

Project Report



A cell phone with a heart and a head

Description automatically generated

TU Dublin

04/10/2024

B00135198, B00138007, B00148322

Contents

[Introduction 2](#_Toc180607058)

[Background 2](#_Toc180607059)

[Motivation for this project 3](#_Toc180607060)

[Project Novelty 3](#_Toc180607061)

[Deliverables 3](#_Toc180607062)

[Plan 4](#_Toc180607063)

[System Design 4](#_Toc180607064)

[Functional Requirements 4](#_Toc180607065)

[Core Features 6](#_Toc180607066)

[Journal 9](#_Toc180607067)

[Design Template 11](#_Toc180607068)

[Bibliography 12](#_Toc180607069)

# Introduction

For our Major Group Project, we have decided to develop a mental well-being mobile application. The aim of this project is to design and implement a working prototype that helps users manage and improve their mental health through app features related to monitoring and maintaining each user’s mental well-being’*.* With growing awareness of mental well-being, our group collectively agree that in 2024 this project is fitting and meaningful. While there are now quite a few mental well-being apps on the market, they all focus a lot of their resources into one standout core feature. We aim to pick the most beneficial features on the market and implement them into one true mental well-being application with possibly developing a new feature that may be lacking. This report outlines the project's background and motivation, defines the problem, and presents its aims and objectives.

“ App-based self-help has an enormous potential to reach a large number of people and provide high quality services at a low cost. Its impact on public mental health can be enormous. ”

[(Kerber1, et al., 2023)](#_Bibliography)

# Background

Our project focuses on developing a functional and useful mental well-being app prototype. The specific objectives include:

**Assess the Situation:**  Initially our group will study the problem of mental well-being and current solutions available in the market, documenting this analysis using appropriate techniques such as SWOT analysis (Strengths, Weaknesses, Opportunities, Threats) and requirements gathering through online research.

**Design a Solution:** Based on our findings, we will design a solution for the problem. This includes creating detailed specifications on how the app should function, what tools will be used, and how it will meet user needs. Evaluate resources needed (technologies, skills, time), and prepare functional requirements and UI design

**Implement a Working Prototype:** we will develop a functional prototype of the mental well-being app that provides core features such *as’ Social Support’, ‘Journaling’* and necessary RESTful api’ssuchas *‘Mood Tracking’ and ‘Guided Meditation’.*

**Evaluate the Prototype:** we will conduct functional and user testing to assess how well the prototype meets its objectives and gather feedback on areas of improvement.

**Propose Future Work:** we aim to provide recommendations for taking the prototype from this stage to a fully working system. This will include identifying areas like enhanced security, scalability, and improved user experience that need to be addressed.

## Motivation for this project

We chose this project due to the growing importance of mental well-being. With rising stress levels and mental health challenges, especially from social media and the pandemic, there is a clear need for digital tools to support mental health. One of the members of the group has relatively recently lost a cousin through poor mental health so this project holds extra incentive to achieve. While existing apps often focus on one area (like meditation or journaling), we aim to create a more balanced platform that combines several mental wellness tools in one app. This project also draws on the skills we've developed throughout our course, such as:

**JavaScript**: A versatile language used for both front-end and back-end development. On the server-side, JavaScript (through **Node.js**) allows us to manage databases like MongoDB

**Backend and Database Management:** We each have a background in server-side development **(PHP**) and database management (**SQL, MongoDB**), enabling us to create a complete application.

**System Design and Integration:** We've learned design principles and system structure, which will help us create a smooth app experience.

## Project Novelty

The novelty of this project lies in the integration of multiple mental well-being tools into one app, allowing users to track their mood, meditate, journal, social support. The app will host a comprehensive range of different respective mental health issues which will offer an initial brief background followed by research and respective services available. A fundamental feature of any mental well-being app is a journal which is now coined as ‘journaling’, we want to design a smart journal that is automatically updated in real time. The mood tracker will also have an activities recommendation feature based on the users mood. Additionally, the project is built to be scalable and future-proof, using a modern mobile development framework and focusing on cloud-based backends. Our project uses data analytics to help users track their mental health over time and spot patterns.

## Deliverables

The project will deliver the following:

* **Working Prototype**: A functional mobile app prototype that includes the core features:
  + - Social Support
    - mood tracking
    - journaling
    - Planner
    - guided meditation
    - find a therapist
* **User Documentation**: A basic user manual for the prototype, explaining how to use the app and its features.
* **Evaluation Report**: A thorough review of the system based on functional testing and user acceptance testing (UAT) results.
* **Recommendations for Further Work**: A set of recommendations outlining what needs to be done to move from the prototype to a fully working system. This will include considerations like enhanced security, user interface improvements, and support for additional features such as social integration or real-time counselling services.

## Plan

A screenshot of a calendar

Description automatically generated

# System Design

This will act as a contract between the development team and the clients, outlining the system's functionality and data requirements. The following sections will break down these details:

## Functional Requirements

* **User Registration and Authentication:** Users will be able to register, log in, and manage their accounts. Basic authentication using email and password.
* **Social Support:** a community-based feature where users can share positive experiences, mental health tips, or inspirational quotes.
* **Mood Tracking:** Users can log their mood daily, choose from predefined moods, and add notes.
* **Guided Meditations:** The app will offer a library of guided meditations that users can access and listen to.
* **Journaling:** A space for users to write daily reflections or entries, stored and accessible for future reference.
* **Analysis**: Users can view patterns in their mood and journaling data over time through charts and graphs. Reminders and Notifications: Push notifications to remind users to log their mood or complete daily meditation/journal tasks.
* **Frontend Development**: We will use React Native for frontend development as it is a popular framework for building mobile applications using JavaScript and React. React Native allows developers to write a single codebase in JavaScript, which can be used to create applications for both iOS and Android. This saves time and resources compared to developing separate native applications for each platform.

Vue.js is a popular JavaScript framework known for being simple and flexible when building user interfaces and single-page apps. It’s often considered an alternative to React Native for mobile app development, especially with tools like Vue Native, which let developers create native apps using Vue’s syntax. Vue.js is easy to learn and lightweight, making it a good option for smaller projects. However, we chose React Native because it has a larger, more active community, giving us better access to resources, support, and third-party libraries if we run into issues.

* **Backend Development:** We will use Node.js for the backend because it handles asynchronous operations well and works with JavaScript, allowing us to use the same language for both the frontend and backend. Node.js also offers a strong framework for building RESTful APIs with Express.js.

Django, a Python-based web framework, is a strong alternative to Node.js for backend development. It comes with built-in features like authentication and database management, which can speed up development. Its Model-View-Template (MVT) architecture makes it great for building secure and scalable apps. However, while Django is a good option, we chose Node.js because it’s better for real-time applications and allows us to use JavaScript for both the frontend and backend, making development more streamlined.

* **Database Management:** We will use MongoDB as our database because of its flexibility in handling unstructured data and its compatibility with Node.js through the Mongoose library. MongoDB's document-based structure makes it ideal for storing dynamic user data, such as mood entries and journaling data, allowing for easy updates and scalability as our application grows. MongoDB's document-based structure makes it ideal for storing dynamic user data, such as mood entries and journaling data, allowing for easy scalability as the application grows. This structure allows us to run detailed queries, helping us analyse user data and offer personalized insights

SQL databases, such as MySQL or PostgreSQL, are traditional relational databases that use structured tables and are ideal for applications with well-defined schemas and relationships between data. They offer strong data integrity, complex querying through SQL, and are widely used in many industries. However, our group has decided to use MongoDB instead because it provides more flexibility for handling unstructured or semi-structured data, which suits our mental well-being app. MongoDB’s document-based structure allows us to easily store and manage dynamic user data, like mood entries and journal logs, without needing to predefine a rigid schema. Additionally, MongoDB scales more easily with large amounts of data, making it better suited to our app's potential growth and varied data requirements.

## Core Features

**Mood Tracking API**

A REST API which allows users track their mood.

**Features**

1. Allows users to Login to their account, and privately submit their current mood to a persistent database. Users can view their mood entries over time.
2. Users who submit at least 1 mood rating daily will maintain a "streak" (a measure of consecutive days in which they submitted to the app).
3. The user's streak percentile compared to other users will be tracked, and if a user is in the 50% percentile, it will be retrievable
4. The development server for the application is containerized using Docker

**How it works:**

* **User Authentication** 
  + Login Process: Users authenticate themselves by providing their credentials (username/email and password).
  + Token Generation: Upon successful login, the API generates a JSON Web Token (JWT) that the client app uses for future requests, ensuring secure access to the user’s data.
* **Submitting Mood Entries**
  + Mood Submission: Users can submit their current mood through the mobile app interface. This typically involves selecting a mood from predefined options and optionally adding notes.
  + Data Storage: When a mood is submitted, the API receives the request and stores the entry in a persistent database (MongoDB).
  + Response: The API responds with a success message or the details of the stored entry, confirming that the mood has been logged.
* **Maintaining User Streaks** 
  + Daily Tracking: The API checks if the user has submitted at least one mood entry per day. If so, their streak is updated.

* + Streak Calculation: The streak is calculated based on consecutive days of mood submissions, and the data is stored alongside the user’s mood entries.
* **Calculating Streak Percentiles** 
  + Streak Percentile Tracking: The API tracks all users' streaks and calculates percentiles. For example, if a user falls within the 50th percentile, it indicates that they have maintained a streak longer than half of all users. Retrieval: Users can retrieve their percentile ranking through the API, which can help motivate them to maintain or improve their mood logging habits.
* **Data Retrieval**
  + Viewing Mood History: Users can retrieve their mood entries over time via a GET request to the API. The response includes all logged moods, allowing users to view trends and patterns.
  + Analysis of Mood Patterns: The API may also provide analytics based on the historical data to help users understand their emotional well-being better.
* **Docker Containerization** 
  + Development Environment: The entire application is containerized using Docker, allowing for easy setup and deployment. Docker ensures that the development, testing, and production environments are consistent, reducing discrepancies caused by different setups.
  + Image Creation: A Docker image is created for the application, which includes the Node.js server and dependencies, making it easier to run on any platform that supports Docker.
* **Backend Setup:** 
  + We will use Node.js for the backend because its asynchronous capabilities allow it to handle multiple API requests efficiently, which is essential for real-time data retrieval and submission.
  + Axios Library: We will install Axios to simplify our HTTP requests to the Mood Tracker API. Its promise-based structure will make it easier to handle asynchronous operations, and its built-in error handling will enhance the reliability of our API interactions.
* **Security Measures:**
  + Data Encryption: We will implement HTTPS for secure data transmission, ensuring that user information is protected during communication with the API.
  + Authentication: We will incorporate JWT (JSON Web Tokens) for user authentication, allowing us to restrict access to authorized users only.
    - **Fundamental Security Measures:**
      * Data Encryption: Use HTTPS to encrypt data during transmission. Encrypt sensitive data stored in the database.
      * Authentication and Authorization: Implement secure authentication for authorized access. Use role-based access control to limit user access to sensitive features.
      * Data Anonymization: Anonymize user data to reduce risks associated with data breaches.
      * User Consent and Transparency: Obtain explicit user consent before collecting mood data. Clearly communicate data usage and provide options for users to manage their data.
      * Data Minimization: Collect only necessary data for app functionality, avoiding excessive information.
      * Secure API Practices: Implement rate limiting to prevent API
* **Conclusion:**

The Mood Tracking API allows users to track their mood securely, maintain daily streaks, and compare their performance with others. With Docker containerization, the development process becomes streamlined, enabling consistent deployment and testing.

## Journal

Creating a journaling feature that communicates with the Mood Tracker API can enhance the overall user experience by allowing users to reflect on their moods and experiences. Here’s a step-by-step approach to implementing a journal that integrates with the Mood Tracker API:

1. Design the Journal Feature

* User Interface (UI): Create a user-friendly interface where users can add, edit, and delete journal entries. Each entry should include fields for the date, mood selection (linked to the Mood Tracker API), and a text area for writing reflections.
* Data Structure: Decide on a data structure for the journal entries, typically including fields like entryId, date, moodId, content, and userId.

2. Set Up API Endpoints

* Create Journal Entry: Implement an endpoint to create a new journal entry. This can use a POST request to the server, which will then save the entry in the database.
* Retrieve Journal Entries: Use a GET request to fetch journal entries for a specific user, potentially filtering by date or mood.
* Update Journal Entry: Implement a PUT or PATCH request to update existing journal entries.
* Delete Journal Entry: Allow users to delete entries using a DELETE request.

3. Database Integration

* MongoDB Structure: Create a collection in MongoDB to store journal entries. Each entry can be linked to the user's mood tracked by the Mood Tracker API by storing the moodId.
* Connection Logic: Set up connection logic between your backend (Node.js) and MongoDB, using a library like Mongoose for easier data handling.

4. Frontend Implementation

* React Native Forms: Use React Native to create forms for users to submit their journal entries.
* State Management: Implement state management (e.g., using Redux) to manage the journal entries within your app, ensuring that updates to the journal are reflected in the UI immediately.

5. API Communication

* Axios or Fetch API: Use Axios or the Fetch API to handle HTTP requests to your backend server. This will enable your app to send and receive data from the journal API endpoints.
* Mood Tracking Integration: When a user logs their mood, capture the mood ID and store it with the corresponding journal entry. This linkage allows users to reflect on how their mood correlates with their written thoughts.

Calendar API

* Integrating a calendar API can enhance the journaling feature by allowing users to select dates easily when logging their mood or journal entries.
* A calendar view can provide a visual representation of mood data, helping users identify trends over time.
* The calendar API can also facilitate reminders for users to log their moods consistently, increasing engagement with the app.

6. Security Measures

* User Authentication: Ensure that users are authenticated before accessing their journal entries. Use methods like JWT for secure authentication.
* Data Validation: Implement input validation both on the client and server sides to protect against malicious entries.

7. Testing and Iteration

* User Testing: Conduct user testing to ensure the journal feature is intuitive and meets user needs. Gather feedback for improvements.
* Iterate on Design: Use the feedback to iterate on your design and functionality, making adjustments to the UI or features as necessary.

## A screenshot of a login screen Description automatically generatedDesign Template

# Bibliography

[Kerber1, A., Beintner, . I., Burchert, . S. & Knaevelsrud, . C., 2023.](https://mental.jmir.org/2023/1/e45068) *[https://mental.jmir.org/2023/1/e45068.](https://mental.jmir.org/2023/1/e45068)* [[Online]   
Available at: mental.jmir.org  
[Accessed October 2024].](https://mental.jmir.org/2023/1/e45068)