EVENT BOOKING SYSTEM WITH TICKETING AND SCHEDULING

A PROJECT REPORT

Bachelor of Technology
in
COMPUTER ENGINEERING

Major Project I (01CE0716)
Submitted by

MANISH KUMAR
92201703116
VINAY KUMAR
92201703097
PAWAN KUMAR YADAV
92201703113

Faculty of Engineering & Technology
Marwadi University, Rajkot

August, 2025

Major Project I (01CE0716)

Department of Computer Engineering

Faculty of Engineering & Technology

Marwadi University

A.Y. 2025-26

CERTIFICATE

This is to certify that the project report submitted along with the project entitled **Event Booking System With Ticketing and Scheduling** has been carried out by **Manish Kumar** (92201703116.),**Vinay Kumar**(92201703097), **Pawan Kumar Yadav** (92201703113) under Assistant Professor Sahir Suma guidance in partial fulfilment for the degree of Bachelor of Technology in Computer Engineering, 7th Semester of Marwadi University, Rajkot during the academic year 2025-26.

Sign	Sign
Sahir Suma	Dr. Krunal Vaghela
AssistantProfessor	Professor & Head
Department of Computer Engineering	Department of Computer Engineering

Major Project I (01CE0716)

Department of Computer Engineering

Faculty of Engineering & Technology

Marwadi University

A.Y. 2025-26

CERTIFICATE

This is to certify that the project report submitted along with the project entitled **Event Booking System With Ticketing and Scheduling** has been carried out by **Manish Kumar** (92201703116.) under Assistant Professor Sahir Suma guidance in partial fulfilment for the degree of Bachelor of Technology in Computer Engineering, 7th Semester of Marwadi University, Rajkot during the academic year 2025-26.

Sign	Sign
Sahir Suma	Dr. Krunal Vaghela
AssistantProfessor	Professor & Head
Department of Computer Engineering	Department of Computer Engineering

Major Project I (01CE0716)

Department of Computer Engineering

Faculty of Engineering & Technology

Marwadi University

A.Y. 2025-26

CERTIFICATE

This is to certify that the project report submitted along with the project entitled**Event Booking System With Ticketing and Scheduling** has been carried out by **Vinay Kumar**(92201703097) under Assistant Professor Sahir Suma guidance in partial fulfilment for the degree of Bachelor of Technology in Computer Engineering, 7th Semester of Marwadi University, Rajkot during the academic year 2025-26.

Sign	Sign
Sahir Suma	Dr. Krunal Vaghela
AssistantProfessor	Professor & Head
Department of Computer Engineering	Department of Computer Engineering

Major Project I (01CE0716)

Department of Computer Engineering

Faculty of Engineering & Technology

Marwadi University

A.Y. 2025-26

CERTIFICATE

This is to certify that the project report submitted along with the project entitled **Event Booking System With Ticketing and Scheduling** has been carried out by **Pawan Kumar Yadav** (92201703113) under Assistant Professor Sahir Suma guidance in partial fulfilment for the degree of Bachelor of Technology in Computer Engineering, 7th Semester of Marwadi University, Rajkot during the academic year 2025-26.

Sign	Sign
Sahir Suma	Dr. Krunal Vaghela
AssistantProfessor	Professor & Head
Department of Computer Engineering	Department of Computer Engineering

Major Project (01CE0716)

Department of Computer Engineering

Faculty of Engineering & Technology

Marwadi University

A.Y. 2025-26

DECLARATION

We hereby declare that the **Major Project-I (01CE0716)** report submitted along with the Project entitled **Event Booking System With Ticketing and Scheduling** submitted in partial fulfilment for the degree of Bachelor of Technology in Computer Engineering to Marwadi University, Rajkot, is a bonafide record of original project work carried out by me / us at Marwadi University under the supervision of Assistant Professor **Sahir Suma Sir** and that no part of this report has been directly copied from any students' reports or taken from any other source, without providing due reference.

S.No	Student Name	Sign
1	Manish Kumar (manishkumar.118351@marwadiuniversity.ac.in)	
2	Vinay Kumar (vinaykumar.118013@marwadiuniversity.ac.in)	
3	Pawan Kumar Yadav Pawankumar.yadav118327@marwadiuniversity.ac.in)	

Acknowledgement

We would like to express our sincere gratitude to all those who have contributed directly or indirectly to the successful completion of our Major Project **“Event Booking System with Ticketing and Scheduling”**.

First and foremost, we are deeply thankful to **Marwadi University** for providing us with the opportunity, resources, and a supportive environment to carry out this project.

We owe our special thanks to our project guide, **Mr. Sahir Suma (Assistant Professor, Department of Computer Engineering)**, for his constant support, valuable guidance, and encouragement throughout the project development. His insights, technical expertise, and continuous feedback helped us to overcome challenges and improve the quality of our work.

We also extend our heartfelt gratitude to **Prof Dr. Krunal Vaghela (Professor & Head, Department of Computer Engineering)** for his motivation and for providing the necessary facilities to successfully complete this project.

Finally, we would like to thank our **teammates** for their support, cooperation, and motivation. Without their encouragement, this project would not have been possible.

Abstract

The growing demand for efficient event management solutions has led to the development of online booking platforms that simplify the process of organizing and attending events. The project **"Event Booking System with Ticketing and Scheduling"** aims to provide a centralized and user-friendly platform for users to browse, search, and book tickets for various events, while enabling administrators to manage events, tickets, users, and bookings in a secure and efficient manner.

This system has been implemented using the **MERN stack (MongoDB, Express.js, React.js, Node.js)**, offering scalability, flexibility, and performance. It provides two primary roles: **User** and **Admin**. Users can register, log in, view available events, select seats (VIP/General), make secure payments, and check their booking history. Admins can create, edit, approve or reject events, manage users, configure tickets, and monitor bookings. The system also integrates a **payment gateway** to ensure smooth and reliable transactions.

The backend is developed with **Node.js and Express.js**, providing RESTful APIs for authentication, event management, ticketing, booking. **MongoDB** serves as the database to handle collections such as Users, Events, Tickets, and Bookings. The frontend, built using **React.js**, provides an interactive, responsive, and role-based user interface. Security features include **JWT-based authentication, role-based access control, and password encryption**.

Testing was conducted at multiple levels—unit testing, integration testing, and user acceptance testing—to ensure the system's reliability, usability, and performance. The results demonstrate that the system successfully handles event scheduling, ticket booking, seat allocation, and payment processing.

In conclusion, the developed system provides a complete solution for **event booking and scheduling with ticketing**, addressing the limitations of existing platforms and offering scope for future enhancements like refund management, mobile app integration, and advanced analytics.

List of Figures

- **Fig 3.3.1**Use Case Diagram for User Role………………………………………. .Page 10
- **Fig 3.3.2** Use Case Diagram for Admin Role……………………………………...Page 11
- **Fig 3.4.1** Activity Diagram for Ticket Booking Flow……………………………..Page 12
- **Fig 3.4.2** Activity Diagram for Event Creation Flow……………………………...Page 13
- **Fig 3.5** Entity Relationship Diagram of Event Booking System………………….Page 15
- **Fig 4.2.1** Backend Folder Structure………………………………………………..Page 18
- **Fig 4.3.1** Frontend Folder Structure………………………………………………..Page 20
- **Fig 5.6** Screenshot: Login & Registration (User/Admin)………………………….Page 26
- **Fig 5.6a** Screenshot: Homepage with Event……………………………………….Page 26
- **Fig 5.6b** Screenshot: Event Details………………………………………………...Page 27
- **Fig 5.6c** Screenshot: Booking Page ……………………………………………….Page 27
- **Fig 5.6d** Screenshot: Admin Dashboard (Manage Events)………………………..Page 28

List of Tables

- **Table 2.3** Comparative Analysis of Existing System……………………………… Page 6
- **Table 2.5** Technology Comparision…………………………………………………Page 7
- **Table 5.3.1** Authentication Test Cases…………………………………………….Page 23
- **Table 5.3.2** Event Management Testing…………………………………………...Page 24
- **Table 5.3.3** Ticket & Booking Testing ……………………………………………Page 24
- **Table A3.1** Regular Report Diary (Weekly Progress) …………………………….Page 34

	Abbreviations
API	Application Programming Interface
MERN	MongoDb Express.js React.js Node.js
JWT	JSON Web Token
ERD	Entity Relationship Diagram
DBMS	Database Management System
CRUD	Create Read Update Delete
DFD	Data Flow Diagram

Table of Contents

Acknowledgement.. .i

Abstract ii

List of Figures iii

List of Tables iv

List of Abbreviations v

Table of Contents vi

Chapter 1 Introduction.. 1

1.1 Background 1

1.2 Problem Statement 1

1.3 Objective of the Project..2

1.4 Scope of Work 2

1.5 Advantages of the System 3

1.6 Limitation of the System 3

Chapter 2 Liturature Survey.. 4

2.1 Introduction 4

2.2 .Existing Event Booking Platforms 4

2.3 Comparative Analysis of Existing System 4

2.4 The Proposed System 6

2.5 Technology Comparison 7

Chapter 3 System Analysis & Design 8

3.1 Introduction 8

3.2.Software Requirement 8

3.2.1 Functional Requirements 8

3.2.2 Non- Functional Requirements 9

3.3 Use Case Diagrams 10

3.3.1 User Role Use Case 10

3.3.2 Admin Role Use Case 11

3.4 Activity Diagram 12

3.4.1 Ticket Booking Flow 12

3.4.2 Event Creation Flow 13

3.5 Entity Relationship Diagram 14

Chapter 4 System Implementation 17

4.1 Introduction 17

4.2 Backend Implementation 18

4.2.1 Folder Structure 18

4.2.2 Authentication 19

4.2.3 Event management API 19

4.2.4 Ticket & Booking API 19

4.3 Frontend Implementation 20

4.3.1 Folder Structure 20

4.3.2 Example:Event Booking Component 21

4.4 Security Implementation 22

4.5 Deployment 22

Chapter 5 Testing & Results 23

5.1 Introduction 23

5.2 Test Environment 23

5.3 Test Cases 23

5.3.1 Authentication Testing 23

5.3.2 Event Management Testing 24

5.3.3 Ticket & Booking Testing 24

5.4 Integration Testing 25

5.5 User Acceptance Testing 25

5.6 Results & Screenshots 26

5.7 Conclusion of Testing 28

Chapter 6 Conclusion & Future 29

6.1 Conclusion 29

6.2 Achievements .... .30

6.3 Challenges Faced .... .30

6.4 Future Scope 30

6.5 Final Words 31

Appendix 32

References 33

Regular Report Diary 34

Review Cards 35

Consent Letter 36

CHAPTER 1

INTRODUCTION

1.1 Background

Event management has always been an important activity in society. From cultural festivals, music concerts, corporate conferences, to workshops and seminars – events are organized on a daily basis and require systematic planning. Traditionally, event booking and ticketing were handled manually, which often led to problems such as mismanagement, duplicate bookings, lack of transparency and difficulties in tracking attendance.

With the rise of technology and the internet, online event booking platforms like **BookMyShow, Eventbrite, Paytm Insider and Ticketmaster** have transformed the way users interact with event organizers. Users can now browse upcoming events, book tickets instantly, and receive digital confirmations, while organizers (admins) can manage ticket inventory, scheduling and bookings efficiently.

This project, **Event Booking System with Ticketing and Scheduling**, is designed using the **MERN stack (MongoDB, Express.js, React.js, Node.js)** to provide a robust, scalable and user-friendly platform. Unlike traditional systems, this platform offers seamless integration of **event scheduling, ticket booking and seat management**. It also introduces an **Admin role (instead of Owner)** who has higher privileges such as approving/rejecting events, managing users and monitoring bookings.

1.2 Problem Statement

Despite the availability of multiple event booking applications, several challenges remain:

- Lack of **customized scheduling** for small/medium-scale events.
- Limited **seat management systems** (e.g., VIP vs General).
- No role-based access control in smaller solutions.
- Manual systems leading to **errors in ticket allocation and overbooking**.
- Difficulty for admins to monitor bookings and manage event approval.
- Limited features in local event booking systems as compared to large-scale commercial solutions.
- Thus, there is a strong need for a **centralized, scalable and role-based event booking system** where users can easily book tickets and admins can efficiently manage events.

1.3 Objectives of the Project

The major objectives of the project are:

- To design and develop a **MERN-based event booking system**.
- To provide **role-based access** with two roles:
    - **User**: View events, search, book tickets and check bookings.
    - **Admin**: Create/edit events & tickets, approve/reject events, manage users and view bookings.
- To implement a **secure authentication system** using JWT tokens.
- To integrate **event scheduling** features for better planning.
- To implement **seat selection (VIP & General)** while booking tickets.
- To create an intuitive **React frontend** with search, filters and responsive design.
- To provide **RESTful APIs** for authentication, event, ticket and booking management.
- To ensure **data consistency and security** using MongoDB and middleware.
- To allow **export of bookings and event statistics** for admin insights.
- To develop a scalable system that can handle multiple concurrent users.

1.4 Scope of Work

The scope of this project covers both **users and admins**.

For Users:

- Browse upcoming events.
- Search and filter events by category, date, or keyword.
- View event details including date, time, location and ticket availability.
- Book tickets with seat selection.
- View booking history.

For Admins:

- Create, edit and delete events.
- Approve or reject event requests.
- Manage ticket pricing and categories (VIP/General).
- View user bookings and event statistics.
- Manage registered users.
- Export bookings and statistics to CSV.

1.5 Advantages of the System

- **Centralized Event Management**: All event information is stored in a single platform.
- **Role-Based Access**: Ensures secure and restricted operations for users and admins.
- **Real-Time Ticketing**: Users get instant confirmation and admins see live updates.
- **Seat Management**: Avoids double bookings by marking seats as reserved.
- **Scalability**: Built on MERN, can scale horizontally to support thousands of users.
- **Automation**: Reduces manual work by automating scheduling, approval and booking.

1.6 Limitations of the System

- Currently does not include **payment gateway integration** (future scope).
- Seat layout is limited to **VIP and General categories** (can be extended).
- Requires internet connectivity; no offline mode.
- Admin panel is web-based only; mobile app version is planned for the future.

CHAPTER 2

LITERATURE SURVEY

2.1 Introduction

A literature survey is an essential step in any project, as it provides insights into existing solutions, identifies gaps, and helps to position the proposed system in the broader technological context. In the field of **event booking and ticketing systems**, several platforms exist globally, such as **BookMyShow, Eventbrite, Ticketmaster, Paytm Insider and Cvent**. These platforms provide various features like event discovery, ticket booking, payment integration and event analytics.

However, most of these systems are either:

- **Highly commercialized** and target large-scale events, making them unsuitable for smaller organizations or colleges.
- **Feature-restricted**, especially when it comes to **seat selection, scheduling or role-based access**.
- **Not open-source**, which limits customization for institutions or smaller enterprises.

This motivates the development of a **custom MERN-based solution** that addresses these gaps and is both flexible and scalable.

2.2 Existing Event Booking Platforms

(a) BookMyShow

- One of India's most popular platforms for booking movie and event tickets.
- Provides features like seat selection, digital payments, and booking history.
- Limitation: Primarily designed for **movies and large-scale concerts.**

(b) Eventbrite

- Global event management and ticketing platform.
- Offers event promotion, ticket sales and attendee management.
- Limitation: Heavily dependent on online payments and suited more for **professional events**.

(c) Ticketmaster

- International ticketing giant used for sports, concerts and theatre.
- Features include fraud detection, reserved seating and large-scale event handling.
- Limitation: Very **complex and expensive**, not feasible for smaller organizations.

(d) Paytm Insider

- Popular in India for local events, stand-up comedy shows and college fests.
- Provides quick event listing and mobile booking.
- Limitation: Lacks deep **scheduling and seat management features**.

(e) Cvent

- Enterprise-level event management software.
- Offers end-to-end event lifecycle management including marketing and reporting.
- Limitation: Designed for large-scale conferences and corporate events.

2.3 Comparative Analysis of Existing Systems

Platform	**Seat Selection**	**Scheduling**	**Role-Based Access**	**Pricing**	**Suitability**
BookMyShow	Yes	Limited	No	Paid	Movies, Concerts
Eventbrite	No	Yes	No	Paid	Professional Events
Ticketmaster	Yes	Yes	No	High	Large-scale
Paytm Insider	No	Limited	No	Paid	Local Events
Cvent	Yes	Yes	No	High	Corporate Events
Proposed System (MERN)	Yes	Yes	Yes (User/Admin)	Free (Custom)	College + Public

From the comparison above, it is evident that the proposed system fills multiple gaps:

- **Seat Management** for both VIP and General tickets.
- **Scheduling Integration** for event timing and availability.
- **Role-Based Access** (User vs Admin).
- **Customizable and scalable** as it is built using MERN stack.

2.4 The Proposed System

The review of existing platforms highlights several reasons why a new system is needed:

- **Flexibility** – Current solutions are rigid and do not allow full customization.
- **Affordability** – Most platforms charge organizers a commission or subscription.
- **Role Management** – Lack of separate roles (User vs Admin) in smaller systems.
- **Scheduling** – Limited or no support for managing event timings dynamically.
- **Institutional Usage** – Universities, colleges, and local communities need a lightweight but powerful solution.

Thus, the **Event Booking System with Ticketing and Scheduling** aims to provide an **open, flexible, role-based and scalable platform** for both users and admins.

2.5 Technology Comparison

Technology	**Strengths**	**Weaknesses**
PHP + MySQL (Traditional)	Easy to learn, widely used	Less scalable, monolithic
Java + Spring Boot	High performance, enterprise-level	Complex setup
Python + Django	Fast development, good ORM	Slower in real-time apps
MERN (MongoDB, Express, React, Node)	Scalable, full-stack JS, real-time support	Requires expertise in JS

Table-2.5

Why MERN?

- Entire stack uses JavaScript (simplifies development).
- **MongoDB** handles flexible event data (tickets, schedules, seat availability).
- **Express + Node** power backend APIs for authentication, events, tickets, bookings.
- **React** provides interactive, responsive UI with search, filters and booking flow.
- Highly suitable for modern web applications that require **real-time updates** (like seat booking).

_

CHAPTER 3

SYSTEM ANALYSIS & DESIGN

3.1 Introduction

System analysis and design is a crucial stage of the Software Development Life Cycle (SDLC). It helps in understanding the functional requirements of the project and defining the system architecture before implementation.

For the **Event Booking System with Ticketing & Scheduling**, system analysis ensures that the platform meets the needs of **two main roles**:

- **User**: Can search, view and book events.
- **Admin**: Can create/edit events, approve/reject events, manage users, tickets and bookings.

The design phase focuses on **diagrams, models, and database structures** that describe how the system operates.

3.2 Software Requirements

3.2.1 Functional Requirements

Authentication & Authorization

- Users and Admins must be able to register and log in securely.
- JWT tokens used for secure session handling.

User Module

- Search and view events.
- Book tickets with seat selection (VIP/General).
- View booking history.

Admin Module

- Create, edit, and delete events.
- Approve or reject event requests
- Manage tickets (pricing, categories).
- Manage users and bookings.
- Export bookings/statistics to CSV.

Event Management

- Store event details (title, description, date, venue, category, image).
- Scheduling support (start & end time).

Ticketing & Booking

- Support multiple categories (VIP/General).
- Prevent duplicate bookings via seat locking.
- Maintain booking history.

3.2.2 Non-Functional Requirements

- **Scalability**: System should handle thousands of concurrent users.
- **Security**: Use JWT authentication, password hashing, and role-based middleware.
- **Performance**: API response time should be < 200 ms.
- **Reliability**: Booking data must be consistent and error-free.
- **Usability**: Intuitive user interface (React, responsive design).
- **Portability**: Should run on different browsers and devices.

3.3 Use Case Diagrams

3.3.1 User Role Use Case

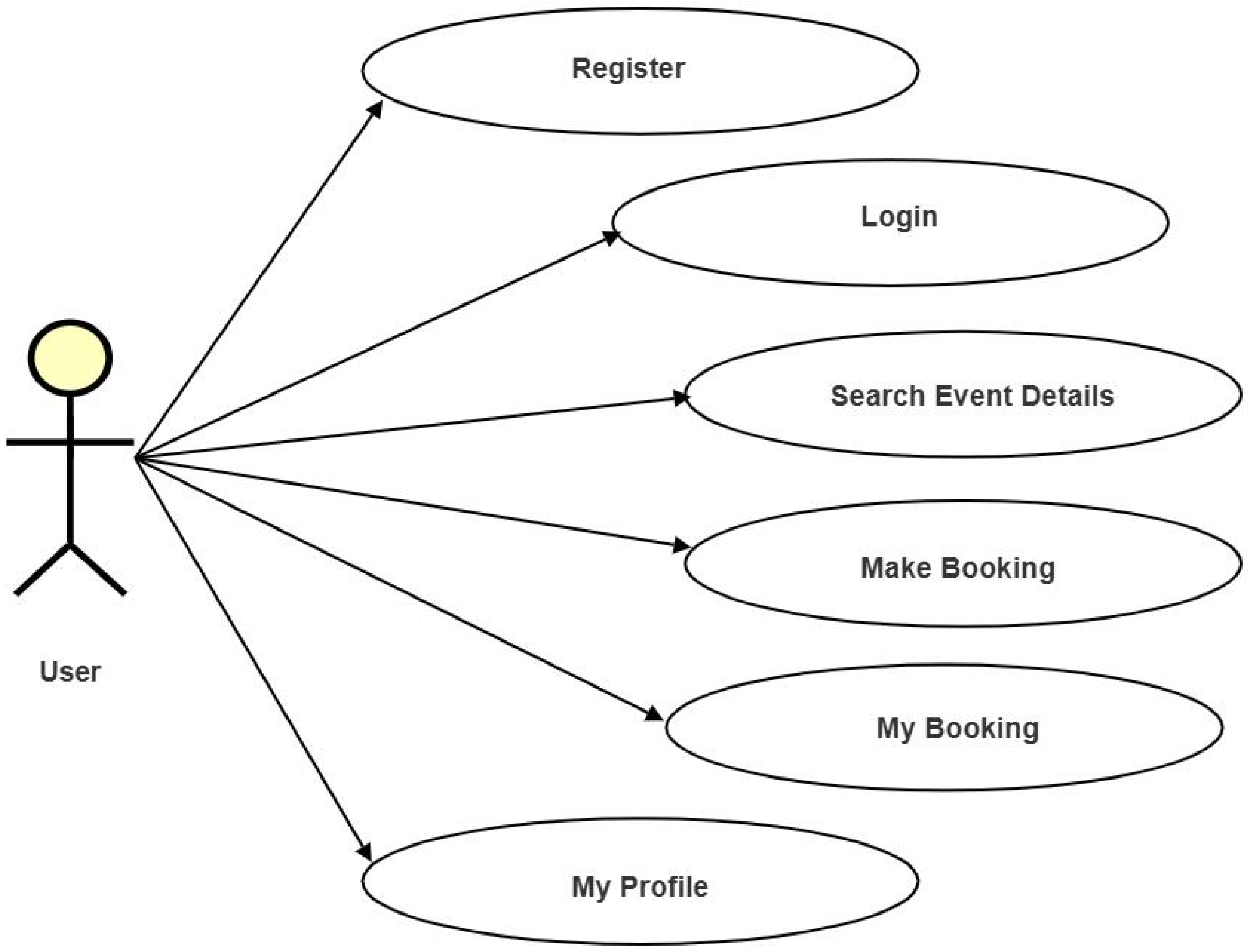
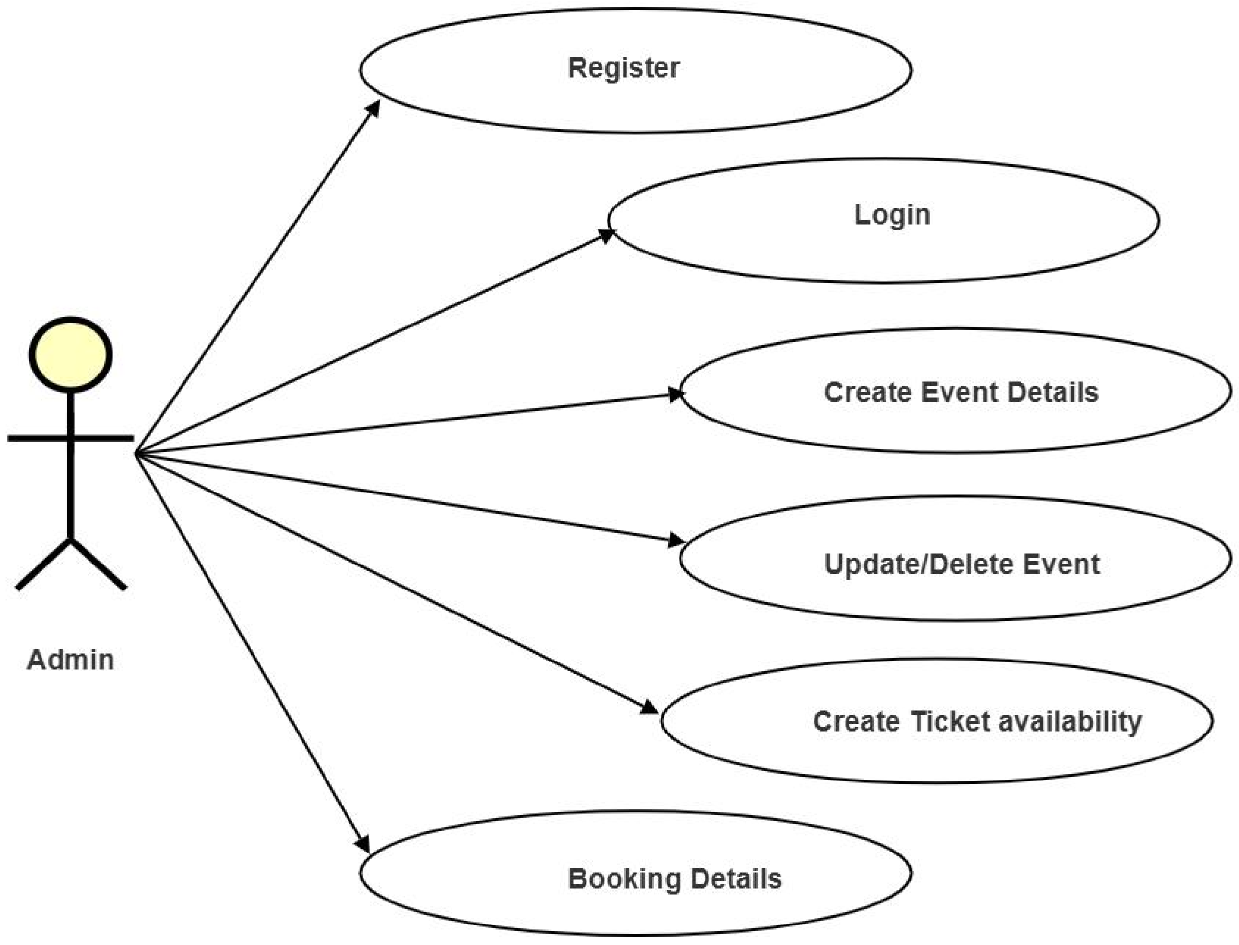


Fig-3.3.1

3.3.2 Admin Role Use Case



.

Fig-3.3.2

3.4 Activity Diagrams

3.4.1 Ticket Booking Flow

- User logs in.
- User searches for an event.
- User selects event → chooses seat → confirms ticket.
- System checks seat availability.
- If available → Booking confirmed → Record stored in DB.
- If not available → Show “Seat already booked” message.

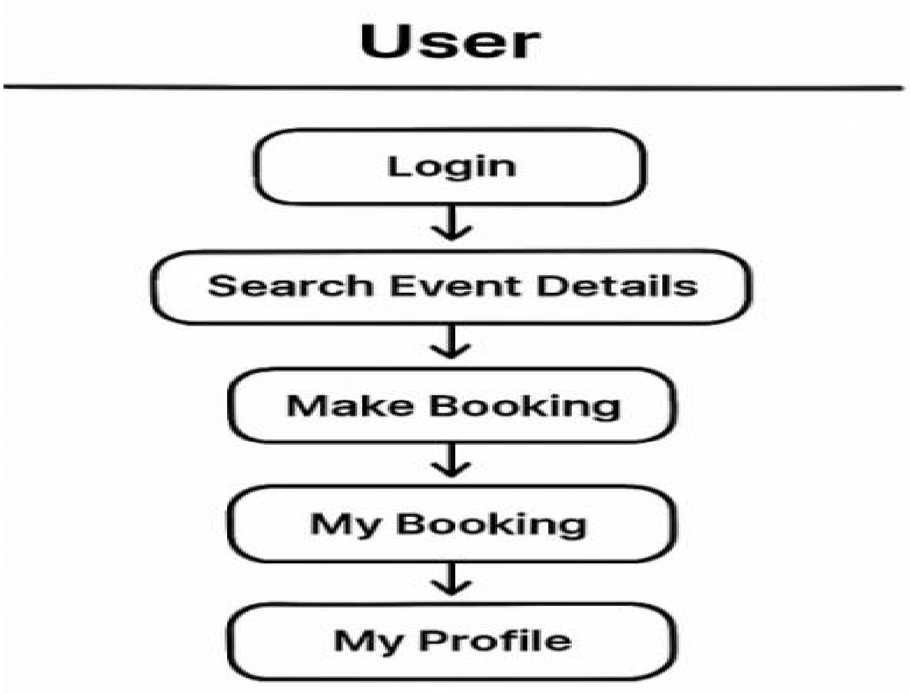


Fig-3.4.1

3.4.2 Event Creation Flow (Admin)

- Admin logs in.
- Admin creates new event (title, date, category, ticket pricing).
- System validates inputs.
- Event stored in database.
- Users can now see event on homepage.

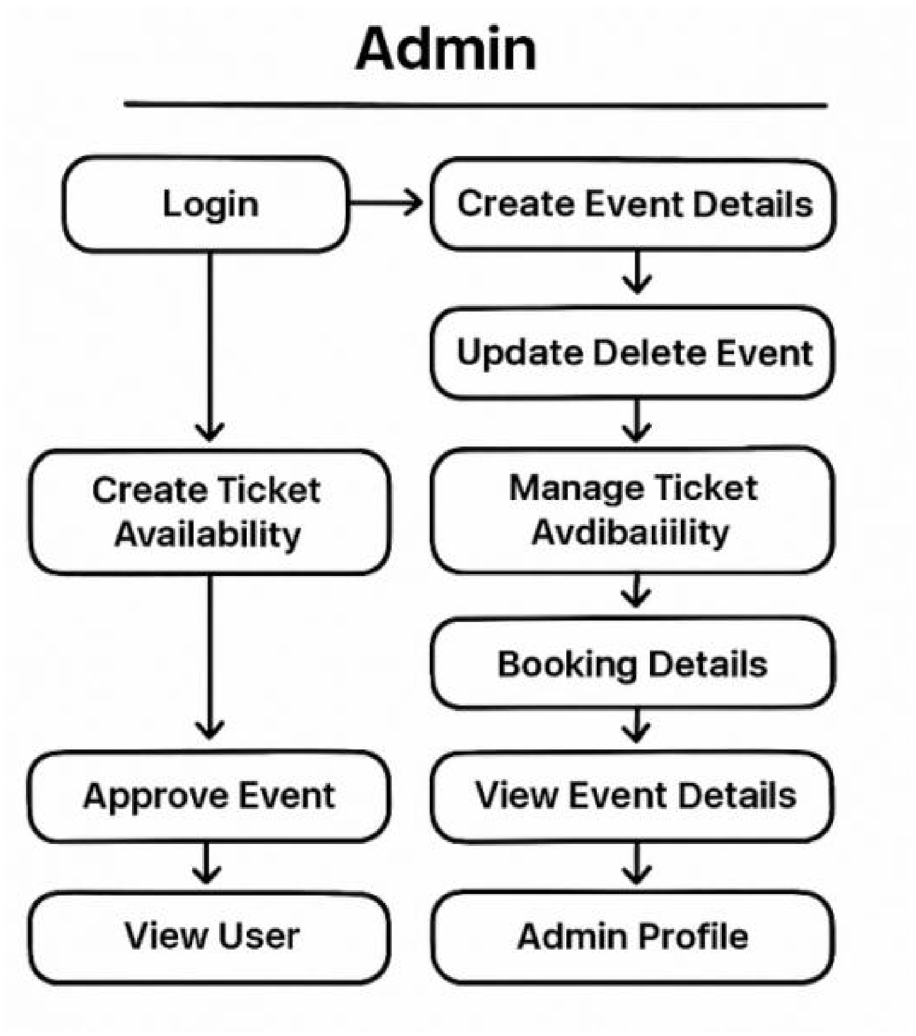


Fig-3.4.2

3.5 Entity Relationship Diagram (ERD)

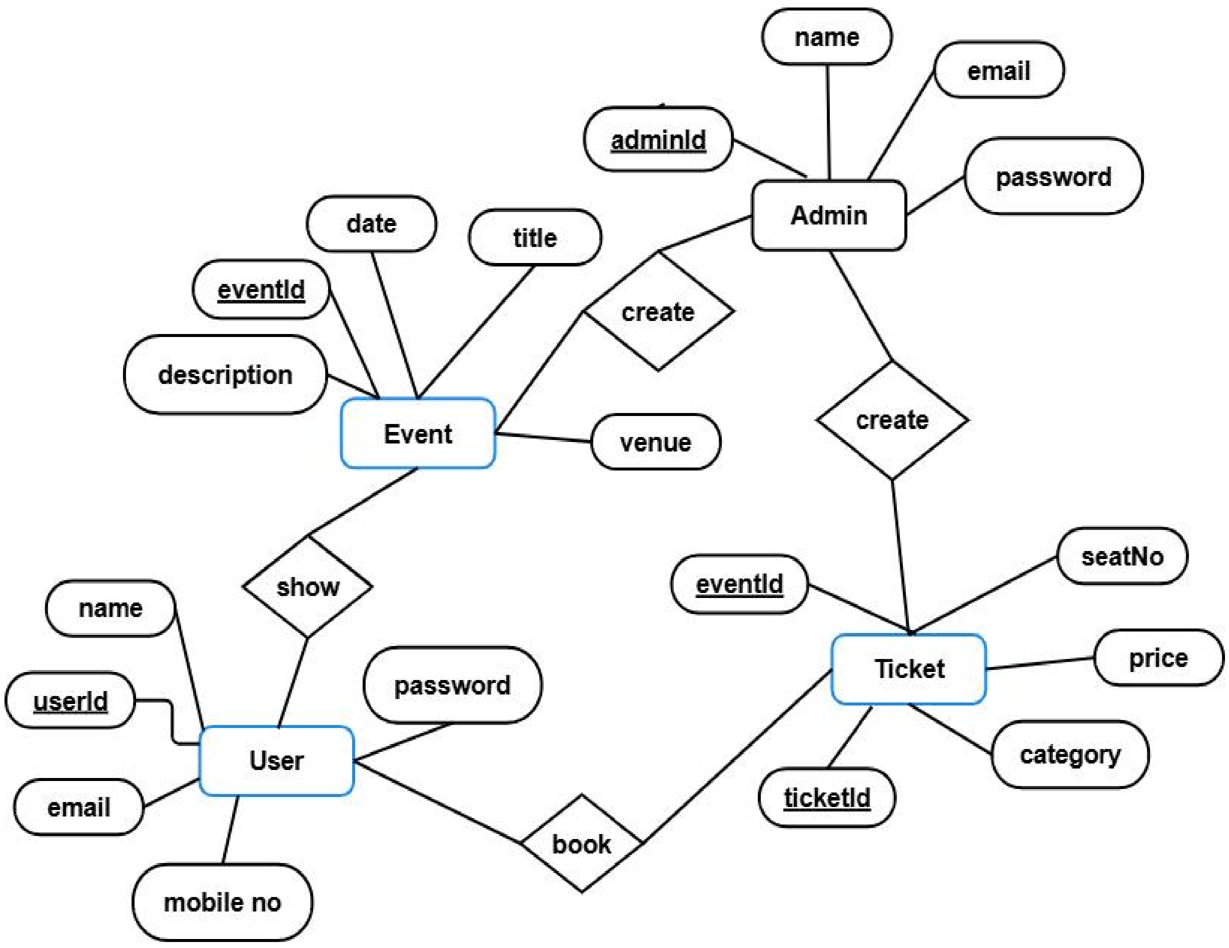
Entities:

- **User** (userId, name, email, password,Mobile no)
- **Admin** (adminId, name, email, password)
- **Event** (eventId, title, description, date, venue, createdBy)
- **Ticket** (ticketId, eventId, price, category, seatNo)

Relationships:

- User **books** Ticket.
- Ticket **belongs to** Event.
- Admin **creates/approves** Event.
- Booking **links User, Event, and Ticket**.

.



ER Diagram

Fig-3.5

3.6 Database Schema (MongoDB Models)

User Model

```
{
  _id: ObjectId,
  name: String,
  email: String,
  password: String (hashed),
  role: { type: String, enum: ["user", "admin"] }
}
```

Event Model

```
{
  _id: ObjectId,
  title: String,
  description: String,
  category: String,
  date: Date,
  venue: String,
  image: String,
  status: { type: String, enum: ["pending", "approved", "rejected"] },
  createdBy: ObjectId (Admin)
}
```

Ticket Model

```
{
  _id: ObjectId,
  eventId: ObjectId,
  price: Number,
  category: { type: String, enum: ["VIP", "General"] },
  seatNo: String,
  status: { type: String, enum: ["available", "booked"] }
}
```

Booking Model

```
{
  _id: ObjectId,
  userId: ObjectId,
  eventId: ObjectId,
  ticketId: ObjectId,
  bookingDate: Date,
  status: { type: String, enum: ["confirmed", "cancelled"] }}
```

CHAPTER 4

SYSTEM IMPLEMENTATION

4.1 Introduction

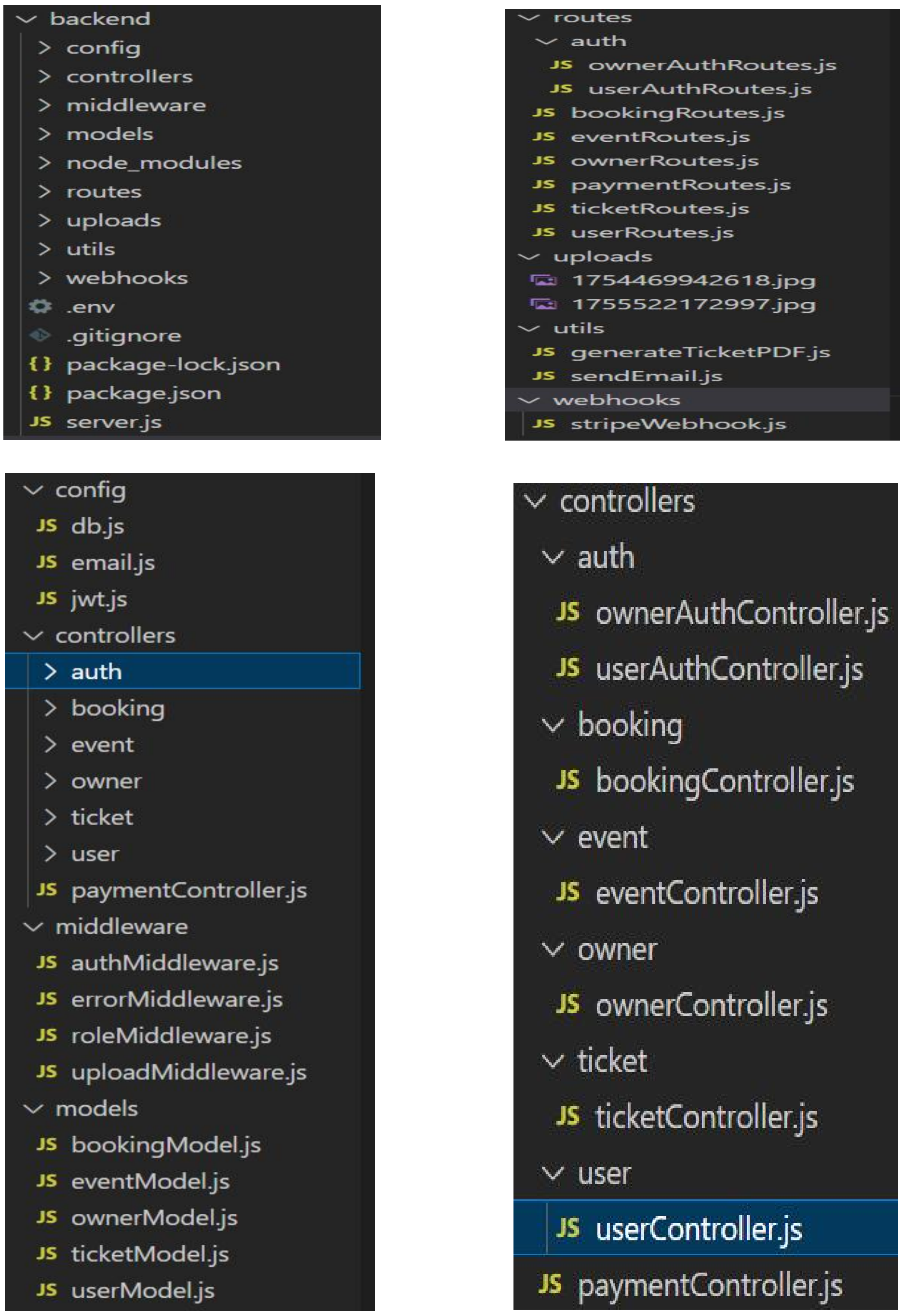
System implementation is the process of translating the design into a working software solution.
For this project, we have used the **MERN stack** (MongoDB, Express.js, React.js, Node.js), which allows full-stack development using **JavaScript**.

The implementation covers:

- **Backend** → REST APIs, authentication, event/ticket/booking management.
- **Frontend** → React-based UI with role-specific features.
- **Database** → MongoDB for storing users, events, tickets, bookings.
- **Security** → JWT authentication, bcrypt password hashing, middleware.

4.2 Backend Implementation

4.2.1 Folder Structure



```
v backend
  > config
  > controllers
  > middleware
  > models
  > node_modules
  > routes
  > uploads
  > utils
  > webhooks
  .env
  .gitignore
  package-lock.json
  package.json
  server.js
```

```
v routes
  v auth
    ownerAuthRoutes.js
    userAuthRoutes.js
  bookingRoutes.js
  eventRoutes.js
  ownerRoutes.js
  paymentRoutes.js
  ticketRoutes.js
  userRoutes.js
v uploads
  1754469942618.jpg
  1755522172997.jpg
v utils
  generateTicketPDF.js
  sendEmail.js
v webhooks
  stripeWebhook.js
```

```
v config
  db.js
  email.js
  jwt.js
v controllers
  > auth
  > booking
  > event
  > owner
  > ticket
  > user
  paymentController.js
v middleware
  authMiddleware.js
  errorMiddleware.js
  roleMiddleware.js
  uploadMiddleware.js
v models
  bookingModel.js
  eventModel.js
  ownerModel.js
  ticketModel.js
  userModel.js
```

```
v controllers
  v auth
    ownerAuthController.js
    userAuthController.js
  v booking
    bookingController.js
  v event
    eventController.js
  v owner
    ownerController.js
  v ticket
    ticketController.js
  v user
    userController.js
  paymentController.js
```

Fig-4.2.1

4.2.2 Authentication (JWT)

- Users/Admins register with **name, email, password**.
- Passwords stored using **bcrypt hashing**.
- Login generates **JWT token** with role info.
- Middleware checks if user is authorized.

```
// utils/generateToken.jsimport jwt from "jsonwebtoken";
const generateToken = (id, role) => {
return jwt.sign({ id, role }, process.env.JWT_SECRET, {
  expiresIn: "7d",
});
};export default generateToken;
```

4.2.3 Event Management API (eventRoutes.js)

```
import express from "express";import { createEvent, getEvents, getEventById, updateEvent } from "../controllers/eventController.js";import { protect, adminOnly } from "../middleware/authMiddleware.js";
const router = express.Router();
router.post("/", protect, adminOnly, createEvent);   // Admin creates event
router.get("/", getEvents);                          // Users view all events
router.get("/:id", getEventById);                    // View event details
router.put("/:id", protect, adminOnly, updateEvent); // Admin updates
export default router;
```

4.2.4 Ticket & Booking API

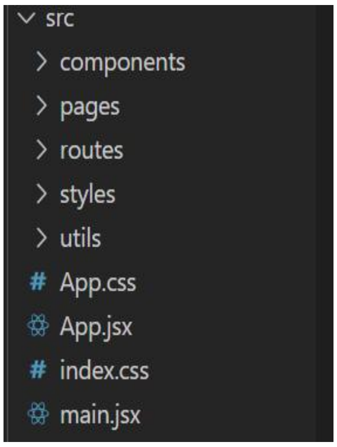
```
// ticketRoutes.jsimport express from "express";import { createTicket, getTicketsByEvent } from "../controllers/ticketController.js";import { protect, adminOnly } from "../middleware/authMiddleware.js";
const router = express.Router();
router.post("/create", protect, adminOnly, createTicket);
router.get("/event/:eventId", getTicketsByEvent);
export default router;
// bookingRoutes.jsimport express from "express";import { bookTicket, myBookings } from "../controllers/bookingController.js";import { protect } from "../middleware/authMiddleware.js";
const router = express.Router();
router.post("/book", protect, bookTicket);
router.get("/mybookings", protect, myBookings);
export default router;
```

4.3 Frontend Implementation

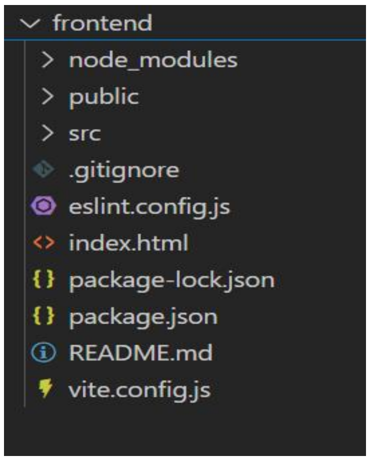
The frontend is built with **React.js + Vite**, using React Router for navigation and Axios for API calls.

4.3.1 Folder Structure

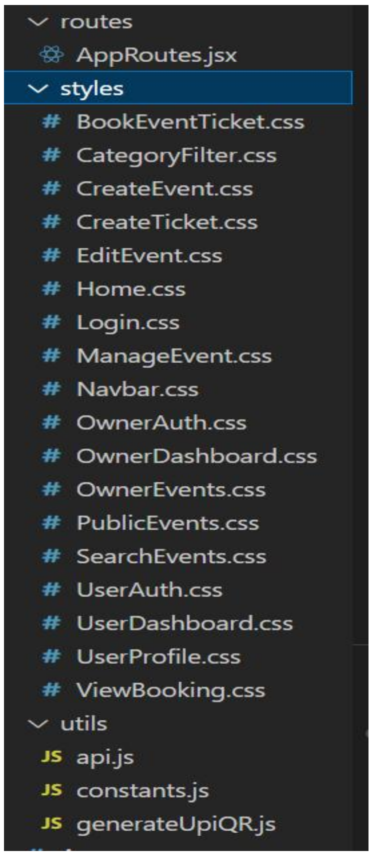
frontend



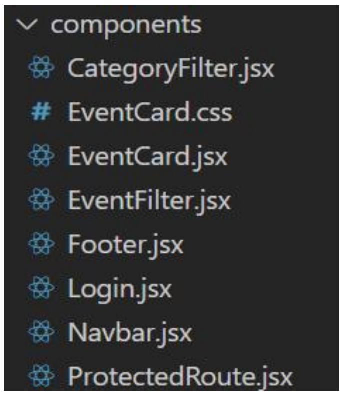
```
src
  > components
  > pages
  > routes
  > styles
  > utils
  App.css
  App.jsx
  index.css
  main.jsx
```



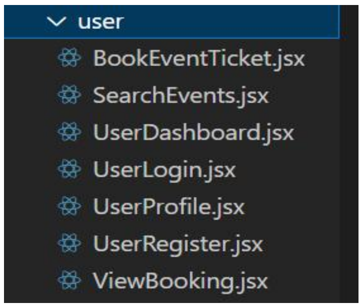
```
frontend
  > node_modules
  > public
  > src
  .gitignore
  eslint.config.js
  index.html
  package-lock.json
  package.json
  README.md
  vite.config.js
```



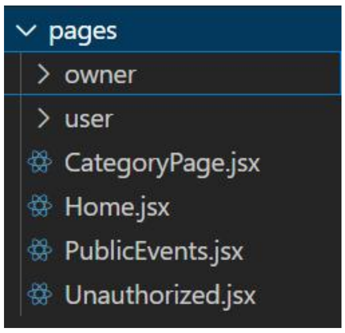
```
routes
  AppRoutes.jsx
styles
  BookEventTicket.css
  CategoryFilter.css
  CreateEvent.css
  CreateTicket.css
  EditEvent.css
  Home.css
  Login.css
  ManageEvent.css
  Navbar.css
  OwnerAuth.css
  OwnerDashboard.css
  OwnerEvents.css
  PublicEvents.css
  SearchEvents.css
  UserAuth.css
  UserDashboard.css
  UserProfile.css
  ViewBooking.css
utils
  api.js
  constants.js
  generateUpiQR.js
```



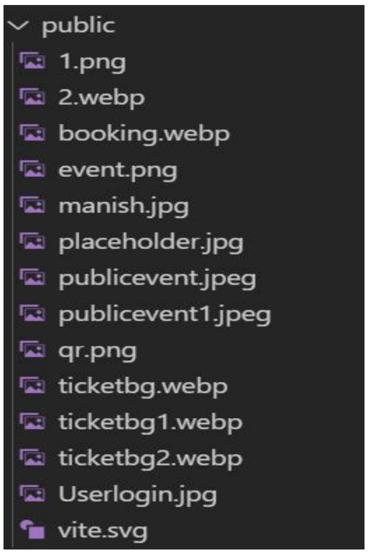
```
components
  CategoryFilter.jsx
  EventCard.css
  EventCard.jsx
  EventFilter.jsx
  Footer.jsx
  Login.jsx
  Navbar.jsx
  ProtectedRoute.jsx
```



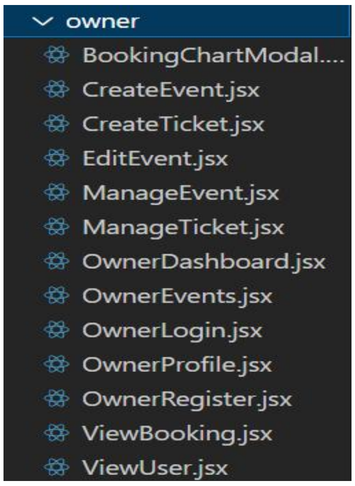
```
user
  BookEventTicket.jsx
  SearchEvents.jsx
  UserDashboard.jsx
  UserLogin.jsx
  UserProfile.jsx
  UserRegister.jsx
  ViewBooking.jsx
```



```
pages
  > owner
  > user
  CategoryPage.jsx
  Home.jsx
  PublicEvents.jsx
  Unauthorized.jsx
```



```
public
  1.png
  2.webp
  booking.webp
  event.png
  manish.jpg
  placeholder.jpg
  publicevent.jpeg
  publicevent1.jpeg
  qr.png
  ticketbg.webp
  ticketbg1.webp
  ticketbg2.webp
  Userlogin.jpg
  vite.svg
```



```
owner
  BookingChartModal....
  CreateEvent.jsx
  CreateTicket.jsx
  EditEvent.jsx
  ManageEvent.jsx
  ManageTicket.jsx
  OwnerDashboard.jsx
  OwnerEvents.jsx
  OwnerLogin.jsx
  OwnerProfile.jsx
  OwnerRegister.jsx
  ViewBooking.jsx
  ViewUser.jsx
```

Fig-4.3.1

4.3.2 Example: Event Booking Component

```
// BookEventTicket.jsximport { useEffect, useState } from "react";import api from "@/utils/api";import { useNavigate } from "react-router-dom";
export default function BookEventTicket() {
 const [events, setEvents] = useState([]);
 const [selectedEvent, setSelectedEvent] = useState(null);
 const [seats, setSeats] = useState([]);
 useEffect(() => {
  api.get("/events").then((res) => setEvents(res.data));
 }, []);
 const handleBooking = async (eventId, seatId) => {
  await api.post("/bookings/book", { eventId, seatId });
  alert("Booking Confirmed!");
 };
 return (
  <div>
   <h1>Book Event Ticket</h1>
   {events.map((event) => (
    <div key={event._id}>
     <h2>{event.title}</h2>
     <button onClick={() => setSelectedEvent(event)}>View Seats</button>
    </div>
   ))}
   {selectedEvent && (
    <div>
     <h3>Seats for {selectedEvent.title}</h3>
     {seats.map((seat) => (
      <button
       key={seat._id}
       disabled={seat.status === "booked"}
       onClick={() => handleBooking(selectedEvent._id, seat._id)}
      >
       {seat.seatNo} ({seat.category})
      </button>
     ))}
    </div>
   )}
  </div>
 );}
```

4.4 Security Implementation

- **Password Hashing** → bcrypt
- **JWT Tokens** → verify identity & role
- **Middleware** → protect (auth) and adminOnly (role-based access)

```
// middleware/authMiddleware.jsimport jwt from "jsonwebtoken";import User from
"../models/userModel.js";
const protect = async (req, res, next) => {
 let token;
 if (req.headers.authorization?.startsWith("Bearer")) {
   token = req.headers.authorization.split(" ")[1];
   try {
     const decoded = jwt.verify(token, process.env.JWT_SECRET);
     req.user = await User.findById(decoded.id).select("-password");
     next();
   } catch (error) {
     res.status(401).json({ message: "Not authorized" });
   }
 }
};
```

4.5 Deployment

- **Backend** → Node.js server hosted on Heroku/Render.
- **Frontend** → React app hosted on Netlify/Vercel.
- **Database** → MongoDB Atlas (cloud).

CHAPTER 5

TESTING & RESULTS

5.1 Introduction

Software testing is an essential phase of the **Software Development Life Cycle (SDLC)** to ensure that the system is reliable, error-free, and meets user requirements. The testing strategy for the **Event Booking System with Ticketing & Scheduling** involved a mix of:

- **Unit Testing** → testing individual functions and APIs.
- **Integration Testing** → testing interaction between modules (e.g., booking + ticket).
- **System Testing** → complete end-to-end validation of features.
- **User Acceptance Testing (UAT)** → ensuring the system fulfills user/admin needs.

5.2 Test Environment

- **Backend**: Node.js, Express, MongoDB Atlas
- **Frontend**: React.js (Vite), Axios for API calls
- **Database**: MongoDB Cloud
- **Testing Tools**: Postman (API), Jest (Unit Testing), Browser Testing (UI).

5.3 Test Cases

5.3.1 Authentication Testing

Test ID	**Module**	**Input**	**Expected Output**	**Result**
TC-01	User Register	New user details	User created, token generated	Pass
TC-02	User Login	Correct email + password	Login success, JWT returned	Pass
TC-03	User Login	Wrong password	Error message: Invalid credentials	Pass
TC-04	Admin Login	Admin credentials	Admin login success	Pass

Table-5.3.1

5.3.2 Event Management Testing

Test ID	Module	Input	Expected Output	Result
TC-05	Create Event	Event details (Admin)	Event saved in DB	Pass
TC-06	Approve Event	Event ID	Status updated to "approved"	Pass
TC-07	Edit Event	Update title/date	Event updated successfully	Pass
TC-08	Delete Event	Event ID	Event removed from DB	Pass

Table-5.3.2

5.3.3 Ticket & Booking Testing

Test ID	Module	Input	Expected Output	Result
TC-09	Create Ticket	Admin adds ticket for event	Ticket linked to event in DB	Pass
TC-10	Book Ticket	User selects seat	Booking confirmed, seat status=booked	Pass
TC-11	Double Booking	User selects already booked seat	Error message: Seat not available	Pass
TC-12	My Bookings	User checks history	Display list of past bookings	Pass

Table-5.3.3

5.4 Integration Testing

- **Case 1**: User registers → logs in → browses events → books ticket → booking saved in DB. Passed.
- **Case 2**: Admin creates event → users see event on homepage → tickets linked → users can book. Passed.
- **Case 3**: Unauthorized user tries to create event → system rejects. Passed.

5.5 User Acceptance Testing (UAT)

Feedback from **test users**:

- Interface was **easy to use**.
- Booking confirmation flow was **clear and instant**.
- Admin panel allowed full **control of events and users**.
- Suggested **payment gateway integration** for real-world use.
- Mobile responsiveness could be further improved.

5.6 Results & Screenshots

In the actual report, you should insert screenshots such as:

- **Login/Register page** (User/Admin).

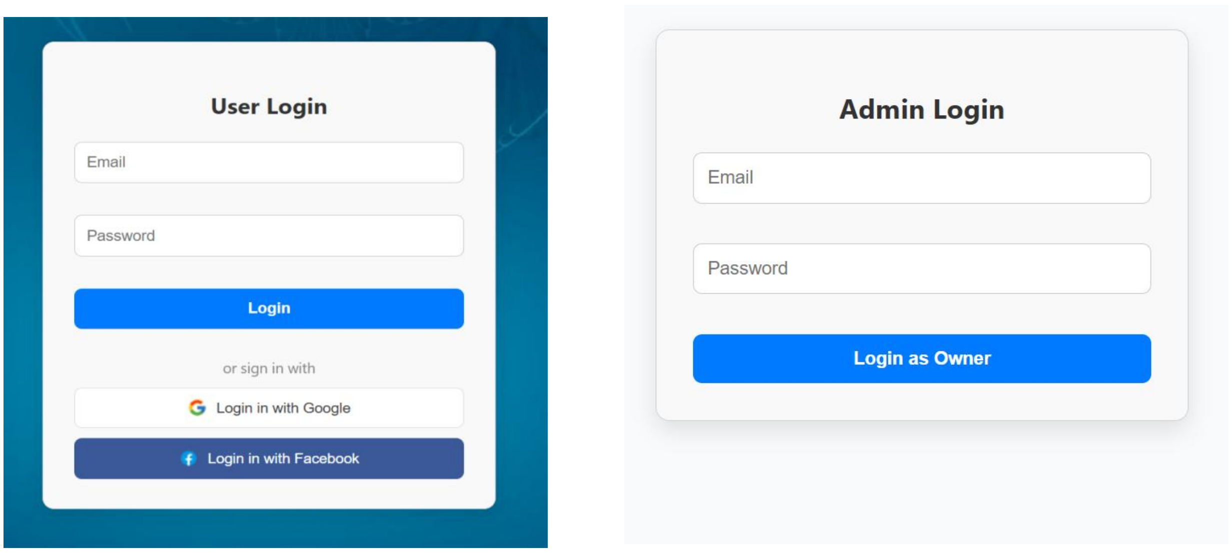


Fig-5.6

- **Homepage with events list**.

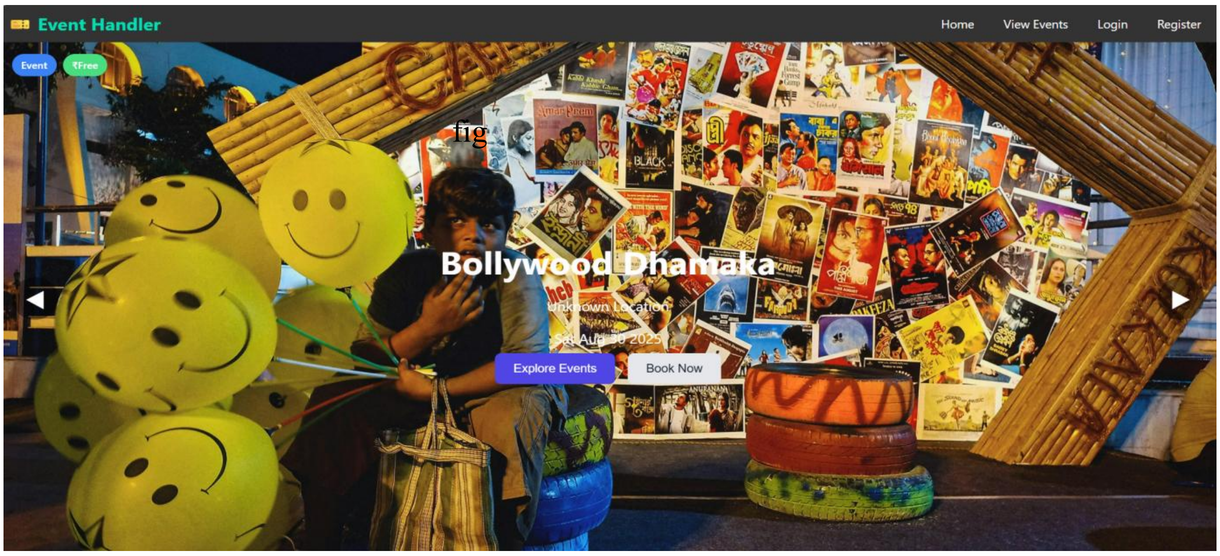


Fig-5.6a

- **Event details**

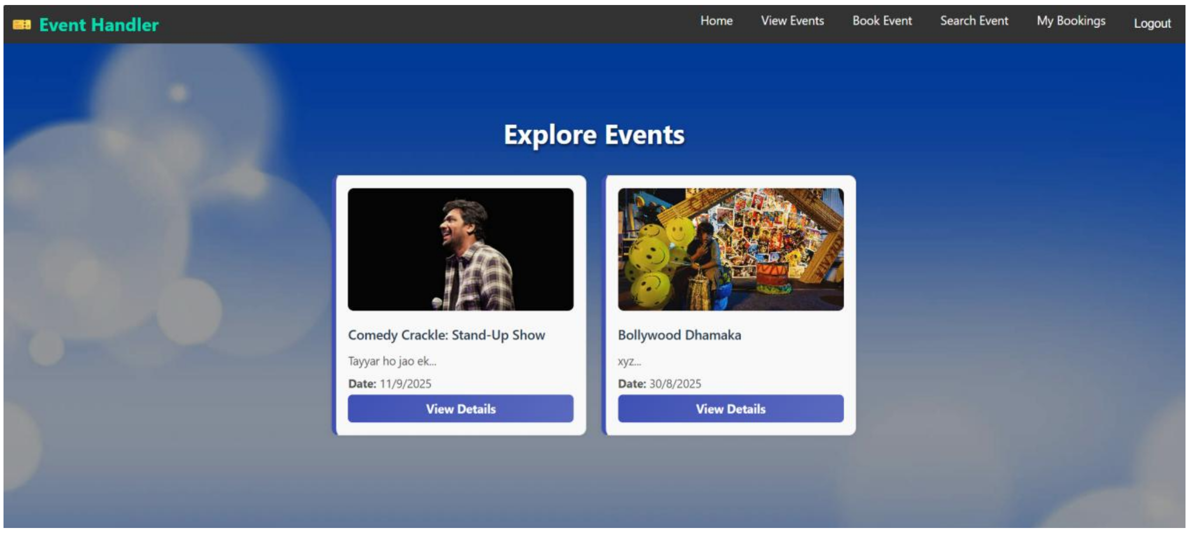


Fig-5.6b

- **Booking page.**

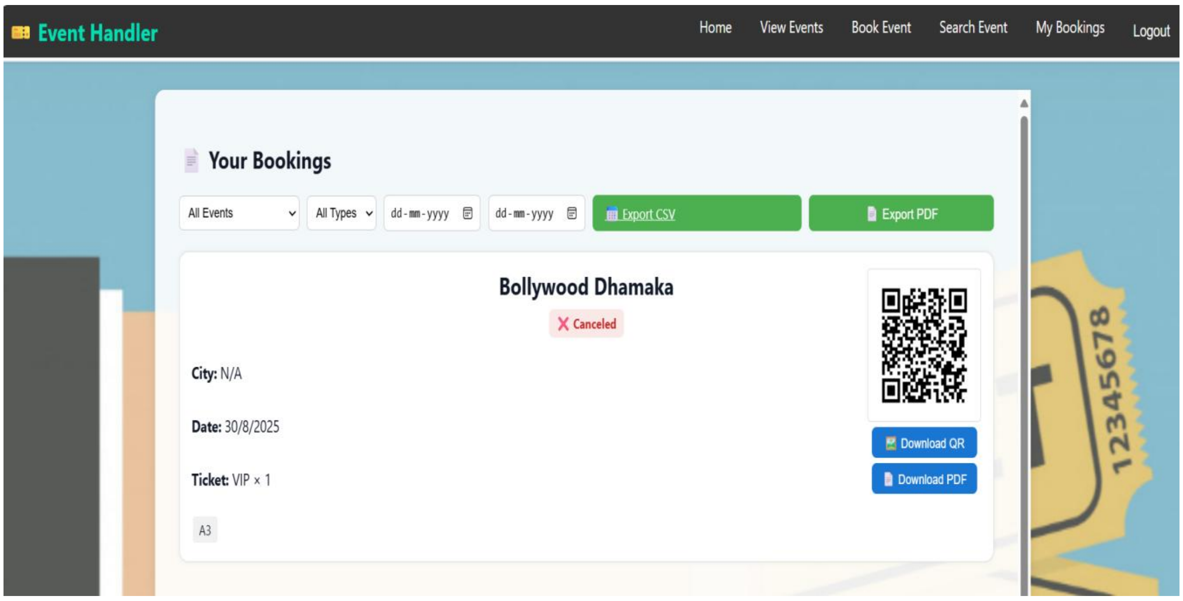


Fig-5.6c

- **Admin Dashboard** (event approval, booking stats).

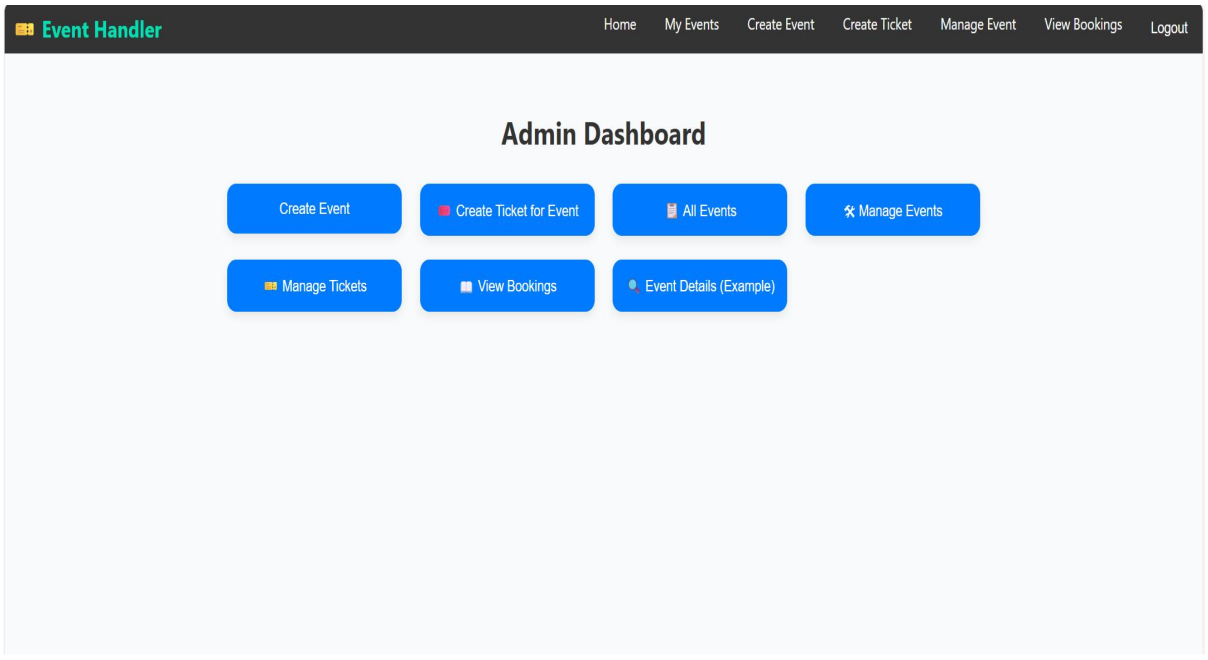


Fig-5.6d

5.7 Conclusion of Testing

- All **functional requirements** were met successfully.
- System is **stable, secure, and usable** for both Users and Admins.
- Minor improvements identified (payment gateway, advanced seat layout).
- The system is ready for deployment and real-world usage in college and community events.

CHAPTER 6

CONCLUSION & FUTURE SCOPE

6.1 Conclusion

The project **"Event Booking System with Ticketing and Scheduling"** was successfully developed using the **MERN stack (MongoDB, Express.js, React.js, Node.js)**. The system provides a **role-based platform** for managing events, tickets, and bookings with two roles:

- **User** → can browse events, search, book tickets, select seats (VIP/General) and view booking history.
- **Admin** → can create/edit events, approve/reject events, manage tickets, users and monitor bookings with analytics.

Key outcomes of the project include:

- Secure **JWT-based authentication & authorization**.
- Centralized **event management** with scheduling features.
- **Seat booking system** to prevent duplicate reservations.
- Responsive **React.js frontend** with search, filters and dynamic event views.
- **REST APIs** for modular backend communication.
- Successful **testing & validation** proving system reliability.

This project not only addresses the limitations of existing platforms but also demonstrates the scalability of MERN applications for real-world event management.

6.2 Achievements

- Designed and implemented a **full-stack MERN web application**.
- Developed **role-based access** with Admin/User distinction.
- Implemented **ticketing and seat selection** (VIP & General).
- Provided **event approval workflow** for admins.
- Integrated **search, filters and booking history** for users.
- Conducted **system testing and validation** to ensure quality

6.3 Challenges Faced

- Understanding the **seat booking logic** to prevent double booking.
- Managing **role-based routes** in both frontend and backend.
- Handling **async API calls** with React and Node.
- Deployment issues (MongoDB Atlas & frontend hosting).

Despite these challenges, the system was implemented successfully with stable performance.

6.4 Future Scope

The current system provides a strong foundation for event management. However, several **enhancements** can be implemented in future versions:

1. **Payment Gateway Integration**
    - Support for UPI, PayPal, Credit/Debit cards.
    - Automated refund system for cancellations.
2. **Mobile Application (React Native / Flutter)**
    - Dedicated Android & iOS apps for better accessibility.
3. **Notification System**
    - Email/SMS/Push notifications for booking confirmation & reminders.
4. **Advanced Seat Layouts**
    - Theater-style seat maps with drag-and-drop selection.
5. **Analytics Dashboard for Admin**
    - Graphical insights of revenue, most booked events and attendance trends.
6. **Multi-Language & Multi-Currency Support**
    - For global usability.
7. **AI Recommendations**
    - Suggesting events to users based on past bookings and preference

6.5 Final Words

This project has demonstrated how modern **full-stack web development** can be applied to solve real-world problems in event management. By combining the power of **MERN stack, RESTful APIs and role-based access**, the system provides a robust, scalable, and user-friendly platform.

It can be deployed at **colleges, universities, communities, and local organizations** for managing cultural fests, workshops, concerts and seminars. With further improvements like **payment integration, AI-based recommendations and mobile app support**, this system can grow into a **complete event management solution** comparable to leading platforms like BookMyShow or Eventbrite.

_Appendices

Appendix 1 – List of Abbreviations

- **API** – Application Programming Interface
- **CRUD** – Create, Read, Update, Delete
- **DBMS** – Database Management System
- **DFD** – Data Flow Diagram
- **ERD** – Entity Relationship Diagram
- **JWT** – JSON Web Token
- **MERN** – MongoDB, Express.js, React.js, Node.js

Appendix 2 – Screenshots of System

- Login & Registration (User/Admin)
- Event Listing (Homepage)
- Event Details Page
- Ticket Booking with Seat Selection
- Payment Gateway Integration Screen
- Booking Confirmation Page
- Admin Dashboard (Manage Events, Users, Reports)

References

[1] Rajesh Kumar, P., & Kumar, M. (2021). *Design and Implementation of Online Event Management System Using MERN Stack.* International Journal of Computer Applications, 174(5), 25–32.

[2] MongoDB Inc. (2025). *MongoDB Documentation.* Available at: https://www.mongodb.com/docs/

[3] Node.js Foundation. (2025). *Node.js Documentation.* Available at: https://nodejs.org/en/docs

[4] React Team. (2025). *React Official Documentation.* Available at: https://react.dev/

[5] Express.js Team. (2025). *Express Official Documentation.* Available at: https://expressjs.com/

[6] Stripe Inc. (2025). *Stripe API Reference.* Available at: https://stripe.com/docs/api

[7] Razorpay (2025). *Razorpay Payment Gateway Documentation.* Available at: https://razorpay.com/docs

[8] Sommerville, I. (2016). *Software Engineering* (10th Edition). Pearson Education.

[9] Pressman, R. S. (2019). *Software Engineering: A Practitioner's Approach* (8th Edition). McGraw Hill.

[10] IEEE Std 830-1998. *IEEE Recommended Practice for Software Requirements Specifications.* IEEE Computer Society.

Regular Report Diary (Original Copy)

Consent for Filing Patent

We, **assistant Professor Sahir Suma**, **Manish Kumar, Vinay Kumar, Pawan Kumar Yadav** hereby give our full consent and authorization for the filing of a patent/research publication application for the project titled **"Event Booking System with Ticketing and Scheduling**".

We hereby authorize Marwadi University and/or its legal representatives to file the patent/research publication application and act on our behalf regarding any matters related to this filing.

Date:
Name: Assistant Professor Sahir Suma
Signature:

Date:
Name: Manish Kumar
Signature:

Date:
Name: Vinay Kumar
Signature:

Date:
Name: Pawan Kumar Yadav
Signature: