Full-Stack Assignment: Event Management Platform

Objective

Build a full-stack event management platform where users can create, manage, and view events. The platform should include:

- User authentication.
- Event creation and management tools.
- Real-time updates for attendees.
- Deployed on free-tier hosting services.

Features

Frontend:

1. User Authentication: Allow users to register and log in. Option for "Guest Login" to access limited features.
2. Event Dashboard: Display a list of upcoming and past events with filters for categories and dates.
3. Event Creation: Form to create an event with fields like event name, description, date/time, and more.
4. Real-Time Attendee List: Show the number of attendees for each event in real-time.
5. Responsive Design: Ensure the platform works seamlessly on all devices.

Backend:

1. Authentication API: Use JWT for secure authentication.
2. Event Management API: CRUD operations for events with ownership restrictions.
3. Real-Time Updates: Use WebSockets for real-time updates.
4. Database: Store event and user data efficiently.

Deployment

1. Frontend Hosting: Use Vercel or Netlify for free hosting.
2. Backend Hosting: Use Render or Railway.app for free hosting.
3. Database: Use MongoDB Atlas (Free) or Planetscale (Free).
4. Image Hosting: Use Cloudinary Free Tier.

Assignment Deliverables

1. GitHub Repository: Code for frontend and backend with a clear README.md.
2. Live Deployment: Provide live URLs for both frontend and backend.
3. Test User Credentials: Include test credentials for evaluation.

Free Tools Required

1. Frontend: React.js (Free), Vercel or Netlify (Free hosting).
2. Backend: Node.js with Express.js (Free), Render or Railway.app (Free hosting).
3. Database: MongoDB Atlas (Free) or Planetscale (Free).
4. Image Hosting: Cloudinary Free Tier.
5. Real-Time Communication: Socket.IO (Free).

Evaluation Criteria

1. Functionality: Are all features implemented as described?
2. Deployment: Is the app live and accessible?
3. Code Quality: Is the code clean, modular, and well-documented?
4. UI/UX: Is the design intuitive and responsive?
5. Performance: Does the app handle multiple users efficiently?