Emo

Mini Project Report Submitted by

Amal Antoney

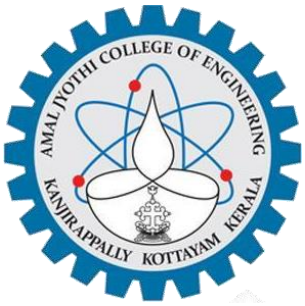
Reg. No.: AJC20MCA-I010

In Partial fulfillment for the Award of the Degree Of

INTEGRATED MASTER OF COMPUTER APPLICATIONS

(INMCA)

APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY



AMAL JYOTHI COLLEGE OF ENGINEERING AUTONOMOUS

KANJIRAPPALLY

[Affiliated to APJ Abdul Kalam Technological University, Kerala. Approved by AICTE, Accredited by NAAC. Koovappally, Kanjirappally, Kottayam, Kerala – 686518]

2024-2025

DEPARTMENT OF COMPUTER APPLICATIONS

AMAL JYOTHI COLLEGE OF ENGINEERING AUTONOMOUS

KANJIRAPPALLY

CERTIFICATE

This is to certify that the Project report, "Emo**"** is the bona fide work of

Amal Antoney (Regno: AJC20MCA-I010) in partial fulfillment of the requirements for the award of the Degree of Integrated Master of Computer Applications under APJ Abdul Kalam Technological University during the year 2024-25.

Mr. T. J. Jobin
Internal Guide

Mr. Binumon Joseph
Coordinator

Rev. Fr. Dr. Rubin Thottupurathu Jose
Head of the Department

DECLARATION

I hereby declare that the project report **"Emo"** is a bona fide work done at Amal Jyothi College of Engineering Autonomous, towards the partial fulfilment of the requirements for the award of the **Integrated Master of Computer Applications (MCA)** from **APJ Abdul Kalam Technological University**, during the academic year **2024-2025**.

Date: 07/11/24
KANJIRAPPALLY

Amal Antoney
Reg: AJC20MCA-I010

ACKNOWLEDGEMENT

First and foremost, I thank God almighty for his eternal love and protection throughout the project. I take this opportunity to express my gratitude to all who helped me in completing this project successfully. It has been said that gratitude is the memory of the heart. I wish to express my sincere gratitude to our Director (Administration) **Rev. Fr. Dr. Roy Abraham Pazhayaparampil** and Principal **Dr. Lillykutty Jacob** for providing good faculty for guidance.

I owe a great depth of gratitude towards our Head of the Department **Rev.Fr.Dr. Rubin Thottupurathu Jose** for helping us. I extend my whole hearted thanks to the project coordinator **Mr. Binumon Joseph** for his valuable suggestions and for overwhelming concern and guidance from the beginning to the end of the project. I would also express sincere gratitude to my guide **Mr. T. J. Jobin** for his inspiration and helping hand.

I thank our beloved teachers for their cooperation and suggestions that helped me throughout the project. I express my thanks to all my friends and classmates for their interest, dedication, and encouragement shown towards the project. I convey my hearty thanks to my family for the moral support, suggestions, and encouragement to make this venture a success.

Amal Antoney

ABSTRACT

.

Introduction

Maintaining mental well-being requires ongoing attention and effort, which can often feel overwhelming. To address this, Emo is developed with the primary goal of keeping users actively engaged in their mental health through gamification. By offering streak-based rewards, daily challenges, and mood tracking, the app makes mental health management both accessible and motivating. It provides users with an interactive experience that fosters regular participation and positive behavioral reinforcement.

Objectives

The main objective of this project is to develop a user-friendly mobile platform that engages users in their mental health journey through gamification. Emo will provide personalized mental health insights and encourage users to maintain consistent engagement with their well-being. The app will offer features such as mood tracking, daily challenges, streak-based rewards, and AI-driven recommendations to help users stay motivated and focused on improving their mental health.

Methodology

To achieve these objectives, the development will utilize modern mobile and server-side technologies, including React Native for the Android app interface, Node.js for backend management, and Firebase for real-time database and authentication services. The machine learning backend, powered by Flask, Scikit-Learn and Meta's LLaMA LLM, will process user data to provide personalized mood predictions and mental health insights. The app will have an intuitive interface designed to keep users engaged, regardless of their experience level, ensuring that mental health management becomes accessible and rewarding for everyone.

CONTENT

List of Abbreviation

AI - Artificial Intelligence
ML - Machine Learning
LLM - Large Language Model
NN - Neural Network
MAE - Mean Absolute Error
RMSE - Root Mean Squared Error
R^2 - Coefficient of Determination
CSV - Comma-Separated Values
JSON - JavaScript Object Notation
UI - User Interface
UX - User Experience
SDK - Software Development Ki

CHAPTER 1

INTRODUCTION

1.1 PROJECT OVERVIEW

Emo is a mobile application designed to engage users in their mental health journey by offering a centralized platform for mood tracking, personalized insights, and daily challenges. Mental well-being often requires a combination of mindfulness, self-awareness, and consistent habits, but tracking all these aspects can be difficult across multiple tools or resources. Emo simplifies this process by providing users with everything they need in one place. The app includes features such as mood tracking, streak-based rewards, AI-driven mood predictions, and expert advice, making it easier for users to stay engaged with their mental health.

The main objective of Emo is to create a user-friendly mobile platform that motivates users to consistently track their mental health and stay mindful of their emotional well-being. Through gamification, the app encourages regular participation, making mental health management accessible, rewarding, and engaging.

1.2 PROJECT SPECIFICATION

The proposed project is a mobile application that helps users manage their mental health through mood tracking, daily challenges, and AI-powered insights. The app uses gamification techniques, such as streaks and rewards, to encourage users to remain consistent in their mental health routines.

Functional Requirements:

- Mood Tracking
- Personalized AI Insights
- Daily Challenges
- Streak-Based Rewards
- Daily Journal
- Support Groups
- Admin Analytics

CHAPTER 2

SYSTEM STUDY

2.1 INTRODUCTION

The proposed project is a mobile application designed to enhance users' engagement in their mental health journey by providing mood tracking and personalized insights. The system study will focus on understanding the current practices and challenges faced by individuals managing their mental health and identifying the requirements and features of the proposed application. Additionally, the study will explore the technical feasibility of the app, including the technologies and infrastructure needed for development and deployment.

The system study will employ a combination of qualitative and quantitative research methods. It will begin with a literature review of existing research on mental health management practices, common challenges faced by users, and the technologies used in mobile mental health applications. This review will include industry reports, academic articles, and online resources.

Furthermore, the study will involve interviews and surveys with potential users of the app to understand their current mental health management practices and the challenges they encounter. These interviews and surveys will target a diverse group of individuals, ranging from those who are new to mental health tracking to seasoned users of wellness applications.

2.2 LITERATURE REVIEW

A study offering a profound insight of AI utilization in mental health context is Jin et al. [20]. They note that due to the availability of the extensive datasets across modalities, AI particularly, ML techniques, have shown a significant possibility. In this regard, the authors observe that DL methodologies have attracted quite attention given their enhanced performance over classical ML in many mental health use cases.

Schueller et al. [1] focused on the user’s attitude toward mood charting apps for finding that users use mood-related applications for tracking changes in their state. As such the study highlights the importance of features that help to promote self awareness and offer visual feedback on affective patterns which are some of the key components of analytical engines. While Overdijk et al. [2] discussed the opportunities for designing mood self-tracking, the subject of Seth, Mussel, Cramer, and Bos [13] was the user viewpoints on affective self-tracking. With 46 participants, their study showed various peculiarities of preferences and concerns regarding the sharing of mood data. It proposed integrating innovative design techniques, for instance tangible interactions, which could

be incorporated into the bespoke mood analytics engines to improve the experience of using them. Branco et al. [3] described an interaction design analysis, where features concerning people with mental health issues are required. Some of the issues they discover suggest that mental health experts be included in the creation of mood analytical engines. As such, this approach makes sure that the tools not only capture the mood but also offers recommendations based on the mood of the user.

Van Cuylenburg and Ginige [4] developed an AI-powered emotion tracking application that offers personalized suggestions to users during low moods. With a 98.46% accuracy in emotion detection, their system demonstrates the potential of AI in creating highly accurate and personalized mood analytics engines. These engines can adapt to individual emotional patterns, offering tailored interventions.

Asif et al. [5] introduced a proactive emotion tracker that uses a modified BERT model to analyze depressive text from social media and web browsing data. Achieving a 93% test accuracy, this system integrates physiological signals from wearables, providing a comprehensive analysis of the user's emotional state. Such integration is crucial for developing personalized mood analytics engines that offer a holistic view of the user's mental health.

Shah et al. [6] utilized machine learning to create personalized predictions of depressed mood, considering data from wearables and neurocognitive sampling. Their study found variability in each individual's best-fit model, highlighting the necessity for personalized, ML-guided strategies in mood analytics engines. This approach ensures that the engines cater to the unique emotional needs of each user.

Alslaity and Orji [7] conducted a systematic review of machine learning techniques used in emotion detection and sentiment analysis. The review identified supervised learning methods like Support Vector Machine (SVM) and Naïve Bayes (NB) as the most common approaches. The study emphasized the importance of standardized datasets and the exploration of non-textual data sources to improve personalization in mood analytics engines.

Recent advancements in deep learning have significantly improved the accuracy and efficiency of facial emotion recognition (FER) systems. Huang et al. (2023) employed a convolutional neural network (CNN) combined with a squeeze-and-excitation network and residual neural network for FER tasks. Their study found that features around the nose and mouth are critical facial landmarks for neural networks in emotion recognition. They achieved high accuracy rates, particularly when

using transfer learning techniques between datasets [8].

Dada et al. (2023) applied a CNN-10 model for facial expression recognition, highlighting its ability to detect spatial features, manage translation invariance, and understand expressive feature representations. Their model outperformed other architectures like VGG19 and InceptionV3 on multiple datasets, achieving accuracy scores of 99.9% on CK+, 84.3% on FER-2013, and 95.4% on JAFFE [9].

Cîrneanu et al. (2023) conducted a systematic review of neural network architectures for emotion recognition, endorsing CNN-based architectures for their feature extraction capabilities and computational efficiency. They noted the potential of these models in various applications, including healthcare, education, and security [10].

Bhattacharya et al. [16] developed an Android application for real-time mood detection and prediction using conventional machine learning techniques. Their work demonstrated the potential for near real-time emotion analysis on mobile devices, which could be particularly useful for personalized mood analytics engines. The study compared nine conventional learning methods, finding that Random Forest, Decision Tree, and Complement Naive Bayes classifiers performed marginally better than other classifiers in terms of precision, recall, F1 score, and accuracy.

Madanian et al. [17] conducted a systematic review of speech emotion recognition (SER) using machine learning. They highlighted the importance of data augmentation to address data imbalance and insufficiency issues in SER. The review also emphasized the potential of ensemble-model architectures to significantly enhance SER performance. The authors noted that transforming audio signals to spectrograms and applying feature normalization can increase SER performance and reduce the effects of speakers' diversity in recognition.

Balliu et al. [21] demonstrated the potential of using digital behavioral phenotypes captured passively and continuously from smartphones to predict depressive mood. Their study achieved high prediction accuracy of depression severity up to three weeks in advance ($R2 \geq 80\%$) using a combination of cubic spline interpolation and idiographic models. This research highlights the feasibility of obtaining high-quality longitudinal assessments of mood from a clinical population and predicting symptom severity weeks in advance using passively collected digital behavioral data.

Totterdell et al. [22] investigated the accuracy and effects of self-predictions of mood. Their study found that participants' predictions were reliably associated with their subsequent moods, although

they explained less than 10% of the variance in daily mood. Interestingly, mood was more likely to improve when participants expected it to improve, even after controlling for hassles. This suggests that self-predictions of mood may initiate processes that regulate and improve people's subsequent moods, which could be an important consideration in the design of personalized mood analytics engines.

Cho et al. [24] conducted a prospective observational cohort study to predict mood in patients with mood disorders using machine learning and passive digital phenotypes based on circadian rhythm. The study collected various digital log data through wearable devices and smartphone apps over a period of about 2 years. Their findings highlight the potential of using passive data related to circadian rhythms for mood prediction in clinical settings. This approach aligns with the growing interest in leveraging digital technologies for mental health monitoring and intervention.

Xie Ying [25] proposed a method for emotion analysis and prediction based on ECG signal acquisition. The study outlines a comprehensive approach including ECG signal acquisition, denoising, feature analysis and extraction, and emotion classification. This research contributes to the growing body of work on physiological signal-based emotion recognition, highlighting the potential of ECG signals for accurate emotion analysis. The method provides a foundation for comparing various emotion analysis techniques and expands the theoretical basis for emotion analysis using physiological signals.

Narayana et al. [26] introduced a novel approach to mood prediction by learning emotion changes via spatio-temporal attention. Their study differentiates between emotions and moods, focusing on the temporal aspect of mood prediction. The researchers explored spatial and temporal attention mechanisms, as well as parallel and sequential arrangements of these modules to improve mood prediction performance. Their findings demonstrate that incorporating emotion change information is inherently beneficial to mood prediction, and that prediction performance can be enhanced through the integration of sequential and parallel spatial-temporal attention modules. This work represents a significant advancement in applying state-of-the-art machine learning techniques to the challenge of mood prediction.

Building upon the existing research in mood prediction, we implemented a neural network-based approach using a Multi-Layer Perceptron (MLP) regressor. The implementation considers multiple factors that influence daily mood, including:

- Sleep quality

- Stress levels
- Physical activity
- Temporal patterns (day of week)

The model architecture incorporates several key design choices:

1. Feature Engineering
    - Cyclical encoding for day of week using sine and costal transformations
    - Feature normalization using MinMaxScaler
    - Careful handling of temporal dependencies
2. Neural Network Architecture
    - Two hidden layers (10 and 5 neurons respectively)
    - Maximum iteration limit of 1000 epochs
    - Consistent random state for reproducibility

B. Performance and Results The model demonstrated robust performance in mood prediction:

- Mean Absolute Error (MAE): 0.58
- Root Mean Square Error (RMSE): 0.67
- R^2 Score: 0.86

Notably, the model substantially outperformed the baseline prediction (using mean value) with a baseline MAE of 1.47, representing a 60.5% improvement in prediction accuracy. This significant performance differential demonstrates the effectiveness of the neural network approach in capturing complex relationships between input features and mood outcomes.

The high R^2 score of 0.86 indicates that the model explains 86% of the variance in mood predictions, suggesting strong predictive capability. The relatively low MAE and RMSE values further support the model's precision in predicting mood states.

Cummins et al. [18] explored the use of AI in detecting mood disorders, particularly depression and

bipolar disorder. They highlighted the potential of combining mobile and wearable technology with AI analysis to provide objective markers for these conditions. The authors emphasized the advantages of using AI-based technologies in clinical psychology settings and provided an overview of data sources, particularly information streams collectable using mobile technologies.

Roemmich et al. [19] conducted a qualitative analysis of U.S. adults' perceptions of emotion AI use in mental healthcare. Their study revealed both potential benefits and concerns. Participants noted that emotion AI could improve mental healthcare assessments, diagnoses, and treatments, as well as facilitate information disclosures and identify potential self-harm. However, they also expressed concerns about inaccurate assessments, reduced patient-provider interactions, and potential misuse of AI inferences. The authors concluded that emotion AI use in mental healthcare may be an insufficient techno-solution that could exacerbate various challenges and potentially lead to distributive, procedural, and interactional injustices.

Pandey et al. (2022) emphasized the challenge of recognizing facial emotions across different cultures, ages, and environments. They highlighted the importance of diverse and representative datasets for training robust FER models [11].

Ballesteros et al. (2022) suggested combining computer vision algorithms with psychological theories of emotion for more accurate emotion recognition. They emphasized the need for further training with diverse images and additional algorithms to distinguish between closely related emotional patterns [12].

Bhattacharya et al. (2022) developed an Android application for real-time mood detection and prediction using conventional machine learning techniques. Their work demonstrated the potential for near real-time emotion analysis on mobile devices, which could be particularly useful for personalized mood analytics engines [13].

Roemmich et al. [19] highlighted the need for careful consideration of ethical implications and user privacy when implementing emotion AI in mental healthcare settings. Future research should focus on developing frameworks for responsible AI use that protect user privacy and ensure equitable access to mental health services.

Tornero-Costa et al. [23] conducted a systematic review of methodological and quality flaws in the use of AI in mental health research. They identified several significant shortcomings, including:

- Unbalanced distribution of AI applications across mental health categories

- Poor reporting of data preprocessing and preparation steps
- Lack of assessment of model suitability before comparison
- Limited external validation of models
- Insufficient reporting of strategies for adjusting hyperparameters and model explainability
- Lack of international collaboration and data sharing

These findings highlight the need for improved methodological rigor and transparency in AI applications for mental health research.

2.3 PROPOSED SYSTEM

Emo is a mobile application designed to provide users with comprehensive tools for mood tracking and mental health management through gamification. The aim of the app is to empower users to make informed decisions about their mental health by engaging them with interactive features that promote daily check-ins and healthy habits.

Mental health is a critical aspect of overall well-being, and Emo leverages technology to make it easier for users to access resources and insights. The demand for engaging mental health tools is increasing, and Emo aims to fill this gap by offering a platform that motivates users to return regularly, helping them build resilience and awareness about their mental health.

2.4 ADVANTAGES OF PROPOSED SYSTEM

- Gamification Elements
- Comprehensive Data
- Advanced Analysis Tools
- Customizable Interface
- Personalized Support
- Daily Journaling
- Community Engagement
- User-Friendly Interface
- Support Groups

CHAPTER 3
REQUIREMENT ANALYSIS

3.1 FEASIBILITY STUDY

The proposed mobile application, **Emo**, aims to address the limitations of existing mental health apps by offering users a comprehensive, engaging, and affordable platform that incorporates gamification elements to encourage regular usage and promote mental well-being. This feasibility study will evaluate the technical, operational, and economic viability of the project to ensure successful implementation.

The study will assess the technical requirements of the application, including the necessary software and hardware to support the platform, as well as the operational needs such as staffing, training, and user support. [27] The economic feasibility will analyse the costs and benefits of the project, including potential revenue streams and the expenses associated with development, marketing, and ongoing maintenance. This feasibility study will provide valuable insights into the project's viability, identify potential risks or limitations, and develop a strategic plan for successful implementation.

3.1.1 ECONOMICAL FEASIBILITY

This analysis will evaluate the financial sustainability of Emo and determine whether the anticipated revenue streams can sufficiently cover development, marketing, and maintenance costs. The economic feasibility study will enable the project team to make informed decisions regarding the financial aspects of the application's development [27]

- Development Cost: Estimates for initial software development, including gamification features and machine learning integration.
- Marketing Cost: Budget for promoting the app to potential users through various channels.
- Maintenance Cost: Ongoing costs related to app updates, server hosting, and customer support.
- Revenue Streams: Potential income sources, including subscription models, in-app purchases, and partnerships with mental health organizations.

3.1.2 TECHNICAL FEASIBILITY

Developing Emo will require a moderate level of technical expertise, including proficiency in mobile app development, data analysis, and user experience design. [27] Key technical aspects will include ensuring the app is secure, scalable, and user-friendly to maximize user engagement and retention.

3.1.3 BEHAVIORAL FEASIBILITY

The behavioral feasibility study for Emo will evaluate user acceptance and potential adoption rates of the app. This study will analyze the target audience, their behaviors, and preferences to determine if Emo meets their mental health needs and expectations. [27]

- User Acceptance: Assessing the likelihood of users embracing the app's features and gamification elements.
- Ease of Use: Ensuring the app's interface is intuitive, allowing users of all skill levels to navigate it effortlessly.
- Training and Support: Identifying the necessary resources and materials to educate users about the app's features and providing ongoing support.

3.1.4 FEASIBILITY STUDY QUESTIONNAIRE

Have you used any digital tools or apps for mental health in your practice?

- Yes, I have used various digital tools and apps for mental health in my practice, including mood tracking apps and online therapy platforms.

Which features do you believe are most beneficial for a mental health app aimed at users?

- I believe that personalized recommendations, mood tracking and analysis, and virtual therapy sessions are most beneficial for a mental health app aimed at users.

How important is the gamification of mental health apps (e.g., streaks, rewards) in engaging users?

- I think gamification is crucial in engaging users, as it provides motivation and encouragement to prioritize their mental well-being.

Which gamification elements do you think would be most effective in motivating users?

- I think streaks, rewards, and leaderboards would be most effective in motivating users.

What key features would you need in an app to effectively provide virtual therapy?

- I would need features like audio/video calls, screen sharing, and secure messaging to effectively provide virtual therapy.

What information or metrics would you like to have access to for monitoring and tracking user progress?

- I would like to have access to metrics like mood tracking data, task completion rates, and user engagement metrics.

What types of tasks would be beneficial for users to perform in the daily tasks feature?

- I think tasks like meditation, journaling, and gratitude exercises would be beneficial for users.

What types of content should be included in the app to support user well-being?

- I think content like mental health resources, articles, and videos should be included in the app.

What features or aspects do you think these competing apps are missing?

- I think competing apps are missing personalized recommendations and virtual therapy sessions.

What factors contribute to the effectiveness of online support groups?

- I think factors like anonymity, moderation, and community engagement contribute to the effectiveness of online support groups.

3.2 SYSTEM SPECIFICATION

3.2.1 Hardware Specification

PROCESSOR - 2 Core CPU

RAM - 2 G B

Hard disk - < 1 G B

3.2.2 Software Specification

Front End	-	React-Native, CSS, React
Backend	-	Node.js, Firebase, Python
Client	-	Android Mobile Devices, Website
Technologies used	-	JS, CSS, Chart JS, Meta Llama LLM, NodeJS, Flask

Scikit-Learn, NumPy

3.3 SOFTWARE DESCRIPTION

3.3.1 React-Native

React Native is a framework for building mobile applications using JavaScript and React. Developed by Facebook, it allows developers to create native apps for Android by using a single codebase. React Native's component-based architecture simplifies the development process, enabling developers to create reusable UI components. [32]

One of the key advantages of React Native is its ability to provide a smooth and responsive user experience by leveraging native components. [32] The framework allows for hot reloading, enabling developers to see changes in real time without rebuilding the entire app. Additionally, React Native supports a wide range of third-party libraries and tools for enhanced functionality, such as navigation, state management, and UI design.

3.3.2 React

React is a JavaScript library developed by Meta (formerly Facebook) for building user interfaces, particularly single-page applications (SPAs) for the web. Known for its component-based architecture, React enables developers to create reusable UI components, which simplifies the development process and promotes modularity [31].

One of React's key strengths is its ability to provide a highly dynamic and responsive user experience through its virtual DOM (Document Object Model), which optimizes rendering and updates efficiently. React also supports features like fast reloading, allowing developers to instantly see changes in the browser without a full page reload. Additionally, React has a rich ecosystem with numerous libraries and tools that extend its functionality, including powerful tools for state management (e.g., Redux, Recoil), routing (React Router), and UI design, making it a flexible choice for building robust, interactive web applications.

3.3.3 Firebase

Firebase, developed by Google, is a powerful cloud-based platform that provides essential tools like real-time database management, authentication, and storage, streamlining backend support for mobile and web applications. Its real-time database enables instant data synchronization, making Firebase ideal for applications requiring live updates and user collaboration.

In the Emo app, Firebase handles critical functions such as file storage, user authentication, and real-time data management. Firebase Cloud Storage securely manages user-generated content, while Firebase Authentication ensures secure access for each user. The real-time database enables instant updates to mood data and activity logs, enhancing user experience by providing immediate feedback and a responsive app environment [33].

3.3.4 Python Flask

Flask is a lightweight web framework for Python that is easy to use and highly customizable. It is well-suited for building RESTful APIs, making it an excellent choice for the back end of the Emo app. With Flask, you can create endpoints to handle requests, process data, and return responses seamlessly.

Flask's simplicity and flexibility allow developers to integrate various libraries, such as scikit-learn for machine learning and NumPy for numerical computing, facilitating the implementation of advanced features like mood prediction and data analysis in the Emo app.

CHAPTER 4
SYSTEM DESIGN

4.1 INTRODUCTION

The system design phase is a critical step in the development of the Emo mobile application, which aims to enhance users' mental health and awareness through gamification and user engagement. In this phase, the project team will define the technical specifications of the platform and create a blueprint for its development. The system design will identify the various components and functionalities of the application, the relationships between them, and the technology stack that will be used to build the platform [28].

The system design will also specify the user interface design and the interaction between the application and the users. It will define the various types of users who will interact with Emo and the permissions and access levels assigned to them. Furthermore, the design will identify the different types of data that will be stored and processed by the application, as well as the data flow within the system.

4.2 UML DIAGRAM

The Unified Modelling Language (UML) is a standardized notation system used for modeling software systems. The UML diagram provides a graphical representation of the system design and is an essential part of the system design phase [29].

In the UML diagram, the various components and functionalities of the Emo application are represented using different types of diagrams, such as use case diagrams, class diagrams, activity diagrams, and sequence diagrams. These diagrams help visualize the architecture, behavior, and interactions between different components.

- **Use Case Diagrams** are employed to identify the various types of users who will interact with Emo and the actions they will perform, such as tracking moods, engaging in gamified activities, and accessing mental health resources.
- **Class Diagrams** model the data structure and relationships between different classes, such as user profiles, mood entries, and gamification elements.
- **Activity Diagrams** illustrate the workflow and sequence of actions needed to complete specific tasks, such as logging mood data or participating in a gamified challenge.
- **Sequence Diagrams** model the interactions between different components of the system, demonstrating how user actions trigger specific responses from the application.

4.2.1 Use Case Diagram

The use case diagram is a type of UML diagram used to identify the various actors or users of the Emo application and the actions or tasks they will perform. This diagram is essential in the system design phase as it helps identify the requirements of stakeholders and the functionalities of Emo.

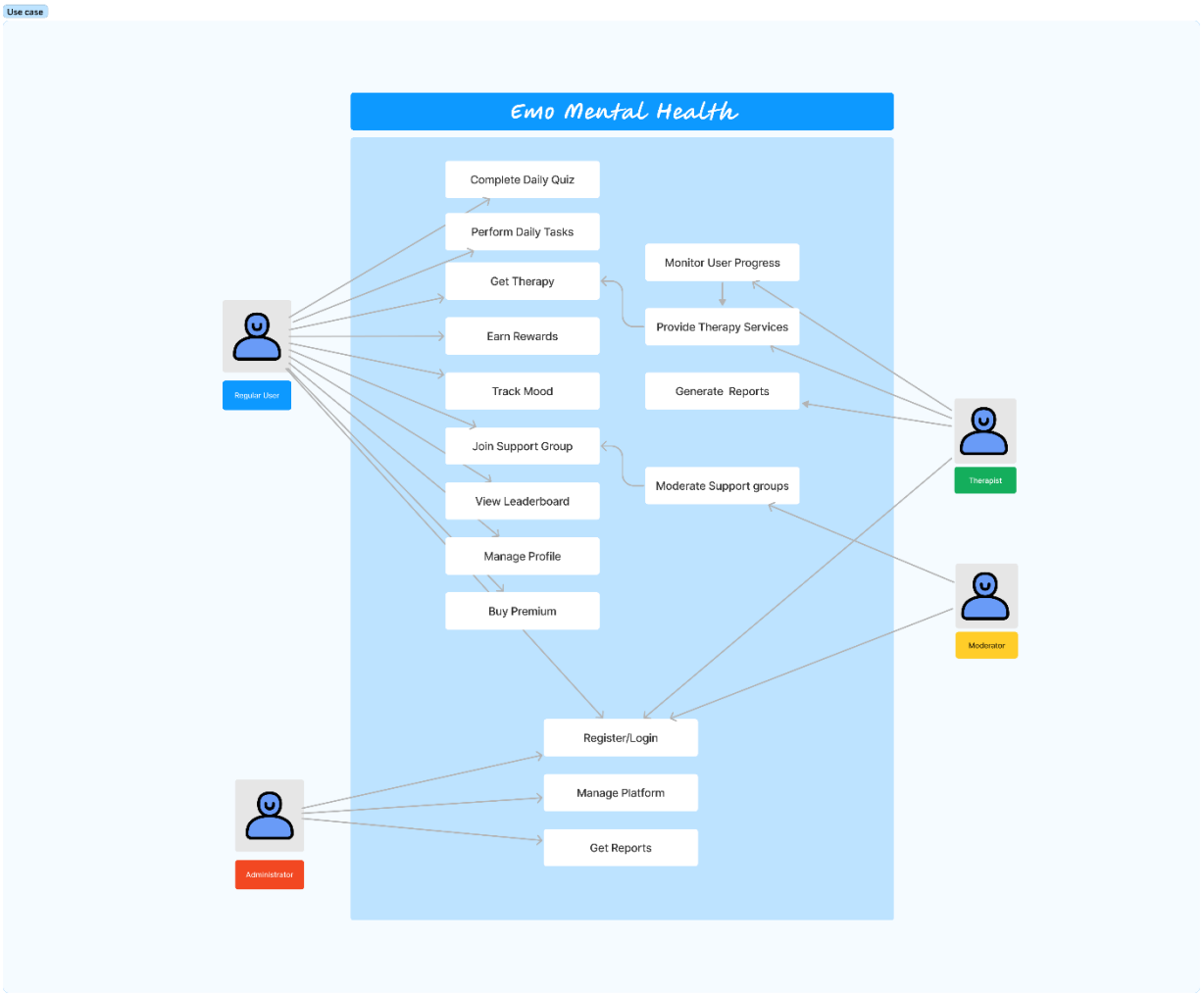


Figure 4.1

4.2.2 Sequence Diagram

The sequence diagram models the interactions between various components of the Emo application, illustrating how users interact with the system's components, including the database, analysis engine, and gamification features. It shows the order of operations, such as logging mood data,

receiving personalized feedback, and engaging in gamified activities.

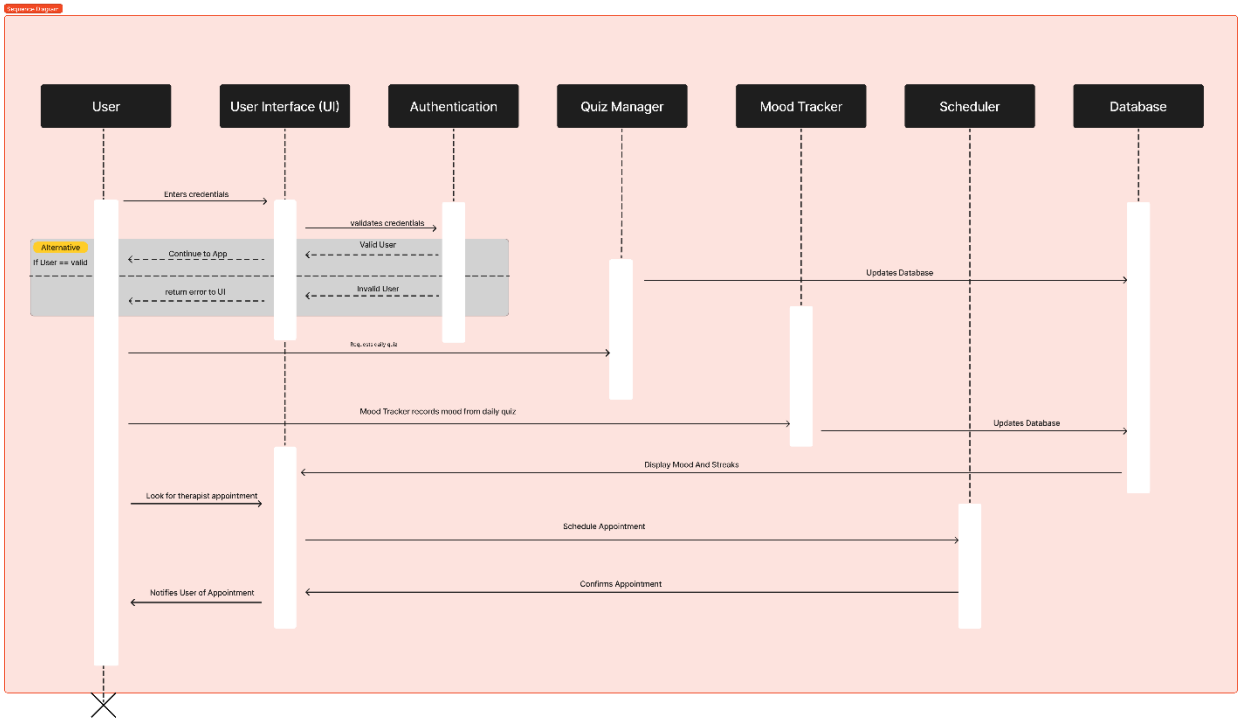


Figure 4.2

4.2.3 State Chart Diagram

The state chart diagram visualizes the behavior of the Emo application in response to different events and stimuli. It represents the various states the application can be in—such as "Idle," "Logging Mood," "Engaging in Gamification," and "Accessing Resources"—and the transitions between these states triggered by user actions or system events. This helps in understanding how the app responds to user input and manages different user interactions

4.2.4 Activity Diagram

The activity diagram models the workflow and processes within the Emo application. It outlines various activities required to complete tasks such as logging a mood, participating in a gamified challenge, or accessing mental health resources. This diagram helps to clarify the steps involved and the sequence in which they occur, enhancing the understanding of user interactions and application flow.

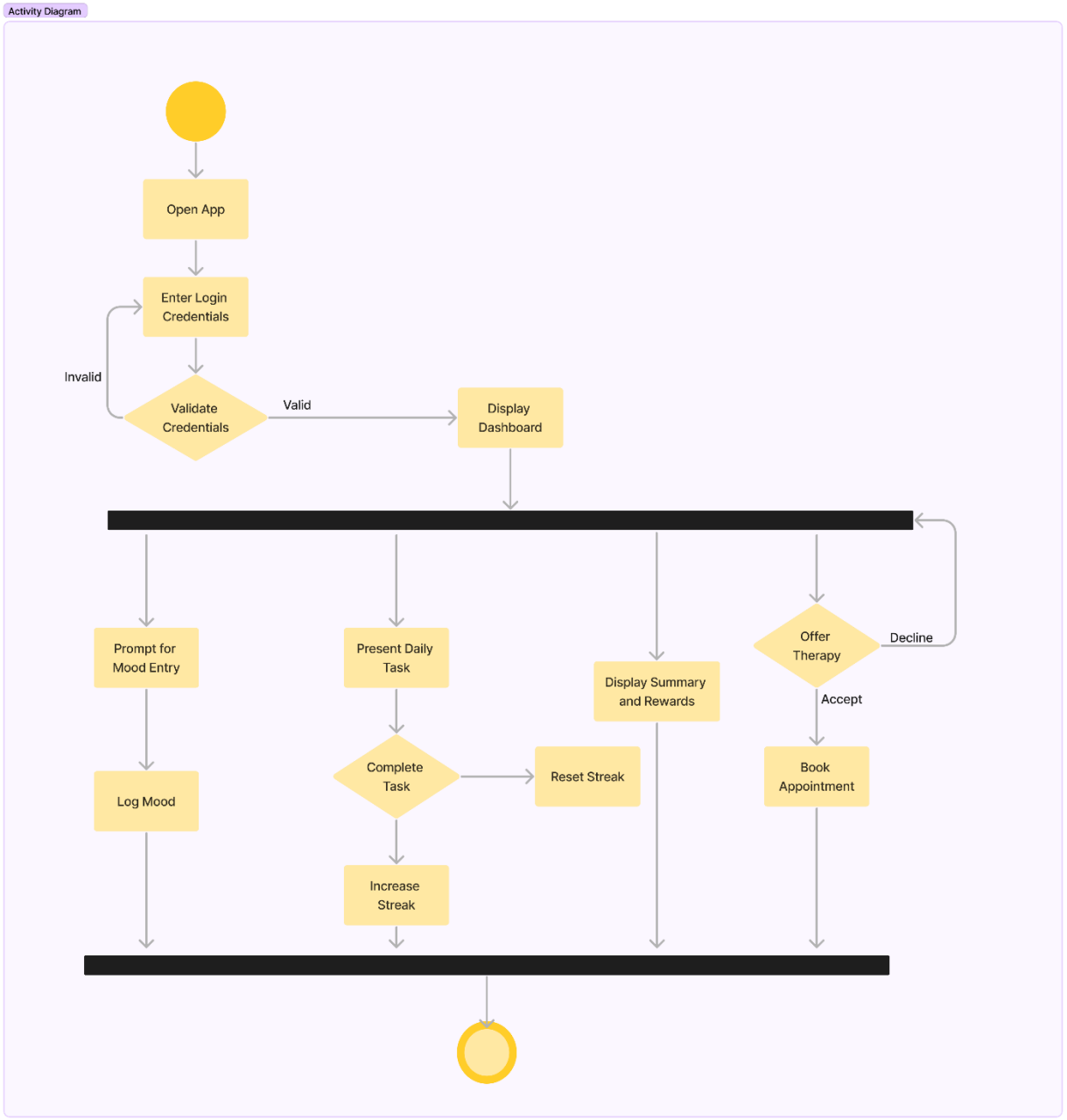

Activity Diagram
Open App
Enter Login Credentials
Invalid
Validate Credentials
Valid
Display Dashboard
Prompt for Mood Entry
Log Mood
Present Daily Task
Complete Task
Reset Streak
Increase Streak
Display Summary and Rewards
Offer Therapy
Decline
Accept
Book Appointment

Figure 4.3

4.2.5 Class Diagram

The class diagram illustrates the static structure of the Emo application, detailing the classes that make up the system.

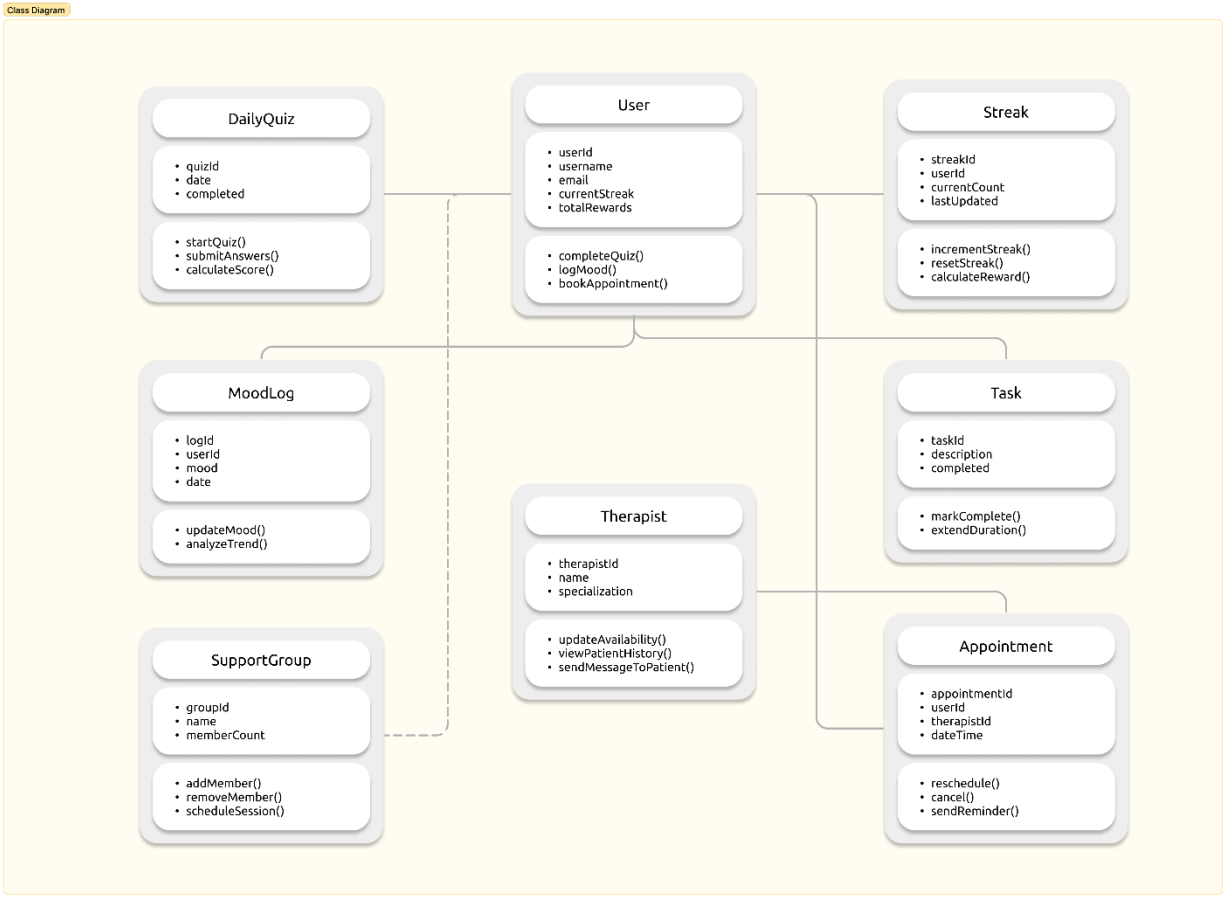


Figure 4.4

4.2.6 Object Diagram

The object diagram provides a snapshot of specific instances of the Emo application at a particular moment, showing the objects that are created when a user interacts with the app. For example, it can illustrate the specific user object that is created upon login, alongside instances of mood entries and gamified challenges. This aids in understanding the system's behavior and data management at a given point in time.

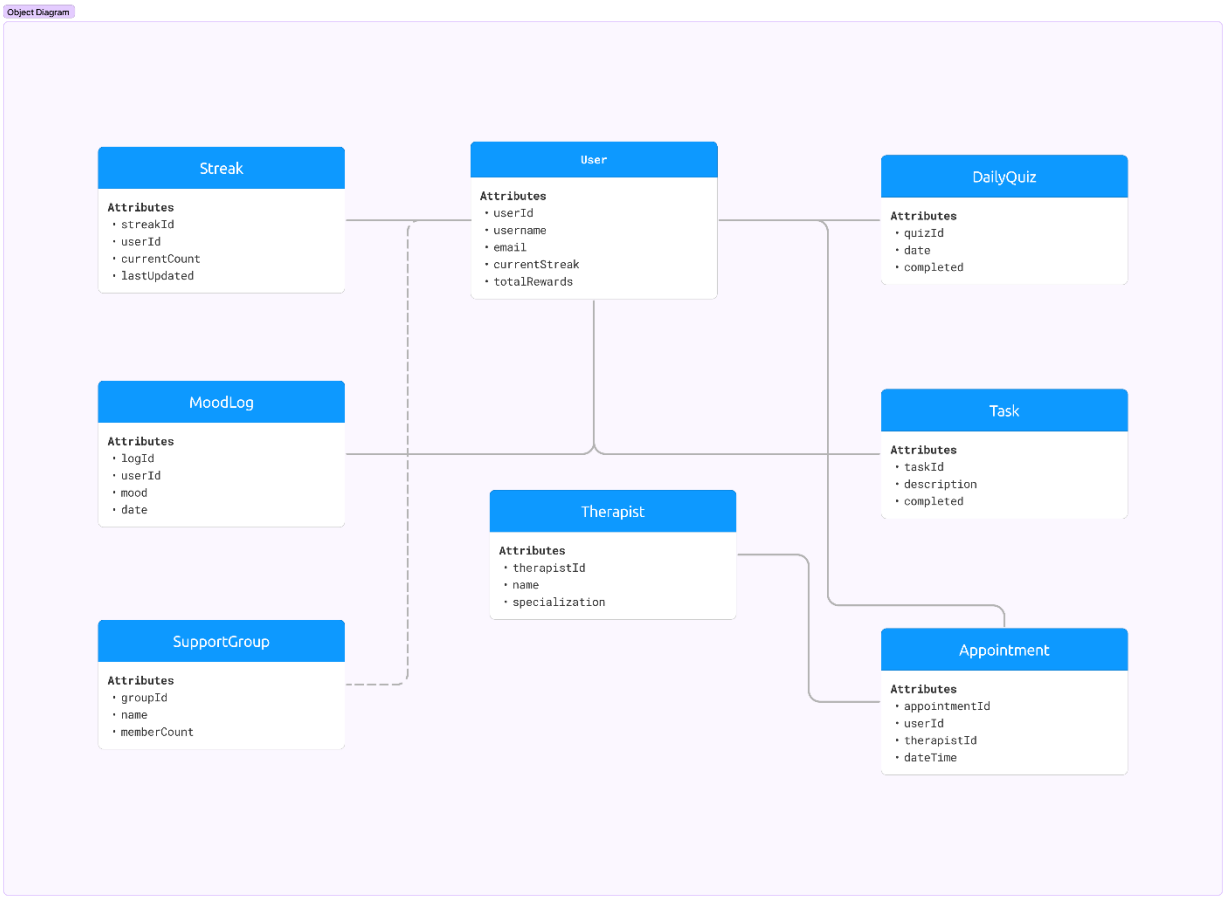


Figure 4.5

4.2.7 Comonent Diagram

A component UML diagram is a powerful tool that can help to visualize the architecture of a system, and the relationships between its different components. It is a type of structural diagram that shows the components of a system as rectangles, and the interfaces between them as lines connecting the rectangles.

Component UML diagrams are commonly used in software engineering and other fields to model complex systems, such as software applications, network infrastructures, and business processes. They can be used to identify the components of a system and their relationships, and to design more effective and efficient solutions.

4.2.8 Deployment Diagram

A deployment UML diagram is a powerful tool that can help to visualize the physical architecture

of a system, and the distribution of its components across different nodes. It is a type of structural diagram that shows the relationship between the hardware components of a system and the software components that run on them.

Deployment UML diagrams are commonly used in software engineering and other fields to model the deployment of complex systems, such as software applications, web services, and distributed systems. They can be used to identify the physical nodes that host the different components of a system, and to design more effective and efficient deployment strategies.

4.2.9 Collaboration Diagram

A collaboration UML diagram is a powerful tool that can help to visualize the interactions between the objects and components of a system. It is a type of behavioral diagram that shows how different objects and components communicate and collaborate with each other to accomplish specific tasks. Collaboration UML diagrams are commonly used in software engineering and other fields to model the interactions between different parts of a system, such as software modules, network protocols, and user interfaces. They can be used to identify the messages and events that are exchanged between different objects and components, and to design more effective and efficient communication protocols.

4.3 USER INTERFACE DESIGN USING FIGMA

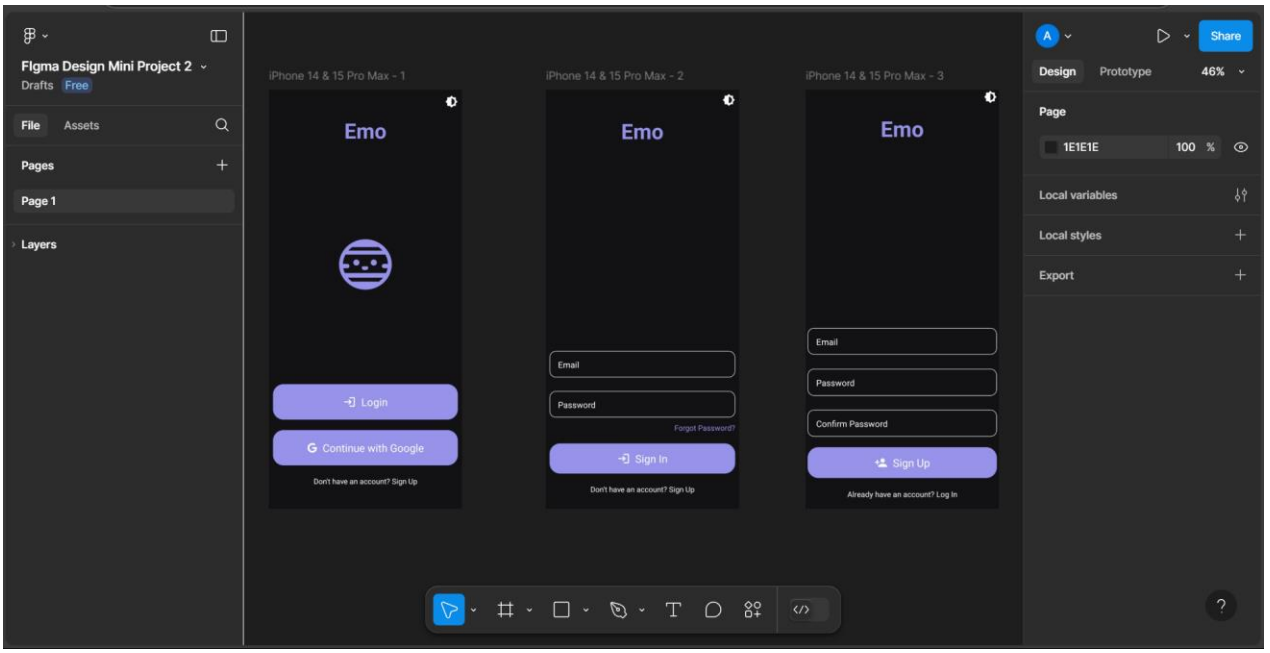


Figure 4.6

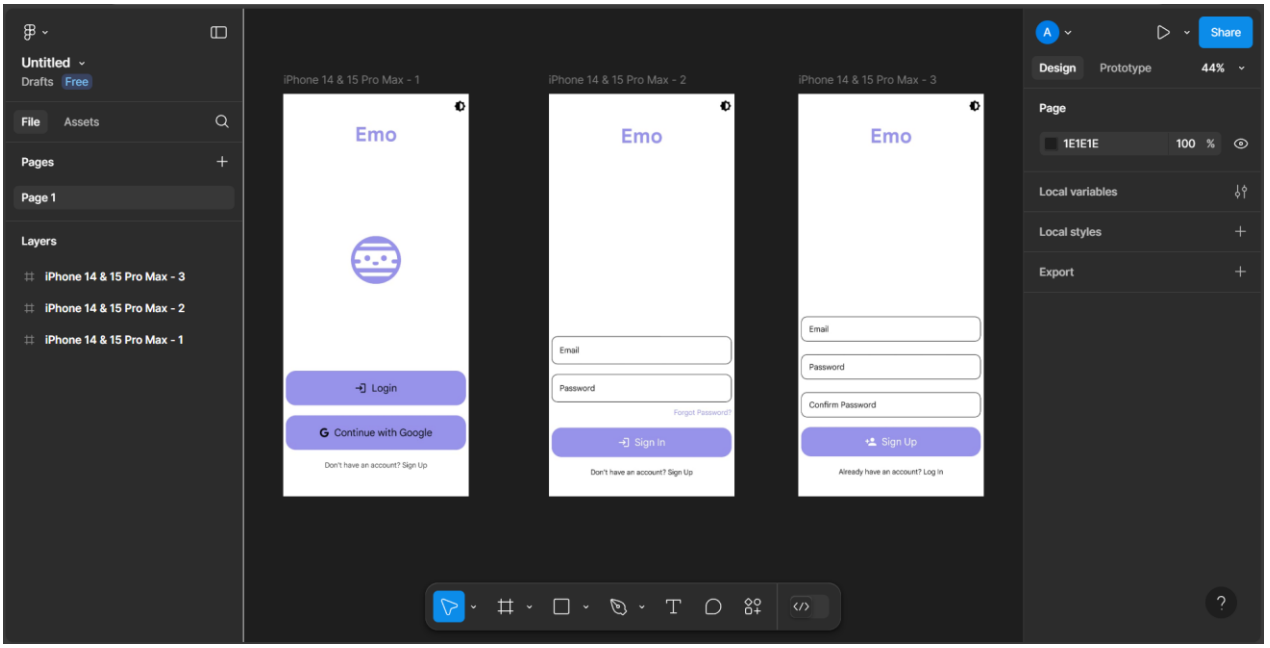


Figure 4.7

4.4 DATABASE DESIGN

4.4.1 Database Management System (DBMS)

A Database Management System (DBMS) is a software system that manages databases, allowing users to store, modify, and extract information efficiently. A DBMS provides a standardized method for organizing and managing data, ensuring data integrity and consistency across an organization. For the Emo application, Firebase will be used as the database management system. Firebase is a cloud-based platform that offers a real-time NoSQL database, making it ideal for mobile applications. It allows for the storage of structured data in JSON format, which can be easily synchronized across devices in real-time. Firebase also provides various features such as authentication, hosting, and analytics, enabling developers to build robust applications without the need for extensive backend infrastructure [33]. Its scalability and ease of integration with React Native make it a suitable choice for the Emo app, which aims to engage users in mental health awareness and support through gamification.

4.5 COLLECTION DESIGN

1. Groups Collection

Column Name	Type	Description
name	String	Name of the group
messages	Object	Collection of messages in the group
typing	Object	Object tracking typing status of users

1.1. Messages Sub-collection

Column Name	Type	Description
text	String	Content of the message
timestamp	Number	Unix timestamp of when message was sent
userId	String	ID of user who sent the message

userName	String	Name of user who sent the message
readBy	Object	Object tracking which users have read the message
deletedAt	Number (optional)	Timestamp when message was deleted
isSystemMessage	Boolean (optional)	Whether message is a system notification
replyTo	String (optional)	ID of message being replied to

2. Users Collection

Column Name	Type	Description
name	String	User's full name
email	String	User's email address
age	String	User's age
gender	String	User's gender
role	String	User's role (admin/user)
isActive	Boolean	Whether user account is active
photoURL	String (optional)	URL to user's profile photo
languagePreference	String	Preferred language
sleepHabits	String	Sleep preference (Early bird/Night owl)
preferredTherapyType	String	Preferred type of therapy
previousTherapyExperience	String	Level of therapy experience
completedChallenges	Number	Number of completed challenges
concerns	Array	List of user's mental health concerns
goals	Array	List of user's goals

interests	Array	List of user's interests

2.1. Challenges Sub-collection

Column Name	Type	Description
exercise	Number	Exercise challenge progress
gratitude	Number	Gratitude challenge progress
journal	Number	Journal challenge progress
mindfulness	Number	Mindfulness challenge progress
positivity	Number	Positivity challenge progress
sleep	Number	Sleep challenge progress
social	Number	Social challenge progress

2.2. Entries (Journal) Sub-collection

Column Name	Type	Description
content	String	Journal entry content
date	String	Date of entry

2.3. MoodEntries Sub-collection

Column Name	Type	Description
date	String	Date of mood entry
mood	Number	Mood rating (1-10)
notes	String	Additional notes
physicalActivity	Number	Physical activity rating (1-10)
sleepQuality	Number	Sleep quality rating (1-10)

stressLevel	Number	Stress level rating (1-10)

2.4. Predictions Sub-collection

Column Name	Type	Description
predictedMood	Number	Predicted mood score
recommendation	Object	Contains message and suggestion
message	String	Recommendation message
suggestion	String	Specific suggestion

2.5. EmoElevate (Subscription) Sub-collection

Column Name	Type	Description
active	Boolean	Subscription status
amount	String	Subscription amount
startDate	String	Start date of subscription
expiryDate	String	Expiry date of subscription
subscriptionType	String	Type of subscription

CHAPTER 5
SYSTEM TESTING

5.1 INTRODUCTION

The System Testing phase is a critical component of the software development life cycle, aimed at ensuring that the Emo application meets the specified requirements and performs as expected. This phase involves comprehensive testing of the entire application, including all its components and subsystems. Various testing methodologies will be employed, such as functional testing, performance testing, security testing, and usability testing [30][31].

This section outlines the results of the system testing phase for Emo. The testing was conducted to verify that the application satisfies both functional and non-functional requirements specified in the project documentation. This report includes details about the testing approach, environment, and outcomes, as well as any defects or issues identified and the steps taken to resolve them. Overall, this report provides an overview of the system testing phase, demonstrating how Emo has been validated and the results of the testing.

5.2 TEST PLAN

The purpose of this test plan is to define the testing approach, activities, and deliverables for the Emo mobile application. This plan encompasses the system testing phase and describes the various types of testing to be performed to ensure that the application meets specified requirements.
The following testing types will be performed during the system testing phase.

1. Functional Testing
2. Performance Testing
3. Security Testing
4. Usability Testing

5.2.1 Unit Testing

Unit testing is a method where individual components of Emo are tested in isolation to ensure they function correctly. This will involve testing specific modules of the application to confirm they meet specified requirements. Unit testing will be conducted by developers during the development phase to identify and fix any bugs before deployment. The goal is to ensure that each component works

as expected, facilitating smooth integration later.

5.2.2 Integration Testing

Integration testing involves testing the interactions and data flow between different modules of Emo. This ensures that the integrated components work together seamlessly. Conducted after successful unit testing, integration testing aims to identify defects or issues that may arise when combining the various modules, ensuring stability and readiness for deployment.

5.2.3 Validation Testing or System Testing

Validation testing involves testing the entire Emo application to ensure it meets specified requirements and user needs. This phase simulates real-world scenarios to validate that the application functions as intended and fulfills business objectives. Validation testing will be conducted after integration testing to ensure the system meets stakeholder expectations.

The goal of validation testing is to ensure that the final product meets the business objectives and user requirements, and is fit for purpose. Validation testing will be conducted after integration testing to ensure that the entire system is working as expected, and that it meets the expectations of the stakeholders. The testing will involve simulating real-world scenarios to validate that the system meets the requirements and can handle various user inputs and scenarios.

5.2.4 Output Testing or User Acceptance Testing

User Acceptance Testing (UAT) will involve end-users testing Emo to ensure it meets their needs and requirements. This phase will assess user satisfaction and identify any issues not detected in earlier testing stages. UAT will occur after validation testing and before deployment, ensuring users are satisfied with the application. The goal of UAT is to ensure that the system meets the user requirements and is fit for purpose, as well as identifying any issues or discrepancies that were not identified during the earlier stages of testing. UAT will be conducted after the validation testing phase and before the system is deployed to ensure that the users are satisfied with the system and are willing to use it. The testing will involve end-users performing tasks and scenarios in a simulated production environment, and

providing feedback on their experience and any issues they encountered during testing.

5.2.5 Automation Testing

Automation testing will utilize automated tools and scripts to test Emo, aiming to enhance testing efficiency and accuracy while minimizing human error. In the context of this project, automation testing will involve developing automated test scripts and executing them to test the web-based application. The goal of automation testing is to increase the efficiency and speed of testing, reduce the time and effort required for manual testing, and ensure the consistency and accuracy of testing. Automation testing will be conducted alongside manual testing to ensure that the system is functioning as intended and to reduce the risk of human error. The testing will involve developing test scripts for the different functionalities and scenarios of the system, and executing them on a regular basis to identify any defects or issues that arise. The results of the automated testing will be monitored and analyzed to identify any patterns or trends, and to make any necessary improvements to the testing process [30].

5.2.6 Jest Testing

Jest testing is a type of automation testing used to test JavaScript applications, particularly those built with React-Native. Jest is a powerful open-source testing framework that allows developers to write and execute test scripts to verify the functionality of their applications. In the context of this project, Jest testing will involve developing automated test scripts to test the web-based application [31].

The goal of Jest testing is to ensure that the application is functioning correctly across different scenarios and to identify any defects or issues that may arise during the testing process. Jest testing will be conducted alongside other types of automation testing and manual testing to ensure that the system is working as intended.

The testing will involve creating test scripts for various functionalities and scenarios of the system using Jest and executing them to identify any defects or issues that arise. The results of the Jest testing will be monitored and analysed to identify any patterns or trends, and to make any necessary improvements to the testing process.

5.2.7 Selenium Testing

Selenium is a widely used open-source tool for automating web application testing across multiple browsers and platforms. It enables developers to write test scripts that simulate user interactions, helping to ensure that applications behave as expected under various scenarios. Selenium is especially valuable for end-to-end testing, as it can mimic real user actions like clicking, typing, and navigating through the application, providing comprehensive feedback on the application's functionality [30].

In the Emo app project, Selenium testing will focus on automating tests for the web-based admin panel, which is developed to manage and analyze user data. Using Selenium, test scripts will be created to validate key functions, including user authentication, data submission, report generation, and page navigation within the admin panel. By running these tests across different browsers, Selenium ensures that the admin panel is cross-platform compatible and robust. Regular test execution and result monitoring will help in identifying and resolving potential issues, ensuring a smooth and consistent experience for administrators.

Test Case 1 – App Launching

Code

```
/**
 * @format
 */

import React from 'react';
import { render } from '@testing-library/react-native';
import App from '../App';
import { NavigationContainer } from '@react-navigation/native';
import { MenuProvider } from 'react-native-popup-menu';
import { ThemeProvider } from '../src/context/ThemeContext';

// Ensure NavigationContainer is correctly mocked
jest.mock('@react-navigation/native', () => ({
  ...jest.requireActual('@react-navigation/native'),
  NavigationContainer: ({ children }: { children: React.ReactNode }) => <>{children}</>,
}));

// Ensure GoogleSignin is correctly mocked
jest.mock('@react-native-google-signin/google-signin', () => ({
  GoogleSignin: {
    configure: jest.fn(),
    hasPlayServices: jest.fn().mockResolvedValue(true),
    signIn: jest.fn().mockResolvedValue({
```

```
      user: {
        id: '12345',
        name: 'Test User',
        email: 'test@example.com',
      },
    }),
  },
}));

jest.mock('../src/navigation/Navigation', () => {
  return function MockNavigation() {
    return <div>MockNavigation</div>;
  };
});

describe('App', () => {
  it('renders correctly', () => {
    console.log('\n🧪 Testing: App component renders correctly');
    const { toJSON } = render(
      <NavigationContainer>
        <MenuProvider>
          <ThemeProvider>
            <App />
          </ThemeProvider>
        </MenuProvider>
      </NavigationContainer>
    );
    expect(toJSON()).toBeTruthy();
    console.log('✅ App component rendered successfully');
  });
});
```

Output

```
PASS __tests__/App.test.tsx
  App
    √ renders correctly (1023 ms)

----------------------|---------|----------|---------|---------|-------------------
File                  | % Stmts | % Branch | % Funcs | % Lines | Uncovered Line #s
----------------------|---------|----------|---------|---------|-------------------
All files             |   50.38 |    10.52 |   36.36 |   51.18 |
 Emo                  |      60 |    11.11 |   56.25 |   62.26 |
  App.tsx             |      60 |    11.11 |   56.25 |   62.26 | ...88,104-118,134
 Emo/src/assets       |     100 |      100 |     100 |     100 |
  logo.png            |     100 |      100 |     100 |     100 |
 Emo/src/context      |      65 |       25 |   42.85 |   66.66 |
  ThemeContext.tsx    |      65 |       25 |   42.85 |   66.66 | 92,99,112-116
 Emo/src/utils        |   34.54 |        0 |       0 |   34.54 |
----------------------|---------|----------|---------|---------|-------------------Test Suites: 1 passed, 1 total
Tests:       1 passed, 1 total
Snapshots:   0 total
Time:        4.446 s
```

Figure 5.1

Test Report

Test Case 1

Project Name: Emo					
Login Test Case					
Test Case ID: Test_1			**Test Designed By:** Amal Antoney		
Test Priority(Low/Medium/High): High			**Test Designed Date:** 27-09-2024		
Module Name: App			**Test Executed By** : Mr. T.J. Jobin		
Test Title : Verify App component renders with valid mock configurations			**Test Execution Date:** 27-09-2024		
Description: Test App component rendering with required providers and mocks					
Pre-Condition : Required modules and mocks are set up for the test environment.					
Step	**Test Step**	**Test Data**	**Expected Result**	**Actual Result**	**Status(Pass/ Fail)**
1	Render the App component	N/A	App component should render successfully	App rendered successfully	Pass
2	Mock the NavigationContainer	N/A	NavigationContainer should load correctly	NavigationContainer mock loaded	Pass
3	Mock the Google Sign-In module	N/A	Mock Google Sign-In configuration runs	Google Sign-In mock executed	Pass
4	Test App component rendering output	N/A	toJSON() produces non-null output	toJSON() output verified	Pass
5	Log render success to console output	Console output	Console logs successful render	Log output as expected	Pass
Post-Condition: App component is rendered with all mock components and configurations as specified, ensuring compatibility with the Jest testing environment and confirming component stability under test conditions.					

Test Case 2 – Sign In

Code

```
import React from 'react';
import { render, fireEvent, waitFor } from '@testing-library/react-native';
import SignInScreen from '../src/screens/SignInScreen';
import { ThemeProvider } from '../src/context/ThemeContext';
```

```
import { NavigationContainer } from '@react-navigation/native';
import * as auth from '../src/utils/auth';

// Mock the necessary modules and functions
jest.mock('@react-native-firebase/database', () => ({
  __esModule: true,
  default: jest.fn(() => ({
    ref: jest.fn(() => ({
      once: jest.fn(() => Promise.resolve({ exists: jest.fn(() => true) })),
    })),
  })),
}));

jest.mock('react-native-snackbar', () => ({
  show: jest.fn(),
}));

jest.mock('../src/utils/auth', () => ({
  useFormValidation: jest.fn(() => ({
    emailError: '',
    passwordError: '',
    validateFields: jest.fn(),
  })),
  login: jest.fn(),
  resendVerificationEmail: jest.fn(),
  resetPassword: jest.fn(),
  logout: jest.fn(),
}));

const mockNavigation = {
  navigate: jest.fn(),
};

describe('SignInScreen', () => {
  const renderComponent = () =>
    render(
      <NavigationContainer>
```

```
      <ThemeProvider>
        <SignInScreen navigation={mockNavigation as any} />
      </ThemeProvider>
    </NavigationContainer>
  );

 it('renders correctly', () => {
  console.log('\n🧪 Testing: SignInScreen renders correctly');
  const { getByText, getByTestId } = renderComponent();

  expect(getByText('Emo')).toBeTruthy();
  expect(getByTestId('email-input')).toBeTruthy();
  expect(getByTestId('password-input')).toBeTruthy();
  expect(getByText('Sign In')).toBeTruthy();
  expect(getByText("Don't have an account? Sign Up")).toBeTruthy();
  console.log('✅ All expected elements are rendered');
 });

 it('handles sign in with valid credentials', async () => {
  console.log('\n🧪 Testing: Sign in with valid credentials');
  const { getByTestId, getByText } = renderComponent();

  (auth.login as jest.Mock).mockResolvedValue({ uid: 'testUid' });

  fireEvent.changeText(getByTestId('email-input'), 'test@example.com');
  fireEvent.changeText(getByTestId('password-input'), 'password123');
  fireEvent.press(getByText('Sign In'));

  await waitFor(() => {
   expect(auth.login).toHaveBeenCalledWith('test@example.com', 'password123');
   expect(mockNavigation.navigate).toHaveBeenCalledWith('MainScreen');
  });
  console.log('✅ Sign in successful and navigated to Home');
 });

 it('handles sign in error', async () => {
```

```
  console.log('\n 🧪 Testing: Sign in with invalid credentials');
  const { getByTestId, getByText } = renderComponent();

  (auth.login as jest.Mock).mockRejectedValue(new Error('auth/wrong-password'));

  fireEvent.changeText(getByTestId('email-input'), 'test@example.com');
  fireEvent.changeText(getByTestId('password-input'), 'wrongpassword');
  fireEvent.press(getByText('Sign In'));

  await waitFor(() => {
    expect(auth.login).toHaveBeenCalledWith('test@example.com', 'wrongpassword');
  });
  console.log('✅ Sign in error handled correctly');
 });

 it('navigates to sign up screen', () => {
  console.log('\n 🧪 Testing: Navigation to Sign Up screen');
  const { getByText } = renderComponent();

  fireEvent.press(getByText("Don't have an account? Sign Up"));

  expect(mockNavigation.navigate).toHaveBeenCalledWith('SignUp');
  console.log('✅ Successfully navigated to Sign Up screen');
 });

});
```

Output

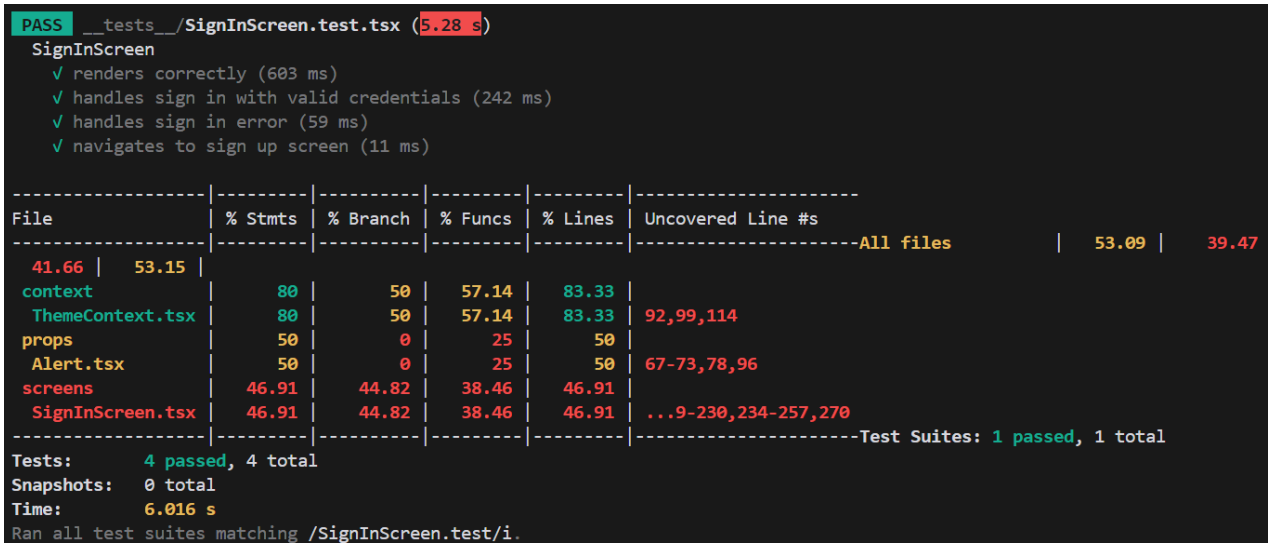


Figure 5.2

Test Report

Test Case 2

Project Name: Emo					
Login Test Case					
Test Case ID: Test_2			**Test Designed By:** Amal Antoney		
Test Priority(Low/Medium/High): High			**Test Designed Date:** 27-09-2024		
Module Name: SignInScreen			**Test Executed By** : Mr. T.J. Jobin		
Test Title : Verify SignInScreen rendering, login functionality, and navigation			**Test Execution Date:** 27-09-2024		
Description: Test SignInScreen component rendering, handling of valid and invalid login attempts, and navigation actions					
Pre-Condition : Required modules and mock configurations are set up for the test environment.					
Step	**Test Step**	**Test Data**	**Expected Result**	**Actual Result**	**Status(Pass/ Fail)**
1	Render the SignInScreen component	N/A	SignInScreen elements are rendered correctly	All elements rendered as expected	Pass
2	Enter valid credentials and submit	Email: test@example.com Password: password123	Auth module login is called with valid credentials; navigation to	Auth login successful; navigation verified	Pass

			MainScreen occurs		
3	Enter invalid credentials and submit	Email: test@example.com Password: wrongpassword	Auth module login fails; error handled correctly	Error handled as expected	Pass
4	Navigate to Sign Up screen	Press "Don't have an account? Sign Up"	Navigation to SignUp screen is triggered	Navigation to SignUp confirmed	Pass
Post-Condition: SignInScreen component renders with all elements and handles valid and invalid login attempts as expected. Correct navigation actions occur upon sign-in and sign-up link clicks.					

Test Case 3 – Profile Edit

Code

```
import React from 'react';
import { render, fireEvent, waitFor } from '@testing-library/react-native';
import EditProfileScreen from '../src/screens/EditProfileScreen';
import { ThemeProvider } from '../src/context/ThemeContext';
import { NavigationContainer } from '@react-navigation/native';
import * as ImagePicker from 'react-native-image-picker';

// Mock the necessary modules and functions
jest.mock('@react-native-firebase/database', () => ({
  firebase: {
    database: jest.fn(() => ({
      ref: jest.fn(() => ({
        once: jest.fn(() => Promise.resolve({ val: () => ({}) })),
        update: jest.fn(() => Promise.resolve()),
      })),
    })),
    auth: jest.fn(() => ({
      currentUser: { uid: 'testUid', updateProfile: jest.fn() },
    })),
  },
}));

// Update the storage mock
jest.mock('@react-native-firebase/storage', () => ({
  firebase: {
    storage: jest.fn(() => ({
      ref: jest.fn(() => ({
        putFile: jest.fn(() => Promise.resolve({ ref: { getDownloadURL: jest.fn(() =>
Promise.resolve('https://example.com/image.jpg')) } })),
      })),
    })),
  },
```

```
}));

jest.mock('react-native-image-picker', () => ({
  launchImageLibrary: jest.fn(),
}));

const mockRoute = {
  params: {
    userData: {
      name: 'Test User',
      age: 25,
      gender: 'Male',
      previousTherapyExperience: 'None',
      sleepHabits: 'Early bird',
      interests: ['Reading', 'Sports'],
      languagePreference: 'English',
      goals: ['Reduce Stress'],
      concerns: ['Anxiety'],
      preferredTherapyType: 'Cognitive Behavioral Therapy',
      photoURL: null,
    },
  },
};

const mockNavigation = {
  goBack: jest.fn(),
};

describe('EditProfileScreen', () => {
  const renderComponent = () =>
    render(
      <NavigationContainer>
        <ThemeProvider>
          <EditProfileScreen navigation={mockNavigation as any} route={mockRoute as any} />
        </ThemeProvider>
      </NavigationContainer>
    );

  it('renders correctly', () => {
    console.log('\n 🧪 Testing: EditProfileScreen renders correctly');
    const { getByText } = renderComponent();

    expect(getByText('Edit Profile')).toBeTruthy();
    expect(getByText('Basic Information')).toBeTruthy();
    expect(getByText('Gender')).toBeTruthy();
    expect(getByText('Previous Therapy Experience')).toBeTruthy();
    expect(getByText('Sleep Habits')).toBeTruthy();
    expect(getByText('Interests/Hobbies')).toBeTruthy();
    expect(getByText('Language Preference')).toBeTruthy();
    expect(getByText('Your Goals')).toBeTruthy();
    expect(getByText('Your Concerns')).toBeTruthy();
```

```
  expect(getByText('Preferred Therapy Type')).toBeTruthy();
  expect(getByText('Save')).toBeTruthy();
  console.log('✅ All expected elements are rendered');
 });

 it('handles profile picture change', async () => {
  console.log('\n💉 Testing: Profile picture change');
  const { getByText } = renderComponent();

  (ImagePicker.launchImageLibrary as jest.Mock).mockImplementation((options, callback) => {
   callback({ assets: [{ uri: 'file://test-image.jpg' }] });
  });

  fireEvent.press(getByText('Change Photo'));

  await waitFor(() => {
   expect(ImagePicker.launchImageLibrary).toHaveBeenCalled();
  });
  console.log('✅ Profile picture change handled correctly');
 });

 it('handles save profile information', async () => {
  console.log('\n💉 Testing: Save profile information');
  const { getByText, getByDisplayValue } = renderComponent();

  // Find and update the name input
  const nameInput = getByDisplayValue('Test User');
  fireEvent.changeText(nameInput, 'New Name');

  // Find and update the age input
  const ageInput = getByDisplayValue('25');
  fireEvent.changeText(ageInput, '30');

  // Update other fields
  fireEvent.press(getByText('Female'));
  fireEvent.press(getByText('Some'));
  fireEvent.press(getByText('Night owl'));
  fireEvent.press(getByText('Music'));
  fireEvent.press(getByText('Hindi'));
  fireEvent.press(getByText('Improve Sleep'));
  fireEvent.press(getByText('Depression'));
  fireEvent.press(getByText('Mindfulness-Based Therapy'));

  // Press the save button
  fireEvent.press(getByText('Save'));

  await waitFor(() => {
   expect(mockNavigation.goBack).toHaveBeenCalled();
  });
  console.log('✅ Profile information saved successfully');
```

```
  });
});
```

Output

```
PASS  __tests__/EditProfileScreen.test.tsx (5.504 s)
  EditProfileScreen
    √ renders correctly (687 ms)
    √ handles profile picture change (275 ms)
    √ handles save profile information (929 ms)

----------------------|---------|----------|---------|---------|-------------------
File                  | % Stmts | % Branch | % Funcs | % Lines | Uncovered Line #s
----------------------|---------|----------|---------|---------|-------------------
All files             |   81.81 |    65.78 |   81.25 |    85.1 |
 context              |      80 |       50 |   57.14 |   83.33 |
  ThemeContext.tsx    |      80 |       50 |   57.14 |   83.33 | 92,99,114
 screens              |   82.27 |    66.66 |      88 |   85.52 |
  ...rofileScreen.tsx |   82.27 |    66.66 |      88 |   85.52 | ...88,124,130-134
----------------------|---------|----------|---------|---------|-------------------
Test Suites: 1 passed, 1 total
Tests:       3 passed, 3 total
Snapshots:   0 total
Time:        6.217 s
Ran all test suites matching /EditProfileScreen.test/i.
```

Figure 5.3

Test Report

Test Case 3

Project Name: Emo	
Login Test Case	
Test Case ID: Test_3	**Test Designed By:** Amal Antoney
Test Priority(Low/Medium/High): High	**Test Designed Date:** 27-09-2024
Module Name: EditProfileScreen	**Test Executed By** : Mr. T.J. Jobin
Test Title : Verify EditProfileScreen rendering, profile picture change, and save functionality	**Test Execution Date:** 27-09-2024
Description: Test EditProfileScreen component rendering, changing profile picture, and saving edited	

profile information					
Pre-Condition : Required modules, mocks, and user data are set up for the test environment.					
Step	**Test Step**	**Test Data**	**Expected Result**	**Actual Result**	**Status(Pass/ Fail)**
1	Render the EditProfileScreen component	N/A	All profile elements are displayed correctly	All elements rendered as expected	Pass
2	Change profile picture	Image: file://test-image.jpg	ImagePicker launches, allowing user to select an image	ImagePicker launched; image change handled correctly	Pass
3	Edit and save profile information	Name: New Name, Age: 30, etc.	Profile information saved; navigation back occurs	Profile information saved; navigation verified	Pass
Post-Condition: EditProfileScreen renders correctly and supports editing of profile information and image change functionality. All edits are saved, and the screen navigates back on save completion..					

Test Case 4 –Web User Login

Code

```
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
import unittest
import time
from datetime import datetime
import sys
from unittest.runner import TextTestResult

class CustomTestResult(TextTestResult):
    def __init__(self, stream, descriptions, verbosity):
        super().__init__(stream, descriptions, verbosity)
        self.start_time = None
        self.test_results = []

    def startTest(self, test):
        self.start_time = time.time()
        super().startTest(test)

    def addSuccess(self, test):
        elapsed_time = time.time() - self.start_time
        self.test_results.append({
```

```
            'name': test.id().split('.')[-1],
            'result': 'PASS',
            'time': f"{elapsed_time:.2f}s"
        })
        super().addSuccess(test)

    def addError(self, test, err):
        elapsed_time = time.time() - self.start_time
        self.test_results.append({
            'name': test.id().split('.')[-1],
            'result': 'ERROR',
            'time': f"{elapsed_time:.2f}s",
            'error': str(err[1])
        })
        super().addError(test, err)

    def addFailure(self, test, err):
        elapsed_time = time.time() - self.start_time
        self.test_results.append({
            'name': test.id().split('.')[-1],
            'result': 'FAIL',
            'time': f"{elapsed_time:.2f}s",
            'error': str(err[1])
        })
        super().addFailure(test, err)

class TestLogin(unittest.TestCase):
    def setUp(self):
        self.driver = webdriver.Chrome()
        self.driver.maximize_window()
        self.driver.implicitly_wait(10)
        self.base_url = "http://localhost:3000"

    def tearDown(self):
        self.driver.quit()

    def test_successful_login(self):
        # Navigate to login page
        self.driver.get(f"{self.base_url}/login")

        # Find and fill in email field
        email_input = self.driver.find_element(By.ID, "email-address")
        email_input.send_keys("test@example.com")  # Replace with valid test email

        # Find and fill in password field
        password_input = self.driver.find_element(By.ID, "password")
        password_input.send_keys("testpassword123")  # Replace with valid test password

        # Click login button
        login_button = self.driver.find_element(
            By.XPATH,
```

```
            "//button[contains(text(), 'Sign in to your account')]"
        )
        login_button.click()

        # Wait for either dashboard redirect or error message
        try:
            # First check if we get redirected to dashboard
            WebDriverWait(self.driver, 5).until(
                EC.url_contains("/dashboard")
            )
            self.assertTrue("/dashboard" in self.driver.current_url)
        except:
            # If not redirected, we should see an error message
            error_element = WebDriverWait(self.driver, 5).until(
                EC.presence_of_element_located((By.CLASS_NAME, "bg-red-100"))
            )
            self.assertTrue(error_element.is_displayed())

    def test_invalid_credentials(self):
        # Navigate to login page
        self.driver.get(f"{self.base_url}/login")

        # Find and fill in email field with invalid credentials
        email_input = self.driver.find_element(By.ID, "email-address")
        email_input.send_keys("wrong@example.com")

        # Find and fill in password field
        password_input = self.driver.find_element(By.ID, "password")
        password_input.send_keys("wrongpassword")

        # Click login button
        login_button = self.driver.find_element(
            By.XPATH,
            "//button[contains(text(), 'Sign in to your account')]"
        )
        login_button.click()

        # Wait for error message to appear
        error_message = WebDriverWait(self.driver, 10).until(
            EC.presence_of_element_located((By.CLASS_NAME, "bg-red-100"))
        )

        # Verify error message is displayed and contains expected text
        self.assertTrue(error_message.is_displayed())
        self.assertIn("Invalid email or password", error_message.text)

    def test_google_login_button_exists(self):
        # Navigate to login page
        self.driver.get(f"{self.base_url}/login")

        # Find Google login button
```

```
        google_button = self.driver.find_element(
            By.XPATH,
            "//button[contains(text(), 'Sign in with Google')]"
        )

        # Verify button is displayed
        self.assertTrue(google_button.is_displayed())

    def test_forgot_password_link(self):
        # Navigate to login page
        self.driver.get(f"{self.base_url}/login")

        # Find and click forgot password link
        forgot_password_button = self.driver.find_element(
            By.XPATH,
            "//button[contains(text(), 'Forgot password?')]"
        )
        forgot_password_button.click()

        # Wait for reset email input to be visible
        reset_email_input = WebDriverWait(self.driver, 10).until(
            EC.presence_of_element_located((By.ID, "reset-email"))
        )

        # Verify reset password form is displayed
        self.assertTrue(reset_email_input.is_displayed())

def print_results(result):
    print("\n" + "="*80)
    print(f"Test Run Summary - {datetime.now().strftime('%Y-%m-%d %H:%M:%S')}")
    print("="*80)

    print("\nTest Results:")
    print("-"*80)
    print(f"{'Test Name':<40} {'Result':<10} {'Time':<10}")
    print("-"*80)

    for test_result in result.test_results:
        name = test_result['name']
        result_status = test_result['result']
        time_taken = test_result['time']
        print(f"{name:<40} {result_status:<10} {time_taken:<10}")
        if 'error' in test_result:
            print(f"\nError details for {name}:")
            print(f"{test_result['error']}\n")

    print("-"*80)
    print(f"\nTotal Tests: {result.testsRun}")
    print(f"Passed: {len([t for t in result.test_results if t['result'] == 'PASS'])}")
    print(f"Failed: {len([t for t in result.test_results if t['result'] == 'FAIL'])}")
    print(f"Errors: {len([t for t in result.test_results if t['result'] == 'ERROR'])}")
```

```
    print("="*80 + "\n")

if __name__ == "__main__":
    # Create a test suite
    suite = unittest.TestLoader().loadTestsFromTestCase(TestLogin)

    # Create a runner that uses our custom result class
    runner = unittest.TextTestRunner(resultclass=CustomTestResult)

    # Run the tests
    result = runner.run(suite)

    # Print the formatted results
    print_results(result)

    # Set exit code based on test results
    sys.exit(not result.wasSuccessful())
```

Output

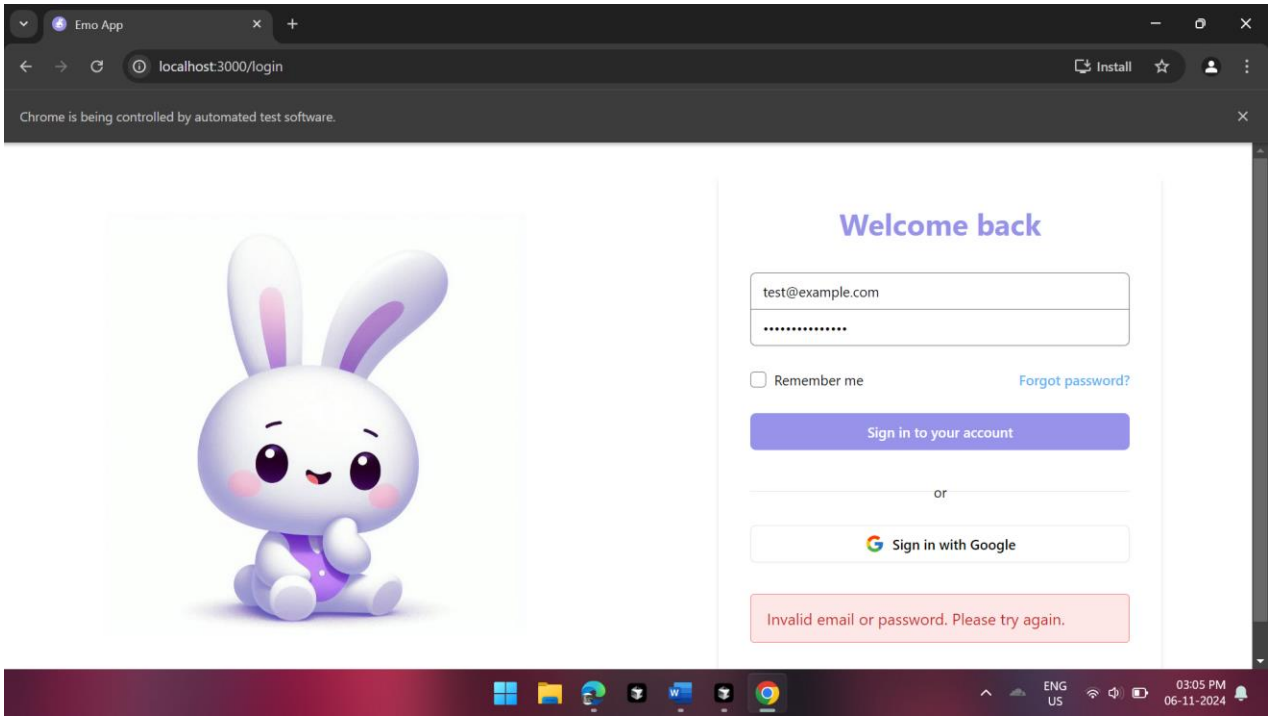


Figure 5.4

```
----------------------------------------------------------------------
Ran 4 tests in 47.321s

OK

================================================================================
Test Run Summary - 2024-11-06 15:05:12
================================================================================

Test Results:
--------------------------------------------------------------------------------
Test Name                                Result     Time
--------------------------------------------------------------------------------
test_forgot_password_link                PASS       6.22s
test_google_login_button_exists          PASS       5.93s
test_invalid_credentials                 PASS       7.31s
test_successful_login                    PASS       11.63s

Total Tests: 4
Passed: 4
Failed: 0
Errors: 0
================================================================================
```

Figure 5.5

Test Report

Test Case 4	
Project Name: Emo	
Login Test Case	
Test Case ID: Test_4	**Test Designed By:** Amal Antoney
Test Priority(Low/Medium/High): High	**Test Designed Date:** 27-09-2024
Module Name: Web LoginScreen	**Test Executed By** : Mr. T.J. Jobin
Test Title : Verify LoginScreen rendering, user authentication, and successful login	**Test Execution Date:** 27-09-2024
Description: Test the rendering of the LoginScreen component, validate user login functionality, and ensure successful login navigation.	
Pre-Condition : Required modules, test environment, and valid user credentials are set up for the test.	

Step	Test Step	Test Data	Expected Result	Actual Result	Status(Pass/ Fail)
1	Render the LoginScreen component	N/A	All login elements (email, password fields, buttons) displayed correctly	All elements rendered as expected	Pass
2	Enter valid login credentials	Email: testuser@example.com, Password: password123	User is authenticated, login button enabled	Authentication succeeded, login button enabled	Pass
3	Click on the login button	N/A	User is logged in, redirected to the home screen	User successfully redirected to the home screen	Pass
4	Enter invalid login credentials	Email: testuser@example.com, Password: wrongpassword	Error message is shown for invalid credentials	Error message displayed as expected	Pass
Post-Condition: EditProfileScreen renders correctly and supports editing of profile information and image change functionality. All edits are saved, and the screen navigates back on save completion..					

CHAPTER 6
IMPLEMENTATION

6.1 INTRODUCTION

The implementation phase is essential in the development process, translating project designs and plans into a working system. This chapter details the implementation of Emo, a mobile app focused on mental health engagement. The app leverages gamification to encourage regular use and mental health awareness. Here, we cover the tools and technologies used in developing Emo, the app's main features and functionalities, and the testing and deployment strategies employed to ensure a seamless rollout.

6.2 IMPLEMENTATION PROCEDURES

1. Deploying the Application

   The Emo app's code is deployed to the Android platform, enabling mobile access for users. This involves setting up the necessary deployment processes, configuring the server and Firebase environment, and adhering to guidelines for publishing on app stores.

2. Database Deployment

   The Firebase database is deployed and configured to work with the app, providing real-time data synchronization and secure user data storage. Firebase manages the backend data for Emo, with scalability and flexibility for future feature additions.

3. User Management

   User account management is established, allowing users to sign up, log in, and access app features. Authentication and authorization mechanisms are implemented to ensure secure handling of user data, including mood tracking, profile information, and activity logs.

4. Integration with Third-Party Services

   Emo integrates with Meta's LLaMA LLM for personalized recommendations and insights based on users' mental health status. This integration ensures that users receive engaging and meaningful content tailored to their needs. The integration is tested rigorously to guarantee data consistency and accuracy.

5. Performance Optimization

   To ensure optimal app performance, various optimization techniques are applied. Firebase's real-time data handling is streamlined, and the mobile app's frontend is optimized to reduce load times and ensure smooth interactions, including caching static assets and leveraging CDN support.

6. Quality Assurance

   Comprehensive testing is performed on Emo to verify that all features function as intended and are free from bugs. This includes automated and manual testing to ensure usability, security, and performance under varying usage conditions. Additionally, Jest is used for unit testing and integration testing for the app's core features.

7. Rollout

   After successful testing, Emo is rolled out to target users on Android platform. The rollout process includes creating awareness about the app's release, providing in-app onboarding tutorials, and offering user support for a smooth experience.

6.2.1 User Training

User training is integral to successful app adoption, helping users understand how to engage with the features that support their mental health journey. For Emo, training resources are made available to assist users in navigating the app, using mood tracking and gamification elements, and understanding the personalized recommendations provided by the AI.

6.2.2 Training on the Application Software

To help users make the most of Emo, training covers:

- Logging in and setting up profiles
- Navigating the mood-tracking features
- Utilizing expert recommendations and content tailored to mental health
- Engaging with gamified elements, tracking progress, and understanding insights provided by the AI.

6.2.3 System Maintenance

Maintaining Emo is vital for ensuring its smooth operation and security as user needs evolve. Maintenance was planned in two main areas: corrective and preventive. Corrective maintenance focuses on resolving any bugs or issues that may arise, while preventive maintenance involves regularly updating the app and backend to align with technological advancements, optimize performance, and safeguard against security vulnerabilities.

Routine updates ensure Emo remains current with the latest technology stack, which is essential for security and functionality. Security assessments are performed regularly to protect user data, and the app is continually enhanced based on user feedback to improve features and meet the evolving expectations of our user base. Regularly gathering and acting on feedback allows us to identify areas for improvement and keep Emo aligned with users' needs.

6.2.4 Hosting

1. Created a Vercel account by visiting vercel.com and signing up using GitHub credentials.
2. Connected Vercel to the project's GitHub repository for seamless integration.
3. Imported the React project repository into Vercel's dashboard.
4. Vercel automatically detected the project as a Create React App and configured the build settings.
5. Set up the build configuration with npm run build as the build command and build as the output directory.
6. Added necessary environment variables for Firebase configuration in the Vercel project settings.
7. Configured Firebase Authentication settings to include the Vercel deployment domain.
8. Updated Firebase security rules to ensure proper database access from the deployed application.
9. Initiated the deployment process by clicking the "Deploy" button.

Hosted Link: https://emo.amalantoney.com/

Hosted QR:

Screenshots:

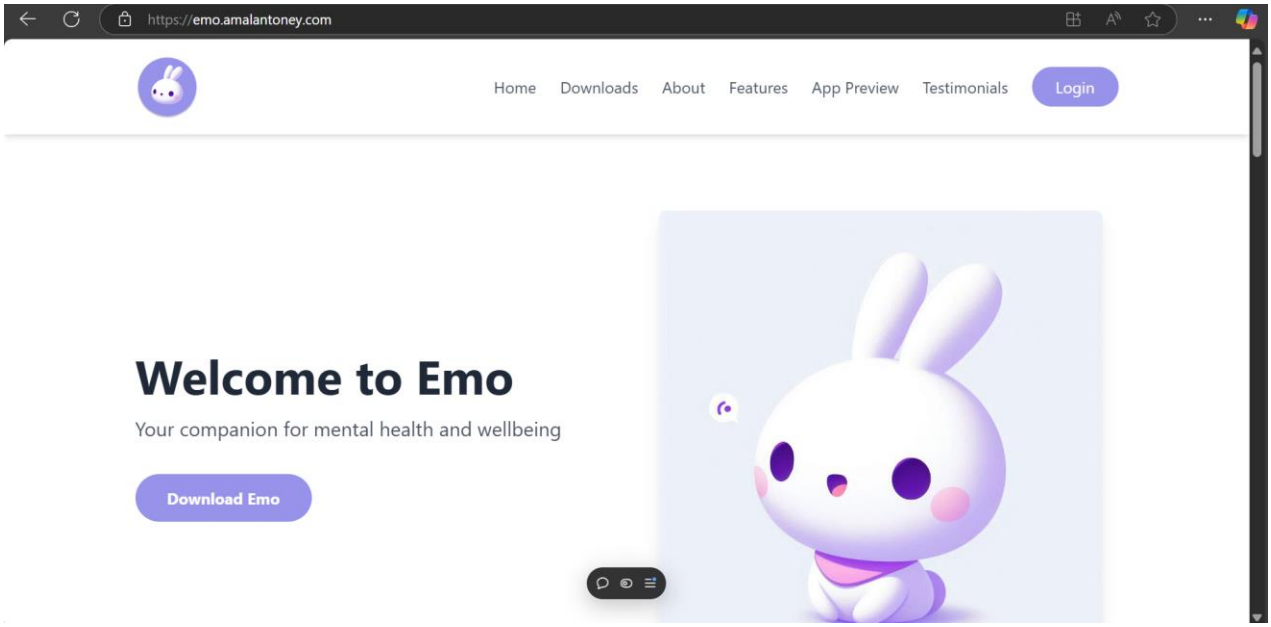


Figure 6.1

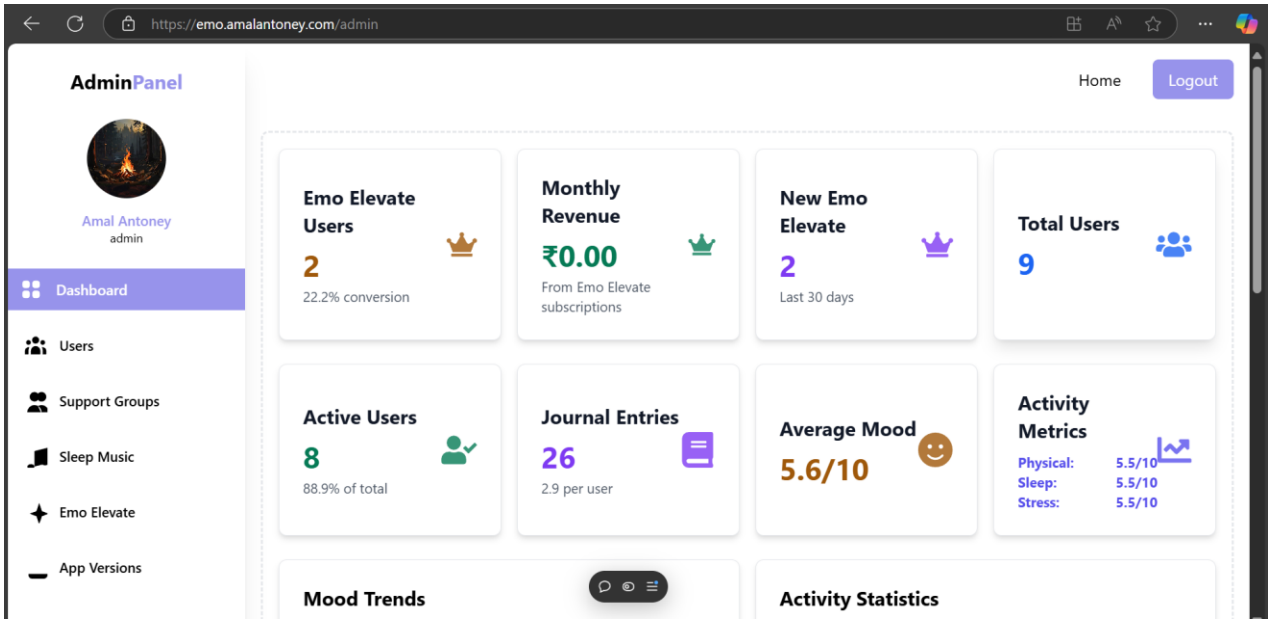


Figure 6.2

CHAPTER 7

CONCLUSION AND FUTURE SCOPE

7.1 CONCLUSION

In conclusion, the development of the Emo mobile application for mental health engagement has been both a challenging and rewarding endeavor. Utilizing modern technologies such as React Native, Firebase, and Meta's LLaMA LLM, we successfully created a robust and user-friendly platform that empowers users to actively manage their mental well-being. The application features tools for mood tracking, gamification elements, personalized insights, and educational resources, making it a valuable asset for users seeking to enhance their mental health.

Throughout the project, we conducted a thorough feasibility study, confirming the project's economic, technical, and behavioral viability. We designed and implemented effective testing strategies, ensuring that Emo meets high standards for usability and security. The user training modules were instrumental in helping users navigate the app seamlessly, enabling them to fully utilize its features.

Overall, we believe that Emo has the potential to transform how individuals approach mental health management. The insights and resources provided by the application can guide users in making informed decisions about their mental well-being, ultimately leading to improved health outcomes. We are proud of the work accomplished and are optimistic about the widespread adoption of Emo as a valuable tool in the mental health community.

7.1 FUTURE SCOPE

As technology continues to advance rapidly, there is significant potential for further improvement and innovation within Emo. Several areas can be explored to enhance the application and better meet the evolving needs of users.

These include:

1. **Expanded Features:** Adding more mood-tracking features and analytics tools to assist users in understanding their emotional patterns and triggers more effectively.
2. **Integration with Wearable Devices:** Developing integrations with wearable technology to monitor physiological indicators of mental health, such as heart rate and sleep patterns.
3. **Personalized Recommendations:** Implementing machine learning algorithms to provide tailored insights and recommendations based on users' interaction history and preferences,

promoting a more personalized experience.

4. **Mobile Accessibility Enhancements:** Continuing to enhance the mobile application's user interface and experience to ensure seamless access to mental health resources anytime and anywhere.
5. **Social Features:** Incorporating community-building features that enable users to connect with others, share experiences, and learn from each other, fostering a supportive environment.
6. **Natural Language Processing (NLP):** Integrating NLP capabilities to allow users to ask questions and receive relevant answers, creating a more interactive and responsive experience.
7. **Educational Resources:** Providing additional educational materials, such as articles, videos, and webinars, to help users improve their understanding of mental health topics and effective coping strategies.
8. **Partnerships:** Collaborating with mental health professionals and organizations to offer additional resources and expert guidance to users, enhancing the overall value of the app.

By addressing these potential developments, *Emo* can remain at the forefront of mental health support and continue to adapt to the needs of its users, ultimately contributing to a healthier and more informed society.

.

\

CHAPTER 8
BIBLIOGRAPHY

REFERENCES:

[1] S. M. Schueller, M. Neary, J. Lai, and D. A. Epstein, "Understanding People's Use of and Perspectives on Mood-Tracking Apps: Interview Study," JMIR Mental Health, vol. 8, no. 8, p. e29368, Aug. 2021. https://doi.org/10.2196/29368

[2] R. Overdijk, D. Iren, and A. Karahanoğlu, "Personalized Mental Health Mood Analytics Engine," in DRS2022: Bilbao, 25 June - 3 July, Bilbao, Spain, 2022. https://doi.org/10.21606/drs.2022.522

[3] R. Branco, M. Neves, P. Noriega, and M. Casais, "An Interaction Design Analysis of Mood Trackers," in Human Interaction and Emerging Technologies, 2020, pp. 52–58. https://doi.org/10.1007/978-3-030-61671-7_3

[4] H. C. Van Cuylenburg and T. N. D. S. Ginige, "Emotion Guru: A Smart Emotion Tracking Application with AI Conversational Agent for Exploring and Preventing Depression," in 2021 International Conference on UK-China Emerging Technologies (UCET), 2021, pp. 1–5. https://doi.org/10.1109/ucet54125.2021.9674993

[5] M. Asif et al., "Proactive Emotion Tracker: AI-driven Continuous Mood and Emotion Monitoring," arXiv:2401.13722 [cs.HC], Jan. 2024. https://doi.org/10.48550/arXiv.2401.13722

[6] R. V. Shah et al., "Personalized Machine Learning of Depressed Mood Using Wearables," Transl. Psychiatry, vol. 11, no. 1, p. 338, Jun. 2021. https://doi.org/10.1038/s41398-021-01445-0

[7] A. Alslaity and R. Orji, "Machine Learning Techniques for Emotion Detection and Sentiment Analysis: Current State, Challenges, and Future Directions," Behav. Inf. Technol., vol. 43, no. 2004, pp. 1–26, Dec. 2022. https://doi.org/10.1080/0144929X.2022.2156387

[8] Z. Y. Huang et al., "A study on computer vision for facial emotion recognition," Scientific Reports, vol. 13, Article number: 8425, 2023. https://doi.org/10.1038/s41598-023-35504-x

[9] E. G. Dada et al., "Facial Emotion Recognition and Classification Using the Convolutional Neural Network-10 (CNN-10)," Applied Computational Intelligence and Soft Computing, vol. 2023, Article ID 2457898, 2023. https://doi.org/10.1155/2023/2457898

[10] A. L. Cîrneanu, D. Popescu, and D. Iordache, "New Trends in Emotion Recognition Using Image Analysis by Neural Networks, A Systematic Review," Sensors, vol. 23, no. 16, p. 7092, 2023. https://doi.org/10.3390/s23167092

[11] A. Pandey et al., "Facial Emotion Detection and Recognition Using the Convolutional Neural Network-10 (CNN-10)," International Journal of Engineering Applied Sciences and Technology, vol. 7, no. 1, pp. 176-179, 2022. https://doi.org/10.33564/IJEAST.2022.v07i01.027

[12] J. A. Ballesteros et al., "Facial emotion recognition through artificial intelligence," Social Network Analysis and Mining, vol. 12, Article number: 139, 2022.

[13] S. Bhattacharya, A. Agarwala, and S. Roy, "Mood detection and prediction using conventional machine learning techniques on COVID19 data," Social Network Analysis and Mining, vol. 12, Article number: 139, 2022. https://doi.org/10.1007/s13278-022-00957-x

[14] W. Mellouk and W. Handouzi, "Facial emotion recognition using deep learning: review and insights," Procedia Computer Science, vol. 175, pp. 689-694, 2020. https://doi.org/10.1016/j.procs.2020.07.101

[15] A. Jaiswal, A. K. Raju, and S. Deb, "Facial Emotion Detection Using Deep Learning," 2020 International Conference for Emerging Technology (INCET), Belgaum, India, 2020, pp. 1-5. https://doi.org/10.1109/INCET49848.2020.9154121

[16] S. Bhattacharya, A. Agarwala, and S. Roy, "Mood detection and prediction using conventional machine learning techniques on COVID19 data," Social Network Analysis and Mining, vol. 12, article number 139, 2022. https://doi.org/10.1007/s13278-022-00957-x

[17] S. Madanian et al., "Speech emotion recognition using machine learning — A systematic review," Information Systems and Applications, 2023. https://doi.org/10.1016/j.iswa.2023.200266

[18] N. Cummins, F. Matcham, J. Klapper, and B. Schuller, "Artificial intelligence to aid the detection of mood disorders," in Artificial Intelligence in Precision Health, 2020, pp. 231-255. https://doi.org/10.1016/B978-0-12-817133-2.00010-0

[19] K. Roemmich et al., "Emotion AI Use in U.S. Mental Healthcare: Potentially Unjust and Techno-Solutionist," Proceedings of the ACM on Human-Computer Interaction, vol. 8, issue CSCW1, article no. 47, pp. 1-46, 2024. https://doi.org/10.1145/3637324

[20] K. W. Jin, Q. Li, Y. Xie, G. Xiao, "Artificial intelligence in mental healthcare: an overview and future perspectives," British Journal of Radiology, vol. 96, no. 1150, 20230213, 2023. https://doi.org/10.1259/bjr.20230213

[21] B. Balliu et al., "Personalized mood prediction from patterns of behavior collected with smartphones," npj Digital Medicine, vol. 7, article number: 49, 2024. https://doi.org/10.1038/s41746-024-00999-9

[22] P. Totterdell, B. Parkinson, R. B. Briner, S. A. Reynolds, "Forecasting feelings: The accuracy and effects of self-predictions of mood," Journal of Social Behavior and Personality, vol. 12, pp. 631-650, 1997.

[23] R. Tornero-Costa et al., "Methodological and Quality Flaws in the Use of Artificial Intelligence in Mental Health Research: Systematic Review," Journal of Medical Internet Research, 2022. https://doi.org/10.2196/40676

[24] C. H. Cho et al., "Mood Prediction of Patients With Mood Disorders by Machine Learning Using Passive Digital Phenotypes Based on the Circadian Rhythm: Prospective

Observational Cohort Study," J Med Internet Res, vol. 21, no. 4, p. e11029, Apr. 2019. https://doi.org/10.2196/11029

[25] X. Ying, "Research and design of mood analysis and prediction method based on ECG signal acquisition," in 2022 IEEE 5th International Conference on Information Systems and Computer Aided Education (ICISCAE), Dalian, China, 2022, pp. 1-5. https://doi.org/10.1109/ICISCAE55891.2022.9927658

[26] S. Narayana, R. Subramanian, I. Radwan, and R. Goecke, "Focus on Change: Mood Prediction by Learning Emotion Changes via Spatio-Temporal Attention," arXiv:2303.06632 [cs.HC], Mar. 2023.

[27] I. Sommerville, Software Engineering, 10th ed., Pearson, 2015. ISBN: 9780133943030

[28] F. P. Brooks, The Mythical Man-Month: Essays on Software Engineering, Anniversary Edition, Addison-Wesley, 1995. ISBN: 9780201835953

[29] M. Seidl, M. Scholz, C. Huemer, and G. Kappel, UML @ Classroom: An Introduction to Object-Oriented Modeling, 2015th ed., Springer, 2015. ISBN: 9783319127415. https://doi.org/10.1007/978-3-319-12742-2

[30] N. Garg, Test Automation using Selenium WebDriver with Java: Step by Step Guide, 1st ed., 404 pages, December 2014.

[31] H. Di Francesco, The Jest Handbook: Learn Advanced JavaScript Testing Patterns with Jest, 100 pages. Available: https://codewithhugo.com. Accessed: Nov. 2024.

[32] M. Grzesiukiewicz, Hands-on Design Patterns with React Native: Proven Techniques and Patterns for Efficient Native Mobile Development with JavaScript. Birmingham, UK: Packt Publishing, 2018.

[33] A. Kumar S., Mastering Firebase for Android Development: Build Real-time, Scalable, and Cloud-enabled Android Apps with Firebase. 1st ed. Packt Publishing, 2018.

WEBSITES:

- Firebase Documentation: https://firebase.google.com/docs
- React Native Documentation: https://reactnative.dev/docs/getting-started
- Flask Documentation: https://flask.palletsprojects.com/en/2.0.x/
- scikit-learn Documentation: https://scikit-learn.org/stable/user_guide.html
- NumPy Documentation: https://numpy.org/doc/stable/
- Machine Learning Mastery: https://machinelearningmastery.com/

CHAPTER 9

APPENDIX

9.1 SAMPLE CODE

Main app

```
import React, { useEffect, useState } from 'react';
import { AppRegistry, View, Image, Platform } from 'react-native';
import AsyncStorage from '@react-native-async-storage/async-storage';
import { ThemeProvider } from './src/context/ThemeContext';
import Navigation from './src/navigation/Navigation';
import { GoogleSignin } from '@react-native-google-signin/google-signin';
import { MenuProvider } from 'react-native-popup-menu';
import Toast from 'react-native-toast-message';
import { scheduleNotification, scheduleMorningNotification, scheduleRandomMotivation } from './src/utils/notificationService';
import PushNotification from 'react-native-push-notification';
import TrackPlayer from 'react-native-track-player';
import { GestureHandlerRootView } from 'react-native-gesture-handler';
import AnimatedSplash from "react-native-animated-splash-screen";


// Add this function to create the notification channel
const createNotificationChannel = () => {
  PushNotification.createChannel(
    {
      channelId: "journal-reminders", // This is used to identify the channel
      channelName: "Journal Reminders", // This is the name visible to the user
      channelDescription: "Reminders to upload daily journal",
      playSound: true,
```

```
      soundName: "default",
      importance: 4, // 4 = Importance.HIGH
      vibrate: true,
    },
    (created) => console.log(`createChannel returned '${created}'`)
  );
};

// Add a new channel for morning motivation
const createMorningMotivationChannel = () => {
  PushNotification.createChannel(
    {
      channelId: "morning-motivation",
      channelName: "Morning Motivation",
      channelDescription: "Daily motivational quotes",
      playSound: true,
      soundName: "default",
      importance: 4,
      vibrate: true,
    },
    (created) => console.log(`createMorningMotivationChannel returned '${created}'`)
  );
};
```

```
// Add a new channel for random motivation
const createRandomMotivationChannel = () => {
  PushNotification.createChannel(
    {
      channelId: "random-motivation",
      channelName: "Random Motivation",
      channelDescription: "Random motivational messages throughout the day",
      playSound: true,
      soundName: "gentle_chime.mp3", // Use the same sound as journal reminders or choose a different one
      importance: 4,
      vibrate: true,
    },
    (created) => console.log(`Random motivation channel created: ${created}`)
  );
};

const requestNotificationPermissions = async () => {
  if (Platform.OS === 'ios') {
    const authStatus = await PushNotification.requestPermissions(['alert', 'badge', 'sound']);
    console.log('Notification authorization status:', authStatus);
    return authStatus.alert && authStatus.badge && authStatus.sound;
  } else {
```

```
    // For Android, permissions are requested when creating the notification channel
    return true;
  }
};

createNotificationChannel();
createMorningMotivationChannel();
createRandomMotivationChannel();

GoogleSignin.configure({
  webClientId: '945356403658-ob6huter034u78cf4d7o5k1ufc84akiu.apps.googleusercontent.com',
});

PushNotification.configure({
  onRegister: function (token) {
    console.log("TOKEN:", token);
  },

  onNotification: function (notification) {
    console.log("NOTIFICATION:", notification);
  },
  popInitialNotification: true,
  requestPermissions: false, // Set this to false, we'll request permissions
```

```
manually
});

const App: React.FC = () => {
 const [isLoaded, setIsLoaded] = useState(false);

 useEffect(() => {
  const setupApp = async () => {
   try {
    await TrackPlayer.setupPlayer();
    console.log('Track Player initialized');

    const permissionGranted = await requestNotificationPermissions();
    if (permissionGranted) {
     const lastScheduled = await
AsyncStorage.getItem('lastNotificationSchedule');
     const currentDate = new Date().toDateString();

     if (lastScheduled !== currentDate) {
      scheduleNotification();
      scheduleMorningNotification();
      scheduleRandomMotivation();
      await AsyncStorage.setItem('lastNotificationSchedule', currentDate);
      console.log('Notifications scheduled for:', currentDate);
     } else {
```

```
            console.log('Notifications already scheduled for today');
          }
        } else {
          console.log('Notification permissions not granted');
        }
      } catch (error) {
        console.error('Error setting up app:', error);
      } finally {
        // Delay hiding the splash screen to ensure it's visible for a minimum time
        setTimeout(() => setIsLoaded(true), 2000);
      }
    };

    setupApp();

    return () => {
      const cleanupTrackPlayer = async () => {
        try {
          await TrackPlayer.reset();
          await TrackPlayer.remove([]);
        } catch (error) {
          console.error('Error cleaning up TrackPlayer:', error);
        }
      };
```

```
    cleanupTrackPlayer();
  };
}, []);

return (
  <AnimatedSplash
    translucent={true}
    isLoaded={isLoaded}
    logoImage={require("./src/assets/logo.png")}
    backgroundColor={"#1E1E1E"}
    logoHeight={200}
    logoWidth={200}
    logoOpacity={0}
    disableBackgroundImage={false}
    duration={2000}
    animationDuration={1500}
  >
    <GestureHandlerRootView style={{ flex: 1 }}>
      <ThemeProvider>
        <MenuProvider>
          <Navigation />
          <Toast />
        </MenuProvider>
      </ThemeProvider>
    </GestureHandlerRootView>
```

```
            </AnimatedSplash>
          );
        };

        AppRegistry.registerComponent('Emo', () => App);
export default App;
```

9.2 Screen Shots

Login

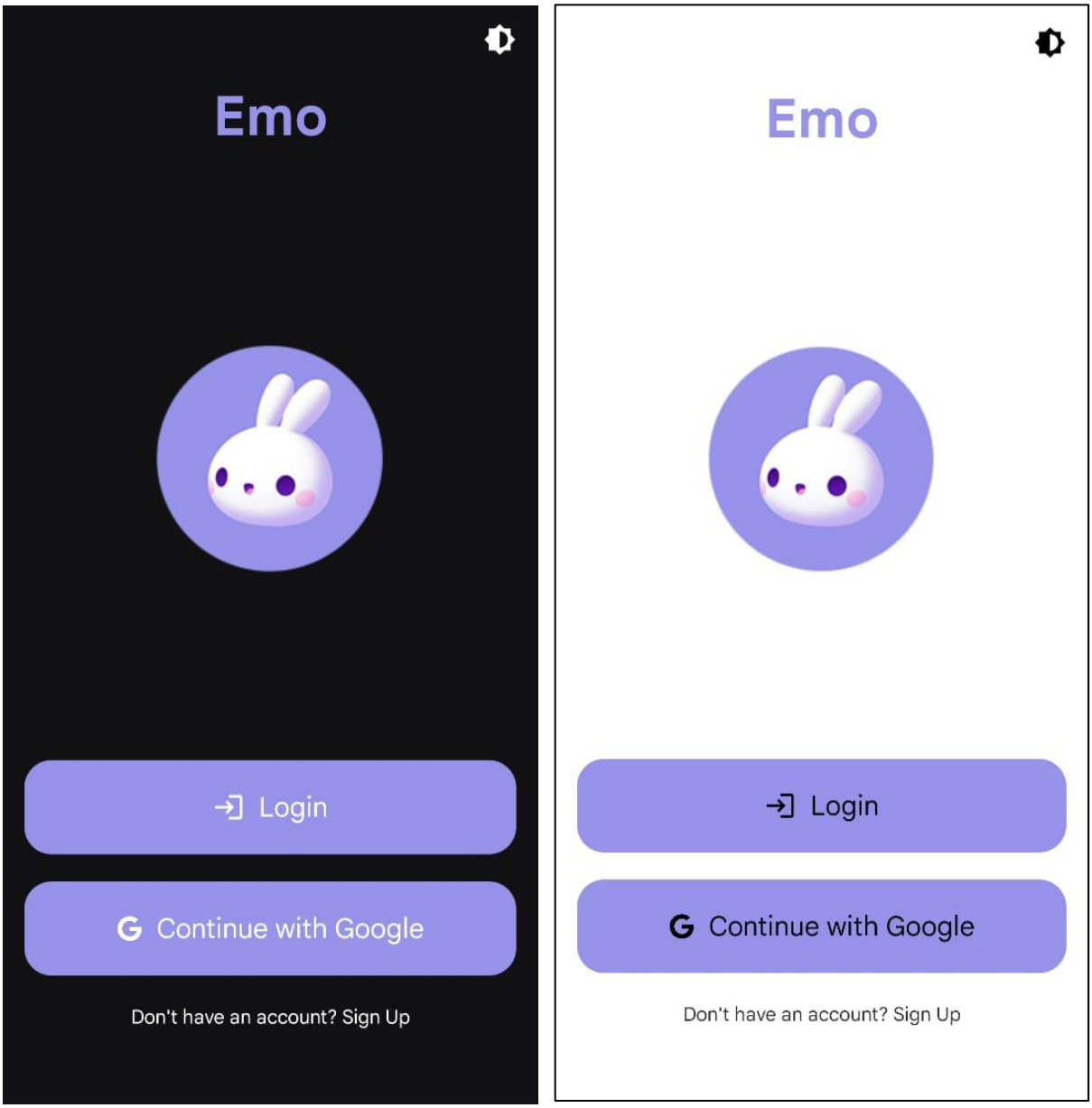


Figure 9.0

Dashboard

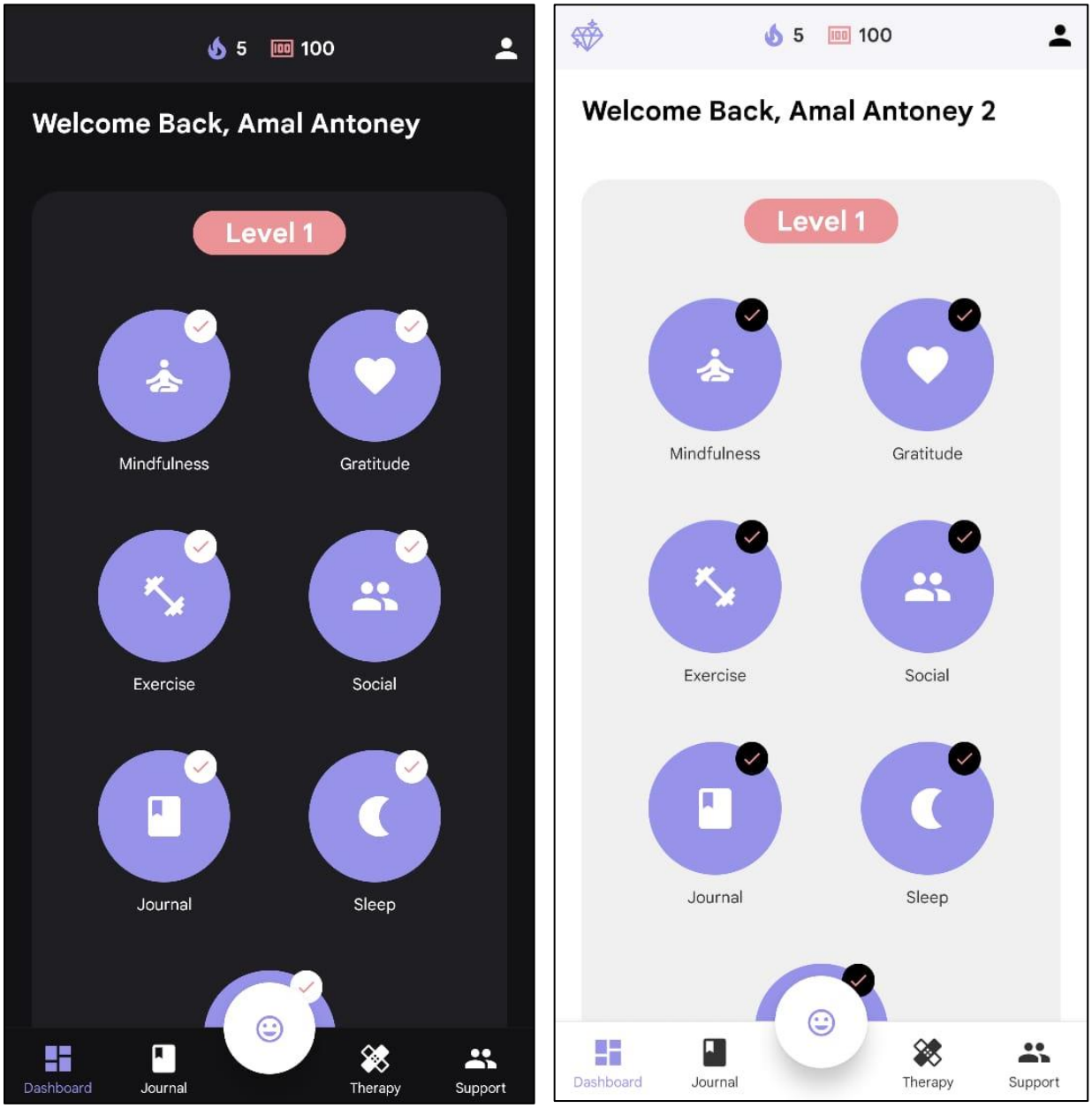


Figure 9.1

Profile

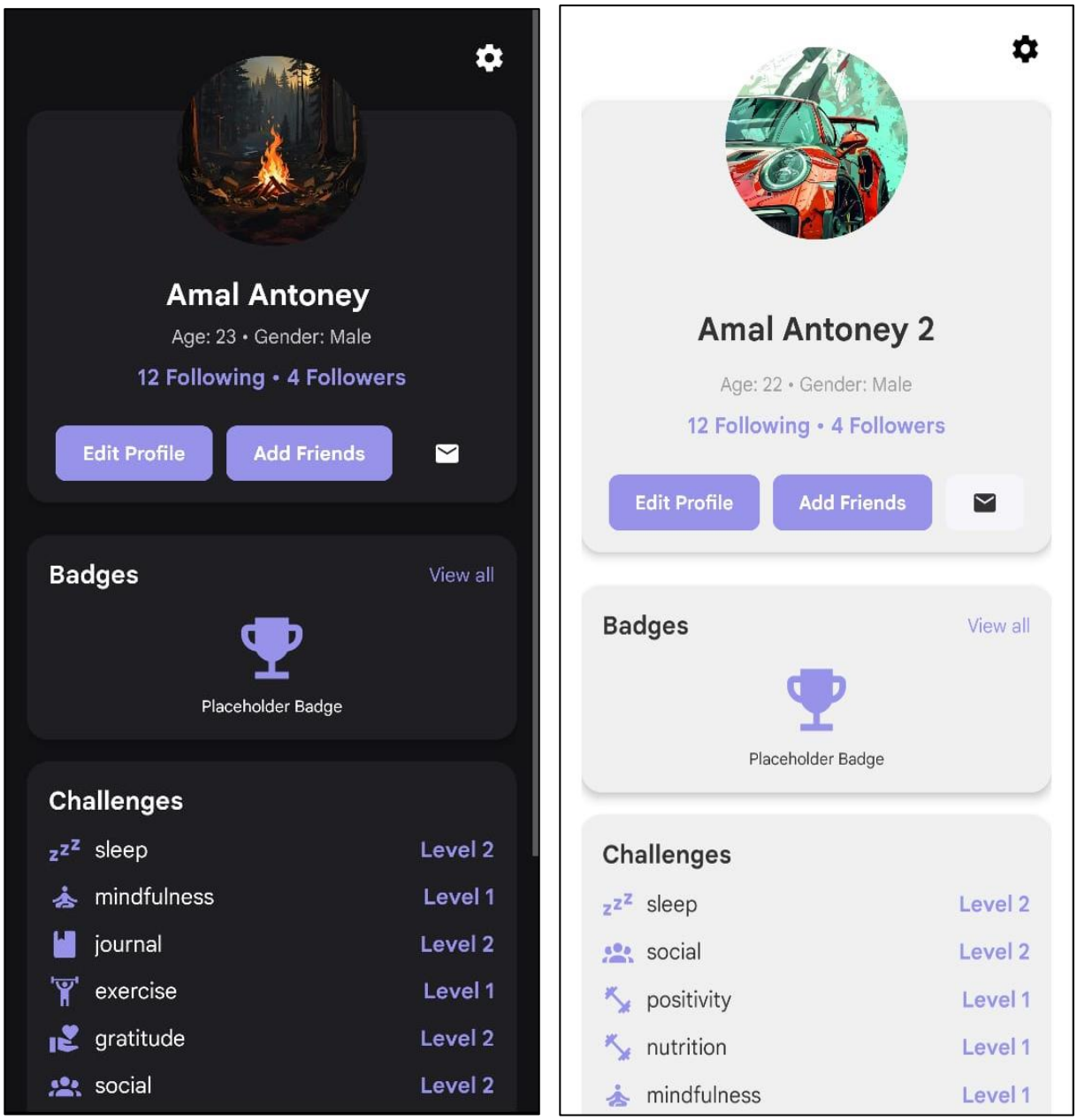


Figure 9.2

Daily Journal

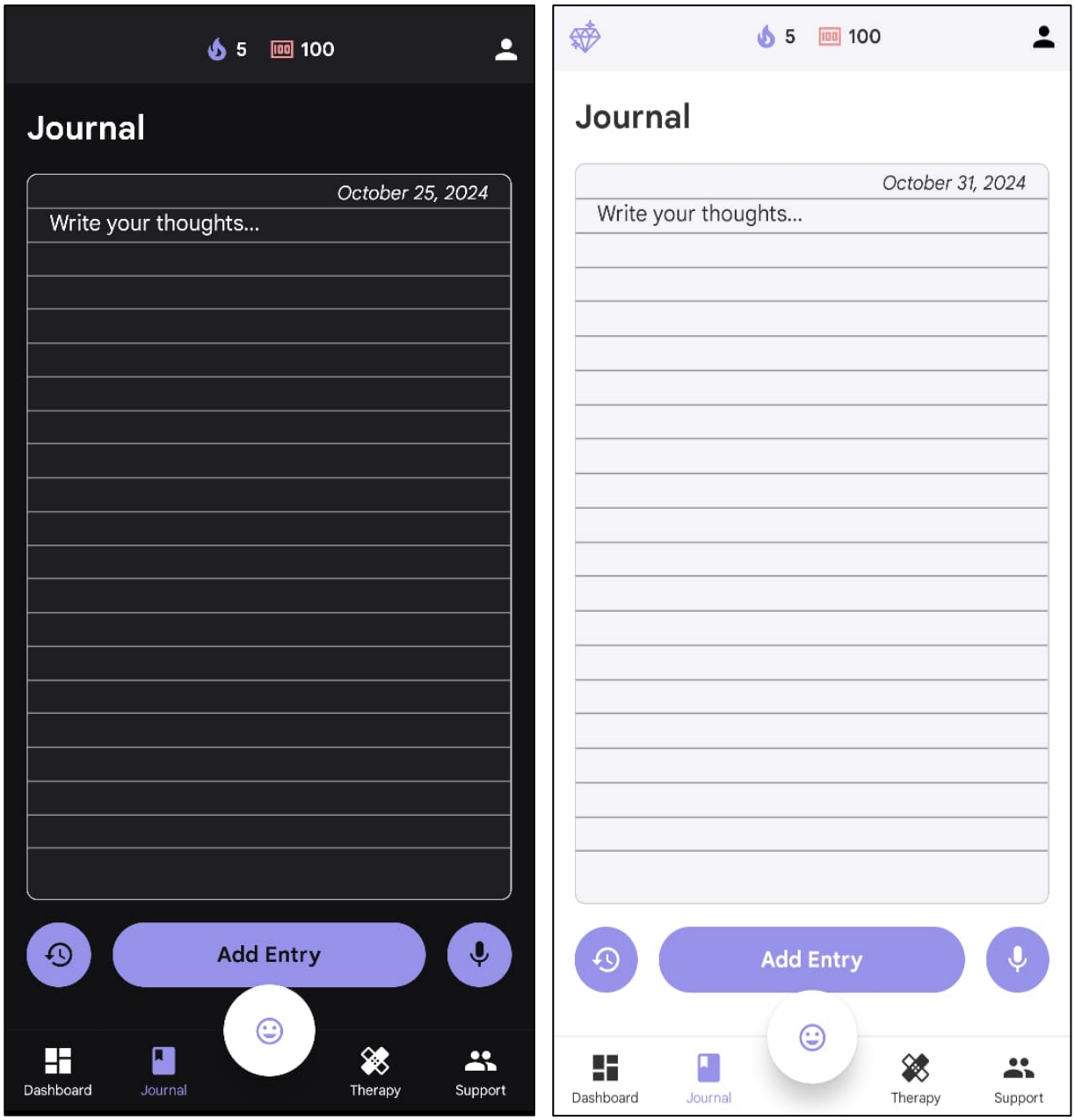


Figure 9.3

Support Groups

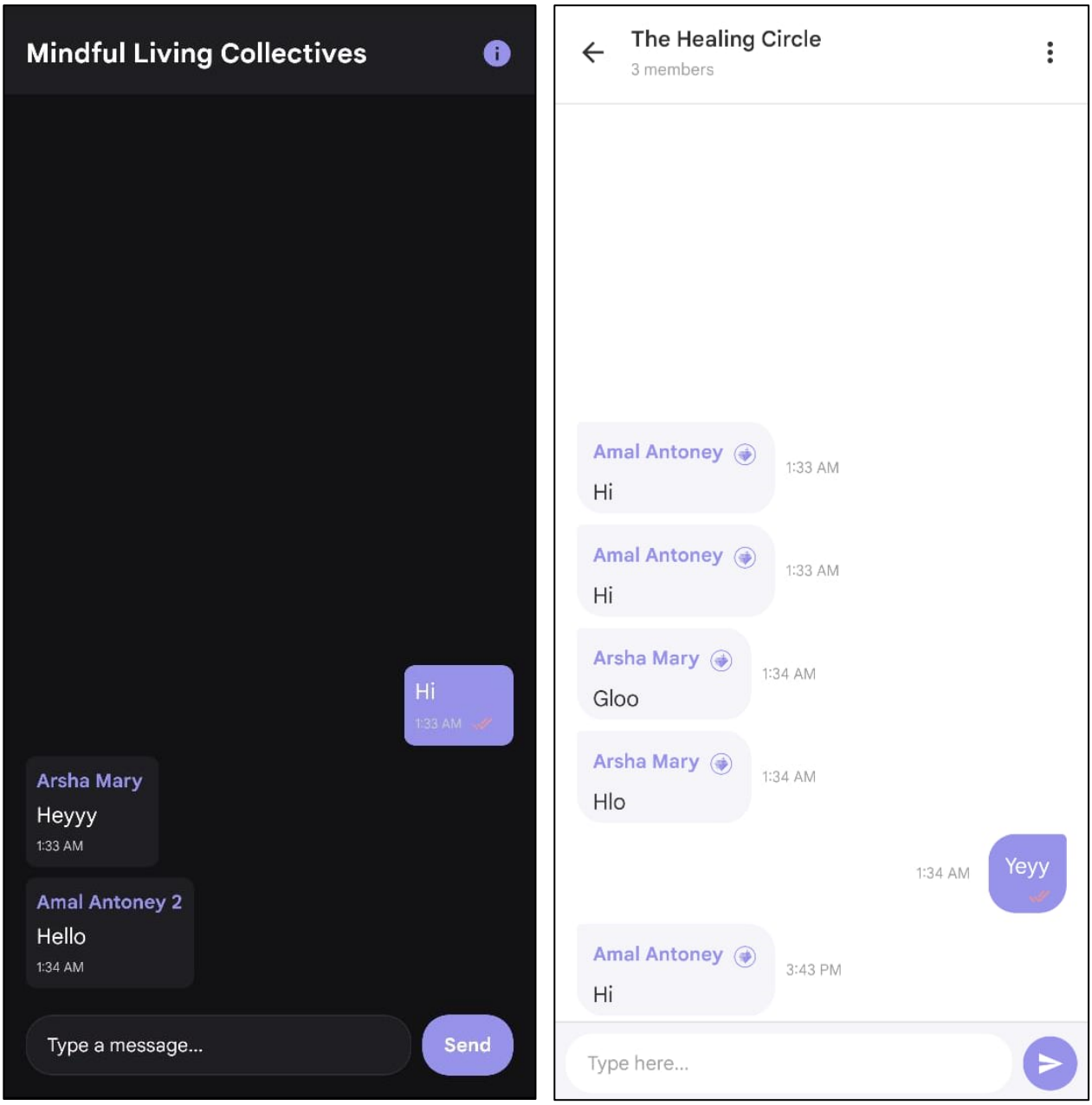


Figure 9.4

Mood Analytics

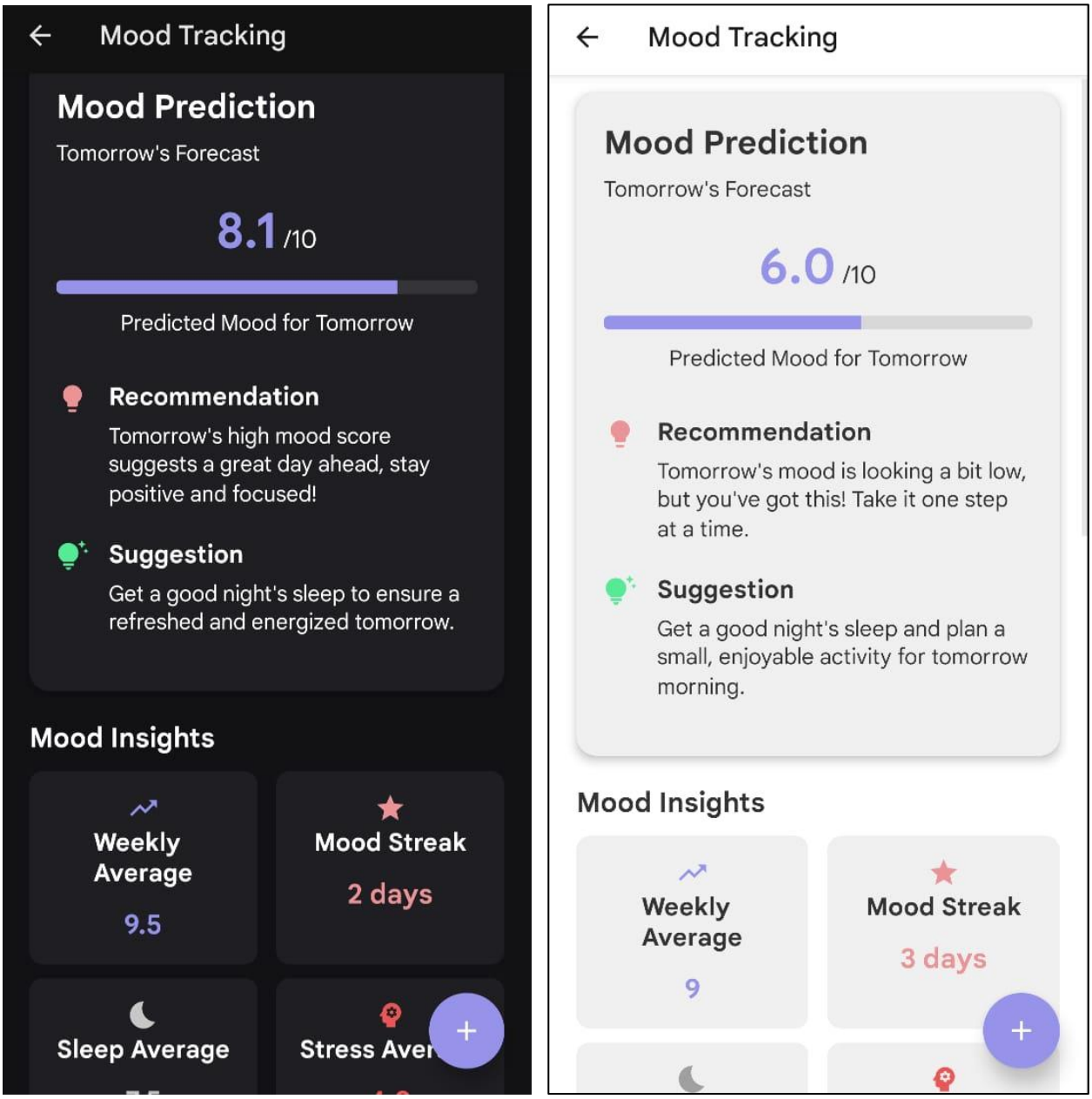


Figure 9.5

Admin Panel

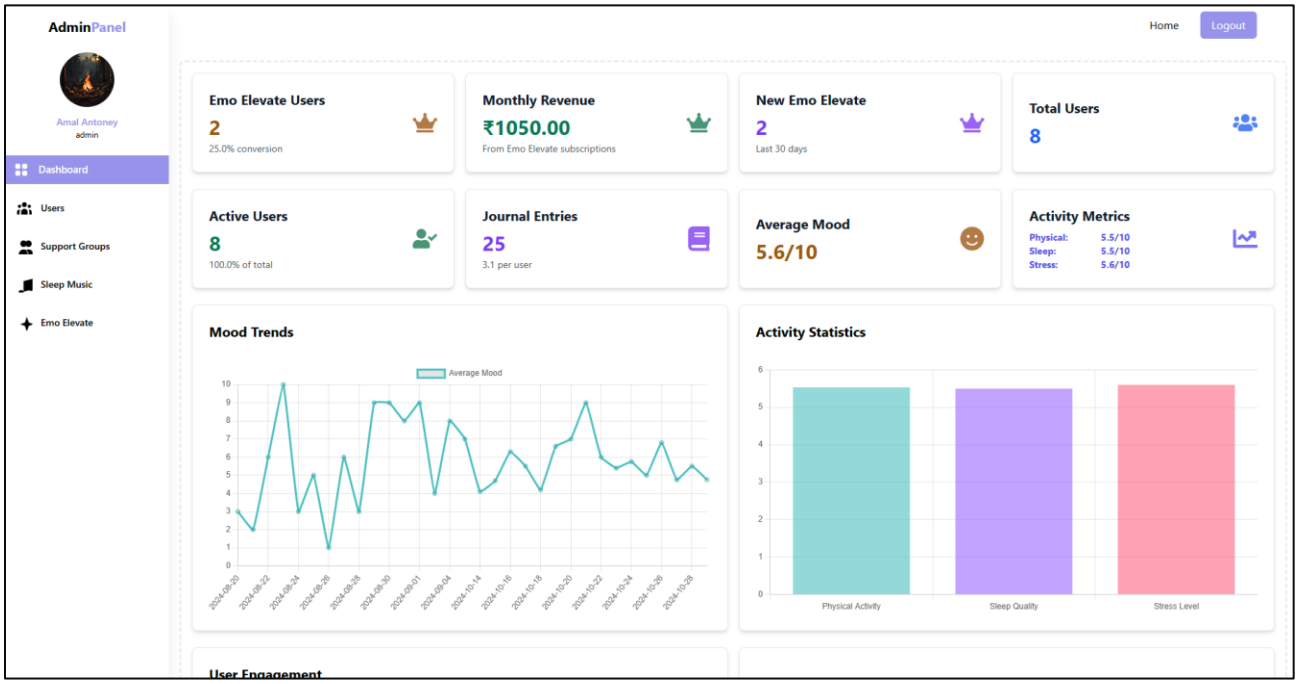


Figure 9.6

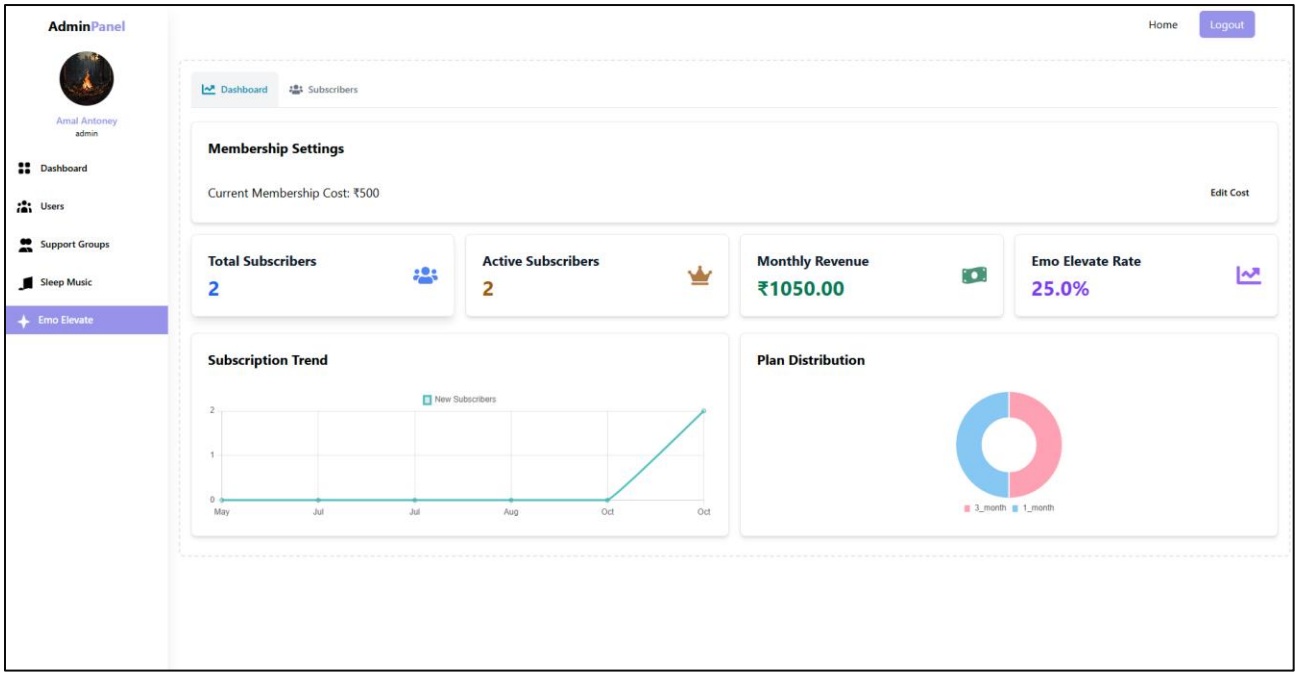


Figure 9.7

9.3 GIT LOG