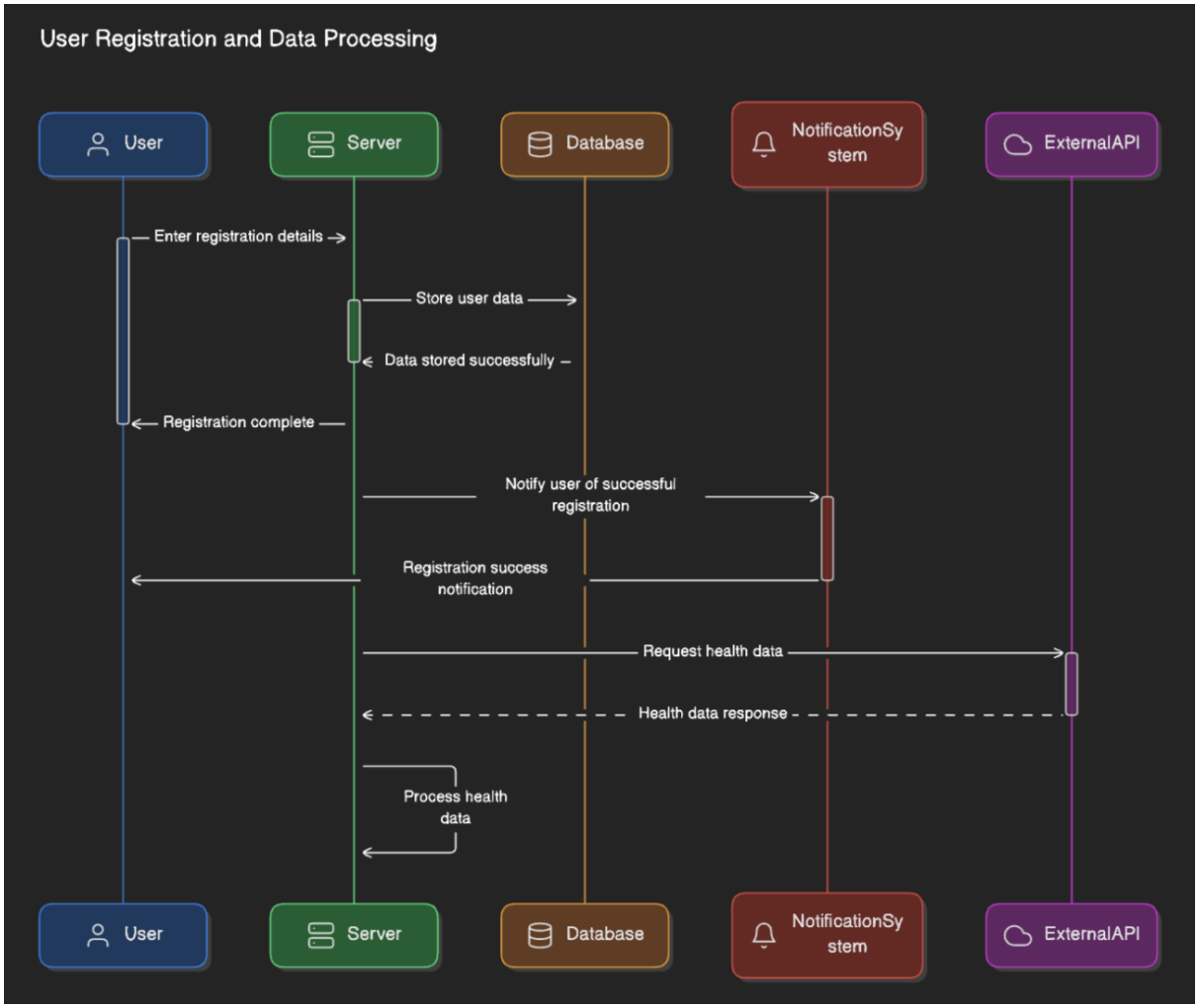
Team Project Deliverable 2 – Requirements Specifications

Course: CSCE 5430 (Fall 2024)



Overall Structure of the System:

- **Subsystems**: Each box or entity in the diagram should represent a key subsystem of your project (User Interface, Backend Processing, Database, External APIs).

- **Connections**: Arrows should show how each subsystem interacts with others, such as data flows or communications.

Brief Descriptions of Subsystems:

1.1. User Interface (UI) Subsystem

The **UI** is the front-end interface that allows users to interact with the system. This can be either a web-based platform. The UI subsystem helps:

- **User Interaction**: Allows users to input health data, manage appointments, and view health trends.

- **Data Visualization**: Provides charts, graphs, and dashboards for users to interpret their health data.
- **Ease of Access**: Ensures that the platform is user-friendly, accessible, and intuitive, making it easy for users of all ages and technical abilities to navigate the system.

The UI subsystem must be tightly integrated with the backend services to provide real-time data synchronization and updates to users.

1.2. Backend Processing Subsystem

This subsystem manages the core logic of the software. It includes:

- **Data Processing**: It processes health data (like heart rate, blood pressure, etc.) collected from users.
- **Business Logic**: Implements the algorithms that generate insights, and trigger alerts.
- **Interaction with the Database**: Retrieves and updates data stored in the database, ensuring that the user's data is always up-to-date.

1.3. Database Subsystem

This is where all user data is stored, including personal information, health metrics, logs, and historical trends. The **Database Subsystem** ensures:

- **Data Integrity**: Maintains consistency and accuracy of user health data.
- **Data Retrieval**: Quickly retrieves data for the UI to display to the user or for the backend to process.
- **Scalability**: Able to store large amounts of data efficiently and scale as the number of users grows minimum of 10000 users.

This subsystem is crucial as it handles all the storage and querying operations. Without a reliable database, the system would be unable to provide accurate historical data or perform real-time analysis.

1.4. Notification and Alert Subsystem

This subsystem handles all user notifications and alerts, such as:

- **Reminders for Medication**: Sends reminders based on user-set schedules or healthcare provider inputs.
- **Health Alerts**: Triggers alerts when abnormal health patterns are detected.

- **Customization**: Allows users to customize which alerts or notifications they receive and how they receive them (email, or push notifications).

The Notification Subsystem helps in increasing user engagement with health recommendations by keeping them informed and proactive.

Requirements Specifications:

User Registration and Authentication

- **Database:** We can use a **SQL-based database** for storing user credentials, ensuring security and scalability.
- **Functions:**
  - **Sign-Up:** A `POST` API to store user credentials securely, using hashing algorithms like **bcrypt** to encrypt passwords.
  - **Login:** A `POST` API to authenticate users, verifying their credentials against the stored hashed password.
  - **Password Recovery:** A function that sends a password reset link to the user's email.
  - **MFA (Multi-Factor Authentication):** Can be implemented using APIs like **Twilio** for SMS or **Google Authenticator** for one-time passwords (OTPs).

Health Data Entry

- **Database:** The health data can be stored for structured data.
- **Functions:**
  - **Manual Entry:** A `POST` API that takes user input and stores it in the database.

Health Metrics Data Collection:

1. Weight
2. Height
3. Blood Pressure (Systolic/Diastolic)
4. Heart Rate
5. Body Temperature
6. Body Mass Index (BMI)
7. Blood Glucose Level
8. Cholesterol Levels (Total, HDL, LDL, Triglycerides)
9. Oxygen Saturation
10. Activity Level
11. Dietary Intake (Calories, Macronutrients)
12. Sleep Patterns (Average Hours, Quality)
13. Medications (Name, Dosage, Frequency)
14. Symptoms or Concerns

2.2 Non-Functional Requirements

- **Performance:** The system must process data and generate health insights within a few seconds, ensuring a responsive experience that builds user trust.
- **Security:** All user data must be encrypted both in transit and at rest, complying with data privacy regulations such as HIPAA. This protects sensitive health information from unauthorized access, maintaining user trust and legal compliance.
- **Scalability:** The system should handle increased loads as more users join the platform, allowing it to grow and serve a larger user base without performance degradation.
- **Usability:** The platform must be user-friendly for both tech-savvy individuals and those with limited technical skills. An intuitive design will enhance user engagement and accessibility.

2.3 Interfaces

- **User Interface:** The front-end interface will enable users to interact with the system, input data, and receive insights. A smooth and intuitive design will enhance the user experience.
- **Backend and Database Interface:** This component will manage communication between the backend and the database, ensuring data is stored and retrieved smoothly. It guarantees that information is readily available when needed.

Plan for Implementing the Project

The implementation is divided into three phases, each focusing on building essential functionalities progressively.

Phase 1: Core System Development

This phase lays the foundation of the platform by implementing critical functions such as user registration and health data collection.

- **User Registration & Authentication**:
    - Implement basic user registration and login using a SQL-based database to store user credentials securely.
    - Integrate **Google Authenticator** for multi-factor authentication (MFA) to enhance security.

- **Database Setup**:
    - Set up the SQL-based database schema to handle user data, including health metrics like blood pressure, glucose levels, heart rate, etc.
    - Ensure the database is optimized for handling multiple health data types and scalable for future growth.
- **Health Data Collection**:
    - Build the backend logic for collecting and storing health data from wearables (if applicable) or manual data entry.
    - Integrate APIs or forms for data input via the **Streamlit** UI, allowing users to enter information such as weight, blood pressure, or glucose levels.

Objective: By the end of Phase 1, users should be able to register, authenticate, and securely input their health data, which will be stored in the system.

Phase 2: Health Insights and Notification System

This phase focuses on analyzing health data, identifying trends, and adding notification systems to keep users informed.

- **Health Data Analysis**:
    - Implement algorithms to analyze the collected health data and identify trends (e.g., blood pressure patterns, glucose level fluctuations).
    - Visualize the data in the **Streamlit** UI using charts, graphs, and dashboards, making it easier for users to interpret their health metrics.
- **Notifications & Alerts**:
    - Create a notification system that alerts users about important health events (e.g., abnormal heart rate) or sends reminders for medications and appointments.
    - Allow users to customize their notification preferences, selecting between email or push notifications.
- **Customizable Alerts**:
    - Develop a feature where users can customize the types of alerts they receive, enabling them to set thresholds for health metrics (e.g., an alert when blood glucose goes above a certain level).

Objective: By the end of Phase 2, users will receive health insights through the system and receive notifications or alerts based on the health data trends and thresholds.

Phase 3: Full Feature Rollout

The final phase includes the development of the advanced features such as a full health dashboard, machine learning models for predictive analysis, and a chatbot for user interaction.

- **Complete Health Dashboard**:
    - Finalize and refine the health dashboard in the **Streamlit** interface. Users should be able to track multiple health metrics over time, compare trends, and get a comprehensive view of their health.
    - Add advanced data visualization tools for users to drill down into specific metrics and time frames.
- **Machine Learning for Predictive Health Analysis**:
    - Implement machine learning models to predict potential health risks based on user data (e.g., predict risks of hypertension or diabetes based on current trends).
    - Provide users with proactive insights about their health, helping them take preventive measures.
- **Chatbot for User Support**:
    - Develop a simple chatbot using a pre-trained model (e.g., IBM Watson or a basic conversational AI) to assist users with queries such as understanding their health trends or how to use the platform.
    - Integrate the chatbot into the **Streamlit** UI for easy access.

Objective: By the end of Phase 3, the system will include predictive health insights, a full health dashboard, and an integrated chatbot to guide users through the platform.

Summary of the Phases:

1. **Phase 1**: Core System – User registration, authentication, health data collection.
2. **Phase 2**: Insights & Notifications – Health data analysis, customizable alerts.
3. **Phase 3**: Full Rollout – Health dashboard, machine learning models, and chatbot integration.

Member contribution table:

Member Name	Contribution Description	Overall Contribution (%)	Note
Jaideep Tripurani	Authored the introduction and requirements section	15%	Focused on project scope and goals.
Ajay Eedara	Developed the backend processing logic	20%	Ensured efficient data handling.
Tagore Hari Prasad Chintamaneni	Designed the database schema and queries	15%	Optimized for scalability and performance.
Satish Velaga	Implemented user authentication and security features	20%	Integrated multi-factor authentication.
Siddhartha Alapati	Worked on the frontend UI and data visualization	10%	Created user-friendly interfaces.
Devendra Kumar Gaddipati	Contributed to health data analysis algorithms	10%	Focused on generating actionable insights.
Sai Shruthik Errammagari	Managed API integrations for wearable devices	5%	Ensured seamless data syncing.
Sai Venkata Manish Lingamallu	Compiled the report, edited final submission, and developed the chatbot	5%	Created a simple chatbot for user support.
Ajay Kumar Aitha	Assisted with testing and debugging	5%	Ensured the platform's functionality.

Notes

- **Jaideep Tripurani**: Contributed to aligning the project with user needs and regulatory requirements.
- **Ajay Eedara**: Utilized robust programming practices to ensure backend efficiency.
- **Tagore Hari Prasad Chintamaneni**: Worked on ensuring data integrity and security.
- **Satish Velaga**: Conducted extensive research on best practices for user authentication.

- **Siddhartha Alapati**: Collaborated with users for feedback on UI design.
- **Devendra Kumar Gaddipati**: Engaged in iterative testing of algorithms for health data analysis.
- **Sai Shruthik Errammagari**: Actively collaborated with wearable device vendors for API access.
- **Sai Venkata Manish Lingamallu**: Developed chatbot capabilities to enhance user engagement.
- **Ajay Kumar Aitha**: Led testing sessions and addressed bugs reported by team members.