Software Design Specification
for Active Site

Prepared by:

Núria Mitjavila
Mònica Torner
Marta Ortigas

ESCI, School of International Studies, Barcelona
Software Engineering

Daniel Soto Álvarez

Barcelona, June 2023
VERSION 1.0

Contents

Introduction

Project repository with all the data: `https://github.com/NuriaMitjavila/Software-Engineering.git`

1.1 Document Purpose

The purpose of this document is to provide a detailed and comprehensive overview of all the design aspects of the ActiveSite software that we have been developing. It aims to describe the software architecture, the user interface design, the database design, the API design, the requirements, etc. This document serves as a blueprint for the development team, guiding them in implementing the system according to the specified design.

1.2 Product Purpose

The purpose of the software is to create artificial Intelligence for the detection of glucose, hydration, and other nutritional and health items with continuous monitoring from a sensor. The global goals are to provide an easy way to monitor nutrition and health, provide personalized recommendations based on real-time data analysis, help users achieve their health goals, promote health culture by making it easy for users to keep their nutrition and health and improve app's functionality and user experience based on user feedback.

1.3 Document Scope

This document covers all the relevant software design aspects of the ActiveSite project. It includes the architecture, the user interface design, the software requirements, the use case diagram, the dynamic diagrams (activity and sequence), the class diagram, etc. The document outlines the boundaries within which the development team operates and provides guidance for building the system.

1.4 Product Scope

The project scope for the Active Site app includes the development of an AI-based system for continuous monitoring and analysis of glucose, hydration, and other nutritional and health items using a sensor-based approach. The app is designed to provide personalized recommendations and insights to users based on their individual health needs and goals.

1.5 Document Conventions

The document follows a set of conventions to ensure consistency and clarity. These conventions include formatting guidelines (font, size, indentation), naming conventions (e.g., useCase), notation standards (UML diagrams), and any other standards specific to the project. These conventions help maintain a uniform and professional look throughout the document, making it easier to read and understand. The intended audience for this document is the development team, the project stakeholders, the designers, and anyone involved in the software development process. The document provides a comprehensive understanding of the system's design.

1.6 Definitions and acronyms

- Active Site: The name of the mobile application being developed as part of this project.
- Glucose: A type of sugar found in the blood that provides energy to the body's cells.
- Hydration: The process of adding water to the body or the state of having enough water in the body.
- Artificial Intelligence: A branch of computer science that involves the development of algorithms.
- Machine Learning: A subset of AI that focuses on the development of algorithms and statistical models.

- SRS: Software Requirements Specification
- AI: Artificial Intelligence
- ML: Machine Learning
- iOS: iPhone Operating System

1.7 Assumptions

- The development team has access to the necessary hardware and software tools required for the development.
- The development team has the necessary skills to develop a mobile application with artificial intelligence.
- The application will be developed in English language only.

Overall Description

2.1 Perspective

Active Site is a mobile application designed to help users monitor and manage their glucose, hydration, and other nutritional and health data. The app uses sensor-based technology and artificial intelligence algorithms to provide real-time analysis of user data and personalized recommendations for improving their health.

2.2 Functions

Continuous monitoring of glucose and hydration levels using a wearable sensor. Analysis of nutritional intake and activity levels. Real-time alerts and notifications to remind users to take medication or drink water and personalized recommendations.

2.3 User Characteristics

Diabetes patients familiar with monitoring their glucose levels. Fitness enthusiasts who may have experience using mobile applications to track their workouts and nutrition. Health-conscious individuals who may be new to monitoring their health data but are motivated to make positive changes to their lifestyle

2.4 Constraints

The app must comply with relevant data privacy and security. Must be compatible with iOS and Android mobile devices and wearable sensors. And it must be with a user-friendly interface.

2.5 Assumptions

Users will have access to compatible mobile devices and wearable sensors. Users will provide accurate and up-to-date information about their health and wellness. The app will be developed and maintained using industry best practices for software development and data privacy and security. The app may require regular updates and maintenance to ensure compatibility with new mobile devices and operating systems. Requires user feedback and input to continually improve its performance and effectiveness.

Requirements

Functional	Non-functional
User registration – F-REQ001 Data collection – F-REQ002 Data processing – F-REQ003 User interface – F-REQ004 Sensor pairing – F-REQ005 Data Storage and Security – F-REQ006 User Profile Management – F-REQ007 External Panel (for Doctors) – F-REQ008 Notifications and Alerts – F-REQ009	Performance – NF-REQ001 Usability – NF-REQ002 Reliability – NF-REQ003 Compatibility – NF-REQ004

3.1 Functional

F-REQ001	User registration	Version (v1.2)
Users can: - Create an account (name, surnames, email, cell phone num. and selection mode) - Provide personal information (age, gender, height and medical history). - Set fitness and health goals. - View and edit their personal information and goals.		
Relationship: None		

F-REQ002	Data collection	Version (v1.3)
The sensor gains the data and sends it to the app. Type of data the sensor gains: - Glucose and hydration levels - Heart rate - Sleep cycles - Fitness levels (this will vary depending on the mode that the user selected) - Meal habits (this will vary depending on the mode that the user selected) The app can also gain data manually added by the user. The data is validated and stored securely.		
Relationship: None		

F-REQ003	Data processing	Version (v1.2)
- Process the data gained by the sensor or the user. - Provide personalized recommendations and insights for the user. - Alert the user if any parameter is not inside the regular values (this will vary depending on the mode that the user selected). - Real-time analysis of the parameters mentioned in F-REQ002.		
Relationship: F-REQ002		

F-REQ004	User interface	Version (v1.3)
The app includes a user interface that allows users to: - Input data manually. - View real-time metrics (metrics of the parameters presented in F-REQ002). - Access personalized recommendations and insights. - Set goals related to their health and nutrition. - Customize alerts and notifications based on their preferences - Visualization of user data over time, such as graphs or charts.		
Relationship: F-REQ003		

F-REQ005	Sensor pairing	Version (v1.1)
The sensor pairs for the first time and always with the mobile phone of the user and therefore with the app via Bluetooth. If Bluetooth is deactivated, the sensor will not be pairing, so no data will be recollected.		
Relationship: F-REQ001		

F-REQ006	Data Storage and Security	Version (v1.2)
- Store user data securely, ensuring privacy and compliance with data protection regulations. - Implement appropriate data encryption and access controls. - Regularly back up user data to prevent data loss.		
Relationship: None		

F-REQ007	User Profile Management	Version (v1.1)
- Allow users to view and edit their personal information and goals. - Enable users to modify their health and nutrition goals as needed.		
Relationship: F-REQ001 and F-REQ004		

F-REQ008	External Panel (for Doctors)	Version (v1.1)
- Provide an external panel for doctors or healthcare professionals to observe patient data obtained from the app. - Allow doctors to monitor and analyze the health metrics of their patients remotely. - Ensure secure and restricted access to patient data for authorized healthcare professionals.		
Relationship: F-REQ006		

F-REQ009	Notifications and Alerts	Version (v1.1)
- Deliver alerts and notifications to users based on their preferences and customizable settings. - Notify users of critical health events or deviations from their defined thresholds.		
Relationship: F-REQ003		

3.2 Non-functional

NF-REQ001	Performance	Version (v1.1)
The app must be able to: - Process large amounts of data quickly and accurately. - Handle high levels of user traffic without experiencing significant slowdowns or crashes.		
Relationship: F-REQ003		

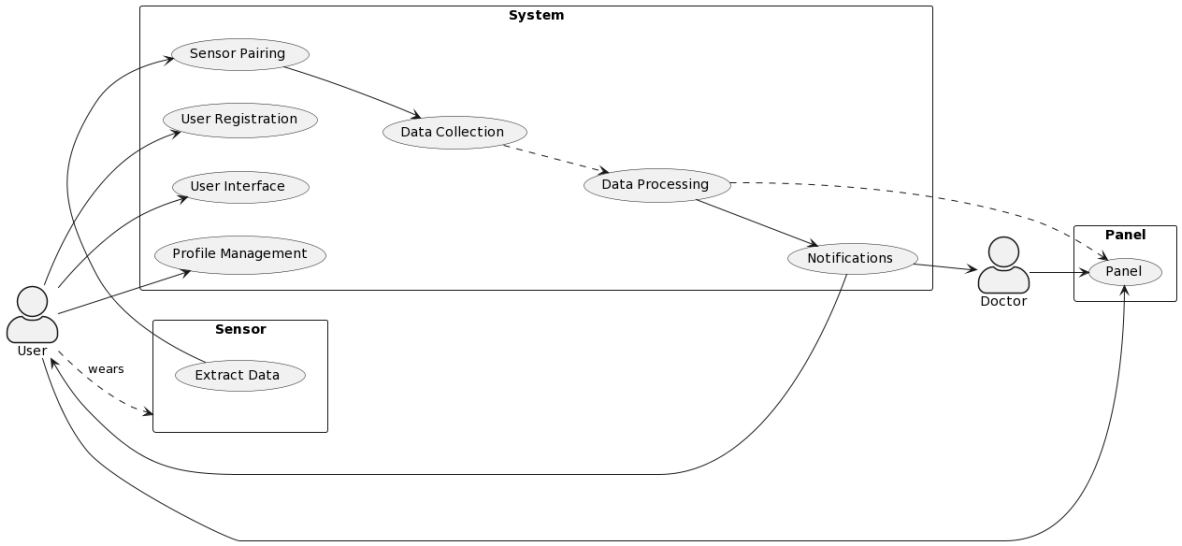
NF-REQ002	Usability	Version (v1.1)
The app must: - Be easy to use and navigate (clear interface). - Have clear instructions and an intuitive design. - Be accessible to users with disabilities (features such as text-to-speech and high-contrast modes).		
Relationship: F-REQ004		

NF-REQ003	Reliability	Version (v1.1)
The app must be reliable and available at all times. Have minimal downtime for maintenance or updates (one update per month if necessary). The app must be able to recover quickly from any errors or malfunctions.		
Relationship: None		

NF-REQ004	Compatibility	Version (v1.1)
The app must be compatible with a wide range of devices and operating systems. App available for Windows, iOS, Android and Linux.		
Relationship: None		

Diagrams

4.1 Use Case Diagram



1. Actors:
- User: Represents the individuals who interact with the app and that wears a sensor to monitor their data.
- Doctor: Represents healthcare professionals like family doctors or nurses which has access to an external panel

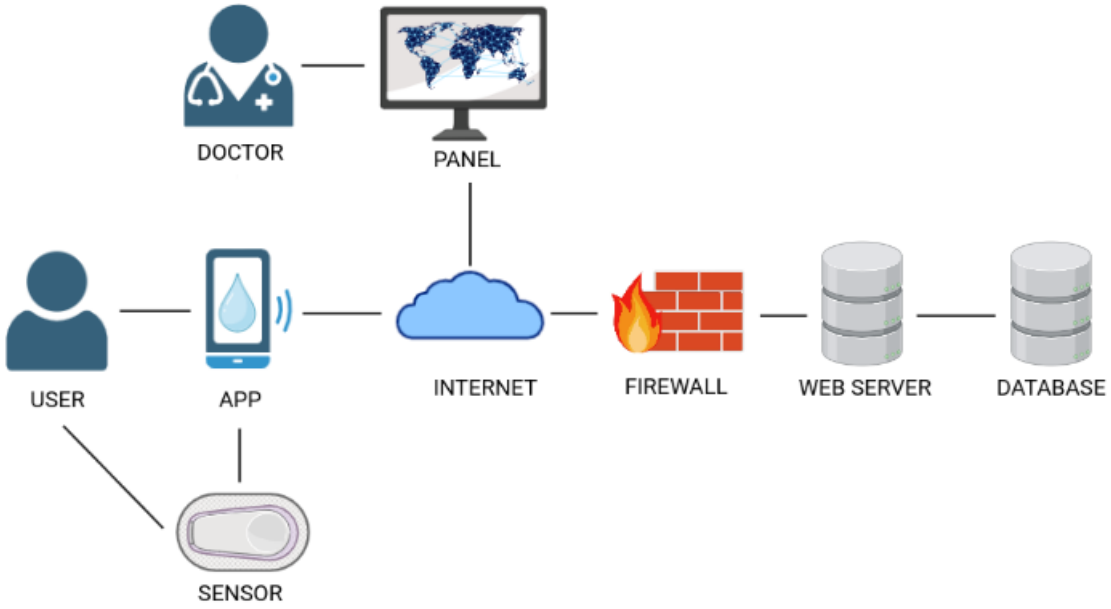
2. System:
- User Registration: represents the functionality for users to create an account with personal information.
- User Interface: interaction between users and the app's user interface (set goals, alerts, view metrics...).
- Sensor pairing: represents the way the system (the phone's app) is capturing data from the sensor.
- Data Collection: collection of health-related data from users with metrics such as glucose, hydration...
- Data Processing: represents the processing and analysis of the collected data (meaningful information).
- Notifications: represents the alerts that can arrive to the user and the doctor from the processed data.

3. Panel:
- This use case refers to an external panel that allows doctors or healthcare professionals to observe patient data obtained from the app. It enables doctors to monitor and analyze the patients health metrics.

4. Sensor:
- Extract Data: This use case represents the functionality of extracting data from a sensor.

4.2 Architectural design

The system architecture is mainly divided into the Mobile App (Active site), the sensor and the database.



1. Sensor:
- Represents the physical device that captures health-related data such as glucose levels, hydration levels, etc.
- It uses specific technologies like biosensors or wearable devices to measure and collect data.
- The sensor communicates with the mobile app, transmitting real-time data for processing and analyzing.
- Employs industry-standard protocols like Bluetooth or Wi-Fi to establish a connection with the mobile app.

2. User:
- Represents the individuals who interact with the app and utilize the sensor data for monitoring their health
- Users can create an account within the app, providing personal information such as age, gender, height, etc.
- They can set goals related to their health and nutrition and track their progress over time.
- Users can view and edit their personal information and goals through the app's user interface.
- They can also access real-time and historical sensor data captured by the sensor and visualize it in the app.

3. App:
- The app component, known as ActiveSite, is the software application installed on the user's mobile device.
- It provides the user interface and functionality for users to interact with the system.
- The app allows users to register an account, log in, and securely store their personal information and goals.
- Provide features like data visualization, trend analysis, goal tracking, and personalized recommendations.
- It communicates with the web server through APIs to synchronize data and perform various operations.
- The app relies on an internet connection to exchange data with the web server and provide real-time updates.

4. Panel:
- The panel component is a separate interface designed for doctors or healthcare professionals.
- Provides a platform for doctors to access and analyze the data of their patients collected by the app.
- Doctors can view patient data, including glucose levels, hydration levels, and health metrics.
- The panel offers advanced visualization tools, trend analysis, and data filtering options.
- Monitor the progress of their patients, identify patterns or anomalies, and recommendations.

5. Doctor:
- Represents healthcare professionals who utilize the panel interface to monitor and manage patients.
- They have authorized access to the panel, allowing them to securely log in and access patient data.
- Doctors can view and analyze the health metrics and trends of their patients.
- They can monitor multiple patients simultaneously and track their progress over time.
- The doctor can customize alerts and notifications based on specific patient conditions or thresholds.

6. Internet:
- Represents the network infrastructure that allows communication between the app, web server, and other
- It enables the exchange of data between the user's mobile device and the web server.
- Can be established through various means, including Wi-Fi, cellular networks, or other technologies.
- The app uses the internet to transmit data to the web server and receive responses or updates in real-time.
- A stable and reliable internet connection is essential for data transmission and communication.

7. Firewall:
- Represents the set of tools, libraries, and protocols used to build and expose the APIs of the web server.
- Provides the infrastructure to define, implement, and manage the APIs that the app communicates with.
- Include features like request routing, authentication and authorization mechanisms and error handling.
- It ensures secure and standardized communication between the app and the web server.

8. Web Server:
- Receives API requests from the app and processes them to perform various operations.
- Interacts with the database system to retrieve or store data, ensuring data consistency and integrity.
- It employs an API framework to define and expose the APIs that the app utilizes.
- Implement additional security measures like encryption, secure protocols, and access control for privacy.

9. Database System:
- The database system is responsible for storing and managing the persistent data of the system.
- It includes tables or collections to store user profiles, personal information, goals, and historical sensor data.
- Ensures data integrity, concurrency control, and backup strategies to safeguard the stored data.
- Interacts with the database system to store and retrieve user-specific data, enabling personalized experiences
- Relational database management system or a NoSQL database.

4.3 User Registration

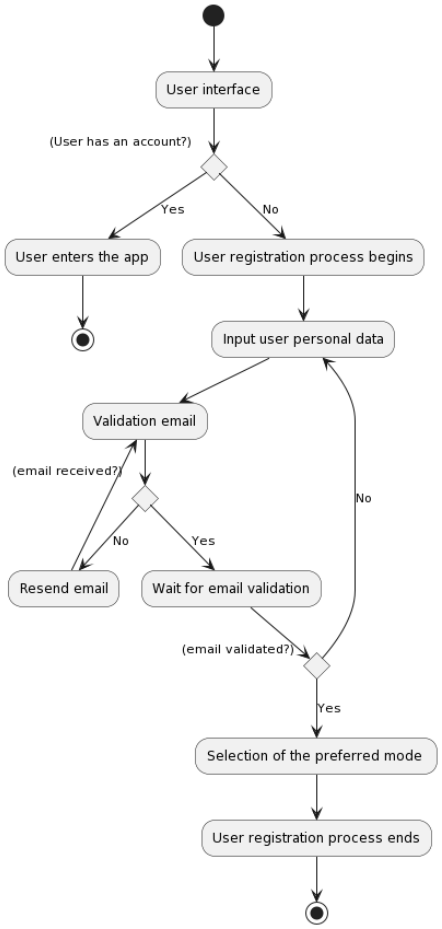


Figure 4.1: Activity diagram

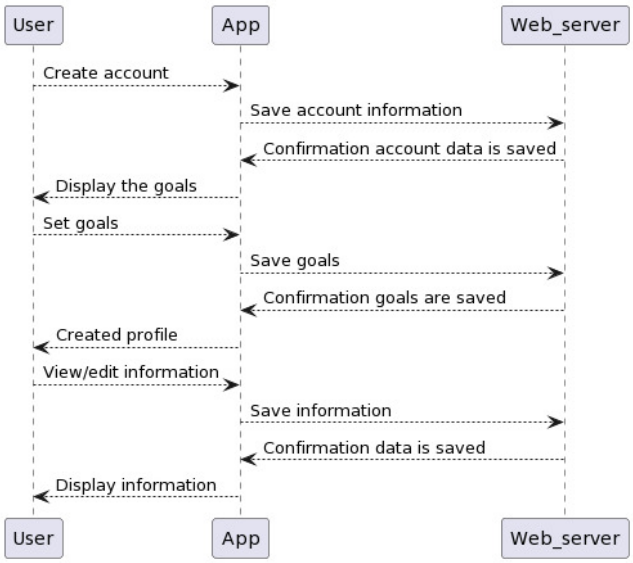


Figure 4.2: Sequence diagram

The user registration process involves capturing user input, validating the information provided, creating a user profile, setting goals, and allowing users to view and edit their profile information.

1. User Input: User provides the necessary information to create an account: username, email and password.

2. User Created: The system checks if the user account has been successfully created. If the account creation is successful, the process continues, if not, an error message is displayed, so the user can correct any errors.

3. Profile Information: Provide additional profile information, such as age, gender, height, and medical history.

4. Valid Information: The system validates the information provided by the user. If any required information is missing or there are validation errors, an error message is displayed, so that the user correct the information.

5. Set Goals: Once the user's profile information is validated, the user can set goals of health and nutrition.

6. View/Edit Information: the ability to view and edit their profile information and goals at any time.

The user registration process ensures that users provide the necessary information to create their account, verify the accuracy, set goals and have the flexibility to view and modify their information as required.

4.4 User interface

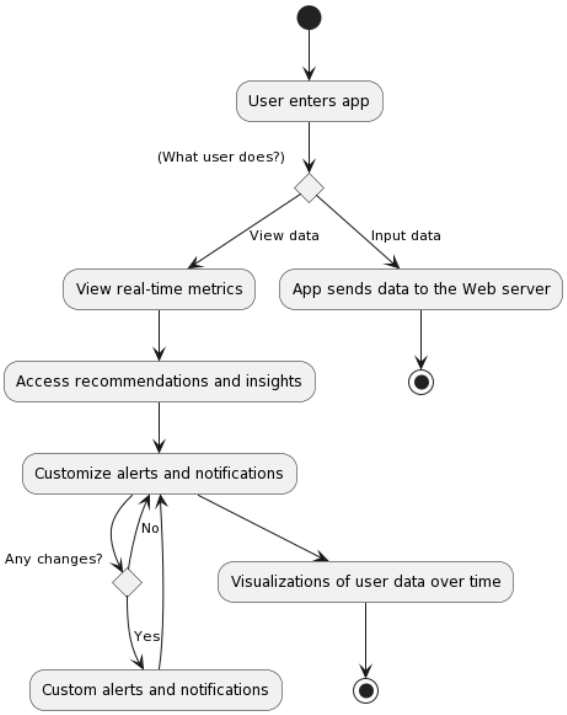


Figure 4.3: Activity diagram

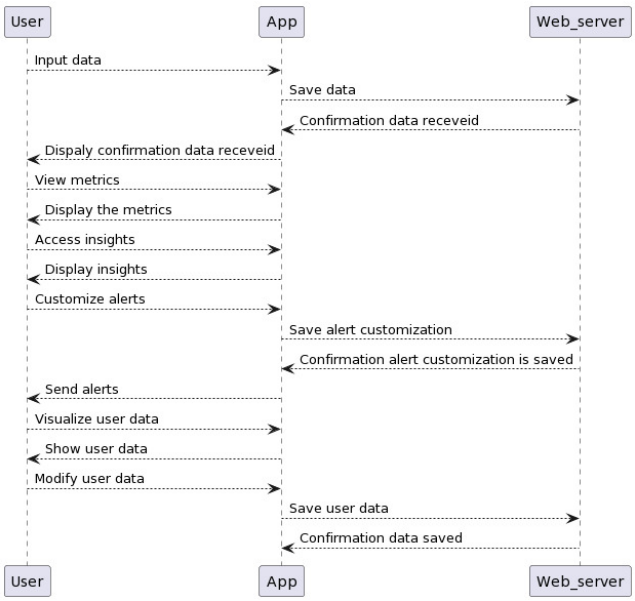


Figure 4.4: Sequence diagram

The user interface provides an intuitive and user-friendly platform for users to interact with the app.

1. Input Data: allows users to input their health-related data through dedicated input fields or predefined.
2. View Metrics: enables to access real-time metrics related to their health in an easy-to-understand format.
3. Access Insights: provides valuable recommendations and insights based on their input and historical data.
4. Customize Alerts: allows personalizing the notification settings related to their health and nutrition goals.
5. Visualize Data: provides comprehensive visualizations of their data over time; interactive charts, graphs...
6. Modify User Data: allows users to edit or update their personal information and goals.

The app focuses on simplicity, ease of use, and accessibility. Presents information in a visual manner, supports interactions, and empowers users to take control of their health and nutrition through personalized features.

4.5 Data processing

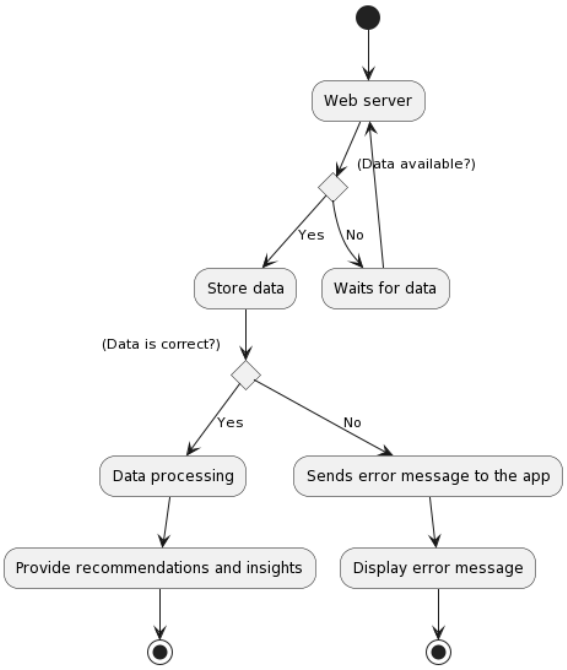


Figure 4.5: Activity diagram

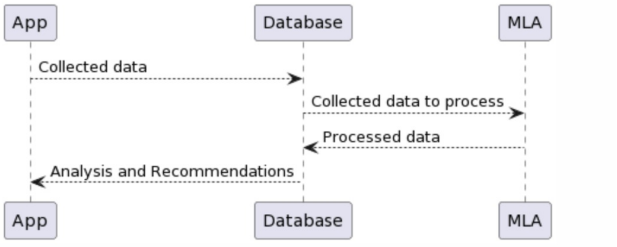


Figure 4.6: Sequence diagram

The data processing involves the utilization of machine learning algorithms to analyze and take relevant information from the collected user data. These activities aim to provide personalized recommendations and real-time analysis to the users for better health and nutrition management.

1. Machine Learning Algorithm: designed to process and analyze the data, identifying patterns, correlations, and trends. It can adapt and improve its analysis based on new incoming data and user interactions.

2. Store Data: the processed data is stored for future reference and analysis. The app maintains a secure storage system to store the user data in a structured format and that serves as a historical record that can be used for long-term analysis, trend identification, and generating personalized recommendations.

3. Provide Recommendations and Insights: based on the analysis performed by the machine learning algorithm, the app provides personalized recommendations and insights to the users. These recommendations are specific to each user's health and nutrition goals, preferences, and historical data.

4. Real-Time Analysis: enables the app to provide immediate feedback, alerts, and notifications based on the latest data inputs. The real-time analysis ensures that users receive timely information about their health metrics, allowing them to make proactive adjustments to their habits and behaviors.

The data processing activities combine machine learning algorithms, data storage, personalized recommendations, and real-time analysis to provide users with valuable insights into their health and nutrition. The app aims to empower users to make informed decisions and take proactive steps to achieve their health goals.

4.6 Data collection

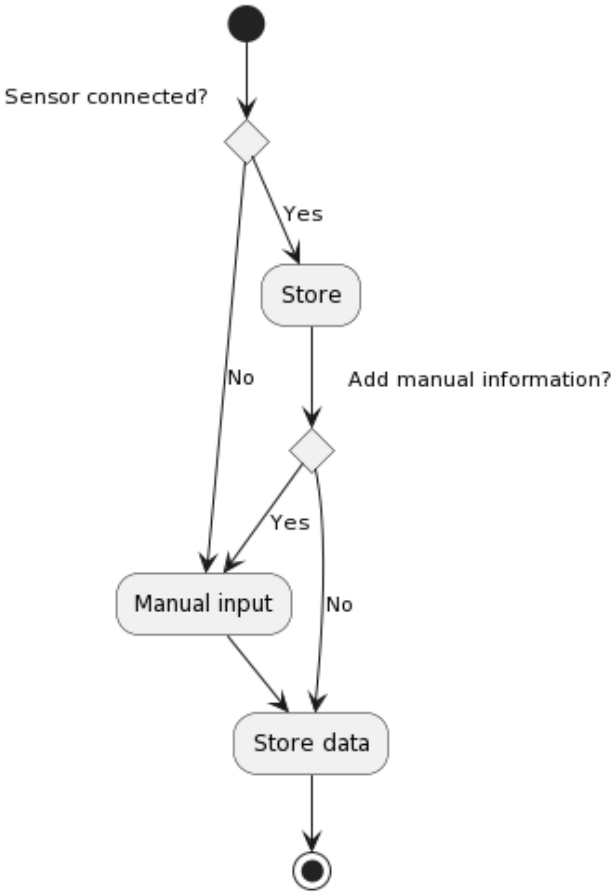


Figure 4.7: Activity diagram

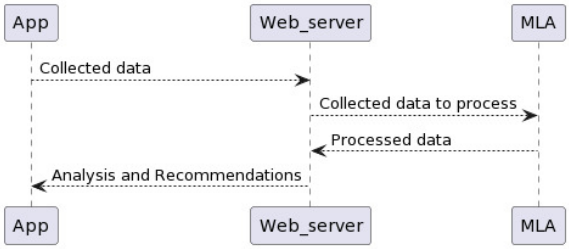


Figure 4.8: Sequence diagram

The data collection process involves capturing and storing user data, which can be collected either through a sensor or manual input.

1. Sensor Connection Check: the process begins by checking if a sensor is connected to the app.

2. Manual Input: if a sensor is not connected, the app allows users to manually input their health-related data. This can include information such as glucose levels, hydration levels, and other nutritional metrics.

3. Data Storage: after the data is collected, the app proceeds to store the data. The collected data is associated with the user's profile or account, ensuring that it is stored in a personalized manner. The storage mechanism may involve saving the data locally on the user's device or securely transmitting it to a remote server.

4. Data Collection Completion: After storing the collected data, the app returns to the starting point.

Overall, the data collection process offers flexibility for users to provide their health-related data either through a connected sensor or manual input. The app ensures the secure storage of the collected data.

4.7 Class Diagram

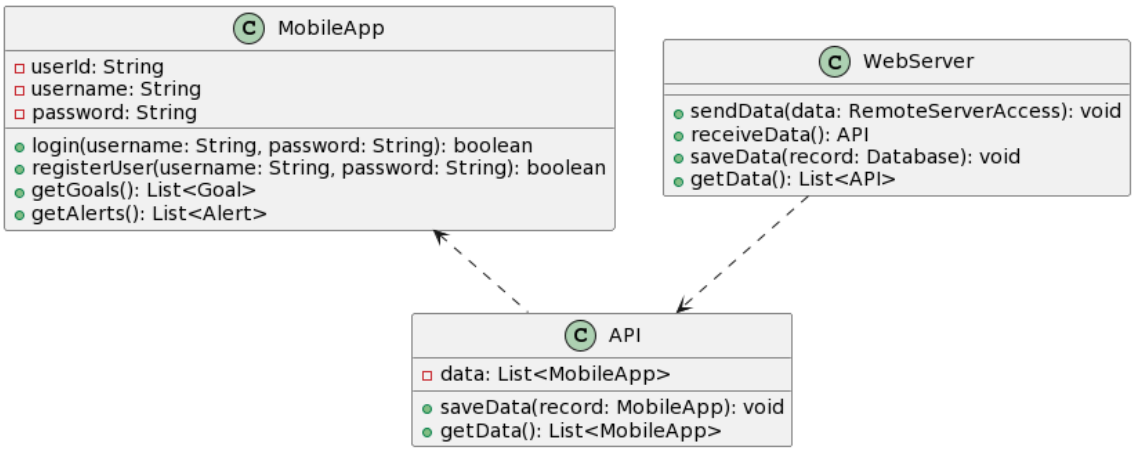


Figure 4.9: Global class diagram

This is the global structure of the class diagram. It is mainly divided in two parts, the mobile app and the web server, and they are connected by an API. From the mobile app, we have several types of data, like it can be the login information, with the username and the password, all saved in a specific and unique user ID. We also have some lists with diverse information as it can be for the user goals or the different alerts. Finally, we have the data, that is a list with all the processed data, obtained from the sensor.

This list with the processed data is the one that, with the API, is sent to the web server. The API, application programming interface, is a way for two or more computer programs to communicate with each other. Web APIs are a service accessed from client devices (Mobile Phones, Laptop, etc.) to a web server using the Hypertext Transfer Protocol (HTTP). This is the reason why for the API, we are having two lists, get data, that is obtained from the mobile app and the save data, to store precisely this data, to be sent to the web server.

Finally, there is the web server, which receives the data from the API, has a place to save all this obtained data, and also has the variable send data, that which be useful for displaying the data in a panel, as we previously explained. This is this global diagram but with different classes descending from those:

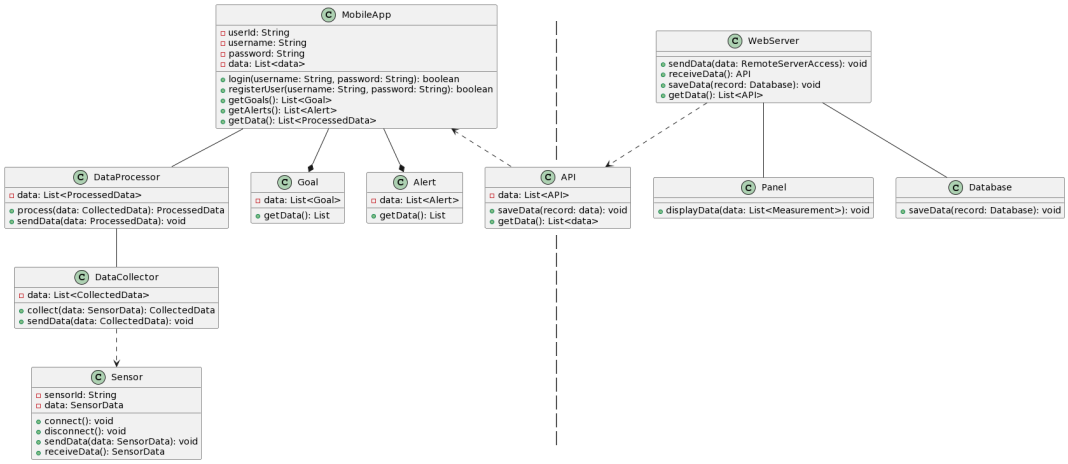


Figure 4.10: Global class diagram

The following class diagram represents the relationships and properties of various classes in the system, the ones that have a relationship with the mobile app. Here's an explanation of each class:

1. MobileApp: Represents the mobile application used by users to interact with the system. Stores user-related information such as userId, username, and password. Manages data, including goals, alerts, and processed data. Provides methods for user authentication, registration, and accessing data.

2. Sensor: Represents the physical sensor device used to collect data. Stores sensor-specific information such as sensorId. Provides methods to connect and disconnect from the sensor, send data, and receive data.

3. Data Collector: Represents the component responsible for collecting raw sensor data. Stores collected data in a list. Provides methods to collect sensor data and send collected data.

4. Data Processor: Represents the component responsible for processing collected data. Stores processed data in a list. Provides methods to process collected data and send processed data.

5. API: Represents an external API used for data exchange. Stores API-related information. Provides methods to save data records and retrieve data.

6. Goal: Represents a goal set by a user. Stores goal-related information. Provides methods to retrieve goal data.

7. Alert: Represents an alert configured by a user. Stores alert-related information. Provides methods to retrieve alert data.

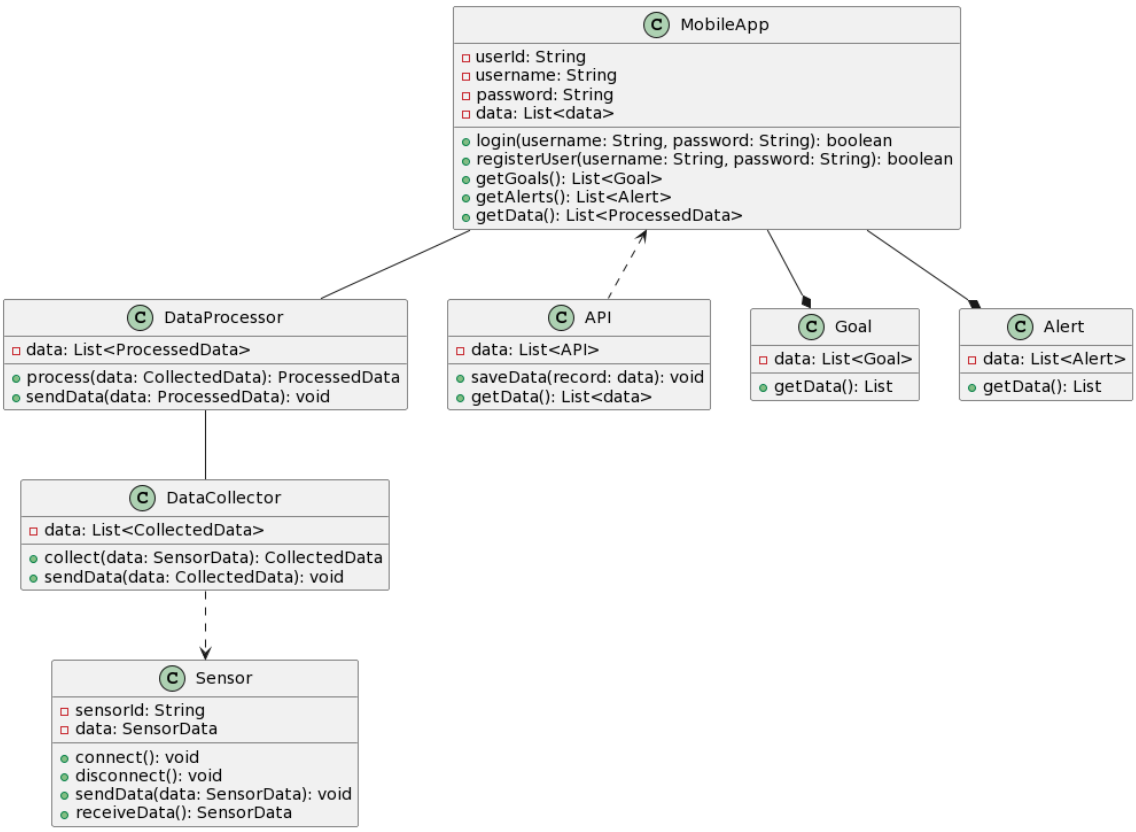


Figure 4.11: App class diagram

The following class diagram represents the relationships and properties of various classes in the system, the ones that have a relationship with the web server. Here's an explanation of each class:

1. Panel: Represents a panel used by the doctor or other authorized personnel to view data. Provides a method, displayData, to present a list of measurements.

2. API: Represents an external API used for data exchange. Stores API-related information. Provides methods to save data records and retrieve data.

3. WebServer: Represents a web server that serves as an intermediary between the app, panel, and database. Provides methods to send and receive data, as well as save and retrieve API data. Has an association with the Panel class, indicating that the web server is connected to the panel. Has an aggregation relationship with the API class, indicating that the web server utilizes the API. Has a composition relationship with the Database class, indicating that it owns and manages the database.

4. Database: Represents a database used for storing data. Provides a method, saveData, to store records.

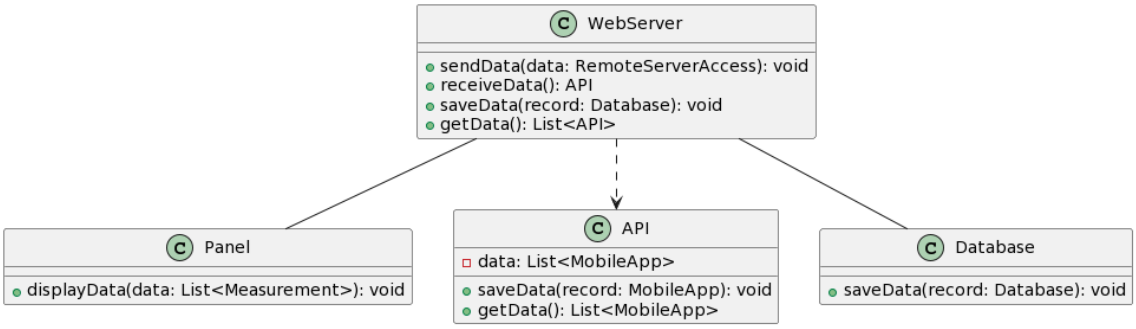


Figure 4.12: Web class diagram

The associations and dependencies in the diagram are as follows:

- The Panel class has an association with the WebServer class, indicating that the panel interacts with the web server to display data.

- The WebServer class has an association with the API class, indicating that it communicates with the API for data exchange.

- The WebServer class has a composition relationship with the Database class, indicating that it owns and manages the database.

Overall, this class diagram illustrates the classes involved in the system, their relationships, and their respective methods and properties. It showcases the architecture and data flow between the panel, web server, API, and database components.

Conclusions

1. Purpose:
The purpose of the SDS document is to provide a detailed overview of the design aspects of the ActiveSite application. It outlines the architecture, components, interactions, and requirements necessary for the successful development of the software.

2. Scope:
The SDS document covers the entire scope of the project, including the architecture, user interface, data collection and processing, user registration. It also has the use case diagrams, the class diagrams and the functional and non-functional requirements. It serves as a comprehensive guide for the design and implementation of the ActiveSite application.

3. Architecture:
The architecture of the ActiveSite application follows a client-server model, with the mobile app serving as the client, the web server acting as a middleware, and the database server handling data storage and retrieval. The integration of the sensor and the external panel provides real-time data collection and analysis capabilities.

4. Components and Interactions:
The ActiveSite application consists of several key components, including the mobile app, sensor, panel, data processor, and database. These components interact with each other to facilitate user registration, user inferface, data collection and data processing. The mobile app communicates with the sensor for real-time data capture, while the panel allows doctors to access patient data.

5. Requirements:
The SDS document outlines the functional requirements of the application, including user registration, data collection, data processing, user interface, sensor pairing, data storage and security... It also addresses non-functional requirements such as compatibility, performance, usability and reliability.

6. Designs:
The designs of the ActiveSite application are huge. In this SDS, we include the use case diagram, the architectural design, the class diagram and some activity and sequential diagrams of some requirements, like the user registration, the user interface, the data processing and the data collection. .

7. Conclusion:
The SDS document provides a comprehensive overview of the software design for the ActiveSite application. It gives an important reference for developers, designers, and stakeholders involved in the development process. With a clear understanding of the system architecture, components, and interactions, the project can proceed with the implementation phase in a structured and efficient manner.

By following the guidelines and design principles outlined in the SDS document, the ActiveSite application can be developed to meet the needs and expectations of users. Although in our case, the SDS document could be improved by explaining better and adding more requirements and diagrams, as well as adding more detailed information in the class diagram.