A

Project Report on

Event Management System

By

Ms. Neha Thakare

MCA – I, SEM – II

2024-25

To

Savitribai Phule Pune University,Pune

In Partial Fulfillment of the Degree of

Master of Computer Application (M. C. A.)

Under The Guidance Of

Prof. Shilpa Deshmukh

Suryadatta Group of Institutes, Pune

Suryadatta Institute of Business Management and Technology

(SIBMT)

Date:-

CERTIFICATE

This is to certify that **Ms. Neha Mohan Thakare** has successfully completed her project work entitled **"Event Management System"** in partial fulfillment of MCA – I Semester-2 program for the year A.Y. 2024-25 She have worked under our guidance and direction.

Prof . Shilpa Deshmukh **Dr . Manisha Kumbhar.**

(Project Guide) **Professor & Director(SIBMT)**

Examiner 1 **Examiner 2**

_____ _____

Date :

Place :

Acknowledgment

We are the student of MCA first year. Here we express our thanks to our project guide for allowing us to do the project on‖ **Event Management System"** This project work has been the most exciting part of our learning experience which would be an asset for our future carrier. We would especially like to thank our guide and mentor **Prof. Shilpa Deshmukh** , who constantly guided us in developing, pushing us to search for more answers to her numerous questions. Also I would like to thanks to **Prof. Shilpa Deshmukh**, project coordinators for their support. As a building block of MCA Department, I thank **Dr. Manisha Kumbhar**, Professor & Director, MCA Department for her continuous support and help. We are grateful to many classmates who contributed their suggestions. Their hard work and examples push us to limits of our capability and encourage us daily.

Thank You

Student Name

Neha Mohan Thakare .

INDEX

INTRODUCTION

The "Event Management System" has been developed to override the problems prevailing in the practicing manual system. Events play an important role in our society. Any happening or an activity can be referred as an event.

• Individuals often find they lack the expertise and time to plan events themselves. Independent planners are needed to step in and give these special events the attention they deserve.

• This web application will maintain all the event booking and its records of the members, places, food, bills etc. It is made for recording the all-event booking done by the user. Thus, by this all it proves it is user-friendly.

• The application provides all facility for book any event. It is help to user to any event make successfully and also help to organizer to keep all records and maintains its.

• java-based web application which will run on localhost.

• An ―Event Management System project‖ is a solution that helps organizations and individuals plan, organize and execute different types of events such as Birthday, weeding and meetings. •

The system typically includes features for event booking, scheduling, and communication with attendees.

• The project generally includes the design, development, testing and implementation of the event management system.

• The scope and specific features of the project will depend on the requirements of the client or organization for which it is being developed.

• The goal of the event management system project is to automate event management processes and provide a seamless experience for organizers and user.

• Event management is a process of organizing a professional and focused event, for a particular target audience. It involves visualizing concepts, planning, budgeting, organizing and executing events such as wedding, musical concerts, corporate seminars, exhibitions, birthday celebrations, theme parties, etc.

• Event Management is a multi-million-dollar industry, growing rapidly, with events hosted regularly. Surprisingly, there is no formalized research conducted to access the growth of this industry. The industry includes fields such as the MICE (Meetings, Incentives and Events), exhibitions, conferences

and seminars as well as live music and sporting events.

• On the profession side, event management is a glamorous and exciting profession that demands a lot of hard work and dynamism.

• Event management is the application of project management to the creation and development of large-scale events.

• The process of planning and coordinating the event is usually referred to as event planning and which can include budgeting, scheduling, site selection, acquiring necessary permits, coordinating transportation and parking, arranging for speakers or entertainers, arranging decor, event security, catering, coordinating with third party vendors, and emergency plans.

• The events industry now includes events of all sizes from the Olympics down to business breakfast meetings. Many industries, charitable organizations, and interest groups hold events in order to market themselves, build business relationships, raise money, or celebrate achievement.

• An event refers to a social gathering or activity, such as a festival, (for example a musical festival), a ceremony (for example a marriage) and a party (for a birthday party)

Existing System

Before the development of specialized Event Management Systems, the organization and execution of events were largely handled through **manual or semi-automated methods**, often using general-purpose tools that were not specifically designed for end-to-end event management. While some modern tools do exist, they are often **fragmented, limited in scope, or not customizable** to the specific needs of every organizer .

Currently many event planners and organizations use a combination of:

- **Manual processes** (notebooks, Excel sheets, calls/emails)
- **Generic tools** like Google Forms, WhatsApp groups, or Eventbrite
- **Individual software** for ticketing, communication, or vendor coordination

These disjointed tools lack integration and are not tailored for comprehensive event management. For example:

- **Google Forms** handles registration but not ticketing or payments.
- **Spreadsheets** offer flexibility but are prone to human error and lack real-time updates.
- **Third-party platforms** like Eventbrite charge service fees and may not offer full customization or local vendor integration.

These existing systems do not provide a unified solution for event creation, task delegation, attendee management, and payment processing in one place. As a result, organizers struggle to manage everything efficiently and often face issues like:

- Miscommunication with vendors
- Double bookings or data entry errors
- Inaccurate reporting and budgeting
- Delayed responses and poor attendee engagement

1. Manual Event Management (Traditional Approach)

In many cases, events are still managed manually, which includes:

- Maintaining attendee lists in **Excel sheets**
- Collecting registration data via **Google Forms**
- Sending invitations and updates through **email or messaging apps**
- Handling ticket sales in **offline counters or over-the-phone**
- Recording vendor tasks and logistics in **notebooks or documents**

- Often targeted toward specific types of events (e.g., conferences)
- Premium features require subscription or high fees
- Limited flexibility in integrating local vendors or specific workflows

Need of System

Traditional event management often relies heavily on manual processes, such as maintaining spreadsheets for guest lists, phone calls for vendor coordination, and physical tickets for entry. These methods are time-consuming, error-prone, and lack real-time data access. As events grow in size and complexity, managing logistics, attendees, vendors, and finances becomes increasingly difficult without a centralized system.

The **Event Management System (EMS)** addresses these challenges by offering:

Automation of routine tasks like registration, ticketing, and communication.
Real-time updates and reporting for organizers and admins.
Role-based access, ensuring each user type (admin, organizer, vendor, attendee) can perform relevant
Secure transactions for ticket booking and payments.
Improved communication between organizers, vendors, and attendees.
Data-driven insights into event performance, attendance, and budgeting.

Thus, the system not only reduces manual effort but also enhances efficiency, accuracy, and the overall event experience.

The need for a centralized **Event Management System (EMS)** arises from the following critical challenges faced in manual or partially digitized systems:

1. **Coordination**
   Events involve multiple parties such as organizers, vendors, attendees, and sponsors. Coordinating their roles, timelines, and responsibilities manually increases the risk of miscommunication, delays, and resource conflicts.
2. **Scattered Data Sources**
   Using multiple tools (email, spreadsheets, forms, messaging apps) creates data silos, making it

difficult to get a unified view of registrations, finances, vendor statuses, and feedback. EMS integrates all components into a single platform.

3. **Time and Resource Efficiency**

   Manual handling of registrations, ticketing, and reporting consumes significant time and labor. EMS automates repetitive tasks, reducing manpower requirements and freeing up organizers to focus on strategic planning.

4. **Real-Time Information Access**

   In dynamic event environments, real-time updates are crucial—for example, attendee count, payment confirmation, or vendor task status. EMS ensures instant access to live data, enabling timely decisions.

5. **Scalability**

   As the scale of events increases (e.g., concerts or expos), manual systems fail to keep up. EMS is designed to handle small private events as well as large public gatherings with thousands of participants efficiently.

6. **Financial Transparency and Security**

   Manual payment tracking can lead to errors and potential fraud. EMS integrates secure payment gateways and maintains transaction histories for transparency, budgeting, and auditing.

7. **Improved Attendee Experience**

   With features like online booking, automated reminders, e-tickets, and digital check-ins, EMS significantly improves the attendee journey and satisfaction, enhancing the event's reputation.

8. **Data-Driven Decision Making**

   Event success depends on accurate insights. EMS provides organizers with analytics and reports on attendance, revenue, engagement, and feedback to help improve future events.

9. **Adaptability to Different Event Types**

   Whether it's a wedding, a corporate seminar, or a music festival, EMS can be customized to fit the unique workflows and needs of each type of event.

10. **Pandemic-Driven Digital Shift**

    The COVID-19 pandemic accelerated the need for contactless, remote-friendly, and hybrid event solutions. EMS supports virtual and hybrid models with digital engagement tools and safety protocols.

Operating Environment Hardware and Software

Operating Environment

The **Event Management System** is designed to function efficiently in a standard computing environment, ensuring compatibility with commonly available hardware and software platforms. The system supports both web-based and desktop deployment, making it accessible across a range of devices.

Hardware Environment

The hardware requirements for running the system are minimal and suitable for general-purpose computing environments. The specifications are as follows:

Software Environment

The software environment includes the operating system, programming languages, and supporting libraries or tools used for system development and deployment.

HARDWARE CONFIGURATION

- **Processor : AMD A4 –APU with Radeon™**
- **RAM : 4.00 GB**
- **Hard Disk : 2.50 GHz**

SOFTWARE CONFIGURATION

- **Operating System : Windows 10 Pro**
- **Backend Language : Python**
- **Frontend Language : HTML, CSS.**
- **Database : MySql, Sqlite.**
- **Connection : Sqlite3**

Purpose

The purpose of the Event Management System **is to provide an efficient and user-friendly platform for planning, organizing, and managing events of various types, such as conferences, seminars, weddings, corporate meetings, and social gatherings. This system aims to streamline event-related tasks, including venue selection, guest management, scheduling, ticket booking, and resource allocation.**

The purpose of the Event Management System **is to provide an efficient and user-friendly platform for planning, organizing, and managing events of various types, such as conferences,** seminars, weddings, corporate meetings, and social gatherings. This system aims to streamline event-related tasks, including venue selection, guest management, scheduling, ticket booking, and resource allocation.

The key objectives of the **Event Management System** are:

1. **Automation** – To automate manual event planning processes, reducing human effort and errors.
2. **Efficiency** – To improve the efficiency of managing event-related data, registrations, and logistics.
3. **User Convenience** – To offer an intuitive interface for organizers, vendors, and attendees.
4. **Real-time Updates** – To provide real-time tracking of event status, registrations, and resource availability.
5. **Cost-effectiveness** – To reduce operational costs by digitizing event management workflows.
6. **Scalability** – To support various event sizes, from small gatherings to large-scale conferences.
7. **Secure Data Handling** – To ensure the secure management of user data, payments, and event-related information

Module Specification Scope

The **Event Management System** is a web-based or desktop application designed to simplify and automate the event planning and management process. It is intended for event organizers, attendees, vendors, and other stakeholders involved in various event types, including corporate conferences, weddings, seminars, concerts, and exhibitions. The system ensures efficient event execution by providing tools for scheduling, ticketing, resource allocation, guest management, and financial transactions.

System Scope

1. User Roles & Functionalities

The system will cater to multiple user roles, each with specific functionalities:

- **Admin**:
  - Manage users (organizers, vendors, attendees).
  - Monitor event activities and system performance.
  - Handle payment transactions and security controls.
- **Event Organizers**:
  - Create, edit, and manage event details.
  - Set schedules, venues, and themes.
  - Handle registrations and guest lists.
  - Assign tasks to vendors (catering, logistics, etc.).
  - Generate reports on ticket sales, attendance, and budget.
-
  - Browse available events and register.
  - Book tickets and make online payments.
  - Receive event notifications and reminders.
- **Vendors (Catering, Equipment, etc.)**:

- Receive service requests from organizers.
- Manage event-related tasks and updates.
- Communicate with event planners.

2. Features & Functional Modules

- **Event Creation & Management**:
- Organizers can create and manage events with details like date, time, venue, and category.
- Option to define event types (free, paid, invite-only).
- **User Registration & Authentication**:
- Secure sign-up and login for different user roles.
- User profile management (organizers, attendees, vendors).
- **Ticket Booking & Payment Integration**

User Management Module

The **Event Management System** consists of several key **modules** that handle different functionalities. Below is a breakdown of the modules along with their respective outputs.

Description:

- Allows users (Admin, Organizer, and Attendee) to **register and log in**.
- Maintains **user roles** (Admin, Organizer, User, Vendor).

Functionalities:

- User Registration & Login
- Profile Management
- Role-based Access Control

Output:

- **User Dashboard** (for each role)
- **Login Confirmation Message**
- **User Profile Details**

1.2. Event Management Module

Description:

- Enables **Organizers** to **create, update, and manage** events.
- Allows **Users** to **browse and view** events.

Functionalities:

- Add/Edit/Delete Events
- View Event Details
- Search & Filter Events

Output:

- **Event Details Page**
- **List of Upcoming Events**
- **Event Confirmation Message (for Organizers)**

1.3. Venue Management Module

Description:

- Organizers can **add and manage** venues for events.
- Users can **view venue details** before booking.

Functionalities:

- Add/Edit Venue Details
- View Available Venues

Output:

- **Venue Details Page**
- **Venue Booking Confirmation**

1.4. Ticket Booking Module

Description:

- Allows users to **book tickets** for an event.
- Ensures ticket availability and **updates seat count**.

Functionalities:

- Book Event Tickets
- View Booking Details
- Cancel Booking (if needed)

Output:

- **Ticket Booking Confirmation Page**
- **E-Ticket Generation (PDF or QR Code)**
- **Booking Status Update (Booked/Canceled)**

Booking Management Module

Description:

- Allows users to **view and manage** their bookings.
- Organizers can **check attendee lists**.

Functionalities:

- View Booking History
- Cancel or Modify Bookings
- Check Booking Status

Output:

- **Booking Confirmation or Cancellation Message**
- **List of Booked Events for Users**
- **Attendee List for Organizers**

1.7. Feedback & Review Module

Description:

- Attendees can **rate and review** events.
- Organizers can **view feedback** to improve future events.

Functionalities:

- Submit Ratings & Reviews
- View Feedback Reports

Output:

- **User Feedback Section**
- **Event Rating Summary**
- **Organizer Report on Event Reviews**

Objectives of the Event Management System

The **Event Management System** is designed to enhance the efficiency, accuracy, and convenience of organizing and managing events. The key objectives of this system include:

The primary objective of the **Event Management System** is to provide a centralized, user-friendly platform to streamline the planning, coordination, and execution of various types of events. The system aims to improve efficiency, reduce manual workload, enhance user experience, and ensure seamless communication between all stakeholders.

1. Automation of Event Planning

- Minimize manual efforts by automating event scheduling, attendee registration, and ticketing.
- Provide an intuitive dashboard for event organizers to manage multiple events.

2. Efficient User & Role Management

- Allow different user roles: **Admin, Event Organizers, Attendees, and Vendors.**
- Enable secure login and role-based access control.

3. Seamless Event Registration & Ticketing

- Allow users to register for events and book tickets online.
- Support multiple payment methods for ticket purchases.
- Generate e-tickets with QR codes for easy check-in.

4. Vendor & Resource Management

- Help organizers assign and manage vendors (catering, audio-visual, logistics, etc.).
- Enable seamless communication between organizers and service providers.

5. Real-Time Notifications & Updates

- Send automated event reminders via **email and SMS**.
- Notify users of schedule changes, cancellations, or important announcements.

6. Data Security & Access Control

- Protect sensitive data (user information, payments) through **encryption and authentication**.
- Ensure secure database access to prevent unauthorized modifications.

7. Reporting & Analytics

- Generate **event performance reports** including attendance, revenue, and feedback.
- Provide insights for future event improvements based on attendee behavior.

8. User-Friendly Experience

- Offer a responsive and intuitive interface for users of all technical backgrounds.
- Implement **search & filter options** for event discovery.

9. Scalability & Multi-Event Support

- Support a variety of events, from **small meetings** to **large conferences**.
- Enable event organizers to manage multiple events simultaneously.

10. Cost-Effectiveness

- Reduce operational costs by replacing paper-based processes with a **digital system**.
- Optimize resource allocation to avoid unnecessary expenses.

Proposed System

The **Proposed Event Management System** is a comprehensive, user-friendly, and role-based application designed to automate and streamline the entire event planning and execution lifecycle. Unlike the existing manual or fragmented digital systems, the proposed system offers an **integrated solution** that combines event creation, user management, ticket booking, payment processing, vendor coordination, and reporting in a single platform.

Key Features of the Proposed System :

Role-Based Access Control:

Separate login panels for **Admins**, **Organizers**, **Attendees**, and **Vendors**.

Each role gets access to specific modules based on their responsibilities.

Event Creation & Management:

Organizers can create and manage events with full details like name, date, time, venue, type (paid/free/invite-only), and description.

Ability to update or cancel events and notify users automatically.

User Registration & Authentication:

Secure sign-up and login for all user types.

Password protection, profile updates, and role verification features.

Ticket Booking & Payment Gateway:

Attendees can view available events, select tickets, and make online payments.

Integration with digital payment systems (UPI, debit/credit cards, etc.).

Vendor Management Module:

Organizers can assign tasks to registered vendors (catering, sound, décor, etc.).

Vendors receive notifications, update task status, and coordinate directly with organizers.

Notifications & Reminders:

Automated emails or SMS alerts for event updates, booking confirmations, or schedule changes.

Report Generation:

Admins and Organizers can generate reports for ticket sales, budget tracking, user activity, and event

Benefits of the Proposed System

Eliminates manual workload and reduces errors.

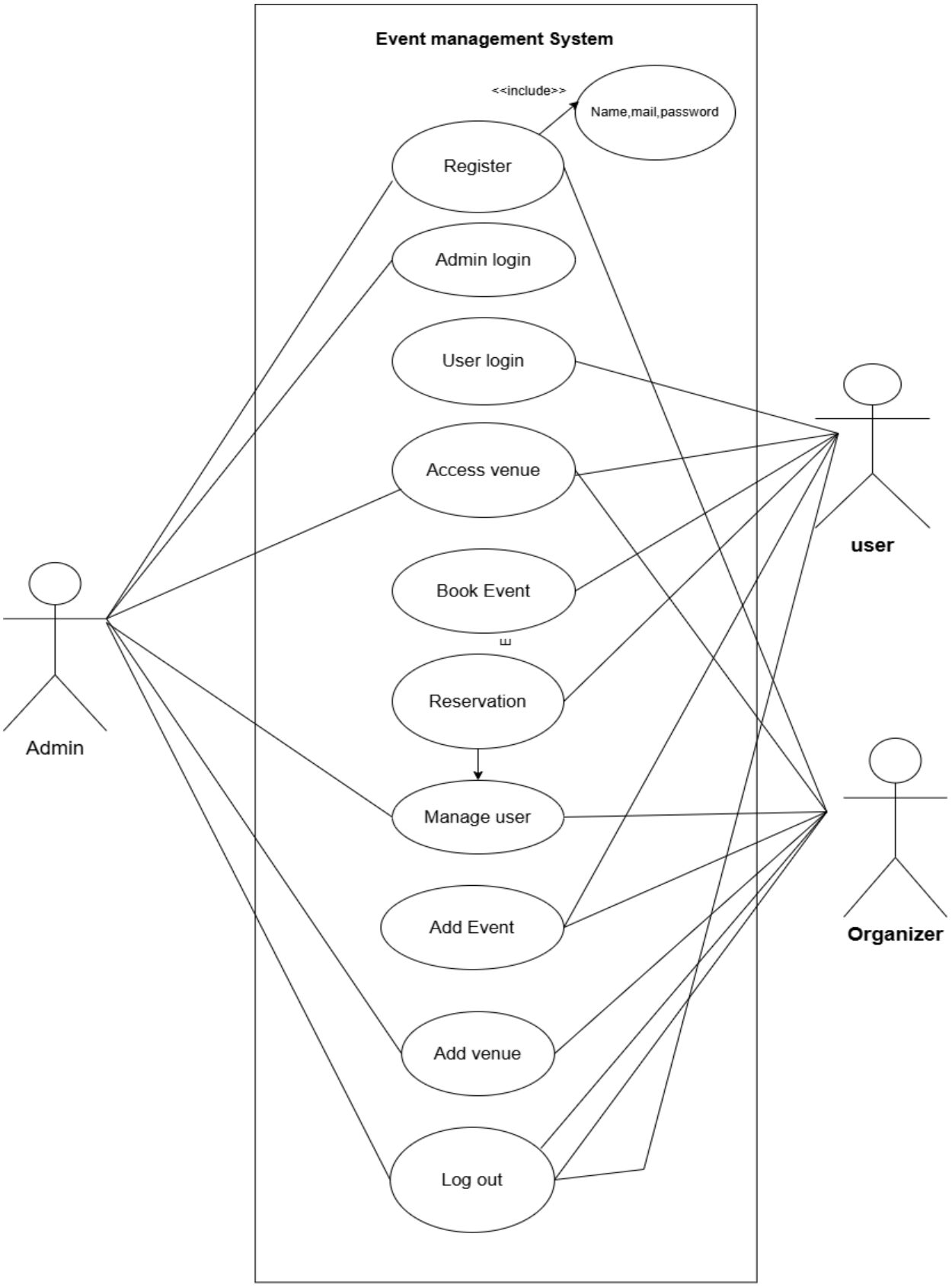
Offers real-time updates and data access.

Improves communication among all event participants.

Enhances overall event experience and satisfaction.

Scalable for any type or size of event—local, corporate, social, or academic.

Use case diagram



Use case Diagram Explanation

Actors Involved:

1. **Admin** – Manages users, events, and reservations.
2. **User (Attendee)** – Registers, logs in, books events, and accesses venues.
3. **Organizer** – Adds and manages events, venues, and bookings.

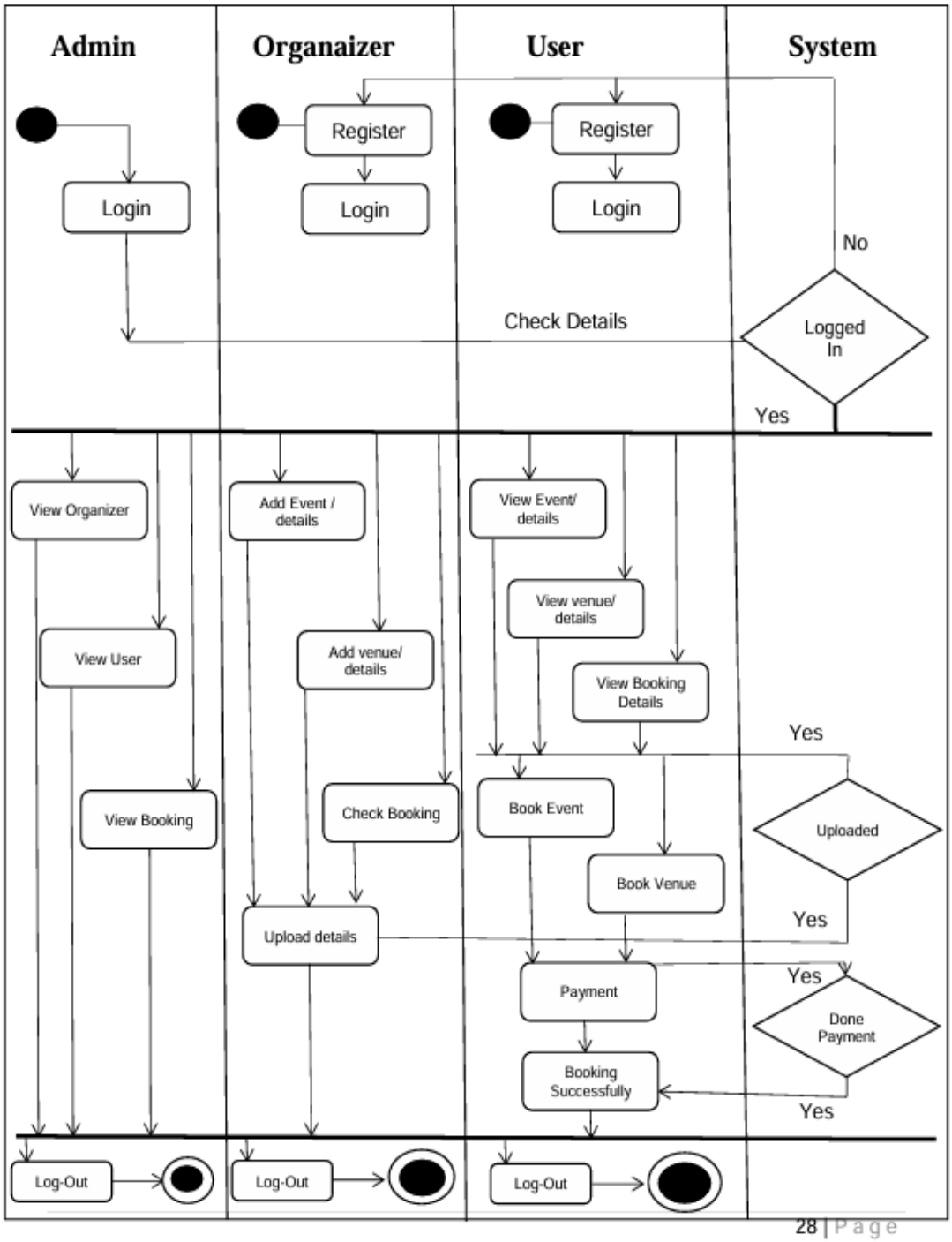
Relationships in the Diagram:

✔ **<<include>> Relationship**: The "Register" use case includes "Name, mail, password" since these details are essential for registration.

✔ **Arrows Represent Interactions**: Arrows connect each actor to their respective use cases, showing their interaction with the system.

Use Case	Description
Register	Users register with Name, Email, and Password.
Admin Login	Admin logs into the system.
User Login	Users log into their accounts.
Access Venue	Users browse event venues.
Book Event	Users book an event.
Reservation	Users confirm event reservations.
Manage Users	Admin manages registered users.
Add Event	Organizers add new events.
Add Venue	Organizers add venue details.
Log Out	Users/Admin can log out from the system.

3.5 Activity Diagram



Activity Diagram Functionality

Admin

- **Login** to the system.
- View different entities:
  - **View Organizer**
  - **View User**
  - **View Booking**
- **Logout** after performing tasks.

Organizer

- **Register** and **Login**.
- **Add Event Details** and **Add Venue Details** to the system.
- **Check Booking** made by users.
- **Upload Details** for events and venues.
- **Logout** after managing the system.

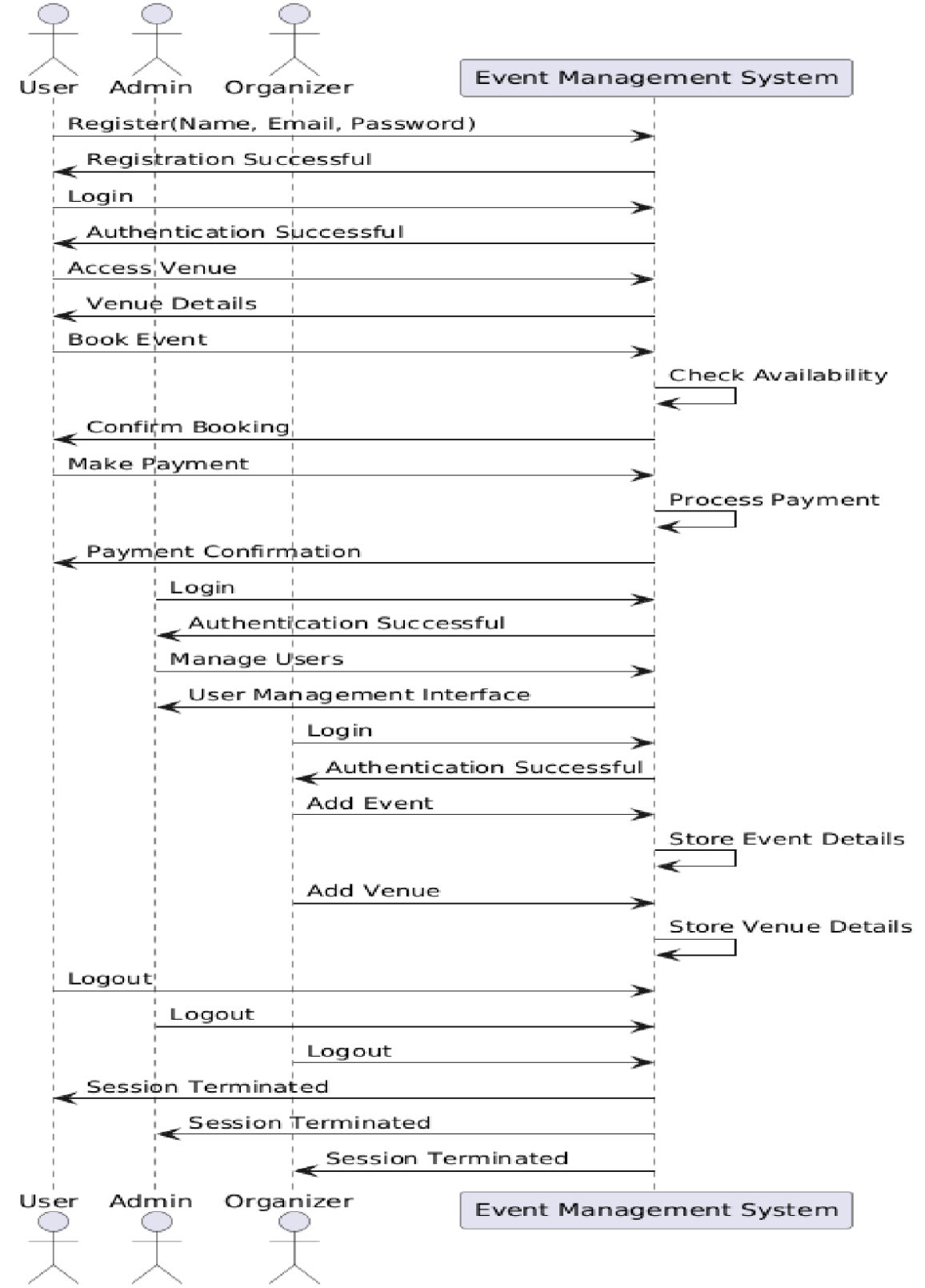
User

- **Register** and **Login**.
- **View Event and Venue Details** before booking.
- **View Booking Details** for their reservations.
- **Book Event** or **Book Venue**.
- **Make Payment** after booking.
- **Receive Booking Confirmation** after a successful payment.
- **Logout** after completing tasks.

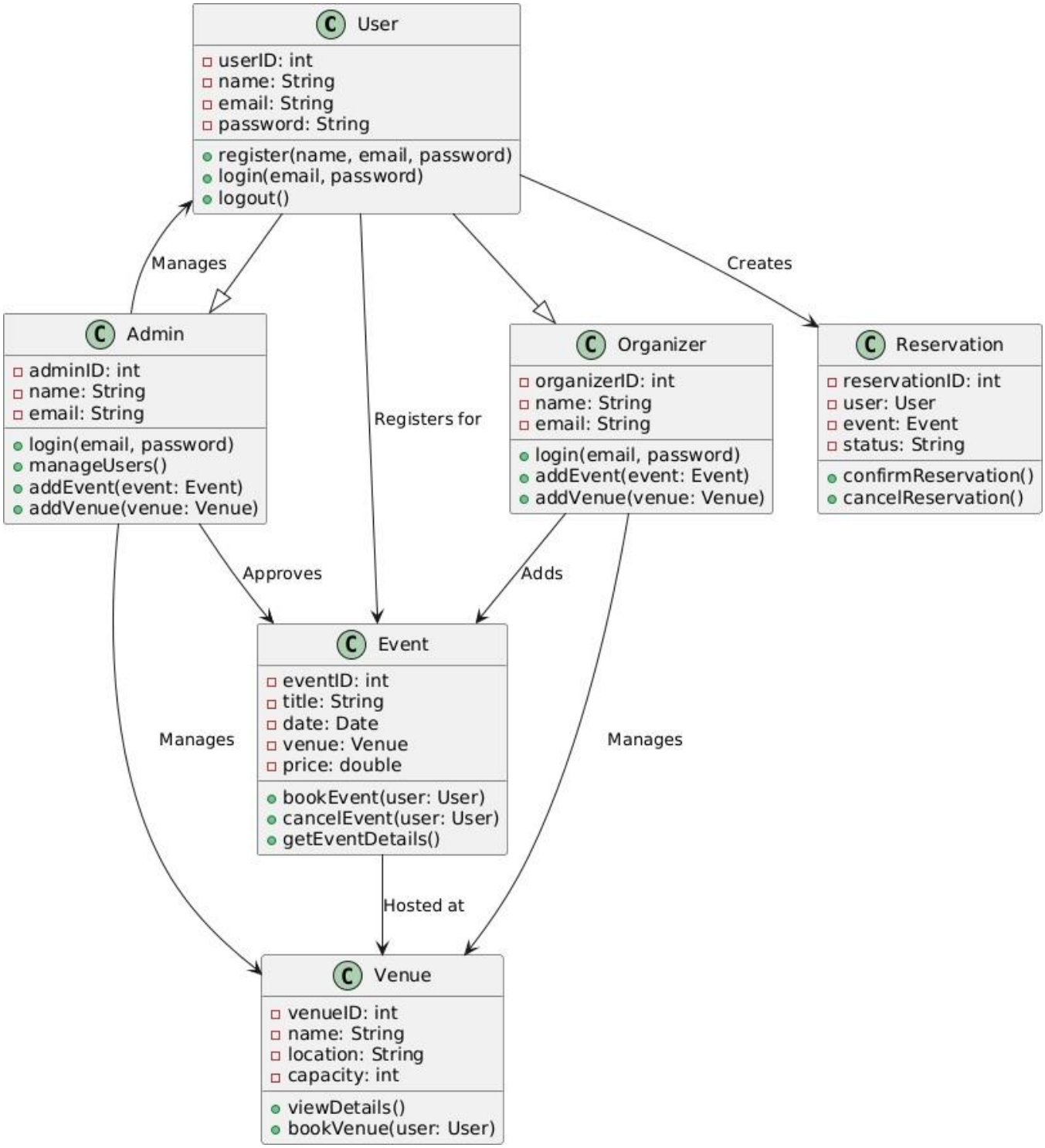
System

- Checks if the **user is logged in**.
  - If **not logged in**, the process stops.
- Ensures event/venue details are **uploaded** before booking.

Sequence Diagram:



Class Daigram:



Class Diagram functions

User Class

Represents general users who register and book events.

Attributes:

`userID`: Unique ID for the user.

`name`, `email`, `password`: Stores user details.

Methods:

`register(name, email, password)`: Registers a user.

`login(email, password)`: Logs in the user.

`logout()`: Logs out the user.

□ Admin Class (inherits from User)

Represents an administrator with extra privileges.

Attributes:

`adminID`: Unique identifier for the admin.

`name`, `email`: Stores admin details.

Methods:

`login(email, password)`: Logs in the admin.

`manageUsers()`: Manages system users.

`addEvent(event: Event)`: Adds an event.

`addVenue(venue: Venue)`: Adds a venue.

□ Organizer Class (inherits from User)

Represents event organizers who can create and manage events.

Attributes:

`organizerID`: Unique identifier for organizers.

`name`, `email`: Stores organizer details.

Methods:

`login(email, password)`: Logs in the organizer.

`addEvent(event: Event)`: Organizers can add events.

`addVenue(venue: Venue)`: Organizers can manage venues.

□ Event Class

Represents events available for booking.

Attributes:

`eventID`: Unique event identifier.

`title`, `date`, `venue`, `price`: Event details.

Methods:

`bookEvent(user: User)`: Allows users to book events.

`cancelEvent(user: User)`: Allows users to cancel bookings.

`getEventDetails()`: Retrieves event information.

□ Reservation Class

Handles event reservations.

Attributes:

`reservationID`: Unique reservation identifier.

`user`: Links reservation to a user.

`event`: Links reservation to an event.

`status`: Stores the booking status.

Methods:

`confirmReservation()`: Confirms a reservation.

`cancelReservation()`: Cancels a reservation.

Module Hierarchy Diagram

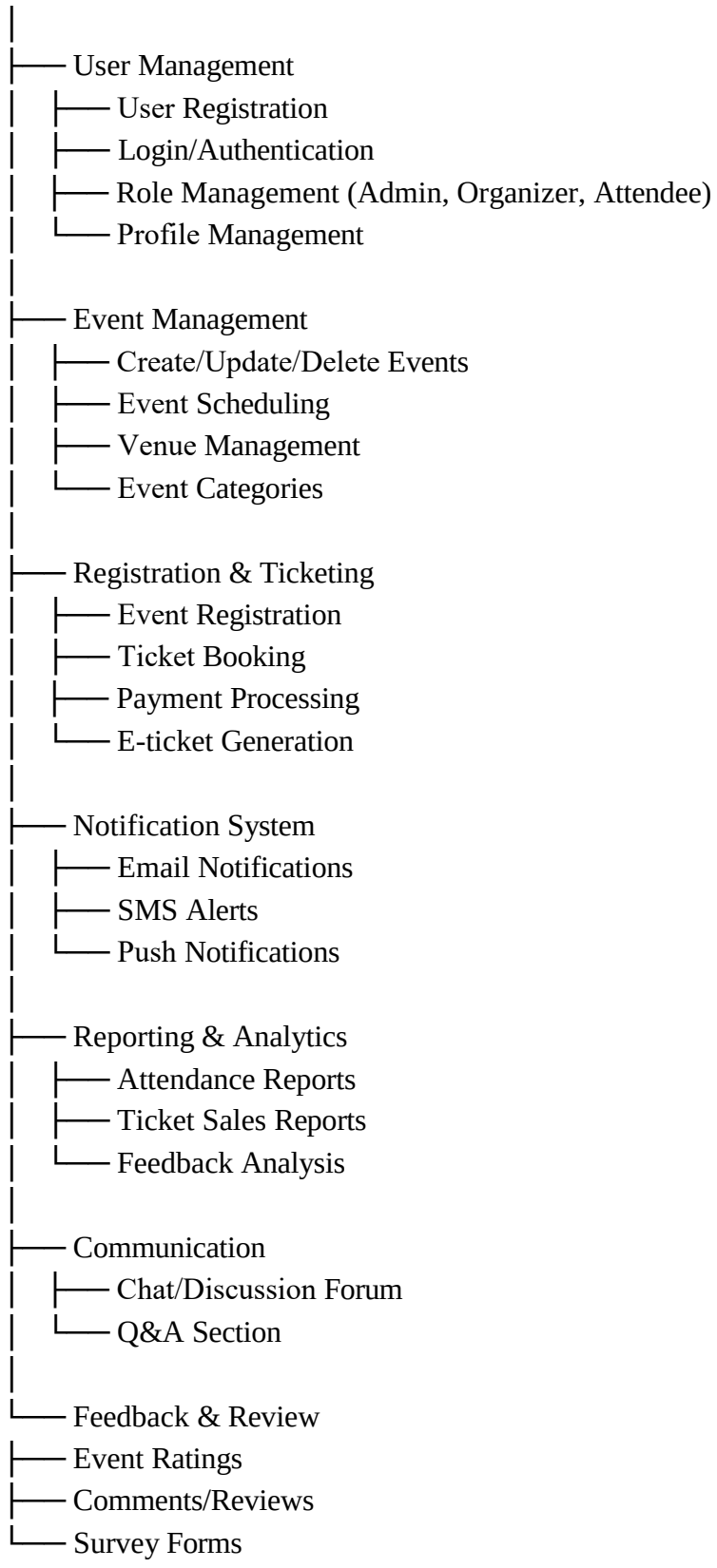


Table specifications (Database design)

1. Users Table

Stores information about individual events.

Field Name	DataType	Datasize	Constriants	Description
User_id	Integer	60	Primary key, auto increment	Unique id for each other
Name	Varchar	100	Not null	Full name of the user
Email	Varchar	60	Unique, not null	Email address
Password	Varchar	40	Not null	Encrypted password
Phone number	Varchar	20	Null	User's contact number
Role	Enum('organizer', 'attendee','admin')	50	Default 'attendee'	Role(e.g., admin, organizer, guest)
Created_at	Datetime	60	Default current timestamp	Account creation timestamp

2. Events Table

Stores information about individual events.

Field Name	Data Type	Data size	Constriants	Description
Event_ d	Integer	30	Primary key	Unique id for each event
Organizer_ d	Integer	20	Foreign key users(user id)	User id of the organizer
Title	Varchar	50	Not null	Event title
Description	Varchar	40	Null	Detailed description
Location	Varchar	60	Not null	Venue address
Category	Varchar	30	Null	Event Category(e.g. wedding, concert)
Status	Varchar	50	Default	Status(e.g. upcoming, completed)

3. **Venues Table**

Holds data on locations where events are held.

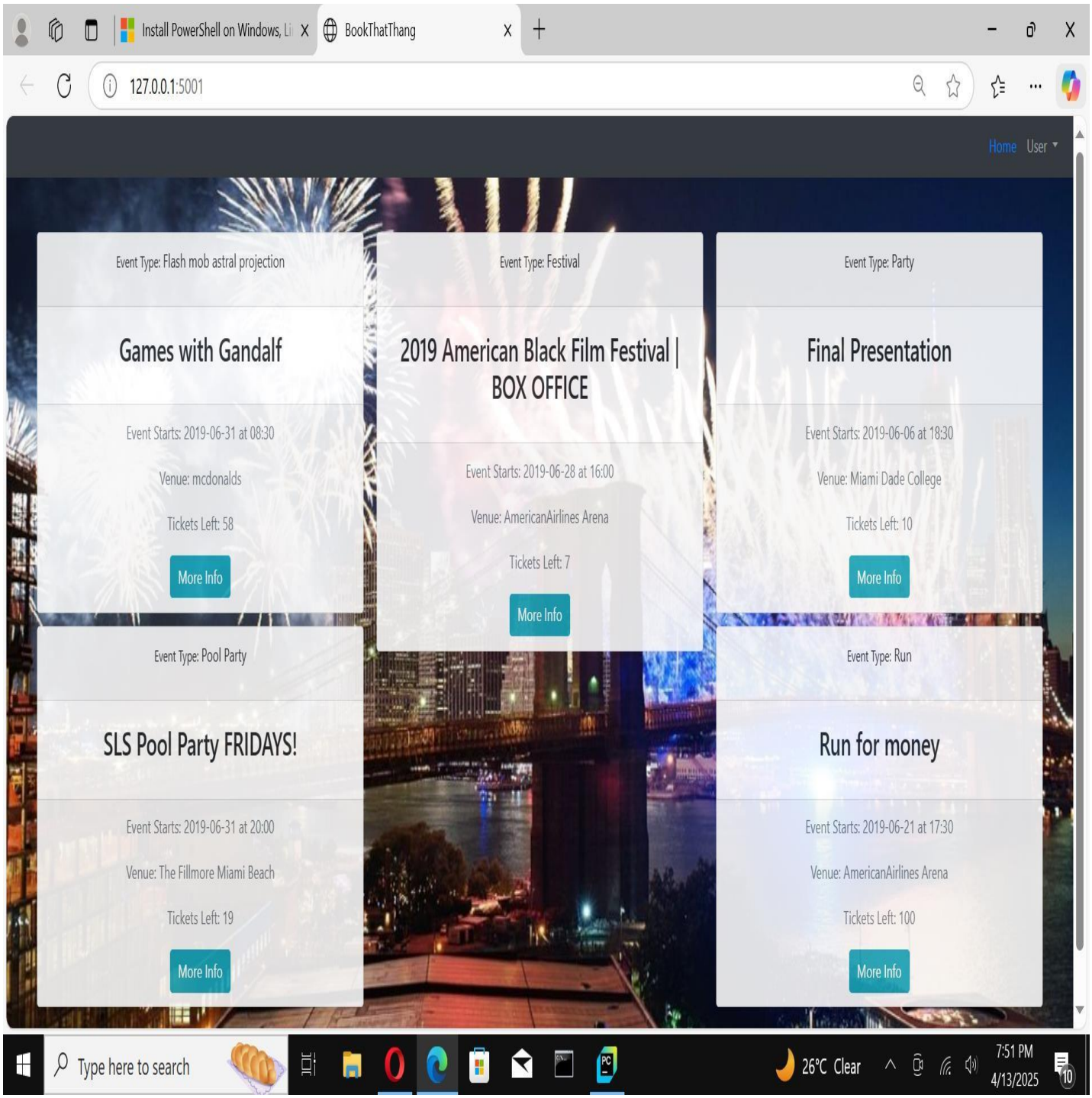
Field Name	Data Type	Data size	Constratints	Description
Venue id	Integer	30	Primary key	Unique id for each venue
Name	Varchar	150	Not null	Venue name
Address	Varchar	50	Not null	Venue address
City	Varchar	50	Not null	Venue city
Capacity	Integer	40	Null	capacity

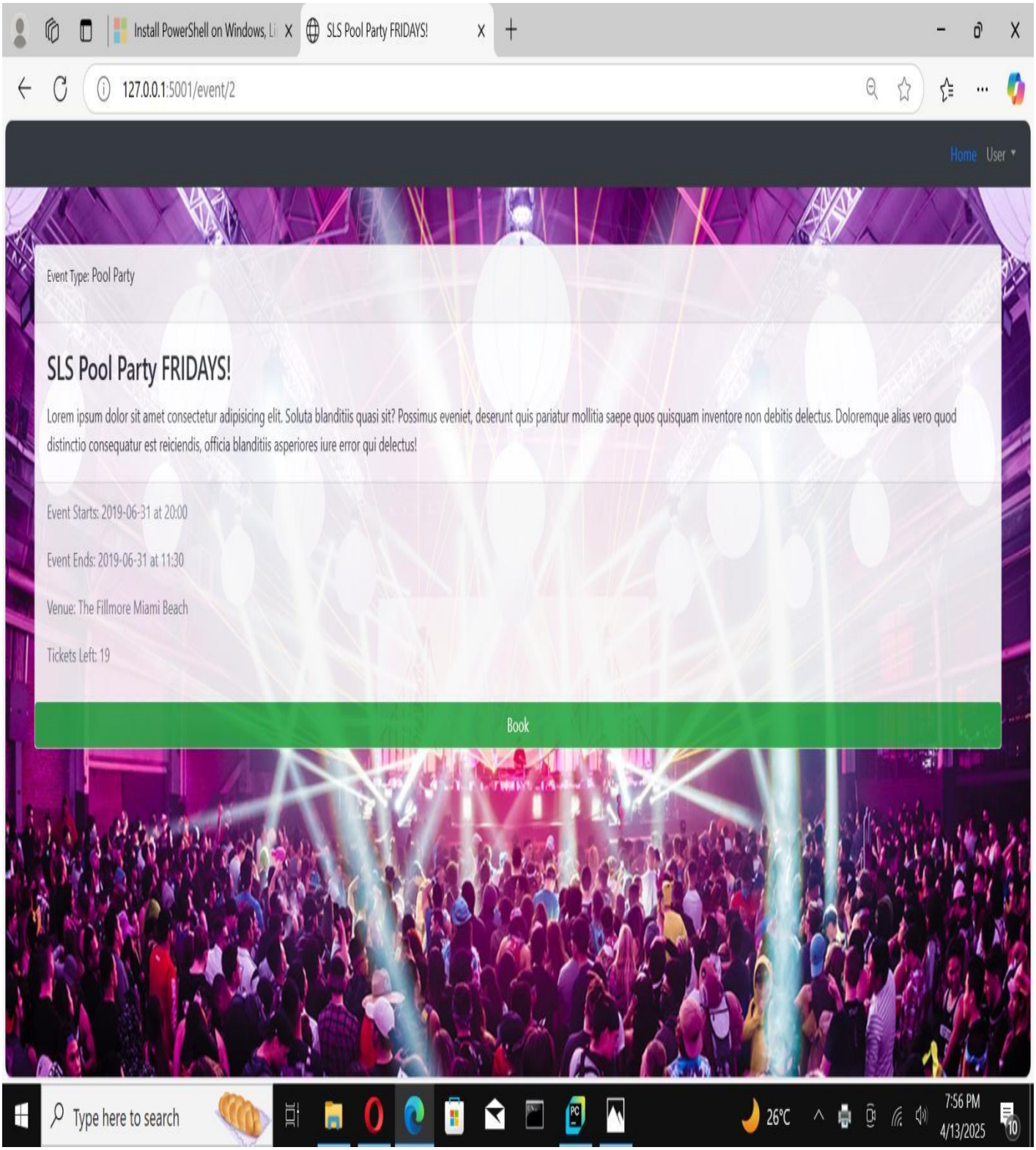
4. **Bookings Table**

Tracks event registrations by attendees.

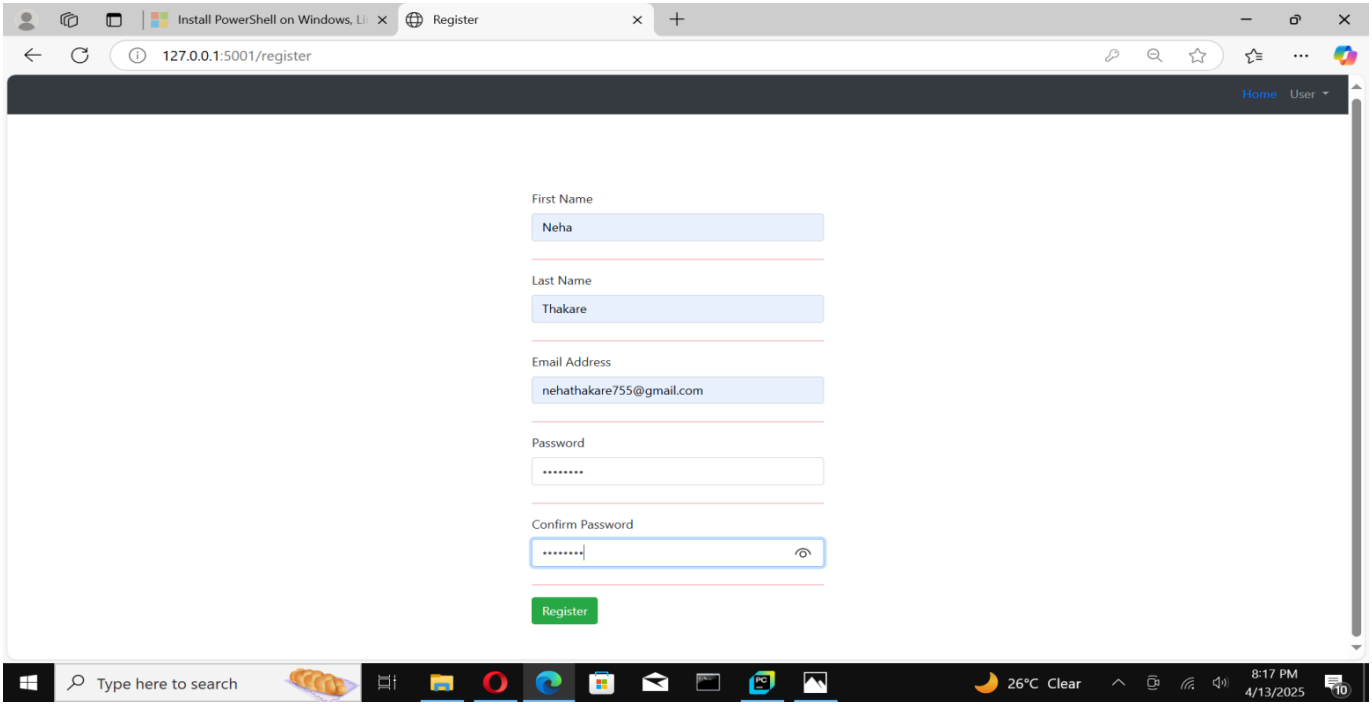
Field Name	Data Type	Data size	Constriants	Description
Booking id	Integer	30	Primary key, auto increment	Unique id for ticket
User id	Integer	40	Foreign key Users(user id)	User who registered
Event id	Integer	30	Foreign key Evet(event id)	Event being registered for
Booking date	Date	40	Default current timestamp	Date of registration
Status	Enum('confirmed' , 'cancelled')	50	Default 'confirmed'	Registration status(confirmed/cancelled)

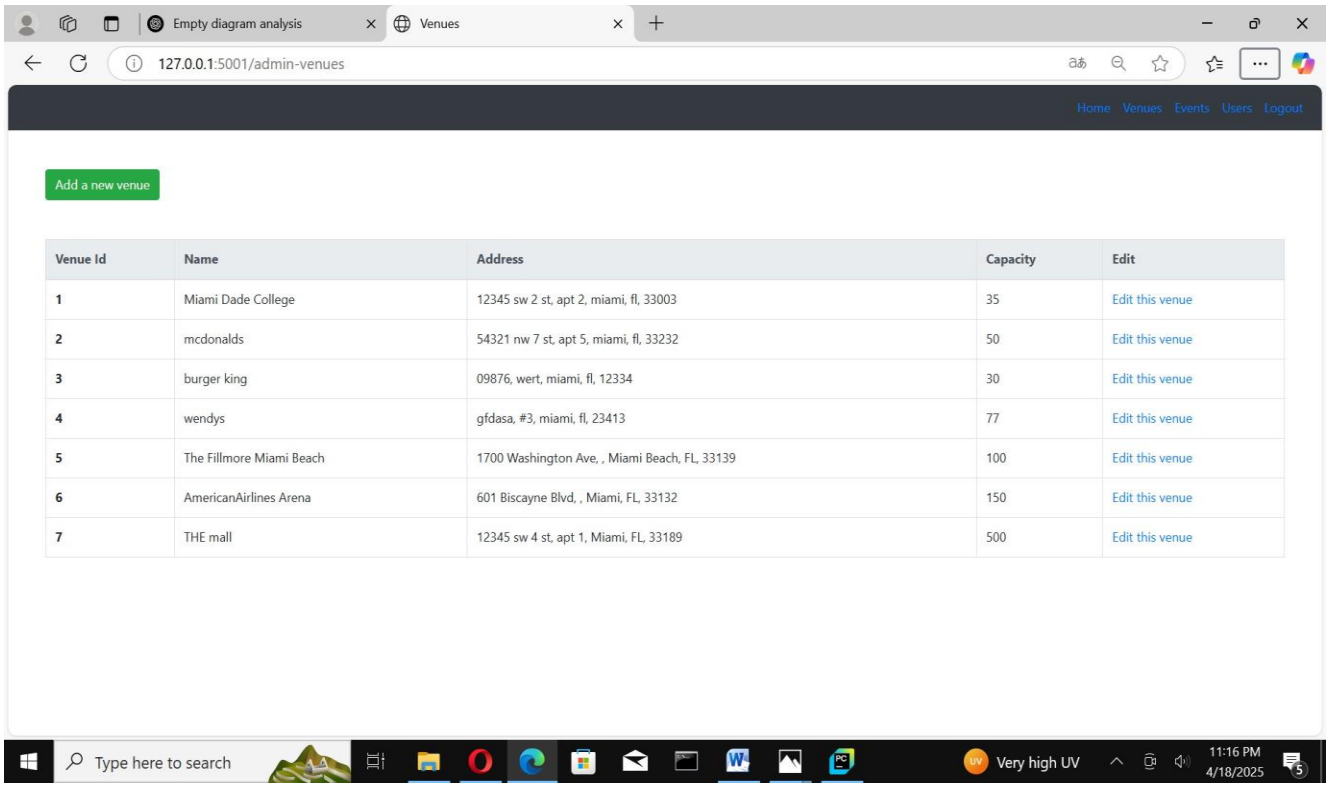
2.1 User Interface Screens

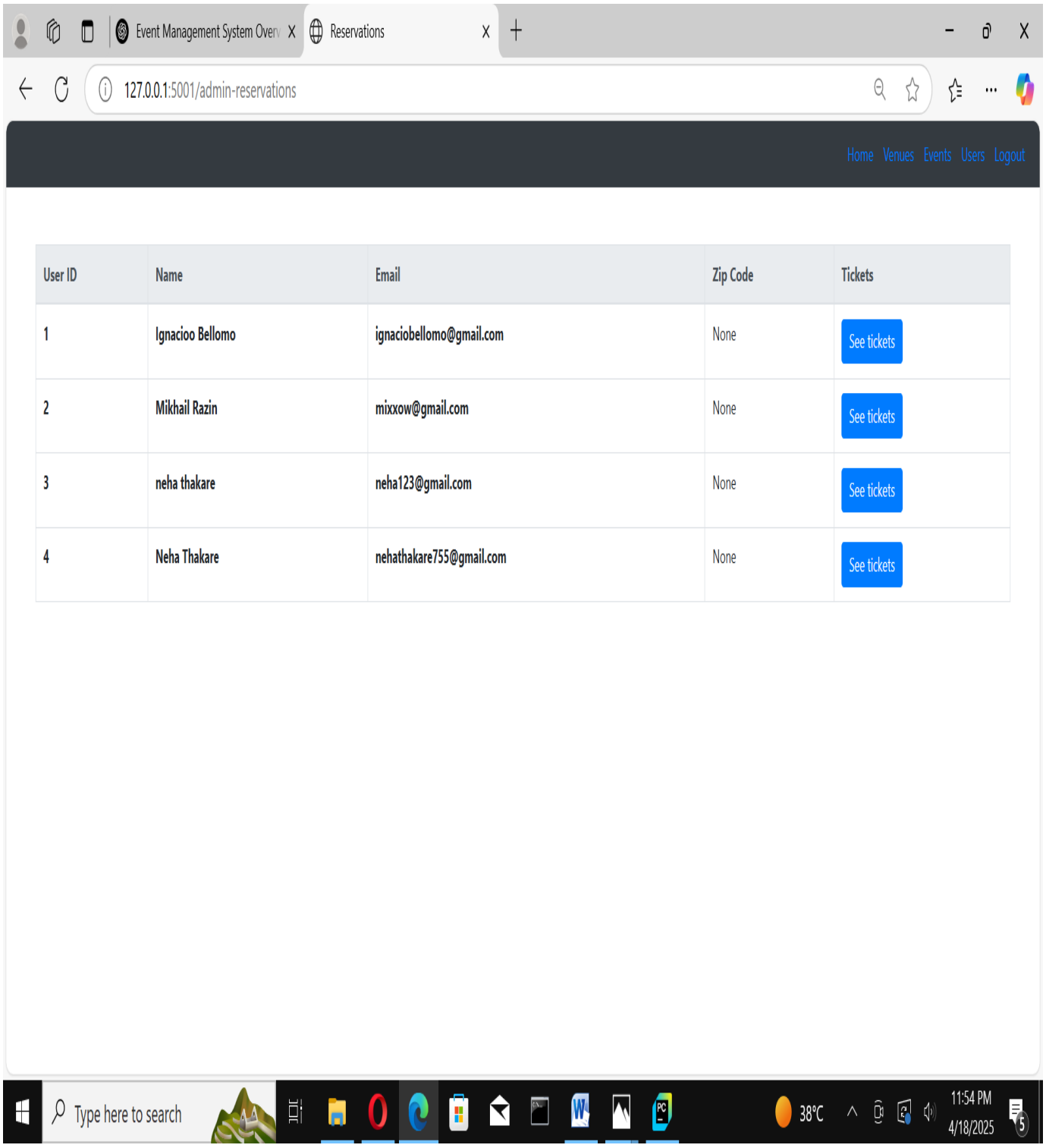



Install PowerShell on Windows, Li
SLS Pool Party FRIDAYS!
127.0.0.1:5001/event/2
Home
User
Event Type: Pool Party
SLS Pool Party FRIDAYS!
Lorem ipsum dolor sit amet consectetur adipisicing elit. Soluta blanditiis quasi sit? Possimus eveniet, deserunt quis pariatur mollitia saepe quos quisquam inventore non debitis delectus. Doloremque alias vero quod distinctio consequatur est reiciendis, officia blanditiis asperiores iure error qui delectus!
Event Starts: 2019-06-31 at 20:00
Event Ends: 2019-06-31 at 11:30
Venue: The Fillmore Miami Beach
Tickets Left: 19
Book
Type here to search
26°C
7:56 PM
4/13/2025

Register Admin Portal






Event Management System Overv
Reservations
127.0.0.1:5001/admin-reservations
Home Venues Events Users Logout
User ID
Name
Email
Zip Code
Tickets
1
Ignacioo Bellomo
ignaciobellomo@gmail.com
None
See tickets
2
Mikhail Razin
mixxow@gmail.com
None
See tickets
3
neha thakare
neha123@gmail.com
None
See tickets
4
Neha Thakare
nehathakare755@gmail.com
None
See tickets
Type here to search
38°C
11:54 PM
4/18/2025

Code Example

```
@app.route("/admin-login", methods=["GET", "POST"])
def admin_Login():
    """Login page for admin"""
    # Forget any user_id
    session.clear()
    # User reached route via POST (as by submitting a form via POST)
    if request.method == "POST":
      if not request.form["email"]:
          msg = "You didn't enter an email."
          return render_template("error.html", msg=msg)
        # Ensure password was submitted
        if not request.form["password"]:
          msg = "You must provide password!"
          return render_template("error.html", msg=msg)
        else:
          msg = "You must provide password!"
          return render_template("error.html", msg=msg)
         """Check password against hash using hash function"""
          email=request.form["email"]
```

```
        password=request.form["password"]
        verifyAdmin = db.execute(adminLogin, email=email)
        if len(verifyAdmin) != 1 or not check_password_hash(verifyAdmin[0]["pwdHash"], password):
            msg = "invalid username and/or password"
            return render_template("error.html", msg=msg)
        else:
            # Remember which user has logged in
            session["user_id"] = verifyAdmin[0]["adminID"]
            session["admin"] = "yes"
            # Redirect user to admin home page
            return redirect("/admin")
    else:
return render_template("admin-login.html")
@app.route("/admin", methods=["GET"])
@login_required
def admin():
    if not session["admin"]:
        msg = "You must be an admin to access that page. Please log in."
        return render_template("admin-login.html", msg=msg)
        events = db.execute(allEventQry)
```

```
@app.route("/admin-register", methods=["GET", "POST"])
def adminRegister():
    # code must be matched to register
    adminCode = "1001"
    if request.method == "POST":
        # Form validation
        if not request.form.get("adminCode"):
            msg = "You didn't enter an admin code."
            return render_template("error")
        if not request.form.get("email"):
            msg = "You didn't enter an email."
            return render_template("error.html", msg=msg)
        elif not request.form.get("password"):
            msg = "You didn't enter a password."
            return render_template("error.html", msg=msg)
        elif not request.form.get("confirmPassword"):
            msg = "You didn't confirm your password."
            return render_template("error.html", msg=msg)
        elif not passwordValid(request.form.get("password")):
            msg = "Password must contain at least 1 letter,1 number, and be at least 8 characters long."
```

```
    # User reached route via GET (as by clicking a link or via redirect)
    else:
        return render_template("admin-register.html")
@app.route("/admin-events", methods=["GET"])
                def adminEvents():
  events = db.execute(allEventQry)
  if events:
      return render_template("admin-events.html", events=events)
@app.route("/add-venue", methods=["GET", "POST"])
@login_required
def addVenue():
  if not session["admin"]:
      msg = "You must be an admin to access that page. Please log in."
      return render_template("admin-login.html", msg=msg)
  if request.method == "POST":
      if not request.form.get("venueName"):
          msg = "You did not enter a name for the venue."
          return render_template("error.html", msg=msg)
      elif not request.form.get("venueAddress1"):
          msg = "You didn't fill out the Address 1 line."
          return render_template("error.html", msg=msg)
      elif not request.form.get("venueAddress2"):
          msg = "You didn't fill out the Address 2 line."
          return render_template("error.html", msg=msg)
```

Data Report

This data report outlines the key data requirements, storage structures, and processing needs for the Event Management System (EMS) based on the provided class diagram. The EMS facilitates user registrations, event management, reservations, and venue management.

2. Data Requirements

2.1 User Data

- **userID** (int) - Unique identifier for users
- **name** (String) - Full name of the user
- **email** (String) - Contact email
- **password** (String) - Encrypted password for authentication

2.2 Admin Data

- **adminID** (int) - Unique identifier for admins
- **name** (String) - Full name of the admin
- **email** (String) - Contact email

2.3 Organizer Data

- **organizerID** (int) - Unique identifier for organizers
- **name** (String) - Full name of the organizer
- **email** (String) - Contact email

2.4 Event Data

- **eventID** (int) - Unique identifier for events

- **title** (String) - Name of the event
- **date** (Date) - Date of the event
- **venue** (Venue) - Venue hosting the event
- **price** (double) - Ticket price

2.5 Venue Data

- **venueID** (int) - Unique identifier for venues
- **name** (String) - Venue name
- **location** (String) - Physical location
- **capacity** (int) - Maximum number of attendees

2.6 Reservation Data

- **reservationID** (int) - Unique identifier for reservations
- **user** (User) - User making the reservation
- **event** (Event) - Event being reserved
- **status** (String) - Status of reservation (Confirmed, Pending, Canceled)

3. Data Storage

- **Database System**: MySQL/PostgreSQL
- **Tables**: Users, Admins, Organizers, Events, Venues, Reservations
- **Relationships**:
-
    - One-to-Many: An admin manages multiple users
    - One-to-Many: An organizer can create multiple events
    - One-to-Many: A venue can host multiple events
    - One-to-Many: A user can register for multiple events
    - One-to-One: A reservation links one user to one event

SYSTEM IMPLEMENTAION

Implementation is the stage in the project where the theoretical design is turned into a working system. The implementation phase constructs, installs and operates the new system. The most crucial stage in achieving a new successful system is that it will work efficiently and effectively.

There are several activities involved while implementing a new project.

- End user training
- End user Education
- Training on the application software
- System Design
- Parallel Run and To New System
- Post implementation Review
- End user Training:

The successful implementation of the new system will purely upon the involvement of the officers working in that department. The officers will be imparted the necessary training on the new technology

End User Education:

The education of the end user start after the implementation and testing is over. When the system is found to be more difficult to understand .

Limitations

1. Scalability Constraints

   The current system is designed for small to medium-sized events. It may not efficiently handle large-scale event operations with thousands of attendees or complex logistical needs.
2. Limited Payment Integration

   Only a few payment gateways are integrated. Users from certain regions may face difficulties completing transactions if their preferred method isn't supported.
3. Internet Dependency

   The system requires a stable internet connection. Offline functionality is limited or unavailable.
4. Customization Limitations

   Event themes, registration forms, and ticketing templates have limited customization options in the current version.
5. Security & Data Privacy

   Basic security measures are in place, but the system may not meet enterprise-level compliance standards such as GDPR or HIPAA without further enhancement.
6. User Training Requirements

   Users with low technical proficiency may require guidance to navigate the system, especially for backend/admin tasks.
7. No Real-time Communication

   The system does not currently support real-time chat or video conferencing features for virtual events.

Bibliography

- **Laudon, K. C., & Laudon, J. P. (2020).** ***Management Information Systems: Managing the Digital Firm***
- **(16th ed.). Pearson Education.**
- **Sommerville, I. (2016).** ***Software Engineering*** **(10th ed.). Pearson Education.**
- **Pressman, R. S., & Maxim, B. R. (2014).** ***Software Engineering: A Practitioner's Approach*** **(8th ed.). McGraw-Hill Education.**
- **IEEE Computer Society. (2014).** ***IEEE Std 830-1998 – IEEE Recommended Practice for Software Requirements Specifications*****. IEEE.**
- **W3Schools. (n.d.).** ***HTML, CSS, JavaScript Tutorials*****. Retrieved from https://www.w3schools.com Stripe API Documentation. (n.d.). Retrieved from https://stripe.com/docs/api**
- **OWASP Foundation. (n.d.).** ***OWASP Top Ten Web Application Security Risks*****. Retrieved from https://owasp.org/www-project-top-ten/**