**Full Stack Project**

**Title Page**

* **Project Title**

EventEase: Event Booking and Ticket Management for College Fests

* **Institution / Organization**

IBM

* **Mentor/Supervisor**

Jaivik Sir

* **Submission Date**

22/07/2025

**Certificate**

This is to certify that the project titled **"EventEase: Event Booking and Ticket Management for College Fests"** submitted by **Shreya Shirish Bhangale** is an original work carried out under the guidance of Jaivik Sir,in partial fulfillment of the requirements for the awardat **IBM**.

Place: \_\_\_\_\_\_\_\_  
Date: \_\_\_\_\_\_\_\_

Signature of Guide/Supervisor Signature of Head of Department

**Acknowledgment**

We would like to express our sincere gratitude to **[Supervisor/Guide Name]** for their valuable guidance and constant encouragement. We also thank our peers and family for their support, and **[Institute Name]** for providing us with the resources to complete this project successfully.

**Table of Contents**

1. Abstract
2. Objective
3. System Architecture
4. Technology Stack
5. Modules / Features
6. Frontend Development
7. Backend Development
8. Database Design
9. Data Flow Diagrams
10. Testing (optional)
11. Security Measures
12. Limitations
13. Future Enhancements
14. Screenshots
15. Annexures
16. References

**Abstract**

* **What**: EventEase is a full-stack web application designed to manage college fests by allowing students to browse, book, and manage tickets for various events.
* **Why**: To simplify and digitize event registration, reduce manual effort, and improve user experience.
* **Key features**: User authentication, ticket booking, event dashboard, admin panel to add/update events, and automated email confirmations.

**Objective**

* Develop an intuitive platform to manage college events.
* Enable students to book, view, and cancel tickets easily.
* Provide an admin interface for event organizers to manage events.

**System Architecture**

**Architecture Diagram:**

Frontend (React)

↕ (API calls)

Backend (Node.js + Express)

↕

Database (MongoDB)

**Description**:

* Users interact via the React frontend.
* Frontend communicates with backend REST APIs.
* Backend handles business logic and interacts with MongoDB to store user and event data.

**Technology Stack**

| **Layer** | **Technology** |
| --- | --- |
| Frontend | HTML, CSS, JavaScript, React |
| Backend | Node.js, Express.js |
| Database | MongoDB |
| Tools | VS Code, GitHub, Postman, npm |

**Modules / Features**

| **Module** | **Description** | **Technologies** |
| --- | --- | --- |
| Authentication | User signup/login, password hashing | bcrypt, JWT |
| Event Dashboard | View list of events | React |
| Ticket Booking | Book/cancel tickets, see history | MongoDB, Express |
| Admin Panel | Add/update/delete events, view booked tickets | React, Node.js |

**Frontend Development**

* **Framework**: React.js
* **Page structure**: Login, Signup, Dashboard, Event Details, My Tickets, Admin Panel
* **UI/UX Strategy**: Simple and mobile-friendly with modern UI components.

**Backend Development**

* **Framework**: Node.js with Express.js
* **API Structure**: RESTful APIs
* **Routing & Controllers**:
  + /api/auth: login, register
  + /api/events: CRUD for events
  + /api/tickets: booking, cancellation
* **Authentication**: JWT-based, password hashing with bcrypt

**Database Design**

* **DBMS**: MongoDB
* **ER Diagram**:

| **Collection** | **Fields** |
| --- | --- |
| Users | userId, name, email, passwordHash, role (user/admin) |
| Events | eventId, title, description, date, price, capacity |
| Tickets | ticketId, userId, eventId, bookingDate, status |

* **Sample Data**: Show example JSON documents.
* **Relationships**: One user → many tickets; one event → many tickets.

**Data Flow Diagrams**

* **Level 0**: User interacts with system to view/book events.
* **Level 1**: Data travels from frontend → backend → database and back as JSON.

**Testing (Optional)**

* API testing with Postman: /api/login, /api/events
* Manual testing of booking flows.
* Test cases: valid login, invalid login, double booking, etc.
* Bug fixing log.

**Security Measures**

* Passwords stored as hashed (bcrypt).
* JWT for session management.
* Input validation & sanitation to prevent XSS and SQL injection.

**Limitations**

* Admin panel not yet role-protected fully.
* No email notifications for cancellations.
* Limited scalability for large-scale events.

**Future Enhancements**

* Payment integration.
* QR code-based check-in.
* Email & SMS alerts.
* Responsive mobile app version.

**Annexures**

* Key code snippets (e.g., booking logic, JWT middleware)
* API Documentation: list of routes, methods, and response format

**References**

* Node.js Docs
* React Docs
* MongoDB Docs
* NPM Libraries used (bcrypt, express-validator, etc.)