**Event Management Dashboard Documentation**

**Overview**

The Event Management Dashboard is a web-based application designed to streamline event planning and management. It allows users to manage events, attendees, and tasks effectively.

**1. Instructions to Set Up and Run the Project**

Prerequisites

Ensure the following are installed on your system:

* Node.js (version 14 or higher)
* MongoDB (local or cloud instance)
* npm (comes with Node.js)
* React (bundled with the frontend setup)

**Steps to Set Up**

**1. Download the Project Files**

Download the zip file from the provided link and extract it.

**2. Backend Setup**

Navigate to the backend folder:

cd event\_management\_dashboard/backend

Install dependencies:

npm install

Start the backend server:

node server.js

The backend will run at http://localhost:5000.

**3. Frontend Setup**

Navigate to the frontend folder:

cd ../frontend

Install dependencies:

npm install

Start the frontend server:

npm start

The frontend will run at http://localhost:3000.

**4. Database Setup**

Ensure MongoDB is running locally or connect to your MongoDB Atlas instance.

The database will be created automatically upon starting the backend.

**2. API Details**

Base URL

The API base URL for the backend is:

http://localhost:5000

Event Management APIs

**1. Create an Event**

Endpoint: POST /events

Description: Adds a new event to the system.

Request Body: {

"name": "Event Name",

"description": "Event Description",

"location": "Event Location",

"date": "2024-12-25",

"attendees": [],

"tasks": []

}

Response:

Returns the created event.

{

"\_id": "event\_id",

"name": "Event Name",

"description": "Event Description",

"location": "Event Location",

"date": "2024-12-25",

"attendees": [],

"tasks": []

}

**2. Get All Events**

Endpoint: GET /events

Description: Fetches a list of all events.

Response:

[

{

"\_id": "event\_id",

"name": "Event Name",

"description": "Event Description",

"location": "Event Location",

"date": "2024-12-25",

"attendees": [],

"tasks": []

}

]

**3. Update an Event**

Endpoint: PUT /events/:id

Description: Updates details of an event.

Request Body: (Example)

{

"name": "Updated Event Name"

}

Response:

Returns the updated event.

4. Delete an Event

Endpoint: DELETE /events/:id

Description: Deletes an event by its ID.

Response:

HTTP 204 No Content.

**Attendee Management APIs**

**1. Add an Attendee**

Endpoint: POST /events/:id/attendees

Description: Adds an attendee to a specific event.

Request Body:

{

"name": "Attendee Name"

}

Response:

Returns the updated event.

**2. Remove an Attendee**

Endpoint: DELETE /events/:eventId/attendees/:attendeeId

Description: Removes an attendee from an event.

Response:

Returns the updated event.

**Task Management APIs**

**1. Create a Task**

Endpoint: POST /events/:id/tasks

Description: Adds a task to a specific event.

Request Body:

{

"name": "Task Name",

"deadline": "2024-12-30",

"status": "Pending",

"assignedTo": "Attendee Name"

}

Response:

Returns the updated event.

**2. Update Task Status**

Endpoint: PUT /events/:eventId/tasks/:taskId

Description: Updates the status of a specific task.

Request Body:

{

"status": "Completed"

}

Response:

Returns the updated task

**3. Project Structure**

The project is organized into two main folders:

1. Backend:

Contains the Node.js server (server.js) and handles APIs and database integration.

2. Frontend: Built with React, includes pages for managing events, tasks, and attendees.