Project Aim, Objectives, Functional and Non-Functional Requirements

Ali Suhail 21072712

1. Project Aim

To design and implement a software solution focused on helping users achieve their fitness objectives. This involves developing a user-friendly platform for creating and customising workout routines, monitoring nutrition and workout progress, and offering health and fitness guidance. The main aim is to build an engaging and efficient fitness tool that encourages users to live healthier, more active lives.

2. Project Objectives

2.1. Implement a Central Database Management System (DBMS).

2.1.1. Objective: Create a DBMS using MySQL Workbench connected to the UWE server. The database will store predefined workouts, exercises, meals, food items and user login information.

2.2. Implement a Local Database.

2.2.1. Objective: Create a local database that contains all predefined information from the central database with the addition of user custom workouts, meals and watch data. All the updates from the central database are transferred to the local database.

2.3. Implement an Account Management (AM) Website.

2.3.1. Objective: Develop a website with React.js, enabling users to sign up and log in to access the workout tracker application. There are two user roles: Admin and User. Users can update their account information and provide feedback on the website. Admins possess the authority to modify user data, make changes to the database and modify their own personal information.

2.4. Feedback Tab for User Feedback.

2.4.1. Objective: Implement a feedback system for users to submit suggestions and issues. This will be implemented on the account management website.

2.5. Implement the Alistana Fitness and Nutrition Tracker (AFNT) Application.

2.5.1. Objective: Create an application that is used to track users' health goals. The application will be connected to the central and local database, AM website, and the Arduino watch. Objectives 2.6-2.11 are all objectives that come under AFNT.

2.6. Implement Personalized & Custom Workout Plan Functionality.

2.6.1. Objective: Create a function that generates workout plans based on user goals or users can create their own custom workout plans.

2.7. Implement Workout Tracker Functionality.

2.7.1. Objective: Develop a feature that allows users to record exercises, sets, reps, and rate/describe their performance.

2.8. Implement Nutrition Tracker Functionality.

2.8.1. Objective: Create a function that allows users to track their food and water intake.

2.9. Implement Body Progress Tracker Functionality.

2.9.1. Objective: Develop a function that allows users to see their body performance based on their workout plan. This feature will employ graphs and charts to present user advancements, such as a six-month graph depicting the one-repetition maximum for the Bench press.

2.10. Implement Body Measurement Tracker Functionality.

2.10.1. Objective: Develop a function that permits users to input their body measurements for the purpose of monitoring their weight, height, BMI, muscle mass, and fat mass. Implement visual representations through graphs and charts to track these metrics over time.

2.11. Implement Health & Nutrition Advice Functionality.

2.11.1. Objective: Develop a feature that grants users access to nutritional information and goal-based recommendations. This function should also offer tutorials on executing various exercises.

2.12. Implement Gym Locator.

2.12.1. Objective: Implement a feature that finds the closest gyms based on the user's current location.

2.13. Create an Arduino Watch:

2.13.1. Objective: Build an Arduino watch that uses an oximeter to measure the user's blood oxygen level and heart rate. This data is then transferred to the application via Bluetooth or wire. This data will only be stored in the local database for privacy and security.

2.14. Design an Accessible UI.

2.14.1. Create a user interface for the AM website, application and watch that is accessible and easy to use for all users, including those with disabilities.

2.15. Efficient and Sustainable Code.

2.15.1. Write code that is efficient, easily maintainable and resource-friendly.

2.16. Implement Privacy and High-Security Standards.

2.16.1. Implement privacy and security measures that adhere to GDPR.

2.17. Implement Data Backup and Recovery Systems.

2.17.1. Establish reliable data backup and recovery mechanisms.

3. Functional Requirements

3.1. User Registration and Authentication:

3.1.1. The application shall link to the AM website for user login and registration.

3.1.2. Users should be able to create accounts with unique usernames and passwords.

3.1.3. Users should log in securely with their credentials.

3.1.4. Users should enter their email and phone details for security and verification reasons.

3.1.5. Users should be able to enter their age during registration to receive age-related workouts, meals, and advice suggestions.

3.2. User profiles:

3.2.1. Admins can modify the server database.

3.2.2. Admins can edit their profiles with personal information (profile picture, name, age, gender, dob, email, password, phone, address, and postcode)

3.2.3. Admins can modify user accounts.

3.2.4. Users can create and edit their profiles with personal information.

3.3. Personalized & Custom Workout Plans:

3.3.1. Users can customize or create their own workout plan.

3.3.2. Users can choose pre-designed workout plans based on their goals.

3.3.3. Users can receive suggested workouts based on age.

3.4. Workout Tracker:

3.4.1. Users can track their workouts by recording exercises, sets, reps, and weights lifted and rate their performance.

3.4.2. Users can keep track of their workout progress based on the user's workout history.

3.5. Nutrition Tracker:

3.5.1. Users can track their daily food intake and calories.

3.5.2. Users can track daily water intake.

3.5.3. Users can generate detailed nutrition stats using charts and graphs to track their progress.

3.6. Body Measurements Tracker:

3.6.1. Users can input body information like height, weight, muscle mass, fat mass etc.

3.6.2. The system can calculate measurements like BMI based on user body data.

3.6.3. Users can generate detailed body stats, charts and graphs to check their progress.

3.6.4. Users can also check heart rate and blood oxygen stats using data from the Arduino watch.

3.7. Gym Locator:

3.7.1. Users can find a list of nearby gyms using the built-in gym finder in the application.

3.8. Health & Nutrition Advice:

3.8.1. The program can provide nutrition advice to the user.

3.8.2. The application can also provide health and fitness advice to the user.

3.9. Feedback and Reviews:

3.9.1. Users can provide feedback and suggestions to improve the service.

3.10. Arduino Watch:

3.10.1. The application shall interface with an Arduino-based smartwatch to measure blood oxygen concentration and heart rate using an oximeter sensor.

3.10.2. The application shall allow the watch to transfer health data to the app via Bluetooth or wired connection.

3.10.3. The watch data will only be stored in the local database. No watch data will be stored in the central database server for privacy and security purposes.

3.10.4. The watch should measure heart rate and blood oxygen levels in real-time.

4. Non-Functional Requirements

4.1. Performance:

4.1.1. The AM website shall load within 2 seconds, ensuring an optimal user experience.

4.1.2. The AM website shall respond to user input within 200 milliseconds, providing a smooth and responsive interaction.

4.1.3. The application shall load within a reasonable time frame, ensuring users can access the features promptly.

4.2. Efficiency & Sustainability:

4.2.1. The AM website should efficiently use system resources.

4.2.2. The application must efficiently utilize system resources.

4.2.3. The watch shall collect and store data in an efficient manner.

4.2.4. The watch should transfer data to the application in an efficient manner.

4.3. Data Storage Optimization:

4.3.1. Implement efficient data storage techniques to minimize database query times.

4.4. Privacy & Security:

4.4.1. The system shall adhere to GDPR guidelines.

4.4.2. The system shall ensure data privacy and security for user personal and health data.

4.4.3. User data should be encrypted and stored safely.

4.4.4. The Bluetooth or wired data transfer from the smartwatch to the app shall be secure and encrypted.

4.5. Reliability:

4.5.1. The AM website will have minimal downtime using robust error handling.

4.5.2. The application shall implement robust error handling to maintain consistent functionality and prevent application crashes.

4.5.3. The database and server shall implement robust error handling and backup mechanisms to prevent data loss and maintain data integrity.

4.5.4. The smartwatch hardware shall be constructed with high-quality materials to ensure longevity and durability.

4.5.5. The smartwatch shall have a robust power management system to maximize battery life and reliability during extended use.

4.6. Usability:

4.6.1. The AM website shall follow Web Content Accessibility Guidelines (WCAG).

4.6.2. The application shall have a user-friendly interface for easy navigation and usage.

4.6.3. The AM website UI shall also be intuitive and easy to navigate.

4.6.4. Accessibility features for users with disabilities on the website and application.

4.6.5. The application shall provide updates of health data from the smartwatch to the app.

4.6.6. The application shall provide error messages and support for troubleshooting in case of data transfer issues from the watch.

4.7. Data Backup & Recovery:

4.7.1. The system shall have a backup and recovery system to prevent data loss.

4.7.2. Regular backups of the program using GitHub and OneDrive.

4.8. Third-Party Service Integration:

4.8.1. Add external services such as mapping APIs, and fitness data providers.