MOODBOOST

by

Sami Khan
22341083
Arnab Das
19301029
Zarin Tasnim
19301119
MD Golam Shahnewaz
20101466
Soumitro Sharkar Debu
20301063

A thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science

Department of Computer Science and Engineering
Brac University
June 2025

Declaration

It is hereby declared that

1. The thesis submitted is my/our own original work while completing degree at Brac University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

Student's Full Name & Signature:

Sami Khan
22341083

Arnab Das
19301029

Zarin Tasnim
19301119

MD Golam Shahnewaz
20101466

Soumitro Sharkar Debu
20301063

Approval

The project titled "MOODBOOST" submitted by

1. Sami Khan (22341083)
2. Arnab Das (19301029)
3. Zarin Tasnim (19301119)
4. MD Golam Shahnewaz (20101466)
5. Soumitro Sharkar Debu (20301063)

Of Spring, 2025 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science on June 20, 2025.

Examining Committee:

Supervisor:
(Member)

Dewan Ziaul Karim
Lecturer
Department of Computer Science and Engineering
BRAC University

Co-Supervisor:
(Member)

Rafeed Rahman
Lecturer
Department of Computer Science and Engineering
BRAC University

Program Coordinator:
(Member)

Md. Golam Rabiul Alam
Professor
Department of Computer Science and Engineering
BRAC University

Head of Department:
(Chair)

Sadia Hamid Kazi
Chairperson and Associate Professor
Department of Computer Science and Engineering
BRAC University

Ethics Statement

This research involves the development of MoodBoost, an application designed to enhance mental well-being through personalized affirmations and motivational content. The ethical considerations for this project are outlined as follows:

0.1 Informed Consent

Prior to using the MoodBoost application, users will be informed about the nature and purpose of the app, the functionalities it offers, and how their data will be utilized. Users will provide explicit consent to participate in this project, ensuring they understand their right to withdraw at any time without any consequences.

0.2 Data Privacy and Confidentiality

The MoodBoost application will prioritize user privacy and data security. All personal information collected will be anonymized and securely stored, ensuring that individual identities remain confidential. Data will only be used for the intended purposes of improving the application and enhancing user experience.

0.3 AI and User Interaction

The use of artificial intelligence in MoodBoost requires careful consideration of ethical implications. The app will analyze user interactions to deliver tailored content while ensuring that this process respects user autonomy and emotional well-being. Users will be made aware of how AI is utilized to customize their experience.

0.4 Compliance with Ethical Guidelines

This project adheres to the ethical standards established by Brac University and complies with applicable laws and regulations regarding data protection and user rights.

0.5 Integrity of Research

The authors commit to the integrity of the research process, ensuring that all findings related to the efficacy of MoodBoost are reported accurately and transparently. There will be no fabrication or manipulation of data, and all sources of information will be properly cited to uphold academic honesty.

0.6 Potential Conflicts of Interest

The authors declare that there are no conflicts of interest in the development and research of the MoodBoost application.

Through the adherence to these ethical principles, this project aims to contribute meaningfully to the field of mental health technology, ensuring that users receive a safe, effective, and personalized mental wellness experience.

Abstract

An AI-Powered App for the Future. An App For Mental Health — Revolutionizing How we approach Mental Wellness. What is MoodBoost? MoodBoost is an AI-Powered application designed to revolutionize the mental health space by providing personalized, real-time interventions based on user emotional needs. In a time of mental health crises, which affect over 970 million people across the globe (WHO, 2022) with problems like stress, anxiety, and depression, current solutions provide static content and generalized recommendations that are often not enough. MoodBoost fills this gap by weaving in Gemini's state-of-the-art Natural Language Processing (NLP)—which analyzes user inputs in real time—to allow the app to create contextually relevant affirmations, actionable recommendations, and therapeutic content. The app's central innovation is its ability to dynamically adjust to users' emotional states. For instance, a user who logs a "stressed" mood may receive a curated meditation guide, while one who feels "motivated" could be offered suggestions for goal-setting activities. This level of personalization is enabled by a sophisticated technical architecture like Frontend Built with Flutter for cross-platform performance on Android and iOS, For Backend, Utilizes Django REST Framework to manage API logic, handle user authentication, and validate data. For Database, The Firebase is used for structured and scalable storage of user profiles, mood logs, and AI generated content. For AI Integration, We are using Gemini, it uses NLP models to process mood data for insights and affirmations — replies are relevant and empathetic. The app's approach integrates the latest advancements in AI with an emphasis on user-centered design, positioning it as a vital component in the landscape of contemporary mental health care, providing a forward-looking, data-guided experience with the potential to supplant traditional offerings.

Keywords: AI-Powered App, Mobile App, Mental Health, Mental Wellness, MoodBoost, Personalized Interventions, Real-time Interventions, Emotional Needs, Mental Health Crises, Stress, Anxiety, Depression, Static Content, Generalized Recommendations, DeepSeek, Natural Language Processing (NLP), User Inputs, Contextually Relevant Affirmations, Actionable Recommendations, Therapeutic Content, Flutter, Django REST Framework, AI Integration, User-centered Design.

Dedication

This project is dedicated to our family and friends, whose unwavering support and encouragement have been our guiding light throughout this journey. To our parents, whose love and sacrifices have laid the foundation for our education and personal growth. To our friends, who have been our constant companions and confidants. Your laughter and camaraderie have made the challenges of this journey more bearable and the successes even sweeter.Lastly, to those who strive for mental well-being and seek support through innovative technologies. May this work contribute to your journey towards a healthier, happier life.

Acknowledgement

Firstly, To our supervisor Mr.Dewan Ziaul Karim sir for his kind support and advice in our work. He helped us whenever we needed help.
And finally to our parents without their throughout support it may not have been possible. With their kind support we are now on the verge of our graduation.

Table of Contents

List of Figures

List of Tables

Nomenclature

Term	Definition
AI	Artificial Intelligence
API	Application Programming Interface
CBT	Cognitive Behavioral Therapy
CRUD	Create, Read, Update, Delete
CSS	Cascading Style Sheets
DB	Database
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
HTTPS	HyperText Transfer Protocol Secure
iOS	iPhone Operating System
JSON	JavaScript Object Notation
ML	Machine Learning
MVP	Minimum Viable Product
NLP	Natural Language Processing
NoSQL	Not Only Structured Query Language
PWA	Progressive Web App
REST	Representational State Transfer
SDK	Software Development Kit
SSL	Secure Sockets Layer
TLS	Transport Layer Security
UI	User Interface
URL	Uniform Resource Locator
UX	User Experience

Table 1: Nomenclature for Key Terms in the MoodBoost Project

This nomenclature provides a clear and concise reference for the key terms used throughout the thesis, ensuring consistency and clarity in the academic discussion of the MoodBoost project.

Chapter 1

Introduction

1.1 What is MoodBoost?

MoodBoost is an application designed to assist individuals in monitoring their emotional well-being and mental health. It serves as a personal mood journal that leverages advanced technology to provide insights and access to mental health resources. The app combines mood tracking with AI-powered insights to deliver a personalized mental wellness experience.

1.2 Who Can Benefit from MoodBoost?

MoodBoost is beneficial for various user groups:

- **Individuals managing mental health:** Those seeking to track and enhance their emotional well-being can utilize the app.
- **Therapy support:** It can aid individuals undergoing therapy by providing a tool for mood tracking between sessions.
- **Wellness-focused individuals:** People interested in emotional self-awareness and personal growth will find value in the app.
- **Stress and anxiety management:** The app offers tools to assist users in managing daily stress.

1.3 Key Benefits of MoodBoost

MoodBoost offers several advantages:

- **Mental Health Tracking:** Regular mood tracking fosters self-awareness, helps identify mood triggers and patterns through visualization, and allows for monitoring progress over time with analytics.
- **Personalized Support:** The app provides AI-powered guidance and contextual recommendations based on individual mood patterns. It also offers daily affirmations tailored to the current emotional state and adaptive content that adjusts to changing mental health needs.

- **Accessibility:** Most apps correlate mood data with actionable insights, leaving users without a means to track progress.
- **Privacy and Security:** Most apps correlate mood data with actionable insights, leaving users without a means to track progress.

1.4 What Is Seen and Done on the App

Upon opening MoodBoost, a clear and easy-to-use screen is presented. It features:

- **Navigation Bar:** At the bottom, there are five tabs: Home, Chat, Analytics, Library, and Profile, allowing for easy navigation.
- **Top Bar:** At the top, the app's title, theme toggle, settings, and logout buttons are visible.
- **Smooth Experience:** The app features smooth animations between screen transitions, enhancing the user experience.

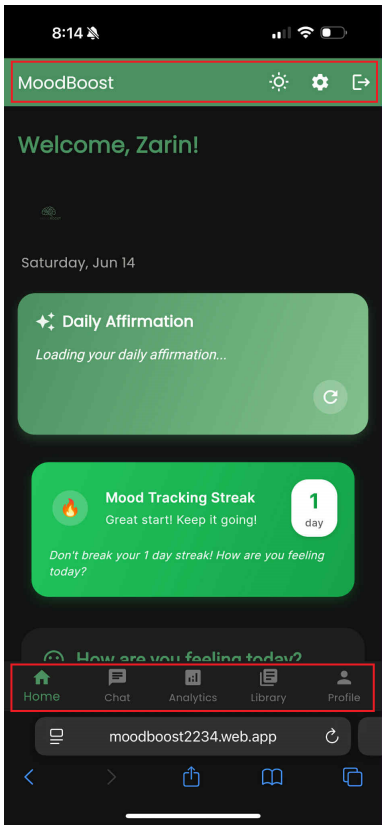

Figure 1.1: Interface Overview

On the main screen, the following components are found:

- **Welcome:** The app logo and a welcome message with the current date.
- **Daily Inspiration:** A card providing randomly generated positive messages.

- **Mood Tracker:** A mood selector with various emotional states (happy, sad, angry, etc.) and an emoji picker for expressive mood tracking are available
- **Journal:** A note input field for journaling feelings.
- **Recommendations:** Cards suggesting mood-boosting games (navigates to games matching the selected mood) and music suggestions (recommends music based on selected mood).

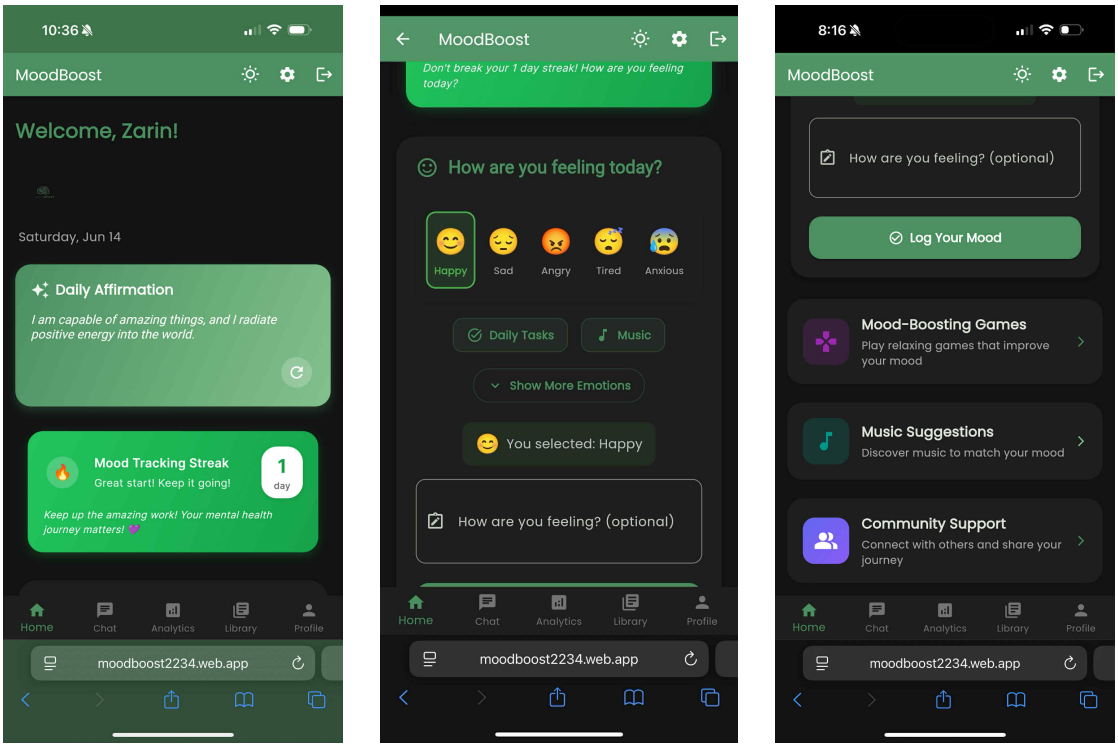

Figure 1.2: Main Screen Overview.

The app allows toggling between light and dark mode with animation and provides visual feedback via a snackbar when the theme changes. From the main screen, access to chat support, mood analytics visualization, wellness content library, user profile management, settings configuration, and authentication (logout functionality) is available.

Chapter 2

Frameworks and Technologies

2.1 Frontend Framework: Flutter

This is a cross-platform UI framework designed for building applications that are natively compiled. The application utilizes Material Design 3 components and is primarily developed using the Dart programming language.

2.2 Backend Framework: Django

This is a Python web framework that adheres to the model-view-template (MVT) architecture. It is employed to create a REST API for the Flutter application and is responsible for managing communication with the Google Gemini API.

2.3 AI Integration: Google Gemini 1.5 Pro

This advanced AI model is integrated for the purpose of generating mental health content, formulating chat responses, and providing personalized recommendations. Its integration is primarily handled through the Django backend.

2.4 Database and Authentication: Firebase

Firebase Authentication is a key component utilized for secure user login and registration processes.Firestore serves as the primary cloud database for storing structured user data, while Firebase Storage is specifically designated for managing media files.

Chapter 3

MoodBoost Architecture and Code Workflow

3.1 Technology Stack

The development of MoodBoost incorporates a robust set of technologies:

- **Frontend:** The application is built using Flutter (SDK 3.0+) with Material Design 3. State management is handled by the Provider pattern for reactive updates, and custom animations are implemented using AnimationController and Tween. UI components are custom-themed widgets based on Material components.
- **Backend:** The server is powered by Django REST framework, which serves API endpoints. AI integration utilizes Google's Gemini API (version 1.5 Pro) for generating personalized content. SQLite is used for development databases, while Firebase Firestore is employed for production data.
- **Authentication & Storage:** Firebase Authentication is used for user authentication, specifically with Google Sign-In. Cloud storage for user data and media is managed by Firebase Storage. Firebase Firestore is also used for user mood entries and other structured data.
- **Cross-Platform Support:** The app supports web with custom Firebase web patches, mobile (Android and iOS), and desktop (macOS, Windows, Linux)

3.2 Core Architecture

The app adheres to a layered architecture:

- **Presentation Layer:** This layer encompasses screens and widgets, located in the lib/screens/ and lib/widgets/ directories.
- **Service Layer:** Services in the lib/services/ directory manage business logic.
- **Data Layer:** Models in lib/models/ define data structures.
- **External Services:** This layer handles integrations with Firebase and the backend API.

3.3 State Management

Various components manage the application's state:

- **AuthInterface:** An abstract class providing authentication methods.
- **AuthService:** A concrete implementation utilizing Firebase Authentication.
- **ThemeProvider:** Manages the app's theme state, such as light/dark mode.
- **MoodService:** Handles the storage and retrieval of mood data.

3.4 Navigation Flow

The application's navigation sequence is as follows:

- App entry at main.dart initializes Firebase and providers.
- **SplashScreen:** The initial loading screen checks the authentication state.
- **AuthScreen:** If not authenticated, this screen displays login options.
- **HomeScreen:** The main screen accessed after authentication, featuring bottom navigation.
- Specialized screens for different features like Analytics, Chat, Content, and Profile are also available.

3.5 Interoperability of Components

Flutter-Django Communication

The Flutter application initiates HTTP requests to the Django backend utilizing the `http` package. For instance, a typical request to the backend for an affirmation based on a user's mood might be structured as follows:

```
final response = await http.post(
  Uri.parse('$_baseUrl/affirmation/'),
  headers: {'Content-Type': 'application/json'},
  body: jsonEncode({'mood': _moodController.text}),
);
```

Listing 3.1: Flutter HTTP POST Request

Django receives these incoming requests, processes them through its defined views, and subsequently returns JSON responses to the Flutter application.

Django-Gemini Integration

The Django backend serves as the intermediary facilitating communication with Google's Gemini API. For example, within the affirmation view on the backend, the Gemini model is explicitly utilized to generate the required textual content:

```
model = genai.GenerativeModel('gemini-1.5-pro')
response = model.generate_content(prompt)
affirmation_text = response.text.strip().strip('"')
```

Listing 3.2: Django Calling Gemini API

3.6 Architectural Advantages

This particular architectural design provides several significant and distinct advantages for the MoodBoost application:

- **Separation of concerns:** The Flutter framework is exclusively dedicated to managing the user interface, while the Django framework solely handles the underlying business logic, promoting modularity and maintainability.
- **AI capabilities:** The direct and seamless integration with Gemini enables the implementation of highly sophisticated mental health features, enhancing the application's core functionality.
- **Scalability:** The backend component possesses the inherent capability to be deployed and scaled independently from the mobile application, allowing for efficient resource management as user demand grows.
- **Cross-platform functionality:** The Flutter application demonstrates remarkable versatility by operating consistently and effectively across various platforms, including Android, iOS, web, and desktop environments.

Chapter 4

Game and its Mechanics, Algorithms, and Data Structures

4.1 Games

The MoodBoost application incorporates a diverse set of interactive games crafted to engage users and elevate their emotional well-being. Each game is thoughtfully designed with intuitive mechanics and vibrant visuals, ensuring an enjoyable experience that encourages repeated play while fostering a positive mood through immersive and entertaining interactions.

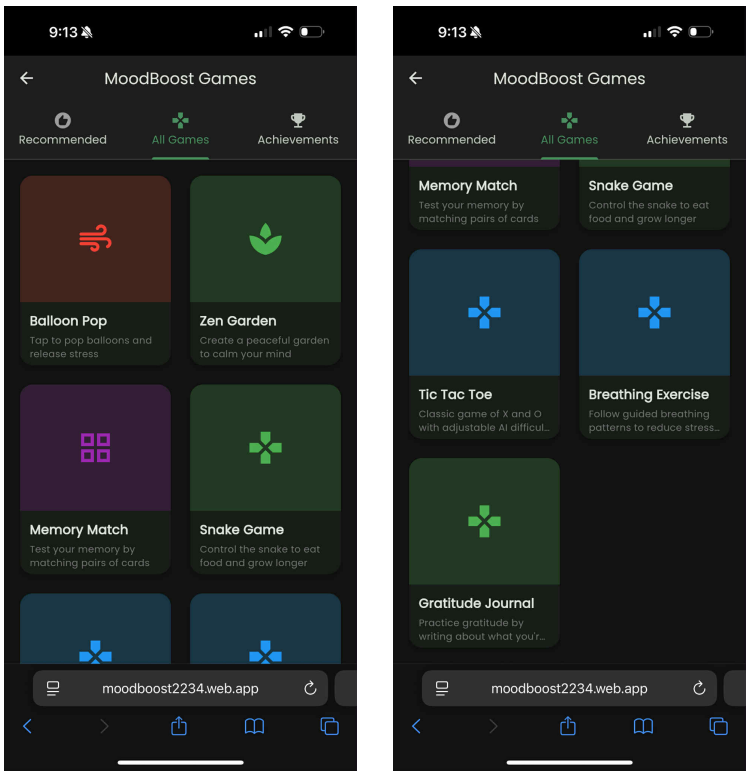

Figure 4.1: Games Overview.

4.2 Game Features and Algorithms

MoodBoost's interactive experience is powered by a suite of cutting-edge algorithms and visionary design, crafted to ignite inspiration and captivate players. Harnessing optimized data structures and sophisticated computational techniques, these systems dynamically orchestrate game states, respond to user inputs, and calibrate challenges in real time. AI-driven progression adapts to player performance, ensuring an evolving experience that sparks engagement across skill levels. Responsive multisensory feedback, including physics-based motion and vibrant visual effects, creates an immersive atmosphere that delights the senses. Intelligent decision-making, powered by advanced probabilistic strategies, delivers tailored challenges for both novices and experts. Procedural content generation crafts unique configurations for every session, keeping experiences fresh and unpredictable. Efficient state management through streamlined data frameworks ensures seamless tracking of interactions and progress, while cognitive and emotional enhancement tools foster mindfulness and strategic thinking, empowering players with transformative experiences that inspire joy and resilience.

Snake Game

Snake is redesigned to focus on basic mechanics and difficulty that gets harder over time.

- **Snake Body Representation:** Snake bodies are stored in a `List<Point<int>>` that works as a queue. The snake uses this data structure well, since segments are easily added at the start (forward movement) and removed at the end (retreating).
- **Collision Detection:** Sophisticated algorithms oversee the snake's location, noticing if the snake touches the edges or itself. These necessitate players to know when their game turns and if they are done.
- **Food Generation:** The game ensures food is placed randomly in a free cell which gives the game new challenges each time.
- **Difficulty Scaling:** As the player scores, a time-based algorithm raises the snake's speed to bring more challenges and make the game more engaging.

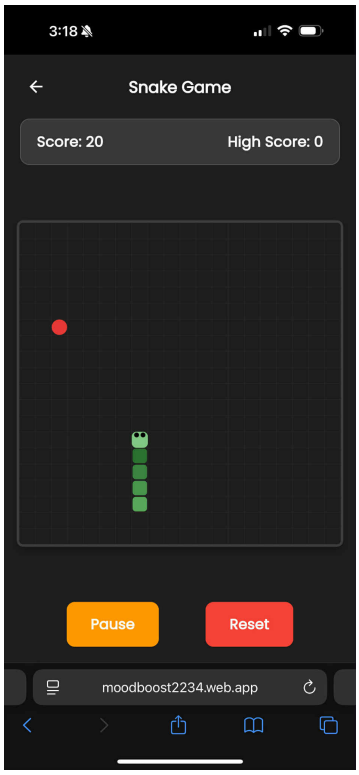

Figure 4.2: Snake Game

Tic Tac Toe Game

The game comes with an AI that you can set to different skill levels, so it is fun for new players and very tough for advanced ones.

- **AI Algorithm:** The game uses a version of minimax to keep the AI brain intact, with configuration for three different challenge levels.
  - **Easy:** Choosing movements at random using uniform distribution is a good pick for those who are just starting out.
  - **Medium:** A hybrid strategy combines 60% smart moves (e.g., blocking or winning opportunities) with 40% random choices, offering a balanced challenge.
  - **Hard:** The AI is guided by a decision tree that looks at different moves in this way: securing a win; blocking the opponent from winning; taking the center; claiming corners; and, if none of these are possible, picking randomly among remaining spaces. That makes things tough for the opponents.
- **Board Representation:** The game uses a 2D array (`List<List<int>>`) to model the 3×3 grid, a standard choice for such games.
- **Win Detection:** Having an algorithm in place, the game scans the rows, columns and diagonals to see if anyone has won or whether the result is a draw.

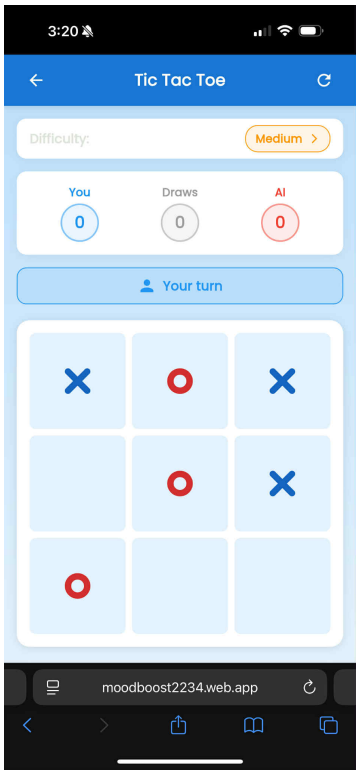

Figure 4.3: Tic Tac Toe

Memory Match Game

It strengthens your memory by having you match cards.

- **Card Shuffling:** In the Fisher-Yates shuffle, the positions of the cards are randomly rearranged to provide fairness in each turn.
- **Pair Matching:** An algorithm tracks card states (flipped or unflipped) and confirms matches, driving the game's core mechanic.
- **Scoring System:** Scoring is set up so it rewards people who are careful and quick, encouraging them to do well and quickly.

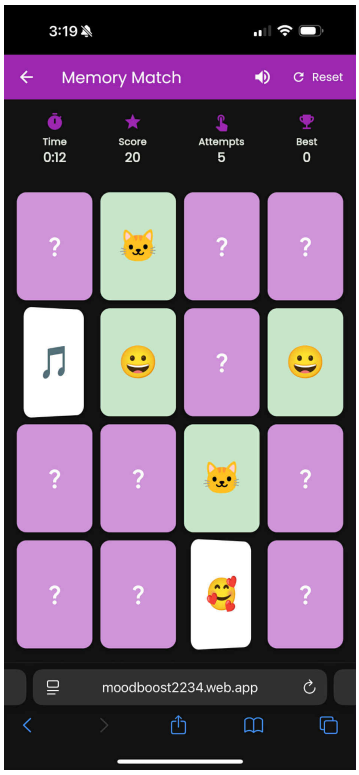

Figure 4.4: Memory Matching

Balloon Pop Game

An arcade game where you can pop balloons that move around, made even better by bright, colorful graphics.

- **Procedural Generation:** With procedural generation, balloons are designed in a variety of sizes and colors, making everything more interesting.
- **Physics-Based Movement:** The AI algorithms develop a movement for the balloons that makes them describe sinusoidal curves, just like how real balloons usually move.
- **Explosion Effects:** Thanks to the algorithm, popping the balloons produces impressive explosions which delights the senses.

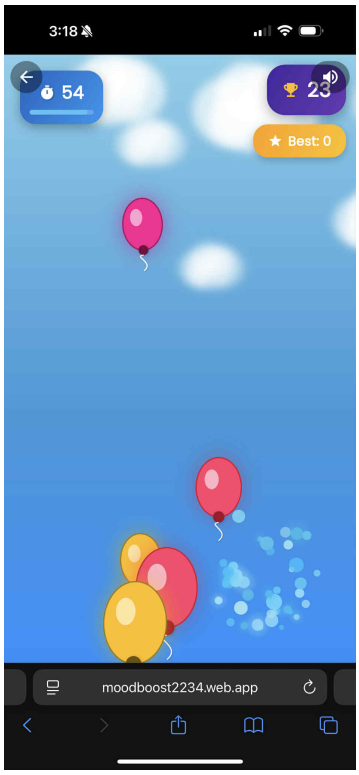

Figure 4.5: Ballon Pop

Gratitude Journal

A reflective exercise to enhance mindfulness and positivity.

- **Prompt System:** Provides daily prompts to encourage thoughtful responses, such as identifying things taken for granted.
- **Time Tracking:** Records the time spent on each entry to promote consistent practice.
- **Submission Process:** Allows users to submit responses, tracking progress through multiple entries.

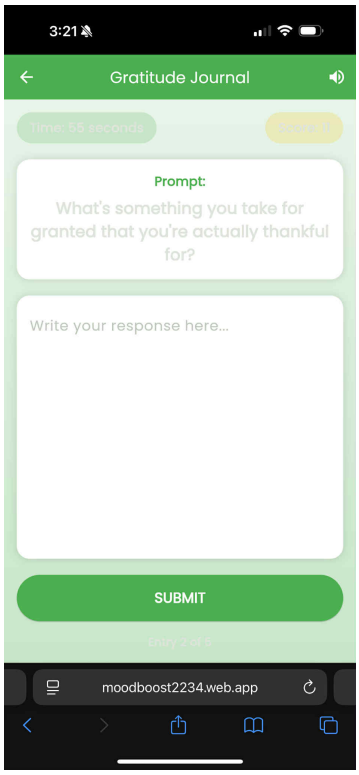

Figure 4.6: Gratitude Journal

Zen Garden

A calming activity to promote relaxation and creativity.

- **Element Placement:** Features a variety of colorful elements (trees, flowers, water) that users can arrange freely.
- **Random Distribution:** Elements are scattered randomly across the screen for a unique experience each time.
- **Customization Options:** Offers tools to add or remove elements, enhancing user engagement.

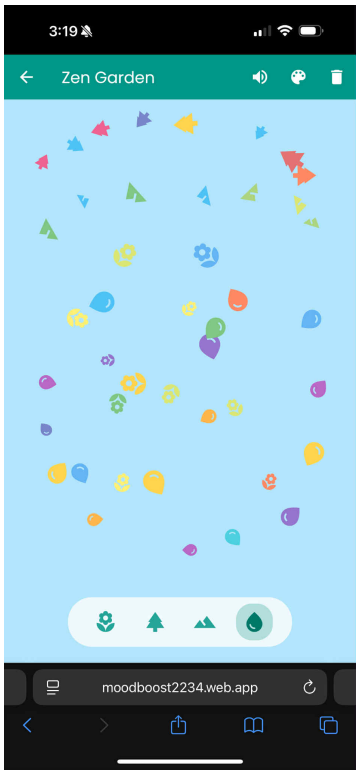

Figure 4.7: Zen Garden

Breathing Exercise

A guided session to improve focus and reduce stress through controlled breathing.

- **Timer Mechanism:** Displays a countdown to guide the duration of each breathing phase (e.g., hold, inhale, exhale).
- **Instruction Display:** Shows clear instructions like "Hold" with a timer to assist users.
- **Session Control:** Includes an option to end the session, allowing flexibility in practice duration.

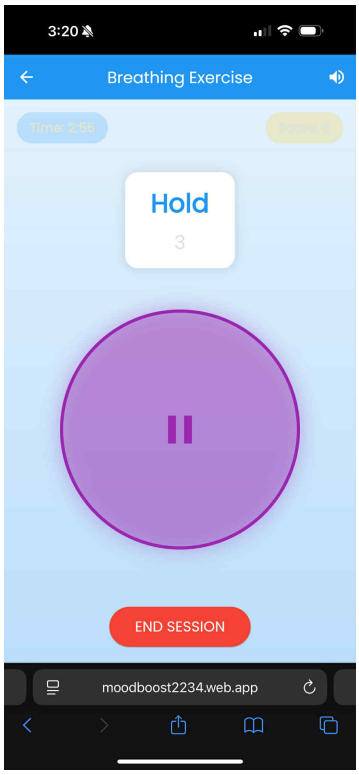

Figure 4.8: Breathing Exercise

4.3 Analytics and Data Processing

Along with entertainment, MoodBoost gives users insights into how they feel in certain situations using complex data and analysis.

Mood Analysis

- **Time-Series Processing:** Mood entries are sorted by date so users can follow their mood day by day.
- **Frequency Analysis:** This is use to highlight the emotions that appear most often in the music.
- **Weighted Scoring:** Moods are rated between numerical values meaning how significant a mood is can be represented by its score.
- **Visualization:** Graph algorithms let you create simple graphs, like line and pie charts which illustrate your mood from one period to another.

Recommendation System

- **Mood-Based Filtering:** Content is shown based on the user's current emotions, so it relates to their feelings and helps.
- **Category Diversification:** The content selected includes a variety which keeps the user from getting bored and allows for a more varied experience.

- **Novelty Enhancement:** By using a novelty algorithm, a recommendation system keeps the content interesting by ensuring it is not shown many times.

4.4 AI Integration

Personalized MoodBoost with artificial intelligence which made it easier for users to communicate and receive encouragement.

Chat System

- Using Prompt Engineering, algorithms support the AI in using appropriate responses depending on what the user says.
- Formatting in history means a special algorithm records all dialogue, so replies reflect the context.
- AI results are formatted in a way that makes them clearer which helps the user experience.

Affirmation Generator

- **Template Selection:** Picks the best affirmations for the user by checking their emotions with a moodbased process.
- **Fallback Mechanism:** If the API cannot supply information, a fallback method makes sure the app can still advise, keeping its supportive purpose.

4.5 Data Structures

App efficiency and organization rely on a set of data structures built for its purpose.

Collections

- **Lists:** Used to keep items sorted such as games or suggestions.
- **Maps:** Using keys and values, maps can make sure specific choices are made for moods or with scores.
- **Sets:** Ensure that recommendations from a set are completely different from the rest.
- **Queues:** Queues deal with ordered tasks such as controlling the snake body or playing animation sequences.

Custom Data Models

- **MoodEntry:** A MoodEntry holds the correct times and information for mood data which is crucial for analytics.
- **Game:** Organizes all game details and makes it easier to run the game.

- **GardenElement:** Representing things in the zen garden gives the app a more calming look.
- **Direction Enums:** Create Direction Enums for Snake or other games, to help organize code related to movement.

4.6 Algorithms in Backend

MoodBoost's backend manages the system and data well which helps its front-end to work smoothly.

API Processing

- **JSON Handling:** JSON Parsing and formatting help the app communicate smoothly with services outside itself.
- **Pattern Matching:** Using regular expressions, pattern matching can find and organize data in AI answers which is fundamental to natural language processing(NLP)
- **GardenElement:** Representing things in the zen garden gives the app a more calming look.
- **Error Handling:** Algorithms with error handling allow a clear user experience even if there are unexpected problems.

A well-designed combination of algorithms, data structures and AI-based logic makes the MoodBoost app enjoyable and helpful. Thanks to the combination of game mechanics, data and informed recommendations, MoodBoost allows users to track their emotions and improve them whenever they want.

Chapter 5

Core App Features and Implementation

5.1 Core App Features Overview

MoodBoost is an application designed to assist users in managing their moods, engaging with a supportive AI, receiving personalized suggestions, and utilizing helpful well-being resources. It uses modern cloud platforms, AI, and a Flutter frontend to ensure a responsive and scalable design.

Core App Features

- **Mood Logging:** Users can express their daily moods using text or emojis. Weekly insights help track emotional trends.
- **AI Chatbot:** An assistant powered by Gemini AI provides support and mental health tips.
- **AI-Generated Daily Affirmations:** Users receive AI-created affirmations daily.
- **Curated Content Library:** AI matches content to user emotions and recent actions for personalized encouragement.
- **Personalized Recommendations:** Content is tailored to user preferences and data.
- **User Feedback System:** Feedback is collected to enhance user experience.

5.2 Authentication

The authentication system employs Firebase Auth with Google Sign-In integration.

- It provides cross-platform authentication, including web-specific patches.
- The system maintains a persistent login state across sessions.
- Proper sign-out functionality with navigation redirection is included.

- The authentication flow in AuthService involves triggering the Google Sign-In process. Authentication details are obtained from Google Sign-In. A Firebase credential is then created. Finally, the system signs in with Firebase using this credential and returns the user credential.

```
// Authentication flow in AuthService
Future<UserCredential?> signInWithGoogle() async {
  // Trigger Google Sign-In flow
  final GoogleSignInAccount? googleUser = await _googleSignIn.
      signIn();

  // Get authentication details from Google Sign-In
  final GoogleSignInAuthentication googleAuth = await googleUser.
      authentication;

  // Create Firebase credential
  final credential = GoogleAuthProvider.credential(
    accessToken: googleAuth.accessToken,
    idToken: googleAuth.idToken,
  );

  // Sign in with Firebase and return user credential
  return await _auth.signInWithCredential(credential);
}
```

Listing 5.1: Authentication flow in AuthService

5.3 Mood Tracking and Logging

The mood tracking system enables individuals to:

- Select from predefined emojis or choose custom emoji moods.
- Add optional notes to their mood entries.
- Record entries with server timestamps for accurate tracking.
- View their mood history with timestamps.
- The UI for mood selection offers an interactive, animated experience with haptic feedback and visual confirmation of selection.
- Adding a mood entry in MoodService involves checking if a user is logged in; if not, an exception is thrown. A MoodEntry object is created with the user's ID, mood, note, and a server timestamp. This entry is then added to the 'moods' collection in Firestore.

```
// Adding a mood entry in MoodService
Future<void> addMoodEntry(String mood, {String? note}) async {
  final user = _authService.currentUser;
  if (user == null) throw Exception('User not logged in');

  final entry = MoodEntry(
```

```
    userId: user.uid,
    mood: mood,
    note: note,
    timestamp: FieldValue.serverTimestamp(),
  );

  await _firestore.collection('moods').add(entry.toMap());
}
```

Listing 5.2: Adding a mood entry in MoodService

5.4 Analytics and Visualization

The analytics feature utilizes:

- Real-time data streaming from Firestore.
- Chart visualization facilitated by the fl_chart package.
- Time-based filtering and aggregation.
- Pattern detection for mood trend insights.
- The UI presents data in visually appealing charts with interactive elements and helpful insights based on the user's mood history.
- The stream of weekly mood entries for analytics involves checking if a user is logged in; if not, an exception is thrown. The system calculates the date one week ago. It then queries the 'moods' collection in Firestore, filtering by user ID and timestamps greater than or equal to one week ago, and orders results by timestamp in descending order. This snapshot is then mapped to a list of MoodEntry objects.

```
// Stream of weekly mood entries for analytics
Stream<List<MoodEntry>> getWeeklyMoodEntries() {
  final user = _authService.currentUser;
  if (user == null) throw Exception('User not logged in');

  final oneWeekAgo = DateTime.now().subtract(const Duration(days:
     7));

  return _firestore
    .collection('moods')
    .where('userId', isEqualTo: user.uid)
    .where('timestamp', isGreaterThanOrEqualTo: Timestamp.fromDate(
       oneWeekAgo))
    .orderBy('timestamp', descending: true)
    .snapshots()
    .map((snapshot) {
      return snapshot.docs.map((doc) =>
        MoodEntry.fromFirestore(doc.id, doc.data())).toList();
    });
}
```

Listing 5.3: Stream of weekly mood entries for analytics

5.5 Content Library

The content library provides:

- Personalized wellness resources based on mood patterns.
- Categorized content, including articles, videos, and exercises.
- YouTube video integration.
- External link handling using `url_launcher`.
- Recommendations are generated by the Gemini API based on the current mood state, historical mood patterns, and evidence-based mental health resources.

5.6 Theme Support and UI/UX

The application incorporates various UI/UX features:

- Light and dark mode support with persistent user preference.
- Custom color schemes aligned with the app's branding.
- Animated transitions between screens and states.
- Consistent typography using the Poppins font family.
- Haptic feedback for interactive elements.
- Responsive design for different screen sizes.

Theme initialization and switching involve building a `ThemeData` object based on a `ColorScheme`, enabling Material 3, and setting the font family to 'Poppins'. The `toggleTheme()` function switches between light and dark mode, saves the preference, and notifies listeners of the change.

```
// Theme initialization and switching
ThemeData _buildTheme(ColorScheme colorScheme) {
  return ThemeData(
    colorScheme: colorScheme,
    useMaterial3: true,
    fontFamily: 'Poppins',
    // Additional theme configurations...
  );
}

void toggleTheme() {
  _themeMode = _themeMode == ThemeMode.light ? ThemeMode.dark :
      ThemeMode.light;
  _saveThemePreference();
  notifyListeners();
}
```

Listing 5.4: Theme initialization and switching

Chapter 6

How AI Helps in MoodBoost

6.1 AI Integration Overview

The MoodBoost app integrates Google's Gemini 1.5 Pro AI model to provide several key mental health and wellness features.

6.2 The AI Chat Companion

The app features a dedicated chat interface where interactions with an AI mental health companion can occur.

- **Empathetic AI:** The AI is configured with a specific persona as an "empathetic mental health companion" that uses cognitive behavioral therapy (CBT) techniques.
- **Contextual Conversations:** The system maintains conversation history to provide contextually relevant responses.
- **Conversation Starters:** Pre-defined suggested prompts help users start conversations (e.g., "I'm feeling anxious today")

6.3 AI-Powered Insights with Gemini API

The application integrates Google's Gemini 1.5 Pro AI model to deliver comprehensive mental health support through multiple intelligent features:

- Daily personalized affirmations based on the current mood
- AI-powered chat support for mental health assistance
- Personalized content recommendations derived from mood patterns
- The backend implementation for affirmation generation involves a POST request that receives the user's mood. A prompt is then formulated to generate a short, positive, uplifting affirmation for someone feeling that mood. The gemini-1.5-pro model is used to generate content based on this prompt, and the resulting affirmation text is returned. The client-side implementation in

GeminiService handles API communication and includes robust fallback mechanisms if the API is unavailable.

```python
# Backend implementation for affirmation generation
@api_view(['POST'])
def affirmation(request):
    """Generate a positive affirmation based on mood"""
    mood = request.data.get('mood', '')  # Default to happy if no mood is provided
    prompt = f"Generate a short, positive, uplifting affirmation for someone feeling {mood}."
    model = genai.GenerativeModel('gemini-1.5-pro')
    response = model.generate_content(prompt)
    affirmation_text = response.text.strip()
    return Response({"affirmation": affirmation_text})
```

Listing 6.1: Backend implementation for affirmation generation

6.4 Support Chat with AI

The chat functionality leverages the Gemini API to offer comprehensive mental health support:

- Contextual responses based on the conversation history
- Mental health support incorporating CBT (Cognitive Behavioral Therapy) techniques
- Safety mechanisms for critical situations
- Persistent chat history
- The client-side implementation for chat sends a POST request to the chat API endpoint, including the message and chat history. If the response status code is 200, the response data is decoded, and the AI's response is returned. Otherwise, a fallback message indicating connection issues is returned.

```dart
// Client-side implementation for chat
Future<String> sendChatMessage(String message, List<Map<String, String>> chatHistory) async {
    final response = await http.post(
        Uri.parse('$_baseUrl/chat/'),
        headers: {'Content-Type': 'application/json'},
        body: jsonEncode({
            'message': message,
            'chat_history': chatHistory,
        }),
    );

    if (response.statusCode == 200) {
        final data = jsonDecode(response.body);
        return data['response'] as String;
    } else {
```

```
16          return 'Sorry, I\'m having trouble connecting right now.
                Please try again later.';
17      }
18  }
```

Listing 6.2: Client-side implementation for chat

6.5 Personalized Positive Messages

The app generates customized positive affirmations through intelligent mood analysis:

- **Mood-Based:** When users input their mood (via text or emoji), the AI creates uplifting, supportive statements
- **Always Available:** Fallback affirmations are provided locally if the API connection fails

6.6 Content Tailored for Individuals

Based on the current and recent moods, the AI suggests personalized wellness content:

- **Tailored Suggestions:** Recommendations include articles, videos, exercises, meditations, and podcasts, each with a title, content type, and category

6.7 Behind the Scenes: AI Implementation

The Flutter frontend communicates with a Django backend API to provide seamless AI integration:

- **Backend Operations:** The backend uses Google's Gemini API with carefully crafted prompts for each feature
- **System Prompts:** Ensure the AI provides evidence-based support while avoiding medical diagnosis
- **Response Formatting:** Keeps AI outputs concise and supportive (typically 2-3 sentences)
- **Security:** All AI communications are secured with proper API authentication

6.8 Offline Functionality

The app includes comprehensive local fallback options when the AI service is unavailable:

- **Backup Messages:** Pre-defined affirmations based on mood categories ensure users always receive support

- **Cached Recommendations:** Cached recommendations provide value even without an active connection

Chapter 7

MoodBoost Backend

7.1 Backend Overview

The MoodBoost app backend is coded in the Django REST framework and acts as the core logic middle tier between Firebase, AI services and the Flutter frontend. The backend processes user requests, talks to Google's Gemini 1.5 pro AI for mental health and securely stores and authenticates user data using Firebase.

7.2 Key AI-powered Endpoints

The backend includes three main AI-driven endpoints. The affirmation endpoint accepts a user's mood input and generates personalized supportive affirmations. The chat endpoint allows users to engage with a virtual mental wellbeing companion, replying with empathic and motivational responses without delivering clinical advice. The recommendations endpoint examines current and historical mood data to return personalized wellness content such as articles, exercises or meditations.

7.3 API Key Management and Security

To prevent unsafe integration with Gemini, the backend loads the API key from environment variables. For tests, a dedicated endpoint provides temporary API key configuration, with input validation to ensure key match Google's prescribed format. This setup maintains protection for the system while enabling flexibility for development and testing.

7.4 Endpoint Implementation and AI Integration

Every endpoint is built Django REST framework decorators and is compatible with JSON requests and responses. The backend interprets user input, constructs Gemini queries, calls the AI model and decodes the output passed back. Responses are carefully formatted before being sent back to the frontend with error checking for dependability in case the AI service becomes unavailable.

7.5 Frontend-Backend Communication

Communication between Django backend and frontend Flutter app is achieved via HTTP requests. The frontend sends data like mood or conversation history and the backend processes it, sending structured JSON responses back. This separation of concerns allows the frontend to focus on user experience and backend to deal with business logic and AI interaction.

7.6 Scalability and Maintainability

This design is a scalable, maintainable and flexible foundation for MoodBoost. With UI issues separated from backend processing and AI integrating, the system can easily accommodate future feature additions, updates and cross-platform deployment to deliver an immediate and user-friendly app experience.

7.7 Django REST Framework API

The backend provides several API endpoints:

- `/api/affirmation/`: Generates mood-based affirmations.
- `/api/chat/`: Facilitates AI-powered conversations.
- `/api/recommendations/`: Delivers personalized content suggestions.

7.8 Gemini API Integration

The backend's `GeminiClient` is responsible for interacting with the Gemini API:

- It handles error handling with sensible fallbacks.
- Rate limiting protection is implemented.
- Response parsing and formatting are managed.
- Prompt engineering is employed for optimal AI responses.
- The `_call_gemini_api` method constructs a request body with the prompt and generation configuration, including temperature, topP, and topK values, and then sends a POST request to the Gemini API.

```
class GeminiClient:
    def __init__(self):
        self.api_key = settings.GEMINI_API_KEY
        self.model = "gemini-1.5-pro"

    def _call_gemini_api(self, prompt, temperature=0.4, max_tokens=None):
        url = f"{self.base_url}/{self.model}:generateContent?key={self.api_key}"
```

```
        request_body = {
            "contents": [{"parts": [{"text": prompt}]}],
            "generationConfig": {
                "temperature": temperature,
                "topP": 0.8,
                "topK": 40
            }
        }
        response = requests.post(url, json=request_body)
        # Process response and return text
```

Listing 7.1: GeminiClient Implementation

7.9 Deployment Infrastructure

The deployment infrastructure includes:

- **Web Setup**: Firebase Hosting capability and CORS configuration for API access are provided. Web-specific JavaScript patches are used for Firebase compatibility.
- **Development and Production Environments**: The setup includes a local development proxy server, environment variable management, and API key security measures.

7.10 Backend Structure Details

The backend system is organized within a Django project named `moodboost_api`. It comprises a dedicated Django application for AI-related requests, which is named `gemini_api`. RESTful API endpoints are systematically defined in `urls.py`, while API views located in `views.py` are responsible for processing incoming requests. The core Gemini API client implementation resides within `gemini_client.py`.

Chapter 8

The MoodBoost API

8.1 API Overview

The MoodBoost API on Render works as a backend service that connects the Flutter frontend app with Google's Gemini AI model.

8.2 API Deployment

The backend is deployed on Render.com as a web service named "moodboost-api". It uses a Python environment with the Django REST framework. Critical environment variables like `GEMINI_API_KEY` and `SECRET_KEY` are configured in Render's dashboard. The service is built using `pip install -r requirements.txt` and started with `gunicorn moodboost_api.wsgi:application`.

8.3 API Endpoint Structure

The main API is accessible at https://moodboostapi.onrender.com/api/. Key endpoints include:

- `/api/status/` – Health check endpoint
- `/api/affirmation/` – Generates mood-based affirmations
- `/api/chat/` – Provides AI chatbot functionality
- `/api/recommendations/` – Delivers personalized content suggestions

The Flutter app connects to the API using the GeminiService class. The production URL (https://moodboostapi.onrender.com/api) is hardcoded for mobile platforms. For web deployment, the app uses different URLs based on the environment: production web app uses the Render.com endpoint, while development builds use localhost for testing.

8.4 Defined Backend Endpoints

The backend system exposes several distinct API endpoints, each designed to fulfill a specific functional role within the application:

- `/api/status/` – Utilized to verify the operational status of the backend server.
- `/api/set_api_key/` – Enables the secure configuration of the Gemini API key.
- `/api/affirmation/` – Generates affirmations that are specifically tailored to a user's current mood.
- `/api/chat/` – Facilitates conversational interactions with the integrated AI assistant.
- `/api/recommendations/` – Provides personalized content recommendations based on user data.

8.5 The Request and Response Process

The request flow is as follows:

1. **Frontend Request:** The Flutter app makes HTTP POST requests to the API endpoints.
2. **Backend Processing:** The Django backend receives the requests and extracts parameters.
3. **AI Interaction:** The backend constructs appropriate prompts for Google's Gemini 1.5 Pro model.
4. **AI Response:** The Gemini API returns generated content.
5. **Backend Formatting:** The backend processes and formats the responses.
6. **Frontend Display:** The Flutter app displays the results to the user.

8.6 Error Handling

The backend includes fallback options when API calls fail. The frontend has built-in retry mechanisms and offline support. Local caching ensures the app remains functional even with connectivity issues. The API design follows REST principles and includes CORS configuration to support cross-origin requests from web clients. The service leverages Render's infrastructure for reliable hosting with automatic deployments from GitHub.

Chapter 9

Data Flow and Code Workflow

9.1 Application Data Flow Sequence

The data flow within the MoodBoost application adheres to a clear and defined sequential process:

1. A user initiates an interaction with the Flutter user interface.
2. The Flutter application transmits an HTTP request to the Django backend.
3. Django processes the incoming request and initiates a call to the Gemini API if the request necessitates AI processing.
4. Django then structures and appropriately formats the generated response.
5. Django returns a JSON response back to the Flutter application.
6. The Flutter application subsequently updates its user interface based on the received data, providing dynamic feedback to the user.

9.2 Core Workflow

Here's a simplified step-by-step of how the MoodBoost app functions:

1. **User Interaction (Flutter Frontend):** Mood is inputted via the UI (e.g., `lib/screens/home_screen.dart`). Flutter services (`lib/services/mood_service.dart`) format the request.
2. **API Request:** Flutter sends HTTP POST/GET requests to Django endpoints (e.g., `/api/generate_affirmation/`).
3. **Backend Processing (Django):** The Django backend receives the request and extracts the mood. A GeminiClient is used to call the Gemini API, which generates a response.
4. **AI Generation (Gemini API):** The GeminiClient calls Google's Gemini API via `call_gemini_api()`, handling errors and fallback responses.

5. **Database Interaction:** Mood data is stored in SQLite using Django models (e.g., MoodEntry model). Recent moods are retrieved for recommendation generation.

6. **Response to Frontend:** A JSON response is sent back to the Flutter app for display.

9.3 Specific Data Flows

- **Mood Tracking:** Mood is submitted via AuthScreen / ContentScreen. It is saved to `moodboost_api/moods/models.py` (assumed model) and retrieved via `/api/moods/recent/` for recommendation generation.

- **Chat Functionality:** Messages are sent via ChatScreen (Flutter). Django's `chat_view` processes the message with `GeminiClient.chat()`, and the response is formatted for Flutter's chat widget.

Chapter 10

Detailed Feature Workflows

10.1 Google Sign-In (User Authentication)

- **User Experience:** Users tap the *"Sign in with Google"* button.
- **Implementation:**
  - Utilizes Firebase Authentication for secure, one-tap login.
  - The Flutter plugin simplifies integration (2-click setup).
  - Upon sign-in, Firebase returns a unique user ID to the Flutter app.

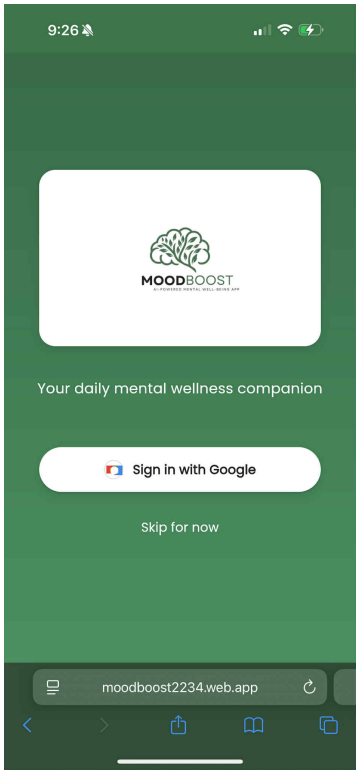

Figure 10.1: Google Sign-In Button Interface

10.2 Mood Tracking

- **User Experience:** Users select an emoji and/or enter a text description to log their mood.
- **Data Handling:** Entries are timestamped and securely stored in Firebase Firestore.
- **Analytics:**
  - Firebase aggregates mood entries for trend analysis.
  - Flutter renders interactive weekly charts for visual feedback.

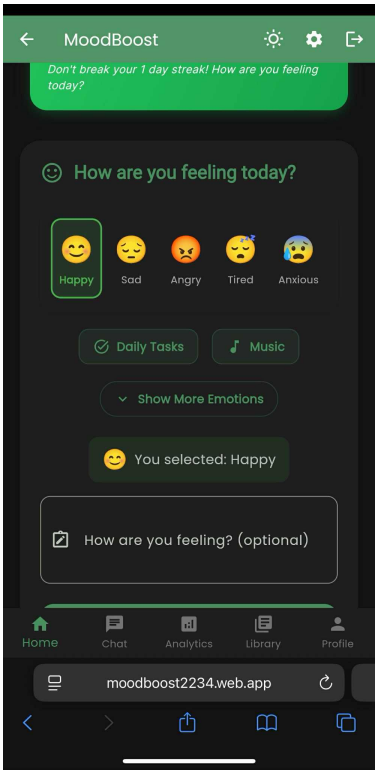

Figure 10.2: Mood Tracking Interface

10.3 AI Chatbot (Gemini Integration)

- **User Experience:** Users initiate a chat with the AI assistant for advice, support, or conversation.
- **Process Flow:**
  - Messages are sent from Flutter to a Django backend.
  - Django forwards the message, along with recent mood history, to Gemini AI (Gemini 2.0 Pro).
  - Gemini generates a context-aware response, which is displayed in the app.

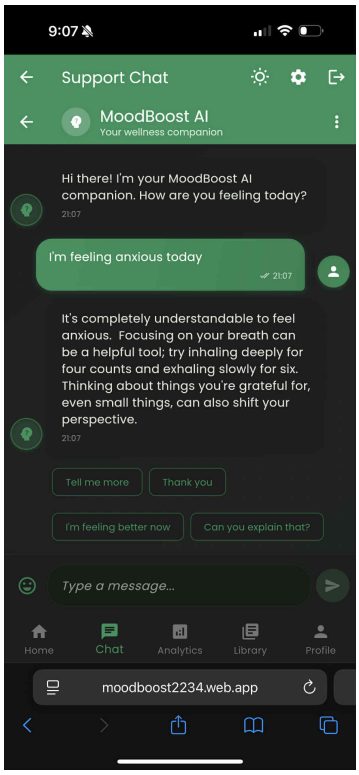

Figure 10.3: AI Chatbot.

10.4 Personalized Content and Recommendations

- **User Experience:** The app suggests articles or videos based on the user's recent moods and engagement.
- **Mechanism:**
  - A change in mood entry triggers a Firebase event.
  - Firebase notifies the Django backend, which consults Gemini for content recommendations based on mood patterns.
  - The app surfaces new, relevant content to the user, ensuring diversity and novelty.

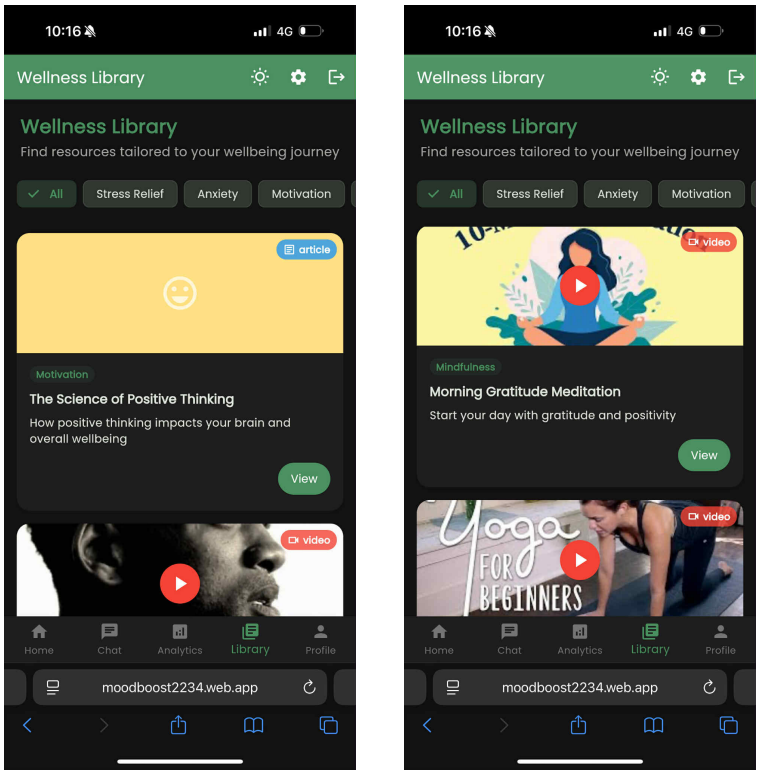

Figure 10.4: Recommendations.

10.5 Mood Analytics

- **User Experience:** Users can view summaries of their mood trends and app activity.
- **Implementation:**
  - Firebase automatically tracks mood frequency and feature usage.
  - Flutter visualizes this data using pie charts and other graphical elements.

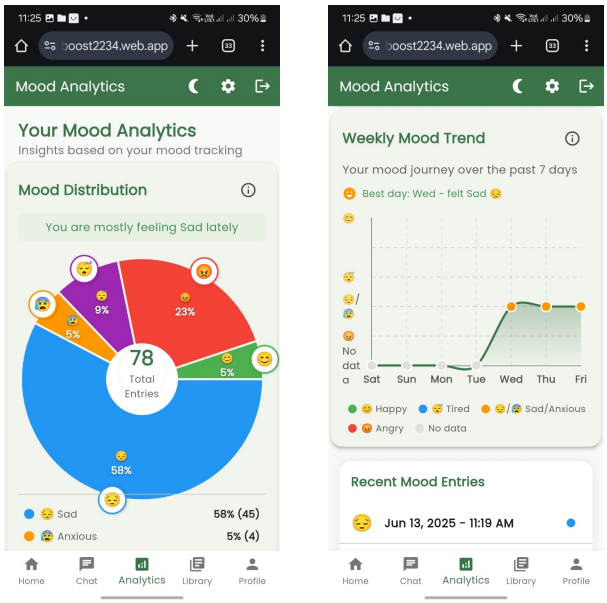

Figure 10.5: Mood Analytics.

10.6 User Feedback System

- **User Experience:** Feedback, suggestions, or problems can be shared with developers right through the app.
- **Handling Data:** Feedback is stored in Firebase for review and action by the development team.

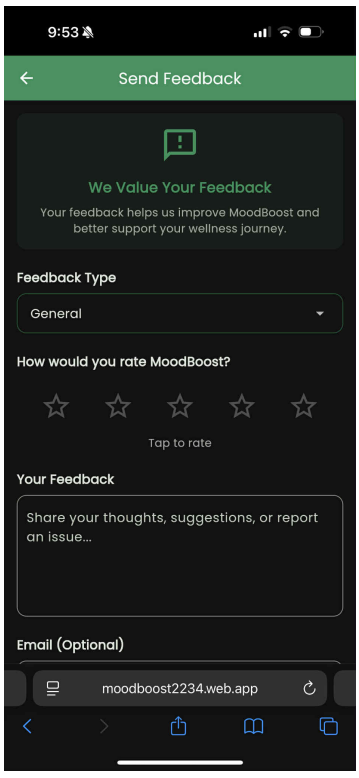

Figure 10.6: User Feedback Interface.

10.7 Why the Architecture Works

- **Free Tier Sustainability:**
  - Firebase: 1GB storage and 50,000 daily database operations support MVP needs.
  - Gemini: 1,000 free AI requests per day, enough for ˜50 active users.
- **No Model Training:** Gemini is pre-trained for mental health prompts—no custom ML model building required.
- **Scalable Platform:** Both Firebase and Gemini allow seamless upgrades to paid plans as usage grows.

10.8 Framework Features Explained

Here's a table outlining the framework features, their solutions, and how they work:

Feature	Framework Solution	How It Works
Google Sign-In	Firebase Authentication	2-click integration via Flutter plugin
Mood Tracking	Flutter UI + Firebase DB	Stores emoji/text logs with timestamps
AI Chatbot	Django + Gemini API	Processes messages through Gemini 2.0 Pro
Personalized Content	Firebase Triggers + Django	Auto-generates recommendations when new mood data arrives
Weekly Analytics	Flutter Charts + Firebase	Aggregates and visualizes mood data as interactive charts

10.9 Gemini API: Suitability & Considerations

Why Gemini Is Ideal

- **Free Tier Generosity:** 1,000 daily requests—sufficient for tens of active users during MVP.
- **Safety Filters:** Automatically blocks inappropriate or unsafe mental health advice.
- **Contextual Intelligence:** Remembers and utilizes chat/mood history (via Firebase storage) for richer conversations.

Limitations

- **Scaling Cost:** After 1,000 free requests/day, costs are $0.00025 per 1,000 characters.
- **Response Specificity:** May deliver generic answers unless prompts are engineered carefully.

May deliver generic answers unless prompts are engineered carefully. The features in MoodBoost have been thoughtfully planned to ensure people get value, AI helps and the system can grow easily. With help from Firebase, Django and Gemini, the app gives users a safe, flexible and aware wellness service at a low price that is scalable as needed.

Chapter 11

MoodBoost Frontend Architecture

11.1 Overview

MoodBoost is a multi-platform mental wellbeing support app developed using Flutter. Leveraging Flutter's capabilities, the app is easily deployable to Android, iOS, and web platforms. The frontend architecture is highly modular, ensuring high performance and a strong user experience. Below is a high-level overview of its main components and technical implementation.

11.2 Application Architecture

- The MoodBoost app initializes from `main.dart`, where firebase services are launched, providers are registered and the root `MyApp` widget is started.
- Navigation is handled through a combination of named routes for primary screens and direct navigation for specific features.
- The provider pattern is used for dependency injection and state management, enabling better testability and scalability.
- A theming framework is implemented to support both light and dark modes, with dynamic text scaling to ensure accessibility.

11.3 User Interface Elements

- The home screen serves as the main dashboard, featuring mood tracking, an inspirational quote of the day and quick links to core sections.
- A persistent bottom navigation bar provides links to five core modules: Home, Chat, Analytics, Library and profile.
- Animated transitions improve navigation flow and enhance the overall user experience.
- The UI is constructed using reusable widgets, ensuring consistency and simplifying updates and scalability.

11.4 Key Feature Implementations

Mood Tracking

- The user interface allows for mood selection through emoji with optional note input.
- Upon mood selection, the `_saveMood()` function calls `MoodService.addMoodEntry()`.
- `MoodService` stores the data in Firebase Firestore.
- A confirmation snack-bar provides feedback to the user, along with additional action options.

AI Chat Assistant

- Implemented in `ChatScreen`, this feature displays messages in chat bubble format with typing effects.
- Messages are sent to the backend via `GeminiService.sendChatMessage()`, which creates a JSON payload containing the user message and chat history.
- Responses from the Django backend (integrated with the Gemini API) are parsed and displayed with animations.
- Fallback mechanisms handle backend delays or failures gracefully.

Analytics Visualization

- Mood data is retrieved in real-time from Firestore using `MoodService.getMoodEntries()`.
- Interactive mood trend charts are generated using the FL chart library.
- Firestore data is transformed into a chart-friendly format before rendering.

Content Suggestions

- Suggestions are personalized based on both historical and current mood data.
- The frontend sends relevant data to the backend API, which returns content recommendations.
- Cached content is used as a fallback when online data is not available.

11.5 Technical Implementation

Service Layer

- `AuthService` handles user authentication through Firebase.
- `MoodService` manages all mood data operations in Firestore.
- `GeminiService` facilitates communication with the Django backend for AI functionalities.
- Mock services are available to simulate actual service behavior during testing.

Data Flow

- The MoodBoost app adopts a reactive UI approach, where widgets respond to live data streams.
- Firestore streams provide real-time data updates.
- The provider pattern enables structured dependency injection throughout the widget tree.

Frontend-backend Communication

- The frontend interacts with Django REST endpoints for AI and content services.
- Robust error handling ensures a smooth user experience even when failures occur.
- The app auto-detects its environment (development, staging, production) to use the appropriate backend URL.

Offline Capabilities

- Firestore's caching allows access to mood data offline.
- Fallback content ensures continued functionality during network issues.
- User preferences and settings are stored locally using `SharedPreferences`.

11.6 UI/UX Design Principles

- A unified color scheme reflecting brand identity is used consistently across the app.
- Dynamic text scaling and optimized contrast improve readability.
- Smooth animations enhance visual experience without compromising performance.
- The layout is responsive, adapting to various screen sizes and orientations to ensure a consistent experience across all devices.

11.7 Frontend Structure Details

The Flutter application is composed of multiple distinct screens, including specialized sections for user authentication, the primary home interface, and feedback submission. State management within the frontend is effectively achieved through the Provider pattern. Comprehensive theme support, encompassing both light and dark modes, is meticulously implemented. Furthermore, various testing utilities, such as `test_api.dart`, are strategically employed for robust API integration testing.

Chapter 12

Database for MoodBoost

12.1 Overview

In order to deliver a user-centered experience across platforms, MoodBoost uses a hybrid database strategy that combines both cloud-based and local database to store user data, settings and game scores. This chapter extends a deep analysis of MoodBoost's data structure, data security measures, data models, offline support and synchronization strategies.
MoodBoost stores data using two types of data storage:

- **Cloud Storage (Firebase Firestore):** used for persistent data that need to be synchronized across devices such as user analytics, mood entries, and content recommendations.
- **Local Storage (SharedPreferences):** Used for storing small, device specific data such as first-login status, user settings, and mini-game progress.

12.2 Cloud Storage: Firebase Firestore

Firebase Firestore is the primary cloud database used for persistent data storage across devices providing NoSQL document based storage. It manages content delivery preference and user specific mood tracking in order that users can access their emotional wellness information on multiple devices.

Mood Data Structure

The data from the mood tracking is stored in a Firestore collection called `moods`. The following schema is adhered by each document:

- **userId (String):** A unique user ID associated to the user who successfully authenticated with Firebase.
- **Mood (String):** An emoji that represents the user emotion state.
- **Note (String):** a textual note to add further detail about user's emotions.
- **Timestamp (timestamp):** A server-generated timestamp that records the creation date of the entry.

Database Security

MoodBoost enforces strict security rules with Firebase Firestore to ensure the confidentiality of user's data. Specifically, it protects sensitive information that are related to the user's mood and preferences.
Firestore security rules enforce that users can:

- **Access Control:** Firestore security rules ensure that users can read and modify such as update and delete record data and write only. The entry is associated with userId which is a unique identifier that matches Firebase user authentication. As a result, any unauthorized access must be prevented.
- **Authentication:** Firestore Authentication is required for all data operations involving personal data. Any unauthenticated request must be denied automatically by Firestore security rules.
- **Access to collections:** Firestore allows global accessibility on certain collections which are protected by role based permission. The certain collections including affirmations, libraries and recommendations are only allowed to read.

Database Indexing

Custom composite indexes are defined for efficient queries:

- **userId (ASC) + timestamp (DESC):** It retrieves reverse chronological display of mood entries.
- **userId (DESC) + timestamp (ASC):** To support chronological analysis.

Query Patterns

Query structures used in the app include:
All Moods:

```
_firestore
    .collection('moods')
    .where('userId', isEqualTo: user.uid)
    .orderBy('timestamp', descending: true);
```

Weekly Moods:

```
_firestore
    .collection('moods')
    .where('userId', isEqualTo: user.uid)
    .where('timestamp', isGreaterThanOrEqualTo: oneWeekAgo)
    .orderBy('timestamp', descending: true);
```

12.3 Local Storage: SharedPreferences

MoodBoost app uses SharedPreferences for the data that does not require cloud synchronization that is optimized for fast access.

Stored Data and Data Types

Game Scores:

- Key: `highscore_{gameId}` (e.g., `highscore_balloon_pop`)
- Type: Integer

User Settings:

- Notification Preferences
  - `notifications_enabled`: Boolean
  - `daily_reminder_enabled`: Boolean
- Appearance Settings:
  - `text_size`: Double
  - `app_language`: String
  - `theme_mode`: String (light/dark/system)
- User Experience:
  - `user_has_logged_in_{userId}`: Boolean (to detect first-time users)
- Game Progress:
  - Keys: `progress_{gameId}`
  - Type: JSON-encoded process data (stored as strings)

12.4 Database Interaction

Data Models

MoodBoost uses Dart model classes like `MoodEntry` to handle data:

- **Serialization:** converts object to Firestore-compatible maps.
- **Deserialization:** creating objects from Firestore data.
- **Type Safety:** enforce consistent data access.

Service Layer

A dedicated service layer has an important role to separate the business logic from UI. Services such as `MoodService` abstract database operations from UI. The following are key functions of the service layer:

- Service layer ensures to handle authentication requirements. For data security purposes, `MoodService` fetches user ID then allows for writing mood entries from `AuthService`.

- Service layer covers the data in robust try-catch blocks which provides error handling.
- It manages the transformation of data between formats such as raw data like Firestore and structured Dart model objects like `MoodEntry` which are used in the app.

Offline Support

- Firestore provides built-in caching which allows users to write and read without internet connection.
- SharedPreferences provides local fallback access in game scores and settings. Moreover, UI shows placeholders when cloud data is unavailable.

Data Flow Architecture

MoodBoost app uses a reactive data flow pattern where user action data can make changes originate such as recording a mood. The current changes are written automatically in the appropriate database. UI components listen to database streams. Then, if the data changes, it is automatically updated by UI.

For Firestore data:

```
// stream-based reactive pattern
return _firestore
    .collection('moods')
    .where('userId', isEqualTo: user.uid)
    .snapshots()
    .map((snapshot) => snapshot.docs
        .map((doc) => MoodEntry.fromFirestore(doc.id, doc.data()))
        .toList());
```

For SharedPreferences:

```
// Load-modify-save pattern
final prefs = await SharedPreferences.getInstance();
await prefs.setInt('highscore_balloon_pop', score);
```

Chapter 13

How YouTube Videos Work in MoodBoost

13.1 How YouTube Videos Work in MoodBoost

When MoodBoost suggests a YouTube video, it utilizes specific YouTube APIs for embedding videos.

13.2 Embedding Videos

- **YouTube iFrame API:** This API is used by the Youtubeer_flutter package, and the required script has been added to web/index.html.
- **No Data API Key Needed:** For basic embedding and playback, there is no need to enable the YouTube Data API or obtain an API key; the iFrame API is sufficient. The Youtubeer_flutter package already handles the player creation with the appropriate parameters.
- **YouTube Data API v3:** This API is only necessary if the app intends to search for videos, get video details, or manage YouTube resources.

Chapter 14

Technical Considerations

14.1 Error Handling

The application incorporates comprehensive error handling:

- A try-catch block is used to enclose API calls or database operations.
- In debug mode, errors are printed.
- Fallback functionality is provided to ensure continued operation.

```
try {
    // API call or database operation
} catch (e) {
    if (kDebugMode) {
        print('Error: $e'); // [cite: 29]
    }
    // Provide fallback functionality
    return fallbackValue; // [cite: 30]
}
```

14.2 Cross-Platform Compatibility

The app's code includes mechanisms for cross-platform compatibility:

- It differentiates between web and mobile implementations.
- For mobile, it can detect if the platform is Android or iOS and execute platform-specific code.
- Fallback implementations are included to handle unexpected platform issues.

```
if (kIsWeb) {
    // Web-specific implementation
} else {
    // Mobile implementation
    try {
        if (Platform.isAndroid) {
            // Android-specific code
```

```
        } else {
            // iOS-specific code
        }
    } catch (e) {
        // Fallback implementation
    }
}
```

14.3 Mock Services for Testing

Mock implementations of services are included for testing and development purposes:

- A MockAuthService extends AuthInterface and provides mock authentication methods.
- The isLoggedIn() method in MockAuthService returns a boolean indicating a mocked login status.
- Additional mock implementations are also present.

```
// Mock auth service for testing
class MockAuthService extends AuthInterface {
    bool _isLoggedIn = false; // [cite: 31]

    @override
    Future<bool> isLoggedIn() async {
        return _isLoggedIn; // [cite: 32]
    }

    // Additional mock implementations
}
```

Chapter 15

The Community Chat System: Connecting for Mental Health Support

15.1 Overview

This system is all about letting people join real-time chatrooms to talk with others about mental health and peer support. It's like a dedicated online space where individuals can come together and share their experiences.

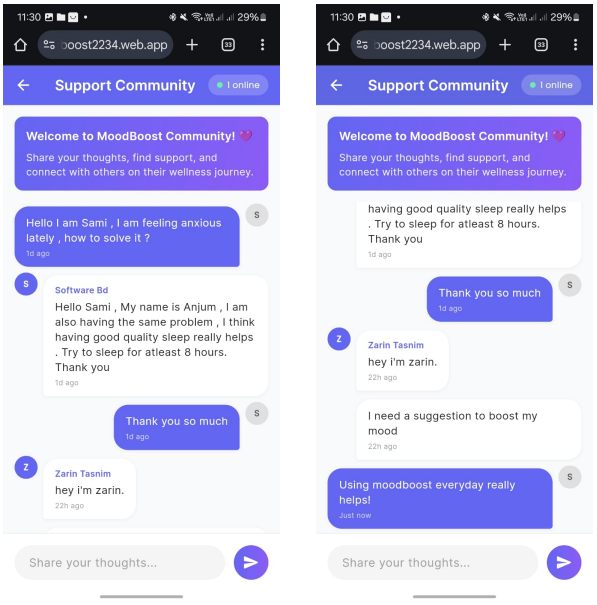

Figure 15.1: Community Chat System.

15.2 Core Functionality: Getting People Talking in Real Time

The Community Chat System welcomes users to a vibrant, common chatroom dedicated to exploring a wide range of mental health topics. In this dynamic space, par-

ticipants can actively engage by sending messages, sharing insights, and connecting with others in real-time. The system provides a clear view of who's currently online and contributing, fostering a sense of community and collaboration.

15.3 User Experience: Making it Easy and Comfortable to Chat

The way the chat looks and feels is designed to be super user-friendly. Messages are displayed clearly, which makes conversations easy to follow. To make things even smoother, there are features like auto-scroll, so you don't miss new messages, and user avatars, which add a personal touch to each person in the chat. These details help create a comfortable and supportive environment for sensitive discussions.

15.4 Security and Access: Keeping the Chat Safe and Private

It's really important that this system is secure, especially since people are talking about personal stuff. All actions within the chat, like joining a room or sending messages, are secure and require users to be logged in. This helps ensure that the space remains safe and private for everyone.

Chapter 16

The Mood Streak Tracking & Notification System: Building Habits for Well-being

16.1 Overview

This system is pretty neat because it encourages users to log their mood daily by tracking streaks (consecutive days logged) and sending smart notifications. It's a way to build a habit of checking in with yourself about your feelings.

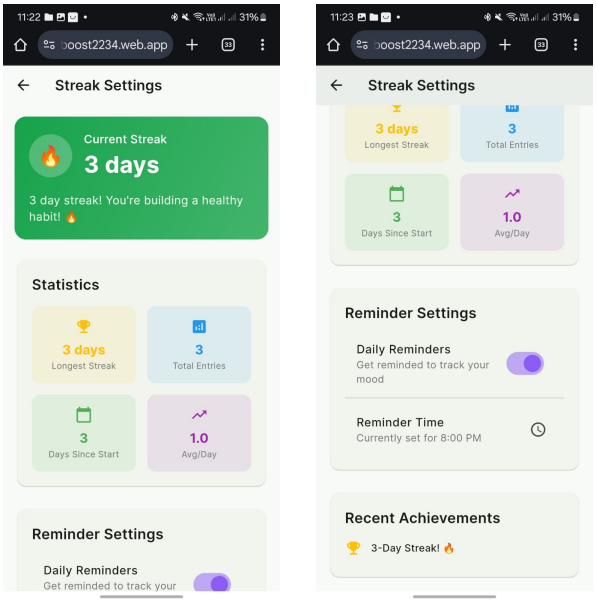

Figure 16.1: Mood Streak and Notification.

16.2 Encouraging Daily Engagement: Streaks as a Motivator

The main idea here is "streaks," which are basically how many days in a row you've logged your mood. Users can see their current streak, which helps motivate them

to keep going, and also their longest streak ever. The system also sends reminders to log your mood, which is super helpful if you forget.

16.3 Celebrating Milestones: Rewards for Progress

To keep users motivated, the system celebrates milestones. For example, if you log your mood for 7 days in a row or even 30, you'll get notifications and visual rewards. It's like getting a small prize for sticking with it!

16.4 Interactive Interface: Showing Progress Visually

The way your progress is shown is really engaging. The interface uses animations, color changes, and motivational messages to display your progress. This makes tracking your mood more interactive and fun.

16.5 Seamless Data Management: Always Accessible

A really practical feature is that all your streak data syncs across different devices. So, you can see your progress no matter what device you're using. Plus, it even works offline, which means you can log your mood even if you don't have internet. This makes it super convenient and reliable.

Chapter 17

Daily Task Feature: A Comprehensive Productivity Solution

17.1 Introduction

The Daily Task Feature is a strong, user-focused system designed to make personal productivity and organization much better. By bringing together easy-to-use task management, smart ways to check progress, and helpful advice, it assists individuals in staying focused, setting priorities effectively, and keeping track of their daily goals. This feature is great for anyone looking to improve time management, build better task management habits, and understand their work patterns more deeply.

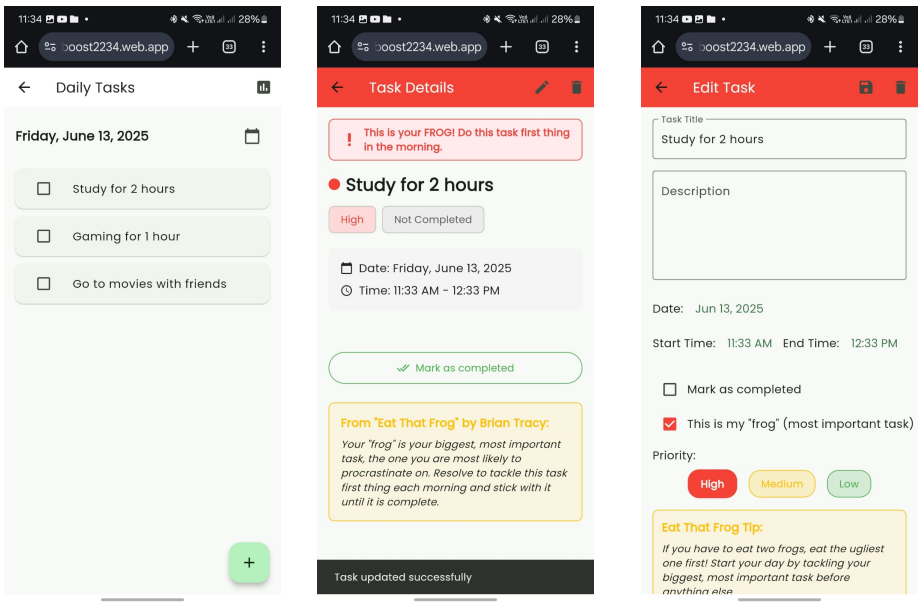

Figure 17.1: Daily tasks.

17.2 Task Management System

The core part of the Daily Task Feature is a flexible and powerful system for managing tasks. It lets users easily create, view, edit, and delete tasks. Each task is made more complete with detailed information to make things clear and organized:

- **Title and Description**: A concise title summarizes the task, while a detailed description provides context, instructions, or additional notes.
- **Due Date and Time**: Users can specify a due date, as well as start and end times, to align tasks with their schedules.
- **Priority Level**: Tasks can be assigned priority levels (e.g., low, medium, high, critical) to help users focus on what matters most.
- **Completion Status**: A clear indicator tracks whether a task is pending, in progress, or completed.
- **"Frog" Status**: Inspired by the "Eat That Frog" methodology, users can designate tasks as "frogs"—critical tasks that should be tackled first to maximize productivity.

This system ensures users have all the tools they need to manage their daily responsibilities effectively.

17.3 Key Features

The Daily Task Feature is packed with functionality to streamline task management and enhance user productivity. Below are the primary components:

Task Creation

- **Detailed Task Input**: Users can create tasks with comprehensive details, including titles, descriptions, due dates, time slots, and priority levels.
- **"Frog" Designation**: Tasks can be marked as "frogs," signaling their importance and encouraging users to address them early in the day.
- **Priority Settings**: A customizable priority system helps users categorize tasks based on urgency and importance.
- **Time Slot Assignment**: Users can allocate specific time blocks for tasks, fostering better time management and reducing scheduling conflicts.

Task Organization

- **Date-Based Sorting**: Tasks are automatically organized by date, allowing users to focus on daily priorities.
- **Flexible Date Navigation**: A calendar interface enables users to view tasks for any selected date, with intuitive navigation for past, present, and future tasks.

- **Visual Indicators**: Color-coded or icon-based indicators highlight task priority, completion status, and "frog" designation for quick reference.
- **Filtering and Sorting**: Users can filter tasks by status, priority, or "frog" designation and sort them to suit their preferences.

Task Evaluation

- **Daily Performance Tracking**: The system evaluates task completion rates daily, providing users with a clear picture of their productivity.
- **Feedback Mechanism**: Personalized feedback is generated based on task completion, offering encouragement and constructive suggestions.
- **"Frog" Recognition**: Completing "frog" tasks triggers special recognition, reinforcing the importance of tackling high-priority items.
- **Progress Insights**: The system identifies patterns in task completion, helping users understand their productivity habits over time.

17.4 Benefits for Users

The Daily Task Feature provides real benefits that help improve productivity, organization, and personal development. These benefits are categorized as follows:

Improved Productivity

- **"Eat That Frog" Methodology**: By prioritizing "frog" tasks, users tackle their most impactful tasks first, setting a positive tone for the day.
- **Time Management**: Assigning time slots to tasks helps users structure their day and avoid procrastination.
- **Priority-Driven Focus**: The priority system ensures users focus on high-value tasks, reducing distractions and inefficiencies.

Better Task Management

- **Clear Visualization**: A clean, intuitive interface displays all tasks with relevant details, making it easy to stay on top of responsibilities.
- **Completion Tracking**: Real-time updates to task status provide a sense of accomplishment and keep users motivated.
- **Flexible Editing**: Tasks can be edited or rescheduled as needed, accommodating changes in priorities or schedules.
- **Reminder System**: Automated reminders for due dates and "frog" tasks ensure users stay on track.

Progress Tracking

- **Daily Evaluations**: The system provides daily summaries of task completion, offering a snapshot of productivity.
- **Performance Insights**: By analyzing completion rates and patterns, users gain a deeper understanding of their work habits.
- **Learning Opportunities**: Feedback and recommendations help users refine their task management strategies.
- **Pattern Recognition**: Over time, users can identify recurring trends, such as peak productivity hours or common obstacles.

Enhanced User Experience

- **Intuitive Interface**: A user-friendly design ensures that even non-technical users can navigate the system with ease.
- **Seamless Navigation**: Calendar-based date selection and task filtering make it simple to access relevant information.
- **Visual Feedback**: Dynamic visual cues (e.g., progress bars, color coding) provide instant feedback on task status.
- **Detailed Task Views**: Users can drill down into task details for additional context or instructions.

17.5 Smart Features

The Daily Task Feature includes smart functions to make the user experience even better and give helpful ideas for action. These clever features include:

Task Evaluation

- **Automated Analysis**: The system automatically evaluates daily task completion, calculating metrics like completion rates and adherence to priorities.
- **Personalized Feedback**: Based on performance, users receive tailored feedback, such as praise for consistent "frog" task completion or tips for improving time allocation.
- **"Frog" Task Recognition**: Completing "frog" tasks earns special acknowledgment, reinforcing positive behavior.
- **Continuous Improvement**: The evaluation system suggests actionable steps to enhance productivity, such as adjusting task priorities or breaking large tasks into smaller steps.

Insights and Recommendations

- **Pattern Analysis**: The system analyzes historical task data to identify trends, such as frequently delayed tasks or optimal work hours.
- **Personalized Recommendations**: Based on insights, users receive suggestions for improving task management, such as scheduling "frog" tasks during peak energy periods.
- **Learning from Past Performance**: The system highlights lessons from previous days, helping users avoid common pitfalls.
- **Proactive Tips**: General tips for better task management, such as batching similar tasks or setting realistic deadlines, are provided regularly.

17.6 Data Management

The Daily Task Feature prioritizes secure, efficient, and user-friendly data handling to ensure a seamless experience:

- **Local Storage**: Tasks are stored locally on the user's device, enabling offline access and fast performance.
- **User-Specific Organization**: Each user's tasks are isolated and organized according to their preferences, ensuring privacy and personalization.
- **Secure Data Management**: Robust encryption and security protocols protect task data from unauthorized access.
- **Synchronization Capabilities**: For users with multiple devices, the system supports cloud synchronization, ensuring tasks are up-to-date across platforms.

17.7 User Goals and Outcomes

The Daily Task Feature is designed to help users achieve the following objectives:

1. **Stay Organized**: A structured approach to task management keeps daily responsibilities clear and manageable.
2. **Prioritize Effectively**: The "frog" system and priority levels ensure users focus on high-impact tasks.
3. **Track Progress**: Daily evaluations and insights provide a clear view of productivity and growth.
4. **Learn from Patterns**: By analyzing task completion trends, users can refine their habits and strategies.
5. **Improve Time Management**: Time slot assignments and reminders help users make the most of their day.
6. **Maintain Structure**: The system fosters a disciplined approach to daily activities, reducing stress and enhancing focus.

17.8 Target Audience

This feature is particularly valuable for individuals who aim to:

- Boost their productivity by focusing on high-priority tasks.
- Enhance time management through structured scheduling.
- Monitor daily progress and celebrate achievements.
- Build sustainable task management habits.
- Gain insights into their work patterns for continuous improvement.
- Stay organized amidst busy schedules and competing responsibilities.

17.9 Conclusion

This Daily Task Feature stands as a comprehensive tool that brings together task management, smart ways to check work, and personal advice to give individuals more control. By utilizing the 'Eat That Frog' methodology, simple organization, and feedback based on information, it assists in staying focused, being effective, and managing daily goals. For both school and work, this feature acts as a powerful ally for achieving better organization, handling time efficiently, and fostering personal development.

Chapter 18

Discussion

The discussion part of the MoodBoost project dives into what the app means for us, what challenges we faced, and where we see it going next. This section really breaks down what we achieved, how well the app's design and features actually work, and how it might change mental health technology.

18.1 Impact on Mental Health

How it Helps Mental Health

MoodBoost is designed to totally change how people get help for their mental health. It's a personal, AI-powered app that gives you custom messages and motivational stuff. We've built in techniques from Cognitive-Behavioral Therapy (CBT) and positive psychology to help users stop negative thought patterns and feel more positive. Since the app learns what you like and how you interact with it, it can really improve how you feel, making mental health support more effective and engaging than older methods.

Its Role in Mental Health Technology

Creating MoodBoost is a big step for mental health technology because it uses advanced AI and Natural Language Processing (NLP). Most current mental health apps don't really offer personalized support, but MoodBoost's AI model ensures that users get content that truly fits their psychological needs. By using NLP to analyze feelings and recommend personalized content, MoodBoost sets a new standard for these kinds of apps, showing what AI can do to improve mental well-being.

User Engagement and Satisfaction

One major thing about MoodBoost is its potential to keep users engaged and satisfied. We focused on a user-centered design, making sure the app's interface is easy to use, fun, and works for all sorts of people. The calming colors, clear fonts, and cool images create a supportive and relaxing space, making the whole experience better. Features like personalized affirmations and motivational content are designed to keep users motivated and committed to their mental health journey.

Wider Impact on Society

Beyond just helping individuals, MoodBoost could have a big impact on society. By offering accessible and personalized mental health support, the app can help reduce the stigma associated with mental health issues. It can also help public health by providing a solution that can reach a lot more people, including those who might not have access to traditional mental health services. This could lead to a mentally stronger and healthier society overall.

18.2 Challenges and Limitations

Technical Challenges

Bringing in advanced tech like NLP models and AI-driven chatbots means we'll always have technical challenges, requiring constant maintenance and updates. Making sure these technologies work perfectly with all parts of the app is key for it to run smoothly. Our app's flexible design allows for adding new features easily, but we'll need continuous technical support to fix any issues that pop up.

User Adoption and Engagement

Getting Users and Keeping Them For MoodBoost to really take off, we need people to download and keep using it. This can be affected by things like our marketing, how we promote it, and what users say. Getting the app to the right audience and keeping them engaged is a significant challenge. We'll need to use strategies like personalized content and AI-powered recommendations to boost how many people use it and how happy they are.

Data Privacy and Security

The app collects and analyzes user data, which brings up serious concerns about data privacy and security. Keeping user data safe is super important, so we need strong security measures like encryption and controls over who can access what. The app also has to follow all the ethical rules and laws about data protection and user rights to make sure users trust us.

Cultural and Linguistic Diversity

Another challenge is making the app work for people from different cultures and who speak different languages. MoodBoost needs to be able to adapt to various languages and cultural backgrounds to be effective for users all around the world. This means a lot of work to make it "local" for different places and integrating NLP models that understand multiple languages.

Chapter 19

Conclusion

MoodBoost is a comprehensive solution that addresses the pressing need for accessible and evidence-based mental health care. By harnessing the power of NLP technology, scalable architecture, and user-centered design, we have created a platform that adapts to individual needs and provides personalized support.With its ability to produce context-sensitive affirmation sentences and real-time chatbot interventions, MoodBoost offers a unique and effective approach to mental health care. Its scalable architecture enables seamless performance, even with a large user base, and its curated content library provides users with relevant and engaging content.As we move forward, we are committed to continuing to develop and improve MoodBoost, with features such as multilingual support, wearable integration, and predictive mood modeling. Our goal is to make mental health care more accessible, more effective, and more personalized, and we believe that MoodBoost is a significant step towards achieving this goal.

Bibliography

[1] "Personalized content recommendation using natural language processing," *IEEE Transactions on Knowledge and Data Engineering*, 2019.

[2] "Ai in mental health: A systematic review," *Journal of Medical Systems*, 2020.

[3] "Effectiveness and safety of using chatbots to improve mental health: Systematic review and meta-analysis," *Journal of Medical Internet Research*, 2020, Accessed: 2024-10-22. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC10242473/.

[4] "Machine learning in mental health: A systematic review," *Journal of Medical Systems*, 2020.

[5] "Ai-powered chatbots for mental health support," *Journal of Medical Internet Research*, 2021.

[6] "Artificial intelligence for chatbots in mental health: Opportunities and challenges," 2021, Accessed: 2024-10-22. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9931284/.

[7] "An overview of chatbot-based mobile mental health apps: Insights from app description and user reviews," *PMC*, 2022, Accessed: 2024-10-22. [Online]. Available: https://pmc.ncbi.nlm.nih.gov/articles/PMC5977660/.

[8] "Database management systems for healthcare," *Journal of Healthcare Engineering*, 2022.

[9] "Effectiveness of ai-based chatbots in mental health support: A systematic review," *Journal of Healthcare AI and ML*, 2022, Accessed: 2024-10-22. [Online]. Available: https://www.mdpi.com/2076-3417/14/13/5889.

[10] *Mental disorders*, Accessed: 2025-06-18, World Health Organization, 2022. [Online]. Available: https://www.who.int/news-room/fact-sheets/detail/mental-disorders.

[11] "Mobile apps for mood tracking: An analysis of features and user reviews," *PMC*, 2022, Accessed: 2024-10-22. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC10242473/.

[12] "Nlp-based interventions for mental health," *Journal of Clinical Psychology*, 2022.

[13] "Chatbots and mental health: Insights into the safety of generative ai," *Journal of Consumer Psychology*, 2023, Accessed: 2024-10-22. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2949916X24000525.

[14] "Natural language processing for mental health interventions: A systematic review and research framework," *Translational Psychiatry*, 2023, Accessed: 2024-10-22. [Online]. Available: https://www.nature.com/articles/s41398-023-02592-2.

[15] "Ai-driven mental health support: A systematic review," *Journal of Medical Internet Research*, 2024, Accessed: 2024-10-22. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8132982/.

[16] A. Chaudhry, "User perceptions and experiences of an ai-driven conversational agent for mental health support," *mHealth*, 2024, Accessed: 2024-10-22. [Online]. Available: https://mhealth.amegroups.org/article/view/126211/html.

[17] "Self-administered interventions based on natural language processing models for reducing depressive and anxiety symptoms: Systematic review and meta-analysis," *Journal of Medical Internet Research*, 2024, Accessed: 2024-10-22. [Online]. Available: https://mental.jmir.org/2024/1/e59560.

[18] Binariks, *Ai in mental health: Applications, benefits & challenges*, https://binariks.com/blog/ai-mental-health-examples-benefits/, Accessed: 2024-10-22.