trigger a notification when exceeded, adds an app to the whitelist, and verifies that the whitelisted app no longer contributes to its category usage or triggers notifications.

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Open the Time Usage Tracker module and navigate to the Manage tab.	Manage tab loads with NFC task list and options.	Manage tab loaded successfully with NFC tasks and options.	Pass
2	Tap “Scan or Register NFC” to open the NFC scanning page.	NFC scanning interface is displayed.	NFC scanning interface opened successfully.	Pass
3	Scan an unregistered NFC tag.	Tag is detected and available tasks/subtasks are shown for selection.	NFC tag detected and task/subtask list displayed.	Pass
4	Select a task or subtask and register it to the NFC tag.	NFC tag is successfully bound and displayed in the NFC task list.	NFC tag was successfully registered and shown in the list.	Pass
5	Attempt to delete the associated task or subtask in the Goal Assistance module.	A pop-up error dialog appears informing the user that the task or subtask is linked to an NFC tag and cannot be deleted until the NFC registration is removed.	An error dialog appeared notifying that the task/subtask is linked to an NFC tag and cannot be deleted.	Pass
6	Return to the Manage tab and check the NFC task list.	The NFC task list remains unchanged because the task/subtask deletion was prevented, and the NFC mapping is still active.	NFC task list remained unchanged, and the task/subtask mapping was still present.	Pass
7	Open the Overall Time tab.	Daily screen-on time is displayed.	Daily screen-on time was displayed correctly.	Pass

8	Open the Screen Time tab.	Daily app usage durations, names, categories, and icons are displayed.	App usage data and app icons displayed correctly.	Pass
9	Tap “Set Screen Time Limit” to open app category controls.	App category limit list and whitelisted apps list are displayed.	Category limit list and whitelist list loaded successfully.	Pass
10	Edit one app category limit and save changes.	Updated limit is saved and reflected in the list.	Limit was successfully updated and displayed correctly.	Pass
11	Exceed the updated category limit.	A notification is received when the limit is exceeded.	Notification was received as expected.	Pass
12	Add an app to the whitelist.	App appears in the whitelist list.	App successfully added to whitelist.	Pass
13	Use the whitelisted app beyond the category limit.	Usage does not trigger notification or count toward the limit.	Whitelisted app did not trigger notifications and did not count toward total usage.	Pass

Table 5.2.3.2: Test Cases - Time Usage Tracker Module

Anonymous Community Module

Community

Tester's Name	Lim Jun Wei	**Date Tested**	5-Dec-2025	**Test Case (Pass/Fail/Not Executed)**	Pass

S #	**Prerequisites:**	**S #**	**Test Data**
1	User is logged into the BetterU mobile app	1	Community Name = "Test Community A"
2	Mobile device connected to same Wi-Fi network with the laptop connected network	2	Community Description = "This is the Test Community A"
3	Stable Internet connection	3	Community Post Title = "Test Post Title"
4	Admin test account is available for moderation testing	4	Community Post Content = "This is a test community post"
		5	Community Comment Content = "This is a test comment"
		6	Flag Reason = "Inappropriate Content"
		7	Member Account A = "test0001@gmail.com"
		8	Member Account B = "test0003@gmail.com"
		9	Admin Account = "test000@gmail.com"

Test Scenario	Creates a new community, suibmits a post and comment, enables anonymous community mode to verify anonymous name and avatar display, flags the post using another member account, and verifies that the flagged post is displayed and removed through the admin moderation panel.

Step #	**Step Details**	**Expected Results**	**Actual Results**	**Pass / Fail / Not executed / Suspended**

1	Ensure anonymous community mode is turned off in the user settings.	Anonymous community mode is confirmed to be disabled.	Anonymous community mode was disabled by default.	Pass
2	From the member site, create a new community (forum) named “Test Community A”.	Community creation request is submitted successfully and appears as pending approval.	Community creation request was submitted and shown as pending.	Pass
3	Log in to the admin site and approve the community creation request for “Test Community A”.	The community is approved and becomes available to members.	Community was approved and listed as active.	Pass
4	Return to the member site, open “Test Community A”, and create a new post.	The post is created and displayed inside the community.	Post was created and displayed in the community successfully.	Pass
5	React to the created post with like.	The reaction is recorded and shown on the post.	Reaction was recorded and displayed correctly.	Pass
6	Add a comment to the community post.	The comment is saved and displayed under the post.	Comment was displayed correctly under the post.	Pass
7	Enable anonymous community mode in the user settings.	Anonymous community mode is switched on successfully.	Anonymous community mode was enabled successfully.	Pass
8	Go back to the community post page and view the same post.	The post now displays the anonymous display name and avatar instead of the real identity.	Post showed anonymous name and avatar as expected.	Pass
9	Log in using another member account and view the same post.	The post is visible with the anonymous identity to other members.	Second member saw the post with anonymous name and avatar.	Pass
10	From the second member account, flag the community post.	The system displays a confirmation that the post has been flagged.	Flag confirmation was shown and the post was marked as flagged.	Pass
11	Log in to the admin site and open the community moderation/flagged posts tab.	The flagged community post appears in the flagged content list.	Flagged post appeared in the admin flagged list.	Pass
12	Approve the flagged post report and drop (remove) the post.	The post is removed from the community and no longer visible to members.	Post was dropped and removed from the community view.	Pass
13	Log back into the original post creator’s member account and open “Test Community A”.	The removed post is no longer displayed in the community.	Post was no longer visible in the community for the creator.	Pass

Table 5.2.3.3: Test Cases - Community Section of Anonymous Community Module

Forum

Tester's Name	Lim Jun Wei	Date Tested	5-Dec-2025	Test Case (Pass/Fail/Not Executed)	Pass

S #	Prerequisites:
1	User is logged into the BetterU mobile app
2	Mobile device connected to same Wi-Fi network with the laptop connected network
3	Stable Internet connection
4	At least two test member accounts are available
5	Admin test account is available for moderation testing

S #	Test Data
1	Forum Question = "How to improve study focus?"
2	Forum Question Content = "I find it hard to focus during long study sessions."
3	Forum Question Tag = "#StudyTips"
4	Forum Question Image Attachment = sample_forum_image.jpg
5	Forum Answer Content = "Try using a timer and taking breaks."
6	Flag Reason = "Inappropriate Content"
7	Moderation Reason = "Violates community guidelines"
8	Member Account A = "test0001@gmail.com"
9	Member Account B = "test0003@gmail.com"
10	Admin Account = "test000@gmail.com"

Test Scenario	Creates a forum question, enables anonymous forum mode to verify anonymous identity display, performs voting and answering, reports the question using another member account, and verifies that the question is flagged, moderated, and displayed as removed with a moderation reason.

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Verify anonymous forum mode is disabled in user settings.	Anonymous forum mode is disabled.	Anonymous forum mode was disabled by default.	Pass
2	Open the Forum module.	Forum page loads successfully.	Forum page loaded successfully.	Pass
3	Create and submit a new forum question.	Forum question is created and displayed in the list.	Forum question was created and displayed successfully.	Pass
4	Enable anonymous forum mode in user settings.	Anonymous forum mode is switched on.	Anonymous forum mode enabled successfully.	Pass
5	Return to the forum question page.	Question displays anonymous name and avatar.	Anonymous name and avatar displayed correctly.	Pass
6	Log in using another member account.	Second member account loads successfully.	Second member account loaded successfully.	Pass
7	Upvote or downvote the forum question.	Vote count updates correctly.	Vote was recorded and reflected correctly.	Pass
8	Post an answer to the forum question.	Answer is saved and displayed under the	Answer displayed successfully.	Pass

		question.		
9	Report the forum question.	System confirms the question has been reported.	Report confirmation was displayed successfully.	Pass
10	Log in to the admin panel and open the forum moderation tab.	Flagged forum question appears in the flagged list.	Flagged question appeared correctly in the admin list.	Pass
11	Approve the report and remove (drop) the forum question.	The question is removed and no longer visible to members.	Question was removed successfully.	Pass
12	Log back in using the original member account and open the Forum module.	The reported question is marked as removed and displays the moderation reason.	The question was marked as removed and the moderation reason was displayed correctly.	Pass

Table 5.2.3.4: Test Cases - Forum Section of Anonymous Community Module

Group Chat

S #	**Prerequisites:**	**S #**	**Test Data**
1	User is logged into the BetterU mobile app	1	Group Name = "Test Anonymous Group"
2	Mobile device connected to same Wi-Fi network with the laptop connected network	2	Anonymous Mode = true
3	Stable Internet connection	3	Chat Message 1 = "Hello, this is a test message."
4	At least two test member accounts are available	4	Chat Message 2 = "This is a real-time message test."
5	Admin test account is available for moderation testing	5	Blocked Message = [inappropriate_word]
6	FCM tokens are stored in the logged in user member account in Google Firestore	6	Attached Task = "Play Badminton"
		7	Attached Post = "Test Post Title"
		8	Flag Reason = "Inappropriate Message"
		9	Member Account A = "test0001@gmail.com"
		10	Member Account B = "test0003@gmail.com"
		11	Admin Account = "test000@gmail.com"

Test Scenario	Creates a new group chat with anonymous mode enabled, sends messages between two members to verify real-time delivery and push notifications, switches identity from anonymous to real, attempts to send a blocked message to validate content filtering, attaches a task and a post, flags a message, and confirms that the admin moderation process successfully notifies the sender.

Step #	**Step Details**	**Expected Results**	**Actual Results**	**Pass / Fail / Not executed / Suspended**
1	Create a new group chat and enable anonymous mode.	Group chat is created successfully with anonymous mode activated.	Group chat was created successfully with anonymous mode enabled.	Pass

2	Open the group chat conversation as Member A.	Chat interface loads successfully.	Chat interface loaded successfully.	Pass
3	Log in to a second device using Member B account.	Second member account logs in successfully.	Second member account login successful.	Pass
4	Send a normal chat message from Member A.	Member B receives a push notification for the message.	Push notification was received by Member B.	Pass
5	Member B opens the chat conversation.	Chat loads with the latest message.	Chat loaded with the latest message.	Pass
6	Switch chat identity from anonymous to real identity.	Display name and avatar update to real identity.	Display name and avatar updated successfully.	Pass
7	Send a message containing blocked or inappropriate words.	System blocks the message and displays a warning.	Message was blocked and warning was displayed.	Pass
8	Send another normal chat message.	Member B chat interface updates in real time via WebSocket.	Chat interface updated in real time.	Pass
9	Attach a task and a community post to the chat.	Attachments are added and displayed in the chat.	Task and post attachments appeared correctly.	Pass
10	Flag a chat message using Member B account.	System confirms the message has been flagged.	Flag confirmation was displayed.	Pass
11	Log in to the admin panel and view flagged chat messages.	Flagged message appears in the admin moderation list.	Flagged message appeared correctly.	Pass
12	Approve the report and moderate the message.	Member A receives a moderation notification.	Moderation notification was received by Member A.	Pass

Table 5.2.3.5: Test Cases - Group Chat Section of Anonymous Community Module

Personal Chat Module

Tester's Name	Lim Jun Wei	**Date Tested**	5-Dec-2025	**Test Case (Pass/Fail/Not Executed)**	Pass

S #	**Prerequisites:**	**S #**	**Test Data**
1	User is logged into the BetterU mobile app	1	Sender account email = "test0001@gmail.com"
2	Mobile device connected to same Wi-Fi network with the laptop connected network	2	Receiver account email = "test0003@gmail.com"
3	Stable Internet connection	3	Friend request identity mode = "Anonymous"
4	At least two test member accounts are available	4	Normal chat message = "Hello, this is a test message"
5	Admin test account is available for moderation testing	5	Blocked word message = [Inappropriate word]

6	Task attachment = "Play badminton"
7	Post attachment = "Test Post Title"
8	Admin account email = "test000@gmail.com"

Test Scenario	Send a friend request using anonymous identity, approve the request, exchange real-time chat messages, send task and post attachments, attempt to send blocked content, and verify that reported messages appear in the admin moderation panel and trigger notification to the sender.

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Send a friend request to another user with anonymous identity selected.	Friend request is sent successfully.	Friend request was sent successfully.	Pass
2	Log in as the second user and approve the friend request.	Friend is added to the friend list.	Friend was added to the friend list.	Pass
3	Open the personal chat with the approved friend.	Chat interface loads successfully.	Chat interface loaded successfully.	Pass
4	Send a normal message in the chat.	Message is delivered and shown in real time.	Message was delivered and displayed correctly.	Pass
5	Verify the second user receives a push notification.	Push notification is received by the receiver.	Push notification was received successfully.	Pass
6	Attach a task and a community post and send them.	Attachments are sent and displayed in the chat.	Attachments were displayed correctly in the chat.	Pass
7	Attempt to send a message containing blocked words.	System blocks the message and shows a warning.	Message was blocked and warning was displayed.	Pass
8	Report a chat message.	System confirms the message has been reported.	Report confirmation was shown successfully.	Pass
9	Log in as admin and view the reported message.	Flagged message appears in moderation panel.	Flagged message appeared correctly in the admin panel.	Pass
10	Approve the report as admin.	Moderator action is saved successfully.	Report was approved successfully.	Pass
11	Log back in as the sender and check notification.	Sender receives a moderation notification.	Moderation notification was received successfully.	Pass

Table 5.2.3.6: Test Cases - Personal Chat Module

5.3 Chapter Summary and Evaluation

This chapter has involved the implementation of all functional modules in the BetterU mobile application and validated their performance via a systematic integration testing. For example, the modules involved are Goal Assistance, Time Usage Tracker, Anonymous Community and Personal Chat modules. Each module was successfully implemented using the selected development tools and managed through the backend and Firebase services.

The integration testing has successfully indicated that all the modules can interact correctly and consistently as a single ecosystem. The speech-based task input was accurately transcribed and forwarded into the NLP pipeline while the task extraction was mapped correctly into the task creation interface and subsequent task updates triggered AI-driven predictions and recommendations as expected. The Time Usage Tracker successfully synchronized Android native services with Flutter. Meanwhile, the NFC-based task sessions updated the task records in real time.

For social modules, the anonymous identity handling, WebSocket real-time updates, attachment sharing and message reporting workflows can also be operated reliably. The administrator moderation panel also updated the flagged content timely and accurately. This can ensure the correctness of cross-module data communication.

In overall, the implementation has aligned closely with the system requirements defined in previous chapters. The successful test outcomes have ensured that the BetterU mobile application functions can act as an integrated, intelligent and user-centric productivity and well-being application. The system shows strong reliability, modular cohesion and readiness. Not only that, the results validated also proved that the selected architectures and AI components have offered a meaningful enhancement in the user productivity and social engagement.

Chapter 6

Discussions and Conclusion

6 Discussions and Conclusion

6.1 Chapter Summary

The BetterU project was developed to address two main problems highlighted in Chapter 1 which are **social isolation and fear of seeking help** as well as the **cognitive overload and difficulty in task management**. These issues are frequently faced by the students and office workers since they always struggle with the overwhelming workloads, poor time allocation, emotional stress and reluctance to seek social support. In order to respond to these challenges, BetterU has introduced 5 integrated modules designed for improving both productivity and well-being. They are **Goal Assistance**, **Time Usage Tracker**, **Anonymous Community**, **Personal Chat** and **Report Module**. The Goal Assistance module incorporates speech-to-text input, delay prediction and personalized recommendations to help the users to manage tasks in a more intelligent way. The Time Usage Tracker monitors the screen time, analyzes the daily usage trends and integrates the NFC-based task registration for seamless real-world task tracking. The Anonymous Community and Personal Chat modules create a safe and fearless social environment where users can share their thoughts, find emotional support, participate in communities and communicate privately with anonymous or real identity. This can effectively reduce social anxiety and fear of judgment. Eventually, the Report module provides visualized productivity reports, time-usage summaries and social engagement insights to help the users to understand their habits and continuously improve their routines.

The choice of technologies used in this project is justified by their suitability for mobile cross-platform development, real-time data handling and AI-driven features. For example, **Flutter** was selected as the main development framework because it allows the deployment to Android platform with various versions using a single codebase. This can reduce the development time and ensures the UI consistency across different Android platforms. **Python with FastAPI** was chosen for backend and AI logic because it contains an abundant NLP or ML ecosystem. Meanwhile, it also offers high performance API handling and clean separation between frontend and backend responsibilities. **Google Firebase** was used as the cloud database because it supports real-time synchronization, low maintenance requirements and it is also cost-effective for cross-platform mobile development. The **Firebase Cloud Messaging** allows for real-time notifications and reminders which are crucial for task management and community engagement. In order to implement the intelligent features, BetterU has also adopted multiple AI frameworks such as **Vosk** and **Whisper**. They are used for speech transcription. Not only that, the **spaCy** is used for task extraction, **SetFit** used for task categorization, **Random Forest** used for delay prediction and **ALS collaborative filtering**

for Smart Advisor recommendations. Each of the selected AI frameworks is selected based on the accuracy, efficiency and suitability for the mobile-app integration.

When it comes to the software development model, the project has applied the **Incremental Methodology**. It allows the system to be built in well-structured increments. For example, Build 1 focused on core modules, Build 2 focused on behavior-tracking modules and Build 3 focused on community and reporting modules. This strategy allows the continuous refinement through the supervisor feedback, early testing and progressive feature improvement. In a nutshell, this chapter summarizes how the BetterU project integrates speech-to-text input, AI-driven task assistance, behavior analytics and anonymous social support into a unified mobile application that solves productivity and emotional well-being in a balanced and user-centric way.

6.2 Achievements

The BetterU project has successfully achieved all the objectives outlined in Chapter 1. It has fulfilled its purpose as a **productivity and well-being application**. This project can effectively overcome social **isolation**, **fear of seeking help** and **cognitive overload in task management**. The system has integrated AI-driven task assistance, behavior tracking, anonymous community engagement, private communication and personalized reporting to achieve this objective. All of these functionalities have successfully provided a result of user productivity improvement, emotional balance and overall well-being.

From the perspective of project objectives, the system has successfully:

- **Reduced cognitive overload and improved task clarity** via speech-to-text input, NLP task extraction, smart scheduling, delay prediction and personalized task suggestions.
- **Enhanced time management** using the app usage tracking from time to time, NFC task registration and automated behavioral insights.
- **Strengthen social connection and reduced fear of judgment** via anonymous community interaction, supportive discussion spaces and private messaging with freedom of selecting real or anonymous identity.
- **Increased user awareness and personal growth** via generating productivity reports, time-usage summaries and social engagement insights.

Module Achievement

Goal Assistance Module

This module has successfully provided the intelligent task management via:

- Accurate speech-to-text conversion for instant task entry.
- Task extraction using NLP for detecting the start and end datetime, category, prioritizes and other relevant details..
- Smart Advisor suggestions help users to reduce cognitive overload when maintaining task management.
- Delay prediction and rescheduling recommendations provide insights which allow the users to maintain a timely progress.

Time Usage Tracker Module

This module has achieved its purpose of improving the self-awareness and reducing the time wastage by:

- Tracking users' daily app usage and identifying unproductive patterns.
- Presenting visual breakdowns of daily or weekly habits.
- Supporting NFC task registration which enables users to clock-in or out of their tasks seamlessly.
- Offering custom app category time limit goals, reminders and improvement suggestions.

Anonymous Community Module

This module successfully solves the issues of social isolation and fear of judgment by:

- Allowing users to post, comment and interact with community posts anonymously.
- Providing topic-based communities for expressing users' thoughts and ideas.
- Enabling a safe engagement by allowing users to flag the inappropriate content.
- Offering a comfortable and stigma-free environment for maintaining mental health.

Personal Chat Module

This module has achieved the private emotional support and meaningful interaction with the world by:

- Enabling one-to-one private messaging using text, images, task and post content.
- Supporting anonymous or real-identity communication depending on user preference.
- Providing user blocking and message reporting when facing any inappropriate chat content..

Report Module

This module fulfilled the objective of enhancing self-reflection and improvement by:

- Generating productivity reports which summarize the completed tasks, missed tasks and performance trends.

- Providing social engagement reports that determine the users' emotional interactions.
- Showing the insights using visual charts and graphs for high readability.
- Offering personalized suggestions for both productivity and social growth.

Strengths of the Project

- The system integrates multiple advanced technologies which include speech recognition, natural language processing, intelligent task classification and behavior-based prediction. These features are useful in offering a highly capable and intelligent task management solution.
- A modular system architecture was developed for enabling the independent development of each module when maintaining a seamless integration across the entire system. This structure can enhance scalability, maintainability and future extensibility.
- The real-time performance is achieved through Firebase's cloud synchronization and the timely delivery of notifications which will ensure the updates, reminders and user actions can be reflected with minimum latency.

Weaknesses of the Project

- Some of the AI components, especially those used for speech transcription and NLP processes would require a high computational resources. They may likely lead to a slower performance on lower-end devices.
- NFC performance can vary across different smartphone models since the hardware sensitivity differences. It will affect the reliability of the task clock-in and clock-out feature in some of the devices.
- The recommendation models are limited by the size and diversity of the available training data. A larger dataset would improve the prediction accuracy and enhance the personalization capabilities of the system.

6.3 Contributions

Creativity and Innovativeness of BetterU Mobile Application

The BetterU mobile application has presented its creativity and innovativeness via combining the intelligent task management, behavioral tracking and privacy-focused social engagement features. One of the most significant points is the **integration of speech-to-text task input**. It allows users to note down the tasks without relying on manual typing. This can efficiently reduce cognitive effort and improves accessibility, especially for the users who are overwhelmed by long task lists or tight schedules. Via the integration of this feature, the

system can perform the task extraction automatically using natural language processing. This allows the BetterU mobile app to identify the start and end date time and other relevant task details from user input and convert them into structured task items. This automation can provide a seamless and intuitive task creation experience that is diverse from traditional task management tools.

Another innovative contribution is the inclusion of **smart rescheduling suggestions** and **task recommendations**. They consist of **behavior-based analysis** and **delay prediction models**. These features enable the system to overcome user challenges, recommend suitable timeslots and guide the users towards better task management strategies. Besides that, BetterU also introduces a comprehensive **app usage tracking mechanism** which monitors the daily screen activity, identifies unproductive patterns and triggers reminders when there is excessive time usage found from time to time. This can help promote healthy digital habits while increasing the user's self-awareness.

Apart from that, the BetterU mobile application also introduces creativity through its thoughtful **consideration of user privacy and social well-being**. Unlike traditional social platforms, BetterU provides a supportive and non-judgmental environment by enabling anonymous posting, identity control and safe communication in community and chat modules. These features are particularly innovative in solving the social fear, emotional stress and the reluctance to seek help from others.

Moreover, the **integration of NFC technology for task clock-in and clock-out also** builds a bridge between the digital task tracking with physical activities. This seamless tracking mechanism can effectively improve the user convenience and creates a tangible connection between planned tasks and completed actions.

In conclusion, the creativity of BetterU not only depends on each individual feature, but also in the holistic integration of productivity tools, AI-driven personalization, behavioural insights and privacy-conscious social interaction. This multi-dimensional methodology is uncommon among the existing productivity applications and reflects a meaningful innovation in supporting both productivity and emotional well-being of users.

Necessity of BetterU Mobile Application

Addresses social isolation and fear of seeking help

BetterU provides users with a safe environment where they can interact anonymously within communities and private chats. This helps those users who feel lonely, overwhelmed or afraid of being judged to express their thoughts in a more comfortable zone. It directly responds to

the social struggles highlighted in Chapter 1, especially for students and workers who often hesitate to reach out for help.

Reduces cognitive overload through automated task support

Many people feel overwhelmed when managing multiple responsibilities. BetterU mobile app can reduce this burden by automatically extracting task details, organizing priorities and generating schedules. This helps users to put more focus on completing their works instead of worrying about the detailed planning.

Supports a healthier balance between work and mental well-being

The BetterU mobile application combines the productivity tools with the emotional support features. This can promise that the users can always stay organized without sacrificing their mental health. Whether through encouraging communities or stress-free chat spaces, BetterU helps the users to maintain a healthier balance in their daily lives.

Provides a private and safe space for emotional expression

BetterU understood that there are a lot of people who always feel uncomfortable when sharing their feelings openly, especially when there are many people in front of them. Thus, BetterU gives users a private and anonymous outlet where they can talk about their worries, ask for advice or simply express themselves without fear of criticism. This secure environment encourages genuine communication and emotional relief.

Promotes healthier digital habits through app usage insights

By monitoring screen time and app usage, BetterU helps users become more aware of their digital behaviour. This allows them to identify unhealthy patterns such as spending too much time on certain apps. Meanwhile, it can also make positive adjustments to their daily routines.

Marketability of BetterU Mobile Application

Appeals to a large and diverse user group

BetterU targets students, young adults and working professionals who regularly seek practical tools to manage their tasks and personal responsibilities. This wide demographic increases the system’s potential adoption rate.

High accessibility due to widespread smartphone usage

Malaysia has strong smartphone penetration and reliable mobile Internet access. This situation can potentially help BetterU to easily face the users without requiring special devices or additional hardware investment. The users can easily access the BetterU mobile application to enjoy the task management and well-being improvement services.

Offers a unique combination of features not commonly found together

BetterU stands out by integrating AI-driven task planning, NFC task clocking and anonymous social interaction into a single platform. This distinctive blend gives the system a competitive edge in the productivity app market.

Challenges and Solutions

Extensive AI integration

- Challenge: The system relies on a lot of AI components, including speech recognition, natural language processing, classification models and behavioural prediction. The integration of different models into a single mobile application will require careful and detailed planning for ensuring good performance.
- Solution: The lightweight models and efficient frameworks were selected while isolating the background process to reduce the processing load and improve responsiveness.

High computational requirements

- Challenge: Some AI features will spend significant computational resources and cause performance drops. For example, task prediction, scheduling suggestions and NLP processing. The operation of these features will cause performance drops, especially during peak usage or when handling multiple tasks at once.
- Solution: The heavy AI processes were shifted to automated background scheduling to make sure the backend server will not be overloaded. For example, the system will perform the computation at a fixed interval every morning at 8 AM. Thus, the device can remain responsive while still delivering updated predictions and recommendations to users. By processing the tasks in the background, the application maintains a seamless user experience without interrupting normal usage.

Consistency between backend and frontend

- Challenge: The interaction and handling between the FastAPI backend, Flutter's UI and data models was a complex task, especially when ensuring that both sides handled data formats and logic consistently.
- Solution: Standardized data models, detailed documentation and iterative testing of API endpoints were used to ensure reliability and smooth integration.

6.4 Limitations and Future Improvements

Absence of a Recurring Task Mechanism

Currently, the Goal Assistance module only supports one-time tasks. Users who depend on repeated routines such as daily habits, weekly study plans or monthly reminders cannot automate them. This will reduce convenience and make users manually recreate the same tasks again and again. Eventually, this will affect the overall efficiency of long-term planning.

Future Improvement:

A recurring task feature can be added to support daily, weekly, monthly or custom repetition intervals. Extra options like setting an end date, skipping certain repeats or automatically creating subtasks would make it even more useful. If this recurring feature is also connected to the Smart Advisor, the system can provide better personalized recommendations and support users in building consistent habits.

Limited and Synthetic Training Data for AI Models

Several AI components in BetterU, such as task classification, suggestion generation and delay prediction, were trained using synthetic or small datasets. Because of this, the models may not fully understand different user behaviours, task-writing styles or real-life patterns. This can affect the accuracy and general performance of the AI features when used by actual users.

Future Improvement:

Future development should focus on expanding the dataset using real and anonymized user data collected through optional feedback. A wider and more diverse dataset will make the models more reliable. More advanced NLP techniques such as fine-tuning BERT-based models or using active learning can also help improve accuracy. Continuous learning can be added so the AI can slowly adapt to each user’s behaviour over time.

Backend Not Deployed on a Public Cloud Environment

At the moment, the FastAPI backend is only running on a local machine. This means the mobile application cannot access any backend services when used outside the development environment. Because of this, real-world testing, multi-user usage and performance monitoring cannot be fully carried out, and external users cannot try the system.

Future Improvement:

Deploying the backend to a cloud platform such as Render or AWS will allow real-time access by users. Cloud hosting also provides better scalability, load balancing, logs and easier maintenance. Any Python dependency conflicts can be solved by using environment isolation, Docker containers or adjusting the dependency list to match cloud requirements.

Basic Community Moderation and Safety Mechanisms

Although BetterU includes keyword filtering, user reporting and basic admin review, the current moderation features are still quite limited. The system does not support advanced features such as automated toxicity detection, spam filtering, image content moderation or sentiment analysis. Because of this, there is still a risk that harmful or inappropriate content may appear inside the anonymous community.

Future Improvement:

Stronger moderation tools can be added by integrating machine learning models that can detect toxicity, hate speech or spam. Sentiment analysis can be used to identify negative emotional patterns in posts. Image moderation APIs can help filter inappropriate pictures. A more complete admin dashboard with automated flagging and alert functions can also help improve community safety.

6.5 Issues and Solutions

Difficulty when Choosing the Right Database and Project Structure

At the beginning of the project, it was very difficult to choose an appropriate database and decide how the overall system architecture should look. In the existing cloud databases market, there are many options available but each offers different benefits and limitations in terms of scalability, real-time features and cost. It took a lot of time to compare them and figure out which one matched the needs of BetterU. Besides that, planning the coding structure and choosing the right architectural pattern for both Flutter and FastAPI also required a lot of research and careful thinking.

Solution:

After comparing a few alternatives, Firebase was chosen because it can offer real-time data syncing and it is platform independent. Meanwhile, it is also very easy to learn and apply into the mobile application. When it comes to the connection between Flutter frontend and Python backend, FastAPI was selected for the backend linkage due to its lightweight design and high performance. The clear folder structures, models and coding conventions were also created early to make sure the whole project remained consistent and easy to maintain.

Challenges Linking Flutter to the FastAPI Backend

Connecting the Flutter application to the FastAPI backend turned out to be more challenging than expected. It worked fine in the emulator, but problems started to appear when testing on real Android devices. Issues like network communication failures, CORS restrictions and IP address accessibility happened quite often, which caused difficulties when trying to connect both sides together.

Solution:

The backend was updated with proper CORS settings, and correct local IP addresses were used when connecting real devices. API testing tools were also used to check whether each request was working as expected. These steps helped ensure stable communication between the app and the backend.

Complexity in Developing AI Features

Developing the AI-related features was one of the toughest parts of the whole project. Since there was limited experience with machine learning, it was difficult to understand how NLP, audio preprocessing, speech recognition and recommendation logic all worked together. Features like task rescheduling suggestions, NLP-based task extraction and combining Vosk with Whisper for transcription required a lot of learning, testing and patience.

Solution:

A lot of self-learning was done through documentation, tutorials and continuous experimentation. Instead of trying to build everything at once, lighter models and step-by-step integration helped make the process easier. The complex modules were split into smaller tasks, which made debugging much more manageable. Following the incremental development plan also helped ensure steady progress.

Python Dependency Conflicts During Backend Development

When working on the backend, several Python libraries for AI and NLP did not work properly with the initial Python version (Python 3.13). Many packages failed to install or broke during runtime due to version incompatibility, which made the backend environment unstable.

Solution:

The backend environment was rebuilt using Python 3.10.0, which is well supported by most AI libraries such as spaCy, Whisper and SetFit. A fresh virtual environment was created, and specific dependency versions were pinned to avoid future conflicts. This provided a stable and reliable setup for further backend development.

6.6 Conclusion

To summarize everything, the BetterU mobile application successfully fulfills its objective of providing a unified platform that can improve users' productivity and emotional well-being by solving the main problems of social isolation, cognitive overload and poor task management. Via integrating the core modules such as Goal Assistance, Time Usage Tracker, Anonymous Community, Personal Chat Module and Report Module, the system shows how the AI-driven

task planning, real-time behavior analytics collaborate with each other. Meanwhile, the privacy-focused social features are also applied to support both personal efficiency and healthier social engagement. The Incremental Model is applied throughout the project, enabling a systematic development, testing and refinement of features to ensure each module was implemented in a way that perfectly fits the user needs. Even though there are several limitations in this developed mobile application, the overall outcome has indicated that BetterU is an impactful solution which can potentially help students and office workers to organize their tasks,monitor their digital habits and reconnect with society in a safe and meaningful way.

References

Aleeya, N., & Binti Roslan, S. (2023). *Social anxiety and social support perceivability among undergraduate students in Universiti Sains Malaysia*. http://eprints.usm.my/61393/1/11%20NUR%20ALEEYA%20SYAFIKAH%20BINTI%20ROSLAN%20(146956)-E.pdf

Alpha Cephei. (2024). *Vosk speech recognition toolkit*. https://alphacephei.com/vosk/

Android Developers. (2019). *Create and manage virtual devices*. https://developer.android.com/studio/run/managing-avds

Android Developers. (n.d.). *Android Studio overview*. https://developer.android.com/studio

Ashraff, S. (2025). *Voice-based interaction with digital services*.

Avast. (2023). *Blog post on self-help app privacy*. https://blog.avast.com/self-help-apps-privacy

Bhagat, S. A. (2022). Review on mobile application development based on Flutter platform. *International Journal for Research in Applied Science and Engineering Technology, 10*(1), 803–809. https://doi.org/10.22214/ijraset.2022.39920

Breiman, L. (2001). Random forests. *Machine Learning, 45*(1), 5–32. https://doi.org/10.1023/A:1010933404324

Chamrah, Y. (2024, January 13). SetFit unpacked: When sentence transformers go to "classification gym." *Medium*. https://medium.com/@youssefchamrah/setfit-unpacked-when-sentence-transformers-go-to-gym-for-classification-muscle-56c16d9e69de

Code.visualstudio.com. (n.d.). *Integrated terminal in Visual Studio Code*. https://code.visualstudio.com/docs/terminal/basics

Crudu, V. (2024, August 28). How to create a mobile app using Dart programming language? *MoldStud*. https://moldstud.com/articles/p-how-to-create-a-mobile-app-using-dart-programming-language

Dart.dev. (n.d.). *Dart overview*. https://dart.dev/overview

Design Project. (n.d.). *Figma collaboration and teamwork*. https://designproject.io/blog/figma-collaboration-teamwork/

Dev.to. (2023). Why manual workflows are the worst enemy of your productivity. https://dev.to/softyflow/why-manual-workflows-are-the-worst-enemy-of-your-productivity--2n0l

Drosopoulou, E. (2024, September 16). Why Flutter picked Dart: A deeper dive. *Java Code Geeks*. https://www.javacodegeeks.com/2024/09/why-flutter-picked-dart-a-deeper-dive.html

Espindola, W. (2025, May 20). FastAPI unleashed: Building modern and high-performance APIs. *DEV Community*. https://dev.to/wallaceespindola/fastapi-your-fast-and-modern-framework-for-apis-3mmo

Explosion AI. (2024). *spaCy: Industrial-strength NLP*. https://spacy.io/

FastAPI. (n.d.). *FastAPI documentation*. https://fastapi.tiangolo.com

Figma. (2024). *Figma: The collaborative interface design tool*. https://www.figma.com/

Firebase. (2019a). *Choose a database: Cloud Firestore or Realtime Database*. https://firebase.google.com/docs/database/rtdb-vs-firestore

Firebase. (2019b). *Firebase Realtime Database*. https://firebase.google.com/docs/database

Firebase. (n.d.). *Firebase documentation*. https://firebase.google.com/docs

FlowWright. (n.d.). *Why manual processes are overwhelming employees*. https://www.flowwright.com/why-manual-processes-are-overwhelming-employees

Flutter. (2024). *Flutter: Build apps for any screen*. https://flutter.dev/

Flutter. (2024). *Flutter – Beautiful native apps in record time*. https://flutter.dev/

GeeksforGeeks. (2024, August 27). *GitHub Desktop*. https://www.geeksforgeeks.org/git/github-desktop/

GetApp. (n.d.). *Task management apps with prioritizing features*. https://www.getapp.com/project-management-planning-software/task-management/f/prioritizing/

GitHub. (n.d.). *GitHub Desktop*. https://desktop.github.com

GoGuardian. (2025). *Go to GoGuardian app*. https://ebpj.e-iph.co.uk/index.php/EBProceedings/article/download/3836/2087/18617

Hiredly. (2025). “Work ends at 6, not 10”: How Malaysians from all generations are drawing the line in 2025. *Hiredly Career Advice*. https://my.hiredly.com/advice/work-culture-report-work-life-balance-malaysia-gen-z-gen-x-millennials-2025

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation, 9*(8), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

IJSRD. (2024). Study on intelligent task management systems. *International Journal for Scientific Research & Development, 12*(2), 60–62. https://www.ijsrd.com/articles/IJSRDV12I20060.pdf

Khawas, C., & Shah, P. (2018). Application of Firebase in Android app development—a study. *International Journal of Computer Applications, 179*(46), 49–53.

Koren, Y., Bell, R., & Volinsky, C. (2009). Matrix factorization techniques for recommender systems. *Computer, 42*(8), 30–37. https://doi.org/10.1109/MC.2009.263

Kula, C. (2015). Metadata embeddings for user and item cold-start recommendations. https://arxiv.org/abs/1507.08439

Liang, E., Liaw, R., Nishihara, R., Moritz, P., Fox, R., Goldberg, K., ... & Stoica, I. (2018). RLlib: Abstractions for distributed reinforcement learning. *Proceedings of the 35th International Conference on Machine Learning (ICML)*. https://arxiv.org/abs/1712.09381

Manish. (2021, April 6). Python mobile app development: Is it right for you? *MobileAppDaily*. https://www.mobileappdaily.com/knowledge-hub/python-for-mobile-apps-development

Microsoft. (n.d.). *Visual Studio Code*. https://code.visualstudio.com

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*. https://arxiv.org/abs/1301.3781

Mir, U. B., & Shardeo, V. (2024). Demystifying resistance: Factors influencing users’ reluctance to share images on social media. *Aslib Journal of Information Management*. https://doi.org/10.1108/ajim-06-2024-0497

Monterail. (2025). Why choose Flutter? 6 top reasons to use Flutter for mobile app development. https://www.monterail.com/blog/top-reasons-for-using-flutter-for-mobile-app-development

Monterail. (n.d.). Python for mobile app development in 2023 – Kivy vs. BeeWare. https://www.monterail.com/blog/python-for-mobile-app-development

Mozilla Foundation. (2023). *Top mental health and prayer apps fail at privacy*. https://www.mozillafoundation.org/en/blog/top-mental-health-and-prayer-apps-fail-spectacularly-at-privacy-security

NNG Group. (n.d.). *Prioritization methods*. https://www.nngroup.com/articles/prioritization-methods/

Nuraly, A. (2020, August 28). Major advantages and disadvantages of Dart language. *Code Carbon*. https://codecarbon.com/pros-cons-dart-language/

OpenAI. (2022, October 9). *Whisper*. https://github.com/openai/whisper

OpenAI. (2024). *Whisper*. https://openai.com/research/whisper

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Duchesnay, É. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research, 12*, 2825–2830. https://www.jmlr.org/papers/volume12/pedregosa11a/pedregosa11a.pdf

Peng, T., Akmal, B., & Kamil, M. (2017). Time management, procrastination and prioritization: A framework for service based learning module. *e-AJUiTMCT, 6*, 1–12. https://e-ajuitmct.uitm.edu.my/v2/images/vol6issue2/PID114_TIMEMANAGEMENTPROCRASTINATION.pdf

Perron, L., & Furnon, V. (2019). OR-Tools: Operations research tools. *Google Developers*. https://developers.google.com/optimization

Pew Research Center. (2013). *Anonymity, privacy, and security online*. https://www.pewresearch.org/internet/2013/09/05/anonymity-privacy-and-security-online/

Public Health Tulane. (2020, December 8). Understanding the effects of social isolation on mental health. https://publichealth.tulane.edu/blog/effects-of-social-isolation-on-mental-health/

Puddot. (2025, June 13). GitHub - Puddot/vosk-api. https://github.com/Puddot/vosk-api

Radford, A., Kim, J. W., Xu, T., Brockman, G., McLeavey, C., & Sutskever, I. (2023). *Robust speech recognition via large-scale weak supervision*. https://arxiv.org/abs/2212.04356

Ramírez, S. (2024). *FastAPI*. https://fastapi.tiangolo.com/

Raschka, S., Patterson, J., & Nolet, C. (2020). Machine learning in Python: Main developments and technology trends in data science, machine learning, and artificial intelligence. *Information, 11*(4), 193. https://doi.org/10.3390/info11040193

Reddit. (2024). *Asana vs Notion user opinions*. https://www.reddit.com/r/Notion/comments/162xqrc/notion_or_asana/

Richman, J. (2023, February 27). Firestore vs. Realtime Database: Which performs better? *Estuary*. https://estuary.dev/blog/firestore-vs-realtime-database/

Saabith, A. S., Fareez, M. M. M., & Vinothraj, T. (2019). Python current trend applications—an overview. *International Journal of Advance Engineering and Research Development, 6*(10).

scikit-learn. (2019). *Scikit-learn: Machine learning in Python*. https://scikit-learn.org/

Scrapinghub. (2024). *Dateparser documentation*. https://dateparser.readthedocs.io

SEEK Limited. (2023, July 10). Malaysian workers aren't getting enough time with their families. *Jobstreet*. https://my.jobstreet.com/career-advice/article/malaysian-workers-arent-getting-enough-time-families

Smartsheet. (2023). *Workers waste quarter of workweek on manual tasks*. https://www.smartsheet.com/content-center/product-news/automation/workers-waste-quarter-work-week-manual-repetitive-tasks

Software Advice. (n.d.). *Asana vs Trello: A project management comparison*. https://www.softwareadvice.com/project-management/asana-profile/vs/trello/

spaCy. (2015). *spaCy: Industrial-strength natural language processing in Python*. https://spacy.io/

Swathiga, U. U. A. S., Vinodhini, P., & Sasikala, V. (2021). An interpretation of Dart programming language. *DRSR Journal, 11*(3), 144–149.

Tamplin, J. (2016, May 20). Firebase expands to become a unified app platform. *Google Blog*. https://blog.google/products/admob/firebase-expands-to-become-unified-app/

Tan, Y. L., Yeow, J. A., & Tai, H. T. (2022). Conceptual paper: The use of social media improved academic performance among university students. *Malaysian Journal of Business, Economics and Management, 1*(1), 14–22. https://doi.org/10.56532/mjbem.v1i1.3

Tech.co. (n.d.). *Asana vs Trello: Which project management tool is better?* https://tech.co/project-management-software/asana-vs-trello

The Digital Project Manager. (2024, October 15). 15 best task management software reviewed for 2024. https://thedigitalprojectmanager.com/tools/best-task-management-software/

The Digital Project Manager. (2025, April 4). 24 best project management software for individuals reviewed in 2025. https://thedigitalprojectmanager.com/tools/best-project-management-software-for-individuals/

Tiangolo. (2024). *FastAPI*. https://fastapi.tiangolo.com/

Tulane University. (2020, December 8). Understanding the effects of social isolation on mental health. https://publichealth.tulane.edu/blog/effects-of-social-isolation-on-mental-health/

Tunstall, L., von Platen, P., Bevendorff, J., Thiel, A., & Wolf, T. (2022). *Efficient few-shot learning with SetFit*. https://arxiv.org/abs/2209.11055

Unknown. (2015). [Malaysia] Second worst country for work-life balance in new ranking. *GPA*. https://gpa.net/blogs/emea/malaysia-second-worst-country-for-work-life-balance-in-new-ranking

Viquez, J. (2025). *Figma: The ultimate design collaboration platform*. https://www.jimmyviquez.design/resources/figma

Zimmer, L. (2023). SetFit: Efficient few-shot learning with sentence transformers [Computer software]. *GitHub*. https://github.com/huggingface/setfit

Appendices

Development Environment Requirements

- IDEs: Android Studio, Visual Studio Code
- Python 3.10.0
- Flutter 3.35.6 (Channel Stable)
- Dart 3.9.2

Backend Setup

1. Install Python 3.10.0 via the URL if not installed: https://www.python.org/ftp/python/3.10.0/python-3.10.0-amd64.exe
2. Install Node.js if not installed: https://nodejs.org/
3. Open Visual Studio Code.
4. Open the "betteru-frontend" project folder.
5. Create a python environment using command: python -m venv env310
6. Activate the environment from the project directory: .\env310\Scripts\activate
7. Install all the required python dependencies: pip install -r 310-requirements.txt
8. Start the backend server
    - If using emulator, enter this command: uvicorn main:app --reload
    - If using real device, enter this command: uvicorn main:app --reload --host 0.0.0.0 --port 8000

Frontend Setup

1. Open Android Studio
2. Open the "betteru-backend" project folder
3. Open terminal section in Android Studio
4. Clean up the flutter environment: flutter clean
5. Install all the packages required: flutter pub get
6. Establish connection with backend server:
    - If using an emulator, open the "dio_client.dart" file and replace the "DEVICE_LOCAL_BASE_URL" with "EMULATOR_BASE_URL".
    - If using a real device, make sure your real device is connected to the network same with your computer connected network. Check your computer connected IPv4 address by opening the windows terminal and entering "ipconfig". Then, open the ".env" file and replace the value of DEVICE_LOCAL_BASE_URL with "http://[your-IPv4-address]:8000".
    - On the top panel, select your connected real device.
7. Click on the "Play" button to start the flutter application.

8. If your real device screen popped out the installation permission request, please allow them to proceed with the BetterU application installation.
9. Once the BetterU application has been installed on the device, it will be automatically started and opened.

User Account in BetterU Mobile Application

- Member Account:
    - Member 1:

Email Address	test0001@gmail.com
Password	Abc123

    - Member 2:

Email Address	test0003@gmail.com
Password	Abc123

    - Member 3:

Email Address	test0004@gmail.com
Password	Abc123

    - Member 3:

Email Address	fypbetteru@gmail.com
Password	Abc123

- Admin Account:

Email Address	test000@gmail.com
Password	Abc123

User Guide

1. Member Site
    - Login
        - Open the BetterU app
        - Enter email and password from the provided test member accounts.
    - Goal & Tasks
        - View tasks in Card or Calendar view.

        - Tap the bottom right floating button "+", select "Create Goal" to create a task.
        - Use AI Task Assistant to record voice and auto-fill task details.
    - Smart Task Advisor
        - Shows AI suggestions for task rescheduling and prioritisation.
        - Tap Apply Suggestion to accept recommended changes.
    - Time Usage Tracker
        - View daily screen-on time and focus duration.
        - Set Screen Time Limits and manage Whitelisted Apps.
        - Use Scan NFC to register an NFC tag or start a focus session.
    - Social (Personal and Group Chat)
        - Chat with friends or group members.
        - Send messages, images, tasks, posts, or diary entries.
        - Group creators can invite/remove members and toggle anonymous mode.
    - Q&A Forum
        - Browse questions, filter by tags, or search.
        - View question details and post answers with attachments.
    - Community
        - View joined communities.
        - Create a new community (name, tags, description).
        - Browse community posts and search by title or tag.
    - Reports
        - View task productivity and social performance summaries in Card or Chart view.
        - Export reports to PDF
2. Admin Site
    - Login
        - Open the BetterU app
        - Enter email and password from the provided test member accounts.
    - Community Moderation
        - Review community creation requests.
        - Approve or reject submissions.
    - Chat Moderation
        - Review reported chat messages.
        - Approve or remove messages

This page is intentionally left blank to indicate the back cover. Ensure that the back cover is black in color.