FeelBit Mood Tracker Design Manual (CSCI205 Team 6)

INTRODUCTION

Color codes have no significant purpose, only aim to draw attention to key classes.

FeelBit is a Java-based application designed to help users track and improve their emotional well-being. By integrating mood tracking, journaling, and actionable insights, the app empowers users to reflect on their experiences and better understand their emotions. Developed using JavaFX for an intuitive graphical user interface, FeelBit combines functionality with ease of use. At its core, FeelBit leverages MongoDB for robust data storage, ensuring user datca is securely managed and efficiently accessible.

The application provides a comprehensive user account management system. Users can create accounts, log in securely, and store their credentials in a dedicated MongoDB collection. Account creation ensures that usernames are unique and validates password inputs for security. Upon successful login, users are redirected to the main interface, which acts as a central hub for navigating the app's features. This modular design is facilitated by a Helper class, ensuring smooth transitions between different functionalities, such as mood logging, journal history, and mood statistics.

At the heart of FeelBit is its mood tracking functionality. Users can log their moods by selecting from a range of predefined categories represented by emojis, such as Happy, Sad, Angry, Confused, or Grateful. Each mood is associated with a symbol and description, providing users with a visual and textual representation of their emotional state. To enhance the depth of mood tracking, users can also specify triggers—events or contexts that caused their mood—such as Work, School, Family, or Custom. This information is managed through the MoodEntry and TriggerEntry classes, ensuring that emotional data is captured comprehensively.

FeelBit goes beyond basic mood logging by offering a journaling feature. Users can optionally add written entries to elaborate on their emotions and experiences. These journal entries, managed by the JournalEntry class, allow for deeper reflection and are stored alongside mood and trigger data in MongoDB. This integration of mood, triggers, and journal entries forms the basis for the app's personalized suggestion engine, which uses an external API to generate targeted recommendations. These suggestions, designed to be concise and actionable, help users improve their well-being and address specific emotional challenges.

To help users gain insights into their emotional patterns, FeelBit includes a statistics module that visualizes mood trends over time using interactive charts. This feature, managed by the

MoodStatsController, aggregates data stored in MongoDB to provide users with a clear understanding of recurring emotions and their potential causes. The visualization aids users in identifying patterns and making informed decisions to improve their emotional health.

FeelBit is built with a modular architecture that ensures maintainability and scalability. Controllers for each functionality, such as LogEmotionController for mood logging and JournalHistoryController for accessing journal records, encapsulate specific behaviors, allowing for easy updates and enhancements. A key aspect of its technical infrastructure is the MongoDBConnect class, which manages the connection to the database, ensuring reliable and efficient data operations. Additionally, serialized logs, handled by the LogFile class, provide a mechanism for storing and retrieving user data locally.

In summary, FeelBit offers a seamless and comprehensive solution for emotional well-being. Its integration of mood tracking, journaling, and actionable suggestions, combined with robust data management and user-friendly navigation, makes it an invaluable tool for fostering self-awareness and improving mental health.

USER STORIES

Implemented in the app:

- Arn Utsi
    - **User Story:** As a **Neurodivergent Man,** I want to use the app to track my mental state throughout time, therefore better collaborating with my therapist to improve my mental health.

- Rosemary Ward
    - **User Story:** As an **Overworked Single Mom**, I want to process my emotions without taking time off my kids, so I can feel better.

- Mia Walker
    - **User Story:** As a **Moody Teenager,** I want a tool that helps me explore my thoughts and feelings in a fun and natural way so that I can better share them with my loved ones.

- Marco Sanchez
    - **User Story:** As a user who seeks a fresh start, I want to properly journal my life so that I am able to reflect on it, receive constructive feedback and make wise decisions.

- Lara Fournier
    - **User Story:** As a female industrial manager, I barely get time to sleep let alone checking on my well being. I want an app to help me keep track of my mood and mental well being and give me some quick tips and suggestions without acquiring so much of my time.

Summary & Design Connections to User Stories:

Ultimately, FeelBit succeeded in completing all user stories by implementing aspects that fulfilled the intentions of our users. For **Arn Utsi,** we ensured our code could track aspects of his inputs overtime, giving him the ability to view a selection of stored journal entries or view a graphical interpretation of his most common emoticon inputs. For **Rosemary Ward**, we ensured that design elements in the LogEmotionController enabled a user to do quick inputs of their emotions in one page. She can quickly tap an emoji, select from a drop-down box, and input a quick journal entry seamlessly, only needing to press 'Submit' to store her results. For **Mia Walker's** goal of having the app be fun and engaging, we made sure to use pastel color for the logo and background, alongside having cute emojis to represent various emotions. For **Marco Sanchez**, a man who sought a space to journal his feelings, we created the JournalEntry class to code journaling capabilities. Lastly, busy engineer **Laura Fournier** wanted an app that would suggest her quick suggestions for her inputted problems. For this, we utilized the Suggestions class, which worked with Gemini API to generate recommendations based on someone's immediate journal input.

OBJECT ORIENTED DESIGN

FeelBit employs a modular, object-oriented design that prioritizes encapsulation, abstraction, and reusability to deliver a seamless user experience. The application's architecture revolves around well-defined classes that interact seamlessly to provide key functionalities such as user account management, mood tracking, journaling, and actionable suggestions. This design ensures scalability, maintainability, and clarity in its implementation.

The **user management system** is central to FeelBit, with classes such as UserAccount, MongoDBConnect, LoginController, and CreateAccountController managing account-related operations. UserAccount handles the creation, validation, and storage of user credentials by interacting directly with MongoDB through MongoDBConnect. The LoginController and CreateAccountController facilitate user interactions during login and account creation processes,

ensuring a secure and intuitive flow. The Helper class supports seamless navigation between login and main application interfaces, ensuring users can easily access the app's features.

Mood tracking is a core functionality of FeelBit, facilitated by classes such as MoodEntry, TriggerEntry, Emoji, and LogEmotionController. The MoodEntry class captures and stores mood-related data, including the selected emoji and associated triggers recorded by the TriggerEntry class. LogEmotionController orchestrates the process, collecting user inputs and saving them into structured logs. The Emoji class standardizes mood categories with visual and textual representations, enhancing user engagement. Together, these components allow users to accurately track and contextualize their emotional states.

Journaling and suggestions extend the functionality of FeelBit by encouraging deeper user reflection and providing personalized insights. The JournalEntry class enables users to document their thoughts, which are then integrated into mood logs. The Suggestion class utilizes external APIs to generate targeted recommendations based on the user's mood, trigger, and journal entry. These suggestions are concise and actionable, aiming to improve users' emotional well-being. To ensure data persistence, the LogFile class handles the serialization of mood logs, enabling efficient storage and retrieval of user data.

To provide users with actionable insights, FeelBit includes a **statistics module** managed by the MoodStatsController. This module aggregates mood data from logs and visualizes trends using interactive pie charts, enabling users to identify patterns in their emotional experiences over time. The module integrates seamlessly with the database through MongoDBConnect and allows users to return to the main interface easily with the assistance of the Helper class.

The application's **underlying system utilities**, such as Helper and MongoDBConnect, are essential for enabling its modular architecture. The Helper class streamlines navigation across different app sections, while MongoDBConnect manages database operations, ensuring efficient and reliable data interactions. These utilities collaborate with various controllers and data-handling classes to provide a cohesive and user-friendly experience.

JournalEntry	
Responsibilities	**Collaborators**
• This class represent a journal entry. • This class manages to create, viewing, and deleting journal entries.	• Suggestion • UserLog • MoodSuggestionHubController • LogEmotionController

MoodEntry	
Responsibilities	**Collaborators**
• Contains a couple of emoji attributes showing specific moods and emotions. • The class represents a mood entry in the system.	• Suggestion • JournalEntry • UserLog • MoodSuggestionHubController • LogEmotionController

UserLog	
Responsibilities	**Collaborators**
• This class represents a user's journal log. • Stores all the required data to be serialized.	• TriggerEntry • Suggestion • MoodEntry • JournalEntry • MoodSuggestionHubController • LogEmotionController

CreateAccountController	
Responsibilities	**Collaborators**
• If the user doesn't have an account set, this class creates their account. • Stores the new user's login info in the database.	• MangoDBConnect • UserAccount • LoginController • CreateAccount Controller • UserAccount