ID4: Update for Software Requirements

1. Introduction

1.1 Purpose

Commercial wearable devices have emerged as promising tools for measuring physical activity both in laboratory and real-world settings. However, researchers have faced challenges extracting these data in an efficient, scalable way. In this paper, we describe the Built Environment and Active Populations (BEAP) Engine, a web application/tool we developed to collect, process, and analyze physical activity data exported from Apple Watch and Fitbit devices. As a value-added feature, the tool also predicts past physical activity using machine learning methods and returns the file in a more user-friendly format.

1.2 Intended Audience

The app is intended for researchers focusing on physical activity measurement and prediction. It caters to academics, scientists, and professionals involved in health-related research, wearable technology, and data analysis.

1.3 Scope

The software aims to address the following main goals and objectives:

1. **Deploying the Web Application:** The primary objective is to deploy the web application in a functional preliminary state, ensuring accessibility and usability for researchers. This involves setting up the necessary infrastructure, configuring servers, and deploying the application to a production environment.
2. **Documentation and Readability:** Extensively document all involved repositories to make the project more readable and open for any future work. This includes comprehensive documentation covering installation instructions, codebase structure, API endpoints, and data flow diagrams. Detailed documentation promotes better understanding, collaboration, and maintainability.

3. **Feedback Mechanism:** Provide team feedback on difficulties and challenges during the development process. Establish a feedback mechanism, such as regular meetings or communication channels, to address issues, share insights, and foster a collaborative environment. Feedback helps identify areas for improvement, resolve conflicts, and ensure alignment with project objectives.
4. **Support for Garmin Data Processing:** Create coverage in Java and React components for potential additions of Garmin data processing, as requested by Professor Fuller. This involves extending the application's functionality to support data from Garmin wearable devices, ensuring compatibility, and scalability for future enhancements.
5. **Performance Optimization:** Implement speed (performance) improvements to enhance user experience, particularly in handling large datasets, as identified by stakeholders. This includes optimizing data processing algorithms, improving database queries, and enhancing frontend responsiveness to reduce loading times and enhance overall system performance.

2. System Features and Requirements

2.1 Functional Requirements

1. **Upload Fitbit, Apple Watch, and Garmin Data:**
    - Users should be able to upload data from Fitbit, Apple Watch, and optionally Garmin devices for processing and analysis. The system should support various file formats such as JSON, XML, or CSV.
2. **Improved User Experience:**
    - Enhance overall user experience, particularly in data visualization and presentation. Implement intuitive and interactive charts, graphs, and dashboards to provide researchers with insightful analytics and trends.
3. **User Authentication:**
    - Implement secure login functionality to ensure user authentication and access control. Utilize industry-standard authentication mechanisms such as OAuth 2.0 or JSON Web Tokens (JWT) for secure authentication.

4. **User Logout:**
    - Provide users with the ability to securely log out from the application. Clear session data and invalidate authentication tokens to ensure user privacy and security.
5. **Data Processing and Prediction:**
    - Allow users to process and predict physical activity data using machine learning algorithms such as Support Vector Machine (SVM), Random Forest, and Decision Tree. Provide options for customizing prediction parameters and model selection.

Nice-to-Have Features:

1. **Upload Garmin Data:**
    - Optionally allow users to upload data from Garmin devices for additional analysis and comparison. Ensure seamless integration with existing data processing pipelines and algorithms.
2. **Process/Predict Garmin Data:**
    - Extend processing and prediction capabilities to include data from Garmin devices for comprehensive research insights. Develop specialized algorithms and models tailored to Garmin data formats and metrics.

2.2 External Interface Requirements

1. **User Authentication Interface:**
    - Implement a user-friendly login interface with options for email/password authentication or social login (e.g., Google, Facebook).
    - Provide password recovery mechanisms such as email verification or security questions for account recovery.
2. **User Logout Interface:**
    - Design a simple and intuitive interface for users to log out from the application. Include options for session termination and account logout.
3. **Prediction Functionality:**

- Enable users to initiate the prediction process based on uploaded device data. Provide clear instructions and visual feedback on prediction progress and results.

4. **Data Selection Interface:**
    - Implement a user-friendly interface, such as radio buttons or checkboxes, for users to select specific data for processing or prediction. Allow users to customize data selection based on timestamps, activity types, or device sources.
5. **Data Processing Interface:**
    - Provide users with a streamlined process for data processing, ensuring ease of use and efficiency. Include options for data preprocessing, feature selection, and model training/validation.
6. **File Upload Interface:**
    - Design a user-friendly drop zone interface for seamless file uploads. Support drag-and-drop functionality and provide feedback on file upload progress and success.
7. **Navigation Bar:**
    - Implement a responsive navigation bar for easy access to various application functionalities. Include menu items for home, profile, settings, data upload, and data analysis.
8. **Progress Bar Functionality:**
    - Display a progress bar to indicate the status of file uploads or processing tasks. Provide real-time updates on progress percentage and estimated completion time.
9. **User Registration Interface:**
    - Provide a user-friendly registration interface for new users to create accounts. Collect essential information such as name, email, password, and optional profile details.
10. **Data Deletion Functionality:**

- Allow users to delete uploaded data or processed results as needed for data management purposes. Implement secure data deletion mechanisms to ensure compliance with data privacy regulations.

11. **Device Selection Interface:**
    - Implement a user-friendly interface for selecting the device type (e.g., Fitbit, Apple Watch, Garmin) for data manipulation. Provide clear instructions and visual cues for device selection.
12. **Data Display Format Selection:**
    - Provide users with options to choose the format for displaying processed or predicted data. Support customizable data visualization templates, including tables, charts, heatmaps, and timelines.
13. **Download Processed Data:**
    - Enable users to download processed or predicted data in their preferred format for further analysis or sharing. Support common file formats such as CSV, Excel, JSON, or PDF.

2.3 Nonfunctional Requirements

1. **Scalability and Performance:**
    - Ensure the application can handle large amounts of data files efficiently and performantly. Implement scalable architecture patterns such as microservices, containerization, and horizontal scaling.
    - Set maximum file upload limits and implement efficient data processing algorithms to handle large datasets. Monitor system performance metrics such as CPU utilization, memory usage, and response times to identify bottlenecks and optimize resource allocation.
    - Conduct load testing and performance profiling to validate scalability and performance under different usage scenarios. Optimize database queries, caching strategies, and network bandwidth utilization for maximum throughput and responsiveness.
2. **Security:**

- Employ robust security measures to protect user data and ensure privacy. Implement encryption for data transmission (TLS/HTTPS) and storage (AES/PGP) to prevent unauthorized access.
- Use industry-standard authentication and authorization mechanisms to enforce access control and user

permissions. Implement role-based access control (RBAC) or attribute-based access control (ABAC) to restrict access to sensitive data and functionalities.

- Conduct regular security audits and vulnerability assessments to identify and mitigate potential security risks. Keep software dependencies and libraries up to date to patch known vulnerabilities and security vulnerabilities.
- Implement secure coding practices such as input validation, output encoding, and parameterized queries to prevent common security vulnerabilities such as SQL injection, cross-site scripting (XSS), and cross-site request forgery (CSRF).

1. **Performance Optimization:**
    - Improve data processing efficiency to minimize processing time and server load. Optimize algorithms for data preprocessing, feature extraction, and model inference to reduce computational complexity and resource utilization.
    - Implement caching mechanisms and data indexing strategies to accelerate data retrieval and query execution. Utilize in-memory databases (e.g., Redis) for caching frequently accessed data and results.
    - Leverage asynchronous processing and parallel computing techniques to distribute workload across multiple CPU cores or nodes. Implement task queues and job schedulers for background processing of long-running tasks and batch jobs.
    - Monitor system performance metrics in real-time and use performance profiling tools to identify performance bottlenecks and optimize critical code paths. Continuously benchmark system performance against predefined service level objectives (SLOs) and iterate on performance improvements based on user feedback and usage patterns.

- All of these requirements were generalized in the following requirements that we are using in our Requirement test Matrix
    1. Pages Load and show all main elements
    2. User can log In and Sign up to the page/ User can log out of the page
    3. User can upload their data
    4. User can download their data transformed into csv
    5. User can manage the uploaded Data
    6. User can use the ML models
    7. User can process a great amount of data
    8. **Security : Allow access to certain pages only when user is logged in**