INSTITUTE FOR ADVANCED COMPUTING AND SOFTWARE DEVELOPMENT, AKURDI, PUNE
“Event Booking / Event Management System”
PG-DAC (Project Report)

Submitted By:
Group No: ___
Roll No / Name: _____

Project Guide: _____
Centre Coordinator: _____

ABSTRACT
The Event Booking / Event Management System is a venue booking platform that enables customers to discover venues, view available time slots, create bookings, and complete payments securely through Razorpay. Venue owners can manage venues and availability schedules, while administrators can access and manage resources with elevated privileges.
The backend is built using Spring Boot and exposes REST APIs. Security is implemented using Spring Security with JWT-based authentication and role-based authorization. Persistent data is managed using JPA/Hibernate with a relational database and Flyway migrations.

ACKNOWLEDGEMENT
I would like to express my sincere gratitude to our project guide and faculty members for their guidance and continuous support throughout the project. I also thank our institute for providing the facilities and environment required to complete this work. Finally, I thank my friends and family for their encouragement and support.

Table of Contents

---

1. INTRODUCTION
The Event Booking / Event Management System is designed to simplify venue booking operations by providing a secure and structured process for:
- Venue owners to create and manage venues
- Venue owners to create and manage availability slots (time ranges)
- Customers to search venues, view available slots, and book

- Customers to pay via Razorpay and confirm bookings
- Customers/Admin to cancel bookings (with business rules)

1.1 Roles in the System
The project supports **Role-Based Access Control (RBAC)** using Spring Security:

- **Admin (ROLE_ADMIN)**
  - Full access to protected APIs
  - Can create/update/delete venues and manage bookings
  - Can override ownership checks where applicable

- **Venue Owner (ROLE_VENUE_OWNER)**
  - Create/update/delete own venues
  - Create/delete availability slots for own venues
  - View bookings for a venue (protected endpoint)

- **Customer (ROLE_CUSTOMER)**
  - Search venues (public endpoint)
  - View available slots (public endpoint)
  - Create bookings and confirm payments
  - Cancel own bookings

1.2 Purpose
The purpose of this system is to provide a reliable backend for venue discovery and booking with:
- Secure authentication and authorization (JWT + RBAC)
- Controlled slot scheduling (overlap checks and time validation)
- Safe booking lifecycle (PENDING → CONFIRMED / CANCELLED)
- Payment integration using Razorpay signature verification

1.3 Scope
The project scope includes:
- User registration and login APIs
- Venue management APIs (create/update/delete/list/search)
- Availability scheduling APIs (create/delete/list)
- Booking APIs (create/confirm-payment/cancel/list)
- Database design and migrations

1.4 Objectives
- **Efficient Venue Booking**: Minimize manual coordination by providing available slots and instant booking.
- **Secure System**: Use JWT authentication and role-based access control.
- **Data Integrity**: Prevent overlap scheduling and enforce booking rules.
- **Payment Security**: Confirm payments using Razorpay signature verification.
- **Scalable Backend**: RESTful design with service and repository layers.

---

2. SOFTWARE REQUIREMENT SPECIFICATION (SRS)

2.1 Functional Requirements
1. **User Authentication**
   - Users can register and login using email/password.
   - System generates JWT tokens and validates them on protected routes.

2. **Venue Management**
   - Venue owners/admin can create, update, delete venues.
   - Venue owners can fetch their own venues.
   - Public users can fetch/search active venues.

3. **Availability Scheduling**
   - Venue owners/admin can create availability slots for a venue they own.
   - Slots must be in the future and must not overlap with existing slots.
   - Booked slots cannot be deleted.
   - Public users can view available slots for a venue.

4. **Venue Search**
   - Public users can search venues by city and/or search term.

5. **Booking & Payment**
   - Customers/admin can create bookings for an available slot.
   - Creating a booking marks the slot as BOOKED and creates a Razorpay order.
   - Payment confirmation validates signature:
     - valid → booking CONFIRMED
     - invalid → booking CANCELLED and slot released to AVAILABLE

6. **Booking Cancellation**
   - Customer/admin can cancel a booking.
   - Only the booking owner (customer) or admin can cancel.
   - Cancelled/completed bookings cannot be cancelled again.
   - Cancelling releases the slot back to AVAILABLE.

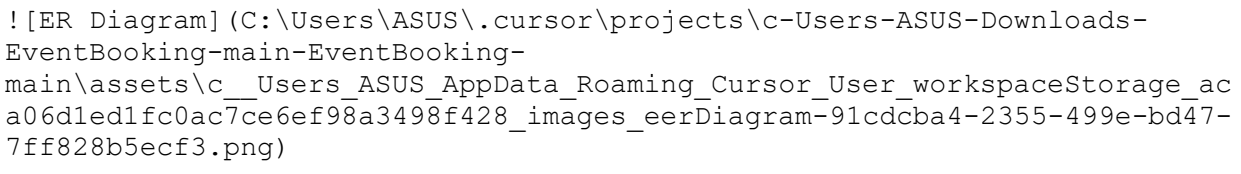
2.2 Non-Functional Requirements
- **Performance**: Standard API response time under normal load should be within acceptable limits (e.g., < 2 seconds typical).
- **Security**: JWT-based authentication, role-based authorization, and secure password hashing (BCrypt).
- **Reliability**: Booking status transitions and availability updates must be consistent (transactional operations).
- **Maintainability**: Layered architecture (Controller → Service → Repository) and DTO-based API design.
- **Scalability**: Stateless JWT auth supports horizontal scaling of backend instances.

2.3 Technology Stack
- **Backend**: Java, Spring Boot, Spring Security, JWT
- **Persistence**: Spring Data JPA / Hibernate
- **Database**: MySQL (or any compatible RDBMS), Flyway migrations
- **Payment**: Razorpay integration + HMAC signature verification

---

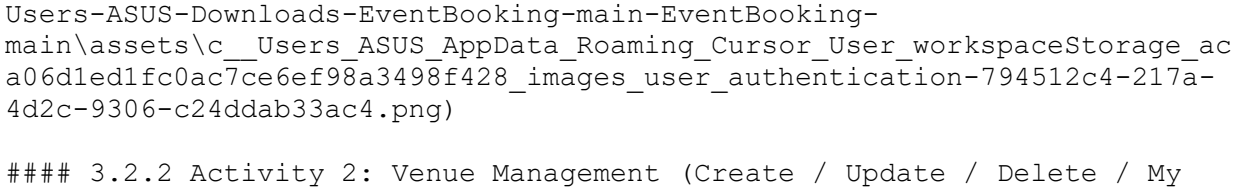
3. DIAGRAMS

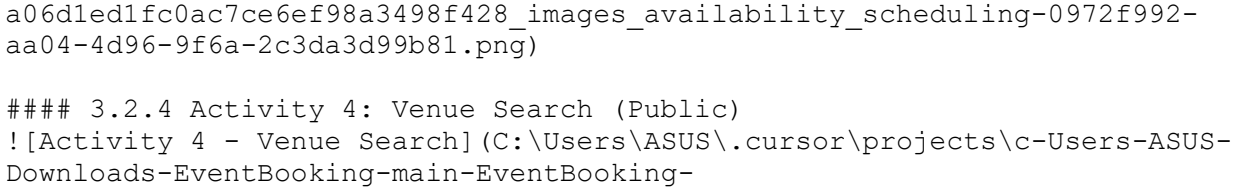
3.1 ER Diagram
Fig: ER Diagram for Event Booking System



Entities and relationships (summary):
- USER (1) owns (N) VENUE
- VENUE (1) has (N) AVAILABILITY
- USER (1) books (N) BOOKING
- VENUE (1) is booked_for (N) BOOKING
- AVAILABILITY (1) reserves (1) BOOKING

3.2 Activity Diagrams

3.2.1 Activity 1: User Authentication (Register + Login + JWT)


3.2.2 Activity 2: Venue Management (Create / Update / Delete / My Venues)


3.2.3 Activity 3: Availability Scheduling (Create / Delete / View Slots)

3.2.4 Activity 4: Venue Search (Public)

3.2.5 Activity 5: Booking & Payment (Razorpay)

![Activity 5 - Booking & Payment](C:\Users\ASUS\.cursor\projects\c-Users-ASUS-Downloads-EventBooking-main-EventBooking-main\assets\c__Users_ASUS_AppData_Roaming_Cursor_User_workspaceStorage_aca06d1ed1fc0ac7ce6ef98a3498f428_images_booking_and_payment-6550fdbf-9f45-4253-a34d-9271425c863e.png)

3.2.6 Activity 6: Booking Cancellation
![Activity 6 - Booking Cancellation](C:\Users\ASUS\.cursor\projects\c-Users-ASUS-Downloads-EventBooking-main-EventBooking-main\assets\c__Users_ASUS_AppData_Roaming_Cursor_User_workspaceStorage_aca06d1ed1fc0ac7ce6ef98a3498f428_images_booking_cancellation-d57f8903-142e-4132-b0a1-1be0ed330244.png)

---

4. DATABASE DESIGN
This section describes the database tables for the system. The following structures are referenced from the database schema and screenshots provided.

4.1 Tables

4.1.1 `users`
![users table](C:\Users\ASUS\.cursor\projects\c-Users-ASUS-Downloads-EventBooking-main-EventBooking-main\assets\c__Users_ASUS_AppData_Roaming_Cursor_User_workspaceStorage_aca06d1ed1fc0ac7ce6ef98a3498f428_images_Screenshot_2026-02-02_132940-31d0dc8b-4a5a-4ea5-89e9-88ec6b2773de.png)

4.1.2 `venues`
![venues table](C:\Users\ASUS\.cursor\projects\c-Users-ASUS-Downloads-EventBooking-main-EventBooking-main\assets\c__Users_ASUS_AppData_Roaming_Cursor_User_workspaceStorage_aca06d1ed1fc0ac7ce6ef98a3498f428_images_Screenshot_2026-02-02_132914-c5607dfb-f20e-4a3e-936d-59771b26eabb.png)

4.1.3 `availabilities`
![availabilities table](C:\Users\ASUS\.cursor\projects\c-Users-ASUS-Downloads-EventBooking-main-EventBooking-main\assets\c__Users_ASUS_AppData_Roaming_Cursor_User_workspaceStorage_aca06d1ed1fc0ac7ce6ef98a3498f428_images_Screenshot_2026-02-02_132756-7d7c328e-1d18-4cb8-b932-5009deb6ad77.png)

4.1.4 `bookings`
![bookings table](C:\Users\ASUS\.cursor\projects\c-Users-ASUS-Downloads-EventBooking-main-EventBooking-main\assets\c__Users_ASUS_AppData_Roaming_Cursor_User_workspaceStorage_aca06d1ed1fc0ac7ce6ef98a3498f428_images_Screenshot_2026-02-02_132857-013681a9-80b2-42eb-9897-372ba4d2aba3.png)

4.1.5 `flyway_schema_history`
![flyway_schema_history table](C:\Users\ASUS\.cursor\projects\c-Users-ASUS-Downloads-EventBooking-main-EventBooking-main\assets\c__Users_ASUS_AppData_Roaming_Cursor_User_workspaceStorage_ac

a06d1ed1fc0ac7ce6ef98a3498f428_images_Screenshot_2026-02-02_132838-3cde5a46-5693-4fca-b86b-115b7007be11.png)

---

5. SNAPSHOTS (Optional)
If you have UI screenshots (Home, Login, Registration, Venue list, Booking page), add them here as figures.

---

6. CONCLUSION
The Event Booking / Event Management System provides a secure and reliable backend for venue discovery, availability scheduling, and booking with integrated payment confirmation.
The system implements strong access control using JWT and role-based authorization, ensures data integrity through validation and overlap checks, and maintains consistent booking lifecycle transitions using transactional service methods.

---

7. REFERENCES
1. Spring Boot Documentation: `https://docs.spring.io/spring-boot/`
2. Spring Security Documentation: `https://docs.spring.io/spring-security/reference/`
3. Flyway Documentation: `https://documentation.red-gate.com/flyway`
4. Razorpay Docs: `https://razorpay.com/docs/`
5. JPA / Hibernate Docs: `https://hibernate.org/orm/documentation/`