**Event Master Events App — Project Documentation**

**Project Overview**

**Event Master** is a full-stack web application designed to help users **create, discover, and register for events** easily. The platform connects people through shared experiences — living up to its slogan:

*“* Where we bring you the best event services*.”*

The app provides a clean and engaging interface for managing events and registering users

**Project Structure**

events-app/

├── backend/

│ ├── prisma/

│ │ └── schema.prisma

│ ├── src/

│ │ ├── controllers/

│ │ ├── routes/

│ │ ├── services/

│ │ ├── middleware/

│ │ ├── utils/

│ │ ├── app.ts

│ │ └── server.ts

│ ├── package.json

│ └── Dockerfile

│

├── frontend/

│ ├── src/

│ │ ├── components/

│ │ ├── pages/

│ │ ├── hooks/

│ │ ├── api/

│ │ └── main.tsx

│ ├── package.json

│ └── Dockerfile

│

├── docker-compose.yml

└── .github/workflows/ci.yml

**Technologies Used**

**Frontend**

|  |  |
| --- | --- |
| **Technology** | **Purpose** |
| **React.js** | Building the user interface |
| **TypeScript** | Type safety and cleaner code |
| **Vite** | Fast frontend build tool |
| **Axios** | API calls to the backend |
| **Tailwind CSS** | Modern and responsive UI styling |
| **Font Awesome** | Social media and navigation icons |

**Backend**

|  |  |
| --- | --- |
| **Technology** | **Purpose** |
| **Node.js** | Backend runtime environment |
| **Express.js** | Server framework for APIs |
| **Prisma ORM** | Database modeling and migrations |
| **PostgreSQL** | Relational database management |
| **bcrypt** | Password hashing for user authentication |
| **jsonwebtoken (JWT)** | Secure authentication using tokens |
| **dotenv** | Manage environment variables |
| **Nodemon** | Automatically restart the server during development |

**DevOps & Tools**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| | **Tool** | **Purpose** | | --- | --- | | **Docker & Docker Compose** | Containerization for frontend, backend, and database | | **Git & GitHub** | Version control and project collaboration | | **GitHub Actions (CI/CD)** | Automated build and testing workflow | | **VS Code** | Integrated development environment | |

**Setup Instructions**

**1️⃣ Clone the Repository**

git clone https://github.com/your-username/wevibe-events-app.git

cd wevibe-events-app

**2️⃣ Install Dependencies**

* Backend:
* cd backend
* npm install
* Frontend:
* cd frontend
* npm install

**3️⃣ Environment Setup**

Create a .env file in the backend:

DATABASE\_URL=postgresql://user:password@localhost:5432/wevibe

JWT\_SECRET=yourSuperSecretKey

**4️⃣ Run Prisma Migrations**

cd backend

npx prisma migrate dev --name init

**5️⃣ Start the Application**

Run both containers using Docker Compose:

docker-compose up --build

Or, to run manually:

# Terminal 1

cd backend

npm run dev

# Terminal 2

cd frontend

npm run dev

Then open in your browser:  
👉 http://localhost:5173

**Frontend Pages**

|  |  |  |
| --- | --- | --- |
| **Page** | **Description** |  |
| **Landing Page** | Welcome screen with sign-in/register buttons and social links |  |
| **Login Page** | User authentication via JWT |  |
| **Register Page** | New user account creation |  |
| **Create Event Page** | Form for event title, date, location, and image upload |  |
| **Events Dashboard** | Displays all upcoming events |  |
| **Event Details Page** | Shows event info and RSVP option |  |

**Authentication Flow**

1. Users register with a username and password.
2. Passwords are **hashed using bcrypt** before saving to the database.
3. On login, credentials are verified and a **JWT token** is issued.
4. The token is stored in the frontend (local storage) and used for secure API requests.
5. Protected routes require the token to be verified using Express middleware.

**Challenges Faced**

|  |  |  |
| --- | --- | --- |
| **Challenge** | **Description** | **Solution** |
| **Backend–Frontend Integration** | Managing CORS errors and API endpoints during development. | Configured CORS middleware and used proxy in Vite. |
| **Database Connection** | Prisma migrations occasionally failed due to schema mismatch. | Ensured consistent schema updates and used prisma migrate reset for sync. |
| **JWT Expiry Handling** | Tokens expiring unexpectedly, logging users out. | Implemented refresh tokens and token expiry checks in middleware. |
| **Docker Setup** | Networking issues between backend and PostgreSQL container. | Used docker-compose network configuration to link services. |
| **Styling Responsiveness** | Maintaining layout consistency across devices. | Used Tailwind CSS utility classes for adaptive design. |
| **User Experience** | Making UI intuitive while keeping code modular. | Implemented reusable React components (EventCard, EventForm, etc.). |
|  |  |  |

**Future Improvements**

* Add event categories and filters.
* Integrate real-time notifications (Socket.io).
* Add calendar view and reminders.
* Implement Google or Facebook OAuth.
* Deploy to AWS or Vercel for production hosting.

**Team & Contributions**

* **Frontend Developer:** Built UI with React and Tailwind CSS.
* **Backend Developer:** Designed APIs and authentication with Express + Prisma.
* **DevOps Engineer:** Managed Docker setup and GitHub CI/CD workflows.

**Conclusion**

The **Event Master App** demonstrates a modern architecture using TypeScript, React, Node.js, Prisma, and Docker. Despite challenges during integration, the final solution achieved an event management platform where users can view events.