Problem Statement 10:

Team Members

- Archana Nair
- Sarabu Jyothika

1. Project Overview

The Emergency and Wellness Management System is designed to address the critical need for personal safety and mental well-being in today's urban environment. This all-in-one platform offers users comprehensive tools to manage emergencies, track their mental health, and access support resources. By integrating features such as SOS help with real-time location sharing, a depression assessment questionnaire, and video recommendations, the system aims to provide a holistic approach to managing both safety and wellness concerns.

2. Scope

This document outlines the design and architecture of the Emergency and Wellness Management System, which is tailored to empower users in high-risk environments. It details the functional and non-functional requirements necessary to deliver a reliable and user-friendly solution. The scope includes a comprehensive overview of the technologies used, system components, interactions, and how these elements work together to meet the needs of users seeking enhanced safety and wellness support.

3. Functional Requirements

- **SOS Help:**
  - This feature allows users to send their current location to pre-defined contacts in case of an emergency. The system ensures that location data is transmitted quickly and accurately to help users get the assistance they need in urgent situations. Integration with GPS and messaging services is critical for the effective operation of this feature.

API USED:

https://console.twilio.com/us1/develop/sms/try-it-out/send-an-sms

- **Questionnaire:**
  - **Assessment**: Evaluates mental health through responses to depression-related questions.
  - **Scoring**: Uses a scoring algorithm to assess the user's mental health status.
  - **Recommendations**: Provides personalized recommendations and resources based on the score.
  - **Support**: Features a dynamic recommendation engine for tailored mental health support.
- **Chatbot Support:**
  - The chatbot offers real-time assistance and answers to users' queries about safety and wellness. It is designed to provide immediate responses and guide users through various features of the application. The chatbot utilizes natural language processing to understand and address user concerns effectively.

API USED:
https://aistudio.google.com/app/apikey?_gl=1*1e9fn51*_ga*MTIyMDc4ODgzMi4xNzI0ODIwNTQ0*_ga_P1DBVKWT6V*MTcyNTQ1MjI4Mi4yLjEuMTcyNTQ1MjI4My41OS4wLjU5MTg1MDYzNg..

- **Video Recommendations:**
  - Users receive recommendations for videos that focus on managing stress, anxiety, and improving overall well-being. The system curates these recommendations based on user preferences and current needs. Integration with video APIs and content filtering mechanisms ensures relevant and useful content is provided.

API USED:

https://developers.google.com/youtube/v3

- **Mood Tracker:**
  - The mood tracker allows users to log their mood on a sliding scale and provides insights into their emotional state over time. Weekly reports are generated to help users identify patterns and trends in their mood. This feature

includes visualization tools to display mood trends and facilitate self-assessment.

- **News API:**
  - The News API fetches and displays articles related to emergency preparedness and health topics. This feature keeps users informed about relevant news and updates that may impact their safety and well-being. Integration with news sources and content aggregation tools ensures the delivery of timely and accurate information.

  API USED:

  https://newsapi.org/sources

- **Task Management:**
  - **Add Tasks:** Enter task details, set a due date, priority, and reminder.
  - **Modify Tasks:** Update task details, including title, due date, and reminder settings.
  - **Delete Tasks:** Remove tasks that are no longer needed.
  - **Reminders:** Receive notifications based on your reminder settings to keep track of tasks.

4. Non-Functional Requirements

- **Performance:**

  The system is designed to deliver a seamless user experience with quick load times and real-time updates. For example, the Overview tab should load within 3 seconds, and changes in transactions or balances should be reflected within 5 seconds. Ensuring high performance is crucial for user satisfaction and effective emergency management.

- **Scalability:**

  The application must accommodate a growing number of users and an increasing volume of data without significant performance degradation. It should support the addition of new features and the expansion of functionalities as user needs evolve. Scalability is achieved through modular design and robust backend infrastructure.

- **Reliability:**

The system aims for 99.9% uptime, ensuring that it is consistently available for users. Reliability also involves robust data handling processes to prevent data loss or corruption. Regular system maintenance and backup procedures are implemented to support high availability and data integrity.

- **Usability:**

  The user interface is designed to be intuitive and accessible, catering to users with varying levels of technical expertise. Clear navigation, user-friendly design elements, and helpful instructions are provided to enhance the overall user experience. Accessibility features ensure that the application is usable by individuals with different abilities.

- **Security:**

  Implementing strong security measures is essential to protect sensitive personal and financial information from unauthorized access. The system uses encryption, secure authentication methods, and regular security audits to safeguard user data. Data protection strategies are in place to address potential vulnerabilities and breaches.

- **Compatibility:**

  The application is compatible with major web browsers such as Chrome, Firefox, and Safari to ensure a broad user base can access its features. Cross-browser testing and responsive design principles are employed to provide a consistent experience across different platforms. Compatibility considerations also include mobile devices to accommodate users on the go.

5. Technology Used

Frontend: React

Develop a simple and responsive user interface using React.

Focus on building a few key pages (e.g., Login, Dashboard, Announcements, Payments).

Backend: Java with Spring Boot

Implement RESTful APIs using Spring Boot to handle the core business logic.

Integrate with MongoDB for storing user data, announcements, and payment records.

Database: Sql

Set up SQL to store user persons data, test score data, conversation on chat bot, tasks added by user, mood entry table.

Notification : Twilio

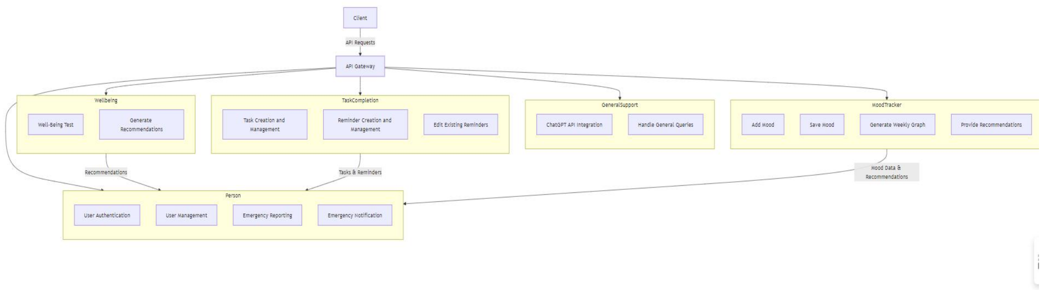
Helps to send notification to the particular mobile number.

Security: JWT

Implemented JWT to secure end points as well as for session management

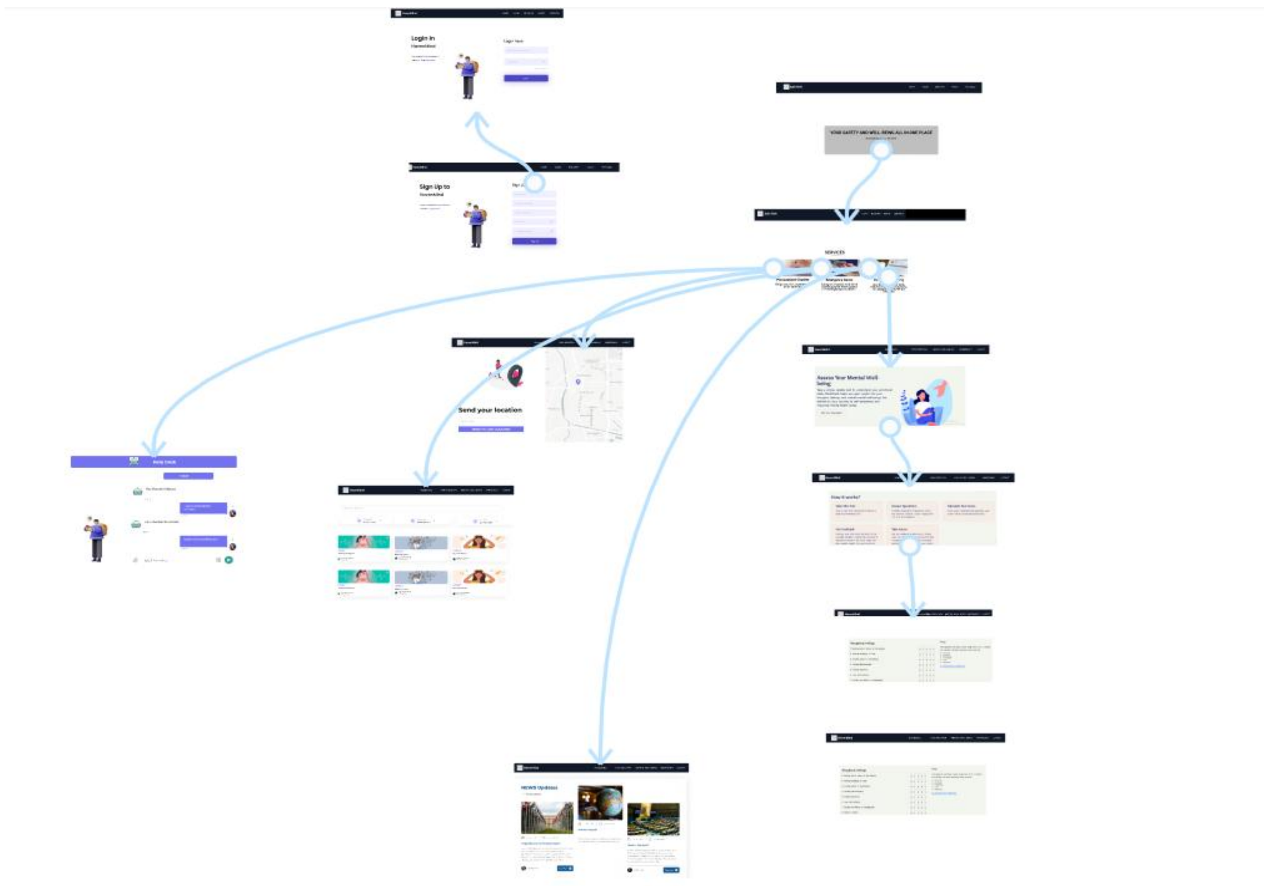
6. High-Level Design

- **Architecture Overview:**
    - The system follows a multi-tier architecture, which includes a Presentation Layer for user interactions, an Application Layer for business logic, and a Data Layer for data storage and management. This architecture supports separation of concerns and modularity, enhancing maintainability and scalability. The Presentation Layer uses React to provide a responsive UI, while the Application Layer is managed by Java Spring Boot.



- **Functional Specifications:**
    - Detailed specifications for each feature are provided, including how the Overview tab displays key information, how the Mood Tracker visualizes trends, and how the SOS Help feature integrates with location services. Each module is designed to interact seamlessly with others to provide a cohesive user experience. Functional specifications also include data flow and interaction diagrams.

- **System Interactions (Wireframe Diagrams):**
    - Wireframe diagrams illustrate the layout and functionality of the user interface, including key components such as the dashboard, mood tracker, and emergency help screens. These diagrams are created using tools like Figma to visualize user interactions and design workflows. They provide a blueprint for frontend development and user experience design.

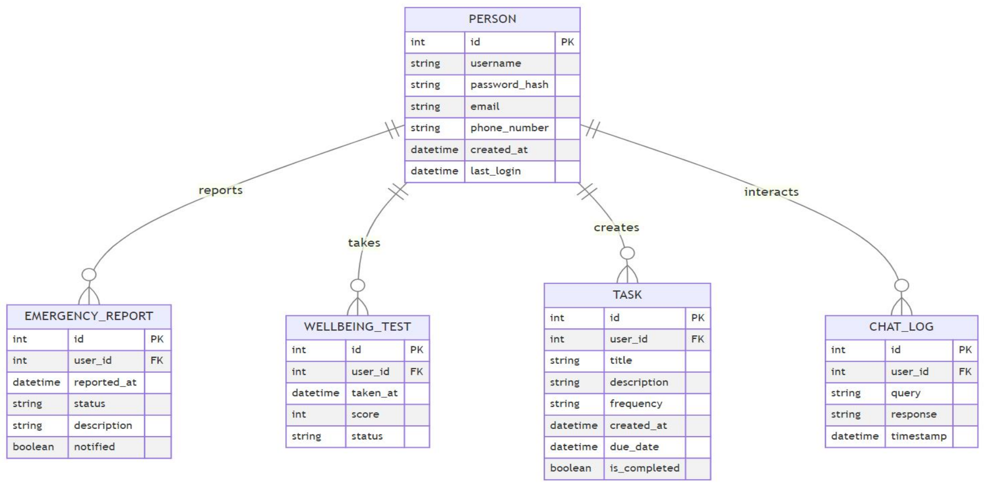


- **Performance Considerations:**
  - Benchmarks for performance are established to ensure that the application meets user expectations for speed and responsiveness. Load testing results are analyzed to identify potential bottlenecks and areas for optimization. Strategies such as caching, database indexing, and efficient query design are implemented to enhance performance and scalability.

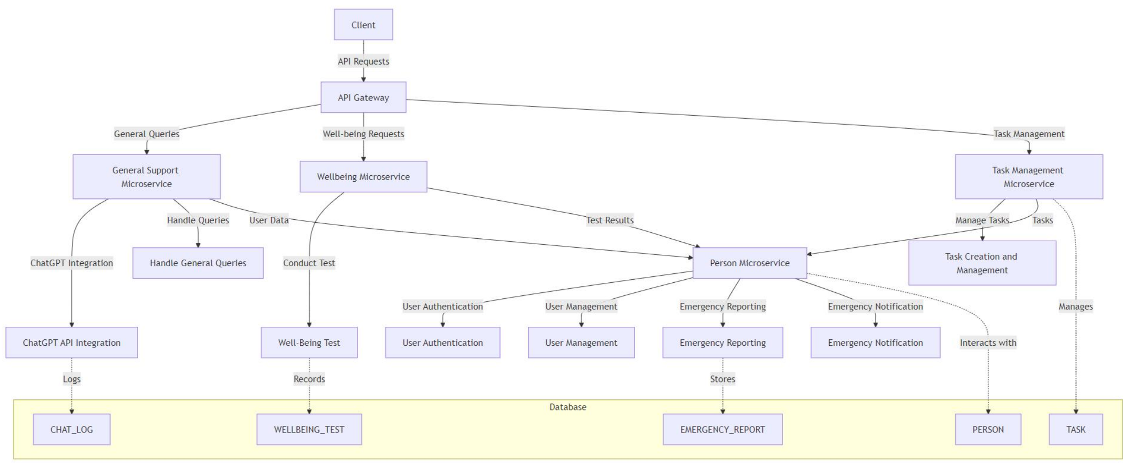
- **Technology Stack:**
  - A summary of the technology stack used in the project, including ReactJS for the frontend, Java Spring Boot for the backend, MySQL for data storage, and Docker, Kubernetes, and AWS for deployment and scaling. Each technology is chosen for its suitability in meeting the system's performance, scalability, and reliability requirements.

7. Low-Level Design

- **Database Diagrams:**
  - The database schema is detailed, including tables for user profiles, mood entries, recommendations, and emergency contacts. Relationships between tables are defined to ensure data integrity and efficient querying. Entity-relationship diagrams (ERDs) illustrate the structure and connections within the database.



- **Detailed Component Specification:**
    - Specifications for individual components are provided, including classes and functions for managing mood entries, handling SOS requests, and generating recommendations. Each component is designed with clear interfaces and responsibilities to support modular development and easy maintenance. Detailed component diagrams and code examples are included.



- **Interface Definitions:**
    - Interface definitions outline how different modules and components interact with each other, including API endpoints and data formats. This section defines inputs, outputs, and data exchange protocols between the frontend and backend. API documentation provides details on request methods, parameters, and response structures.
- **Resource Allocation:**
    - Estimates for memory, storage, and processing requirements are provided to ensure adequate resources are allocated for peak loads. Provisions for scaling and backups are included to accommodate future growth and ensure data protection. Resource allocation strategies support system reliability and performance.
- **Error Handling and Recovery:**

    - A centralized error logging system is implemented to capture and track errors effectively. Automatic recovery mechanisms and failover procedures are in place to ensure system resilience and minimize downtime. User feedback mechanisms are designed to provide clear error messages and support options.

- **Security Considerations:**
    - Security measures include robust authentication using JWTs or OAuth, encryption of sensitive data, and regular security audits. Strategies for protecting user data and preventing breaches are detailed. Security best practices are followed to ensure the application remains secure against potential threats and vulnerabilities.