Project: Event Management System

1. Introduction

This document provides the Low-Level Design (LLD) for an **Event Management System** designed to manage event planning, registrations, ticketing, and attendee engagement.

This design supports both **Java (Spring Boot)** and **.NET (ASP.NET Core)** frameworks for backend development.

2. Module Overview

The system consists of the following modules:

2.1 Event Management

- Manages event details, schedules, and updates.

2.2 User Registration

- Handles user sign-ups and profile management.

2.3 Ticket Booking

- Facilitates booking, cancellation, and payment for event tickets.

2.4 Notifications and Reminders

- Sends event-related updates and reminders to users.

2.5 Feedback and Ratings

- Allows users to provide feedback and rate events.

3. Architecture Overview

3.1 Architectural Style

- Frontend: Angular or React
- Backend: REST API-based architecture
- Database: Relational Database (MySQL/PostgreSQL/SQL Server)

3.2 Component Interaction

- Frontend communicates with the backend via REST APIs for all user interactions.
- Backend manages the database for CRUD operations and triggers notifications.

4. Module-Wise Design

4.1 Event Management Module

Features:

- Create, update, and delete event details.
- Search and filter events by category, date, or location.

Data Flow:

- Admin interacts with the frontend to manage events.
- Requests are sent to the backend for processing and database updates.
- Event data is displayed to users on the frontend.

Entities:

- **Event**:
    - EventID
    - Name
    - Category
    - Location
    - Date
    - OrganizerID

4.2 User Registration Module

Features:

- User sign-up and login.
- Manage user profiles.

Data Flow:

- Users provide details via the frontend.
- Backend validates and stores user data in the database.
- Confirmation is sent to the user interface.

Entities:

- **User**:
    - UserID
    - Name
    - Email
    - Password
    - ContactNumber

4.3 Ticket Booking Module

Features:

- Book tickets for events.
- View and cancel tickets.

Data Flow:

- Users select events and book tickets through the frontend.
- Backend processes the booking, updates ticket availability, and confirms the booking.
- Confirmation and tickets are displayed on the frontend.

Entities:

- **Ticket**:

    - TicketID
    - EventID
    - UserID
    - BookingDate
    - Status (Confirmed/Canceled)

4.4 Notifications and Reminders Module

Features:

- Notify users about event updates.
- Send reminders for upcoming events.

Data Flow:

- Backend triggers notifications based on predefined schedules.
- Notifications are sent via email, SMS, or displayed on the frontend.

Entities:

- **Notification**:
    - NotificationID
    - UserID
    - EventID
    - Message
    - SentTimestamp

4.5 Feedback and Ratings Module

Features:

- Collect user feedback and ratings for events.
- Display average ratings for events.

Data Flow:

- Users submit feedback and ratings via the frontend.
- Backend processes and stores the data in the database.
- Feedback summaries are displayed on the frontend.

Entities:

- **Feedback**:
    - FeedbackID
    - EventID
    - UserID
    - Rating
    - Comments
    - SubmittedTimestamp

5. Deployment Strategy

5.1 Local Deployment

- Frontend: Served using local servers (e.g., ng serve for Angular or equivalent for React).
- Backend: Deployed locally using Spring Boot or ASP.NET Core.

- Database: Local instance setup for testing.

6. Database Design

6.1 Tables and Relationships

1. **Event**
   - Primary Key: EventID
2. **User**
   - Primary Key: UserID
3. **Ticket**
   - Primary Key: TicketID
   - Foreign Keys: EventID, UserID
4. **Notification**
   - Primary Key: NotificationID
   - Foreign Key: UserID
5. **Feedback**
   - Primary Key: FeedbackID
   - Foreign Keys: EventID, UserID

7. User Interface Design

7.1 Wireframes:

- Event Dashboard
- Registration and Login Pages
- Ticket Booking and History Page
- Notifications Panel
- Feedback Submission Page

8. Non-Functional Requirements

8.1 Performance

- The system must handle 200 concurrent users in the local environment.

8.2 Scalability

- Designed for future scalability to support multiple regions.

8.3 Security

- Implement authentication and role-based access control.

8.4 Usability

- Provide a user-friendly interface with responsive design.