ID4: Project Architecture

Our system is meticulously designed to optimize both efficiency and security, employing a modular and layered architecture comprised of three key components:

UI Layer:

The user interface drives interactions and is crafted using a combination of technologies for a seamless experience:

- **JavaScript, React, and TypeScript:** These technologies form the backbone of our frontend, offering a dynamic and responsive user interface with the reliability of static typing. JavaScript ensures interactive elements, React facilitates component-based architecture, and TypeScript adds a layer of type safety for enhanced code reliability.
- **Material UI Library (MUI):** Ensuring a consistent and visually engaging design, Material UI provides a rich set of pre-designed components seamlessly integrated into our React application. These components not only enhance user experience but also expedite development by providing ready-made solutions for common UI elements.
- **Jest Tests:** Our UI components are rigorously tested with Jest to guarantee correct rendering and exceptional quality in user experience. Unit tests, integration tests, and snapshot tests ensure that our UI remains consistent and functional across various scenarios and device resolutions.

Core Modules:

Acting as the intermediary between the UI and backend, the Core Modules house the majority of the system logic, leveraging Redux for state management:

- **Redux Integration:** Redux facilitates efficient state management, ensuring smooth communication between the UI and BEAP Engine's API via HTTPS requests. Redux stores application state in a single immutable state tree, simplifying state changes and enhancing predictability. Middleware such as

Redux Thunk enables asynchronous actions, allowing for seamless integration with backend APIs and external services.

- **REST API:** The representational state transfer (REST) API serves as the gateway for interactions with the BEAP Engine, enabling seamless communication with various applications including web and mobile interfaces. RESTful principles ensure scalability, reliability, and interoperability, making it easier to integrate new features and scale the system horizontally.

Backend Modules:

The backend architecture is bifurcated to enhance scalability and maintainability:

1. R Repository:

Utilizing the R programming language, this module processes raw Apple Watch and Fitbit data, offering flexibility and efficiency in data processing:

- **Modular Structure:** The R repository is subdivided into data processor and feature extractor modules, allowing for independent modifications and updates without the need for recompilation of the entire BEAP Engine project. Each module follows SOLID principles, ensuring high cohesion and low coupling for better maintainability and extensibility.
- **Data Transformation:** The data processor transforms raw wearable device data into a readable format, presenting key metrics in an easily interpretable Xlsx file. Advanced techniques such as data normalization and outlier detection are employed to ensure data accuracy and reliability. Meanwhile, the feature extractor employs machine learning algorithms to categorize user activities, enriching the system's analytical capabilities and providing valuable insights for users and stakeholders.

2. Data Repository:

Securely storing user data is paramount, and our data repository ensures robust security measures:

- **PostgreSQL Integration:** Leveraging PostgreSQL, we store user data securely, employing the pgcrypto module for encryption. PostgreSQL offers advanced features such as ACID transactions, referential integrity, and JSONB data type, making it suitable for handling complex data structures and relationships. The pgcrypto module provides cryptographic functions for encryption, decryption, and key management, ensuring data confidentiality and integrity.

- **Data Management:** CRUD operations are handled within the data repository module, ensuring separation of data logic from other system components. This modular approach enables seamless modifications and enhances overall performance by optimizing data queries and operations. Advanced indexing techniques, query optimization, and database normalization are employed to improve data access efficiency and reduce latency.

Data Flow:

Our system's data flow ensures seamless interaction and processing:

1. **Export Data:** Users export data from Fitbit or Apple Watch in XML or JSON format, ensuring compatibility with our system.
2. **Frontend-Backend Interaction:** The frontend communicates exported data to the backend through RESTful APIs, encapsulating data in JSON payloads for transmission.
3. **R Repository Processing:** Raw data is processed within the R Repository, converting it into a readable format suitable for analysis and visualization. Robust error handling and logging mechanisms are implemented to ensure data integrity and reliability.
4. **Data Storage:** Processed data is securely stored in the PostgreSQL database within the Data Repository, utilizing efficient storage mechanisms and encryption algorithms to safeguard user privacy and comply with regulatory requirements.

Modularity and Flexibility:

Our architecture boasts modular design principles, enhancing flexibility and ease of maintenance:

- **Loose Connections:** Components such as the Core Module, Data Repository, and R Repository are loosely connected, allowing for independent modifications and updates without disrupting other parts of the system. This modularity promotes agility and facilitates future enhancements. Interfaces and abstraction layers are employed to decouple dependencies and promote component reusability.

Navigation and Pages:

The user interface offers intuitive navigation and robust page implementations:

- **Components**:

  Navigation Bar: A comprehensive navigation bar provides access to various pages including File Upload, Processed Files, Home, Logout, and Predicted Files. The navigation bar is implemented as a reusable component, ensuring consistency across different sections of the application.

  Logout Page: This displays a logging out message when the Logout button is clicked and redirects the user back to the landing page.

  Login/SignUp Page: This handles the user login and creation of a user account separated into Researcher and Personal User.

- **Individual Pages:** Each page, meticulously designed with styles and Jest tests, ensures consistency and readiness for additional functionality. Components are organized using a modular folder structure, promoting code reusability and maintainability. These pages include; File Upload, Processed Files, Predicted Files.

  File Upload: This Page allows the user to upload files and get them ready to be processed. It is also on this page that the files can be processed by clicking the Process button. This page will also display all the files already uploaded by the user.

  Processed Files: This Page will display all the files previously processed by the user and gives the user the ability with a “Predict” button, to predict the files that have been previously processed.

Predicted Files: This Page will allow users to download the predicted files.

Security Measures:

Our system prioritizes data privacy and security with robust measures:

- **Transport Layer Security (TLS):** TLS protocol secures connections between components, safeguarding data privacy during transmission across various system layers. Strong encryption algorithms and certificate-based authentication mechanisms are employed to prevent eavesdropping and man-in-the-middle attacks.
- **Symmetric Key Encryption:** User data, encompassing raw, processed, and predicted data, is encrypted using symmetric key algorithms. The encryption key is securely stored on the server, utilizing industry-standard key management practices and hardware security modules (HSMs) to prevent unauthorized access. Data encryption and decryption operations are performed within secure execution environments, ensuring confidentiality and integrity of sensitive information. Access control mechanisms such as role-based access control (RBAC) are enforced to restrict unauthorized access to data and system resources. Regular security audits and penetration testing are conducted to identify and remediate potential vulnerabilities, ensuring compliance with regulatory standards and industry best practices.