Student Alumni Mentorship Portal Architecture & Design

Technology Stack

- Front End:
  - Technology: React.js for the dynamic user interface
  - Component Library: Material-UI
  - State Management: Redux
  - Backend Connection: Axios for API calls
  - Authentication: Auth0
  - Input Validation: Joi
- Back End:
  - Framework: Node.js with Express.js for server-side development, leveraging Mongoose for database interaction.
  - Authentication: Auth0
  - API Documentation: OpenAPI
  - Input Validation: Joi
- Database:
  - MongoDB for its flexibility and scalability

Rationale

- Scalability and Performance:

  - The MERN stack is known for its scalability, capable of handling a growing number of users and data, in line with the system's scalability constraint.
  - MongoDB provides efficient data storage and retrieval, ensuring minimal response times for loading user-profiles and scheduling meetings.
  - Node.js facilitates non-blocking I/O, enhancing real-time features such as chat functionality for smooth communication between students and alumni.

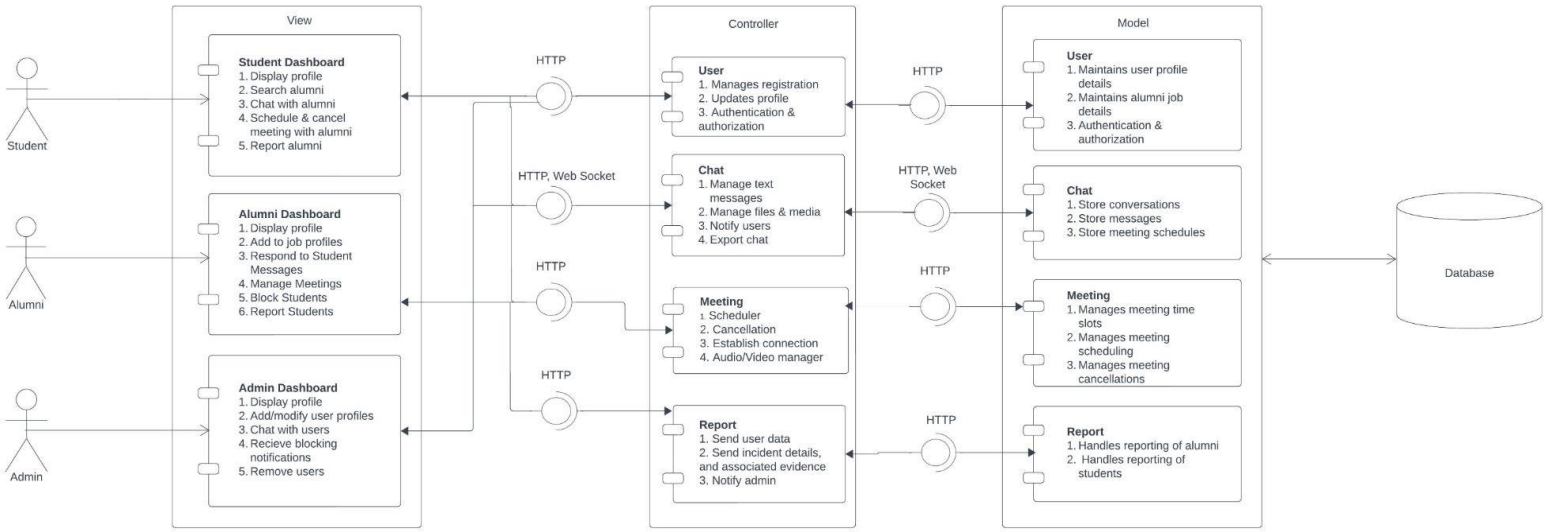
- User Interface and Responsive Design:

    - ❖ ReactJS, along with Material-UI, provides a user-friendly and intuitive design. Material-UI components are designed to be responsive, ensuring accessibility on different devices and screen sizes.

- Security and Data Backup:

    - ❖ Auth0 provides user authentication and authorization features, including encryption of sensitive data, thus enhancing overall system security.
    - ❖ Joi ensures that user inputs are validated and secure, which is crucial for preventing security vulnerabilities. It enforces data integrity, guaranteeing that data stored in the database remains consistent and complies with specified constraints.
    - ❖ MongoDB provides features for data replication and backup, making it a suitable choice for data durability and backup.

High-Level Component and Connector Diagram



https://drive.google.com/drive/folders/1RbGeeskU8SKV9AC_ut9m4TzsokTzXoX4?usp=sharing&hl=en

- **Frontend Components**:
    - User Authentication Module

        - Sub-components: Login, Registration
    - User Profile Module
        - Sub-components: Profile Editing, View Alumni Profiles
    - Mentorship Module
        - Sub-components: Chat, Meeting Scheduling
    - Admin Module
        - Sub-components: User Management, Admin-User Chat
- **Backend Components**:
    - Authentication & Authorization
    - User Management
    - Profile Management
    - Messaging Services
    - Meeting Scheduling
    - Admin Interface
    - Database Interface
- **Connectors**:
    - HTTP/HTTPS for client-server communication
    - WebSocket for real-time chat functionality
- **Sub-components**:
    - **Authentication**:
        - Auth0 Authentication
        - OAuth Integration
    - **Database Interaction**:
        - CRUD Operations for Users, Alumni, and Admin
        - Message Storage and Retrieval
        - Meeting Schedule Management

Existing Component Libraries

- **Front End Components**:
    - **Material-UI**: Material-UI is a popular front-end library for React applications. It provides a set of pre-designed UI components and styles based on Google's Material Design guidelines. Developers can use Material-UI to create aesthetically pleasing and consistent user interfaces quickly.
    - **Redux**: Redux is a JavaScript library for managing the state of an application. It helps ensure data consistency across different parts of the application by centralising the application's state in a store. Redux allows developers to dispatch actions to modify the state and provides a predictable way to manage and update the application's data.
    - **Axios**: Axios is a JavaScript library used for making asynchronous HTTP requests from the front end to a server or backend API. It simplifies the

process of sending and receiving data over HTTP and provides features like handling promises and intercepting requests and responses.

- ❖ **Joi**: Joi is a library commonly used for input validation in JavaScript applications. It allows developers to define schemas for validating and sanitising data, ensuring that incoming data meets specific criteria and is safe to use in the application.

- **Back End Components**:
  - ❖ **Express.js**: Express.js is a popular and minimalist web framework for Node.js. It simplifies the process of building server-side applications and APIs by providing a set of features and middleware for handling routing, request/response handling, and more. Express is known for its speed and flexibility, making it a popular choice for building web applications.
  - ❖ **Auth0**: Auth0 is an authentication and identity management platform that helps developers add secure user authentication and authorization to their applications. It provides features like single sign-on (SSO), social login, multi-factor authentication, and user management, making it easier to implement robust and secure user authentication in a web application.
  - ❖ **Mongoose**: Mongoose is an Object-Document Mapping (ODM) library for MongoDB, a NoSQL database. It simplifies working with MongoDB by providing a schema-based data modelling approach and tools for querying and manipulating data in a more structured way. Mongoose helps developers interact with MongoDB databases in a Node.js environment.
  - ❖ **OpenAPI**: OpenAPI is a specification for documenting and defining the endpoints and behaviour of a RESTful API. It provides a standardised way to describe API endpoints, request/response formats, authentication methods, and more. OpenAPI documentation makes it easier for developers to understand and consume an API and is often used to generate client SDKs and server stubs automatically.

High-Level Design

Front End Design

- **Overall Technology:**
  - React.js for building a dynamic and responsive user interface.

- **Component Library:**
    - Material-UI for UI components and styling.
- **State Management:**
    - Redux to manage the application state, including user authentication, profile data, and chat messages.
- **List of Screens:**
    - **Login/Registration Screen**: Handles user registration, authentication and authorization.
    - **Dashboards:** For
        - Students - Search alumni, chat, calendar, notifications & user profile
        - Alumni - chat, calendar, notifications & user profile
        - Admin - chat, notifications & user profile
    - **Notifications -** For
        - Students - unread messages, meeting reminder
        - Alumni - unread messages, meeting reminder
        - Admin - unread messages, blocking and reporting
    - **User Profile**: Users can see their profiles
    - **View Profile -**
        - Student - alumni profiles
        - Admin - student & alumni profiles, remove profiles
    - **Chat Interface**:
        - Student - chat, schedule meeting, report alumni, download chat
        - Alumni - chat, block student, report student, download chat
        - Admin - chat, remove user
    - **User Calendar**:
        - Alumni - Provide their availability slots, cancel meetings, View scheduled meetings, join meetings
        - Student - View scheduled meetings, join meetings
    - **View Calendar**:
        - Student - Schedule a meeting
- **Technology for Connecting with the Backend:**
    - Axios will be used for making HTTP requests to the backend APIs.

Back End Design

- **Backend Framework**: Node.js and Express.js for building RESTful APIs.

- **Components/Libraries**:
    - **Express.js**: Handles routing and middleware management
    - **Mongoose**: Interacts with MongoDB for data storage
    - **Socket.io**: Facilitates real-time chat functionality
    - **Auth0**: Provides authentication mechanism
    - **Axios**: for communicating with the front-end
    - **Joi**: For input validation

- **Service Descriptions:**
    - Authentication Service:
        - API: /api/auth
        - Description: User registration and login.
        - Inputs: User credentials (email, password)
        - Outputs: Auth0 token upon successful login
    - User Management Service:
        - API: /api/users
        - Description: CRUD operations for user profiles.
        - Inputs: User data (profile information)
        - Outputs: User profile data
    - Messaging Service:
        - API: /api/messages
        - Description: Sending and receiving chat messages.
        - Inputs: Message content, sender, receiver
        - Outputs: Sent/received messages
    - Meeting Scheduling Service:
        - API: /api/meetings
        - Description: Schedule, cancel, and retrieve meetings.
        - Inputs: Meeting details (date, time, participants)
        - Outputs: Meeting information
    - Admin Interface Service:
        - API: /api/admin
        - Description: User management and reporting for administrators.
        - Inputs: Admin actions (user management, blocking, reporting)
        - Outputs: Success or error messages

- **Model Layer**

  Abstraction layer for interacting with the database and providing structured data to the APIs.

    - Mentorship management: This service will be responsible for managing the mentorship relationships between students and alumni mentors.
        - User management: This service will be responsible for user authentication, CRUD operations.
    - Meeting management: This service will be responsible for managing the scheduling and cancellation of virtual meetings between students and alumni mentors.

Database Design

- **DB System**: MongoDB, a NoSQL database, that is well-suited for storing and managing data for complex web applications, chosen for its flexibility and scalability.
- **Schemas**:
    - **User**:
        - Name
        - Email
        - Password
        - Profile Image
        - Additional profile details
    - **Mentorship schema**: This schema will store the information about mentorship relationships between students and alumni mentors, such as the start date, end date, and status of the relationship.
    - **Meeting schema**: This schema will store the information about virtual meetings between students and alumni mentors, such as the date, time, and duration of the meeting.
    - **Message**:
        - Sender ID
        - Receiver ID
        - Message Content
        - Timestamps
    - **Admin Schema**:
        - Fields: username, email, password hash, role, Auth0 token