BetterU: Productivity and Well-Being Mobile Application

By

Chia Ming Yi

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 Mobile App: Goals Assistance(shared), Emotion Diary Note, Focus Timer, User, Report

By

Chia Ming Yi

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.

_____

Chia Ming Yi

ID: 24WMR04090

Abstract

The BetterU mobile application is designed to address common challenges such as social isolation, cognitive overload, and poor task management, which are often faced by students and office workers. By providing solutions to these issues, the application aims to enhance both productivity and overall well-being. BetterU consists of five key modules: Goal Assistance(shared), Emotion Diary Note, Focus Timer, User, and Report. Together, these modules provide an intelligent task scheduling system, a secure communication platform, and personalized insights for users.

The literature review has identified the target market and determined the technologies to be used in BetterU. These include 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 has been conducted to assess the project's technical, economic, and operational viability.

The project adopts the Incremental development model to ensure continuous refinement through requirement gathering, functional specification, and supervisor feedback. During the requirement analysis stage, both functional and non-functional needs were collected, and various system design diagrams were prepared. These include sequence diagrams, activity diagrams, entity–relationship diagrams, class diagrams, data dictionaries, report designs, and deployment architecture diagrams.

Overall, the system design demonstrates how BetterU can serve as a practical and intelligent solution to improve time management and social engagement for students and office workers. By integrating productivity tools with AI-driven personalization and a supportive anonymous community, BetterU is envisioned as a scalable and user-friendly application that promotes efficiency, emotional well-being, and sustainable work-life balance.

Acknowledgement

I would like to express my heartfelt gratitude to everyone who has supported, guided, and accompanied me throughout the journey of completing this project proposal. This accomplishment would not have been possible without the collective encouragement, wisdom, and kindness I received from many individuals around me.

First and foremost, I would like to extend my sincere thanks to my final year project supervisor, Ms. Yeoh Kar Peng, for her continuous guidance, encouragement, and insightful feedback. Her patience and dedication helped me stay focused and motivated, and her professional advice played a vital role in shaping the direction and success of this project.

I am especially grateful to my family, whose support has been the foundation of my strength. To my beloved mother, Chau Sook Khan, thank you for your love throughout this journey. Your support created a motivating environment that enabled me to stay resilient, even during moments of uncertainty and stress.

To my teammate, Lim Jun Wei, I would like to express my deepest appreciation for your cooperation, dedication, and commitment to our shared responsibilities. Your excellent communication, problem-solving skills, and consistent involvement contributed significantly to the overall quality and cohesiveness of our project work.

I would also like to extend my sincere thanks to my friends, Ong Yi Xin, Iris Chiam and Zenith Cheng, for always being there with words of encouragement, companionship and laughter. Your presence brought me comfort during difficult times, and your perspectives helped me find clarity when I felt overwhelmed. Thank you for listening, understanding and reminding me to stay grounded.

Last but not least, I would like to acknowledge all those who, in one way or another, offered support, shared a kind word, or simply believed in me along the way. Whether in the form of emotional support, academic feedback or quiet encouragement, each contribution has left a meaningful impact on my journey.

To all of you, thank you from the bottom of my heart.

Table of Contents

1 Introduction **2**
1.1 Objectives 3
1.2 Problems 5
1.2.1 Mental Health Crisis and Emotional Well-being 5
1.2.2 Work-Life Imbalance and Burnout 6
1.3 Advantages and Contributions 8
1.3.1 Key Advantages 8
1.3.2 Contributions 8
1.4 Project Plan 10
1.4.1 Project Scope 10
1.4.2 Project Schedule 19
1.5 Project Team and Organization 21
1.6 Chapter Summary and Evaluation 22
2 Literature Review **26**
2.1 Project Background 26
2.1.1 Target Markets 26
2.1.2 Technology and Development Tools Used 31
2.2 Literature Review 34
2.2.1 Programming Languages and Frameworks 34
2.2.2 Development Tools and IDE 36
2.2.3 Backend and API Integration 37
2.2.4 AI Tools and Algorithms 38
2.2.5 Database and Data Storage 39
2.2.6 UI and UX Design 39
2.3 Feasibility Study 41
2.3.1 Technical Feasibility 41
2.3.2 Economical Feasibility 41
2.3.3 Operational Feasibility 42
2.3.4 Schedule Feasibility 43
2.4 Chapter Summary and Evaluation 45
3 Methodology and Requirements Analysis **47**
3.1 Methodology 48
3.1.1 Selection of Development Methodology 48
3.1.2 Application of Incremental Model in BetterU Development 49
3.2 Requirements Gathering Techniques 57
3.2.1 Observation 57
3.2.2 Online Research 58
3.2.3 Summary of Fact Gathering Results 60
3.3 Requirements Analysis 63
3.3.1 Use Case (UC) Diagram and Description 63
3.3.2 Functional Requirements 83

3.3.3 Non-Functional Requirements 88
3.4 Development Environment 91
3.4.1 Hardware and Device Specification 91
3.4.2 Software Tools and Platforms 93
3.4.3 Programming Languages and Frameworks 95
3.4.4 Database and Storage Services 96
3.4.5 Application Architecture and Deployment Environment 97
3.4.6 UI/UX Design Tools 97
3.4.7 External Libraries, APIs and Plugins 98
3.5 Chapter Summary and Evaluation 100
4 System Design 102
4.1 Sequence Diagram 103
4.2 State Chart Diagram 122
4.3 User Interface Design 126
4.4 Data Design 145
4.4.1 Class Diagram 145
4.4.2 Entity Relationship Diagram 146
4.4.3 Data Dictionary 147
4.5 Reports Design 197
4.5.1 Emotion Summary Report 197
4.5.2 Focus Performance Report 198
4.5.3 Goal Tracking Report 199
4.6 Process Design 201
4.7 Software Architecture Design 213
4.8 AI Algorithms 214
4.9 Chapter Summary and Evaluation 222
References 224
Appendices 230

Chapter 1

Introduction

1 Introduction

This chapter introduces the BetterU mobile application, a wellness and productivity system designed to support users in managing their daily tasks while maintaining emotional balance. The chapter will provide an overview of the project background, outline its objectives, explain the problem statements that led to its development, and describe the significance and benefits of the system. It will also define the scope and limitations of the project, clarifying which features are included from the current development.

BetterU integrates productivity management with emotional well-being by offering modules such as intelligent goal assistance, emotion diary logging, focus timer sessions, personalized user settings, and progress reporting. These core functions aim to promote mental clarity, structured time management, and self-reflection. The system will support input through speech, text, and image, while also offering gamified elements and AI-powered suggestions to motivate users.

However, the current scope of the project does not include features such as real-time therapist consultation, social media integration, or cloud synchronization across multiple devices. Advanced predictive analytics for diagnosing mental health conditions are also beyond the intended scope, as the system focuses primarily on self-guided improvement and emotional tracking.

1.1 Objectives

The objectives of the proposed BetterU system are **centered around improving users' productivity and emotional well-being** through the functionalities of its core modules. These modules are designed to address the challenges of mental health, work-life balance, and productivity management. Below are the detailed objectives related to each module:

Goals Assistance Module (shared)

The Goals Assistance Module is designed to **reduce users' cognitive burden and enhance planning efficiency by providing intelligent task management**. It extracts actionable tasks from speech input, auto-generates personalized schedules, supports real-time progress tracking, and adapts to changes through dynamic rescheduling. Additionally, the module offers context-aware suggestions via the Smart Advisor to help users stay on track. As a result, users experience improved organization, reduced stress from unmanaged tasks, and better control over their daily responsibilities.

Emotion Diary Note Module

The Emotion Diary Note Module aims to **facilitate emotional awareness and personal reflection to support mental wellness**. It allows diary entries in multiple formats, including text, image, and speech, and includes mood tagging to contextualize the emotional state. By analyzing language patterns, the module detects emotional trends and stress indicators over time. Meaningful entries are saved in a momentary library for future review. This functionality helps users develop greater emotional self-awareness, track their mental state more clearly, and take proactive steps to improve their well-being.

Focus Timer Module

The Focus Timer Module is developed to **promote deep work and self-discipline** through structured and gamified focus sessions. Users can initiate distraction-free periods with app usage restrictions to prevent multitasking. The module incorporates gamification elements, such as treasure digging, to motivate consistent focus and task completion. In-app rewards, including avatar frames, visual themes, and challenge points, are provided as positive reinforcement. This encourages increased concentration, reduces mental fatigue, and enhances overall productivity.

User Module

The User Module **ensures a secure, personalized, and consistent application experience**. It supports essential account functions such as user registration, login, and profile management. User preferences, including notification settings, visual themes, and diary visibility, are stored and applied to deliver a tailored interface. Through this module, users enjoy a seamless experience that aligns with their personal habits and minimizes unnecessary friction in daily use.

Report Module

The Report Module is responsible for **generating meaningful insights based on users' behavioral and emotional data**. It summarizes key activity metrics, including task completion rates, mood fluctuations, and focus durations. These data are presented through clear and intuitive visualizations, such as charts and graphs. The module highlights correlations between behavior and emotional well-being and offers self-assessment reports to support reflective decision-making. It empowers users to better understand themselves and maintain a balanced, informed lifestyle.

1.2 Problems

The proposed BetterU system is designed to address the following critical problems identified in the context of mental health and work-life balance in Malaysia:

1.2.1 Mental Health Crisis and Emotional Well-being

Problem Description:

- **Increasing Prevalence**
  - Mental health disorders, including depression, anxiety, and stress, are among the leading causes of disability globally, with Malaysia showing a similar upward trend. The Ministry of Health Malaysia (MOH, 2024) reports that approximately 29% of Malaysian adults suffer from mental health-related conditions, while 20.8% of individuals aged 16 to 24 experience psychological distress (NHMS, 2023). This highlights a significant increase in mental health issues across all age groups.
- **Stigma and Limited Access**
  - Mental health stigma remains deeply rooted in Malaysian society, discouraging many individuals, especially those in rural communities, from seeking treatment. The shortage of trained mental health professionals (only 0.25 psychiatrists per 100,000 people) exacerbates the problem, with uneven geographical distribution limiting access to mental health services in rural regions (Ahmad Adlan, 2022).
- **Socio-Economic Factors**
  - Financial insecurity, job instability, academic pressure, and social discrimination contribute to psychological distress. The COVID-19 pandemic further compounded these challenges, leading to widespread unemployment, income loss, and social disconnection, which in turn increased mental health-related issues (Bahmad, 2021; Abdul Yazid, 2023).

How BetterU Addresses the Problem:

- **Goals Assistance Module (shared)**
  - Reduces cognitive overload by automatically extracting tasks and generating personalized schedules. This helps users manage their responsibilities more effectively, thereby alleviating stress and promoting emotional stability.
- **Emotion Diary Note Module**
  - Provides a safe space for users to express and reflect on their emotions. Through mood tagging and emotional trend analysis, users gain better

self-awareness and can detect signs of emotional distress early, encouraging proactive mental health management.

- **Focus Timer Module**
    - Encourages users to engage in structured focus periods, reducing mental exhaustion and promoting emotional resilience through gamified rewards and positive reinforcement.
- **Report Module**
    - Offers detailed insights into users' emotional patterns and behaviors, helping them identify triggers and make informed decisions to improve their mental well-being.

1.2.2 Work-Life Imbalance and Burnout

Problem Description:

- **Increasing Work Demands**
    - Work-life imbalance is a major occupational health issue in Malaysia, with 55% of employees unable to achieve adequate balance due to rising employer expectations and technological overreach (MEF, 2023). The pervasive culture of overwork, particularly in high-pressure sectors like finance, healthcare, and education, contributes directly to burnout (HRDF, 2024).
- **Burnout Prevalence**
    - Burnout is characterized by emotional exhaustion, reduced personal accomplishment, and depersonalization. A national survey by PwC Malaysia (2024) revealed that 70% of respondents experienced high levels of stress, with many reporting feelings of detachment, fatigue, and disengagement from work. The normalization of remote work and digital communication platforms has further blurred the boundaries between work and personal life, exacerbating the problem.
- **Consequences**
    - Burnout leads to physical health problems, decreased productivity, higher absenteeism rates, and increased turnover. Dissatisfaction related to burnout is a key driver of attrition, with many workers seeking career changes or early retirement to avoid further psychological strain (PwC Malaysia, 2024).

How BetterU Addresses the Problem:

- **Goals Assistance Module (shared)**
    - Helps users organize their work and personal tasks more efficiently, preventing overcommitment and promoting better time allocation. The

module's smart advisor and real-time progress tracking ensure users maintain a balanced workflow.

- **Emotion Diary Note Module**
    - Allows users to document their emotional responses to work-related stressors, helping them identify patterns of emotional fatigue and take proactive steps to manage burnout.
- **Focus Timer Module**
    - Encourages structured work intervals and scheduled breaks, enhancing focus and preventing multitasking. The gamified reward system maintains motivation and promotes healthier work habits.
- **User Module**
    - Ensures a personalized experience, allowing users to configure settings to align with their working hours and preferences. This integration supports a seamless and less stressful user experience.
- **Report Module**
    - Provides visual and analytical overviews of users' work habits and emotional states, helping them identify signs of imbalance and adjust their routines for better work-life balance.

1.3 Advantages and Contributions

The proposed BetterU system provides multiple benefits by addressing two critical societal issues: mental health challenges and work-life imbalance. The application integrates intelligent modules that work cohesively to enhance productivity, emotional well-being, and daily structure.

1.3.1 Key Advantages

Smarter Time Management

- The Goals Assistance module extracts tasks from user input and auto-generates personalized schedules. This reduces cognitive overload and prevents burnout by promoting balanced daily planning.

Enhanced Emotional Awareness

- The Emotion Diary Note allows users to express emotions via text, image, or speech. It uses natural language processing to detect mood trends and promote reflection, helping users manage stress, anxiety, and emotional fatigue.

Improved Productivity and Focus

- The Focus Timer module encourages task completion by rewarding sustained concentration. Users experience increased efficiency through structured working sessions, reducing time wastage and supporting healthier work routines.

User Empowerment and Customization

- The User module enables users to personalize notifications, interface settings, and privacy preferences, ensuring a non-intrusive and user-friendly experience.

Insightful Feedback for Self-Improvement

- The Report module provides visual summaries of emotional patterns, focus trends, and task performance. This helps users make informed decisions to adjust habits and improve their overall well-being.

1.3.2 Contributions

BetterU offers a **holistic digital solution that bridges the gap between productivity tools and emotional support, effectively addressing both task management and mental wellness within a unified platform.** In an increasingly fast-paced and demanding world, individuals often struggle to maintain a healthy balance between personal responsibilities and professional commitments. BetterU directly responds to this challenge by promoting structured self-management and emotional well-being, ultimately contributing to the development of a more mentally resilient and focused society.

By integrating artificial intelligence, the system enables intelligent task extraction and adaptive planning, which supports personalized and proactive goal setting tailored to individual user behaviors and preferences. This not only improves users' productivity and time management but also reduces cognitive overload and decision fatigue, which are common contributors to stress and burnout.

The application encourages regular emotional check-ins through diary entries, speech input, and mood tagging, supported by sentiment analysis. These features enable users to become more self-aware and emotionally literate, while also supporting the early detection of negative emotional patterns or signs of distress. This proactive approach to mental health empowers users to seek help early, potentially reducing the risk of long-term psychological issues.

BetterU also fosters a sustainable lifestyle by helping users organize their daily activities in a way that balances professional performance with personal well-being. It creates a safe, private, and user-centered environment that encourages consistent self-reflection and mindful living. The platform's integrated reporting features allow users to track their behavior and

emotional trends over time through visual summaries, providing them with actionable insights and reinforcing positive habits.

From a social perspective, BetterU contributes to the broader goals of mental health awareness, digital well-being, and self-improvement. By providing individuals with tools to manage stress, stay motivated, and develop emotional intelligence, BetterU **supports healthier lifestyles and more resilient communities**. In the long term, such tools can ease the burden on mental health services, promote productivity in both academic and professional environments, and foster a more balanced, emotionally aware generation.

1.4 Project Plan

1.4.1 Project Scope

Figure 1.1: Hierarchy Chart for BetterU

Functional and Non-Functional Features

Functional Features

1.0 Emotion Diary Note Module

1.1 Diary Entry Management

1.1.1 The system shall allow users to create, edit, delete, and view diary entries.

1.1.2 The system shall support multiple entries "momentary" per day for quick reflections and future review.

1.1.3 The system shall support diary input via text, emojis, voice (speech-to-text), and image.

1.1.4 The system shall allow users to optionally attach images to diary entries.

1.1.5 The system shall automatically capture and display the creation date and time of each entry.

1.1.6 The system shall allow users to search entries by title, date, mood tags, or keyword highlights.

1.1.7 The system shall support sorting entries by creation date/time or mood.

1.1.8 The system shall support grouping entries by mood or by creation date range using a calendar view.

1.2 Sentiment Analysis & Mood Tagging

1.2.1 The system shall analyze diary content using AI-powered sentiment analysis (e.g., via Scikit-learn or NLP libraries).

1.2.2 The system shall automatically suggest mood tags based on detected emotional tone (e.g., Neutral, Happy, Sad, Anxious, Fearful, Angry, Mixed).

1.2.3 The system shall highlight keywords that indicate possible causes of stress, problems encountered, or sources of happiness.

1.2.4 The system shall allow users to manually edit mood tags and keyword highlights.

1.3 Privacy Control

1.3.1 The system shall offer optional PIN or biometric (e.g., fingerprint) protection for accessing or locking sensitive diary entries.

2.0 Focus Timer Module

2.1 Focus Session Control

2.1.1 The system shall allow users to select a task and specify a focus duration (15, 25, 45, 60 minutes, or custom).

2.1.2 The system shall support pause, cancel, resume, and reset controls during focus sessions.

2.1.3 The system shall support focus tracking tied to specific tasks for progress accumulation.

2.1.4 The system shall support phone flip detection to initiate or maintain focus mode.

2.1.5 The system shall display a visual heatmap (similar to GitHub contribution graph) to represent daily focus completion history.

2.1.6 The system shall allow users to view their session history, rewards earned, and personal progress trends.

2.2 Gamification

2.2.1 The system shall trigger start/end sound effects and visual animations during focus sessions.

2.2.2 The system shall reward users with virtual “treasures” upon completing a focus session, where both the probability of obtaining rare items and the

total number of rewards shall scale with the session duration (e.g., longer sessions yield higher drop rates and multiple items).

2.2.3 The system shall allow users to collect virtual “treasures” and convert duplicate items into gems.

2.2.4 The system shall include a shop where users can exchange gems for cosmetic or premium features (e.g., limited-time avatar frames, wallpapers, feature trial days).

2.2.5 The system shall track and display daily focus streaks to encourage consistency.

2.3 Distraction Prevention

2.3.1 The system shall monitor app usage during focus sessions and pause the timer if apps outside the whitelist are accessed.

2.3.2 The system shall notify users with a warning if distraction rules are violated.

2.3.3 The system shall allow users to manage their app whitelist manually.

2.3.4 The system shall maintain a leaderboard showing top focus durations within the user's community (e.g., StudyTogetherCommunity) and globally (Overall Top 10).

3.0 User Module

3.1 User Registration & Authentication

3.1.1 The system shall allow new users to register by providing a username, password, email, gender, and occupation.

3.1.2 The system shall validate that the email is in the correct format and enforce a strong password policy.

3.1.3 The system shall send a One-Time Password (OTP) to the user’s email for verification during registration.

3.1.4 The system shall allow users to log in using either their username or email together with their password.

3.1.5 The system shall provide a “Forgot Password” function that sends an OTP to the user’s email for password recovery.

3.1.6 The system shall check for duplicate usernames and notify the user if the chosen username is already taken.

3.2 User Profile & Preferences

3.2.1 The system shall allow users to update their profile information, including profile image, avatar frame, username, gender, and occupation.

3.2.2 The system shall ensure that usernames remain unique when users update them.

3.2.3 The system shall assign user roles such as User, VIP, or Admin and restrict access to features based on these roles.

3.2.4 The system shall grant VIP users access to exclusive features, including unlimited AI functionalities.

3.2.5 The system shall allow users to select application themes, such as light mode or dark mode.

3.2.6 The system shall allow users to configure notification preferences, including the option to silence notifications.

3.2.7 The system shall allow users to enable or disable PIN or fingerprint protection for diary entries.

3.3 Administration

3.3.1 The system shall provide administrators with a dashboard that displays the total number of active users along with user retention metrics.

3.3.2 The system shall allow administrators to view and manage new requests for community group creation or joining.

3.3.3 The system shall enable administrators to review and moderate flagged posts, messages, and user accounts.

3.4 Session Control

3.4.1 The system shall manage secure login sessions using token-based authentication with automatic timeout and refresh mechanisms.

3.4.2 The system shall provide users with the option to log out manually and terminate all active sessions.

4.0 Report Module

4.1 Insight Generation

4.1.1 The system shall generate weekly, monthly, and yearly summaries that reflect the user's completed and pending tasks, focus sessions, and emotional records.

4.1.2 The system shall allow users to view and filter reports based on custom time ranges and selected data types, such as goals, focus duration, or mood trends.

4.1.3 The system shall provide a visual comparison between to-do tasks and completed tasks to help users evaluate their productivity.

4.2 Data Visualization

4.2.1 The system shall present data using various chart formats, including line graphs, pie charts, and bar charts.

4.2.2 The system shall visualize trends and correlations among task completion, focus time, and emotional states.

4.2.3 The system shall allow users to view progress and consistency through weekly, monthly, or yearly breakdowns in a calendar-style or timeline layout.

4.3 Notifications & Reflection

4.3.1 The system shall send users personalized summary insights via in-app notifications or email, if enabled.

4.3.2 The system shall provide motivational feedback, encouragement messages, or suggestions for habit improvement based on detected patterns and behavior gaps.

4.3.3 The system shall offer self-assessment prompts to encourage users to reflect on progress, stressors, and achievements.

4.3.4 The system shall allow users to export their reports and visual summaries as downloadable PDF files for personal tracking or external sharing.

5.0 Goal Assistance Module (shared)

5.1 Task Management

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

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

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

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

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

5.1.6 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).

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

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

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

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

5.2 Smart Input & Suggestions

5.2.1 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").

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

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

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

5.3 Smart Advisor

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

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

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

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

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

Non-Functional Features

Product

1.0 Availability

1.1 The system shall maintain 99.95% uptime, minimizing interruptions in daily use.

1.2 Scheduled maintenance shall only occur during non-peak hours and not exceed 2 hours per month.

1.3 In the event of unexpected Firebase service disruption, the app shall attempt automatic reconnection and restore functionality within 5 minutes.

2.0 Functional

2.1 The system shall ensure full feature accessibility and consistency across Android and iOS platforms.

2.2 All core modules (Goals Assistance, Emotion Diary, Focus Timer, Reports, and User Account) must function reliably on smartphones.

2.3 The system shall utilize Firebase services (e.g., Firestore, Authentication, Cloud Functions) for backend functionalities.

3.0 Usable

3.1 The app shall feature an intuitive and beginner-friendly UI, allowing users to navigate core features (e.g., setting goals, starting timers, writing emotion logs) without external help.

3.2 All UI components shall follow consistent design patterns as defined in the Figma design system.

3.3 The system shall support theme personalization for better user comfort and accessibility.

4.0 Reliable

4.1 The app shall support up to 10,000 concurrent users without noticeable performance issues using Firebase backend scalability.

4.2 The system shall prevent data loss by utilizing Firebase offline persistence and real-time synchronization to ensure local changes are synced when connectivity resumes.

4.3 Emotion logs, goal history, and focus sessions shall be backed up in real-time to cloud storage.

5.0 Flexible

5.1 The system shall support localization, including multi-language support, adjustable time/date formats, and culturally adapted content presentation.

5.2 The app shall allow easy updates of localized text through Firebase Remote Config or dedicated translation files.

Organization

1. The system shall be developed using the Dart programming language with the Flutter framework.
2. The system shall use Firebase as the primary backend solution, including Firestore (database), Firebase Auth, and Firebase Cloud Functions.
3. Version control shall be managed via Git and hosted on GitHub, with team collaboration and issue tracking enabled.
4. The UI/UX design shall be created and iterated using Figma.
5. AI-powered features (e.g., emotion analysis, goal suggestion) shall be implemented using Scikit-learn and Pandas via backend Python microservices.

External

1. The app shall integrate Google or Apple Sign-In for secure user authentication.
2. If in-app purchases or donations are introduced, payment processing shall be handled via Google Play and Apple App Store's native payment APIs.
3. All user data handling must comply with relevant privacy laws (e.g., GDPR, CCPA) and Google/Firebase security best practices.

1.4.2 Project Schedule

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

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

Figure 2.2: Incremental Model Diagram for 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

1.1 Objectives

The BetterU system aims to enhance productivity and emotional well-being through five core modules:

- Goals Assistance Module (Shared) :
  Intelligent task management via smart scheduling, task extraction from speech, adaptive rescheduling, and real-time progress tracking.
- Emotion Diary Note Module:
  Emotional tracking through text, image, or speech entries, mood tagging, sentiment analysis, and a personal reflection library.
- Focus Timer Module:
  Gamified focus sessions with app usage restrictions, in-app rewards, and progress tracking to encourage deep work.
- User Module:
  Secure and personalized user management with account control, preference storage, and customizable settings.
- Report Module:
  Visual summaries and insights on user behavior, mood trends, and focus patterns to support self-assessment and decision-making.

1.2 Problems

Mental Health and Emotional Well-being:

- Rising mental health issues in Malaysia, stigma, limited professional access, and socio-economic pressures.
- BetterU Solutions: Smart task management, emotional tracking, gamified focus sessions, and actionable insights for early stress detection and self-care.

Work-Life Imbalance and Burnout:

- Overwork culture, high stress levels, blurred work-life boundaries, and increasing burnout cases.
- BetterU Solutions: Efficient task organization, emotional journaling, structured work intervals, and personalized user experiences to promote balance.

1.3 Advantages and Contributions

- Improves productivity and focus through gamification.
- Enhances emotional awareness via diary and sentiment analysis.
- Supports smarter time management with intelligent scheduling.
- Offers user-friendly customization for comfort and control.
- Provides insightful feedback to empower self-improvement.
- Bridges productivity tools and emotional support in a single app.
- Encourages proactive mental health management.
- Promotes a balanced and sustainable lifestyle.

1.4 Project Plan

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.

1.5 Project Team and Organization

The Table 1.1 outlines the project team and organization for BetterU, specifying the modules each team member is responsible for. Here's a summary:

Chia Ming Yi will be responsible for:

- Emotion Diary Note
- Focus Timer
- User
- Report

Lim Jun Wei will be responsible for:

- Time Usage Tracker
- Anonymous Community
- Personal Chat
- Report

Both of us will be responsible for:

- Goal Assistance

Chapter 2

Literature Review

2 Literature Review

Chapter 2 provides a review of the background, technologies and tools applied in the development of BetterU mobile application. It identifies the target users of this mobile application such as students and working professionals. Meanwhile, it also explores the challenges faced by them when managing productivity and emotional well-being. On the other hand, this chapter also highlights the core technologies used in BetterU, including programming languages, development frameworks, backend systems, AI tools and UI/UX design platforms. Eventually, it examines the feasibility of the project from technical, economic, operational and scheduling perspectives to ensure the practicality and effectiveness of the proposed solution.

2.1 Project Background

2.1.1 Target Markets

BetterU is designed to support **two primary target markets in Malaysia: students and working professionals**. Both groups face unique challenges that impact their productivity, emotional well-being, and ability to manage daily responsibilities.

1. Students (Secondary and University Level in Malaysia)

Students, particularly those at the secondary and university levels, often struggle with overwhelming academic demands, co-curricular activities, and personal obligations. These pressures can lead to high stress, disrupted routines, and emotional exhaustion. The BetterU assists students by offering structured modules that help break down complex study plans into manageable tasks, track emotional trends through a flexible diary feature, and encourage focused study sessions via a gamified timer. Students can personalize their app settings to fit their academic schedules and receive visual progress reports that enhance their self-awareness and promote healthier study habits.

Demographic Analysis:

- Age Range: 13 – 25 years old
    - This age group includes adolescents and young adults who are frequent smartphone users and are highly engaged with mobile applications. They are part of a generation that is comfortable using digital solutions for learning, task management, and mental health support.
- Education Level: Secondary school, pre-university, college, and university students

    - Students within these education levels typically experience escalating workloads, complex assignments, and exam pressures. The demand for efficient planning and task management increases as they progress through their academic journey.
- Occupation: Full-time students
    - The application is tailored for students whose primary focus is on their studies. These students often juggle multiple responsibilities such as coursework, co-curricular activities, part-time jobs, and personal commitments, making them ideal beneficiaries of a productivity-focused solution.
- Location: Primarily urban and semi-urban areas
    - Students in urban and semi-urban regions are more likely to have consistent access to stable and fast internet connections, which are essential for the optimal use of BetterU's mobile application features. These areas also typically show higher rates of mobile app adoption and usage.
- Device Access: Regular access to smartphones
    - The target students commonly own smartphones that run on widely supported operating systems such as Android and iOS. This consistent device access enables them to use BetterU anytime and anywhere, ensuring the app's practicality in their daily academic and personal routines.

Psychographic Analysis:

- Lifestyle: Students within this target market typically manage demanding schedules that include schoolwork, co-curricular activities, frequent examinations, and sometimes part-time employment. These overlapping commitments often leave students with limited personal time, increasing their stress levels and reducing opportunities for socialization. Especially in top-ranked schools and competitive universities, students are pressured to maintain high academic performance while balancing multiple responsibilities.
- Goals: Students aim to achieve their academic targets while efficiently managing their tasks and deadlines. In addition to academic performance, they seek to maintain emotional stability, build self-confidence, and preserve social interactions, which are essential for their overall mental well-being.
    - Academic success
    - Improved time management
    - Emotional balance
    - Stronger peer and social connections

- Values: The target students value continuous growth, effective time use, and becoming more organized. They are also becoming increasingly aware of the importance of maintaining their mental health, seeking solutions that can support both their academic and emotional needs.
    - Self-improvement
    - Productivity and efficiency
    - Mental health awareness
- Challenges: Many students face persistent procrastination and have difficulty structuring their tasks clearly, which often results in incomplete work and increased anxiety. Their high sensitivity to surrounding distractions makes it even more challenging to stay focused, especially when under pressure.
    - Procrastination tendencies
    - Difficulty in task organization
    - High stress from multitasking
    - Struggles with maintaining focus due to environmental distractions

Behavioral Analysis:

Modern students heavily rely on mobile devices for learning, productivity, communication, and relaxation. Their daily routines are closely intertwined with mobile applications that support both academic tasks and social engagement. BetterU is designed to meet their behavioral patterns and preferences by offering a simplified, student-centered task management experience.

- Technology Usage: Students exhibit high smartphone dependency throughout the day for various purposes:
    - Accessing online learning resources (text, images, videos)
    - Using task management applications for scheduling and organizing assignments
    - Engaging with social media to communicate with family, classmates, and online communities
    - Consuming entertainment content for relaxation
- Technology Habits: Students typically access mobile applications at specific times of the day:
    - Before school: Checking social media updates and reviewing task reminders
    - After school: Using entertainment apps, chatting with friends, and reviewing pending tasks or assignments
    - After completing tasks: Returning to social media platforms or entertainment apps to unwind and stay socially connected

- Application Preferences:
    - User-friendly: Simple navigation and intuitive interface
    - Flexible: Adaptable to changing schedules and needs
    - Personalized: Allow customization to fit individual preferences
    - Simple to use: Straightforward features without overwhelming complexity
- Pain Points:
    - Many existing task management applications are overly complex and include features that are not relevant to students' daily needs.
    - The unnecessary features in these apps make them difficult to adopt and discourage consistent usage.
    - Students may feel overwhelmed or disinterested in apps that require extra effort to learn or configure.

2. Working Professionals (Urban Office Workers in Malaysia)

Working professionals, especially urban office workers in Malaysia, frequently encounter work-life imbalance due to heavy workloads, digital overexposure, and the blurred boundaries between work and rest in hybrid work environments. This often results in chronic stress, emotional detachment, and decreased productivity. The system addresses these issues by helping professionals organize tasks, balance personal and work commitments, monitor emotional well-being, and maintain focus through structured timers. The solution is highly customizable, offering settings that align with individual routines and lifestyles, while providing regular reports to help users detect early signs of burnout and optimize their work-life balance.

Demographic Analysis:

- Age Range: 22 – 45 years old
    - This age group includes young to mid-career professionals who are actively engaged in building their careers. They are experienced with digital tools and are frequent users of productivity and communication applications.
- Occupation: Full-time office workers, administrative staff, junior to mid-level professionals
    - These individuals primarily work in office environments or remote settings, dealing with task-heavy roles that require continuous coordination, communication, and deadline management.
- Education Level: College graduates or higher

    - Most urban office workers hold at least a college degree, equipping them with the skills and knowledge to manage complex work responsibilities using digital tools.
- Location: Primarily urban and semi-urban areas with good Internet infrastructure
    - The target professionals are located in areas where fast and stable internet connectivity is accessible, which is crucial for the seamless operation of BetterU's mobile application.
- Device Access: Regular access to smartphones, laptops, and desktop computers
    - These professionals frequently use multiple devices for work and task management, including smartphones, laptops, and desktops, ensuring consistent connectivity and access to task management tools.

Psychographic Analysis:

- Lifestyle: Office workers in this segment manage packed schedules with long hours in front of screens. Their daily routines often involve extended working hours, overtime, and frequent digital communication. As a result, they face limited opportunities for personal and social interactions outside of work.
- Goals: They desire effective ways to balance professional success with personal well-being while minimizing mental fatigue. They aim to achieve:
    - Work-life balance
    - High work efficiency
    - Reduced stress
    - Continuous career growth
- Values: They seek solutions that can help them optimize task completion, reduce unnecessary workload stress, and protect their mental health.
    - Time management
    - Productivity
    - Mental well-being
- Challenges: Without efficient planning tools, they risk confusion, wasted time, and decreased productivity, often leading to frustration and burnout. Many office workers struggle with:
    - Limited time and effort to create detailed work plans
    - Losing focus and feeling overwhelmed by task overload
    - Poor task organization
    - Difficulty balancing work responsibilities with personal life

Behavioral Analysis:

Office workers frequently rely on productivity applications such as Google Calendar, Notion, and Slack to manage their tasks, track meetings, and take notes. They also use communication platforms like Zoom and Microsoft Teams for virtual meetings and real-time collaboration.

- Technology Usage:
    - Heavy use of productivity apps and task management tools
    - Dependence on communication apps for online meetings and workplace coordination
    - Multi-device access throughout the workday (smartphones, laptops, desktops)
- Habits:
    - Checking notifications and task reminders in the morning before work
    - Accessing calendars and scheduling apps during office hours to track daily tasks
    - Using communication platforms to receive assignments and update progress throughout the day
    - Reviewing remaining tasks after work to plan for the next day
- Application Preferences:
    - Minimalist design for easy navigation and quick access
    - Responsive interface with real-time updates and fast synchronization
    - Highly customizable task management features to fit individual workflows
- Pain Points:
    - Current task management apps often lack real-time tracking and smooth synchronization across devices
    - Many apps include generic, non-personalized features that do not support the unique demands of individual users
    - Complex interfaces and manual configuration requirements increase user effort instead of simplifying task management

2.1.2 Technology and Development Tools Used

Development Platform and Framework

BetterU is developed using the Flutter framework with the Dart programming language. Flutter was chosen for its ability to streamline development through a single codebase, significantly reducing complexity and development time. Its extensive library of customizable widgets and strong community support also enable rapid prototyping and flexible UI design, making it well-suited for implementing the features and user experience envisioned in the BetterU solution.

Database

Google Firebase is used by the BetterU mobile application for real-time data storage and synchronization. It offers seamless integration with Flutter and provides a range of backend services such as authentication, cloud storage, and analytics. As a free and developer-friendly platform, Firebase can be easily implemented into the BetterU app, effectively reducing both development time and costs.

AI or ML Features

Speech Recognition (Vosk, Whisper)

BetterU integrates speech recognition capabilities using Vosk and Whisper to provide real-time and high-accuracy transcription from user voice input. These Automatic Speech Recognition (ASR) models support multi-language input and automatic language detection, which enhances accessibility for diverse users. This feature allows users to input task items through speech, improving convenience and usability, especially in mobile contexts where typing may not be practical (Radford et al., 2023; Vosk API, 2020).

Task Parsing and Intent Detection (spaCy, Hugging Face)

To interpret the meaning and structure of user input, BetterU uses Natural Language Processing (NLP) tools such as spaCy and Hugging Face Transformers. These tools tokenize, parse, and extract relevant information like dates and times from both voice and text entries. This process supports automatic scheduling by identifying the most likely task deadlines and refining task descriptions to be concise, relevant, and easily readable (Explosion AI, 2023; Wolf et al., 2020).

Task Classification and Prioritization (Scikit-learn)

Scikit-learn is utilized to classify tasks into different categories (e.g., work, health, personal) and to assign priority levels based on urgency and importance. The model is trained on labeled datasets to detect common patterns, enabling it to automate task sorting and reduce the user's cognitive load. This intelligent prioritization ensures that users focus on what matters most while reducing manual overhead (Pedregosa et al., 2011).

Personalized Recommendations (LightFM, Embeddings)

The Smart Advisor module in BetterU leverages a recommendation system built with LightFM and matrix factorization techniques. By analyzing user preferences, habits, and contextual behavior, the system provides tailored suggestions for goals, tasks, or productivity strategies. Embedding techniques allow the model to learn fine-grained patterns across users and tasks, improving recommendation accuracy over time (Kula, 2015).

Time Optimization Engine (OR-Tools, RLib)

BetterU integrates time optimization algorithms using Google OR-Tools for constraint solving and reinforcement learning through RLib to adaptively plan user schedules. These techniques consider user availability, task durations, and predefined constraints to generate efficient and feasible schedules. Reinforcement learning enables the system to improve scheduling outcomes based on continuous feedback (Google OR-Tools, 2023; Liang et al., 2018).

Adaptive Rescheduling (Random Forest, LSTM)

To dynamically respond to changes in user behavior, BetterU applies machine learning models such as Random Forest and Long Short-Term Memory (LSTM) networks. These models are trained to detect behavior patterns and predict potential delays or interruptions. Based on these predictions, the app can suggest task rescheduling or notify users of potential conflicts in advance, enhancing flexibility and reliability (Hochreiter & Schmidhuber, 1997; Breiman, 2001).

UI Design Tool

Figma is utilized for designing the graphical user interface (GUI) of the BetterU mobile application. Figma is a cloud-based design tool that allows multiple team members to work simultaneously on the same interface design. This real-time collaboration enhances team productivity by enabling parallel work on different modules or screens, reducing design cycle times and minimizing version conflicts (Nelson, 2020).

Figma is accessible across various devices, including smartphones, tablets, and laptops, which ensures flexibility and convenience for remote or hybrid development teams. As an interactive prototyping tool, Figma supports the creation of dynamic components that respond to user actions, such as clicks, hovers, or transitions. This functionality allows for more realistic and engaging prototypes during the development phase, making it easier to visualize user interactions and gather feedback before actual implementation. Moreover, its freemium pricing model helps control design costs, aligning with the project's goal of cost-effective development.

Through Figma, BetterU’s development and design teams can stay synchronized throughout the design process, ensuring a cohesive and user-centered interface experience.

2.2 Literature Review

2.2.1 Programming Languages and Frameworks

Dart

Dart is a client-optimized, object-oriented programming language developed by Google, designed to build fast, scalable applications across multiple platforms (Dart, n.d.). In BetterU, Dart is primarily used to implement the core app logic and to integrate tightly with Flutter's UI framework, enabling smooth and consistent UI adaptation across different platforms.

- **Powerful Performance**
    - Dart employs Ahead-of-Time (AOT) compilation, which compiles the code into native machine code before the app runs. This significantly improves app startup time and runtime performance, resulting in a smooth user experience (Code Carbon, 2023). Additionally, during development, Dart supports Just-In-Time (JIT) compilation that allows developers to see live updates immediately as code changes are made, without restarting the app. This capability greatly enhances development efficiency by shortening the edit-test cycle (MoldStud, 2023).
- **Cross-Platform Development**
    - Dart is designed to work seamlessly with Flutter, allowing a single codebase to produce apps that run consistently on both Android and iOS devices (Java Code Geeks, 2022). This unification of development saves significant time and reduces maintenance efforts, since the same codebase can be reused for multiple platforms rather than developing separate native apps.
- **Object-Oriented and Familiar Syntax**
    - Dart's syntax resembles popular languages such as Java and C++, making it easier for developers with prior experience in these languages to transition smoothly. Its strong typing system helps catch errors early during development, improving code reliability and robustness (Java Code Geeks, 2022; MoldStud, 2023).

Python

Python is a versatile, easy-to-learn programming language widely adopted in mobile app backend development and AI-related functionalities (MobileAppDaily, n.d.; Monterail, n.d.). For BetterU, Python plays a critical role in implementing powerful backend logic and AI features that support advanced user needs.

- **Powerful AI and NLP Libraries**
    - Python offers a rich ecosystem of artificial intelligence (AI) and natural language processing (NLP) libraries such as spaCy, Scikit-learn, and SetFit. These libraries enable BetterU to implement features like speech-to-text

conversion, task extraction, and intelligent task categorization with relatively minimal effort (spaCy, n.d.; Scikit-learn, n.d.; Chamrah, 2023).

- **Backend Development**
    - Using FastAPI, a modern Python web framework, BetterU's backend APIs can efficiently connect the mobile frontend with AI services. FastAPI's straightforward syntax and automatic documentation generation promote rapid prototyping and high accessibility for AI models, streamlining the development of complex AI-driven functionalities (FastAPI, n.d.).
- **Clean Separation of Concerns**
    - Python handles backend processing and AI logic, while Dart combined with Flutter manages the frontend user interface and lighter logic. This clear division of responsibilities makes the system easier to develop, debug, and maintain by isolating concerns into separate layers (FastAPI, n.d.).

Flutter

Flutter is an open-source UI toolkit created by Google for building high-performance, natively compiled applications from a single codebase across multiple platforms (Flutter, n.d.). In BetterU, Flutter serves as the foundation for delivering a visually appealing and consistent user experience.

- **Cross-Platform Development Efficiency**
    - Flutter enables BetterU to be deployed not only on Android and iOS but also on web and desktop platforms with consistent look and behavior. This broad coverage significantly reduces the effort needed for development and maintenance compared to building separate apps for each platform (Monterail, n.d.).
- **Abundant and Customizable UI Components**
    - Flutter offers a rich set of customizable widgets that allow developers to create intuitive, interactive user interfaces with smooth animations and transitions. This flexibility supports the development of a seamless and engaging user experience in BetterU (Anuyat, n.d.).
- **Hot Reload for Fast Development**
    - One of Flutter's standout features is hot reload, which lets developers instantly view code changes reflected in the running app without restarting it. This capability accelerates the design, development, and testing phases, especially when fine-tuning UI elements and behaviors (Unfolding the Advantages, n.d.).

2.2.2 Development Tools and IDE

Android Studio

Android Studio is Google's official Integrated Development Environment (IDE) tailored for mobile application development, providing comprehensive support for Flutter through dedicated plugins (Cellular Insider, n.d.). As the main development environment for the BetterU frontend, Android Studio streamlines the entire UI creation process.

- **Flutter Plugin Integration**
    - The Flutter plugin within Android Studio automates many repetitive tasks such as project setup and configuration, allowing developers to focus on coding and design. It supports Flutter's hot reload feature, which enables immediate preview of UI changes without restarting the application, significantly accelerating the development cycle and enabling rapid iteration (Cellular Insider, n.d.).
- **Built-in Emulator**
    - Android Studio includes Android Virtual Devices (AVDs), which simulate a variety of Android devices with different screen sizes, resolutions, and API levels. This virtual testing environment allows developers to debug and test the BetterU app across multiple device profiles without needing physical hardware, improving testing flexibility and reducing costs (Android Developers, n.d.).
- **Intelligent Code Editor**
    - The IDE offers an intelligent code editor with features like code completion, syntax highlighting, and real-time error detection. Advanced debugging tools such as breakpoints, step-through execution, and variable inspection help developers identify and fix issues efficiently, improving code quality and reducing development time (Cellular Insider, n.d.).

Visual Studio Code (VS Code)

Visual Studio Code (VS Code) is the chosen lightweight yet powerful code editor for BetterU's Python backend development (Microsoft, n.d.). Its versatility and extensibility make it well suited for developing backend APIs and AI logic.

- **Integrated Terminal**
    - VS Code's built-in terminal allows developers to run shell commands, manage virtual environments, and install packages without leaving the editor interface. This seamless integration promotes a smooth workflow by eliminating the need to switch between multiple tools (Microsoft, n.d.).

- **API Development Support**
    - With Python extensions, VS Code offers real-time syntax and error checking, intelligent code completion, and debugging capabilities. These features facilitate the development of FastAPI backend services, which interface efficiently with BetterU's Flutter frontend, enabling quick iteration and integration of new features (FastAPI, n.d.).

GitHub Desktop

GitHub Desktop is an open-source graphical user interface for Git version control that simplifies repository management and collaborative development (GeeksforGeeks, 2022). It plays a critical role in coordinating work among BetterU's development team.

- **Branch and Commit Management**
    - The tool streamlines essential Git operations such as staging changes, writing descriptive commit messages, creating and switching branches, and merging code. This structure helps keep feature development organized and traceable, facilitating parallel workflows and agile development (GeeksforGeeks, 2022).
- **Conflict Resolution**
    - GitHub Desktop provides intuitive interfaces for resolving merge conflicts that arise when multiple developers modify the same files. This reduces integration errors and helps maintain a stable codebase in a collaborative environment (GeeksforGeeks, 2022).
- **Improved Collaboration**
    - By enabling real-time syncing and pushing of code changes to remote repositories, GitHub Desktop minimizes the risk of conflicting edits and ensures all team members work with the latest codebase. This fosters efficient teamwork and continuous integration practices (GeeksforGeeks, 2022).

2.2.3 Backend and API Integration

FastAPI

FastAPI is a modern, high-performance Python web framework for building APIs using standard Python type hints (FastAPI, n.d.). It bridges BetterU's Flutter frontend and Python backend, enabling smooth communication for AI features like task extraction and speech recognition. FastAPI handles HTTP requests and JSON responses efficiently (DEV Community, 2023).

2.2.4 AI Tools and Algorithms

Vosk

Vosk is an offline speech recognition toolkit that supports multiple languages and enables real-time voice-to-text transcription without requiring an internet connection (Vosk API, n.d.). In BetterU, Vosk provides robust offline transcription capabilities, ensuring user privacy and functionality even in environments with limited connectivity.

Whisper

Whisper, developed by OpenAI, is employed alongside Vosk to enhance transcription accuracy, particularly for English and Mandarin voice inputs (OpenAI, n.d.). Whisper's complementary use enables BetterU to deliver precise speech recognition across multiple languages and accents, improving user experience.

spaCy

spaCy is an advanced Natural Language Processing (NLP) library designed for efficient linguistic analysis, including tokenization, part-of-speech tagging, and syntactic parsing (spaCy, n.d.). BetterU leverages spaCy to extract key task information such as descriptions, deadlines, and relevant entities from transcribed voice inputs, enabling intelligent task management.

dateparser

dateparser is a Python library specialized in interpreting natural language date and time expressions (dateparser, n.d.). Integrated with spaCy, it converts ambiguous temporal phrases such as "next Friday" or "at 3 pm" into standardized datetime objects, facilitating accurate task scheduling within BetterU.

Scikit-learn

Scikit-learn is a versatile Python machine learning library used for feature extraction, model building, and training (Scikit-learn, n.d.). BetterU employs Scikit-learn to develop predictive models that categorize user tasks by type and assign priority levels, enabling automated task organization and prioritization.

SetFit

SetFit fine-tunes sentence transformer models to perform classification tasks efficiently on small labeled datasets (Chamrah, 2023). Within BetterU, SetFit enhances the Goal Assistance module by improving task categorization accuracy while minimizing the amount of required training data.

2.2.5 Database and Data Storage

Google Firebase

Google Firebase is a backend-as-a-service (BaaS) platform offering scalable, secure, real-time database and storage solutions for BetterU (Google, n.d.).

- **Realtime Database and Firestore**
  - Firebase's NoSQL databases store data as JSON trees, providing real-time synchronization across clients and enabling instant updates without manual refresh (Firebase Docs, n.d.; Estuary Dev, 2023). This supports BetterU's dynamic task data and user preferences.
- **Authentication and Security**
  - Firebase supports multiple authentication methods (email, Google, Facebook) and enforces security rules, ensuring data privacy and authorized access (Airbyte, n.d.).
- **Cloud Storage**
  - Firebase Cloud Storage handles large media files securely and efficiently, ideal for posts and media in BetterU's community and chat features (Firebase Docs, n.d.).

2.2.6 UI and UX Design

Figma

Figma is a web-based design tool used for UI/UX design and prototyping in the development of BetterU (Jimmy Viquez, n.d.).

- **Seamless Collaboration**
  - Figma supports real-time multi-designer collaboration, allowing team members to work simultaneously on the same design file. Features such as visible collaborator cursors and comprehensive version history facilitate safe, efficient teamwork and minimize conflicts during the design process (Design Project, 2023).
- **Consistent and Scalable Design System**
  - By utilizing reusable components and variants, Figma enables BetterU to maintain a consistent visual language across all user interface elements. This approach allows global updates to be applied quickly and efficiently, ensuring UI scalability and reducing duplicated efforts (Design Project, 2023).
- **Rapid Prototyping**

- Figma's interactive and clickable prototypes can be created rapidly without the need for coding. This capability supports early-stage usability testing, enabling the team to gather user feedback and make design improvements before development, ultimately minimizing costly rework (Design Project, 2023).

2.3 Feasibility Study

2.3.1 Technical Feasibility

Hardware Feasibility

The BetterU mobile application is compatible with standard mobile devices running Android and iOS operating systems since it is built using Flutter, which supports cross-platform development (Flutter, n.d.). For design and coding, a Windows laptop or desktop is sufficient for developing the Flutter frontend and Python backend. Additional GPU resources can further accelerate the training and testing of AI models efficiently (Monterail, n.d.). During the testing phase, Android or iOS physical devices will be used to validate the features within the installed BetterU application.

Software Feasibility

Several essential software and platforms are required for the BetterU development process. Android Studio is used for building the Flutter frontend using Dart and provides virtual devices to simulate physical devices for debugging and testing (Cellular Insider, n.d.). Visual Studio Code is primarily used for backend development, including Python-based API services and AI model training using FastAPI (FastAPI, n.d.). VS Code also supports integration with numerous open-source AI and NLP libraries such as spaCy, Vosk, Whisper, and SetFit, which are crucial for implementing BetterU's AI features (spaCy, n.d.; Vosk API, n.d.; OpenAI, n.d.; Chamrah, 2023). Also, Figma is employed for user interface (UI) design and prototyping. Figma's design tools allow the BetterU team to simulate the user flow and design structure efficiently before proceeding to code implementation (Design Project, 2023).

2.3.2 Economical Feasibility

The BetterU mobile application project is economically feasible as it primarily relies on free and open-source tools, which significantly minimize the financial investment required for development.

- Cost of Development Tools

- All core development tools used in BetterU, including Flutter, Dart, Python, and FastAPI, are open-source and freely available (Flutter, n.d.; FastAPI, n.d.; Monterail, n.d.). This eliminates the need for purchasing costly proprietary software licenses, resulting in a highly cost-efficient development process.

- Cloud Services
  - BetterU utilizes Google Firebase's free-tier cloud services, which provide essential features such as real-time database management, user authentication, and cloud storage (Firebase, n.d.). These free-tier offerings are sufficient to support the application's data storage, user management, and synchronization needs during initial development and deployment phases, further reducing operational expenses.
- Version Control and Collaboration
  - GitHub Desktop and GitHub repositories are freely available tools that facilitate seamless code collaboration, version control, and real-time synchronization among BetterU developers (GitHub Desktop, n.d.). These capabilities enhance team productivity and ensure effective project management without additional cost.
- AI Tools and Libraries
  - The AI-driven components of BetterU leverage open-source libraries such as Vosk for offline speech recognition, Whisper for voice transcription, and spaCy for natural language processing (Vosk API, n.d.; OpenAI, n.d.; spaCy, n.d.). Utilizing these freely accessible tools allows for sophisticated AI functionalities without incurring expenses on commercial AI services or licenses.
- Overall Project Cost
  - By employing open-source software, free cloud services, and no-cost AI libraries, the BetterU project avoids significant direct financial expenditures. This strategic choice enhances the project's sustainability, cost-effectiveness, and overall financial viability for both current development and future improvements.

2.3.3 Operational Feasibility

The BetterU mobile application is operationally feasible, supporting seamless deployment on widely used platforms and devices while enabling easy backend accessibility and web-based management.

Cross-Platform Compatibility

BetterU is developed using the Flutter framework, which facilitates cross-platform deployment (Flutter, n.d.). However, the current project scope focuses exclusively on Android devices, supporting API level 21 (Android 5.0 Lollipop) and above. By targeting this broad range of Android versions, BetterU ensures compatibility with the majority of modern smartphones, providing smooth performance without requiring high-end hardware. The application will not be developed or deployed for iOS platforms.

Backend Accessibility

The backend services of BetterU are implemented with FastAPI, allowing efficient integration with the Flutter frontend through RESTful API calls over HTTP/HTTPS (FastAPI, n.d.). This architecture supports real-time data exchange and enables AI-powered features such as speech recognition and task categorization, ensuring responsive user experiences.

Database Integration

Google Firebase manages BetterU's real-time database, user authentication, and cloud storage functionalities (Firebase, n.d.). Firebase's cloud-based infrastructure securely stores all user data and task information, enabling instant synchronization across multiple devices. This real-time database capability guarantees that users experience consistent, up-to-date information regardless of device.

Web Accessibility for Backend Services

While BetterU is primarily designed as a mobile Android application, its backend APIs can be deployed on free cloud hosting platforms supporting Python applications, such as Heroku or PythonAnywhere. This makes the backend services accessible via web browsers like Google Chrome, simplifying testing, maintenance, and providing a potential pathway for future web-based versions or administrative interfaces.

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

The BetterU mobile application is designed to serve two primary user groups: students and office workers. These groups were chosen due to their shared challenges in maintaining productivity, managing complex schedules, coping with multitasking demands, and dealing with mental fatigue. By analyzing their demographic, psychographic, and behavioral patterns, it has been identified that BetterU addresses key issues they face by improving task management, mental clarity, and overall connection with personal goals and society.

Based on the findings from the literature review, several core technologies were selected for BetterU's system development. In terms of programming languages and frameworks, Dart and Flutter were chosen due to their strong cross-platform capabilities and developer-friendly features, including hot reload powered by Just-In-Time (JIT) compilation. These technologies allow rapid iteration and consistent deployment on Android devices. For backend and AI functionalities, Python was selected for its simplicity, versatility, and extensive library support.

The selected development tools and integrated development environments (IDEs) include Android Studio, Visual Studio Code, and GitHub Desktop, which collectively provide a robust environment for collaborative coding, version control, and cross-functional teamwork. To power BetterU's AI-driven features, a suite of open-source libraries was adopted. These include Vosk and Whisper for speech recognition, spaCy and dateparser for natural language processing and task extraction, and Scikit-learn and SetFit for intelligent task categorization and prediction. These tools enable advanced, user-centric capabilities while maintaining low implementation costs. For cloud services, Google Firebase was selected to handle real-time data storage, user authentication, and cloud synchronization. Its free-tier services are sufficient for early deployment and ensure real-time data updates across user devices. On the design side, Figma was chosen as the primary UI/UX tool due to its reusable components, collaborative editing features, and interactive prototyping support. This allows designers and developers to work in sync and iterate quickly based on feedback.

The feasibility study has concluded that BetterU is viable from technical, economic, operational, and schedule perspectives. The availability of development hardware, such as a Windows laptop with Visual Studio Code, and access to open-source tools and cloud services strongly support its technical and economic feasibility. Operational feasibility is also ensured through the integration of Flutter, FastAPI, and Firebase, enabling smooth backend–frontend communication and user data management. The project follows an incremental development model, which ensures structured progress, supports early testing and validation, and reduces implementation risks.

Chapter 3

Methodology and Requirements Analysis

Methodology and Requirements Analysis

This chapter outlines the foundational stages involved in the development of the BetterU mobile application, including project planning, user research, requirement analysis, and technical preparation. The aim of this chapter is to demonstrate how the project was strategically approached to ensure that the application is both practical and aligned with user needs. By adopting an appropriate development model, conducting relevant research, and selecting suitable tools and technologies, the project establishes a solid groundwork for building a responsive, intelligent, and user-centered wellness application.

2.5 Methodology

2.5.1 Selection of Development Methodology

The BetterU mobile application adopts the **Incremental Development Model** as the foundation for its project development workflow. This model was selected primarily because it supports staged development and continuous improvement, which aligns well with the modular architecture and complex feature set of the BetterU system. The application is divided into several key modules, including Goal Assistance (shared module), Emotion Diary Note, Focus Timer, User, and Report modules.

To avoid overwhelming the development process by building all modules at once, the incremental model allows the system to be **divided into multiple functional builds**, each targeting specific module functionalities. Unlike the traditional Waterfall model, which requires all requirements to be finalized before development begins, the incremental approach enables **progressive implementation**. This method allows each module to be individually developed, tested, refined, and validated before proceeding to the next stage, reducing risk and complexity across the project.

Each increment follows its own cycle of planning, development, testing, and review, focusing on selected modules such as Goal Assistance and Anonymous Community. Through continuous feedback and integration, the development team can **identify issues early and implement necessary improvements in a timely manner**. This ensures a smooth flow for enhancing existing features and integrating new ones into the system.

By breaking the application into incremental builds, the development team can prioritize modules based on importance. Core modules, which are foundational and interlinked with other features, are developed in the early stages. Supportive modules are then integrated in later stages, either to extend the capabilities of the core modules or to enhance user experience.

Planned Functional Builds:

- 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

This structured approach ensures steady progress, reduces overall project risk, and enhances the flexibility and adaptability of the BetterU mobile application throughout its development lifecycle.

2.5.2 Application of Incremental Model in BetterU Development

Requirement Analysis Phase

The requirement analysis phase serves as the foundation for the entire development process of the BetterU mobile application. At the beginning of this phase, the development team **identifies real-world problems related to task management, productivity, and mental well-being.** This is accomplished through online research and observation to gain accurate and relevant insights. Based on these findings, the team proposes effective solutions that directly address the identified challenges.

In parallel, the development team determines the primary target users of the system by analyzing their backgrounds, behaviors, and daily routines. This user profiling helps consolidate the development direction of the application. Furthermore, a **comparative evaluation is conducted between the proposed BetterU mobile application and existing platforms such as Notion, Microsoft To Do, Google Keep, and Confluence**. This analysis supports the identification of unique strengths, weaknesses, and opportunities for improvement in the proposed solution.

Following the research on user needs, real-world issues, and competing applications, the team moves on to define the system and user requirements. This step ensures that the final product is capable of delivering services that meet the users' expectations, while also adhering to quality standards and development constraints such as time and budget.

Prior to collecting requirements, the team selects appropriate fact-finding techniques to ensure efficient and accurate information gathering. Methods such as **online research and user observation are employed** to support this process. Once the requirements are collected, they are translated into technical documentation. This includes Use Case Diagrams, Use Case Descriptions, and comprehensive listings of both Functional and Non-Functional Requirements. These deliverables help convert user needs into clear technical guidelines, providing a structured reference for the development team and minimizing the risk of ambiguity during later stages of the project.

System Design Phase

The System Design Phase is essential for translating the requirements gathered during the analysis stage into a detailed and organized blueprint that guides the implementation of the BetterU mobile application. Prior to initiating module development, various system design components are developed to ensure a clear understanding of application workflows, module interactions, and technical architecture.

The key design elements include the Software Architecture Diagram, Process Design, Sequence Diagrams, State Chart Diagrams, User Interface Design, Data Design, and Reports Design. These elements collectively define the technical and functional structure of the system, providing a solid foundation for efficient development and integration.

Software Architecture Diagram

The software architecture defines the high-level structural framework of the BetterU mobile application, outlining how core components interact with one another. BetterU adopts a client-server architecture that separates the frontend interface from backend processing and data storage.

The frontend is developed using Flutter, enabling cross-platform mobile deployment. The backend is powered by FastAPI using Python, responsible for handling business logic, processing API requests, and executing AI-related tasks such as task extraction and classification. Google Firebase serves as the primary backend database, offering real-time synchronization and user authentication. The frontend communicates with the backend through API calls, allowing Python logic to access data stored in Firebase.

Process Design

Process design defines the internal workflows and logical operations for each core module in BetterU. It outlines how data and control signals flow through the application in response to user actions and system events. Each key feature is represented by an activity diagram, providing a visual map of sequential steps, decisions, and conditions.

These diagrams are created using Draw.io and assist developers in understanding and implementing features according to well-defined logic. The following features are modeled with dedicated activity diagrams:

- Goal Assistance Module
    - Task entry and intelligent task extraction
    - Smart reminder generation
    - Task status updates (e.g., in progress, completed)
- Focus Timer Module
    - Start and stop session
    - Session categorization
    - Timer completion feedback
- Emotion Diary Note Module
    - Daily mood logging

    - Sentiment tagging
    - Note editing and deletion
- User Module
    - Account creation and login
    - Profile editing and preference setting
- Report Module
    - Generation of daily, weekly, and custom reports
    - Data visualization and insights display

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.

Sequence diagrams illustrate the interaction between various system components over time. These diagrams detail how the frontend (Flutter), backend (FastAPI), and Firebase database collaborate to complete user requests and system processes.

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
- Diary note submission
    - Entry saved → stored in Firebase → available for report
- Report generation
    - User opens report → system queries tasks logs → data visualized

State Chart Diagram Design

State chart diagrams are used to model the behavior of key objects as they transition through various states due to user interactions or system triggers. These diagrams help ensure reliable state management and consistent logic across the application.

Examples of key state transitions include:

- Task (Goal Assistance Module)
    - Pending → In Progress → Completed → Overdue
- Reminder (Goal Assistance Module)
    - Scheduled → Triggered → Snoozed
- Focus Session (Focus Timer Module)
    - Not Started → Active → Paused → Completed
- Diary Entry (Emotion Diary Note Module)
    - Draft → Created → Edited → Deleted

User Interface Design

User interface design is a critical aspect of BetterU, as it directly affects usability and user satisfaction. The design process begins in Figma, where all screens are prototyped before development in Flutter. Interfaces are crafted to be clean, responsive, and accessible, ensuring seamless navigation for students and working professionals.

Each module features consistent design elements and screen layouts, including:

- Goal Assistance Module: Task creation screen, smart suggestions, task list, and task editing interface
- Focus Timer Module: Timer dashboard, session summaries, category filters
- Emotion Diary Note Module: Mood selection interface, diary editor, history view
- User Module: Login screen, profile settings, onboarding steps
- Report Module: Graphical dashboards, productivity summaries, suggestion panels

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
- **Emotion Diary Notes**
    - Attributes: diary_id (PK), member_id (FK), emotion_tag_id (FK), title, content, mood_level, context_tags, sentiment_score, created_date, is_momentary, is_protected, status
    - Associated with the Member and EmotionTag by member_id and emotion_tag_id
- **Focus Sessions**
    - Attributes: focus_session_id (PK), member_id (FK), task_id (FK), subtask_id (FK), nfc_tag_id (FK), start_time, end_time, duration, nfc_triggered, interruption_count, status, reward_earned
    - Associated with Member, Task, Subtask and NFCTag by member_id, task_id, subtask_id and nfc_tag_id
- **Person**
    - Attributes: person_id (PK), username, email, password, registered_date, status
    - Inherited by Member (member_id (PK), person_id (FK), anonymous_name, occupation, theme, busy_start_time and others) and Admin entity (admin_id (PK), person_id (FK), last_login_datetime, created_by_id)
- **Rewards**
    - Attributes: reward_id (PK), name, type, description, icon_url, points_required, created_datetime, status

Reports Design

The reports design is centered on generating visual summaries of user behavior and progress to facilitate self-reflection and promote well-being within the BetterU mobile application. Reports are generated using data collected from Google Firebase, aggregated from core modules including the Goal Assistance Module, Emotion Diary Note, and Focus Timer.

To ensure consistency and clarity across all reports, each report will include the following elements:

- A clear and descriptive title
- The report period or issue date
- Filter options that allow users to view reports by month, custom date range, or category
- A summary section presenting key statistics such as total tasks completed and time spent on activities
- Detailed records comprising task lists (categorized by completed, overdue, or in progress) and a breakdown of time usage
- Graphical representations such as bar charts, line graphs, and pie charts to illustrate trends and correlations

Additionally, three distinct types of reports will be produced:

- **Summary Reports:** These provide a concise overview of essential metrics, filtered by date or month, offering users a quick snapshot of their performance
- **Detailed Reports:** These contain comprehensive listings of entries, including individual tasks and time usage histories, enabling in-depth review
- **Exception Reports:** These highlight significant insights derived from user data, such as missed deadlines, disproportionate time allocation to specific tasks or activities, and indicators of low social engagement

This structured reporting approach empowers users to better understand their productivity patterns and emotional well-being, supporting continuous improvement and informed decision-making.

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.

2.6 Requirements Gathering Techniques

2.6.1 Observation

In order to better understand and explore real challenges faced by potential users, informal observations were conducted within my daily environment. These observations focused on individuals such as close friends, cousins, relatives' children, and myself. By doing so, relevant functional and non-functional requirements were identified and incorporated into the BetterU mobile application. This approach helps ensure that the application is capable of providing practical and effective solutions to the problems faced by its target users.

Poor Task Arrangement and Time Mismanagement

Through personal observation, I noticed that many of my friends often **struggle to arrange their tasks effectively**. This frequently leads to missed deadlines or a decline in the quality of their work. Even when they recognize the need for proper planning, their time management tends to be weak. Tasks that require minimal effort are sometimes given excessive attention, while more demanding tasks are rushed or postponed. This imbalance often leads to procrastination and overwhelming pressure near deadlines.

Trouble with Task Recording Habits

Some of my cousins and relatives' children often **lack consistent habits in recording their tasks**, whether academic or personal. They perceive the process of organizing and managing tasks as tedious. Instead of using structured tools or applications, they tend to rely on random paper notes or disorganized note-taking apps on their phones. Over time, these notes are either misplaced or become unreadable, **resulting in confusion and forgotten responsibilities**.

Distraction from Mobile Apps

Another issue I frequently observe among my friends is their tendency to **get distracted by entertainment apps while trying to focus on important tasks**. Without external reminders or supervision, they easily drift into endless scrolling on social media or mobile games. This significantly hampers their productivity and wastes valuable time. **As deadlines approach, the accumulated delays contribute to increased stress levels**.

Lack of Time for Social Life

One of my cousins, who works full-time, struggles to maintain a healthy work-life balance. He often leaves early in the morning and returns home late at night. After dinner, he usually spends a short time on social media before going to bed. He has shared with me how **difficult** it is **to keep in touch with his old friends or to find time for new meaningful social interactions**. This ongoing lack of connection can eventually have a negative impact on mental well-being.

Fear of Reaching Out for Help

I also observed that some of my friends and cousins tend to avoid seeking help even when they encounter problems, especially in academic or work environments. Being more reserved, they **feel uncomfortable reaching out in group discussions or asking colleagues and peers for assistance**. They are often **worried about being judged or misunderstood**. This reluctance causes delays in resolving issues and leads to increased emotional pressure.

Emotional Instability

As for myself, balancing academic responsibilities, part-time work, and personal life often results in emotional instability. There are times when I feel **overwhelmed by the tight schedules and find it difficult to manage my time efficiently**. The lack of structured planning and emotional support can make it **challenging to stay focused, motivated, and mentally well**. This personal struggle further emphasizes the importance of having a mobile application like BetterU to support emotional management and task organization.

2.6.2 Online Research

To comprehensively explore the pain points and expectations of users when engaging with task management tools, a detailed online research study was carried out. This research involved gathering insights from user-generated reviews, blog discussions, productivity forums, and academic articles. Through this method, BetterU aims to uncover the real-world frustrations, preferences, and needs of potential users in diverse environments such as student life, freelancing, and personal productivity. The findings derived from this research help inform the system design by offering a user-centric foundation for functional and non-functional requirement development.

1. Most Task Management Apps Are Designed for Teams, Not Individuals

One prominent observation from online reviews and comparative studies is that most widely used task management tools, including Trello, Asana, and Notion, are primarily created with team collaboration in mind rather than for individual users. These platforms often offer a wide range of functionalities such as Gantt charts, collaborative boards, task assignment features, and shared team spaces. While these capabilities are beneficial in corporate or group project contexts, they may be excessive and even counterproductive for individuals who simply want to manage their own daily routines.

According to feedback from solo users and students, these tools tend to be cluttered with irrelevant functions such as team timelines, permissions, and workflow approvals, which complicate the user experience. Users have voiced that for personal use, they often find it difficult to navigate through these features or customize the app to a more simplified, private workspace (Amplework, 2024; Bateman, 2024; Lindy.ai, 2024). As a result, many users expressed a desire for tools that are streamlined, intuitive, and better suited for individual task planning.

2. Excessive Manual Effort Reduces Efficiency and Motivation

Another recurring theme identified from the research is the extensive amount of manual input required by many existing task management applications. Most apps require users to create tasks manually, assign due dates, update progress, set reminders, categorize items, and periodically reorganize their boards or lists. This constant administrative overhead not only demands time but also interrupts the user's cognitive flow, leading to decreased motivation and, in some cases, app abandonment.

Empirical studies and reports suggest that a significant portion of professionals' weekly working hours is consumed by repetitive manual tasks that could be automated using intelligent systems. According to Smartsheet (2023) and ProcessMaker (2024), employees spend up to 25 percent of their week on manual digital work that lacks automation. Furthermore, the inability of apps to learn user behavior or recommend intelligent actions compounds the issue, as users must continually perform the same repetitive steps. This inefficiency contradicts the core purpose of productivity tools and highlights a gap where AI-enhanced automation could add substantial value (TeamDynamix, 2024).

3. Difficulty in Task Prioritization and Cognitive Overload

A third critical issue users face relates to decision-making fatigue and poor task prioritization support. Although many task management applications allow users to enter tasks and

deadlines, they generally do not provide intelligent guidance on how to prioritize activities or manage workloads based on urgency, time constraints, or personal energy levels. As a result, users are left to make complex decisions without system support, which can quickly become overwhelming.

Research by Nielsen Norman Group (2024) and IJSRD (2024) points out that users frequently encounter decision fatigue when required to manually assign importance to every task without contextual suggestions. This is especially problematic for students or busy professionals managing multiple responsibilities. On platforms like Reddit (2024), users reported that while their tools provided notification systems, they lacked mechanisms for real scheduling assistance or time allocation planning. Thus, instead of functioning as productivity assistants, these tools often end up being static task repositories.

This issue is further compounded by the psychological burden associated with too many simultaneous choices, a phenomenon supported by cognitive psychology literature. Without adequate visual or algorithmic support to break down large to-do lists, users experience mental overload, reduced clarity, and often disengage from the tools altogether.

4. Emotional Discomfort and Lack of Anonymous Support Spaces

Beyond task execution features, emotional well-being emerged as an often-overlooked but essential aspect of digital productivity tool usage. During the online research, it became evident that users frequently deal with emotional challenges such as stress, guilt from unfinished tasks, or burnout. However, existing platforms generally lack built-in support mechanisms to address these issues in a private and safe manner.

Many users are hesitant to share their struggles or seek advice in online communities due to the lack of anonymity. Platforms like Reddit or productivity forums usually require user accounts and track digital footprints, which may discourage users from opening up about their psychological difficulties. According to studies conducted by Pew Research Center (2013) and Kiesler et al. (2013), individuals feel more secure expressing vulnerability when anonymity is guaranteed.

Concerns about data privacy in mental health and productivity-related apps further restrict users from using available emotional support features. The Journal of Medical Internet Research (2025) highlights that several mental health and wellness apps have weak data protection measures, leading users to mistrust their platforms. This highlights a vital requirement for BetterU to integrate anonymous communication spaces or journaling features that offer privacy and psychological safety.

2.6.3 Summary of Fact Gathering Results

The fact-finding phase of the BetterU project included two primary research methods: informal personal observation and structured online research. Together, these approaches provided a multifaceted understanding of the real-world challenges faced by potential users, especially students, young professionals, and individuals managing multiple responsibilities in their daily lives. The findings reveal a consistent pattern of difficulties across several key areas of task management and personal well-being.

1. Users Lack Effective Task Planning and Prioritization Strategies

Observational data revealed that many individuals, particularly students and young adults, struggle to plan their tasks efficiently. They often misjudge the time and effort required for different activities, resulting in procrastination, last-minute stress, and reduced output quality. This issue is further exacerbated by users' inability to prioritize tasks meaningfully based on urgency, complexity, or personal energy levels. Existing task management tools do not provide intelligent support for task sorting or workload balancing, thereby contributing to decision fatigue and cognitive overload.

2. Manual Workload and Lack of Automation Discourage Long-Term Use

Both personal observations and online reviews pointed to the burden of manual input as a major barrier to user engagement. Many users expressed frustration with having to constantly create, update, and reorganize their task entries. This administrative effort consumes time and interrupts the natural flow of work. Moreover, most apps do not adapt to user behavior or offer automated suggestions, which leads to repetitive actions and lower motivation over time. The absence of AI-assisted automation in existing systems is a significant gap that BetterU aims to address.

3. Inconsistent Task Logging Habits and Tool Aversion

Several individuals, especially among younger users such as students and teenagers, exhibit inconsistent or disorganized methods of recording their tasks. Rather than using structured platforms, they rely on fragmented tools like paper notes or default mobile apps, which are often lost or forgotten. Many users perceive task logging as tedious and unnecessary, reflecting a broader aversion to conventional productivity systems. This behavior suggests a need for more engaging, user-friendly interfaces that reduce friction in task entry and tracking.

4. Distraction and Lack of Focus Management Tools

A recurring issue observed among users is their susceptibility to digital distractions, particularly from mobile entertainment apps and social media. Without built-in accountability features, users frequently drift away from important tasks, leading to time mismanagement

and elevated stress near deadlines. Current tools generally do not include mechanisms for managing distractions, such as focus timers, usage blockers, or motivational nudges. This missing functionality limits their effectiveness in supporting sustained attention and discipline.

5. Limited Support for Work-Life Balance and Social Interaction

Many users, especially working professionals, struggle to maintain a healthy work-life balance due to long work hours and tightly packed schedules. There is often little time left for meaningful social interaction or personal reflection, which contributes to emotional fatigue and social isolation. Most task management tools focus narrowly on productivity metrics and fail to incorporate features that support overall well-being, such as time-blocking for personal activities or reminders to reconnect with friends.

6. Emotional Challenges and Barriers to Seeking Help

Users commonly face emotional discomfort related to workload pressure, failure to complete tasks, or fear of judgment when seeking help. Personal observations indicated that many individuals avoid asking for assistance, especially in academic or team settings, due to anxiety or social inhibition. Online research echoed this sentiment, with users expressing concern about lack of privacy in forums and mental health apps. The absence of anonymous and empathetic support channels further discourages users from addressing emotional difficulties openly.

Together, these findings underscore the importance of designing a mobile application that not only organizes tasks efficiently but also supports users holistically. BetterU should prioritize features that offer intelligent task automation, reduce manual workload, encourage consistent task logging, manage distractions, promote social well-being, and provide private emotional support. The insights gathered through observation and research form the foundation for defining the functional and non-functional requirements of the application, ensuring alignment with user needs and expectations.

2.7 Requirements Analysis

2.7.1 Use Case (UC) Diagram and Description

Use Case Diagram

Figure 3.1.1: Use Case Diagram - Goal Assistant Module (shared)

Figure 3.1.2: Use Case Diagram - Emotion Diary Note Module

Figure 3.1.3: Use Case Diagram - Focus Timer Module

Figure 3.1.4: Use Case Diagram - User Module

Figure 3.1.5: Use Case Diagram - Report Module

Use Case Description

Goal Assistance Module (shared) :

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 3.1.1: Use Case Description - Create Task of Goal Assistance Module

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 3.1.2: Use Case Description - Update Task of Goal Assistance Module

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 3.1.3: Use Case Description - Delete Task of Goal Assistance Module

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 3.1.4: Use Case Description - Receive Goal Notifications of Goal Assistance Module

Name of Use Case: Manage Smart Suggestion	
Brief Description: When a user creates or updates a task, the system will evaluate it. If needed, the system will provide a smart suggestion to improve scheduling or 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. The system displays a pop-up at the bottom of the task list view.
5. The user taps on "Apply" on the suggestion pop-out.	6. The system performs the modification and collapses the suggestions.
	7. 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 below the task list. The task is updated accordingly if applied.

Table 3.1.5: Use Case Description - Manage Smart Suggestion of Goal Assistance Module

Emotion Diary Note Module:

<table>
<tr><td colspan="2">Name of Use Case: Manage Diary Entry</td></tr>
<tr><td colspan="2">Brief Description: The user can manage their emotion diary by creating, viewing, updating, deleting, and locking/unlocking entries. The system validates inputs and updates the diary records accordingly.</td></tr>
<tr><td colspan="2">Actor: User</td></tr>
<tr><td colspan="2">Precondition: The user must be logged into the system and the role is member.</td></tr>
<tr><th>Actor Action</th><th>System Response</th></tr>
<tr><td>1. The user selects the “Emotion Diary” module.</td><td>2. The system displays the Emotion Diary dashboard.</td></tr>
<tr><td>3. The user chooses to create a new entry.</td><td>4. The system opens the diary input form.</td></tr>
<tr><td>5. The user writes thoughts and selects a mood tag.</td><td></td></tr>
<tr><td>6. The user submits the diary entry.</td><td>7. The system validates and saves the entry.</td></tr>
<tr><td>8. The user selects a specific date of the past diary in the calendar view.</td><td>9. The system displays the specific past diary overview.</td></tr>
<tr><td>10. The user uses search or filter options.</td><td>11. The system filters entries accordingly.</td></tr>
<tr><td>12. The user selects an entry to update or delete.</td><td>13. The system opens the entry for modification or deletion.</td></tr>
<tr><td>14. The user locks or unlocks an entry.</td><td>15. The system updates the lock status and requests biometric if enabled.</td></tr>
<tr><td colspan="2">Alternative Flow:

A1. Step 5: If the user submits the form without entering content, the system displays: "Diary content cannot be empty."

A3. Step 15: If the user attempts to unlock a locked entry without biometric authentication, the system shows: "Biometric verification failed. Try again."</td></tr>
<tr><td colspan="2">Postcondition: The diary entry is created, updated, deleted, or locked/unlocked as requested and stored in the database.</td></tr>
</table>

Table 3.1.6: Use Case Description - Manage Diary Entry of Emotion Diary Note Module

Name of Use Case: Generate Emotional Trend Report	
Brief Description: The user views emotional patterns through visual graphs generated from existing diary data.	
Actor: User	
Precondition: The user must have at least one existing diary entry.	
Actor Action	**System Response**
1. The user selects “Generate Emotional Trend Report”.	2. The system displays emotion heatmaps, line charts, and emotion frequencies.
Alternative Flow: A1. Step 2: If the system finds no entries, it displays: "You need at least one diary entry to generate a report." A2. Step 2: If the graph fails to load due to connectivity issues, the system displays: "Unable to load graph. Please check your connection."	
Postcondition: A visual trend report is successfully displayed to the user.	

Table 3.1.7: Use Case Description - Generate Emotional Trend Report of Emotion Diary Note Module

Focus Timer Module:

Name of Use Case: Manage Focus Timer Session	
Brief Description: The user can manage a focus timer session by starting, pausing, resuming, stopping, or resetting it. The system tracks the session time, notifies about breaks, and saves focus duration data.	
Actor: User	
Precondition: The user must be logged into the system and the role is member.	
Actor Action	**System Response**
1. The user navigates to the “Focus Timer” module.	2. The system displays the focus timer interface.
3. The user clicks the “Start” button.	4. The system starts the timer countdown and logs the session.
5. The user clicks “Pause” during the session.	6. The system pauses the countdown.
7. The user clicks “Resume”.	8. The system resumes the timer from the paused time.

9. The user clicks “Stop” or “Reset”.	10. The system stops the session, clears or reset the timer.
11. The user finished its focus timer session.	12. The system records the session duration and updates the session history.
Alternative Flow: A1. Step 4: If the session duration set is less than 5 minutes, the system displays: “Minimum session must be 5 minutes.” A2. Step 5: If the user attempts to pause a timer that hasn't started, the system displays: “You must start a session before pausing.” A4. Step 11: If the network is unavailable, the system stores session data locally and syncs it once reconnected.	
Postcondition: The focus timer session is successfully recorded, and break notifications are handled. The session data is available for reporting.	

Table 3.1.8: Use Case Description - Manage Focus Timer Session of Focus Timer Module

Name of Use Case: Monitor App Usage	
Brief Description: The user can receive distraction alerts during focus time, manage a whitelist of allowed apps, and view leaderboard stats related to focus performance.	
Actor: User	
Precondition: The user must be logged in and have a session in progress.	
Actor Action	**System Response**
1. The user begins a focus session.	2. The system monitors app usage in the background.
3. The user opens a distracting app.	4. The system displays a notification: “You are getting distracted.”
5. The user adds an app to the whitelist.	6. The system adds the selected app to the allowed list.
7. The user selects “Leaderboard”.	8. The system shows focus duration rankings among users.
Alternative Flow: A1. Step 4: If the user disables notification permission, the system cannot alert and logs: “Distraction alert blocked by device settings.” A2. Step 5: If the app cannot be added to the whitelist (e.g., invalid app ID), the system shows: “This app cannot be whitelisted.”	

A3. Step 8: If leaderboard data fails to load, the system displays: “Unable to retrieve leaderboard. Please check your connection.”
Postcondition: App usage is monitored and alerts/leaderboard/whitelist functions operate as expected.

Table 3.1.9: Use Case Description - Monitor App Usage of Focus Timer Module

Name of Use Case: Generate Focus Activity Summary	
Brief Description: The user can view a heatmap of their focus activity, including session frequency and duration by time of day.	
Actor: User	
Precondition: The user must be logged into the system and the role is member.	
Actor Action	**System Response**
1. The user selects “Focus Activity Summary”.	2. The system processes focus data and generates a heatmap with highlights high and low productivity periods using color intensity.
Alternative Flow: A1. Step 2: If the user has no focus session data, the system shows: “No focus activity found. Start a session to generate a summary.” A2. Step 2: If the heatmap fails to load due to data corruption or rendering error, the system displays: “Unable to display heatmap. Please try again later.”	
Postcondition: The user receives a visual summary of their productivity patterns.	

Table 3.1.10: Use Case Description - Generate Focus Activity Summary of Focus Timer Module

Name of Use Case: Collect Rewards	
Brief Description: After completing focus sessions, the user may earn points and access the shop to redeem rewards.	
Actor: User	
Precondition: The user must have completed at least one focus session.	
Actor Action	**System Response**
1. The user selects “Collect Rewards”.	2. The system checks earned focus points.

3. The user accesses the reward shop.	4. The system displays available items for redemption.
5. The user selects a reward and confirms.	6. The system deducts points and completes redemption.
Alternative Flow: A1. Step 4: If the user has no points, the system displays: “You don’t have enough points to redeem rewards.”	
Postcondition: Rewards are redeemed successfully, and points are updated.	

Table 3.1.11: Use Case Description - Collect Rewards of Focus Timer Module

User Module:

Name of Use Case: Register Account	
Brief Description: The user registers an account by providing personal credentials. The system validates the inputs and creates a new account.	
Actor: User	
Precondition: The user must not be logged in or already registered.	
Actor Action	**System Response**
1. The user selects “Register”.	2. The system displays the registration form.
3. The user enters email, username, and password.	4. The system validates the email format and username availability.
5. The user submits the form.	6. The system creates a new account and redirects to the login screen.
Alternative Flow: A1. Step 4: If the email format is invalid, the system shows: “Invalid email format.” A2. Step 4: If the username is already taken, the system displays: “Username not available.” A3. Step 6: If account creation fails due to a network issue, the system displays: “Registration failed. Please try again later.”	
Postcondition: A new user account is successfully created and ready for login.	

Table 3.1.12: Use Case Description - Register Account of User Module

Name of Use Case: Login Account

Brief Description: The user logs into the system using valid credentials.	
Actor: User	
Precondition: The user must have a registered account.	
Actor Action	**System Response**
1. The user selects “Login”.	2. The system displays the login form.
3. The user enters the email and password.	4. The system validates the credentials.
5. The user submits the form..	6. The system navigates the user to the home page.
Alternative Flow: A1. Step 4: If credentials are incorrect, the system shows: “Invalid email or password.” A2. Step 6: If the system cannot connect to the server, it shows: “Login failed. Please check your internet connection.”	
Postcondition: The user is logged in and redirected to the main interface.	

Table 3.1.13: Use Case Description - Login Account of User Module

Name of Use Case: Reset or Recover Password	
Brief Description: The user requests to reset a forgotten password. A verification code is sent via email.	
Actor: User	
Precondition: The user must provide a valid registered email address.	
Actor Action	**System Response**
1. The user selects “Forgot Password”.	2. The system prompts for email input
3. The user enters their email.	4. The system sends an OTP to the user’s email.
5. The user enters the OTP and sets a new password.	6. The system validates and updates the password.
Alternative Flow: A1. Step 3: If the email is not registered, the system shows: “No account found with this email.” A2. Step 4: If email sending fails, the system shows: “Failed to send OTP. Try again.” A3. Step 5: If the OTP is incorrect, the system shows: “Invalid verification code.”	

Postcondition: The user's password is successfully updated.

Table 3.1.13: Use Case Description - Reset or Recover Password of User Module

Name of Use Case: Update Profile	
Brief Description: The user must be logged in.	
Actor: User	
Precondition: The user must not be logged in or already registered.	
Actor Action	**System Response**
1. The user navigates to Profile settings.	2. The system displays the current profile information.
3. The user selects Edit Profile.	4. The system displays editable fields for profile details.
5. The user updates the desired information (e.g., name, email, phone, password, profile picture).	6. The system validates the input formats and checks for conflicts (e.g., duplicate email/username).
7. The user saves the changes.	8. The system updates the profile information in the database.
	9. The system confirms the update with a success message and displays the updated profile.
Alternative Flow: A1. Step 3: If the email/phone format is invalid, the system shows: "Invalid format." A2. Step 3: If the new password is weak, the system shows: "Password does not meet complexity requirements." A3. Step 3: If the uploaded profile picture is in an unsupported format, the system shows: "Invalid file format." A4. Step 4: If the update fails due to a system error, the system shows: "Update failed. Please try again."	
Postcondition: The user's profile is updated.	

Table 3.1.14: Use Case Description - Update Profile of User Module

Name of Use Case: Delete Account
Brief Description: The user can permanently delete their account after confirmation.

Actor: User	
Precondition: The user must be logged in and confirm account deletion.	
Actor Action	**System Response**
1. The user selects “Delete Account” from settings.	2. The system prompts a confirmation dialog.
3. The user confirms the deletion.	4. The system deletes the account and logs the user out.
Alternative Flow: A1. Step 3: If the user cancels, the system returns to the settings page without deleting the account. A2. Step 4: If deletion fails due to server or connection issues, the system shows: “Unable to delete account. Try again later.”	
Postcondition: The account is permanently removed, and the user is redirected to the welcome screen.	

Table 3.1.15: Use Case Description - Delete Account of User Module

Name of Use Case: Manage Notification Preferences	
Brief Description: The user can manage push notifications related to reminders, updates, and goals.	
Actor: User	
Precondition: The user must not be logged in or already registered.	
Actor Action	**System Response**
1. The user selects “Notification Preferences” from settings.	2. The system displays available notification categories.
3. The user toggles specific notification types.	4. The system updates preferences and confirms changes.
Alternative Flow: A1. Step 4: If saving preferences fails, the system shows: “Unable to update notification settings, Try again.”	
Postcondition: Notification preferences are successfully updated and saved.	

Table 3.1.16: Use Case Description - Manage Notification Preferences of User Module

Name of Use Case: Logout Account	
Brief Description: The user logs out of the system to end their current session.	
Actor: User	
Precondition: The user must be logged into the system.	
Actor Action	**System Response**
1. The user selects the “Log Out” option from the settings or main menu.	2. The system prompts for confirmation.
3. The user confirms log out.	4. The system ends the session, clears local session data, and redirects to the login or welcome screen.
Alternative Flow: A1. Step 2: If the user cancels the confirmation, the system returns to the previous screen and does not log out. A2. Step 4: If the log out process fails due to system error, the system shows: “Log out failed. Please try again.”	
Postcondition: The user is securely logged out, and all session data is cleared from the device.	

Table 3.1.17: Use Case Description - Logout Account of User Module

Report Module:

Name of Use Case: View Report	
Brief Description: The user can view three types of reports: Emotion Summary Report, Focus Performance Report, and Goal Tracking Report. Each report includes visual insights and can be exported.	
Actor: User	
Precondition: The user must be logged in and have relevant data (diary entries, focus sessions, or goals) available.	
Actor Action	**System Response**
1. The user selects the “Reports” option from the main menu.	2. The system displays available report types: Emotion, Focus, and Goal.
3. The user selects “Emotion Summary Report”.	4. The system generates emotion statistics and graphs.
5. The user clicks “Export Report”.	6. The system creates a downloadable

	file (PDF/CSV/Image).
7. The user selects “Focus Performance Report”.	8. The system shows charts summarizing total focus hours, session distribution, and improvement trends.
9. The user clicks “Export Report”.	10. The system exports the focus performance report.
11. The user selects “Goal Tracking Report”.	12. The system displays visual data on goal completion, progress, and categories.
13. The user clicks “Export Report”.	14. The system generates the downloadable goal report file.
Alternative Flow: A1. Step 4: If no emotion data exists, the system shows: “No emotion data available. Please write diary entries to view this report.” A2. Step 6: If export fails due to a system or network error, the system shows: “Failed to export report. Please try again later.” A3. Step 8: If no focus sessions have been recorded, the system displays: “No focus session data available.” A4. Step 10: If export fails due to file permission issues, the system displays: “Unable to save file. Please allow device storage permissions.” A5. Step 12: If the user hasn’t tracked any goals, the system shows: “No goal tracking data available.” A6. Step 14: If export succeeds but the user cancels download, the system logs: “Export canceled by user.”	
Postcondition: The user successfully views and/or exports reports related to their emotional state, focus performance, and goal achievement based on existing app data.	

Table 3.1.18: Use Case Description - View Report of Report Module

Name of Use Case: View Consolidated Progress Dashboard	
Brief Description: The user can view a unified dashboard showing a summary of emotion trends, focus patterns, and goal progress in one place.	
Actor: User	
Precondition: The user must be logged in and have at least one type of data (diary, focus, or goal) available.	
Actor Action	**System Response**

1. The user selects “Dashboard” from the report or home screen.	2. The system loads and displays a consolidated view of emotion, focus, and goal data.
3. The user scrolls or navigates through the dashboard sections.	4. The system presents charts, stats, and highlights for each module (e.g., top moods, total focus time, goals completed).
Alternative Flow: A1. Step 2: If the system fails to load data due to connectivity issues, it displays: “Failed to load dashboard. Please check your internet connection.” A2. Step 2: If no relevant data is found in all modules, the system displays: “No data available to generate dashboard. Start using the app to see progress here.” A3. Step 4: If data from one module (e.g., focus) is missing, that section will be grayed out with a message: “No focus data yet.”	
Postcondition: The user views an integrated overview of their personal progress across the app modules.	

Table 3.1.19: Use Case Description - View Report of Report Module

Name of Use Case: View Self-Reflection Prompts	
Brief Description: The user is presented with AI-generated or pre-written self-reflection prompts based on recent emotional patterns and app usage.	
Actor: User	
Precondition: The user must be logged into the system and have at least one recent diary entry or emotion record.	
Actor Action	**System Response**
1. The user selects “Self-Reflection Prompts” from the report or diary module.	2. The system analyzes recent emotion data and generates tailored reflection prompts.
3. The user reads through the prompts.	4. The system offers an option to respond or write a diary entry based on the prompt.
Alternative Flow: A1. Step 2: If no emotional data is found in the past week, the system shows: “No recent emotional patterns found. Write a diary entry to receive reflections.”	
Postcondition: The user receives meaningful self-reflection prompts and may optionally respond through a diary note.	

Table 3.1.20: Use Case Description - View Self-Reflection Prompts of Report Module

2.7.2 Functional Requirements

1.0 Emotion Diary Note Module

1.1 Diary Entry Management

1.1.1 The system shall allow users to create, edit, delete, and view diary entries.

1.1.2 The system shall support multiple entries "momentary" per day for quick reflections and future review.

1.1.3 The system shall support diary input via text, emojis, voice (speech-to-text), and image.

1.1.4 The system shall allow users to optionally attach images to diary entries.

1.1.5 The system shall automatically capture and display the creation date and time of each entry.

1.1.6 The system shall allow users to search entries by title, date, mood tags, or keyword highlights.

1.1.7 The system shall support sorting entries by creation date/time or mood.

1.1.8 The system shall support grouping entries by mood or by creation date range using a calendar view.

1.2 Sentiment Analysis & Mood Tagging

1.2.1 The system shall analyze diary content using AI-powered sentiment analysis (e.g., via Scikit-learn or NLP libraries).

1.2.2 The system shall automatically suggest mood tags based on detected emotional tone (e.g., Neutral, Happy, Sad, Anxious, Fearful, Angry, Mixed).

1.2.3 The system shall highlight keywords that indicate possible causes of stress, problems encountered, or sources of happiness.

1.2.4 The system shall allow users to manually edit mood tags and keyword highlights.

1.3 Privacy Control

1.3.1 The system shall offer optional PIN or biometric (e.g., fingerprint) protection for accessing or locking sensitive diary entries.

2.0 Focus Timer Module

2.1 Focus Session Control

2.1.1 The system shall allow users to select a task and specify a focus duration (15, 25, 45, 60 minutes, or custom).

2.1.2 The system shall support pause, cancel, resume, and reset controls during focus sessions.

2.1.3 The system shall support focus tracking tied to specific tasks for progress accumulation.

2.1.4 The system shall support phone flip detection to initiate or maintain focus mode.

2.1.5 The system shall display a visual heatmap (similar to GitHub contribution graph) to represent daily focus completion history.

2.1.6 The system shall allow users to view their session history, rewards earned, and personal progress trends.

2.2 Gamification

2.2.1 The system shall trigger start/end sound effects and visual animations during focus sessions.

2.2.2 The system shall reward users with virtual “treasures” upon completing a focus session, where both the probability of obtaining rare items and the total number of rewards shall scale with the session duration (e.g., longer sessions yield higher drop rates and multiple items).

2.2.3 The system shall allow users to collect virtual “treasures” and convert duplicate items into gems.

2.2.4 The system shall include a shop where users can exchange gems for cosmetic or premium features (e.g., limited-time avatar frames, wallpapers, feature trial days).

2.2.5 The system shall track and display daily focus streaks to encourage consistency.

2.3 Distraction Prevention

2.3.1 The system shall monitor app usage during focus sessions and pause the timer if apps outside the whitelist are accessed.

2.3.2 The system shall notify users with a warning if distraction rules are violated.

2.3.3 The system shall allow users to manage their app whitelist manually.

2.3.4 The system shall maintain a leaderboard showing top focus durations within the user's community (e.g., StudyTogetherCommunity) and globally (Overall Top 10).

3.0 User Module

3.1 User Registration & Authentication

3.1.1 The system shall allow new users to register by providing a username, password, email, gender, and occupation.

3.1.2 The system shall validate that the email is in the correct format and enforce a strong password policy.

3.1.3 The system shall send a One-Time Password (OTP) to the user’s email for verification during registration.

3.1.4 The system shall allow users to log in using either their username or email together with their password.

3.1.5 The system shall provide a “Forgot Password” function that sends an OTP to the user’s email for password recovery.

3.1.6 The system shall check for duplicate usernames and notify the user if the chosen username is already taken.

3.2 User Profile & Preferences

3.2.1 The system shall allow users to update their profile information, including profile image, avatar frame, username, gender, and occupation.

3.2.2 The system shall ensure that usernames remain unique when users update them.

3.2.3 The system shall assign user roles such as User, VIP, or Admin and restrict access to features based on these roles.

3.2.4 The system shall grant VIP users access to exclusive features, including unlimited AI functionalities.

3.2.5 The system shall allow users to select application themes, such as light mode or dark mode.

3.2.6 The system shall allow users to configure notification preferences, including the option to silence notifications.

3.2.7 The system shall allow users to enable or disable PIN or fingerprint protection for diary entries.

3.3 Administration

3.3.1 The system shall provide administrators with a dashboard that displays the total number of active users along with user retention metrics.

3.3.2 The system shall allow administrators to view and manage new requests for community group creation or joining.

3.3.3 The system shall enable administrators to review and moderate flagged posts, messages, and user accounts.

3.4 Session Control

3.4.1 The system shall manage secure login sessions using token-based authentication with automatic timeout and refresh mechanisms.

3.4.2 The system shall provide users with the option to log out manually and terminate all active sessions.

4.0 Report Module

4.1 Insight Generation

4.1.1 The system shall generate weekly, monthly, and yearly summaries that reflect the user's completed and pending tasks, focus sessions, and emotional records.

4.1.2 The system shall allow users to view and filter reports based on custom time ranges and selected data types, such as goals, focus duration, or mood trends.

4.1.3 The system shall provide a visual comparison between to-do tasks and completed tasks to help users evaluate their productivity.

4.2 Data Visualization

4.2.1 The system shall present data using various chart formats, including line graphs, pie charts, and bar charts.

4.2.2 The system shall visualize trends and correlations among task completion, focus time, and emotional states.

4.2.3 The system shall allow users to view progress and consistency through weekly, monthly, or yearly breakdowns in a calendar-style or timeline layout.

4.3 Notifications & Reflection

4.3.1 The system shall send users personalized summary insights via in-app notifications or email, if enabled.

4.3.2 The system shall provide motivational feedback, encouragement messages, or suggestions for habit improvement based on detected patterns and behavior gaps.

4.3.3 The system shall offer self-assessment prompts to encourage users to reflect on progress, stressors, and achievements.

4.3.4 The system shall allow users to export their reports and visual summaries as downloadable PDF files for personal tracking or external sharing.

5.0 Goal Assistance Module (shared)

5.1 Task Management

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

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

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

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

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

5.1.6 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).

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

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

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

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

5.2 Smart Input & Suggestions

5.2.1 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").

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

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

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

5.3 Smart Advisor

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

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

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

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

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

2.7.3 Non-Functional Requirements

Product

1.0 Availability

1.1 The system shall maintain 99.95% uptime, minimizing interruptions in daily use.

1.2 Scheduled maintenance shall only occur during non-peak hours and not exceed 2 hours per month.

1.3 In the event of unexpected Firebase service disruption, the app shall attempt automatic reconnection and restore functionality within 5 minutes.

2.0 Functional

2.1 The system shall ensure full feature accessibility and consistency across Android and iOS platforms.

2.2 All core modules (Goals Assistance, Emotion Diary, Focus Timer, Reports, and User Account) must function reliably on smartphones.

2.3 The system shall utilize Firebase services (e.g., Firestore, Authentication, Cloud Functions) for backend functionalities.

3.0 Usable

3.1 The app shall feature an intuitive and beginner-friendly UI, allowing users to navigate core features (e.g., setting goals, starting timers, writing emotion logs) without external help.

3.2 All UI components shall follow consistent design patterns as defined in the Figma design system.

3.3 The system shall support theme personalization for better user comfort and accessibility.

4.0 Reliable

4.1 The app shall support up to 10,000 concurrent users without noticeable performance issues using Firebase backend scalability.

4.2 The system shall prevent data loss by utilizing Firebase offline persistence and real-time synchronization to ensure local changes are synced when connectivity resumes.

4.3 Emotion logs, goal history, and focus sessions shall be backed up in real-time to cloud storage.

5.0 Flexible

5.1 The system shall support localization, including multi-language support, adjustable time/date formats, and culturally adapted content presentation.

5.2 The app shall allow easy updates of localized text through Firebase Remote Config or dedicated translation files.

Organization

6. The system shall be developed using the Dart programming language with the Flutter framework.
7. The system shall use Firebase as the primary backend solution, including Firestore (database), Firebase Auth, and Firebase Cloud Functions.
8. Version control shall be managed via Git and hosted on GitHub, with team collaboration and issue tracking enabled.
9. The UI/UX design shall be created and iterated using Figma.
10. AI-powered features (e.g., emotion analysis, goal suggestion) shall be implemented using Scikit-learn and Pandas via backend Python microservices.

External

4. The app shall integrate Google or Apple Sign-In for secure user authentication.
5. If in-app purchases or donations are introduced, payment processing shall be handled via Google Play and Apple App Store's native payment APIs.
6. All user data handling must comply with relevant privacy laws (e.g., GDPR, CCPA) and Google/Firebase security best practices.

2.8 Development Environment

2.8.1 Hardware and Device Specification

Development Machine

The development tasks, including frontend (Flutter), backend (Python microservices), AI feature development, and UI/UX design, are performed on a high-spec gaming laptop. This device is chosen for its robust CPU and GPU performance, which is essential for compiling, running emulators, debugging complex logic, and training lightweight machine learning models. It also supports virtual device creation for multi-device simulation and efficient parallel processing.

Aspect	Specification
Device Model	ROG Strix G614JV_G614JV
Processor	13th Gen Intel(R) Core(TM) i7-13650HX, 2.60 GHz (14-core CPU)
RAM	32 GB
Storage	1 TB

Graphics	(GPU 0): Intel(R) UHD Graphics (GPU 1): NVIDIA GeForce RTX 4060 Laptop GPU
Operating System	Windows 11 Home (64-bit)

Table 3.2.1: Specification of Development Machine

This machine is particularly effective for handling the resource-intensive tasks involved in cross-platform mobile app development, such as rendering complex animations in Flutter or executing AI model inference using TensorFlow or PyTorch in Python. It ensures development remains efficient even under multitasking or heavy computational workloads.

Testing Machine

To validate app functionality under real-world conditions, a physical Android device is used for testing. This enables comprehensive evaluation of performance, UI responsiveness, compatibility with device hardware (e.g., camera, NFC), and actual user experience. Real-device testing is crucial to ensuring the BetterU app meets quality expectations, especially in areas such as focus timer alerts, emotion input journaling via touch interface, and NFC-enabled interactions.

Aspect	**Specification**
Device Model	POCO F5
Processor	Qualcomm Snapdragon 7+ Gen 2
RAM	12 GB
Storage	256 GB
Operating System	Xiaomi HyperOS 1.0.18.0.UMRMIXM (Android 14)
NFC Support	Yes

Table 3.2.2: Specification of Testing Machine

This testing phone supports all critical mobile hardware features such as high-refresh-rate display (for UI smoothness), NFC (for smart tag interactions), and advanced multitasking, making it ideal for real-environment testing during the development lifecycle.

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 3.2.3: Specification of Virtual Machine

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 3.2.4: Specification of NFC Tags

2.8.2 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 3.2.5: Software Tools Specification of Android Studio

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 3.2.6: Software Tools Specification of Visual Studio Code

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 3.2.7: Software Tools Specification of Github Desktop

2.8.3 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 and Windows 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 3.2.8: Specification of Dart with Flutter and Python with FastAPI

2.8.4 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 3.2.9: Specification of Google Firebase

2.8.5 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.

2.8.6 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 3.2.10: Specification of Figma

2.8.7 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)

2.9 Chapter Summary and Evaluation

This chapter presented the planning, analysis, and technical preparation stages undertaken for the development of the *BetterU* mobile application. The **Incremental Development Model** was adopted as the software process model for this project. This model was chosen because it aligns well with the nature of *BetterU*, where individual features can be developed, tested, and refined incrementally. This approach supports development flexibility, enables early feedback collection from supervisors, and allows for continuous improvement throughout the development lifecycle.

To ensure the application addresses real user needs, two fact-finding techniques were applied: **observation** and **online research**. The observation method involved monitoring the daily behaviors and habits of people within the developer's surroundings, such as friends and relatives, to gain insights into their challenges in task planning and emotional self-management. In parallel, online research was conducted by reviewing credible digital sources to explore common issues related to personal productivity and emotional well-being. The primary objective of these techniques was to gather relevant information to ensure that *BetterU* is tailored to meet the practical needs of its target users, mainly students and working professionals.

In addition, the chapter outlined the **requirement analysis process**, which included the creation of a use case diagram, detailed use case descriptions, and comprehensive listings of functional and non-functional requirements. These elements serve as a foundation for defining the system's expected behavior and guiding future development in a systematic and traceable manner.

Finally, this chapter described the **development environment and tools** used in the project. The mobile application was developed using the Flutter framework and Dart language for front-end implementation, while the backend was supported by FastAPI. The project also utilized various supportive technologies, including Figma for UI/UX design, Firebase for cloud services, and SQLite for local data storage. Additionally, external libraries and APIs were integrated to enable advanced features such as **speech recognition**, **natural language processing (NLP)**, and **machine learning**

(ML). These tools collectively enhance the app's ability to understand user input and deliver intelligent, context-aware suggestions to support users in achieving personal growth.

Chapter 4

System Design

System Design

This chapter presents the detailed system design of the BetterU mobile application, specifying both the structural and behavioral aspects of the system. It outlines various design models and specifications that illustrate how the system operates and interacts with its users, including students, office workers, and administrators. Sequence diagrams, state charts, and activity diagrams will be used to describe the dynamic processes and user interactions. The user interface design will provide an overview of the application's visual layout, while the data design will include class diagrams, entity-relationship diagrams, and a data dictionary to define the logical structure and flow of information. The report design will describe the required components, data elements, and levels of detail for each report, such as the productivity report and the social performance report. In addition, the process design and software architecture design will demonstrate how the system's components are organized and deployed. Finally, the chapter will highlight the AI algorithms applied within the application, supported by sample training datasets, to enable intelligent features such as task categorization and rescheduling prediction.

2.10 Sequence Diagram

Goal Assistance Module (shared)

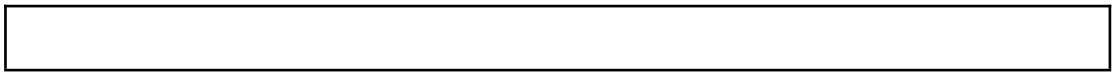


Diagram 4.1.1: Goal Assistance Module - Create Task

Diagram 4.1.2: Goal Assistance Module - Update Task

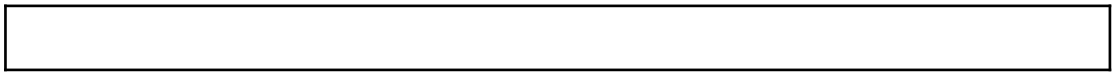


Diagram 4.1.3: Goal Assistance Module - Delete Task

Diagram 4.1.4: Goal Assistance Module - Receive Goal Notification

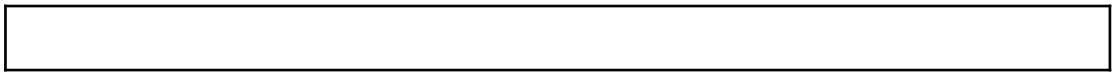


Diagram 4.1.5: Goal Assistance Module - Manage Smart Suggestions

Emotion Diary Note Module

Diagram 4.1.6: Emotion Diary Note Module - Manage Diary Entry

Diagram 4.1.7: Emotion Diary Note Module - Generate Emotional Trend Report

Focus Timer Module

Diagram 4.1.8: Focus Timer Module - Manage Focus Timer Session

Diagram 4.1.9: Focus Timer Module - Monitor App Usage

Diagram 4.1.10: Focus Timer Module - Collect Reward

Diagram 4.1.11: Focus Timer Module - Generate Focus Activity Summary

User Module

Diagram 4.1.12: User Module - Register Account

Diagram 4.1.13: User Module - Login Account

Diagram 4.1.14: User Module - Reset or Recover Password

Diagram 4.1.15: User Module - Update Profile

Diagram 4.1.16: User Module - Delete Account

Diagram 4.1.17: User Module - Manage Notification Preferences

Diagram 4.1.18: User Module - Logout Account

Report Module

Diagram 4.1.19: Report Module - View Report

Diagram 4.1.20: Report Module - View Consolidated Progress Dashboard

Diagram 4.1.21: Report Module - View Self-Reflection Prompts

State Chart Diagram

Diagram 4.2.1: State Chart Diagram - Person

Diagram 4.2.2: State Chart Diagram - EmotionDiary

Diagram 4.2.3: State Chart Diagram - Friendship

Diagram 4.2.4: State Chart Diagram - EmailVerification

Diagram 4.2.5: State Chart Diagram - FocusSession

Diagram 4.2.6: State Chart Diagram - Task

Diagram 4.2.7: State Chart Diagram - TaskAdvisorLog

User Interface Design

Home Page Design

Goal Assistance Module (shared)

	Goal Assistance Task List View Page The Task List View Page allows the users to manage and track their goals and tasks in a structured list format. This page displays the scheduled tasks for the selected date with labeled details such as priority level, task name, due date, category and progress status. Users can also easily trigger the sorting and filtering options to obtain their desired task items. When the users want to create new tasks, they can tap on the "New Task" button to navigate to the task creation page for filling in the new task information.
	Goal Assistance Calendar View Page The calendar view page of the goal assistance module can show a timeline-based visualization of user tasks. This allows them to see the task assignments distributed across days and weeks. The users can freely select the specific date that they want to check on by tapping on the specific date number within the calendar view. The users can also apply the sorting and filtering functions to retrieve their desired task items result. This calendar view is suitable for users to have a glance on the tasks within a certain time range.
	Goal Creation Page The goal creation page enables the users to create a new task or subtasks by filling up the task information. For example, the task title, priority level, type, category, occur

	date and time, description and even images. If the user wants to add subtasks on the main task, they can tap on the "Add Sub Task(s) as Goal Progress" button to navigate to the subtask creation page for entering the subtask information.
	Smart Advisor Page Smart Advisor Page can intelligently provide the task scheduling optimization strategies for helping the users to arrange their tasks in a better manner. It will display the suggestion overview item list in 4 sections which are Task, Time, Emotion and Other. In each suggestion item card, the suggestion source will be mentioned with showing the item description. Thus, the users can easily recognize which suggestions belong to which items. In order to respond to the suggestion item, the users can tap on the option icon at the top right corner of the suggestion item. Then, choose whether to accept or reject the suggestion. If the users want to turn the smart advisor on or off, they can also access this setting via tapping on the "Smart Advisor Visibility" option for toggling the visibility.

Emotion Diary Module

Emotion Diary Overview Page	Page recommends users to enable biometric for emotion diary.
Create Emotion Diary Page	Create Emotion Diary Page - Generate diary content by AI function page.

Focus Timer Module

Focus Timer Module Overview Page	Page that enables focus timer.
Showing window of ongoing focus timer.	Expanded ongoing focus timer.
Confirmation dialog for terminate focus timer session.	Popup message after finished focus timer session.
Redemption Page - user can use points earned to do redemption.	Leaderboard Ranking Page - show ranking of users focus session earned points.

User Module

Allowed users to select log in or register a new account.	Register Page.
Page to verify accounts by activation code sent to their registered email.	Log in Page.
Me Page that shows user information.	Edit profile page.
General user settings page.	Admin first page after log in successfully.
Page that admin can process new community requests from users.	
Page that admin can process new flags from users.	

Report Module

Data cards listing page.	Users can select different modules to view reports.	Users can view in table view or export reports.

2.11 Data Design

2.11.1 Class Diagram

Diagram 4.4.1: Class Diagram of BetterU Mobile Application

2.11.2 Entity Relationship Diagram

Diagram 4.4.2: Entity Relationship Diagram of BetterU Mobile Application

2.11.3 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.
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/forum posts.
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.

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.
member_id	String	UUID (36), Not Null	FK	-	References Member.member_id (the member whose email is being verified).
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.

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.
member_id	String	UUID (36), Nullable	FK	NULL	References Member.member_id. If NULL, the category is system pre-created; if not NULL, it is user-defined.
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.
is_recurring	Boolean	—	—	false	Whether the task is recurring.
recurring_pattern	String	Nullable (e.g., Daily, Weekly, Monthly)	—	NULL	Pattern definition for recurrence (if is_recurring = true).
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.
due_at	DateTime	Nullable	-	NULL	Explicit due date/time if different from planned_end_at.
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=Completed}	-	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

TaskSchedule

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

task_id	String	UUID (36), Not Null	FK	-	References Task.task_id. Defines the task linked to this schedule.
subtask_id	String	UUID (36), Nullable	FK	NULL	References Subtask.subtask_id. Optional, used if the schedule is for a subtask instead of the whole task.
scheduled_start	DateTime	Not Null	-	-	Planned start date/time of the scheduled task/subtask.
scheduled_end	DateTime	Not Null	-	-	Planned end date/time of the scheduled task/subtask.
estimated_duration	Integer	Minutes, >=0	-	0	Estimated duration (in minutes) allocated for this schedule.
actual_duration	Integer	Minutes, Nullable	-	NULL	Actual duration spent (in minutes), recorded after completion.
auto_scheduled	Boolean	-	-	false	Indicates whether the schedule was created automatically by the system or manually by the user.

Table 4.4.11: Data Dictionary - TaskSchedule 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.
type	Enum	{reminder, reschedule, priority_change, focus_tip, other}	-	-	Category or type of advisor suggestion.
suggestion_content	Text	-	-	-	The actual suggestion message or explanation provided to the user.
suggested_action	String	255	-	NULL	Specific recommended action (e.g., “reschedule to tomorrow 9am”, “increase priority”).
context_source	String	100	-	-	Origin of the suggestion (e.g., system, calendar_conflict, focus_history).
suggested_time	DateTime	Not Null	-	CURRENT_TIMESTAMP	Timestamp when the suggestion

					was generated.
user_response_at	DateTime	Nullable	-	NULL	Time when the user responded (accepted/ignored) to the suggestion.
status	Integer	Enum: {1=pending, 2=accepted, 3=ignored, 4=expired}	-	pending	Current state of 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

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 4.4.16: 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 4.4.17: 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.subt

					ask_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 4.4.18: Data Dictionary - NFCTag Entity

NFCLog

Field Name	Data Type	Size/Constraint	Key	Default	Description
nfc_log_id	String	UUID (36)	PK	Auto-generated	Unique identifier for each NFC scan log.
member_id	String	UUID (36), Not Null	FK	-	References Member.member_id. The member who scanned the tag.

nfc_tag_id	String	UUID (36), Not Null	FK	-	References NFCTag.nfc_tag_id. The tag that was scanned.
task_id	String	UUID (36), Nullable	FK	-	References Task.task_id. The task linked to the tag at the time of scan (if any).
subtask_id	String	UUID (36), Nullable	FK	-	References Subtask.subtask_id. The subtask linked to the tag at the time of scan (if any).
focus_session_id	String	UUID (36), Nullable	FK	-	References FocusSession.focus_session_id. If the scan triggered or was part of a focus session.
tag_purpose	String	50, Not Null	-	-	Purpose of the tag at scan time (e.g., Focus, TaskStart, TaskEnd, Reminder).
scan_at	DateTime	Not Null	-	Current timestamp	Timestamp of when the NFC tag was scanned.

Table 4.4.19: Data Dictionary - NFCLog Entity

Community

Field Name	Data Type	Size/Constraint	Key	Default	Description

community_id	String	UUID (36)	PK	Auto-generated	Unique identifier for 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

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 4.4.21: Data Dictionary - CommunityMember Entity

Tag

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

name	String	50, Not Null, Unique	-	-	Display name of the tag (e.g., “Wellness”, “Productivity”, “Study”).

Table 4.4.22: Data Dictionary - Tag Entity

PostTag

Field Name	Data Type	Size/Constraint	Key	Default	Description
post_id	String	UUID (36), Not Null	PK, FK	-	References Post.post_id. Identifies the post being tagged.
tag_id	String	UUID (36), Not Null	PK, FK	-	References Tag.tag_id. Identifies the tag assigned to the post.

Table 4.4.23: Data Dictionary - PostTag 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.

Table 4.4.24: 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 4.4.25: 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 4.4.26: 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 4.4.27: Data Dictionary - Comment Entity

Forum

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

description	Text	-	-	-	Description or purpose of the forum.
created_at	DateTime	Not Null	-	Current timestamp	Timestamp when the forum was created.
icon_url	String	255, Nullable	-	NULL	URL for the forum's icon or image.
status	Integer	Enum: {0 = inactive, 1 = active}, Not Null	-	1	Current status of the forum.

Table 4.4.28: Data Dictionary - Forum Entity

ForumMember

Field Name	Data Type	Size/Constraint	Key	Default	Description
forum_member_id	String	UUID (36)	PK	Auto-generated	Unique identifier for the forum membership record.
member_id	String	UUID (36), Not Null	FK	-	References Member.member_id. The member who joined the forum.
forum_id	String	UUID (36), Not Null	FK	-	References Forum.forum_id. The forum that the member belongs to.
role	Integer	Enum: {1=member, 2=creator}, Not Null	-	-	Role of the member within the forum.
subscription	Boolean	Not Null	-	true	Whether the member is subscribed to forum

					updates/notifications.

Table 4.4.29: Data Dictionary - ForumMember 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.
forum_id	String	UUID (36), Not Null	FK	-	References Forum.forum_id. The forum where the question is posted.
forum_member_id	String	UUID (36), Not Null	FK	-	References ForumMember.forum_member_id. The forum member who posted the 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.

Table 4.4.30: 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.
question_id	String	UUID (36), Not Null	FK	-	References ForumQuestion.question_id. The question this answer belongs to.
forum_member_id	String	UUID (36), Not Null	FK	-	References ForumMember.forum_member_id. The forum member who posted the 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 4.4.31: Data Dictionary - ForumAnwer 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 4.4.32: 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.
forum_member_id	String	UUID (36), Not Null	FK	-	References ForumMember.forum_member_id (who reacted).
type	Integer	Enum: {0=Downvote, 1=Upvote}	-	-	Reaction type.
created_at	DateTime	Not Null	-	CURRENT_TIMESTAMP	Timestamp when the reaction was added.

Table 4.4.33: 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).
created_at	DateTime	Not Null	-	CURRENT_TIMESTAMP	Timestamp when the chat room was created.

Table 4.4.34: 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 4.4.35: 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 4.4.36: 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).
file_name	String	255, Nullable	-	-	Original name of the file uploaded.
file_size	Integer	≥ 0	-	NULL	File size in bytes (for validation or limits).

Table 4.4.37: 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 4.4.38: 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 4.4.39: 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 4.4.40: 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 4.4.41: 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.member_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 4.4.42: 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 4.4.43: 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 4.4.44: 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).
forum_id	String	UUID (36), Nullable	FK	NULL	References the Forum involved in the review (if applicable).
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 4.4.45: 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).
forum_id	String	UUID (36), Nullable	FK	NULL	References the Forum affected by the action (if applicable).

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 4.4.46: 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 4.4.47: Data Dictionary - Report Entity

TaskReport

Field Name	Data Type	Size/Constraint	Key	Default	Description
task_report_id	String	UUID (36)	PK	Auto-generated	Unique identifier for each task report.
report_id	String	UUID (36), Not Null	FK	-	References the Report to which this task report belongs.
task_completed	Integer	Not Null, ≥0	-	0	Number of tasks successfully

					completed in the report period.
task_overdue	Integer	Not Null, ≥0	-	0	Number of tasks not completed by their due date.
task_rescheduled_count	Integer	Not Null, ≥0	-	0	Number of tasks that were rescheduled during the report period.

Table 4.4.48: Data Dictionary - TaskReport 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 4.4.49: Data Dictionary - FocusReport Entity

AppUsageReport

Field Name	Data Type	Size/Constraint	Key	Default	Description
app_report_id	String	UUID (36)	PK	Auto-generated	Unique identifier for each app usage report.
report_id	String	UUID (36), Not Null	FK	-	References the associated report.
top_app_used	String	150 chars, Nullable	-	NULL	Name or package identifier of the most used app during the reporting period.
total_screen_time	Integer	In minutes	-	0	Total screen time in minutes during the reporting period.

Table 4.4.50: Data Dictionary - AppUsageReport 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 4.4.51: Data Dictionary - MoodReport Entity

SocialEngagementReport

Field Name	Data Type	Size/Constraint	Key	Default	Description
social_report_id	String	UUID (36)	PK	Auto-generated	Unique identifier for each social engagement report.
report_id	String	UUID (36), Not Null	FK	-	References the associated report record.
messages_sent_count	Integer	Unsigned, Default 0	-	0	Total number of direct or group messages sent by the

					member during the reporting period.
posts_created_count	Integer	Unsigned, Default 0	-	0	Total number of community/forum posts created by the member during the reporting period.
comments_made_count	Integer	Unsigned, Default 0	-	0	Total number of comments or replies made on posts during the reporting period.
created_at	DateTime	Not Null	-	CURRENT_TIMESTAMP	Date and time the social engagement report was generated.

Table 4.4.52: Data Dictionary - SocialEngagementReport Entity

SuggestionInsights

Field Name	Data Type	Size/Constraint	Key	Default	Description
suggestion_id	String	UUID (36)	PK	Auto-generated	Unique identifier for each suggestion insight record.
report_id	String	UUID (36), Not Null	FK	-	References the associated Report record.
suggestions_json	Text	No fixed size (large string)	-	-	Stores serialized suggestions

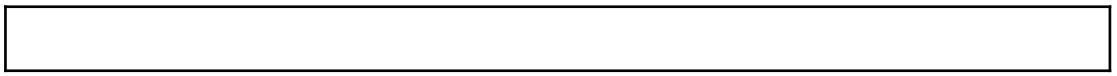
					as a text string (e.g., JSON or plain text).
insight_summary	String	VARCHAR(255), Nullable	-	NULL	Short human-readable summary of the generated insights.
generated_at	DateTime	Not Null	-	CURRENT_TIMESTAMP	Date and time when the suggestion insights were generated.

Table 4.4.53: Data Dictionary - SuggestionInsights Entity

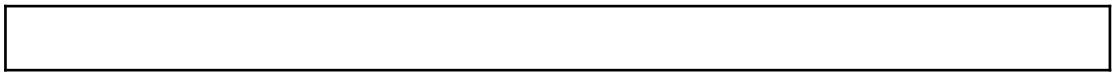
2.12 Reports Design

2.12.1 Emotion Summary Report

2.12.2 Focus Performance Report



Goal Tracking Report



2.13 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

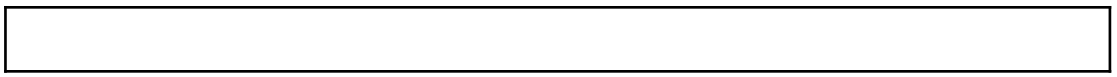


Diagram 4.6.2: Activity Diagram - View Goal Notification (Goal Assistance Module)

Manage Smart Advisor

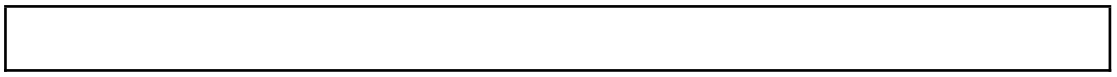


Diagram 4.6.3: Activity Diagram - Manage Smart Advisor (Goal Assistance Module)

Emotion Diary Note Module

Diagram 4.6.4: Activity Diagram - Emotion Diary Note Module: Create Emotion Diary

Focus Timer Module

Diagram 4.6.5: Activity Diagram - Focus Timer Module: Manage Focus Timer Session

Diagram 4.6.6: Activity Diagram - Focus Timer Module: Collect Rewards

User Module

Diagram 4.6.7: Activity Diagram - User Module: Register and Login Account

Diagram 4.6.8: Activity Diagram - User Module: Reset or Recover Password

Diagram 4.6.9: Activity Diagram - User Module: Update Profile

eport module

Diagram 4.6.10: Activity Diagram - Report Module: View Emotion Summary Report

Diagram 4.6.11: Activity Diagram - Report Module: View Focus Performance Report

Diagram 4.6.12: Activity Diagram - Report Module: View Goal Tracking Report

2.14 Software Architecture Design

Diagram 4.7.1: Deployment Diagram of BetterU Mobile Application

2.15 AI Algorithms

Task Categorization and Prioritization

- Algorithm: Random Forest Classifier
- Application: This algorithm helps for predicting the user task input's category and priority levels after training the model. This can efficiently reduce the users' effort to manually set the task category and priority level. Meanwhile, all the urgent tasks can be immediately detected and classified.
- Synthetic dataset: 3270 labeled task entries. Each task is annotated with category (Study, Working, Entertainment, Exercise, Personal) and priority level (1, 2, 3, 4)
- Task Category and Priority Dataset:

task_text	category	priority
watch tutorial on computer science	Study	4
complete chemistry assignment	Study	1
prepare for biology quiz	Study	1
revise biology notes	Study	2
organize study materials for english	Study	3
organize study materials for english	Study	3
prepare for physics quiz	Study	1
prepare for biology quiz	Study	1
summarize history readings	Study	2
review english flashcards	Study	3
call client for feedback	Working	1

update task tracker	Working	3
attend team meeting	Working	3
send project update email	Working	1
write marketing report	Working	2
debug application issue	Working	1
design presentation layout	Working	4
design presentation layout	Working	4
update task tracker	Working	3
review marketing documents	Working	2
attend live event	Entertainment	3
scroll through social media	Entertainment	4
chat with friends	Entertainment	2
watch funny YouTube videos	Entertainment	4
chat with friends	Entertainment	2
watch romantic movie	Entertainment	3
watch funny YouTube videos	Entertainment	4
play console games	Entertainment	3
attend live event	Entertainment	3
watch documentary movie	Entertainment	3
go for a short run	Exercise	1
cycle around the neighborhood	Exercise	2
go for a short run	Exercise	1
lift weights at gym	Exercise	1
attend marketing fitness class	Exercise	3
do 30-minute workout at home	Exercise	2

practice football skills	Exercise	2
follow PC fitness video	Exercise	2
follow console fitness video	Exercise	2
follow console fitness video	Exercise	2
Buy groceries for the week	Personal	1
Call grandparents to check in	Personal	1
Clean out your closet	Personal	2
Schedule a dentist appointment	Personal	1
Write in your personal journal	Personal	2
Plan your monthly budget	Personal	2
Do laundry and fold clothes	Personal	2
Organize your photo gallery	Personal	3
Declutter your workspace	Personal	2
Backup important files to the cloud	Personal	1

Table 4.8.1: Task Categorization and Prioritization - Task Category and Priority Dataset

Smart Task Advisor

- Algorithm: Hybrid Recommendation Model - LightFM (collaborative + content-based filtering) with Matrix Factorization and Embeddings
- Application: The algorithm will learn the user preferences and recommends the suitable tasks or scheduling strategies. This can help the system to intelligently analyze user's tasks info and user preferences for recommending a better schedule of task execution time.
- 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 (for task success / delay prediction) and LSTM (Long Short-Term Memory neural network) for temporal pattern learning
- Application: The algorithm can predict the likelihood that a task will be delayed or missed. If there is a delay predicted, the system will propose a rescheduling to a more optimal time based on the user's past behavior and daily rhythms.
- 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: Constraint Programming (Google OR-Tools) and Reinforcement Learning (RLib)
- Application: The OR-Tools will generate the initial conflict-free schedules based on the logical constraints. Meanwhile, the Reinforcement Learning will then adapt the schedule over time by analyzing which patterns will cause a higher productivity. This can enable the system to continuously improve its scheduling strategies.
- Synthetic Dataset:
    - User time availability: This dataset mainly records each user's available and unavailable day of week, start time and end time. So, it can ease the system to distribute the suitable tasks at suitable time periods without clashing with users' unavailable time period after implementing the features into the system.
    - Constraints: This dataset defines the scheduling constraints for tasks with consideration of hard and soft rules. It contains task id, constraint type (e.g. mush_finish_before, cannot_overlap_with, min_duration_minutes), value for the constraint and constraint strength (e.g. soft, hard).
    - 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:
    - User Time Availability

user_id	day_of_week	start_time	end_time	availability_status
user_001	Monday	6:00	10:00	available
user_001	Monday	10:00	12:00	available
user_001	Monday	13:00	16:00	available
user_001	Monday	12:00	13:00	unavailable

user_001	Tuesday	6:00	10:00	available
user_001	Tuesday	10:00	12:00	available
user_001	Tuesday	13:00	16:00	available
user_001	Tuesday	12:00	13:00	unavailable
user_001	Wednesday	6:00	10:00	available
user_001	Wednesday	10:00	12:00	available

Table 4.8.6: Time Optimization and Adaptive Scheduling - User Time Availability Dataset

- Constraints

task_id	**constraint_type**	**value**	**constraint_strength**
T001	min_duration_minutes	30	hard
T002	min_duration_minutes	45	hard
T003	must_finish_before	20:00	hard
T003	cannot_overlap_with	long_task	hard
T003	min_duration_minutes	120	hard
T004	must_finish_before	20:00	hard
T004	min_duration_minutes	60	hard
T005	cannot_overlap_with	long_task	hard
T005	min_duration_minutes	90	hard
T006	min_duration_minutes	25	hard

Table 4.8.7: Time Optimization and Adaptive Scheduling - Constraints Dataset

- 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 4.8.8: Time Optimization and Adaptive Scheduling - Schedule History Dataset

2.16 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.

References

Abdul Rashid, M. R., Syed Mohamad, S. N., Tajjudin, A. I. A., Roslan, N., Jaffar, A., Mohideen, F. B. S., Addnan, F. H., Baharom, N., & Ithnin, M. (2023). COVID-19 pandemic fatigue and its sociodemographic, mental health status, and perceived causes: A cross-sectional study nearing the transition to an endemic phase in Malaysia. *International Journal of Environmental Research and Public Health, 20*(5), 4476. Retrieved June 18, 2025, from https://doi.org/10.3390/ijerph20054476

Abdul Yazid, N. N. (2023). *Prevalence of mental health help-seeking attitudes among public and private universities students: A case study in Petaling*. Academia.edu. Retrieved June 20, 2025, from https://www.academia.edu/118504834/Prevalence_of_Mental_Health_Help_Seeking_Attitudes_among_Public_and_Private_Universities_Students_A_Case_Study_in_Petaling

Ahmad Adlan, A. S. (2022). *Innovative educational initiatives to train psychodynamic psychiatrists in underserved areas of the world*. Academia.edu. Retrieved June 20, 2025, from https://www.academia.edu/87468843/Innovative_Educational_Initiatives_to_Train_Psychodynamic_Psychiatrists_in_Underserved_Areas_of_the_World

Airbyte. (n.d.). *Firebase vs Firestore: A comparison*. Retrieved June 20, 2025, from https://airbyte.com/data-engineering-resources/firebase-vs-firestore

Amplework. (2024). *AI automation for repetitive tasks and resource efficiency*. Retrieved June 20, 2025, from https://www.amplework.com/blog/ai-automation-repetitive-tasks-resource-efficiency/

Android Developers. (n.d.). *Create and manage virtual devices*. Retrieved June 20, 2025, from https://developer.android.com/studio/run/managing-avds

Anuyat. (n.d.). *Using Flutter for health and wellness apps: A game-changer in user experience*. Retrieved June 20, 2025, from https://anuyat.com/flutter-health-wellness

Bahmad, H. (2021). *PTSD in the COVID-19 era*. Academia.edu. Retrieved June 20, 2025, from https://www.academia.edu/104673949/PTSD_in_the_COVID_19_Era

Bateman, P. (2024). *App fatigue: How too many apps are hurting productivity*. LinkedIn. Retrieved June 20, 2025, from https://www.linkedin.com/pulse/app-fatigue-how-too-many-apps-hurting-productivity-paul-bateman-jr--in7ce

Breiman, L. (2001). Random forests. *Machine Learning, 45*(1), 5–32. Retrieved June 20, 2025, from https://doi.org/10.1023/A:1010933404324

Chamrah, Y. (2023). *SetFit unpacked: When sentence transformers go to classification gym*. Medium. Retrieved June 21, 2025, from https://medium.com/@youssef.chamrah/setfit-unpacked-when-sentence-transformers-go-to-classification-gym-52f42f5a0d8e

Code Carbon. (2023). *Major advantages and disadvantages of Dart language*. Retrieved June 20, 2025, from https://codecarbon.com/major-advantages-and-disadvantages-of-dart-language/

Dart. (n.d.). *Dart overview*. Retrieved June 20, 2025, from https://dart.dev/overview

Design Project. (2023). *Figma collaboration and teamwork*. Retrieved June 20, 2025, from https://designproject.io/blog/figma-collaboration-teamwork

Estuary Dev. (2023). *Firestore vs Realtime Database: Which to choose?* Retrieved June 20, 2025, from https://estuary.dev/blog/firestore-vs-realtime-database

Explosion AI. (2023). *spaCy: Industrial-strength NLP in Python*. Retrieved June 20, 2025, from https://spacy.io/

FastAPI. (n.d.). *FastAPI documentation*. Retrieved June 20, 2025, from https://fastapi.tiangolo.com

Flutter. (n.d.). *Flutter documentation*. Retrieved June 20, 2025, from https://flutter.dev/

GeeksforGeeks. (2022). *GitHub Desktop*. Retrieved June 20, 2025, from https://www.geeksforgeeks.org/github-desktop/

Google. (n.d.). *Firebase overview*. Retrieved June 20, 2025, from https://blog.google/products/admob/firebase-expands-to-become-unified-app/

Google OR-Tools. (2023). *Operations research tools*. Retrieved June 20, 2025, from https://developers.google.com/optimization

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation, 9*(8), 1735–1780. Retrieved June 20, 2025, from https://doi.org/10.1162/neco.1997.9.8.1735

Human Resource Development Fund (HRDF). (2024). *The state of work-life balance in Malaysia: A report on employee well-being (2023)*. Human Resource Development Fund. Retrieved June 20, 2025, from

https://hrdcorp.gov.my/wp-content/uploads/2024/06/HRD-Corp-Annual-Report-2023-Digital-Version.pdf

IJSRD. (2024). *Task prioritization issues in mobile productivity apps*. Retrieved June 17, 2025, from https://www.ijsrd.com/articles/IJSRDV12I90019.pdf

Java Code Geeks. (2022). *Why Flutter picked Dart: A deeper dive*. Retrieved June 20, 2025, from https://www.javacodegeeks.com/2022/02/why-flutter-picked-dart.html

Jimmy Viquez. (n.d.). *Figma resources*. Retrieved June 20, 2025, from https://www.jimmyviquez.design/resources/figma

JMIR. (2025). *Privacy issues in mental health apps*. Retrieved June 20, 2025, from https://www.jmir.org/2025/1/e66762

Kiesler, S., Rainie, L., & Madden, M. (2013). *Anonymity, privacy, and security online*. Pew Research Center. Retrieved June 20, 2025, from https://www.pewresearch.org/internet/2013/09/05/anonymity-privacy-and-security-online/

Kula, M. (2015). Metadata embeddings for user and item cold-start recommendations. *RecSys 2015 Workshop on New Trends in Content-Based Recommender Systems (CBRecSys)*. Retrieved June 20, 2025, from https://doi.org/10.48550/arXiv.1507.08439

Liang, X., Wang, X., & Zhang, Y. (2018). A reinforcement learning approach to personalized schedule planning. *Proceedings of the AAAI Conference on Artificial Intelligence, 32*(1). Retrieved June 16, 2025, from https://doi.org/10.48550/arXiv.1801.00209

Lindy.ai. (2024). *What is an AI planner and why you'll want one*. Retrieved June 16, 2025, from https://www.lindy.ai/blog/ai-scheduling-assistant

Ly, K. H., Trull, T. J., & Ross, R. M. (2020). The role of digital mental health interventions in promoting psychological well-being: A systematic review. *Frontiers in Psychology, 11*, 1–14. Retrieved June 16, 2025, from https://doi.org/10.3389/fpsyg.2020.00123

Malaysian Employers Federation (MEF). (2023). *MEF Salary Survey for Executives 2023*. Malaysian Employers Federation. Retrieved June 16, 2025, from https://mefacademy.mef.org.my/publications/publication_info.aspx?ID=75

Microsoft. (2021). *Microsoft To Do*. Retrieved June 16, 2025, from https://todo.microsoft.com

Microsoft. (2022). *Visual Studio Code documentation*. Retrieved June 16, 2025, from https://code.visualstudio.com/docs

Ministry of Health Malaysia (MOH). (2024). *National Health and Morbidity Survey (NHMS) 2023: Mental health and the COVID-19 pandemic*. Ministry of Health Malaysia. Retrieved June 16, 2025, from https://www.moh.gov.my

MobileAppDaily. (n.d.). *Python for mobile apps development*. Retrieved June 20, 2025, from https://www.mobileappdaily.com/knowledge-hub/python-for-mobile-apps-development

MoldStud. (2023). *How to create a mobile app using Dart programming language?* Retrieved June 20, 2025, from https://moldstud.com/mobile-app-development/how-to-create-a-mobile-app-using-dart-programming-language/

Monterail. (n.d.). *Python for mobile app development*. Retrieved June 20, 2025, from https://www.monterail.com/blog/python-for-mobile-app-development

Monterail. (n.d.). *Why choose Flutter? 6 top reasons to use Flutter for mobile app development*. Retrieved June 20, 2025, from https://www.monterail.com/blog/why-flutter

Nelson, E. (2020). *Collaborative design tools: How Figma is changing the future of UI/UX design*. UX Planet. Retrieved June 16, 2025, from https://uxplanet.org/collaborative-design-tools-how-figma-is-changing-the-future-of-ui-ux-design-8fa42f81a1fa

Nielsen Norman Group. (2024). *Prioritization methods*. Retrieved June 16, 2025, from https://www.nngroup.com/articles/prioritization-methods/

Notion Labs Inc. (2021). *Notion*. Retrieved June 16, 2025, from https://www.notion.so

OpenAI. (n.d.). *Whisper speech recognition toolkit*. Retrieved June 20, 2025, from https://github.com/openai/whisper

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research, 12*, 2825–2830. Retrieved June 16, 2025, from https://dl.acm.org/doi/pdf/10.5555/1953048.2078195

Pew Research Center. (2013). *Anonymity, privacy, and security online*. Retrieved June 16, 2025, from https://www.pewresearch.org/internet/2013/09/05/anonymity-privacy-and-security-online/

ProcessMaker. (2024). *Repetitive tasks at work: Research and statistics 2024*. Retrieved June 16, 2025, from https://www.processmaker.com/blog/repetitive-tasks-at-work-research-and-statistics-2024/

PwC Malaysia. (2024). *Asia Pacific Workforce Hopes and Fears Survey 2024: Malaysia highlights*. PwC. Retrieved June 18, 2025, from https://www.pwc.com/my/en/assets/publications/2024/pwc-my-hopes-and-fears-malaysia-findings.pdf

Radford, A., Gao, L., & Goh, G. (2023). *Whisper: Robust speech recognition via large-scale weak supervision* [Technical report]. OpenAI. Retrieved June 16, 2025, from https://openai.com/research/whisper

Reddit. (2024). *Overwhelmed by decision fatigue – Help me pick a tool*. Retrieved June 16, 2025, from https://www.reddit.com/r/ProductivityApps/comments/1j6eype/overwhelmed_by_decision_fatigue_help_me_pick_a/

Scikit-learn. (n.d.). *Scikit-learn documentation*. Retrieved June 20, 2025, from https://scikit-learn.org/

Smartsheet. (2023). *Workers waste a quarter of the workweek on manual, repetitive tasks*. Retrieved June 16, 2025, from https://www.smartsheet.com/content-center/product-news/automation/workers-waste-quarter-work-week-manual-repetitive-tasks

spaCy. (n.d.). *spaCy NLP library*. Retrieved June 20, 2025, from https://spacy.io/

TeamDynamix. (2024). *Calculating the time wasted on repetitive manual tasks*. Retrieved June 16, 2025, from https://www.teamdynamix.com/blog/calculating-the-time-wasted-on-repetitive-manual-tasks/

Ticina, E., & Cheben, J. (2025). *The use of gamification in time management*. ResearchGate. Retrieved June 16, 2025, from https://www.researchgate.net/publication/389492629_The_Use_of_Gamification_in_Time_Management

Unfolding the Advantages. (n.d.). *10 benefits of Flutter for app development*. Retrieved June 20, 2025, from https://unfoldingtheadvantages.com/benefits-of-flutter

Vosk API. (2020). *Offline speech recognition toolkit*. Retrieved June 16, 2025, from https://alphacephei.com/vosk/

Vosk API. (n.d.). *Vosk offline speech recognition toolkit*. Retrieved June 20, 2025, from https://github.com/Puddot/vosk-api

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., ... & Rush, A. M. (2020). Transformers: State-of-the-art natural language processing. *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, 38–45. Retrieved June 16, 2025, from https://doi.org/10.18653/v1/2020.emnlp-demos.6

World Health Organization. (2023). *Mental health: Strengthening our response*. Retrieved June 20, 2025, from https://www.who.int/news-room/fact-sheets/detail/mental-health-strengthening-our-response