



In association with



**PROJECT TITLE- Capstone Project**  
**Event Management**

**Submitted by:**

**RUDRA PRANAV**

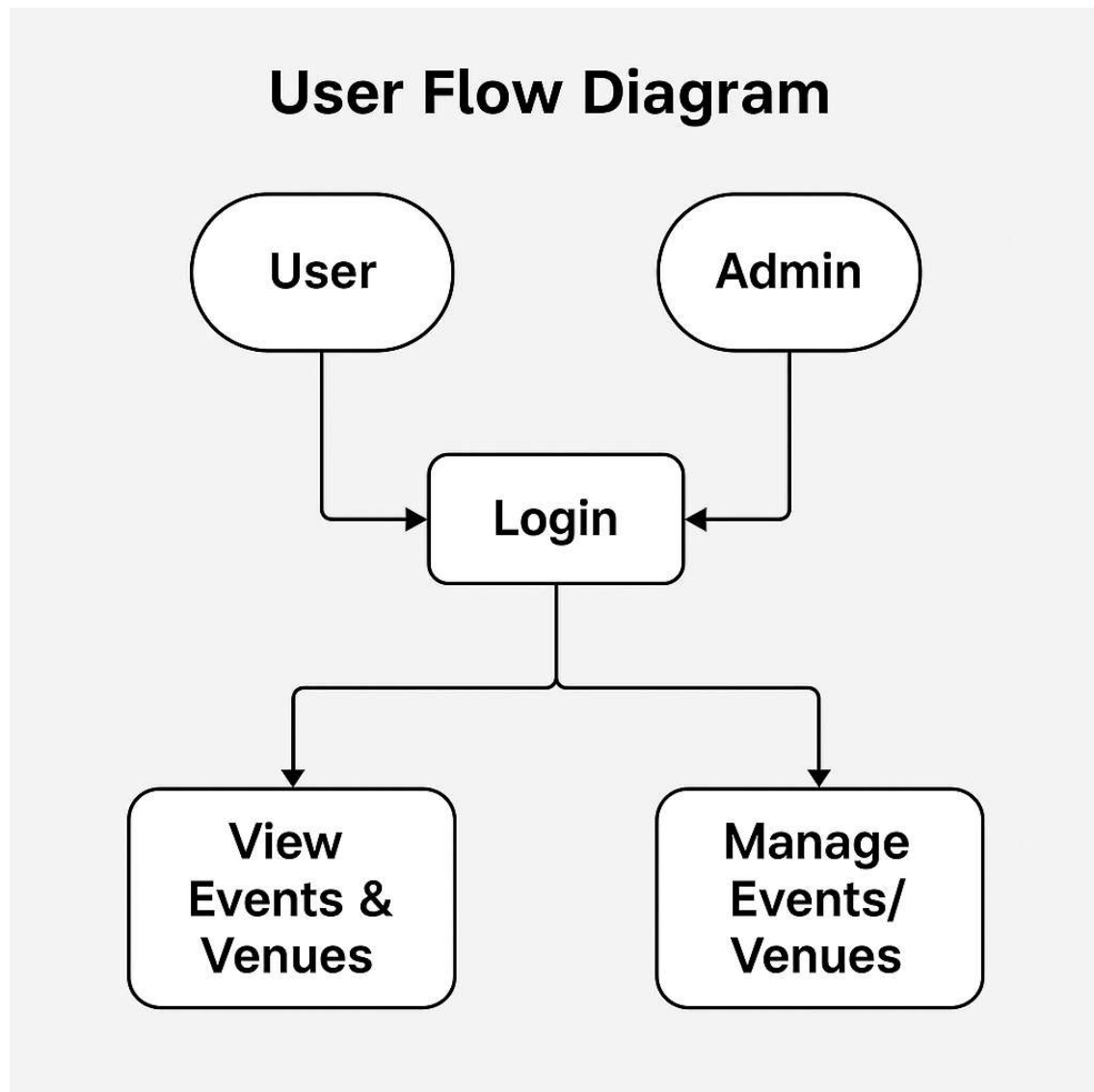
## Introduction:

The Event Management System is a full-stack web application designed to streamline the management of events and venues. It supports role-based access where Admins can perform CRUD operations, and Users can view available events and venues. This project is built using Spring Boot for the backend and React.js for the frontend, offering a modern, secure, and responsive web experience.

## Objective:

- To develop a user-friendly and secure platform to manage events and venues.
- To implement role-based access control using JWT authentication.
- To build a fully functional, modular, and dockerized full-stack application.
- To ensure clean architecture separation between backend and frontend.

Flowchart:



## Technologies Used

Layer	Technology
Backend	Spring Boot, Java, Spring Security, JWT
Frontend	React.js, Axios, React Router
Database	H2 / PostgreSQL (configurable)
Build tool	Maven
Deployment	Docker, Dockerfile
Version Control	Git, GitHub

## Source Code Repository:

<https://github.com/Rudra2215/EventManager>

## Features of the Application:

### 1. User Authentication & Authorization

- Secure login using JWT
- Two roles:
  - **USER:** Can view events and venues
  - **ADMIN:** Can add, update, delete events & venues

### 2. Event Management

- List all events
- View event details
- Create/update/delete events (Admin)

### 3. Venue Management

- Add and manage venues
- Associate events with venues

#### 4. Responsive Frontend

- Built with React.js for a dynamic user experience
- Seamless routing using React Router

#### 5. Security & Deployment

- JWT-secured APIs
- Fully dockerized for portability and easy deployment

#### Modules Overview:

Module	Description
User Module	Login, register, role-based access
Event Module	Manage events, link to venues
Venue Module	Add/edit venues and associate events
Security Module	JWT generation, role authorization (Spring Security)

#### Frontend URLs & Pages (React):

Page	URL	Description
Login	http://localhost:3000/login	Login screen for both USER/ADMIN
Register	http://localhost:3000/signup	New user registration
Dashboard	http://localhost:3000	Landing page showing events
Events	http://localhost:3000/events	View all events
Admin Panel	http://localhost:3000/events/new	Admin area for managing events/venues

**Backend Endpoints (Spring Boot):****Events:**

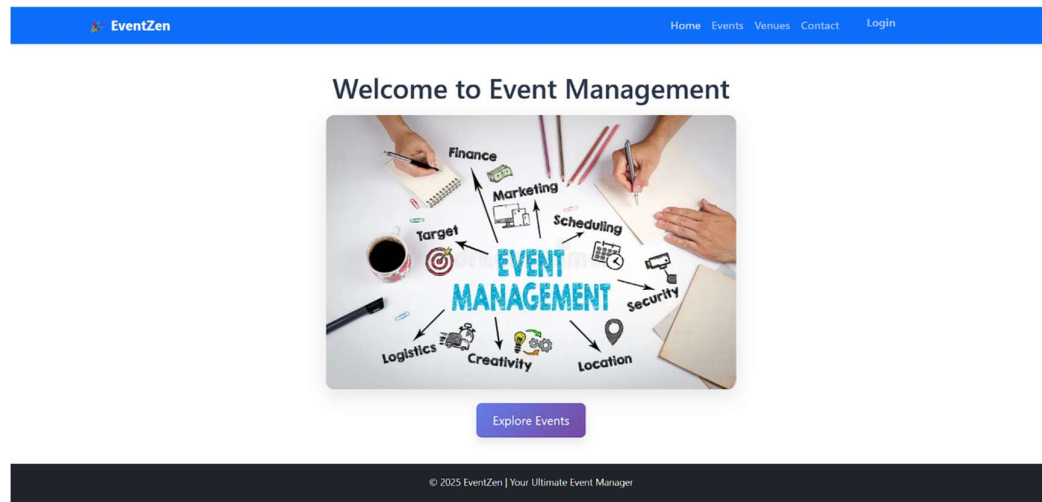
Method	Role	Description
GET	ALL	Get all events
POST	ADMIN	Create all events
PUT	ADMIN	Update event
DELETE	ADMIN	Delete event

**Venue:**

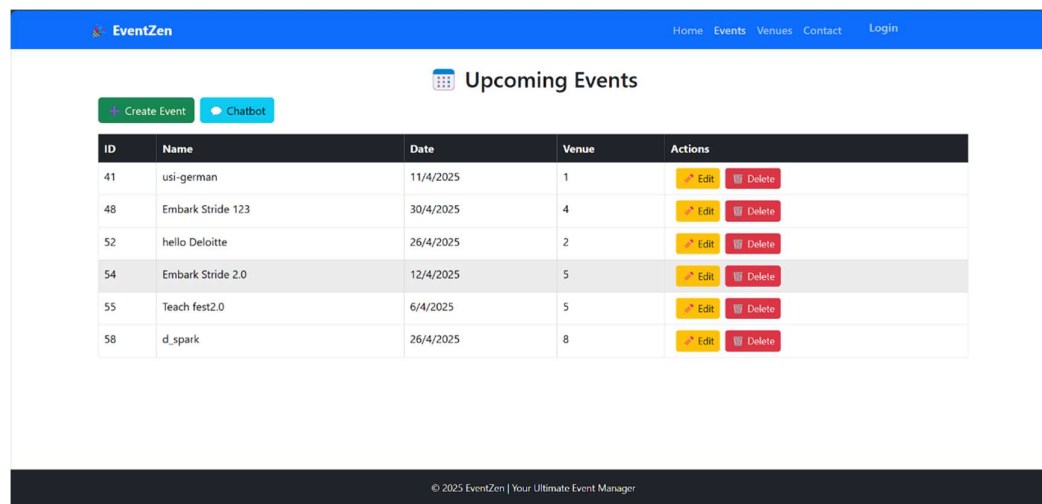
Method	Role	Description
GET	ALL	Get all venues
POST	ADMIN	Create all venues
PUT	ADMIN	Update venues
DELETE	ADMIN	Delete venues

## UI Screenshots (React):

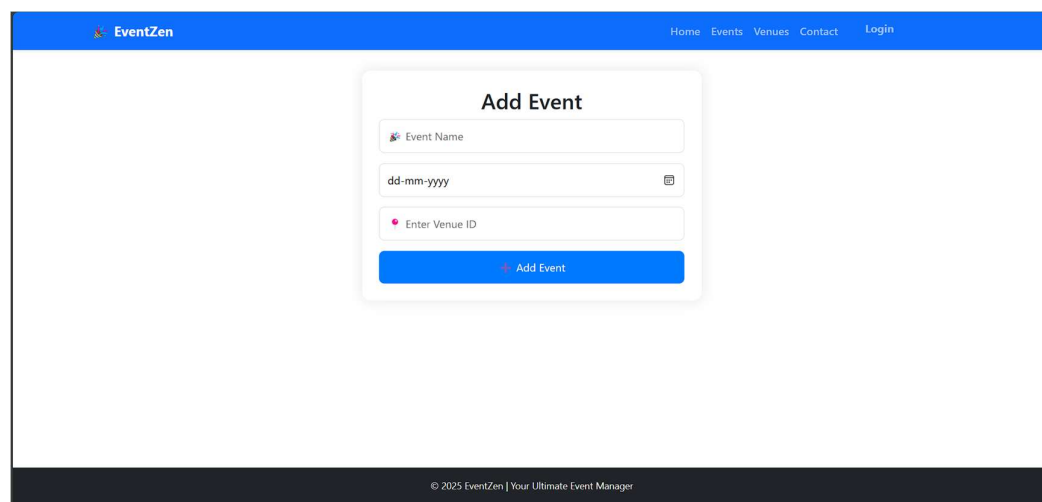
### Home page



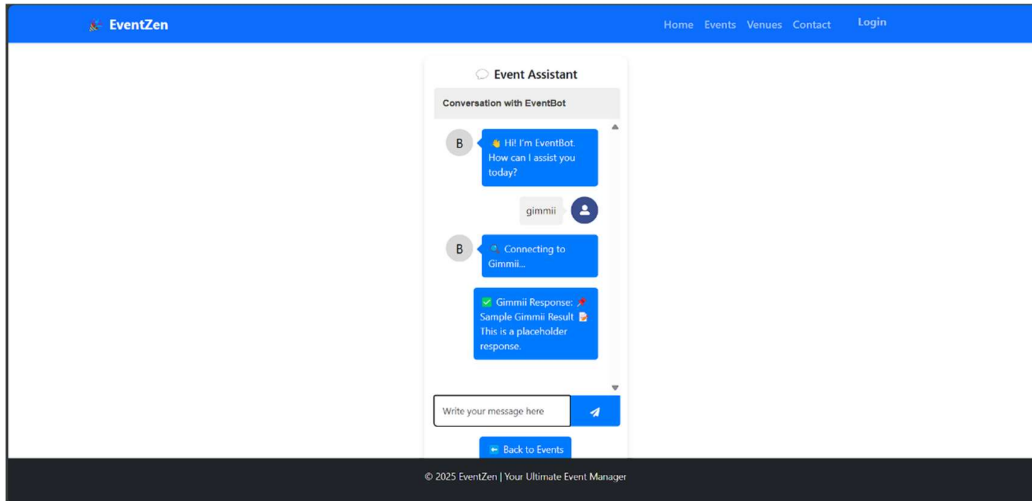
### Events:



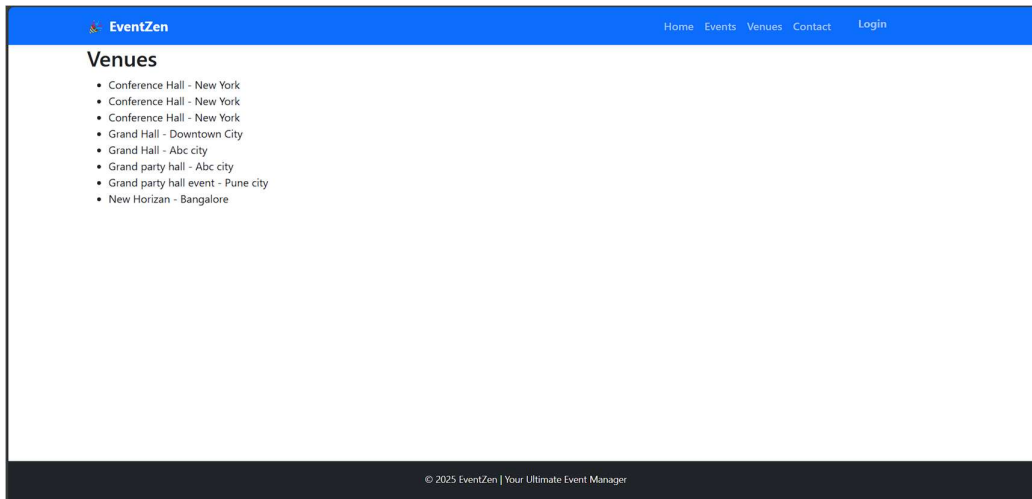
### Create Event:



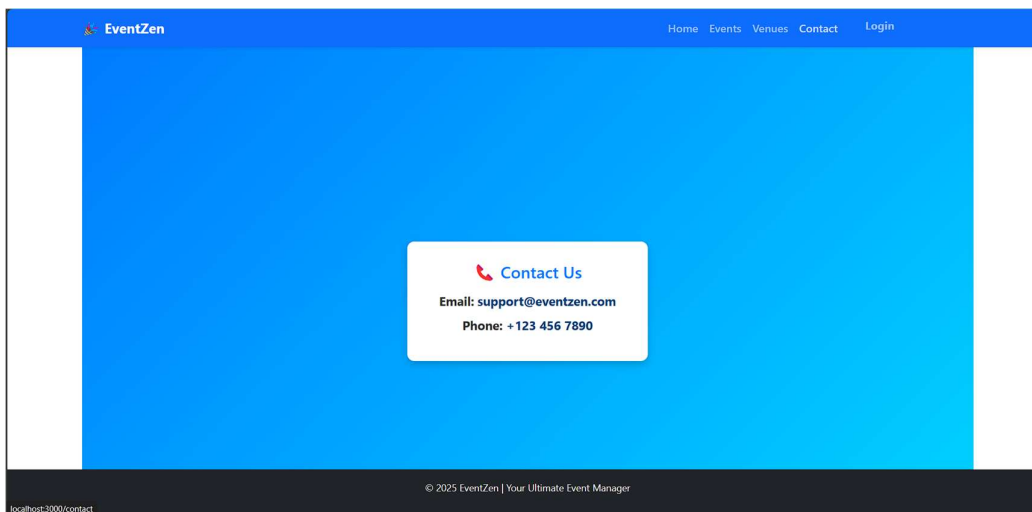
## Chatbot:



## Venues:



## Contact:



# Capstone

Login:

EventZen

Home Events Venues Contact Login

### Login to EventZen

Email:

Password:

Role:

Login

Don't have an account? Sign up

© 2025 EventZen | Your Ultimate Event Manager

Signup:

EventZen

Home Events Venues Contact Login

### Create an Account

Name:

Email:

Password:

Signup

Already have an account? Login

© 2025 EventZen | Your Ultimate Event Manager

React:

```
1 import React from "react";
2 import { Link } from "react-router-dom";
3 import "bootstrap/dist/css/bootstrap.min.css";
4 import "./header.css";
5
6 function Header() {
7   return (
8     <div className="navbar navbar-expand-lg navbar-dark bg-primary shadow-sm py-2">
9       <div className="container">
10        <Link className="navbar-brand fw-bold" to="/" />
11        <div>
12          EventZen
13          <button
14            className="navbar-toggler"
15            type="button"
16            data-bs-toggle="collapse"
17            data-bs-target="#navbarNav"
18          />
19        </div>
20        <span className="navbar-toggler-icon"></span>
21      </div>
22    </div>
23  );
24}
```

Compiled successfully

You can now view event-management-frontend in the browser.

Local: http://localhost:3001  
On Your Network: http://192.168.0.102:3001

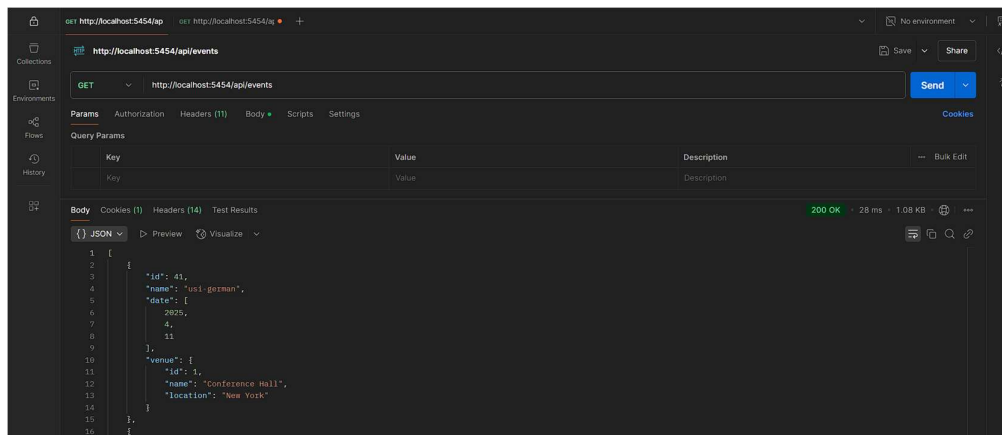
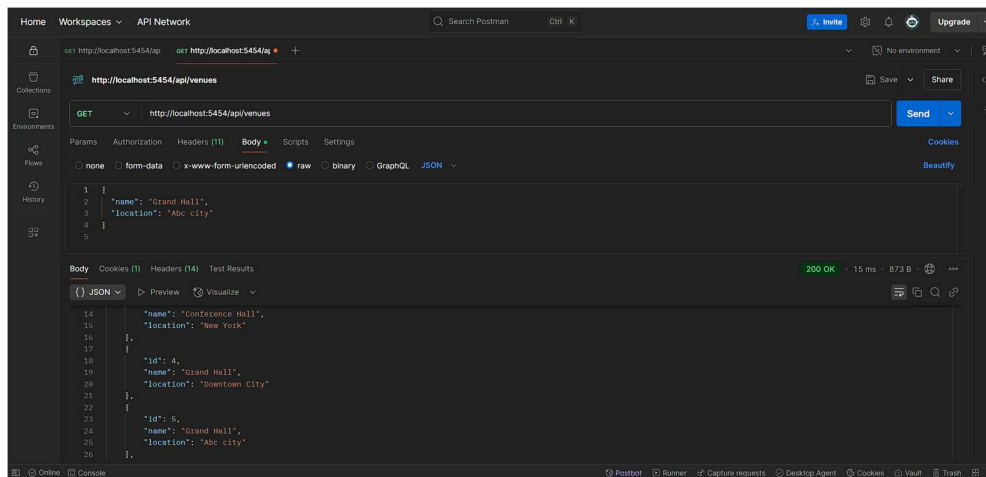
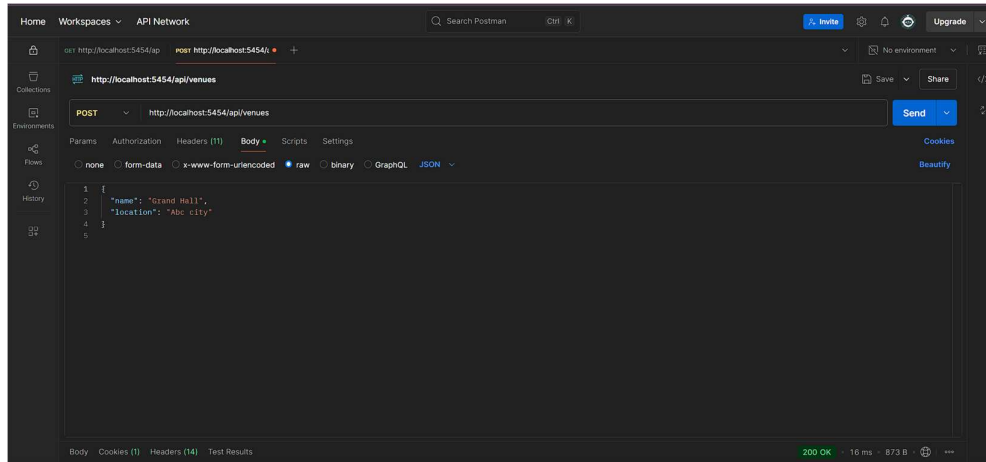
Note that the development build is not optimized.  
To create a production build, use `npm run build`.

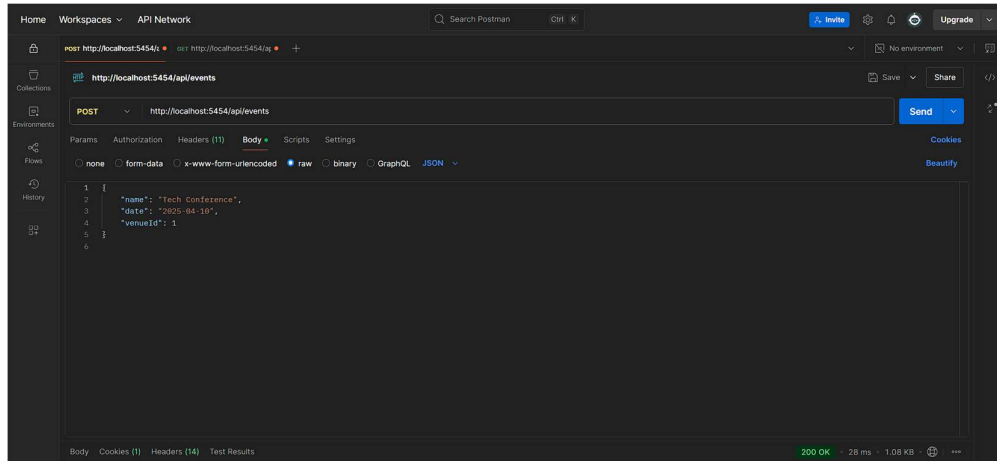
webpack compiled successfully



## Backend:

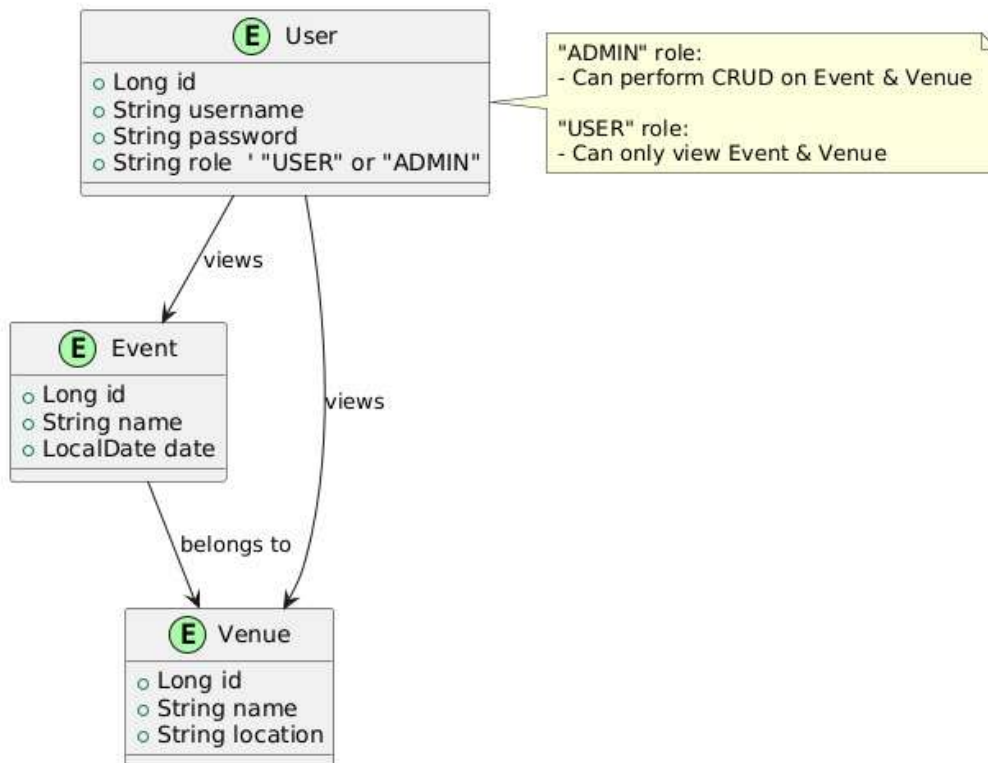
Postman:





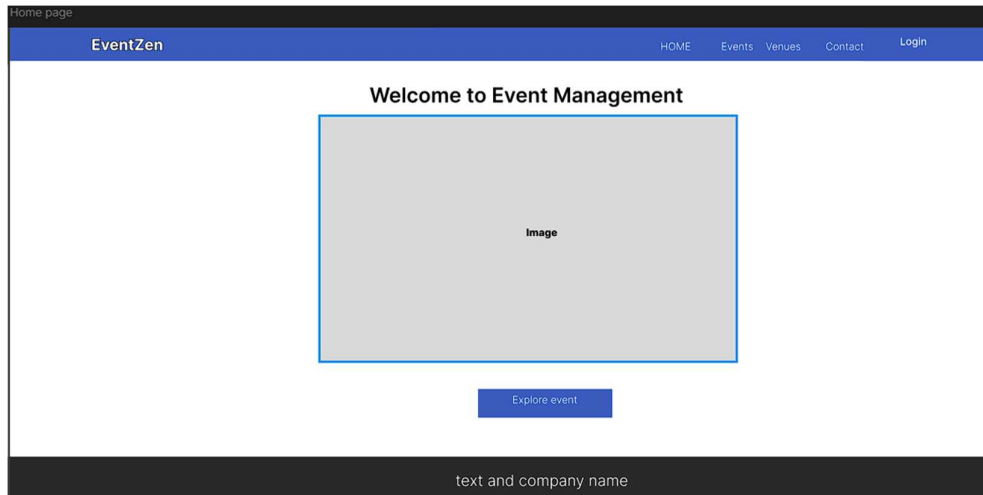
## ER Diagram:

### Event Management - ER Diagram with User/Admin Access

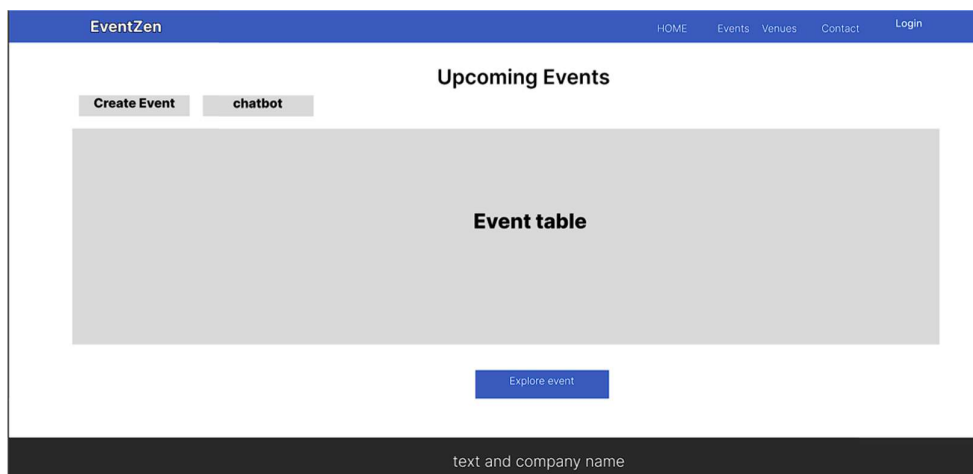


## Wireframe Designs:

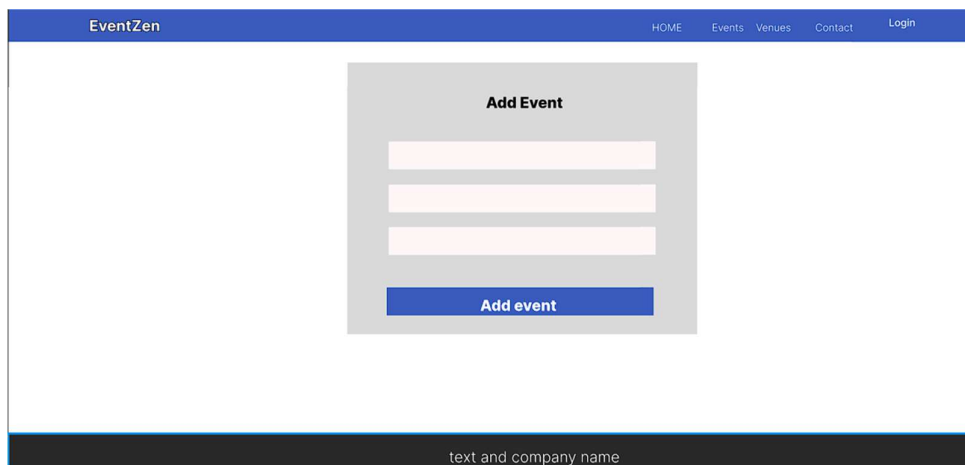
### Home page:



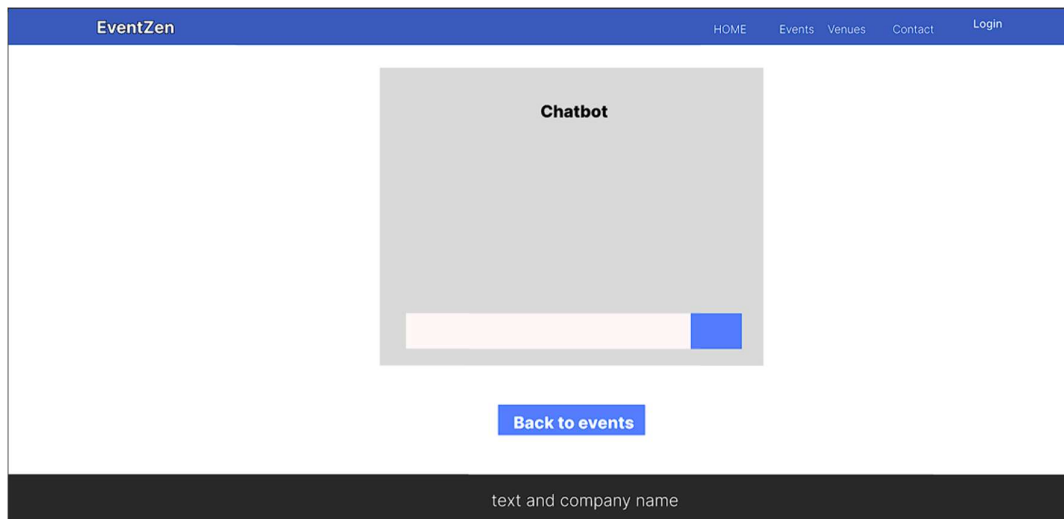
### Events:



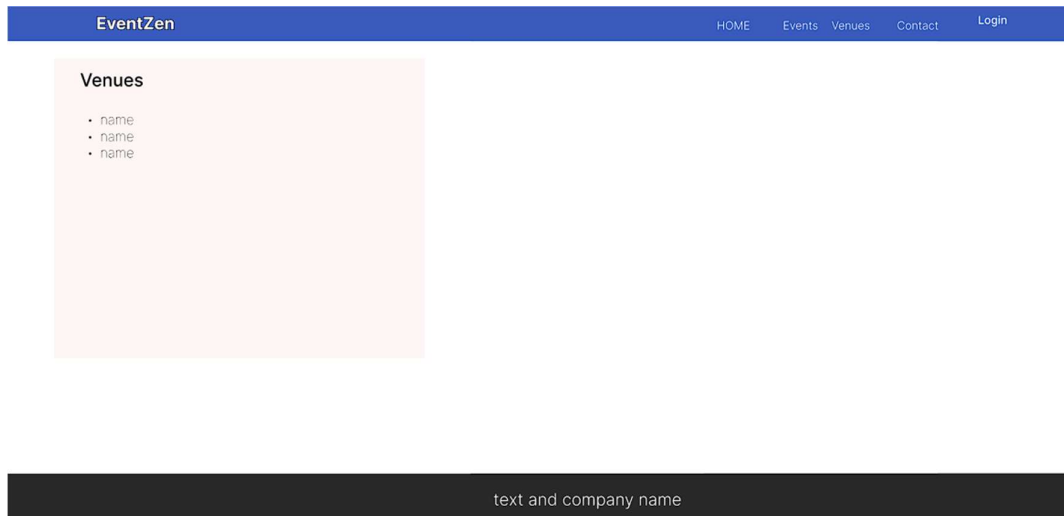
### Form:



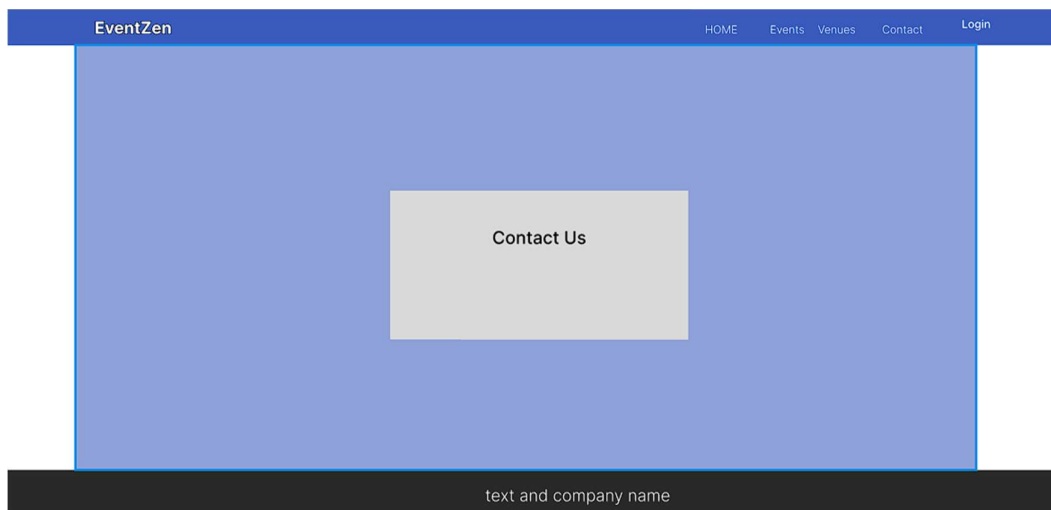
Chatbot:



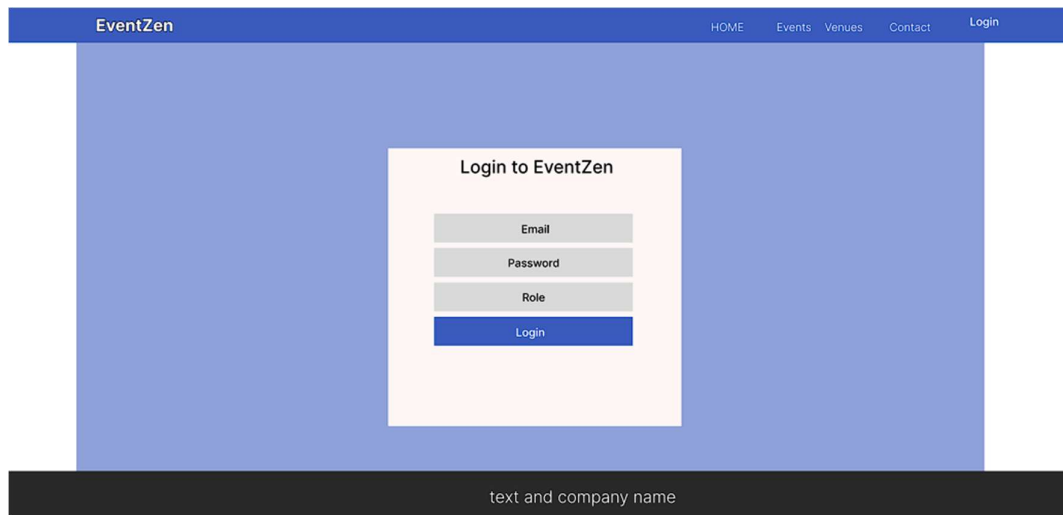
Venues:



Contact:

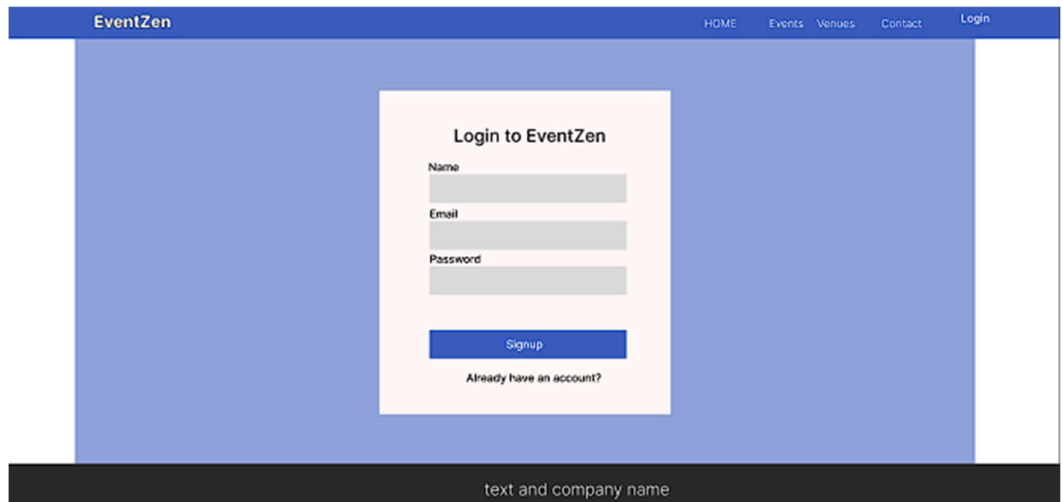


Login:



The screenshot shows the EventZen login page. At the top is a blue navigation bar with the 'EventZen' logo on the left and links for 'HOME', 'Events', 'Venues', 'Contact', and 'Login' on the right. The main content area has a light blue background. In the center is a white box titled 'Login to EventZen'. Inside this box are four input fields: 'Email', 'Password', 'Role', and a blue 'Login' button. Below the white box, a dark grey footer bar contains the text 'text and company name'.

Signup:

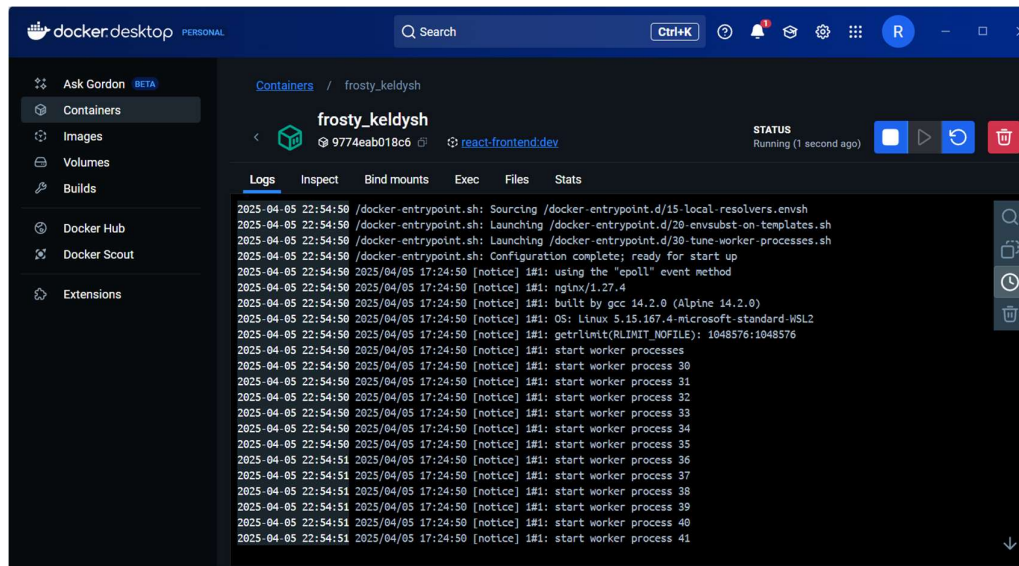


The screenshot shows the EventZen signup page. It features the same blue navigation bar and footer as the login page. The central white box is titled 'Login to EventZen' (though it's a signup form). It contains three input fields: 'Name', 'Email', and 'Password', followed by a blue 'Signup' button. Below the button is a link that says 'Already have an account?'. The footer bar at the bottom contains the text 'text and company name'.

### Docker Configuration:

Docker Command used- `docker run -d -p 3000:80 react-frontend:dev`

- ☐ The frontend runs on port **3000**.
- ☐ The container uses the `react-frontend:dev` image.
- ☐ Backend can be separately dockerized with a Spring Boot container (optional enhancement).



## GitHub Source Code:

Frontend and backend: <https://github.com/Rudra2215/EventManager>

## Conclusion:

This project demonstrates a robust and secure full-stack solution for event and venue management. By utilizing **React and Spring Boot**, the system offers seamless integration and scalability. Dockerization enables simplified deployment across environments.

## Future Enhancements:

- Integrate Email/Push Notifications.
- Add pagination and filters for large event data.
- Role management UI for admins.
- Deploy on cloud platforms using Docker Compose or Kubernetes.

## References:

- [Spring Boot Documentation](#)
- [React Official Docs](#)
- [GeeksforGeeks – Spring Boot Tutorials](#)
- [GeeksforGeeks – React Tutorials](#)
- [Docker Documentation](#)
- [JWT.io](#)
- [Postman API Platform](#)