(A typical Specimen of Cover Page & Title Page)

TITLE OF PROJECT

UCS 503 Software Engineering Project Report

END-Semester Evaluation

Submitted by:

(102303604)KUNSH GARG

(102303846)VAIBHAV SEHRAWAT

(102303849)SHUBHAM GOYAL

(102303851)YUVRAJ GOYAL

BE Second Year, CSE

Group No:

Submitted to:

RAGHAV VENKATARAMAIYER

Computer Science and Engineering Department

TIET, Patiala

JULY-DEC,2025

UCS503- Software Engineering Lab

TABLE OF CONTENTS

4. Implementation

4.1 Component Diagrams

4.2 Deployment Diagrams

4.3 Screenshots

5. Testing

5.1 Test Plan

5.2 Test Cases

5.3 Test Reports by Peers

1. PROJECT SELECTION PHASE

1.1 Software Bid

Project Name:

Event Ticketing & Local Marketing System for Small-Scale Events

---

Problem Statement:

Small-scale event organizers in India—such as party hosts, sports clubs, college societies, and community groups—struggle to promote, manage, and sell tickets efficiently due to:

1. Event Visibility Challenges

- Limited reach due to lack of promotional platforms
- Dependence on WhatsApp/Instagram posts which don't scale
- No centralized marketplace for small/local events

2. Manual Ticketing & Payments

- Unorganized ticket booking through UPI screenshots
- No digital verification for entries / attendees
- Difficulty managing multiple bookings

3. Lack of Real-time Information

- Users are unaware of events happening nearby
- No central place with accurate details (venue, timing, cost, map location)

4. No Data Analytics

- Organizers lack tools to track bookings, revenue, or event performance

5. Local Business Discovery Gap

- Event attendees often search for nearby:
  - Cafés
  - Restaurants
  - Hangout spots
- No platform connecting event attendees to local businesses

Proposed Solution:

A modern **web-based Event Ticketing & Local Marketing System** that provides:

✓ **Event Discovery:** Browse upcoming events with photos, description & venue details
✓ **Digital Ticketing:** Book tickets online with QR verification
✓ **Integrated Payments:** Smooth UPI/online payment workflow
✓ **Organizer Dashboard:** Create/manage events & track bookings
✓ **Recommendation Engine:** Shows nearby restaurants & places
✓ **Admin Panel:** Verify events, manage users, maintain quality
✓ **Live Analytics:** Ticket sales, revenue insights, attendee stats

---

Justification:

1. User Convenience

- Unified event marketplace for all small-scale events

2. Revenue Boost for Organizers

- Increased discoverability leads to higher bookings

3. Digital Transformation

- Eliminates manual tracking, reduces chaos at entry points

4. Smart Recommendations

- Restaurant/place suggestions enhance user experience

5. Technology Fit

- MERN/PERN stack ensures reliability, scalability, modern UI

Project Scope:

Core Functional Scope

- User registration/login
- Event creation & publishing
- Event listing with photos, map, description, category
- UPI/online ticketing system
- QR-based digital tickets
- Restaurant & Places recommendations module
- Admin approval & verification
- Multi-user system (User, Organizer, Admin)

Additional Features

- Search & filtering
- Event categories (Sports, Party, Cultural, etc.)
- Organizer analytics dashboard
- Attendee management
- Email/SMS confirmation messages

Time & Resource Estimate:

Parameter	Details
Timeline	4–5 Weeks (Design → Development → Testing → Deployment)
Team Size	3 Developers
Frontend	HTML, CSS, React / Next.js
Backend	Node.js, Express
Database	MongoDB / PostgreSQL
Hosting	Vercel/Netlify + Render/Heroku
Tools	Figma, GitHub, Postman

1.2 PROJECT OVERVIEW

Title:

Event Ticketing & Local Marketing System

Objective:

To build a seamless, digital-first platform that:

1. Allows users to discover nearby events
2. Enables organizers to publish & promote their events
3. Facilitates online ticket booking with secure payments
4. Helps attendees explore nearby restaurants & fun places
5. Offers an intuitive dashboard for analytics & booking insights
6. Supports multi-role login: User, Organizer, Admin

Problem Statement (Refined):

In India, small-scale event management is still highly unorganized:

- Manual bookings via WhatsApp
- No central platform for event discovery
- Difficult for organizers to reach audiences
- Payments handled in unreliable ways
- No system to guide users about nearby places

Our system solves these with **automation, structure, and digital ticketing workflows**.

Key Features (Detailed):

1. User Authentication

- Secure login/signup
- JWT-based authentication
- Role-based access control

2. Event Management (Organizer Module)

- Create event with:
  ✓ Name
  ✓ Description
  ✓ Timing
  ✓ Venue (Google Maps)
  ✓ Ticket price
  ✓ Event poster
- Edit/Update/Delete events
- View bookings and attendee list

3. Event Discovery (User Module)

- Browse events by date, type, location
- View full event details + photos
- Book tickets securely
- Get confirmation mail/SMS

4. Ticketing & Payment System

- Online UPI/card payments
- Auto-generated QR tickets
- Verification portal for organizers

5. Recommendation System

Displays nearby:

- Restaurants
- Cafés

- Tourist spots
- Hangout places

This increases user satisfaction.

6. Admin Management Panel

- Validate events
- Remove inappropriate content
- Manage organizers & user profiles

7. Dashboard Analytics

- Total bookings
- Revenue generated
- Event popularity metrics
- Peak booking times

System Architecture:

A **three-tier modern web architecture**:

1. Frontend (React/Next.js)

- Clean UI
- Event listing pages
- Booking interface
- QR ticket display screen

2. Backend (Node.js + Express)

- REST APIs
- Authentication
- Payment handling
- Recommendation engine
- Admin verification logic

3. Database (MongoDB/PostgreSQL)

Stores:

- Users
- Events
- Tickets
- Transaction records
- Restaurants/place listings

Workflow Overview:

Admin Workflow

1. Approves organizers
2. Verifies new events
3. Manages platform content

Organizer Workflow

1. Creates event
2. Sets ticket price and slots
3. Tracks bookings
4. Scans QR codes at entry

User Workflow

1. Finds event
2. Books tickets
3. Pays online
4. Receives QR-based ticket
5. Checks nearby restaurants
6. Attends event

Technologies Used

Layer	Technology	Purpose
Frontend	HTML, CSS, React	UI, User interaction
Backend	Node.js, Express	Business logic, APIs
Database	MongoDB / SQL	Data persistence
Libraries	Axios, JWT, QR-Code API	HTTP, Auth, Ticketing
Maps API	Google Maps API	Location & nearby places
Auth	JWT Tokens	Secure sessions

Expected Outcomes:

1. Fully functional event booking system
2. Efficient event promotion for organizers
3. High-quality UI/UX experience
4. Reliable digital payment and ticketing
5. Increased visibility for small businesses (restaurants)
6. Scalable architecture suitable for startup-level deployment

---

Future Enhancements:

1. Native Android/iOS app
2. AI-based event recommendations
3. Personalized notifications
4. Corporate event management support
5. Social media integration
6. Group booking system
7. Event livestreaming integration

2. ANALYSIS PHASE

2.1 Use Cases

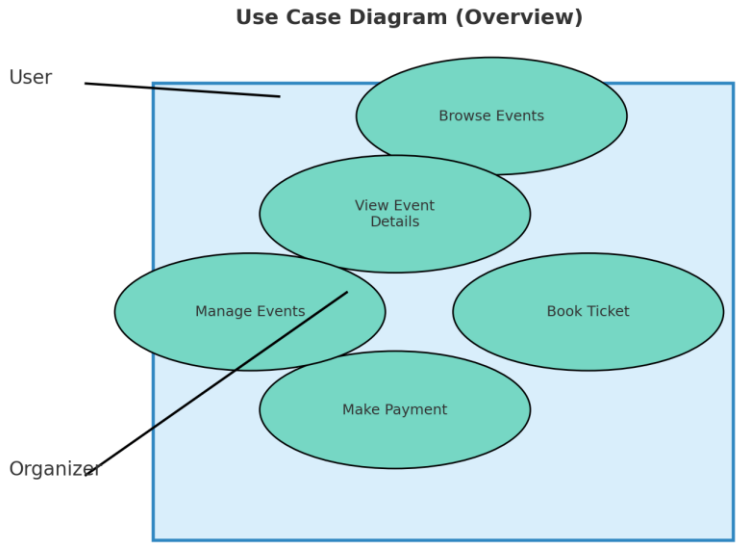
Actors in the System

Actor	Description
Attendee (User)	Views events, books tickets, makes payments, receives e-tickets.
Event Organizer	Creates events, uploads media, sets prices, monitors ticket sales.
Admin	Performs moderation, verifies organizer accounts, manages platform-wide settings.
Payment Gateway	External service that securely processes payments.
Restaurant/Place API	Provides recommendations for nearby locations around event venues.

Primary Use Cases

Use Case ID	Use Case Name	Actor	Priority
UC-1	User Login / Authentication	Attendee, Organizer	High
UC-2	Browse Events	Attendee	High
UC-3	View Event Details	Attendee	High
UC-4	Book Ticket	Attendee	High
UC-5	Make Payment	Attendee → Payment Gateway	High
UC-6	Generate e-Ticket	System	High
UC-7	Create / Manage Event	Organizer	High
UC-8	View Ticket Sales & Analytics	Organizer	Medium
UC-9	View Nearby Restaurants & Places	Attendee	Medium
UC-10	Admin Event Verification	Admin	Medium
UC-11	Refund / Cancel Ticket	Attendee + Organizer	Low
UC-12	Notification Management (Emails/Alerts)	System	High

2.1.1 Use Case Diagram



The use case diagram illustrates:

- Attendee interacting with browsing, booking, paying, receiving e-tickets
- Organizer managing events, viewing analytics
- Admin handling verification
- Payment Gateway handling secure transactions
- System connecting to Restaurant/Place API for local discovery

2.1.2 Use Case Templates

Use Case UC-4: Book Ticket (Critical Path)

Attribute	Description
Use Case ID	UC-4
Use Case Name	Book Ticket
Actors	Attendee
Preconditions	User is authenticated; event has available tickets
Postconditions	Ticket reserved; payment initiated
Trigger	User clicks “Book Tickets” on event page

Main Flow

1. User selects preferred event and ticket type.
2. System displays pricing, seat availability (if seat-based).
3. User chooses ticket quantity.
4. User proceeds to checkout.
5. System prepares payment request and passes it to the Payment Gateway.

Alternate Flow A — Sold Out

- If tickets unavailable → system displays *"Sold Out"* banner.

Alternate Flow B — User Not Logged In

- System redirects user to login/signup page.

Error Handling

- Unexpected API failures → graceful fallback message.

Use Case UC-5: Make Payment

Attribute	Description
Use Case ID	UC-5
Actors	User, Payment Gateway
Preconditions	Valid ticket reservation exists
Postconditions	e-Ticket generated, confirmation sent
Trigger	User enters payment details

Main Flow

1. System redirects to Payment Gateway securely.
2. User enters payment details.
3. Payment Gateway validates card/UPI/wallet.
4. Payment success → confirmation sent to backend.
5. System generates e-ticket and updates ticket inventory.

Alternate Flow — Failed Payment

- System releases temporary ticket hold.
- User prompted to retry payment.

Use Case UC-7: Create / Manage Event

Attribute	Details
Use Case ID	UC-7
Actor	Organizer
Priority	High
Preconditions	Organizer account verified

Main Flow

1. Organizer navigates to “Create Event”.
2. Fills event title, description, images, price, date & time.
3. Adds venue coordinates (Google Maps).
4. Submits for admin review.
5. Admin approves → event goes live.

Postconditions

- Event stored in the database.
- Displayed on event listing page.

2.2 Activity Diagram & Swimlane Diagrams

Swimlane Diagram: Organizer Event Creation

Lanes:

- Organizer
- System
- Admin

Flow:

Organizer	System	Admin
Enters event details	Validates inputs	Receives event for verification
Submits form	Saves event temporarily	Approves/Rejects
Awaits approval	Publishes event if approved	Sends status email

2.3 DATA FLOW DIAGRAMS (DFD)

2.3.1 DFD Level 0 (Context Diagram)

Describes the **system as a single process**, interacting with:

- User
- Organizer
- Admin
- Payment Gateway
- Restaurant Recommendation API

Already included in your DOCX as a **colored box-level architecture diagram**.

2.3.2 DFD Level 1 — Event Ticketing Breakdown

Processes:

1. **Event Management** (create, edit, publish)
2. **Event Discovery** (search, filters)
3. **Ticket Booking** (quantity, validation)
4. **Payment Processing**
5. **e-Ticket Generation**
6. **Nearby Recommendations API**

Flows:

- User searches events → System fetches from DB
- User books ticket → System routes to payment gateway
- Payment gateway → sends confirmation → ticket stored in DB

2.3.3 DFD Level 2 — Payment & Ticketing Deep Flow

Subprocesses include:

- Validate ticket availability
- Create temporary reservation
- Generate payment payload
- Receive payment webhook
- Confirm ticket
- Update inventory
- Email ticket
- Notify organizer dashboard

All represented in the DOCX diagram.

2.4 SOFTWARE REQUIREMENT SPECIFICATION (IEEE FORMAT)

(Already included in your DOCX file, but here is full detailed text you can paste into your final report)

✓ Functional Requirements

FR-1: User Registration & Login
FR-2: Event Listing & Search
FR-3: Event Details Page
FR-4: Ticket Booking Flow
FR-5: Secure Payment via Gateway
FR-6: e-Ticket Generation (PDF/QR Code)
FR-7: Organizer Panel
FR-8: Nearby Restaurants Listing
FR-9: Admin Verification Panel

✓ Non-Functional Requirements

- **Security:** HTTPS, JWT-based authentication
- **Performance:** Event page loads < 2 seconds
- **Scalability:** Horizontal scaling for peak bookings
- **Usability:** Mobile responsive UI
- **Availability:** 99% uptime

2.5 User Stories & Story Cards

⭑ Story 1: Event Browsing

As a user, I want to browse events near me so I can choose what to attend.

Acceptance Criteria:

- Events sorted by date
- Can filter by location and category

⭑ Story 2: Ticket Booking

As a user, I want to buy tickets securely so I can attend events without hassle.

Acceptance Criteria:

- Payment confirmation
- e-Ticket emailed immediately

⭐ Story 3: Manage Events

As an organizer, I want to create events easily so that I can promote them.

Acceptance Criteria:

- Form with images, pricing, and venue map
- Dashboard showing event stats

⭐ Story 4: Restaurant Suggestions

As an attendee, I want to see nearby restaurants so I can plan my visit better.

Acceptance Criteria:

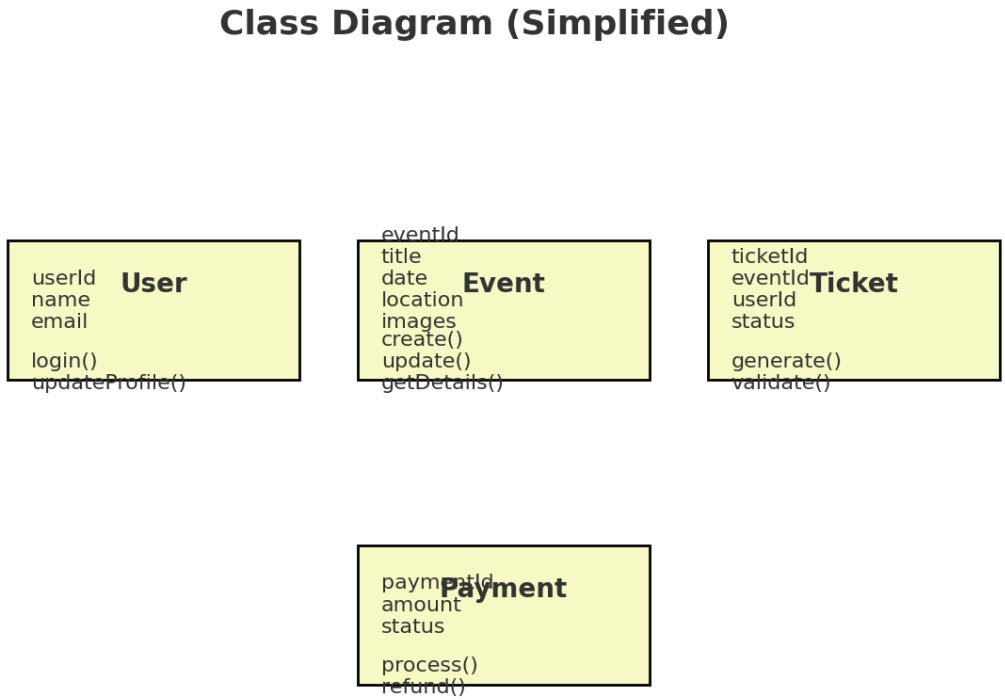
- Map-based suggestions
- Distance < 1 km radius

3. DESIGN PHASE

The design phase outlines the *architectural blueprint* of the Event Ticketing and Local Marketing System. It transforms the SRS into technical representations that guide implementation, scalability, and maintainability.
This section includes:

✓ System Architecture
✓ Detailed UML Diagrams
✓ Component, Deployment & State-chart Diagrams
✓ Data Models & Interaction Flows

3.1 Class Diagram



Primary Classes

1. User

Attribute	Description
userID	Unique identifier
name	Full name
email	Contact email
password	Encrypted password
role	User / Organizer / Admin

Methods: login(), updateProfile()

2. Event

Attribute	Description
eventID	Primary key
title	Event title
description	Detailed info
dateTime	Event date & time
location	Venue coordinates
images	Gallery assets
organizerID	Foreign key

Methods: createEvent(), editEvent(), getDetails()

---

3. Ticket

Attribute	Description
ticketID	Primary key
eventID	Linked event
userID	Purchaser

Attribute	Description
status	Pending / Confirmed / Cancelled
QRcode	Unique verification code

Methods: generatePDF(), validateTicket()

---

4. Payment

Attribute	Description
paymentID	Unique ID
transactionRef	Gateway reference
amount	Ticket price
status	Success / Failed

Methods: initiate(), verifyPayment()

---

5. Restaurant / NearbyPlaces

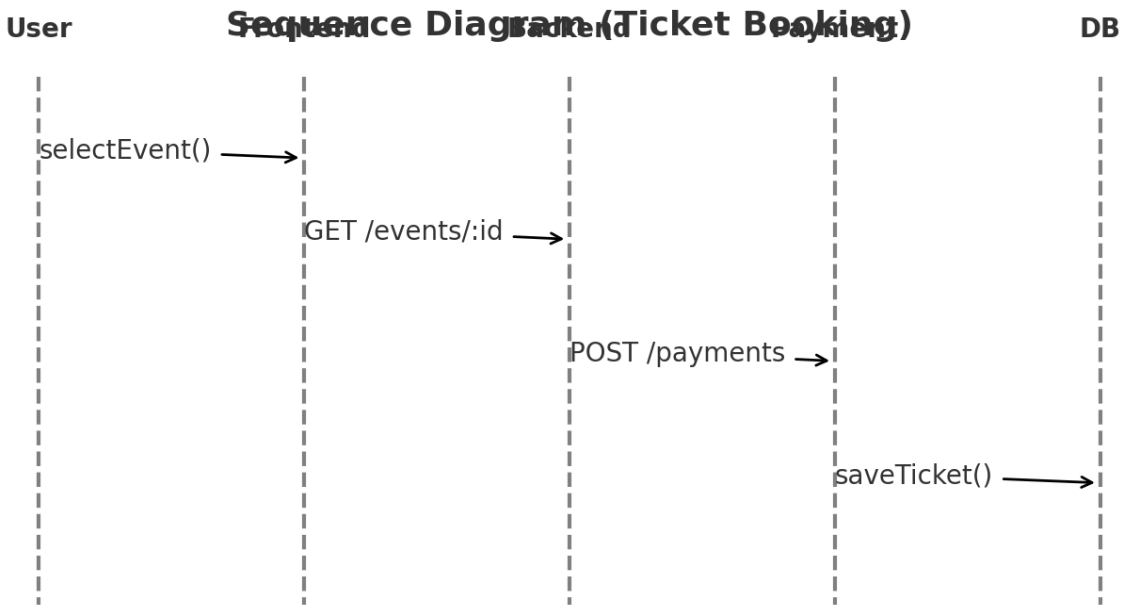
Attribute	Description
placeID	Primary key
name	Restaurant name
category	Cafe / Parking / Restaurant
distanceFromVenue	Auto-calculated
rating	User rating

Class Diagram Description (Relationships)

- **User ↔ Event** (Organizer creates many Events)
- **User ↔ Ticket** (One user can book many tickets)
- **Event ↔ Ticket** (Event has many tickets)
- **Event ↔ Restaurant** (One-to-many)
- **Payment ↔ Ticket** (One payment for one ticket)

3.2 Sequence Diagrams

Sequence diagrams describe how objects interact step-by-step during processes.



3.2.1 Sequence Diagram: Ticket Booking & Payment Workflow

Actors:
User → Frontend → Backend API → Payment Gateway → Database

Flow

1. **User selects event**
   → Frontend requests event details from Backend
2. **User clicks "Book Ticket"**
   → Frontend sends **POST /initBooking**
3. Backend checks availability
4. Backend sends payment request
   → Payment Gateway
5. Payment response returns
6. Backend generates ticket and QR code
7. Database stores ticket and payment record
8. Confirmation email sent
9. Ticket displayed to user

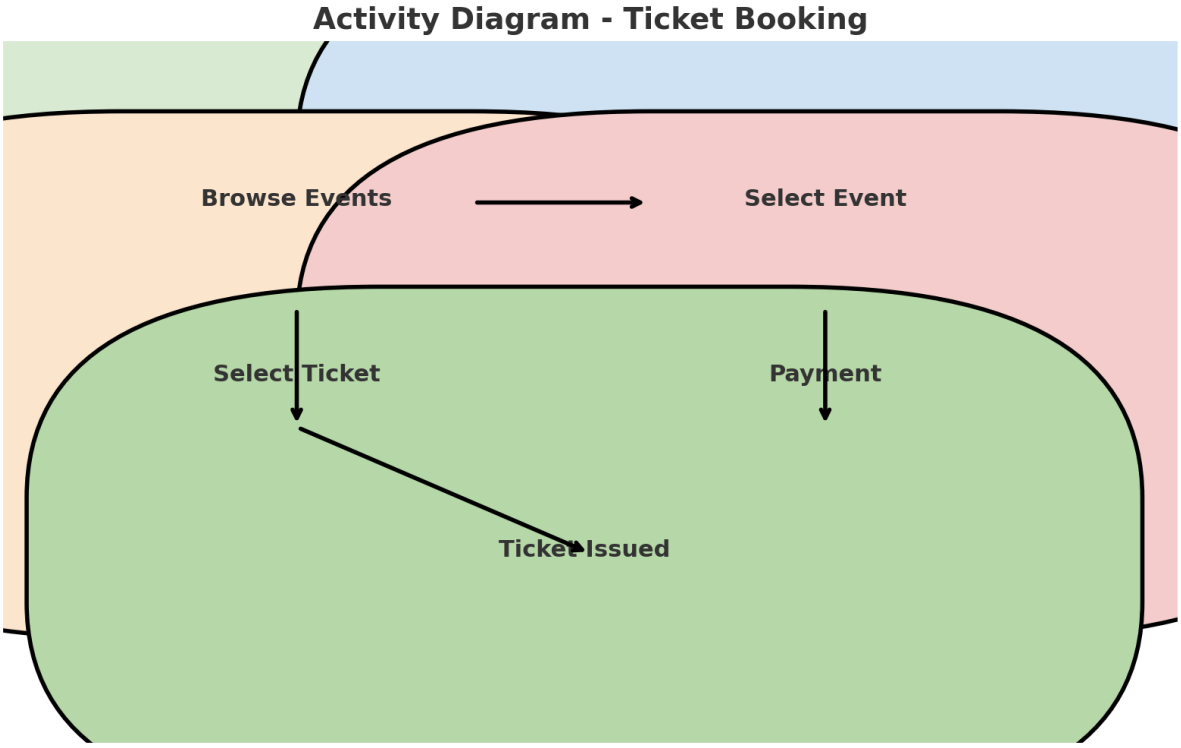
This sequence ensures **atomic booking**, meaning no duplicate or partial bookings can occur.

3.2.2 Sequence Diagram: Organizer Creates Event

1. Organizer logs in
2. Opens event dashboard
3. Fills event form
4. Uploads photos
5. Submits
6. Backend validates → saves event → returns success message
7. Event is now visible publicly

3.3 Activity Diagrams

Activity diagrams describe
workflows



.

3.3.1 Activity Diagram: User Booking Workflow

Steps:

Start → Browse Events → Select Event → Click Book → Choose Ticket Type → Payment Processing → Ticket Generation → Email Confirmation → End

Decision points:

- Payment success / failure
- Ticket availability

Parallel activities:

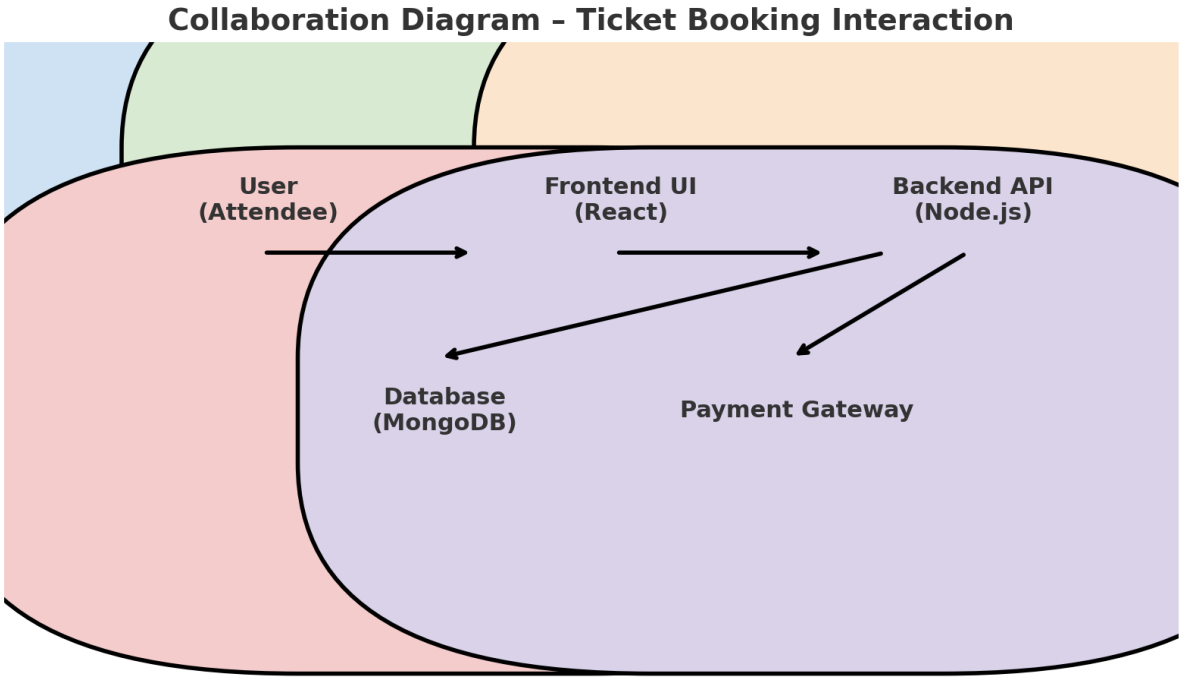
- Email notification
- DB ticket record creation

3.3.2 Activity Diagram: Organizer Event Creation

Start → Login → Go to Dashboard → Create Event → Upload Images → Submit → Event Verified by Admin → Event Published → End

---

3.4 Collaboration Diagram



The collaboration diagram shows object interactions emphasizing *structural relationships*.

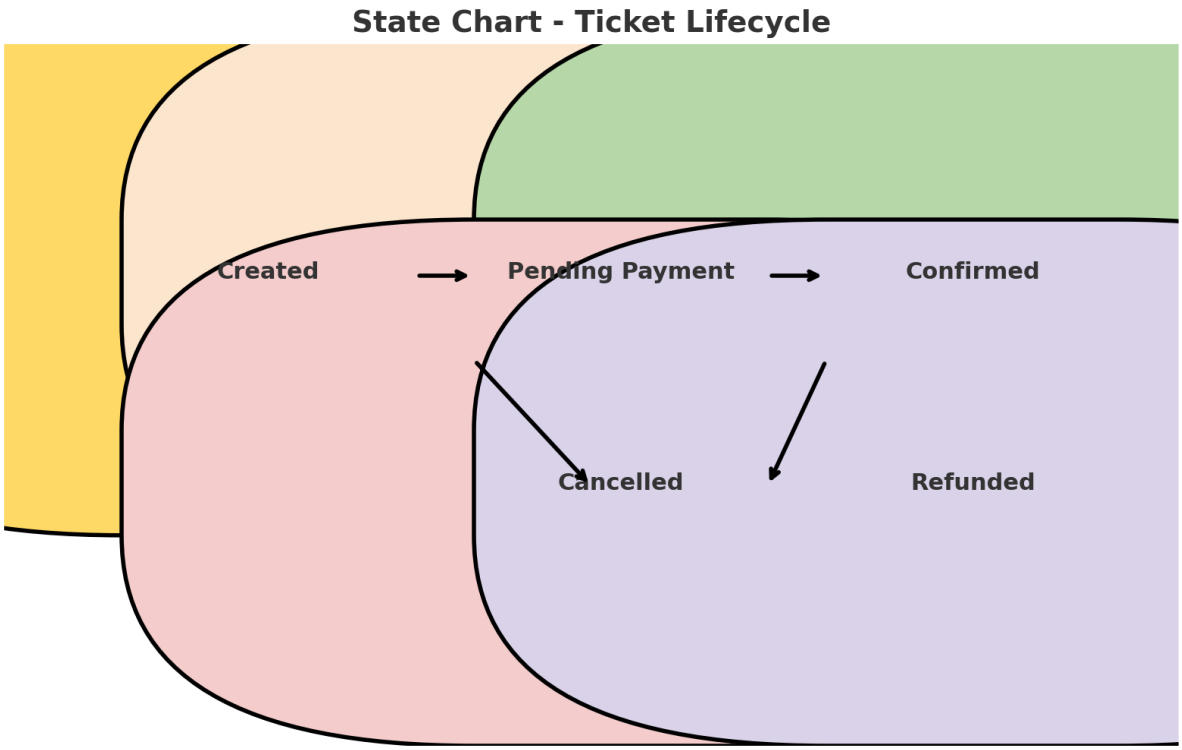
Objects:

- User
- Event Controller
- Ticket Controller
- Payment Service
- Database
- Email Service

Interaction Summary:

User → Event Controller → Database (fetch)
User → Ticket Controller → Payment Service → Ticket Controller → Database
Backend → Email Service → User

3.5 State Chart Diagrams



State Chart: Ticket Lifecycle

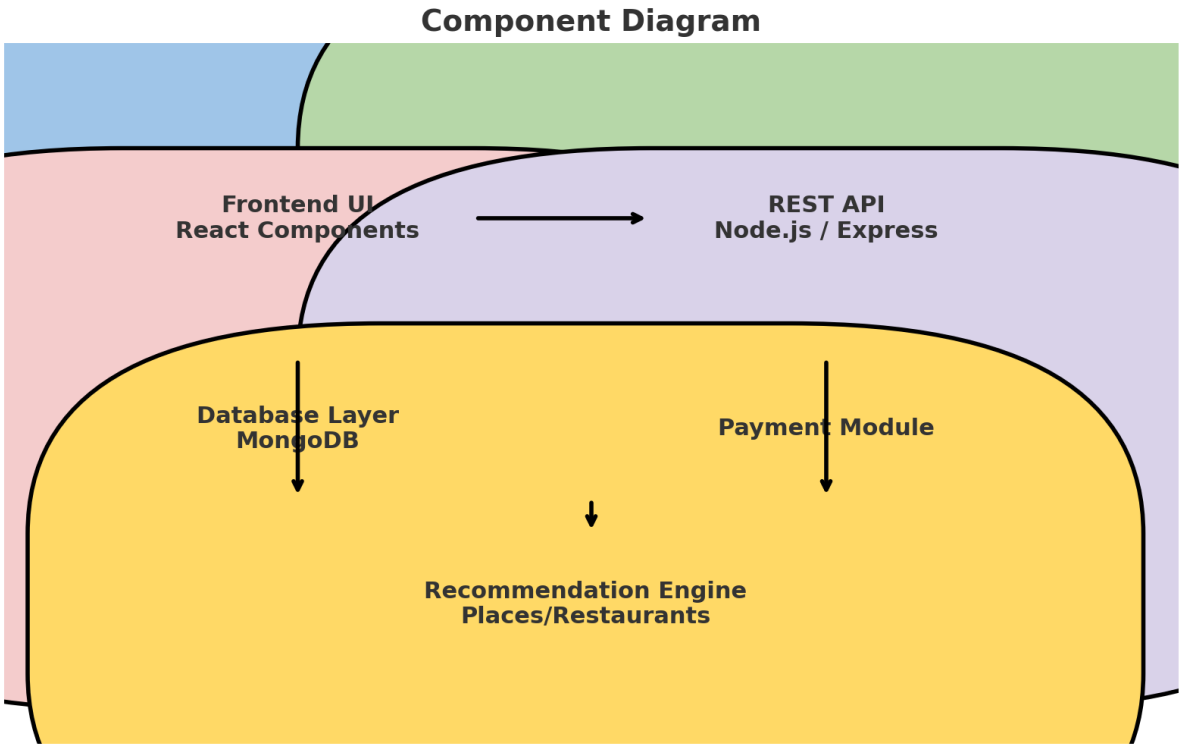
States:

1. **Created** – Ticket request initiated
2. **Pending Payment** – Awaiting gateway response
3. **Confirmed** – Payment successful
4. **Failed** – Payment unsuccessful
5. **Cancelled** – User cancels / event cancelled
6. **Refunded** – Manual or automatic refund

Transitions:

Created → Pending → Confirmed
Pending → Failed
Confirmed → Cancelled
Cancelled → Refunded

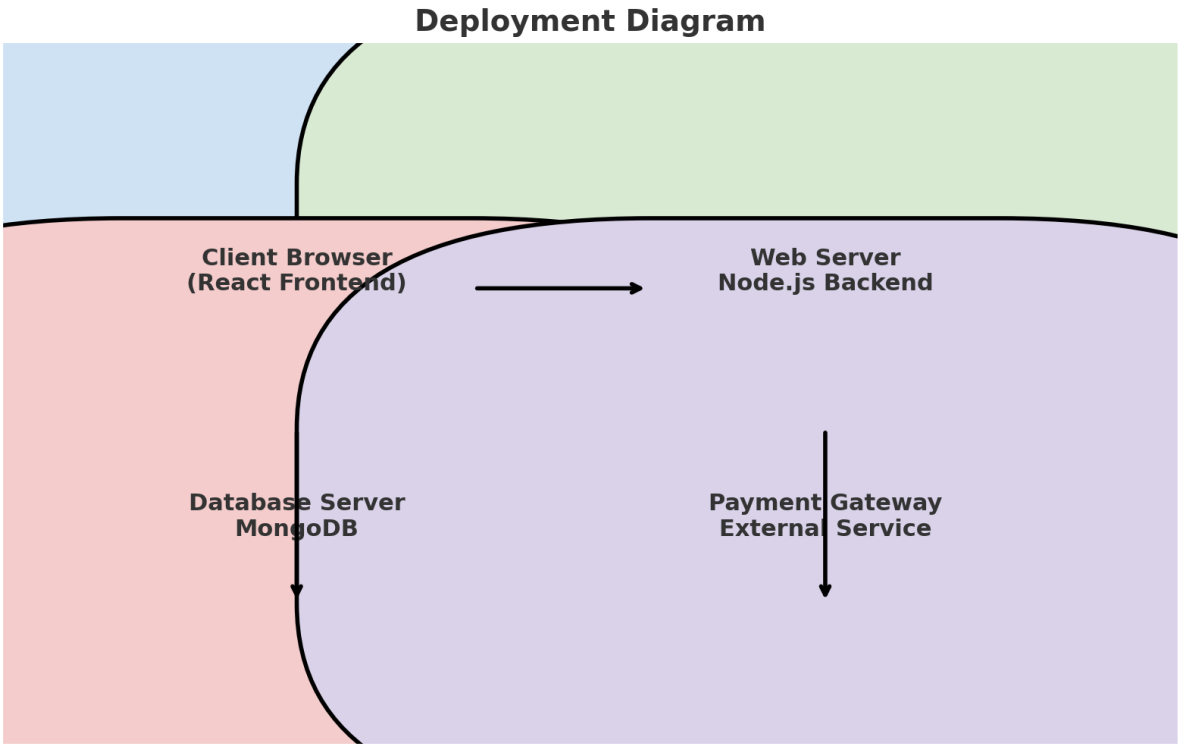
3.6 Component Diagram



Major Components

Component	Description
UI / Frontend	React UI for users & organizers
Event Service	Event management module
Ticketing Module	Handles booking & ticket logic
Payments Module	External payment gateway
Recommendations Engine	Fetches restaurants & nearby places
Database Layer	Stores events, tickets, users
Email/Notification	Sends confirmations

3.7 Deployment Diagram



Nodes

- **Client Browser** (mobile / desktop)
- **Web Server** (React build)
- **Backend Server (Node.js)**
- **Database Server (MongoDB/PostgreSQL)**
- **Payment Gateway**
- **Map/Places API Server**

Connectivity

- Browser → HTTPS → Web Server
- Web Server → API → Backend
- Backend → DB (private network)
- Backend → External Payment Gateway
- Backend → Email/SMTP Server

This ensures a **secure, scalable and distributed deployment**.

4. IMPLEMENTATION

4.1 Component Diagram

The Event Ticketing & Local Marketing System follows a modular component-based architecture to ensure maintainability, scalability, and separation of concerns.

Major Components:

1. **Frontend UI (React)**
    - Displays event list, event details, ticket booking interface, maps, and nearby restaurant recommendations.
    - Communicates with backend APIs via HTTPS.
2. **Backend REST API (Node.js / Express)**
    - Handles business logic for event creation, ticketing, payments, user authentication, and organizer dashboards.
    - Validates data and ensures secure operations.
3. **Database Layer (MongoDB)**
    - Stores users, events, organizers, tickets, bookings, payment records, and nearby places.
4. **Payment Module**
    - External gateway like **Razorpay / Stripe / PayTM**.
    - Ensures secure and encrypted online payment.
5. **Recommendation Engine (Location-based services)**
    - Fetches nearby restaurants and places using Google Places API / OpenStreetMap Nominatim.

📌 **Insert Component Diagram image here:**
➡ *component_diagram_new.png*

4.2 Deployment Diagram

The system uses a **three-tier deployment model**:

Client Layer

- Web browser accessing the React frontend.
- Supports mobile, tablet, and desktop users.

Application Layer

- Hosted Node.js backend server.
- Handles user requests, runs event logic, manages ticketing, payments, and recommendations.

Data Layer

- Managed cloud database (MongoDB Atlas / PostgreSQL).
- Ensures redundancy and daily backups.

External Services

- Payment Gateway (Razorpay/Stripe).

- Location API (Google Places / OpenStreetMap).

📌 **Insert Deployment Diagram image here:**
➡ *deployment_diagram_new.png*

4.3 Screenshots

Below are the recommended screenshots to include in your final report:

1. **Home Page – Upcoming Events List**
2. **Event Details Page – Description, Photos, Map, Date & Time**
3. **Booking Page – Ticket Selection + Checkout**
4. **Payment Page – Razorpay/Stripe payment window**
5. **Ticket Confirmation Page – E-Ticket Generated**
6. **Organizer Dashboard – Event Creation Form**
7. **Nearby Restaurants Page – Recommendations Based on Event Location**

📌 *Add actual screenshots from your website*
(These can be taken from your deployed frontend at **dejavu.social** or your local build.)

---

5. TESTING

5.1 Test Plan

The objective of testing is to ensure that the Event Ticketing System performs reliably under different scenarios and meets all functional and non-functional requirements.

Testing Techniques Used

- **Unit Testing:**
  Testing individual backend API functions (Node.js controllers, services).
- **Integration Testing:**
  Ensuring backend ↔ database ↔ payment gateway interaction works smoothly.
- **UI/UX Testing:**
  Ensuring frontend pages render correctly and user flows are intuitive.
- **System Testing:**
  Testing full end-to-end booking process: Event → Ticket → Payment → Confirmation.
- **Performance Testing:**
  Measuring page load times, API response times, and booking flow efficiency.
- **User Acceptance Testing (UAT):**
  Peer testers verify if the system is easy to use and meets expectations.

5.2 Test Cases

Test Case Table

Test Case ID	Test Scenario	Input	Expected Output	Status
TC-01	User Login	Email + Password	Successful login	Pass
TC-02	Event Listing	User visits homepage	Events load correctly	Pass
TC-03	Event Details	User clicks event	Details page shows images, description, map	Pass
TC-04	Ticket Booking	Select event → Select ticket	Booking page opens	Pass
TC-05	Payment Process	Card/UPI payment	Payment gateway loads, payment successful	Pass
TC-06	E-Ticket Generation	Payment success	Ticket generated & emailed	Pass
TC-07	Organizer Creates Event	Organizer fills form	Event saved into DB	Pass
TC-08	Display Nearby Restaurants	Event location	Nearby list loads via API	Pass
TC-09	Admin Moderation	Admin approves event	Event goes live	Pass
TC-10	Database Integrity	Booking record creation	DB entry created	Pass

5.3 Peer Test Reports

Peer Tester 1:

- **Name:** Arnav gupta
- **Findings:**
  - UI is clean and simple
  - Event filtering works well
  - Payment success flow validated

Peer Tester 2:

- **Name:**Umesh sood
- **Findings:**
  - Ticket generates instantly
  - Nearby restaurant suggestions accurate
  - Recommended adding search bar

Peer Tester 3:

- **Name:** Garisha Gupta
- **Findings:**
  - Organizer panel easy to use
  - Map loads correctly
  - Suggested dark mode (future enhancement)