**Technical Document for Eventure**

**1. Introduction**

**1.1 Project Overview**

Main aim of the Eventure is to streamline the creation, management, and participation in events. This event management platform allows users to create events, assign tasks, manage participants, and track event progress with different users

**1.2 Purpose**

This technical overview of the Eventure platform, Explains its architecture, components, and functionalities. It is intended for developers, system architects, and technical stakeholders who is interested in understanding the platform.

**2. System Architecture**

Eventure is a full-stack application with a clear separation between the frontend and backend. The system architecture is divided into three main components:

1. **Frontend**: Developed using React and HTML/CSS
2. **Backend**:
   * **Node.js :** which handles user authentication and tasks, participants management.
   * **Java with Spring Boot**: Manages event creation, task assignments, and participant management.
   * **Mongodb**: Database management

**3. System Components**

**3.1 Frontend**

Eventure is built using Reactjs and javascript, this serves as the user interface for all user interactions, including event creation, task assignment, and participant management.

* **Dashboard**: Displays an overview of events, tasks, and invite participants
* **Event Management**: Allows users to create, view, and edit events.
* **Task Management**: Enables users to view, update
* **Participant Management**: Participant details

**3.2 Backend**

The Node.js backend handles the authentication. It is responsible for creating user accounts and authenticating users

* **Authentication**: Manages user sign-up, login, and session handling.
* **Role Management**: Determines the user's role (Admin, Organizer, Participant) within the context of an event.

User can do the following tasks:

**sign-up**: Create a new user account.

* Change current user password.

Get current user’s assigned tasks.

* Update task status.
* Update participant status.
* The Java Spring Boot backend manages
* **Event Management:** Creation, updating, and deletion of events.
* **Task Management**: Task updates.
* **Participant Management**: Manages participant

**4. User Roles and Permissions**

**Admin**

* Create/Edit Events
* Task Assignment
* Participant Management

**Organizer**

* Task Management:

**Participant**

* Event Participation

**6. Dashboard**

The dashboard is the central hub for users, which contains:

**Event Section:** Displays all events that the user is involved in events

**Task Section :** Displays tasks assigned to the user.

**Participants:**  Shows all event invitations that the participants has received and no of participants with details

**7. Deployment**

**1. Create a project directory**: mkdir eventure

cd eventure

**2. Initialize react app**:To initiate react app

npx create-react-app client

cd client

npm install this will run application using local host

**3. Access the backend**: Resolve all dependencies and run spring boot using ./mvnw spring-boot:run

4. **Setup front end :** using npm install and npm start.

5. **Set up node :** npm install express and node server.js to connect ti the server

**6. Database connection** : invoke database using mongod (mongodb://localhost:27017/eventure).