Software Design Document

Event Management System

Authors:

Muhammad Bin Nasir	220201012
Aiza Shafqat	220201036
Muzammil Ali	220201061
Ali Jawad	220201013

Group Number: 02

Affiliation:

Department of Computer Science

Institute of Space Technology

Date: 9th January, 2025

Table of Contents

This page left blank intentionally

1. Introduction

1.1. Purpose

This Software Design Document serves as a document describing the architecture and the system design of the software. It is used as a guideline for the developers, all the stakeholders, as well as the clients and the potential end users. It also helps the people involved in the evaluation of the software system. This document provides deep insights into the product, ensuring that the product is being created in accordance with the objectives and requirements.

1.2. Aim

Event Management System is a software created keeping in mind the hefty and tiring management of events, and with a sole purpose to make the work of the management and Admin teams easier. It is designed to simplify the management of different events, like conferences, weddings and corporate fests. Our software includes features like:

1. Event Scheduling
2. Attendee Registration
3. Seats Allocation
4. Food and Seating management
5. Reporting

Its goals and objectives are:

1. Provide a centralized platform for managing events efficiently.
2. Enable seamless collaboration between organizers and participants.
3. Automate repetitive tasks like ticket generation and attendee communication.
4. Offer analytical insights for future event improvements.

The benefits of having a centralized system of Event Management are:

1. Time saving, and cost effective method.
2. A much enhanced user experience.
3. A much improved and efficient Event management.

1.3. Overview:

This document has been organized into sections describing the scope, key features and purpose of the software. The document consists of the information about architecture and design of the software, along with the necessary technical details to supplement the information.

1.4. Reference Material:

- Agile Software Development principles.
- IEEE Std 1016-2009 for Software Design Descriptions.
- User requirements documentation (SRS).

1.5. Definitions and Acronyms:

1. **EMS**: Event Management System.
2. **CRUD**: Create, Read, Update, Delete (basic database operations).
3. **UI/UX**: User Interface/User Experience.
4. **API**: Application Programming Interface.
5. **RSVP**: Répondez s'il Vous Plaît (please respond to an invitation).

2. System Overview

Event Management System is a comprehensive platform which is very handy for an excellent management of all kinds of events that take place, whether at the corporate sector, or the personal events, ranging from weddings to birthday parties. This software is comprised of different modules integrated together, which are:

2.1. Event Management System:

This module helps the end users to create, delete, schedule, update and/or edit the events created, or to be created. This module is the backbone of the whole software, with responsibility of updating and keeping track of all the events in real time.

2.2. User Management:

This module takes care of the user's registration, authentication, and roles assignment. This allows the system to be securely accessed and utilized by administrators, organizers, and attendees.

2.3. Payment Processing

The payment processing module includes an integration with a payment gateway, management of refunds, and creation of receipts. This ensures seamless and secure financial transactions in case of event registration and ticketing.

2.4. Notification System

This system auto-reports communications by email and SMS notifications so that attendees, organizers, and others are informed about the important information for events.

2.5. Analytics and Reporting

The module will report on event attendance and revenue metrics. It gives analysis of the feedback, enabling the organizers to have a look at the performance of their events and the planning for subsequent events.

3. System Architecture

The Event Management System EMS system architecture is designed to be scalable, flexible, and user-friendly. It applies the multi-tier approach in order to separate user interaction and business logic from data management for efficiency and security.

3.1. Architectural Design

The EMS consists of three primary layers:

- ➔ **Presentation Layer:**
  - ◆ It provides user-friendly interfaces for **admins**, **organizers**, and **attendees**.
  - ◆ Developed with **HTML5, CSS3**, and using a framework like **React** or **Angular** for the responsive design.
  - ◆ User login and authentication are implemented along with browsing events and buying tickets.
- ➔ **Business Logic Layer:**
  - ◆ It performs all the primary functionalities of the system, like scheduling events, managing attendees, and sending notifications.
  - ◆ It is built with backend technologies such as Node.js or Python using Django/Flask.
  - ◆ Communicates with the frontend by REST APIs.
- ➔ **Data Layer:**
  - ◆ Manages the secure storage of data.

- It uses MySQL or PostgreSQL for structured data and MongoDB for semi-structured data.
- Data contains user profiles, event details, payment records, and feedback.

This layering architecture helps in making EMS scalable, maintainable, and adaptable to any future enhancements.

3.2. Design Paradigm

The Event Management System adopts the Model-View-Controller (MVC) architectural pattern to ensure a clear separation of concerns and enhance the system's scalability and maintainability.

Model

1. The basal layer with all functionalities of the software, and controls the backend of the program.
2. Interacts with the database to handle user profiles, event details, payments, and feedback.
3. Ensures data consistency and implements business rules.

View

1. Represents the user interface of the system, focusing on delivering an intuitive and responsive experience.
2. Built using React or Angular frameworks to dynamically present data from the Model.
3. Provides separate views for admins, organizers, and attendees.

Controller

1. Is the middle layer between the View and the Model layers.
2. Handles user inputs from the View, processes them, and updates the Model accordingly.
3. Uses REST APIs for smooth communication between the two layers.

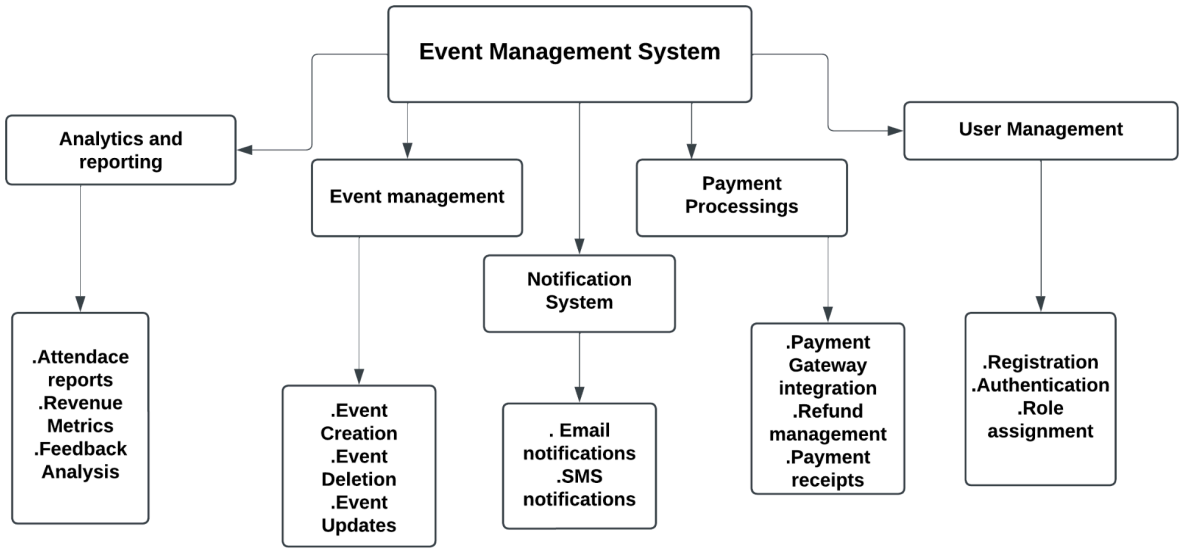
By leveraging the MVC model, the system achieves modularity, making it easier to implement new features, modify existing functionalities, and perform debugging or testing independently within each layer.

3.3. Decomposition Diagram

The EMS is divided into the following functional components:

- ➔ **Authentication System:** Manages secure user login and role-based access.
- ➔ **Event Management Module:** Allows event creation, updates, and cancellations.
- ➔ **Ticketing and Payment System:** Handles ticket sales and integrates with secure payment gateways.
- ➔ **Notification Module:** Sends updates and reminders via email and SMS.
- ➔ **Analytics Module:** Generates reports on event performance and revenue.
- ➔ **Feedback System:** Collects and analyzes attendee feedback.

These components interact through clearly defined interfaces, enabling smooth functionality and easy troubleshooting.



3.4. Design Rationale

The EMS design choices reflect a focus on user experience, scalability, and data security:

- ➔ **User Experience:** The highly responsive and intuitive interface allows for most key actions such as signing up for an event to happen within a couple of clicks.
- ➔ **Scalability:** Cloud deployment in platforms such as AWS or Azure ensures it will scale as well with growing users and increased volume of data.

➔ **Security:** Through HTTPS protocol, SSL/TLS encryption, the application is safeguarding user information as well as the transaction.

This rationale underpins the system's ability to meet user needs while maintaining performance and reliability.

4. Design Considerations

The design considerations for the Event Management System ensure the solution is robust, scalable, user-friendly, and aligns with stakeholder requirements. Below are key aspects to be addressed:

4.1. Assumptions and Dependencies

Assumptions:

a. Users will access the system primarily through web browsers and mobile devices.
b. Internet connectivity will be stable during usage.
c. Admin users are expected to have basic technical knowledge to manage events.

Dependencies:

- Integration with third-party APIs, such as payment gateways (e.g., HBL Secure Pay, PayPal) for ticket purchases.
- Reliance on a cloud database for scalability and data storage (e.g., AWS RDS, Firebase).
- Notifications will be handled via email and SMS services (e.g., Twilio, SendGrid).

4.2. Constraints

- The system must handle a large number of simultaneous users during peak event times (e.g., ticket sales for popular events).
- Budget constraints may limit the use of premium third-party APIs.
- The design must support compliance with GDPR and other regional data protection laws.
- Cross-platform compatibility is essential, ensuring the system works on desktop and mobile devices.

4.3. Risks and Issues

Scalability Risk:

High traffic during ticket sales or event registration may lead to system crashes.

Mitigation: Use a load balancer and deploy the system on a scalable infrastructure.

Data Breaches:

Sensitive data such as attendee information and payment details may be at risk.

Mitigation: Implement encryption for data storage and transmission, and conduct regular security audits.

API Downtime:

Dependency on external APIs (e.g., payment gateways) could lead to service disruption.

Mitigation: Implement fallback mechanisms and notify users in case of delays.

4.4. Performance Goals

- The system should respond to user actions (e.g., ticket booking, registration) within 2 seconds under normal load.
- The platform must support at least 10,000 concurrent users without significant degradation in performance.
- Reports and analytics should be generated within 5 seconds for standard queries.

4.5. User Experience (UX) Considerations

- A simple and intuitive interface for users to browse events, register, and purchase tickets.
- Personalized user dashboards for event organizers and attendees.
- Mobile-first design to ensure seamless operation on smartphones and tablets.

5. Data Design

Effective data design is crucial for the EMS as it involves managing diverse datasets related to users, events, and transactions.

5.1 Data Description

The EMS processes and stores the following types of data:

- ➔ **User Data:** Includes roles (admin, organizer, attendee), credentials, and profile details.
- ➔ **Event Data:** Stores event details like name, description, date, time, and capacity.
- ➔ **Transaction Data:** Includes ticket sales, payment statuses, and refunds.
- ➔ **Feedback Data:** Captures attendee reviews and ratings for events.

This data is structured to ensure easy retrieval and analysis, enabling smooth system operations and insightful reporting.

5.2 Data Dictionary

The following table outlines the key fields in the EMS database:

Field Name	Description	Type	Constraints
UserID	Unique identifier for users	Integer	Primary Key
Username	User's login name	String (50)	Unique,Not Null

EventID	Unique identifier for events	Integer	Primary Key
EventName	Name of the event	String (100)	Not Null
PaymentID	Unique identifier for payments	Integer	Primary Key
Amount	Payment amount	Decimal(10,2)	Not Null
FeedbackID	Unique identifier for feedback	Integer	Primary Key
FeedbackText	Attendee feedback	Text	Optional

This structured data organization ensures that the system can efficiently manage user interactions, event logistics, and financial transactions.

6. Component Design

The EMS is modular in design, with each component handling specific functionalities while seamlessly integrating with others.

6.1 Overview

The modular architecture ensures that components can be developed, tested, and maintained independently. This approach enhances system reliability and scalability.

6.2 Components

➔ Authentication System:

- **Description:** Manages secure login and role-based access control.
- **Technologies:** OAuth2, JWT for token-based authentication.
- **Input:** Username and password.
- **Output:** User-specific dashboard.

➔ Event Management Module:

- **Description:** Enables event creation, updates, and management.
- **Technologies:** Backend APIs with database integration.

- **Input:** Event details (name, date, venue).
- **Output:** Event listings.

➔ Ticketing and Payment System:

- **Description:** Facilitates ticket sales and processes payments.
- **Technologies:** Integration with PayPal/Stripe APIs.
- **Input:** Ticket quantity, payment details.
- **Output:** Payment confirmation and ticket generation.

➔ Notification Module:

- **Description:** Sends email/SMS updates and reminders.
- **Technologies:** SMTP server for emails, SMS gateway.
- **Input:** Notification content.
- **Output:** Delivered notifications.

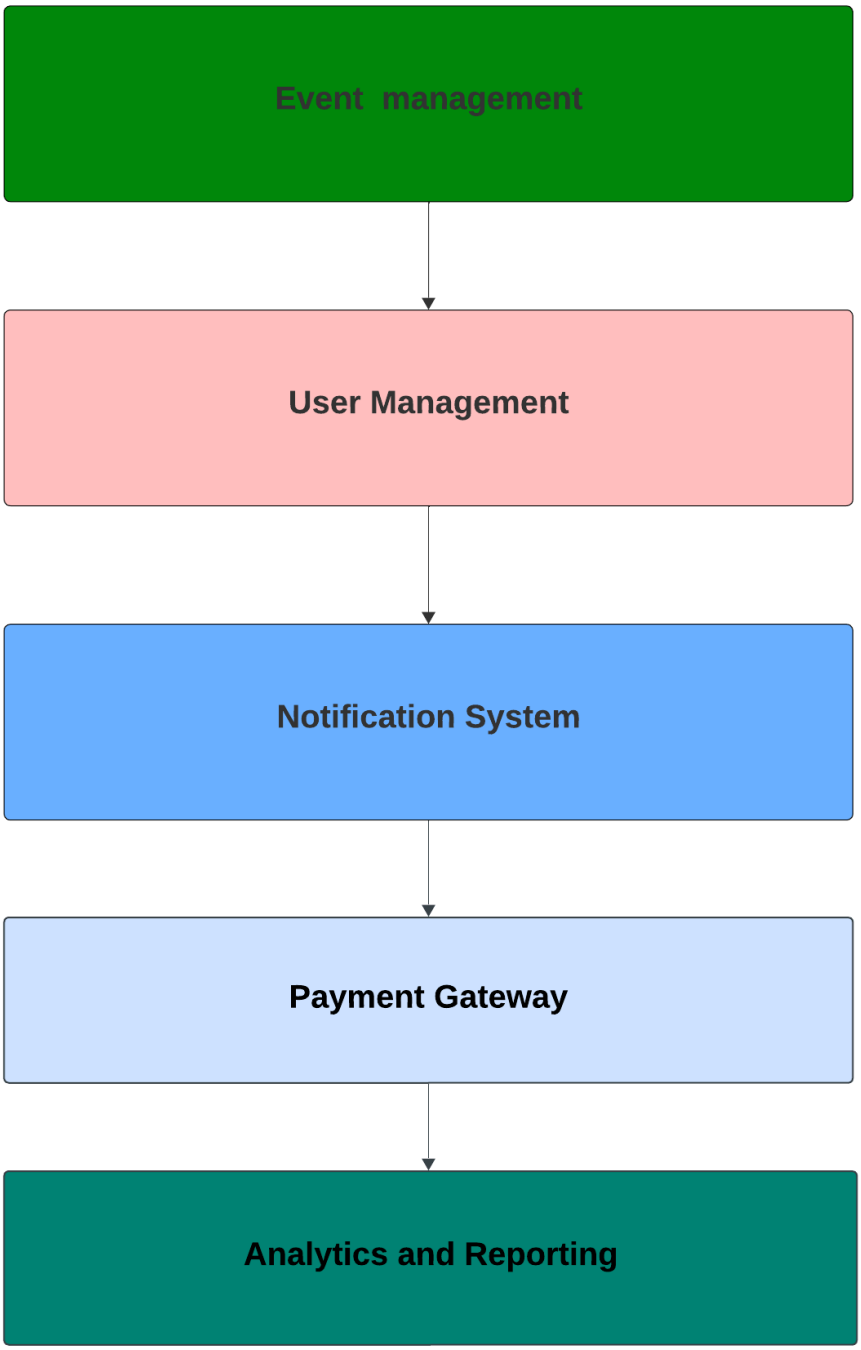
➔ Analytics and Reporting Module:

- **Description:** Provides insights into event performance and revenue.
- **Technologies:** Data visualization tools like Matplotlib, Tableau.
- **Input:** Event and user data.
- **Output:** Graphical reports and statistics.

➔ Feedback System:

- **Description:** Captures and analyzes attendee feedback.
- **Technologies:** Feedback forms integrated with the database.
- **Input:** User-submitted feedback.
- **Output:** Feedback summaries and recommendations.

Component Diagram covering all the essential parts of Event Management System is as follows:


Event management
User Management
Notification System
Payment Gateway
Analytics and Reporting

7. Human Interface Design

7.1 Brief User Interface Overview

Event Management System (EMS) intended to be a user-centric application to evolve event planning, registration and management process. According to a user's perspective, the system actually has these functional features.

7.1.1. Event Organizers:

1. Create and manage events by entering essential details such as event name, date, venue, description, type, price, and capacity.
2. Monitor event performance through a dashboard that displays various metrics including ticket sales value, attendee registrations, and feedback summaries.

7.1.2. Attendees:

1. Browse upcoming events based on type, date, or location.
2. Register for events and buy their passes via an easy checkout procedure.
3. Access event details such as schedule, maps, and organizer contact information from their user accounts.

7.1.3. Admin Users:

1. Oversee all events in the system, manage user accounts, and generate analytical reports.
2. Moderate content and resolve disputes or technical issues raised by users.

Feedback is totally merged into the system. Confirmation is given after successful operations-for example, confirmation in ticket purchase- and error conditions are highlighted with messages redirecting the user to the problem source. The interface supports accessibility features including variable font sizing and responsiveness on mobile and desktop platforms.

7.2 Screen Images Descriptions

Now here are descriptions regarding important screens. Actual images can then be created or sketched to represent the concept:

7.2.1. Login Page

1. Fields: Username, Password and Login button.
2. Actions: Login, Forgot Password and Sign-Up.

7.2.2. Dashboard (Organizer):

1. Event List: Shows all events that have been created by the organizer.
2. Statistics: Show ticket sales, revenue, and attendance.
3. Action Buttons: Create Event, Edit Event, and Delete Event.

7.2.3. Event Browsing Page (Attendee):

1. Search Bar: Filter events by keywords, location, or date.
2. Event Cards: Always show some of the crucial details of the event, such as "Information" or "Buy ticket" button to purchase ticket.

7.2.4. Ticket Purchase Page (Attendee):

1. Fields: Types of ticket and how many tickets per buyer.
2. Action: Proceed to Checkout and Cancel.

7.2.5. Admin Control Panel:

1. User Management: List of users with an option to add, edit, or deactivate accounts.
2. Event Moderation: Approve, edit, or remove events.
3. Reports: Generate usage and revenue statistics.

7.3 Screens: Objects and Actions

7.3.1. Login fields:

1. Objects: Username field and Password field; Login button.
2. Actions: Entering credentials, submitting login request, navigating to sign up or password recovery.

7.3.2. Event cards:

1. Objects: Event title, event date, event location, image, and details button.
2. Actions: Click to view the full information of the event or buy tickets.

7.3.3. Dashboard metrics:

1. Objects: Graphs, charts, and number figures for ticket sale and revenue.

2. Actions: Hover for more details, click to drill down to specific metrics.

7.3.4. Action buttons:

1. Objects: Create Event, Edit Event, Delete Event, Register, and Purchase.
2. Actions: Initiate the corresponding workflow on creating an event or completing a ticket purchase.

7.3.5. Navigation Menu:

1. Objects: Links for Home, Events, Profile, and Settings.
2. Actions: Navigation to the respective sections of the system.

8. Requirement Matrix

Hereinafter, the system components have been merged with the functional requirements as specified in the Software Requirement Specification (SWRS):

Requirement Traceability Matrix							
Project Name:				Event Management System			
Created By:				Group 2			
Date:				9th January, 2025			
Requirements	**Req desc**	**Req type**	**Test 1**	**Test 2**	**Test 3**	**Test 4**	**Test 5**
User Authentication	Registration	User	✔		✔		
Event Creation	Event Management	User		✔			
Ticket Purchase	Ticket Purchasing	Admin			✔		
Event Browsing	Event Browsing	Admin		✔		✔	
User Feedback	Feedback Module	Admin					✔
Report Generation	Admin Control Panel	Admin	✔			✔	
Mobile Accessibility	Platform Support	Admin			✔		

This matrix ensures a clear addressing and tracing of all functional requirements to the system design.

9. Conclusion:

TThe Event Management System is a strong, scalable, and user-centric platform that would streamline event planning and management. By using a layered architecture and by following the MVC design paradigm, the system makes sure that interaction between

its different components is seamless, which would result in a responsive and intuitive user experience for admins, organizers, and attendees.

The EMS is designed with modern technologies for the frontend, backend, and data storage, ensuring high performance, security, and adaptability. Features such as event scheduling, attendee management, and real-time notifications enhance efficiency, while secure payment processing and feedback collection ensure a holistic event experience.

The EMS's modular design and future-ready architecture would make it an indispensable event management tool capable of evolving according to the shifting needs of its users and emerging technologies.