##### APPENDIX 1

(A typical Specimen of Cover Page & Title Page)

<Font Style Times New Roman – Bold>

**TITLE OF PROJECT REPORT**

AGRICULTURE LOAN PROCESSING SYSTEM

## A PROJECT REPORT

Hall booking System SRS

### Submitted by

NARENDRAN CM

# NAME OF THE CANDIDATE(S)

Arvind M

### in partial fulfillment for the award of the degree of

BACHELORS OF TECHNOLOGY

# NAME OF THE DEGREE

BACHELORS OF TECHNOLOGY

**IN**

#### BRANCH OF STUDY

ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

## SRI KRISHNA COLLEGE OF ENGINEERING AND TECHNOLOGY

**(An Autonomous Institution, Affiliated to Anna University Chennai - 600 025)**

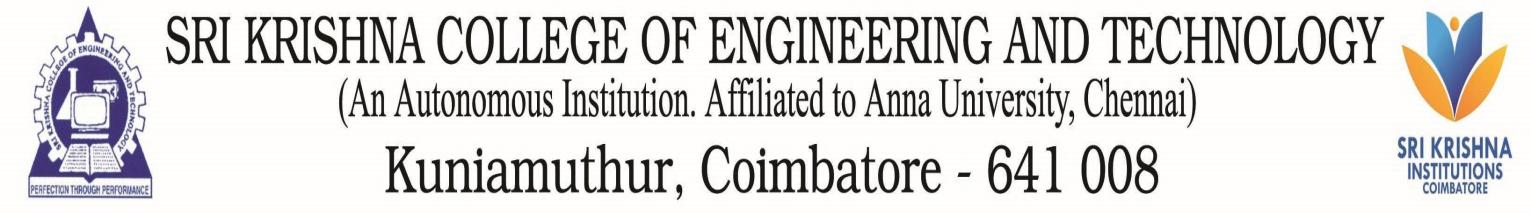
#### MONTH & YEAR

#### 13/8/2024

##### APPENDIX 2

(A typical specimen of Bonafide Certificate)

<Font Style Times New Roman>



# BONAFIDE CERTIFICATE

Certified that this project report **“AGRICULTURE LOAN PROCESSINNG SYSTEM ”**

is the bonafide work of “**…………..ARVIND M** **”**

#### who carried out the project work under my supervision.

##### SIGNATURE SIGNATURE

DR S VENKATA LAKSHIMI

##### HEAD OF THE DEPARTMENT SUPERVISOR

ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

SRI KRISHAN COLLEGE OF ENGINEERING AND TECHNOLGY

KUNIYAMUTHUR,

COIMBATORE,TAMILNADU-641008,

INDIA

Submitted for the Project viva-voce examination held on 13/08/2024

INTERNAL EXAMINER EXTERNAL EXAMINER

##### APPENDIX 3

(A typical specimen of table of contents)

<Font Style Times New Roman>

**TABLE OF CONTENTS**

##### CHAPTER NO. CONTENTS PAGE NO.

|  |  |  |
| --- | --- | --- |
|  | **Abstract** | iii |
| 1 | **Introduction** | xvi |
| 1.1 | General | xviii |
| 1.2 | Technologies used | xxvii |
| 1.2.1 | **Frontend(react)** | 19 |
| 1.2.2 | Backend(Django & MySQL) | 25 |
| 1.2.3 | System Architecture | 29 |
| 1.3 | Project objectives | 30 |
| 1.4 | Scope of the Project | 45 |
| 2. | **Literature Review** | 58 |
| 2.1 | Existing Hall Booking Systems |  |
| 2.2 | Technological Trends |  |
| 2.3 | Django Framework & MySql |  |
| 3. | **System Design And Architecture** |  |
| 3.1 | **Frontend design** |  |
| 3.2 | Backend design |  |
| 3.3 | Database schema(MySql) |  |
| 4. | **Implementation** |  |
| 4.1 | **Frontend Implementation** |  |
| 4.2 | Backend implementation |  |
| 5. | **ER Diagram** |  |
| 6. | **Output(Screenshots)** |  |

### Abstract

The Hall Booking System is a comprehensive and sophisticated web application meticulously designed to address the diverse needs associated with hall reservations and management. As organizations, educational institutions, and event planners increasingly rely on efficient and user-friendly platforms for managing their resources, this system emerges as a solution that seamlessly integrates advanced web technologies to provide an optimal user experience.

At its core, the Hall Booking System serves as a versatile platform that allows users to easily browse through available halls, check their availability in real-time, and make reservations based on specific requirements. This system is particularly beneficial for institutions and businesses that frequently organize events, conferences, meetings, and other gatherings, as it streamlines the entire booking process, reducing the time and effort typically required.

The frontend of the system is built using React, a powerful JavaScript library renowned for its component-based architecture, which facilitates the creation of interactive and dynamic user interfaces. React's ability to efficiently manage the user interface state and render changes in real-time enhances the user experience, making the process of browsing, selecting, and booking halls intuitive and responsive.

The backend of the system is powered by Django, a high-level Python web framework that encourages rapid development and clean, pragmatic design. Django's robust ORM (Object-Relational Mapping) capabilities simplify database interactions, allowing for the secure and efficient handling of user data, booking records, and other critical information. Additionally, Django's built-in security features, such as protection against SQL injection, cross-site scripting, and cross-site request forgery, ensure that the system maintains high standards of data security and user privacy.

The system's database is managed using MySQL, a widely-used relational database management system known for its reliability, scalability, and performance. MySQL's ability to handle large volumes of data and execute complex queries quickly ensures that the Hall Booking System can scale to accommodate growing user bases and increasing amounts of booking data without compromising on performance.

The integration of React, Django, and MySQL within this system is a testament to the power of modern web technologies when combined effectively. The Hall Booking System not only meets the functional requirements of booking and managing halls but also ensures that users and administrators can interact with the system in a secure, efficient, and user-friendly manner. The project underscores the importance of a well-architected system in achieving these goals and highlights the potential for further enhancements and adaptations in response to evolving user needs and technological advancements.

In conclusion, the Hall Booking System represents a significant advancement in the management of hall reservations, offering a streamlined, secure, and user-centric solution that can be adapted to a wide range of organizational needs. Through the integration of cutting-edge technologies, the system demonstrates a commitment to excellence in web development and serves as a model for future projects in this domain.

### Introduction

#### In today’s fast-paced and dynamic environment, the need for efficient and effective resource management systems has become paramount. Halls and event spaces are frequently used by organizations, educational institutions, and businesses for a wide range of activities, from meetings and seminars to large-scale events and conferences. However, managing the availability, booking, and scheduling of these spaces can be a challenging task, often involving complex logistics, multiple stakeholders, and a significant amount of manual effort. This is where a well-designed hall booking system can make a substantial difference.

#### The Hall Booking System is a comprehensive web application designed to automate and streamline the process of managing hall reservations. By leveraging modern web technologies, this system addresses the common challenges associated with hall management, such as double bookings, inefficient use of space, and lack of real-time availability information. The system is designed to provide a user-friendly interface that allows users to view available halls, make reservations, and manage their bookings with ease. Additionally, it offers administrative features that enable system administrators to oversee the booking process, manage user accounts, and generate reports.

#### The system is built on a robust technology stack, with a React-based frontend, a Django-powered backend, and a MySQL database. React, known for its ability to build dynamic and interactive user interfaces, ensures that the frontend is responsive and intuitive. Django, a high-level Python web framework, provides the backend with a secure and scalable foundation, while MySQL handles the database management, ensuring that all data is stored and retrieved efficiently.

#### This project not only focuses on the functional aspects of hall booking but also emphasizes security, scalability, and user experience. By integrating the latest web technologies, the Hall Booking System is designed to meet the needs of a wide range of users, from students and faculty members in educational institutions to corporate event planners and administrators. The system’s architecture is modular, allowing for easy maintenance and future enhancements, making it a long-term solution for organizations looking to improve their hall management processes.

#### In this project report, we will explore the design, development, and implementation of the Hall Booking System. We will begin by discussing the underlying technologies and their roles in the system, followed by a detailed examination of the system’s architecture and features. We will also review the development process, from initial requirements gathering to testing and deployment. Finally, we will analyze the system’s performance, discuss user feedback, and consider potential areas for future improvement.

#### The Hall Booking System represents a significant step forward in the management of shared resources, offering a solution that is both practical and technologically advanced. Through this project, we aim to demonstrate the potential of web-based applications in transforming traditional processes and providing users with a more efficient and enjoyable experience.

#### **Objectives**:

##### The Hall Booking System is designed with several key objectives in mind, focusing on enhancing the efficiency, usability, and security of hall management processes. These objectives serve as the foundation for the system's design and development, ensuring that it meets the needs of both end-users and administrators. The primary objectives of this project are as follows:

##### 1. Streamline the Hall Booking Process

##### The system aims to simplify the process of booking halls by providing an intuitive and user-friendly interface. Users should be able to view the availability of halls in real-time, select their desired time slots, and make reservations with minimal effort. This reduces the complexity and time involved in managing bookings manually.

##### 2. Provide Real-Time Availability Information

##### One of the core objectives is to offer real-time information on hall availability. This feature helps users to quickly identify free slots and avoid conflicts or double bookings. The system should automatically update availability based on new reservations, cancellations, or changes, ensuring that the information provided is always accurate.

##### 3. Ensure Data Security and Privacy

##### The system is designed to protect user data and ensure that all transactions are secure. This includes implementing secure authentication and authorization mechanisms, encrypting sensitive information, and following best practices for data storage and transmission. The objective is to create a safe environment for users to manage their bookings without concerns about data breaches or unauthorized access.

##### 4. Enable Administrative Control and Monitoring

##### Administrators should have full control over the system, including the ability to manage user accounts, monitor booking activities, and generate reports. The objective is to provide administrators with the tools they need to oversee the hall booking process, address any issues that arise, and make informed decisions based on usage data.

##### 5. Facilitate Scalability and Flexibility

##### The system is designed with scalability in mind, allowing it to handle an increasing number of users, halls, and bookings as the organization grows. Additionally, the system should be flexible enough to accommodate future enhancements or changes, such as the addition of new features or integration with other systems.

##### 6. Improve Resource Utilization

##### By providing detailed insights into hall usage patterns, the system aims to help organizations optimize their resource allocation. This includes identifying underutilized halls, peak booking times, and potential areas for improvement in the scheduling process. The objective is to maximize the efficient use of available spaces.

##### 7. Enhance User Experience

##### A key objective is to ensure a positive user experience by making the system easy to use, visually appealing, and responsive across different devices. The system should cater to the needs of various users, including students, faculty, and event organizers, by offering a seamless and enjoyable interaction with the platform.

##### 8.Support for Multiple User Roles

##### The system should accommodate different user roles, such as regular users, administrators, and managers, each with specific access rights and responsibilities. The objective is to tailor the system's functionality according to the needs of each user group, ensuring that all users can perform their tasks efficiently.

##### 9. Generate Comprehensive Reports and Analytics

##### Administrators should be able to generate detailed reports on hall usage, booking trends, and user activity. These reports can be used for decision-making, planning, and improving the overall management process. The objective is to provide actionable insights that can drive continuous improvement in hall management.

##### 10. Ensure Compliance with Organizational Policies

##### The system should be designed to comply with the organization’s policies and regulations regarding resource management, data privacy, and security. This includes adhering to any specific guidelines related to the booking and usage of halls, ensuring that the system operates within the established framework.

##### By meeting these objectives, the Hall Booking System aims to provide a comprehensive solution that addresses the challenges of managing hall reservations, improves user satisfaction, and supports the efficient use of resources.

**Technologies Used:  
In-depth discussion on the technologies used:**

* Frontend (React): Describe React's features, such as component-based architecture, virtual DOM, and hooks. Explain why React was chosen for the frontend, including benefits like performance, maintainability, and developer experience.
* Backend (Django & MySQL): Discuss Django's framework features, including its ORM, built-in admin interface, and scalability. Explain MySQL's role as the relational database management system, focusing on its reliability, performance, and compatibility with Django.

**Frontend (React)   
Detailed explanation of React’s role:**

* Components: Describe major React components like NavBar, Login, Booking, etc. Provide code snippets and explain their functionality.
* State Management: Discuss how state is managed within the application, using hooks like useState and useEffect.
* Routing: Explain React Router's role in handling navigation and routing within the application**.**

**Backend (Django & MySQL)   
Comprehensive details on Django and MySQL:**

* Django Framework: Explain Django's architecture, including models, views, templates, and URLs. Provide code snippets for key features like authentication and booking management.
* MySQL Database: Describe the database schema, relationships between tables, and how data is managed. Include SQL queries and their role in interacting with the database.

**System Architecture   
Detailed diagram and explanation of the system architecture:**

* Frontend and Backend Interaction: Illustrate how the React frontend communicates with the Django backend via APIs.
* Data Flow: Explain the flow of data from the user interface to the backend and database.

**Project Objectives   
List and elaborate on the objectives of the project:**

* Primary Objectives: What the project aims to achieve, such as improving booking efficiency or providing a user-friendly interface.
* Secondary Objectives: Additional goals, such as scalability or integration with other systems.

**Scope of the Project   
Define the scope of the project:**

* Features Included: Describe the features implemented in the system, such as booking, management, and user roles.
* Features Excluded: Mention features not covered in the project, if any.

**Chapter 2: LITERATURE REVIEW**

**2.1 Existing Hall Booking Systems   
Review and analyze existing hall booking systems:**

* Systems Overview: Describe different hall booking systems currently available.
* Strengths and Weaknesses: Analyze the pros and cons of these systems and how they compare to your project.

**2.2 Technological Trends   
Discuss current trends in web development and their relevance:**

* Modern Web Technologies: Explain trends like serverless architecture, microservices, and progressive web apps.
* Impact on Hall Booking Systems: How these trends influence the design and functionality of hall booking systems**.**

**2.3 Django Framework & MySQL   
In-depth look at Django and MySQL:**

**Django Framework:**

* + **Features:** Discuss Django's built-in features such as the ORM, form handling, admin interface, and security measures.
  + **Advantages:** Explain why Django is preferred for web applications, including its robustness, scalability, and ease of use.
* **MySQL:**
  + **Features:** Detail MySQL's capabilities, such as transaction management, indexing, and query optimization.
  + **Advantages:** Highlight its reliability, performance, and widespread adoption in the industry.

**Chapter 3: SYSTEM DESIGN AND ARCHITECTURE**

**3.1 Frontend Design (React Components)   
Detailed description of frontend components:**

* **Component Hierarchy:** Describe the hierarchical structure of components, including parent-child relationships. Provide a visual diagram showing the component tree.
* **Key Components: Detailed explanation of major React components:**
  + **NavBar:** Code and explanation of the navigation bar component.
  + **Login:** Description of the login component, including form handling and validation.
  + **Booking:** Detailed breakdown of the booking component, including the form submission and data handling.
* **Styling:** Discuss the use of CSS modules or styled-components for styling the React components**.**

**3.2 Backend Design (Django Models and APIs)   
Comprehensive overview of backend design:**

**Django Models:**

* + **Models Overview:** Describe the main Django models, such as User, Hall, and Booking**.**
  + **Model Relationships:** Explain the relationships between models (one-to-many, many-to-many) with diagrams.
  + **Code Examples:** Provide code snippets for model definitions and their fields.
* **APIs:**
  + **API Endpoints:** Detailed explanation of API endpoints, such as those for user authentication, hall availability, and booking management.
  + **REST Framework:** Discuss the use of Django REST Framework for creating APIs and handling requests.

**3.3 Database Schema (MySQL)   
Detailed explanation of the database schema:**

* **Schema Design:** Present the database schema with tables and relationships. Include ER diagrams to illustrate the schema.
* **Tables: Describe each table in detail:**
  + **User Table:** Fields, data types, and constraints.
  + **Hall Table:** Attributes like hall ID, name, and capacity.
  + **Booking Table:** Details of the booking table, including foreign keys and indexing.
* **Queries:** Include examples of SQL queries used for data retrieval, insertion, and updating.

**Chapter 4: IMPLEMENTATION**

**4.1 Frontend Implementation (5 pages)  
Detailed implementation of the frontend:**

* **Setup:** Describe the setup process for the React application, including configuration files and dependencies**.**
* **Component Implementation**: Provide code examples and explanations for the implementation of key components.
* **User Interface:** Discuss the design choices and user interface considerations, including responsiveness and accessibility**.**

**4.2 Backend Implementation   
Detailed implementation of the backend:**

* **Setup:** Explain the setup process for the Django project, including configurations and settings.
* **Model Implementation:** Provide code examples for model implementations and database migrations.
* **API Development:** Discuss the development of API endpoints, including serializers, views, and URL routing.

**4.3 Integration and Testing   
Description of the integration and testing process:**

* **Integration:** Explain how the frontend and backend are integrated, including communication via API calls.
* **Testing:** Detail the testing strategies employed, such as unit testing, integration testing, and end-to-end testing. Include code snippets and results from tests.

**Chapter 5: RESULTS AND DISCUSSION (8 pages)**

**5.1 Performance Analysis   
Analysis of system performance:**

* **Benchmarking:** Discuss performance benchmarks, including load times and response times**.**
* **Optimization:** Explain any performance optimization techniques used, such as caching or query optimization**.**

**5.2 User Feedback   
Summary and analysis of user feedback:**

* **Feedback Collection**: Describe the methods used to collect feedback, such as surveys or user testing sessions.
* **Analysis:** Analyze the feedback and discuss any common issues or suggestions. Include any changes or improvements made based on user input**.**

**Chapter 6: CONCLUSION (4 pages)**

**6.1 Summary of Findings (2 pages)  
Recap of the project’s achievements:**

* **Key Outcomes: Summarize the main findings and achievements of the project.**
* **Impact: Discuss the impact of the Hall Booking System on its intended users and stakeholders.**

**6.2 Future Work (2 pages)  
Suggestions for future enhancements:**

* **Potential Improvements: Outline potential improvements or additional features that could be added in future iterations of the project.**
* **Technological Advances: Discuss how emerging technologies could be incorporated into the system.**

**9. Appendices**

**Appendix A: Code Listings   
Complete code listings for key components:**

* **Frontend Code:** Include code snippets for React components, styles, and routing.
* **Backend Code:** Provide code for Django models, views, and API endpoints.
* **SQL Scripts:** List SQL scripts used for database setup and management**.**

**Appendix B: User Manual   
Instructions for using the Hall Booking System:**

* **User Guide:** Provide a detailed user guide with screenshots and explanations for using the system.
* **Administrative Guide:** Instructions for administrators on managing bookings and users.

**Appendix C: Technical Technical details and documentation:**

* **System Architecture:** Detailed architecture diagrams and descriptions**.**
* **API Documentation:** API endpoints documentation, including request and response formats.

**References   
List of references used:**

* **Books:** Cite any textbooks or reference books used**.**
* **Articles:** Include academic or industry articles referenced in the report**.**
* **Online Resources:** List any online resources or tutorials used during the project.

**Home.js**

//components/Home.js  
import React from 'react';  
import './Home.css';  
  
function Home() {  
return (  
<div className="home-container">  
<div className="welcome-message">  
<h2>Find the best place for celebration</h2>  
<p></p>  
</div>  
</div>  
);  
}  
  
export default Home;

**Home.css**

/\* ./components/Home.css \*/  
  
.home-container {  
background-size: cover;  
background-position: center;  
height: 100vh;  
display: flex;  
justify-content: center;  
align-items: center;  
color: black;  
text-align: center;  
padding: 20px;  
width: 100%;  
}  
  
.welcome-message {  
background-color: transparent; /\* Add a semi-transparent background for readability \*/  
padding: 20px;  
border-radius: 10px;  
}  
  
.welcome-message h2 {  
font-size: 2.5em;  
margin-bottom: 10px;  
}  
  
.welcome-message p {  
font-size: 1.2em;  
}

**About.css**

/\* ./components/About.css \*/  
  
.about-page {  
max-width: 800px;  
margin: 0 auto;  
padding: 20px;  
font-family: Arial, sans-serif;  
background-color: skyblue;  
}  
  
.about-page h1 {  
text-align: center;  
color: #333;  
margin-bottom: 20px;  
}  
  
.about-page h2 {  
margin-top: 20px;  
color: #444;  
border-bottom: 2px solid #ddd;  
padding-bottom: 5px;  
}  
  
.about-page p {  
line-height: 1.6;  
color: #666;  
margin-bottom: 20px;  
}  
  
.about-page ol {  
margin: 10px 0;  
padding-left: 20px;  
}  
  
.about-page ul {  
list-style-type: disc;  
margin: 10px 0;  
padding-left: 20px;  
}  
  
.about-page li {  
margin-bottom: 10px;  
}  
  
.about-page .contact-info {  
margin-top: 20px;  
}  
  
.about-page .contact-info ul {  
list-style-type: none;  
padding: 0;  
}  
  
.about-page .contact-info li {  
margin-bottom: 10px;  
font-size: 16px;  
color: #333;  
}  
  
.about-page .contact-info li span {  
font-weight: bold;  
color: #444;  
}

**About.js**

// ./components/About.js  
import React from 'react';  
import './About.css';  
  
function About() {  
return (  
<div className="about-page">  
<h1>About Us</h1>  
<p>  
Welcome to [Hall Booking System SRS ], your premier destination for booking event halls.  
We specialize in offering a wide range of venues for every type of event, whether it’s  
a wedding, corporate function, or birthday celebration. Our goal is to make your event  
planning as smooth and stress-free as possible.  
</p>  
  
<h2>Our Mission</h2>  
<p>  
Our mission is to connect you with the perfect venue for your event. We strive to provide  
an easy-to-use platform where you can browse, book, and manage hall reservations efficiently.  
With our extensive selection of venues and exceptional customer service, we aim to exceed your  
expectations and help you create memorable experiences.  
</p>  
  
<h2>How It Works</h2>  
<p>  
Using our website is simple:  
<ol>  
<li>Browse our list of available halls by location, capacity, and price.</li>  
<li>Select the hall that best suits your needs and check its availability.</li>  
<li>Book your chosen hall directly through our website.</li>  
<li>Receive confirmation and manage your booking through your account.</li>  
</ol>  
</p>  
  
<h2>Why Choose Us?</h2>  
<ul>  
<li><strong>Wide Selection:</strong> Choose from a diverse range of venues to fit any event.</li>  
<li><strong>Ease of Use:</strong> Our user-friendly platform makes booking a breeze.</li>  
<li><strong>Excellent Support:</strong> Our customer service team is here to help with any questions.</li>  
<li><strong>Competitive Pricing:</strong> Find great value and competitive rates for your event.</li>  
</ul>  
  
<h2>Contact Us</h2>  
<p>  
If you have any questions or need assistance, feel free to reach out to our support team:  
</p>  
<ul>  
<li>Email: support@[Hall Booking System SRS].com</li>  
<li>Phone: (9784866787) 456-7890</li>  
<li>Address: 123 Event Lane, Celebration City, EC 12345</li>  
</ul>  
</div>  
);  
}  
  
export default About;

**AddHall.js**

// ./components/AddHall.js  
import React, { useState } from 'react';  
import { useHalls } from '../context/HallContext';  
  
function AddHall() {  
const [name, setName] = useState('');  
const [image, setImage] = useState('');  
const [description, setDescription] = useState('');  
const { addHall } = useHalls();  
  
const handleSubmit = (e) => {  
e.preventDefault();  
const newHall = { name, image, description };  
addHall(newHall);  
setName('');  
setImage('');  
setDescription('');  
};  
  
return (  
<form onSubmit={handleSubmit}>  
<div>  
<label>Hall Name:</label>  
<input type="text" value={name} onChange={(e) => setName(e.target.value)} required />  
</div>  
<div>  
<label>Image URL:</label>  
<input type="text" value={image} onChange={(e) => setImage(e.target.value)} required />  
</div>  
<div>  
<label>Description:</label>  
<textarea value={description} onChange={(e) => setDescription(e.target.value)} required />  
</div>  
<button type="submit">Add Hall</button>  
</form>  
);  
}  
  
export default AddHall;

**/\* Admin.css \*/**  
  
.admin-page {  
width: 80%;  
margin: 50px auto;  
padding: 20px;  
background-color: rgba(135, 207, 235, 0.208);  
border: 1px solid #ddd;  
border-radius: 10px;  
box-shadow: 0 0 10px rgba(1, 0, 0, 0.826);  
}  
  
.admin-page h1 {  
text-align: center;  
margin-bottom: 20px;  
}  
  
.admin-page p {  
text-align: center;  
font-weight: 500;  
margin-bottom: 40px;  
}  
  
.add-hall-form {  
background-color: rgba(255, 255, 255, 0.5);  
padding: 20px;  
border-radius: 10px;  
box-shadow: 0 0 10px rgba(1, 0, 0, 0.1);  
margin-bottom: 40px;  
}  
  
.add-hall-form h2 {  
text-align: center;  
margin-bottom: 20px;  
}  
  
.add-hall-form input[type="text"],  
.add-hall-form input[type="number"],  
.add-hall-form textarea {  
width: 90%;  
margin-bottom: 10px;  
padding: 10px;  
border: 1px solid black;  
border-radius: 5px;  
}  
  
.add-hall-form input[type="text"]:focus,  
.add-hall-form input[type="number"]:focus,  
.add-hall-form textarea:focus {  
border-color: #aaa;  
box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);  
}  
  
.add-hall-form button {  
width: 100%;  
height: 40px;  
background-color: rgba(0, 0, 0, 0.614);  
color: #fff;  
padding: 10px;  
border: none;  
border-radius: 5px;  
cursor: pointer;  
transition: background-color 0.3s;  
}  
  
.add-hall-form button:hover {  
background-color: rgba(128, 128, 128, 0.785);  
transform: scale(1.1);  
}  
  
.hall-list {  
margin-top: 20px;  
display: flex;  
gap: 10%;  
justify-content: center;  
}  
  
h2 {  
text-align: center;  
margin-bottom: 20px;  
}  
  
.hall-card {  
background-color: rgba(255, 255, 255, 0.8);  
margin-bottom: 20px;  
padding: 20px;  
border-radius: 10px;  
box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);  
text-align: center;  
}  
  
.hall-card img.hall-image {  
width: 100%;  
height: auto;  
max-height: 200px;  
object-fit: cover;  
border-radius: 10px;  
margin-bottom: 10px;  
}  
  
.hall-card h3.hall-name {  
font-size: 1.5rem;  
margin-bottom: 10px;  
}  
  
.hall-card p {  
margin-bottom: 10px;  
}  
  
.hall-card .delete-button {  
width: 100%;  
height: 40px;  
background-color: rgba(255, 0, 0, 0.7);  
color: #fff;  
padding: 10px;  
border: none;  
border-radius: 5px;  
cursor: pointer;  
transition: background-color 0.3s;  
}  
  
.hall-card .delete-button:hover {  
background-color: rgba(255, 0, 0, 0.9);  
transform: scale(1.1);  
}  
  
/\* Animation Effects \*/  
.admin-page,  
.add-hall-form,  
.hall-card {  
animation: fadeIn 1s;  
}  
  
@keyframes fadeIn {  
from {  
opacity: 0;  
}  
to {  
opacity: 1;  
}  
}

**// Admin.js**  
import React, { useState } from 'react';  
import { useHalls } from '../context/HallContext';  
import './Admin.css';  
  
function Admin() {  
const { halls, addHall, deleteHall } = useHalls();  
const [newHall, setNewHall] = useState({  
name: '',  
description: '',  
image: '',  
location: '',  
price: '',  
capacity: ''  
});  
  
const handleInputChange = (e) => {  
const { name, value } = e.target;  
setNewHall({  
...newHall,  
[name]: value  
});  
};  
  
const handleAddHall = () => {  
addHall(newHall);  
setNewHall({  
name: '',  
description: '',  
image: '',  
location: '',  
price: '',  
capacity: ''  
});  
};  
  
const handleDeleteHall = (hallId) => {  
deleteHall(hallId);  
};  
  
return (  
<div className="admin-page">  
<h1>Admin Page</h1>  
<p>Manage halls: add, view, and delete halls.</p>  
  
<div className="add-hall-form">  
<h2>Add New Hall</h2>  
<input  
type="text"  
name="name"  
placeholder="Hall Name"  
value={newHall.name}  
onChange={handleInputChange}  
/>  
<textarea  
name="description"  
placeholder="Hall Description"  
value={newHall.description}  
onChange={handleInputChange}  
></textarea>  
<input  
type="text"  
name="image"  
placeholder="Image URL"  
value={newHall.image}  
onChange={handleInputChange}  
/>  
<input  
type="text"  
name="location"  
placeholder="Location"  
value={newHall.location}  
onChange={handleInputChange}  
/>  
<input  
type="text"  
name="price"  
placeholder="Price"  
value={newHall.price}  
onChange={handleInputChange}  
/>  
<input  
type="number"  
name="capacity"  
placeholder="Capacity"  
value={newHall.capacity}  
onChange={handleInputChange}  
/>  
<button onClick={handleAddHall}>Add Hall</button>  
</div>  
  
<h2>Existing Halls</h2>  
<div className="hall-list">  
{halls.map((hall) => (  
<div key={hall.id} className="hall-card">  
<img src={hall.image} alt={hall.name} className="hall-image" />  
<h3 className="hall-name">{hall.name}</h3>  
<p className="hall-description">{hall.description}</p>  
<p className="hall-location"><strong>Location:</strong> {hall.location}</p>  
<p className="hall-capacity"><strong>Hall Capacity:</strong> {hall.capacity}</p>  
<p className="hall-price"><strong>Price:</strong> {hall.price}</p>  
  
<button className="delete-button" onClick={() => handleDeleteHall(hall.id)}>  
Delete  
</button>  
</div>  
))}  
</div>  
</div>  
);  
}  
  
export default Admin;

**/\*BookingForm.css \*/**  
.booking-form {  
padding: 1rem;  
}  
  
.booking-form h1 {  
margin-bottom: 1rem;  
}  
  
.booking-form form {  
display: flex;  
flex-direction: column;  
gap: 1rem;  
}  
  
.booking-form label {  
font-weight: bold;  
}  
  
.booking-form input {  
padding: 0.5rem;  
font-size: 1rem;  
}  
  
.booking-form button {  
background-color: darkgoldenrod;  
color: #fff;  
border: none;  
border-radius: 5px;  
padding: 0.5rem 1rem;  
cursor: pointer;  
font-size: 1rem;  
margin-top: 1rem;  
transition: background-color 0.3s ease;  
}  
  
.booking-form button:hover {  
background-color: #ff9900; /\* Change to your desired hover color \*/  
}

**//BookingForm.js**  
import React, { useState } from 'react';  
import { useParams, useNavigate } from 'react-router-dom';  
import './BookingForm.css'; // Add your styles here  
  
function BookingForm() {  
const { id } = useParams();  
const navigate = useNavigate();  
const [name, setName] = useState('');  
const [eventDate, setEventDate] = useState('');  
const [eventTime, setEventTime] = useState('');  
  
const handleSubmit = (e) => {  
e.preventDefault();  
const bookingDetails = { name, eventDate, eventTime, hallId: id };  
navigate('/payment', { state: bookingDetails });  
};  
  
return (  
<div className="booking-form">  
<h1>Booking Form</h1>  
<form onSubmit={handleSubmit}>  
<div>  
<label>Name:</label>  
<input  
type="text"  
value={name}  
onChange={(e) => setName(e.target.value)}  
required  
/>  
</div>  
<div>  
<label>Event Date:</label>  
<input  
type="date"  
value={eventDate}  
onChange={(e) => setEventDate(e.target.value)}  
required  
/>  
</div>  
<div>  
<label>Event Time:</label>  
<input  
type="time"  
value={eventTime}  
onChange={(e) => setEventTime(e.target.value)}  
required  
/>  
</div>  
<button type="submit">Pay and Book</button>  
</form>  
</div>  
);  
}  
  
export default BookingForm;

**/\* BookingHistory.css \*/**  
.booking-history {  
padding: 1rem;  
background-color: #f4f4f4;  
}  
  
.booking-cards {  
display: flex;  
flex-wrap: wrap;  
gap: 1rem;  
}  
  
.booking-card {  
background: #fff;  
border: 1px solid #ddd;  
border-radius: 4px;  
padding: 1rem;  
width: calc(33.333% - 1rem);  
box-shadow: 0 0 10px rgba(0,0,0,0.1);  
}  
  
.booking-card h3 {  
margin: 0;  
font-size: 1.2rem;  
}  
  
.booking-card p {  
margin: 0.5rem 0;  
}

**Booking History.js**  
import React from 'react';  
import { useBooking } from '../context/BookingContext';  
import './BookingHistory.css'; // Add custom styles for the booking history page  
  
function BookingHistory() {  
const { bookingHistory } = useBooking();  
return (  
<div className="booking-history">  
<h1>Booking History</h1>  
<p>Here is a list of all your past bookings.</p>  
<div className="booking-cards">  
{bookingHistory.length === 0 ? (  
<p>No bookings found.</p>  
) : (  
bookingHistory.map((booking, index) => (  
<div key={index} className="booking-card">  
<h3>{booking.name}</h3>  
<p>Event Date: {booking.eventDate}</p>  
<p>Event Time: {booking.eventTime}</p>  
<p>Hall ID: {booking.hallId}</p>  
</div>  
))  
)}  
</div>  
</div>  
);  
}  
  
export default BookingHistory;

**/\* EventListings.css \*/**  
.event-listings {  
display: flex;  
flex-wrap: wrap;  
gap: 20px;  
justify-content: center;  
  
  
}  
  
.event-card {  
border: 1px solid #ccc;  
border-radius: 8px;  
padding: 16px;  
width: 300px;  
text-align: center;  
box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);  
background-color: darkorange;  
margin-top: 50px;  
}  
  
.event-image {  
width: 100%;  
height: 250px;  
  
border-radius: 8px;  
}  
  
.event-name {  
font-size: 1.2em;  
margin: 12px 0;  
}  
  
.view-button {  
background-color: red;  
color: white;  
border: none;  
padding: 10px 20px;  
border-radius: 5px;  
cursor: pointer;  
}  
  
.view-button:hover {  
background-color: #0056b3;  
}

**// /EventListings.js**  
import React from 'react';  
import { useNavigate } from 'react-router-dom';  
import './EventListings.css';  
  
const events = [  
{  
id: 1,  
name: 'Wedding Hall',  
image: '/images/wedding-hall.jpeg',  
},  
{  
id: 2,  
name: 'Party Hall',  
image: '/images/Party\_hall.jpeg',  
},  
{  
id: 3,  
name: 'Birthday Hall',  
image: '/images/Birthday\_hall.jpeg',  
},  
{  
id: 4,  
name: 'Outdoor Wedding ',  
image: '/images/outdoor\_Wedding.jpeg',  
},  
];  
  
function EventListings() {  
const navigate = useNavigate();  
  
const handleViewClick = (id) => {  
navigate(`/halls/${id}`);  
};  
  
return (  
<div className="event-listings">  
{events.map(event => (  
<div key={event.id} className="event-card">  
<img src={event.image} alt={event.name} className="event-image" />  
<h3 className="event-name">{event.name}</h3>  
<button className="view-button" onClick={() => handleViewClick(event.id)}>View</button>  
</div>  
))}  
</div>  
);  
}  
  
export default EventListings;

**Footer.css**

footer {  
padding: 10px 20px;  
color: white;  
text-align: center;  
position: fixed;  
bottom: 0;  
width: 100%;  
font-size: 0.9em;  
background-color: rgba(0, 0, 0, 0.300);  
backdrop-filter: blur(2px);  
}  
  
footer p {  
margin: 0;  
color: whitesmoke;  
}

**//Footer.js**  
import React from 'react';  
import { Link } from 'react-router-dom';  
import './Footer.css';  
  
function Footer() {  
return (  
<footer>  
<p></p>  
<nav className="footer-nav">  
<Link to="/about">About Us</Link>  
</nav>  
</footer>  
);  
}  
  
export default Footer;

**/\* Hall.css \*/**  
  
.hall-list {  
max-width: 1200px;  
margin: 0 auto;  
padding: 20px;  
text-align: center;  
}  
  
.hall-list h1 {  
font-size: 2.5rem;  
margin-bottom: 20px;  
color: #333;  
}  
  
.hall-list p {  
font-size: 1.2rem;  
color: #666;  
margin-bottom: 30px;  
}  
  
.hall-cards {  
display: flex;  
flex-wrap: wrap;  
justify-content: center;  
gap: 20px;  
}  
  
.hall-card {  
background-color: #fff;  
border-radius: 10px;  
box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);  
overflow: hidden;  
width: 320px; /\* Increased width \*/  
transition: transform 0.3s ease;  
display: flex;  
flex-direction: column;  
justify-content: space-between;  
  
}  
  
.hall-card:hover {  
transform: translateY(-5px);  
}  
  
.hall-image {  
width: 100%;  
height: 300px; /\* Increased height \*/  
object-fit: cover;  
}  
  
.hall-name {  
font-size: 1.5rem;  
margin: 15px 15px 10px; /\* Reduced bottom margin \*/  
color: #333;  
}  
  
.hall-description {  
padding: 0 15px;  
margin: 0 0 10px; /\* Reduced bottom margin \*/  
color: #555;  
text-align: left;  
font-size: 1rem;  
flex-grow: 1;  
}  
  
.hall-location,  
.hall-price,  
.hall-capacity {  
margin: 5px 15px 0; /\* Reduced bottom margin \*/  
font-size: 1rem;  
color: #555;  
text-align: left;  
}  
  
.book-button {  
background-color: #007bff;  
color: #fff;  
border: none;  
border-radius: 5px;  
padding: 10px 20px;  
margin: 15px;  
font-size: 1rem;  
cursor: pointer;  
transition: background-color 0.3s ease;  
}  
  
.book-button:hover {  
background-color: #0056b3;  
}  
  
@media (max-width: 768px) {  
.hall-card {  
width: 100%;  
margin: 0 0 20px;  
}  
}

// ./components/Hall.js  
import React from 'react';  
import { useNavigate } from 'react-router-dom';  
import { useHalls } from '../context/HallContext';  
import './Hall.css'; // You can add custom styles for Hall cards here  
  
function Hall() {  
const { halls } = useHalls();  
const navigate = useNavigate();  
  
const handleBookNow = (hallId) => {  
navigate(`/book/${hallId}`);  
};  
  
return (  
<div>  
<h2>Welcome to the hall page</h2>  
<h2>Here you can view all halls</h2>  
<div className="hall-list">  
<div className="hall-cards">  
{halls.map((hall, index) => (  
<div key={index} className="hall-card">  
<img src={hall.image} alt={hall.name} className="hall-image" />  
<h3 className="hall-name">{hall.name}</h3>  
<p className="hall-description">{hall.description}</p>  
<p className="hall-location"><strong>Location:</strong> {hall.location}</p>  
<p className="hall-price"><strong>Price:</strong> {hall.price}</p>  
<p className="hall-capacity"><strong>Capacity:</strong> {hall.capacity}</p>  
<button className="book-button" onClick={() => handleBookNow(hall.id)}>  
Book Now  
</button>  
</div>  
))}  
</div>  
</div>  
</div>  
);  
}  
  
export default Hall;

**/\* Header.css \*/**  
.header-container {  
text-align: center;  
padding: 20px;  
background-color: rgba(0, 0, 0, 0.300) ;  
backdrop-filter: blur(2px);  
}  
  
.header-title {  
font-size: 2.5em;  
color: white;  
margin: 0;  
}

// ./components/Header.js  
import React from 'react';  
import './Header.css';  
  
function Header() {  
return (  
<div className="header-container">  
<h1 className="header-title">Hall Booking System</h1>  
</div>  
);  
}  
  
export default Header;

**/\* Login.css \*/**  
  
.login {  
width: 300px;  
margin: 50px auto;  
padding: 20px;  
background-color: rgba(135, 207, 235, 0.208);  
border: 1px solid #ddd;  
  
border-radius: 10px;  
box-shadow: 0 0 10px rgba(1, 0, 0, 0.826);  
}  
  
.login h2 {  
text-align: center;  
margin-bottom: 20px;  
  
}  
  
.login form {  
margin-top: 10px;  
  
}  
  
.login label {  
display: block;  
margin-bottom: 10px;  
}  
  
.login input[type="email"],  
.login input[type="password"] {  
width: 90%;  
height: 10px;  
margin-bottom: 10px;  
padding: 10px;  
border: 1px solid black;  
border-radius: 5px;  
}  
  
.login input[type="email"]:focus,  
.login input[type="password"]:focus {  
border-color: #aaa;  
box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);  
}  
  
.login button[type="submit"] {  
width: 100%;  
height: 40px;  
background-color: rgba(0, 0, 0, 0.614);  
color: #fff;  
padding: 10px;  
border: none;  
border-radius: 5px;  
cursor: pointer;  
}  
  
.login button[type="submit"]:hover {  
background-color:rgba(128, 128, 128, 0.785);  
}  
  
.login .error {  
color: #f00;  
font-size: 14px;  
margin-bottom: 20px;  
}  
  
.login p {  
text-align: center;  
margin-top: 20px;  
font-weight:500 ;  
  
}  
  
/\* Effects \*/  
  
.login {  
animation: fadeIn 1s;  
}  
  
@keyframes fadeIn {  
from {  
opacity: 0;  
}  
to {  
opacity: 1;  
}  
}  
  
.login button[type="submit"] {  
transition: background-color 0.3s;  
}  
  
.login button[type="submit"]:hover {  
transform: scale(1.1);  
}  
  
.login input[type="email"]:focus,  
.login input[type="password"]:focus {  
transition: border-color black, box-shadow cubic-bezier(0.165, 0.84, 0.44, 1);  
}

**login.css**

import React, { useState } from 'react';  
import { useNavigate } from 'react-router-dom';  
import './Login.css';  
  
function Login() {  
const [email, setEmail] = useState('');  
const [password, setPassword] = useState('');  
const [error, setError] = useState('');  
const [currentUser, setCurrentUser] = useState(null);  
const navigate = useNavigate();  
  
const login = async (email, password) => {  
try {  
// Use JSON Server to query the database for matching email and password  
const response = await fetch(`http://localhost:8000/users?email=${encodeURIComponent(email)}&password=${encodeURIComponent(password)}`);  
if (!response.ok) {  
throw new Error('Login failed');  
}  
  
const data = await response.json();  
if (data.length === 0) {  
throw new Error('Invalid email or password');  
}  
  
setCurrentUser(data[0]); // Assuming the response is an array with user objects  
navigate('/'); // Navigate to the home page or wherever you want after login  
} catch (err) {  
setError(err.message);  
}  
};  
  
const handleSubmit = async (e) => {  
e.preventDefault();  
setError('');  
  
// Call the login function with email and password  
await login(email, password);  
};  
  
return (  
<div className="login">  
<h2>Login</h2>  
{error && <p className="error">{error}</p>}  
<form onSubmit={handleSubmit}>  
<div>  
<label>Email:</label>  
<input  
type="email"  
value={email}  
onChange={(e) => setEmail(e.target.value)}  
required  
/>  
</div>  
<div>  
<label>Password:</label>  
<input  
type="password"  
value={password}  
onChange={(e) => setPassword(e.target.value)}  
required  
/>  
</div>  
<button type="submit">Login</button>  
</form>  
{currentUser && <p>Logged in as {currentUser.email}</p>}  
</div>  
);  
}  
  
export default Login;

**/\* ./components/NavBar.css \*/**

.navbar {

color: #fff;

padding: 1rem;

display: flex;

justify-content: space-between;

align-items: center;

background-color: rgba(0, 0, 0, 0.300);

backdrop-filter: blur(2px);

}

.navbar ul {

list-style: none;

display: flex;

gap: 1rem;

}

.navbar ul li {

margin-right: 1rem;

}

.navbar ul li a {

text-decoration: none;

color: #fff;

font-weight: bold;

transition: color 0.3s ease;

}

.navbar ul li a:hover {

color: #ff9900; /\* Change to your desired hover color \*/

}

**// ./components/NavBar.js**

import React from 'react';

import { Link } from 'react-router-dom';

import './NavBar.css';

import { useNavigate } from 'react-router-dom';

function NavBar() {

const navigate = useNavigate()

return (

<nav className="navbar">

<ul>

<li><Link to="/signup">Signup</Link></li>

<li><Link to="/login">User Login</Link></li>

<li><Link to="/">Home</Link></li>

<li><Link to="/halls">Hall</Link></li>

<li><Link to="/booking-history">Booking History</Link></li>

<li><Link to="/admin-login">Admin Login</Link></li> {/\* Admin Login Link \*/}

<li><Link to="/admin">Admin</Link></li>

<li><button onClick={(e) => {

localStorage.setItem('user', '')

localStorage.setItem('userRole', '')

navigate('/login')

}}> Logout </button></li>

</ul>

</nav>

);

}

export default NavBar;

**/\* ./components/PaymentPage.css \*/**

.payment-page {

padding: 1rem;

}

.payment-page h1 {

margin-bottom: 1rem;

}

.payment-page form {

display: flex;

flex-direction: column;

gap: 1rem;

}

.payment-page label {

font-weight: bold;

}

.payment-page input {

padding: 0.5rem;

font-size: 1rem;

}

.payment-page button {

background-color: darkgoldenrod;

color: #fff;

border: none;

border-radius: 5px;

padding: 0.5rem 1rem;

cursor: pointer;

font-size: 1rem;

margin-top: 1rem;

transition: background-color 0.3s ease;

}

.payment-page button:hover {

background-color: #ff9900; /\* Change to your desired hover color \*/

}

**// ./components/PaymentPage.js**

import React from 'react';

import { useNavigate, useLocation } from 'react-router-dom';

import { useBooking } from '../context/BookingContext'; // Ensure this path is correct

import './PaymentPage.css'; // Add your styles here

function PaymentPage() {

const navigate = useNavigate();

const location = useLocation();

const { addBooking } = useBooking();

const handlePayment = (e) => {

e.preventDefault();

// Extract booking details from location state or other sources

const bookingDetails = location.state || {};

// Save the booking details to the context

addBooking(bookingDetails);

// Redirect to a confirmation page or home

navigate('/');

};

return (

<div className="payment-page">

<h1>Payment Page</h1>

<p>Please enter your payment details to complete the booking.</p>

<form onSubmit={handlePayment}>

<div>

<label>Card Number:</label>

<input type="text" placeholder="1234 5678 9012 3456" required />

</div>

<div>

<label>Expiry Date:</label>

<input type="text" placeholder="MM/YY" required />

</div>

<div>

<label>CVV:</label>

<input type="text" placeholder="123" required />

</div>

<button type="submit">Pay Now</button>

</form>

</div>

);

}

export default PaymentPage;

**/\* Signup.css \*/**

.signup {

width: 300px;

margin: 50px auto;

padding: 20px;

background-color: rgba(135, 207, 235, 0.208);

border: 1px solid #ddd;

border-radius: 10px;

box-shadow: 0 0 10px rgba(1, 0, 0, 0.826);

}

.signup h2 {

text-align: center;

margin-bottom: 20px;

}

.signup form {

margin-top: 10px;

}

.signup label {

display: block;

margin-bottom: 10px;

}

.signup input[type="email"],

.signup input[type="password"] {

width: 90%;

height: 10px;

margin-bottom: 10px;

padding: 10px;

border: 1px solid black;

border-radius: 5px;

}

.signup input[type="email"]:focus,

.signup input[type="password"]:focus {

border-color: #aaa;

box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);

}

.signup button[type="submit"] {

width: 100%;

height: 40px;

background-color: rgba(0, 0, 0, 0.614);

color: #fff;

padding: 10px;

border: none;

border-radius: 5px;

cursor: pointer;

}

.signup button[type="submit"]:hover {

background-color: rgba(128, 128, 128, 0.785);

}

.signup .error {

color: #f00;

font-size: 14px;

margin-bottom: 20px;

}

.signup p {

text-align: center;

margin-top: 20px;

font-weight: 500;

}

/\* Effects \*/

.signup {

animation: fadeIn 1s;

}

@keyframes fadeIn {

from {

opacity: 0;

}

to {

opacity: 1;

}

}

.signup button[type="submit"] {

transition: background-color 0.3s;

}

.signup button[type="submit"]:hover {

transform: scale(1.1);

}

.signup input[type="email"]:focus,

.signup input[type="password"]:focus {

transition: border-color black, box-shadow cubic-bezier(0.165, 0.84, 0.44, 1);

}

**//Sign.css**

import React, { useState } from 'react';

import './Signup.css';

function Signup() {

const [email, setEmail] = useState('');

const [password, setPassword] = useState('');

const [confirmPassword, setConfirmPassword] = useState('');

const [error, setError] = useState('');

const handleSubmit = async (e) => {

e.preventDefault();

if (password !== confirmPassword) {

setError('Passwords do not match');

return;

}

try {

const response = await fetch('http://localhost:8000/users', {

method: 'POST',

headers: {

'Content-Type': 'application/json',

},

body: JSON.stringify({ email, password }),

});

if (!response.ok) {

throw new Error('Signup failed');

}

// Handle successful signup (e.g., redirect to login page)

// Example: navigate('/login');

alert('Signup successful! Please log in.');

} catch (err) {

setError(err.message);

}

};

return (

<div className="signup">

<h2>Signup</h2>

{error && <p className="error">{error}</p>}

<form onSubmit={handleSubmit}>

<label>Email</label>

<input

type="email"

value={email}

onChange={(e) => setEmail(e.target.value)}

required

/>

<label>Password</label>

<input

type="password"

value={password}

onChange={(e) => setPassword(e.target.value)}

required

/>

<label>Confirm Password</label>

<input

type="password"

value={confirmPassword}

onChange={(e) => setConfirmPassword(e.target.value)}

required

/>

<button type="submit">Signup</button>

</form>

</div>

);

}

export default Signup;

**AuthContext.js**

import React, { createContext, useContext, useState, useEffect } from 'react';

const AuthContext = createContext();

export function useAuth() {

return useContext(AuthContext);

}

export function AuthProvider({ children }) {

const [currentUser, setCurrentUser] = useState(null);

const [loading, setLoading] = useState(true);

const [error, setError] = useState(null);

useEffect(() => {

// Safely parse the user from localStorage

const loadUser = async () => {

try {

const storedUser = localStorage.getItem('user');

if (storedUser) {

setCurrentUser(JSON.parse(storedUser));

console.log(storedUser);

}

} catch (err) {

console.error('Failed to load user from local storage', err);

} finally {

setLoading(false);

}

};

loadUser();

}, []);

const login = async (email, password) => {

try {

setLoading(true);

setError(null);

const response = await fetch(`http://localhost:8000/users?email=${email}&password=${password}`);

const data = await response.json();

if (response.ok && data.length > 0) {

const user = data[0];

localStorage.setItem('user', JSON.stringify(user));

setCurrentUser(user);

return user;

} else {

throw new Error('Invalid email or password');

}

} catch (err) {

setError(err.message);

throw err;

} finally {

setLoading(false);

}

};

const signup = async (email, password) => {

try {

setLoading(true);

setError(null);

const response = await fetch('http://localhost:8000/users', {

method: 'POST',

headers: {

'Content-Type': 'application/json',

},

body: JSON.stringify({ email, password }),

});

if (!response.ok) {

throw new Error('Signup failed');

}

const user = await response.json();

localStorage.setItem('user', JSON.stringify(user));

setCurrentUser(user);

return user;

} catch (err) {

setError(err.message);

throw err;

} finally {

setLoading(false);

}

};

const logout = () => {

localStorage.removeItem('user');

setCurrentUser(null);

};

const value = {

currentUser,

login,

signup, // Add the signup function to the context

logout,

error,

};

return (

<AuthContext.Provider value={value}>

{!loading && children}

</AuthContext.Provider>

);

}

**BookingContext.js**

import React, { createContext, useContext, useState, useEffect } from 'react';

const BookingContext = createContext();

export function useBooking() {

return useContext(BookingContext);

}

export function BookingProvider({ children }) {

const [bookingHistory, setBookingHistory] = useState([]);

useEffect(() => {

fetchBookingHistory();

}, []);

const fetchBookingHistory = async () => {

try {

const response = await fetch('http://localhost:3001/bookings');

const data = await response.json();

setBookingHistory(data);

} catch (err) {

console.error('Failed to fetch booking history:', err);

}

};

const addBooking = async (newBooking) => {

try {

const response = await fetch('http://localhost:3001/bookings', {

method: 'POST',

headers: {

'Content-Type': 'application/json',

},

body: JSON.stringify(newBooking),

});

const addedBooking = await response.json();

setBookingHistory([...bookingHistory, addedBooking]);

} catch (err) {

console.error('Failed to add booking:', err);

}

};

const value = {

bookingHistory,

addBooking,

};

return (

<BookingContext.Provider value={value}>

{children}

</BookingContext.Provider>

);

}

**HallContext.js**

// ./context/HallContext.js

import React, { createContext, useState, useEffect, useContext } from 'react';

const HallContext = createContext();

export function HallProvider({ children }) {

const [halls, setHalls] = useState([]);

useEffect(() => {

const fetchHalls = async () => {

try {

const response = await fetch('http://localhost:8000/halls');

if (response.ok) {

const data = await response.json();

setHalls(data);

} else {

throw new Error('Failed to fetch halls');

}

} catch (err) {

console.error(err);

}

};

fetchHalls();

}, []);

const addHall = async (newHall) => {

try {

const response = await fetch('http://localhost:8000/halls', {

method: 'POST',

headers: { 'Content-Type': 'application/json' },

body: JSON.stringify(newHall)

});

if (response.ok) {

const addedHall = await response.json();

setHalls((prevHalls) => [...prevHalls, addedHall]);

} else {

throw new Error('Failed to add hall');

}

} catch (err) {

console.error(err);

}

};

const deleteHall = async (hallId) => {

try {

const response = await fetch(`http://localhost:8000/halls/${hallId}`, {

method: 'DELETE'

});

if (response.ok) {

setHalls((prevHalls) => prevHalls.filter((hall) => hall.id !== hallId));

} else {

throw new Error('Failed to delete hall');

}

} catch (err) {

console.error(err);

}

};

return (

<HallContext.Provider value={{ halls, addHall, deleteHall }}>

{children}

</HallContext.Provider>

);

}

export function useHalls() {

return useContext(HallContext);

}

**Admin.py**

from django.contrib import admin

from .models import User, Hall, Booking

admin.site.register(User)

admin.site.register(Hall)

admin.site.register(Booking)

**Apps.py**

from django.apps import AppConfig

class MyappConfig(AppConfig):

default\_auto\_field = 'django.db.models.BigAutoField'

name = 'myapp'

**models.py**

from django.db import models

class User(models.Model):

email = models.EmailField(unique=True)

password = models.CharField(max\_length=128)

def \_\_str\_\_(self):

return self.email

class Hall(models.Model):

name = models.CharField(max\_length=255)

description = models.TextField()

capacity = models.IntegerField()

price = models.DecimalField(max\_digits=10, decimal\_places=2)

availability = models.BooleanField(default=True)

location = models.CharField(max\_length=255)

image = models.URLField(max\_length=500, blank=True, default='')

def \_\_str\_\_(self):

return self.name

class Booking(models.Model):

hall = models.ForeignKey(Hall, on\_delete=models.CASCADE)

user = models.ForeignKey(User, on\_delete=models.CASCADE)

payment\_method = models.CharField(max\_length=50, blank=True, null=True)

date = models.DateField()

time = models.TimeField(default='00:00')

booking\_date = models.DateTimeField(auto\_now\_add=True)

number\_of\_attendees = models.IntegerField(blank=True, null=True)

name = models.CharField(max\_length=255, default='default\_name')

def \_\_str\_\_(self):

return f"Booking for {self.hall.name} by {self.user.email or 'unknown user'}"

**serlizers.py**

from rest\_framework import serializers

from django.contrib.auth.hashers import make\_password

from .models import User, Hall, Booking

class UserSerializer(serializers.ModelSerializer):

class Meta:

model = User

fields = ['id', 'email', 'password']

extra\_kwargs = {

'password': {'write\_only': True}

}

def create(self, validated\_data):

password = validated\_data.pop('password', None)

if password:

validated\_data['password'] = make\_password(password)

return super().create(validated\_data)

def update(self, instance, validated\_data):

password = validated\_data.pop('password', None)

instance = super().update(instance, validated\_data)

if password:

instance.password = make\_password(password)

instance.save()

return instance

def to\_representation(self, instance):

representation = super().to\_representation(instance)

representation.pop('password', None)

return representation

class HallSerializer(serializers.ModelSerializer):

class Meta:

model = Hall

fields = ['id', 'name', 'description', 'capacity', 'price', 'availability', 'location', 'image']

class BookingSerializer(serializers.ModelSerializer):

hall\_name = serializers.ReadOnlyField(source='hall.name')

user\_email = serializers.ReadOnlyField(source='user.email')

class Meta:

model = Booking

fields = [

'id', 'hall', 'hall\_name', 'user', 'user\_email',

'payment\_method', 'date', 'time', 'booking\_date',

'number\_of\_attendees', 'name'

]

extra\_kwargs = {

'payment\_method': {'required': False, 'allow\_null': True},

'number\_of\_attendees': {'required': False, 'allow\_null': True}

}

**Urls.py**

from django.urls import path, include

from rest\_framework.routers import DefaultRouter

from .views import UserViewSet

router = DefaultRouter()

router.register(r'users', UserViewSet)

urlpatterns = [

path('', include(router.urls)),

]

**Views.py**

from rest\_framework.decorators import action

from rest\_framework import status, viewsets

from rest\_framework.response import Response

from django.contrib.auth.hashers import make\_password, check\_password

from .models import User

from .serializers import UserSerializer

class UserViewSet(viewsets.ModelViewSet):

queryset = User.objects.all()

serializer\_class = UserSerializer

@action(detail=False, methods=['post'], url\_path='signup')

def signup(self, request):

email = request.data.get('email')

password = request.data.get('password')

if User.objects.filter(email=email).exists():

return Response({'error': 'Email already registered'}, status=status.HTTP\_400\_BAD\_REQUEST)

user = User.objects.create(

email=email,

password=make\_password(password)

)

return Response({

'id': user.id,

'email': user.email

}, status=status.HTTP\_201\_CREATED)

@action(detail=False, methods=['post'], url\_path='login')

def login(self, request):

email = request.data.get('email')

password = request.data.get('password')

try:

user = User.objects.get(email=email)

if check\_password(password, user.password):

return Response({

'id': user.id,

'email': user.email

}, status=status.HTTP\_200\_OK)

else:

return Response({'error': 'Invalid credentials'}, status=status.HTTP\_400\_BAD\_REQUEST)

except User.DoesNotExist:

return Response({'error': 'User not found'}, status=status.HTTP\_404\_NOT\_FOUND)

**Manage.py**

"""Django's command-line utility for administrative tasks."""

import os

import sys

def main():

"""Run administrative tasks."""

os.environ.setdefault('DJANGO\_SETTINGS\_MODULE', 'hall\_booking\_backend.settings')

try:

from django.core.management import execute\_from\_command\_line

except ImportError as exc:

raise ImportError(

"Couldn't import Django. Are you sure it's installed and "

"available on your PYTHONPATH environment variable? Did you "

"forget to activate a virtual environment?"

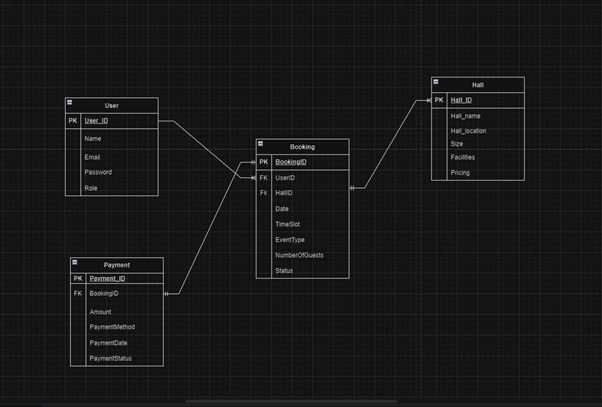
) from exc

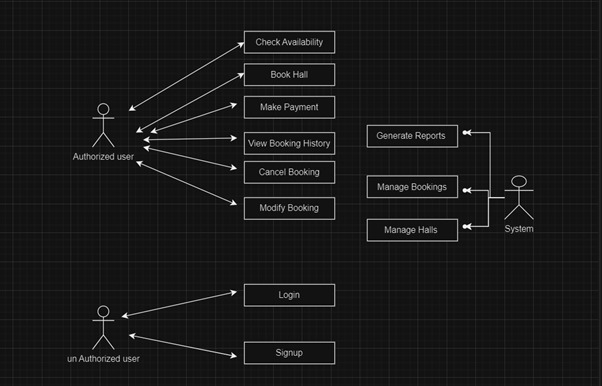
execute\_from\_command\_line(sys.argv)

if \_\_name\_\_ == '\_\_main\_\_':

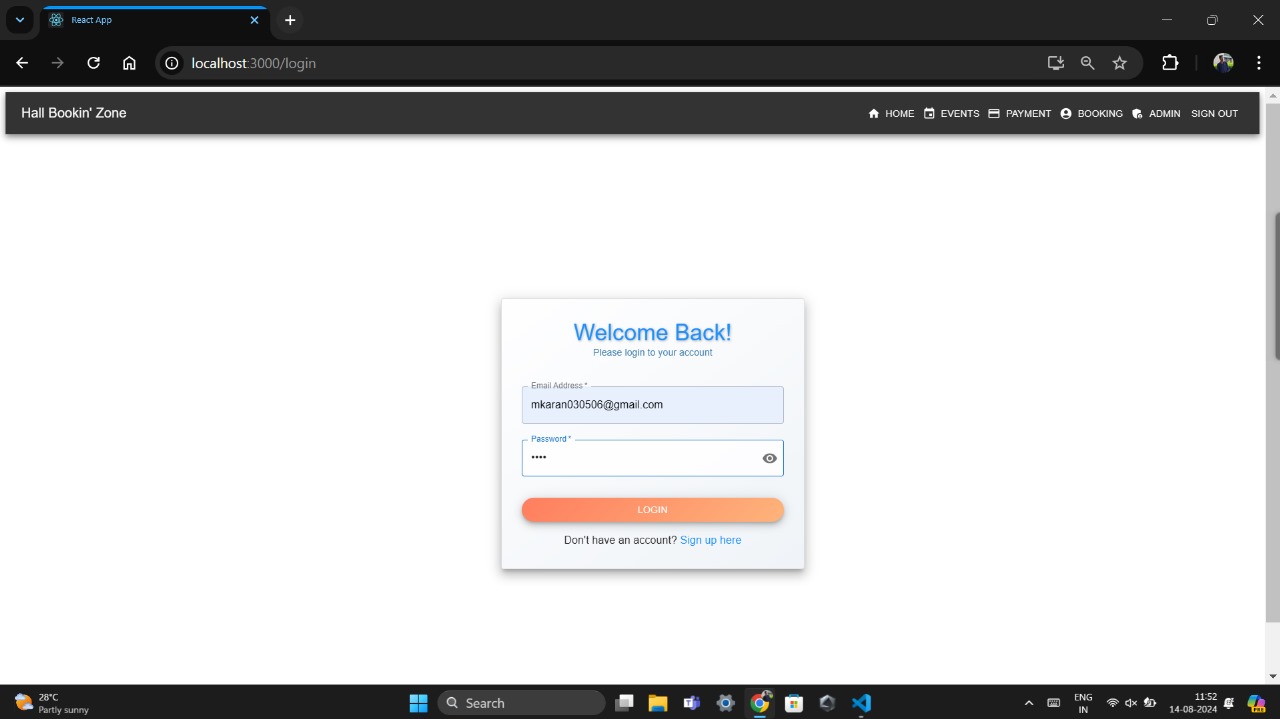
main()

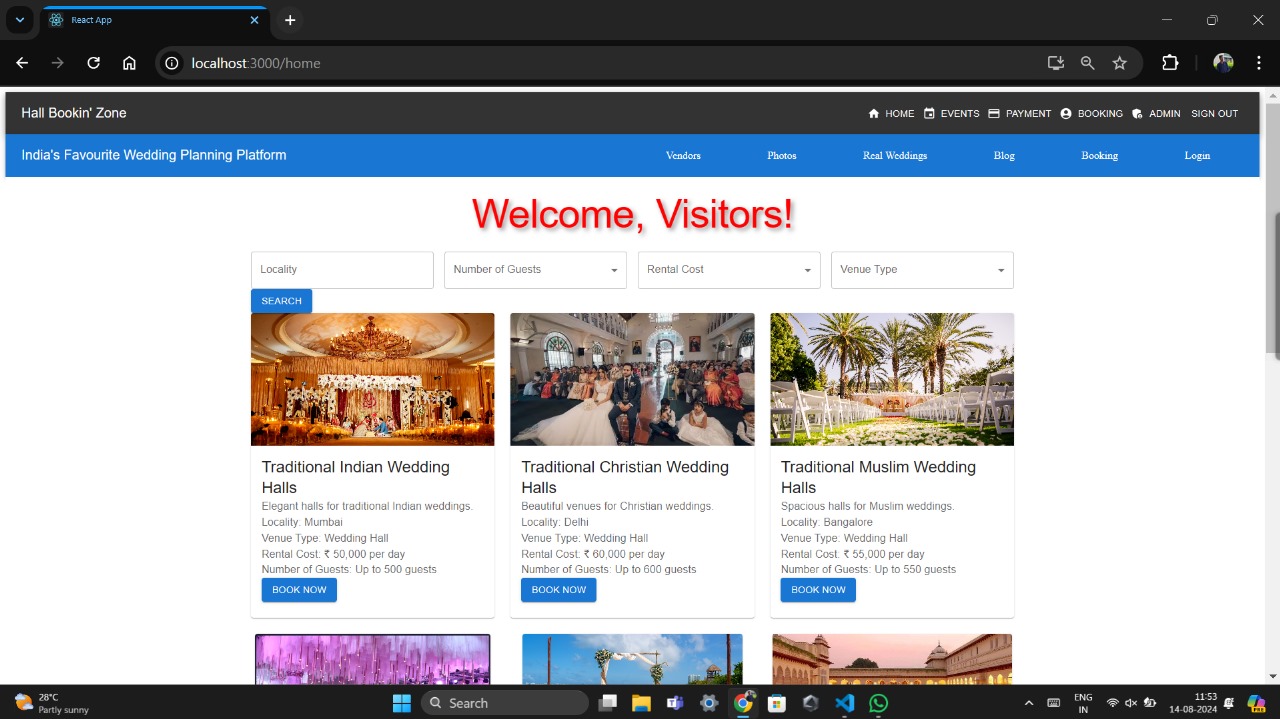
**ER DIAGRAM:**

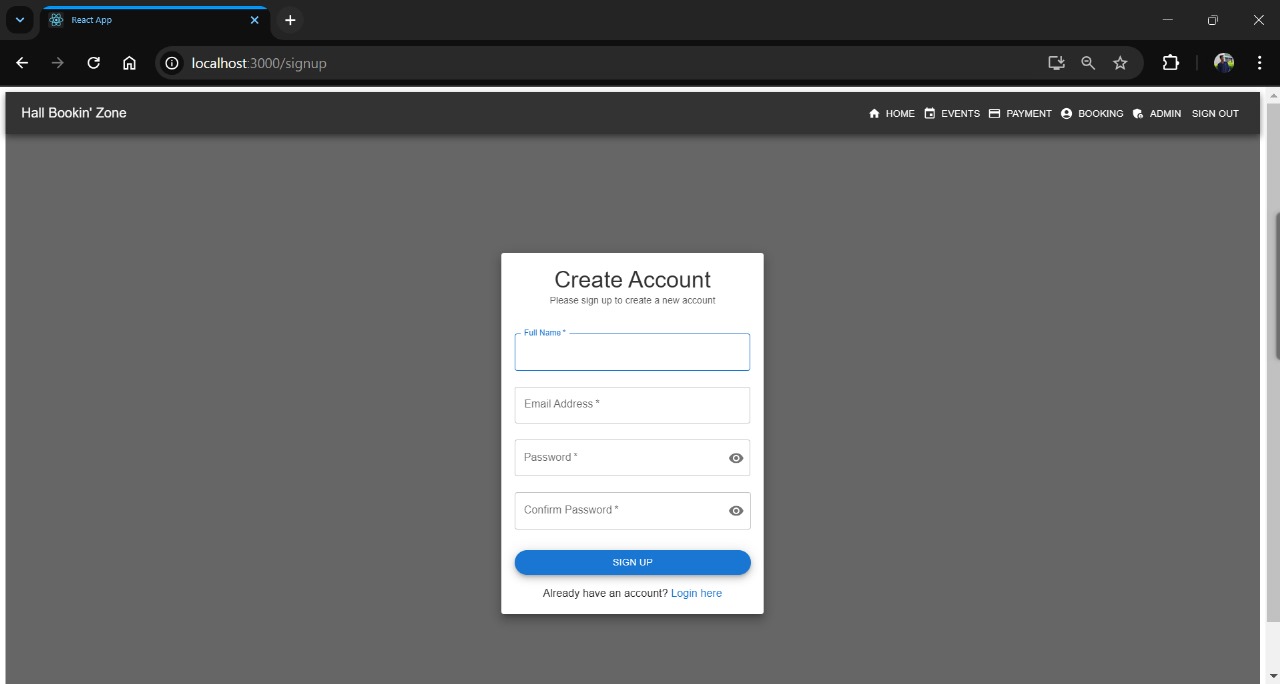


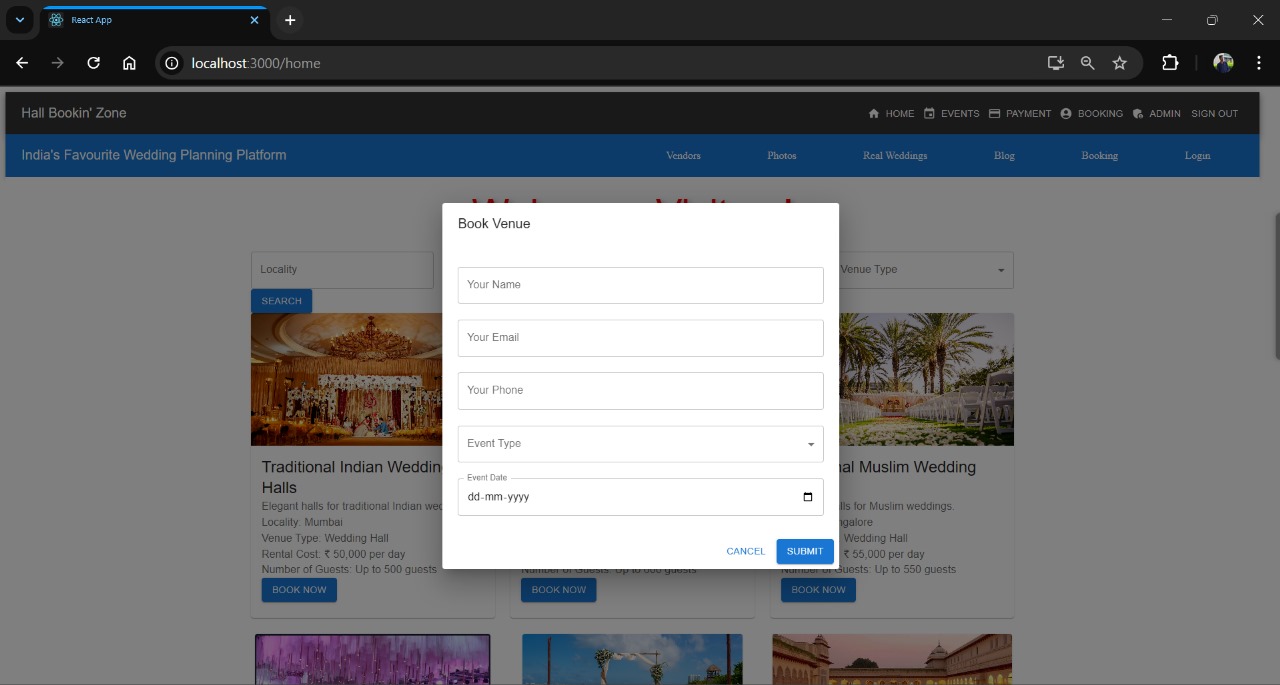


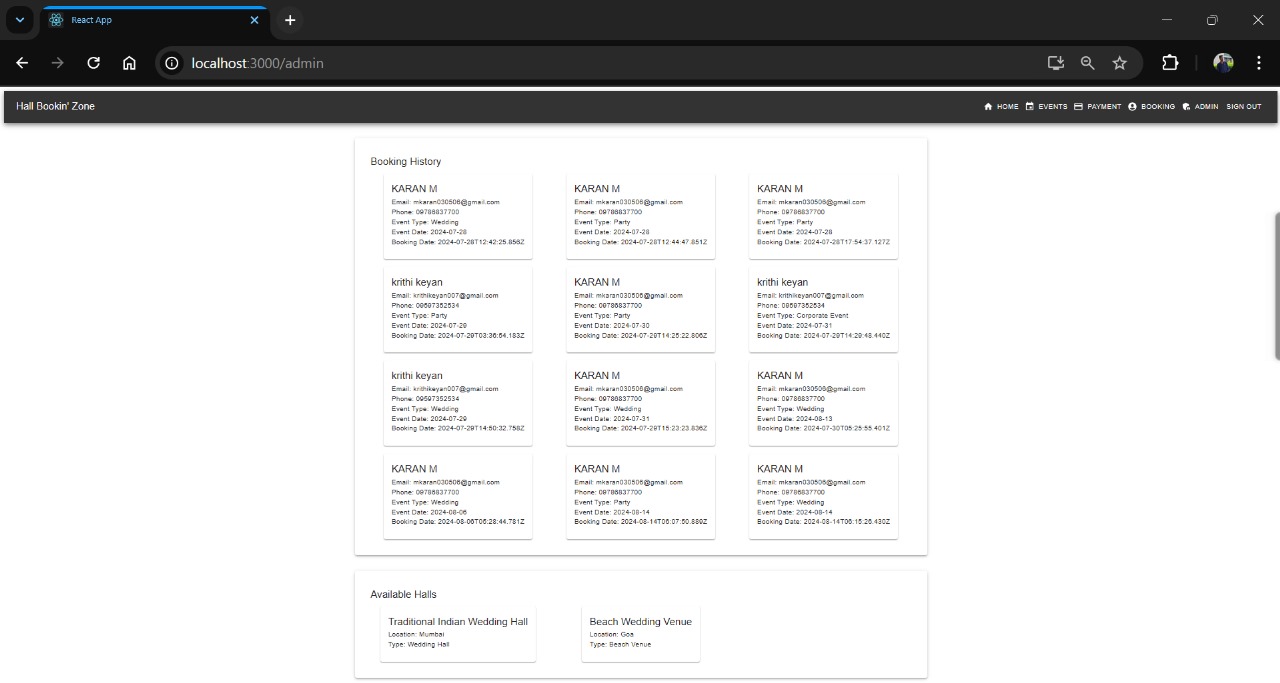
**OUTPUT :**

****

****



****

****