QuickEvent Hub

A Project Report

Submitted in partial fulfilment of the

Requirements for the award of the Degree of

BACHELOR OF SCIENCE (INFORMATION TECHNOLOGY)

BY

Vedant Vikas Thorat

T.22.140

Under the esteemed guidance of

Mrs. Nutan Sawant.

DEPARTMENT OF INFORMATION TECHNOLOGY

SIES(NERUL) COLLEGE OF ARTS, SCIENCE & COMMERCE(AUTONOMOUS)

(Affiliated to University of Mumbai)

NAVI MUMBAI, 400706

MAHARASHTRA

2024

SIES(NERUL) COLLEGE OF ARTS, SCIENCE & COMMERCE(AUTONOMOUS)

(Affiliated to University of Mumbai)

NAVI MUMBAI MAHARASHTRA 400706

DEPARTMENT OF INFORMATION TECHNOLOGY

PROFORMA FOR THE APPROVAL PROJECT

PRN NO: 2022016401778022 **ROLL NO: T.22.140**

1. Name of the Student:
Mr. Vedant Vikas Thorat

2. Title of the Project:

QuickEvent Hub

3. Name of the Guide:
Mrs. Nutan Sawant.

4. Teaching experience of the Guide:

5. Is this your first submission? : Yes: ☐ **No:** ☐

Signature of the Student: **Signature of the Guide:**

Date: _____ **Date:** _____

Signature of The Coordinator:

Date: _____

CERTIFICATE

This is to certify that **Mr. VEDANT VIKAS THORAT** has worked and duly completed his/her Field Project Work for the partial completion of degree of Bachelor in Information Technology (BSC IT) under the Faculty of Information Technology in the subject of Field project and his/her project is entitled "**QUICKEVENT HUB**" under my supervision.

I further certify that the entire work has been done by the learner under my guidance and that no part of it has been submitted previously for any Degree or Diploma of any University.

It is his own work, and facts are reported by her personal findings and investigation.

Date of submission:

Internal examiner **External examiner**

SIES (Nerul) College of Arts, Science and Commerce NAAC Re-Accredited 'A' Grade (3rd Cycle) Sir Chandrasekarendra Saraswati Vidyapuram, Plot I-C, Sector 5, Nerul, Navi Mumbai, Maharashtra 400706.

DECLARATION BY LEARNER

I the undersigned **Mr. VEDANT VIKAS THORAT**, hereby declare that the work embodied in this project work titled **"QUICKEVENT HUB**", forms my own contribution to the research work carried out under the guidance of ***Mrs. Nutan Sawant***., is a result of my own research work and has not been previously submitted to any other University for any other Degree/ Diploma to this or any other University.

Wherever reference has been made to previous works of others, it has been clearly indicated as such and included in the bibliography.

I, hereby further declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct.

Name and Signature of the learner

ABSTRACT

QuickEvent Hub is a digital platform developed to simplify the organization of small-scale events at home by bridging the gap between event organizers and service providers. The core idea behind the application is to offer a marketplace where event organizers can easily rent items such as tables, chairs, and décor or book services like catering for intimate gatherings, birthday parties, family events, or small celebrations. On the other hand, service providers, including home caterers and event rental suppliers, can register and list their offerings, making them accessible to event organizers in their locality.

The platform operates with two primary types of users: the Event Organizer (who needs services) and the Service Provider (who offers services). These users are connected through the Admin, who manages requests, oversees bookings, and ensures smooth transactions between both parties. The system automates several critical processes, including user registration, service listings, booking requests, payment handling, and service confirmation.

A key objective of QuickEvent Hub is to focus on small event management, targeting specific needs like localized catering or renting equipment for home-based gatherings. Unlike traditional event management systems that cater to large-scale events, this platform prioritizes flexibility, ease of access, and cost-effectiveness for users planning more intimate events. The project includes developing an intuitive user interface, secure database integration for managing user and service information, and real-time booking and payment systems.

The platform's overarching purpose is to offer convenience for both event organizers and service providers, minimizing the effort involved in sourcing and booking event essentials. By streamlining the entire process, QuickEvent Hub aims to become the go-to platform for quick, reliable, and efficient service access, empowering users to host successful home events without the complexities of traditional event planning.

Ultimately, QuickEvent Hub addresses a market need by combining event service facilitation with real-time communication, offering a seamless and efficient solution for small-scale event management.

ACKNOWLEDGMENT

To list who all have helped me is difficult because they are so numerous and the depth is so enormous.

I would like to acknowledge the following as being idealistic channels and fresh dimensions in the completion of this project.

I take this opportunity to thank the University of Mumbai and College for giving me the chance to do this project.

I would like to thank my **Principal, Dr. Koel Roychoudhury** for providing the necessary facilities required for the completion of this project.

I take this opportunity to thank our **Dr. Meghna Bhatia Head of the department**, for her moral support and guidance.

I would also like to express my sincere gratitude towards my **project guide, *Ms. Meghna Bhatia*.** whose guidance and care made the project successful.

I would like to thank my College Library, for having provided various reference books and magazines related to my project.

I would like to thank to express my gratitude to

Lastly, I would like to thank each and every person who directly or indirectly helped me in the completion of the project especially my Parents and Peers who supported me throughout my project.

SIES (Nerul) College of Arts, Science and Commerce NAAC Re-Accredited 'A' Grade (3rd Cycle) Sir Chandrasekarendra Saraswati Vidyapuram, Plot I-C, Sector 5, Nerul, Navi Mumbai, Maharashtra 400706

Table of Contents

Chapter 1: Introduction

1. Background

The event management industry has seen substantial growth in recent years, but most platforms and services are primarily designed for large-scale events, leaving a significant gap in the market for smaller, more intimate gatherings. For individuals hosting personal events, such as birthday parties, family gatherings, or small-scale celebrations, organizing such events can still be daunting. From arranging the right equipment and décor to coordinating food, finding reliable and affordable resources on a small scale can be overwhelming.

The **QuickEvent Hub** project seeks to address this issue by providing a seamless and efficient platform for users to access all necessary services for organizing small home events. This application enables users to rent items and hire catering services with ease, helping them organize memorable events without the hassle of navigating multiple service providers. At the same time, service providers such as caterers and rental suppliers can register and offer their services, creating a marketplace that benefits both event planners and providers. QuickEvent Hub focuses on convenience, reliability, and cost-effectiveness, with an emphasis on catering to smaller-scale, home-based events.

2. Objective

The main objective of **QuickEvent Hub** is to create an integrated platform where two types of users can interact efficiently:

- **User Type 1**: Individuals who wish to organize small events in their homes by renting items like tables, chairs, decorations, or hiring catering services.
- **User Type 2**: Service providers, such as catering businesses and rental suppliers, who want to offer their services to users seeking event-related equipment or food services.

The platform aims to simplify the process of planning small events by:

1. Offering an accessible digital marketplace that connects users who want to organize events with relevant service providers.
2. Facilitating efficient communication between users and service providers to ensure that requirements are clearly conveyed and services are properly aligned.

3. Purpose

The **purpose** of QuickEvent Hub is to provide an organized, user-friendly platform for small event planning, catering to users who require event equipment rentals and catering services for home-based functions. With the following key objectives, the app provides benefits for both the users and service providers:

- **For Users**:
    1. **Convenience**: Users can quickly browse and select from a variety of rental options and food services, customized for smaller events, without needing to reach out to multiple vendors.
    2. **Flexibility**: The app allows users to select items and services based on their unique event requirements, providing multiple options for customizing their home event.
    3. **Cost-Effective**: The platform is designed for budget-friendly small events, avoiding the high costs of large event management services.
    4. **Transparency**: With user reviews, ratings, and pricing transparency, users can make informed decisions while selecting service providers.
- **For Service Providers**:
    1. **Wider Reach**: Catering businesses and rental providers can reach more customers, particularly those who would not otherwise know about their services.
    2. **Business Growth**: By using the platform, providers can increase their exposure and secure more bookings.
    3. **Efficient Management**: Providers can easily manage their bookings, update availability, and communicate with users via the platform.
- **For Admin**: The platform allows an administrator to manage requests, oversee service delivery, and address any issues that may arise, ensuring a smooth and satisfactory experience for both users and service providers.
    1.

4. Scope

The scope of **QuickEvent Hub** is to serve as a digital marketplace that supports:

1. **User Registration**: Allow users to sign up as either event planners or service providers (caterers or rental suppliers).
2. **Event Planning Services**: Users can browse and rent items such as tables, chairs, sound systems, decorations, or book catering services suitable for small gatherings.
3. **Service Provider Registration**: Catering businesses and rental providers can register and offer their services through the app.
4. **Admin Control**: The admin can monitor and manage all interactions, ensuring that user requests are matched with appropriate service providers and that any disputes or issues are resolved efficiently.
5. **Scalability**: Although the app is initially focused on small events, it has the potential to scale into larger event management solutions or expand into different geographic regions.

5. Applicability

The **QuickEvent Hub** platform is applicable to a variety of users, including:

1. **Home Event Organizers**: Individuals who want to arrange small-scale gatherings like birthdays, anniversaries, family get-togethers, or home-based parties can use the platform to rent items or hire catering services.
2. **Service Providers**: Local caterers and rental suppliers who want to expand their reach and simplify their service offering process will find the platform useful for connecting with potential customers.
3. **Communities**: Local communities or neighborhoods that regularly organize small events (e.g., potlucks, neighborhood parties) can use the app to streamline the logistics of organizing such gatherings.
4. **Freelancers and Small Businesses**: Freelancers, small event organizers, and part-time caterers can register on the platform and find business opportunities.

6. Achievements

The **QuickEvent Hub** project aims to achieve the following:

1. **Seamless User Experience**: Providing a user-friendly interface that simplifies the process of finding, booking, and managing event rentals and catering services.
2. **Efficient Administration**: Creating a streamlined system where the admin can effectively manage service providers and users, ensuring smooth and reliable interactions.
3. **Increased Visibility for Providers**: Offering rental and catering providers a broader audience, allowing them to showcase their services to more customers.
4. **Cost-Effective Solutions**: Offering users affordable solutions for small-scale event planning while avoiding the higher costs associated with larger event management platforms.

7. Organization of Report

This report is organized as follows:

- **Chapter 1**: Introduction to QuickEvent Hub, including the background, purpose, and objectives of the project.
- **Chapter 2**: Detailed analysis of user requirements and specifications for both event planners and service providers.
- **Chapter 3**: System architecture and design, outlining the platform's front-end, back-end, and database structure.
- **Chapter 4**: Development methodology, including the technologies and frameworks used to build the application.
- **Chapter 5**: Testing and validation of the platform, detailing the process of user acceptance testing and system reliability.
- **Chapter 6**: Conclusion, summarizing the project's outcomes, future prospects, and potential areas of improvement for further iterations.

Chapter 2: Survey of technologies

1. Front-End Technologies

The front-end of QuickEvent Hub is designed using JSP (JavaServer Pages), CSS (Cascading Style Sheets), and JavaScript. These technologies ensure an interactive and user-friendly interface, crucial for engaging users effectively.

a. JSP (JavaServer Pages)

- **Overview**: JSP is a server-side technology used for creating dynamic web pages. It is part of the Java EE (Enterprise Edition) platform and allows embedding Java code in HTML for seamless content rendering.
- **Advantages**:
    - Simplifies the integration of dynamic content
    - Facilitates communication with the back-end using Java.
    - Supports reusable components like JavaBeans and custom tags.
- **Use in QuickEvent Hub**: JSP is employed to render event listings, service provider profiles, and booking forms dynamically, enhancing the interactivity of the platform.

b. CSS (Cascading Style Sheets)

- **Overview**: CSS is a styling language used to format and design the web pages, ensuring consistency and enhancing the visual appeal
- **Advantages**:
    - Provides design flexibility through responsive styling.
    - Reduces redundancy by centralizing styles.
    - Improves accessibility by controlling visual hierarchy.
- **Use in QuickEvent Hub**: CSS is used for styling the platform's UI elements, such as buttons, forms, and navigation menus, ensuring a clean and appealing interface..

c. JavaScript

- **Overview**: JavaScript is a client-side scripting language used to make web pages interactive. It enables dynamic content updates, event handling, and user interactions.
- Advantages:
    - Enhances user experience through interactive features.
    - Supports third-party libraries like jQuery for simplified coding.
    - Enables asynchronous communication using AJAX.
- **Use in QuickEvent Hub**: JavaScript powers form validation, dynamic content loading, and enhances user interactions like pop-ups, modals, and real-time notifications.

2. Back-End Technologies

The back-end of QuickEvent Hub is developed using Java Servlets, ensuring efficient request handling, business logic processing, and server-side management.

a. Java Servlets

- **Overview**: Java Servlets are server-side components that handle requests and responses. They are efficient for developing secure and scalable web applications.
- **Advantages**:
    - Platform-independent and integrates seamlessly with JSP.
    - Supports secure data handling and session management.
    - Provides robust error handling.
- **Use in QuickEvent Hub**: Java Servlets manage user authentication, data processing, and business logic for event booking, user registration, and service provider management.

3. Database Technologies

MySQL is employed as the database management system to store and manage application data securely.

a. MySQL

- **Overview**: MySQL is a popular open-source relational database management system known for its speed, reliability, and flexibility.
- **Advantages**:
    - Supports complex queries and transaction management
    - Ensures data integrity through ACID compliance.
    - High compatibility with Java and JDBC for seamless integration.
- **Use in QuickEvent Hub**: MySQL stores user data, service provider information, booking records, and event details, ensuring efficient data retrieval and management.

4. Authentication and Authorization

Managing user authentication and role-based authorization is essential to control access to different parts of the platform.

- **Overview**: Authentication verifies user identity, while authorization determines user privileges within the application
- **Advantages**:
    - Ensures secure access control
    - Protects sensitive information from unauthorized access.
    - Enhances user trust through data privacy.
- **Use in QuickEvent Hub**: Secure login credentials and role-based access are implemented to distinguish between users, service providers, and administrators.

5. Real-Time Data Handling

Real-time data handling is crucial for updating event statuses, sending notifications, and managing the admin's interaction with users and providers.

- **Overview**: Real-time data handling enables instant updates and communication between users and service providers.
- **Advantages**:
    - Improve user engagement through live notifications.
    - Ensures Quick response to booking requests.
    - Reduces delays in data synchronization.
- **Use in QuickEvent Hub**: Real-time notifications for booking updates, chat support, and availability tracking enhance the overall user experience.

Chapter 3:
REQUIREMENTS AND ANALYSIS

1. Problem definition

In today's fast-paced world, event management is evolving rapidly. While large-scale events are catered to by various professional services, there remains a significant gap in the market for individuals who wish to organize small-scale, home-based events. These events, such as birthday parties, family gatherings, or small community events, require efficient planning but typically do not necessitate the extensive services or high costs associated with large event management platforms.

The problem faced by these individuals lies in the disjointed, time-consuming process of finding the necessary rental items (such as furniture, decorations, sound systems) and hiring catering services in an affordable and convenient manner. Typically, users must contact multiple vendors, negotiate prices, and coordinate services separately, which increases the complexity and reduces the ease of organizing small events. In addition, local caterers and rental providers, particularly small businesses or freelancers, struggle to find a platform where they can easily showcase their services to a broader audience.

Identified Problems

1. **Lack of Integrated Platforms for Small Events**:
    - There is no dedicated platform that allows users to rent items and hire caterers specifically for small-scale, home-based events. Users are often left to search for individual service providers through scattered channels, which is time-consuming and inefficient.
2. **Fragmented Service Offerings**:
    - Most event management platforms focus on large-scale events, with high costs and complex offerings. This leaves people organizing smaller events without an affordable, user-friendly solution tailored to their needs.
3. **Challenges for Service Providers**:
    - Local rental suppliers and home-based caterers face difficulties in reaching potential customers. They typically rely on word of mouth or small-scale advertising, limiting their market reach and business opportunities.
4. **Administrative Overload**:
    - Without a centralized system, event organizers struggle with coordinating multiple vendors, leading to miscommunication, errors, and delays. Additionally, administrators face challenges managing the flow of requests and ensuring that users' requirements are met.

Objective of the Solution

The objective of **QuickEvent Hub** is to provide a seamless, integrated platform that connects users who are organizing small events with local service providers (rental suppliers and caterers). The platform will simplify the process by allowing users to browse and book services in a single location, while service providers can register and showcase their offerings to a wider audience. Additionally, the app will include an administrative panel for monitoring and managing all user interactions, ensuring that requests are handled efficiently.

2. Requirements Specification

1. Functional Requirements

The functional requirements describe the specific behavior and functionalities of the system, including how users interact with the platform and what services the platform offers.

1.1 User Roles and Accounts

1. **User Registration and Login**
    - Users (both event organizers and service providers) must be able to create an account using an email address or social media login (e.g., Google, Facebook).
    - A verification process must be included (email or mobile number verification).
    - Users must be able to log in and log out securely, with support for password recovery.
2. **User Profile Management**
    - **Event Organizers**: Users must be able to manage their profiles, including name, contact information, and event preferences.
    - **Service Providers**: Providers must be able to add/edit information about their services, pricing, availability, and location.
    - **Admin**: The admin must have access to a comprehensive dashboard to manage all user profiles and approve or block accounts if necessary.

1.2 Event Organization and Service Requests

1. **Event Creation**
    - Event organizers must be able to create an event by selecting the type of event (e.g., birthday, family gathering), event date, time, and location.
    - Organizers should be able to specify their requirements, such as the number of attendees, desired rental items (tables, chairs, sound systems), and catering needs.
2. **Service Browsing**
    - Organizers must be able to browse available services, such as rental items and catering, based on their location.
    - A search and filter functionality must be provided, allowing users to filter services by price, availability, ratings, and categories (e.g., food type for caterers, item type for rentals).
3. **Booking Services**
    - Event organizers must be able to add services to a "cart" and proceed with booking.
    - The system must allow users to confirm bookings by selecting the desired services and scheduling pickup or delivery times for rentals and food items for caterers.
    - A booking summary and confirmation page must be provided before finalizing the booking.
4. **Order Management**
    - Organizers must have a dashboard to view their active, pending, and completed orders.
    - Service providers must have a dashboard to manage received orders, update order status (e.g., accepted, in progress, delivered), and communicate with the organizer.
    - The admin must be able to oversee all orders and intervene in case of conflicts.
5. **Real-Time Notifications**
    - Users must receive notifications for important updates, such as order confirmations, changes to bookings, and service provider responses.
    - Notifications can be sent via email, SMS, or in-app alerts.

1.3 Payments and Transactions

1. **Payment Integration**
    - The platform must integrate a payment to allow secure cash-on-delivery.

1.4 Admin Panel

1. **User and Service Provider Management**
    - The admin must have access to a dashboard to manage all users and service providers.
    - The admin must be able to approve or reject new registrations, suspend or block accounts, and resolve any reported issues.

2. Non-Functional Requirements

Non-functional requirements describe the overall system properties and user experience expectations, including performance, security, and usability.

2.1 Usability

- The system must have a user-friendly and intuitive interface, with a clean layout for ease of navigation.
- The platform must be responsive, ensuring that it works seamlessly on all devices (e.g., desktops, tablets, and smartphones).
- The system should support multiple languages to cater to a broader audience, depending on the target regions.

2.2 Performance

- The system must handle multiple concurrent users without slowing down or causing errors.
- Search queries (for browsing services) must return results within 2 seconds, ensuring a smooth user experience.
- The platform must ensure that bookings and payments are processed within acceptable time limits (under 3 seconds for transactions).

2.3 Security

- The platform must implement secure authentication and authorization mechanisms, such as encryption of sensitive data (passwords, payment details) using protocols like SSL/TLS.
- User data, including personal information and payment records, must be securely stored in compliance with data protection regulations (e.g., GDPR).

- The system must provide secure password management, with multi-factor authentication (MFA) options for additional security.

2.4 Scalability

- The platform must be scalable to accommodate an increasing number of users, service providers, and transactions over time without compromising performance.

- Cloud-based infrastructure should be considered (e.g., AWS, Heroku) to allow for dynamic scaling based on demand.

2.5 Availability

- The system must ensure a high level of availability (99.9% uptime) to prevent downtime during critical event bookings and service requests.
- The platform should have mechanisms in place for failover and disaster recovery, ensuring that services remain accessible during server or infrastructure failures.

2.6 Maintainability

- The codebase must follow best practices for maintainability, including modular design, thorough documentation, and adherence to coding standards.
- The platform should support future enhancements and updates, allowing new features to be added without significant disruption to the existing system.

3. System Architecture

3.1 Front-End

- Technologies: **CSS (Cascading Style Sheets)**for building the user interface, with **JSP (JavaServer Pages)** or **JavaScript** for responsive design.
- User interfaces will include event listing pages, user dashboards, booking forms, and admin management panels.

3.2 Back-End

- Technologies: **Java Servlets** for handling server-side logic and **MySQL** for database management.
- APIs will handle user registration, login, service requests, order management, payment processing, and notifications.

3.3 Database

- MYSQL (PostgreSQL) databases will store user data, service provider information, bookings.
- The database will ensure data consistency, security, and scalability.

3.4 Real-Time Updates

- **Socket.io** will be used for real-time notifications, allowing users to receive instant updates on bookings, service confirmations, and order status changes.

3.5 Payment Gateway

- Integration with a secure payment will ensure safe and efficient payment processing cash-on-delivery.

3.Planning and Scheduling

1. Project Phases Overview

The development of **QuickEvent Hub** is divided into six key phases:

1. **Project Initiation and Requirement Analysis**
2. **System Design and Architecture Planning**
3. **Front-End and Back-End Development**
4. **Integration and Testing**
5. **Deployment and Launch**
6. **Post-Launch Support and Maintenance**

Each phase has a set of tasks, team roles, and milestones to ensure the project remains on schedule and within scope.

2. Project Timeline (Gantt Chart)

Below is a general project timeline, showing the phases and duration in weeks:

Phase	Tasks	Duration
Phase 1: Initiation and Analysis	- Project kickoff meeting - Requirement gathering - Functional specification development	2 weeks
Phase 2: Design and Architecture	- System design - Database schema design - API design - Front-end layout prototyping	3 weeks
Phase 3: Front-End Development	- UI/UX design - Front-end coding (JSP) - Integration with backend	5 weeks

Phase 4: Back-End Development	- Back-end coding (Java Servlet) - API integration - Payment gateway integration	6 weeks
Phase 5: Testing and QA	- Unit testing - Integration testing - Bug fixing - Performance optimization	3 weeks
Phase 6: Deployment and Launch	- Final deployment to production - User training sessions - Soft launch - Marketing and promotion	2 weeks
Phase 7: Maintenance and Support	- Ongoing support - Feature enhancements - Bug fixes - Performance monitoring	Ongoing

3. Detailed Task Breakdown

3.1 Phase 1: Initiation and Requirement Analysis (2 weeks)

- **Kickoff Meeting**: Establish project objectives, stakeholders, and timelines.
- **Requirement Gathering**: Conduct meetings with stakeholders to gather detailed functional and non-functional requirements.
- **Functional Specification Document**: Draft a detailed requirement specification document outlining user stories, functional modules, and system behavior.

Deliverables: Requirement specification document, project plan.

3.2 Phase 2: Design and Architecture (3 weeks)

- **System Design**: Create high-level architecture diagrams for the system, including front-end, back-end, and database designs.
- **Database Schema Design**: Design the database schema to support user registration, service listings, bookings, and transactions.
- **API Design**: Define the APIs that will facilitate communication between the front-end and back-end services.

- **UI/UX Prototyping**: Develop wireframes and prototypes for the user interfaces, focusing on user flow and ease of navigation.

Deliverables: Architecture diagram, database schema, UI/UX wireframes, API documentation.

3.3 Phase 3: Front-End Development (5 weeks)

- **UI/UX Development**: Use the wireframes and prototypes to develop the front-end user interface using JSP.
- **Component Development**: Develop reusable front-end components for user registration, service browsing, event creation, and booking.
- **API Integration**: Integrate the front-end with the back-end APIs for data exchange (e.g., user login, service search, bookings).
- **Responsive Design**: Ensure the platform is fully responsive and functions well on desktops, tablets, and mobile devices.

Deliverables: Fully developed and responsive front-end application, integrated with back-end APIs.

3.4 Phase 4: Back-End Development (6 weeks)

- **Back-End Setup**: Set up the back-end infrastructure using **JAVA Servlet**.
- **User Authentication and Authorization**: Implement user login, registration, and authentication, including password recovery and role-based access control (admin, service provider, event organizer).
- **API Development**: Develop APIs to handle user requests, service listings, order management, and notifications.
- **Payment Gateway Integration**: Integrate payment processing through a secure payment gateway (e.g., cash-on-delivery) .
- **Database Integration**: Connect the back-end with the database, ensuring all user data, service listings, and transactions are properly stored and retrievable.

Deliverables: Functional back-end system with integrated database and APIs, including payment processing.

3.5 Phase 5: Testing and QA (3 weeks)

- **Unit Testing**: Conduct unit tests for each component of the system (both front-end and back-end) to ensure individual functionality works as expected.
- **Integration Testing**: Test the entire system by simulating real-world use cases, such as event creation, service booking, and payment transactions.
- **Bug Fixing and Optimization**: Fix any bugs or performance bottlenecks found during testing and optimize the system for speed and reliability.
- **User Acceptance Testing (UAT)**: Allow a select group of users (stakeholders and early testers) to test the platform and provide feedback.

Deliverables: Bug-free, optimized application, ready for deployment.

4. Milestones and Key Deliverables

1. **Requirement Analysis Completed** (End of Phase 1)
2. **System Design Completed** (End of Phase 2)
3. **Front-End Development Completed** (End of Phase 3)
4. **Back-End Development Completed** (End of Phase 4)
5. **Testing and Bug Fixing Completed** (End of Phase 5)
6. **Full Deployment and Live Launch** (End of Phase 6)
7. **Ongoing Maintenance and Updates** (Phase 7)

5. Resource Allocation

- **Project Manager**: Oversees all phases, manages timelines, and ensures deliverables are met.
- **Front-End Developer(s)**: Responsible for UI/UX design and implementation using React.js.
- **Back-End Developer(s)**: Responsible for building the server-side logic, APIs, and database integration using Node.js, Express.js, and a database (e.g., MongoDB or PostgreSQL).
- **QA Engineer(s)**: Ensures that the system is tested thoroughly and meets the quality standards.
- **UI/UX Designer**: Responsible for creating wireframes, mockups, and prototypes for the user interface.
- **Marketing and Support Team**: Focus on promoting the platform and providing support during and after launch.

6.Software and Hardware Requirements

1. Software Requirements

1.1 Front-End Development

- **Operating System**: Windows, macOS, or Linux
- **Framework**: **React.js** for building the front-end interface
- **Languages**: JavaScript (ES6+), HTML5, CSS3
- **Libraries**:
  - **Redux** (for state management)
  - **JavaScript** (for HTTP requests)
  - **JSP or CSS** (for responsive design)
- **Development Tools**:
  - **Apache NetBeans**
  - **JAVA Servlet**
  - **Browser Developer Tools** (for testing and debugging)
- **Version Control**: **Git** with **GitHub** or **GitLab**

1.2 Back-End Development

- **Operating System**: Windows, macOS, or Linux
- **Framework**: **JAVA Servlet** for back-end services
- **Languages**: JavaScript, TypeScript (optional)
- **Database**:
  - **MYSQL**
- **API Development**:
  - **RESTful API** for communication between the front-end and back-end
  - **Postman** for API testing
- **Cloud Hosting**:
  - **AWS**, **Heroku**, or **Google Cloud**
  - **AWS S3** or **Cloudinary** (for storing media files)
  - **Version Control**: Git with GitHub or GitLab

1.3 Database

- **Database**:
    - **MYSQL**: For storing user data, event bookings, rental items, and payments.
- **Database Tools**:
    - **MYSQL**

1.4 Testing and Quality Assurance

- **Testing Frameworks**:
    - **Jest**, **Mocha** with **Chai** (for unit and integration testing)
    - **Selenium**, **Cypress** (for front-end testing)
- **Continuous Integration/Continuous Deployment (CI/CD)**:
    - **GitHub Actions**, **CircleCI**, or **Jenkins**

1.5 Other Software

- **Payment Gateway**: **CASH-ON-DELIVERY.**

2. Hardware Requirements

2.1 Development Environment

For developers working on QuickEvent Hub, the system needs to support modern development environments with high-speed performance.

Recommended Hardware for Development:

- **Processor**: Intel i5/i7 or AMD Ryzen 5/7 or higher
- **RAM**: Minimum 8 GB (16 GB recommended)
- **Storage**: SSD (minimum 256 GB)
- **Graphics**: Integrated graphics or dedicated GPU for heavy UI rendering
- **Monitor**: Dual monitors recommended for coding and testing

2.2 Server Infrastructure

To host the QuickEvent Hub platform, server infrastructure should be scalable depending on traffic and usage.

Cloud Hosting Recommendations:

- **Processor (CPU)**: 2 or more vCPUs for handling multiple requests
- **Memory (RAM)**: Minimum 4 GB (8 GB recommended)
- **Storage**: SSD storage (100 GB or expandable as needed)
- **Backup and Redundancy**: Automated backups through cloud services
- **Load Balancer**: Required to distribute traffic efficiently

2.3 Admin and User Devices

For managing the platform and accessing the app, standard hardware can be used by administrators and users.

Admin Devices:

- **Processor**: Intel i5 or equivalent
- **RAM**: Minimum 8 GB
- **Storage**: SSD (minimum 128 GB)

User Devices:

- **Desktop/Laptop**: Standard machines with web browsers like Chrome, Firefox, or Safari
- **Mobile Devices**: Android and iOS devices, optimized for mobile browsers

3. Network Requirements

- **Internet Speed**: At least 10 Mbps download/upload speed for developers and administrators.
- **Cloud Networking**: AWS or Heroku for reliable cloud hosting with a 99.9% uptime guarantee.
-

4. Security Requirements

- **SSL Certificate**: Ensures secure data transmission over HTTPS.
- **Firewall**: Cloud or server-level firewall for security.
- **Authentication**: Role-based access control, strong password policies, and multi-factor authentication (MFA).
- **Encryption**: Data encryption using AES-256 for sensitive information such as payment details.

5. Preliminary Product

The **QuickEvent Hub** is a platform designed to facilitate small-scale event planning by connecting two types of users: **event organizers** (users who want to rent items or hire home caterers) and **service providers** (users who offer rental items or catering services). The platform is managed by an admin who oversees operations, processes user requests, and ensures efficient interaction between event organizers and service providers.

The preliminary product focuses on core functionalities that address the needs of the target users. The following features are included in the Minimum Viable Product (MVP) for **QuickEvent Hub**:

1. Core Features

1.1 User Registration and Login

- **Event Organizers and Service Providers** can create accounts on the platform by signing up with their email or social media accounts.
- **Authentication**: Users can log in to their accounts securely using email and password, with options for password recovery and multi-factor authentication (MFA).
- **Role-based Access**: Depending on whether the user is an event organizer or service provider, they will see different features after logging in.

1.2 Event Organizer Features

- **Browse Services**: Event organizers can browse available rental items (e.g., chairs, tables, decorations) or catering services offered by service providers.
- **Service Filters**: Users can search for services based on location, availability, price range, or service category.
- **Booking Services**: Event organizers can select items or catering services and book them for specific dates and times.
- **Payment Gateway**: Integration with a secure payment system (like Cash-on-delivery) allows event organizers to make payments for booked services.
- **Order Management**: Users can view their current and past bookings, check the status of ongoing requests, and cancel bookings if necessary.

1.3 Service Provider Features

- **List Rental Items/Services**: Service providers can create listings for the rental items or catering services they offer. Listings include details such as service name, description, price, availability, and photos.
- **Booking Requests**: Providers receive booking requests from event organizers and can either accept or reject them.

- **Update Services**: Service providers can edit their listings and update availability or pricing as needed.
- **Manage Bookings**: Providers can track their active and past bookings, as well as communicate with event organizers for any special instructions.

1.4 Admin Panel

- **User Management**: The admin can manage user accounts (event organizers and service providers), including suspending or approving accounts based on the platform's guidelines.
- **Service Management**: Admins review and approve new service listings to ensure they meet platform quality standards.
- **Request Redirection**: Admins help redirect event organizer requests to the appropriate service providers if manual intervention is required.
- **Transaction Monitoring**: Admins can monitor financial transactions made through the platform for fraud prevention or dispute resolution.
- **Reports and Analytics**: Basic reports are available to help admins track platform performance, user activity, and financial transactions.

2. User Interface and Experience

2.1 Event Organizer UI

- A **user-friendly dashboard** that allows easy navigation between browsing services, booking items, and managing event details.
- A **mobile-optimized design** ensures that event organizers can browse and book services on their phones and tablets.
- **Search and Filter** functionalities provide an intuitive way to find the best services suited to the event organizer’s needs.

2.2 Service Provider UI

- A **service management dashboard** where providers can list items/services, view requests, and manage bookings.
- **Notification system**: Service providers receive real-time notifications for new booking requests or updates to existing bookings.

2.3 Admin UI

- **Centralized control panel**: Admins have access to a well-structured dashboard where they can manage users, services, bookings, and financial data.
- **Automated alerts**: Admins are notified of any flagged bookings or services that require review, enabling quick action.

3. Technical Specifications

3.1 Back-End Architecture

- **Server**: Node.js with Express.js to handle server requests.
- **Database**: MongoDB for flexible and scalable data storage (alternative: PostgreSQL for structured data).
- **APIs**: RESTful APIs to facilitate communication between the front-end and back-end.
- **Authentication**: JWT (JSON Web Token) for secure login and session management.

3.2 Front-End Architecture

- **React.js**: A responsive, single-page application (SPA) to create a seamless user experience for event organizers and service providers.
- **State Management**: Redux for managing the application's global state.
- **CSS Framework**: Bootstrap or Material-UI for styling and responsive design.

3.3 Payment and Security

- **Payment Gateway Integration**: Secure payment processing through Stripe or PayPal for handling transactions between event organizers and service providers.
- **SSL Encryption**: Ensures all user data and transactions are encrypted and securely transmitted.

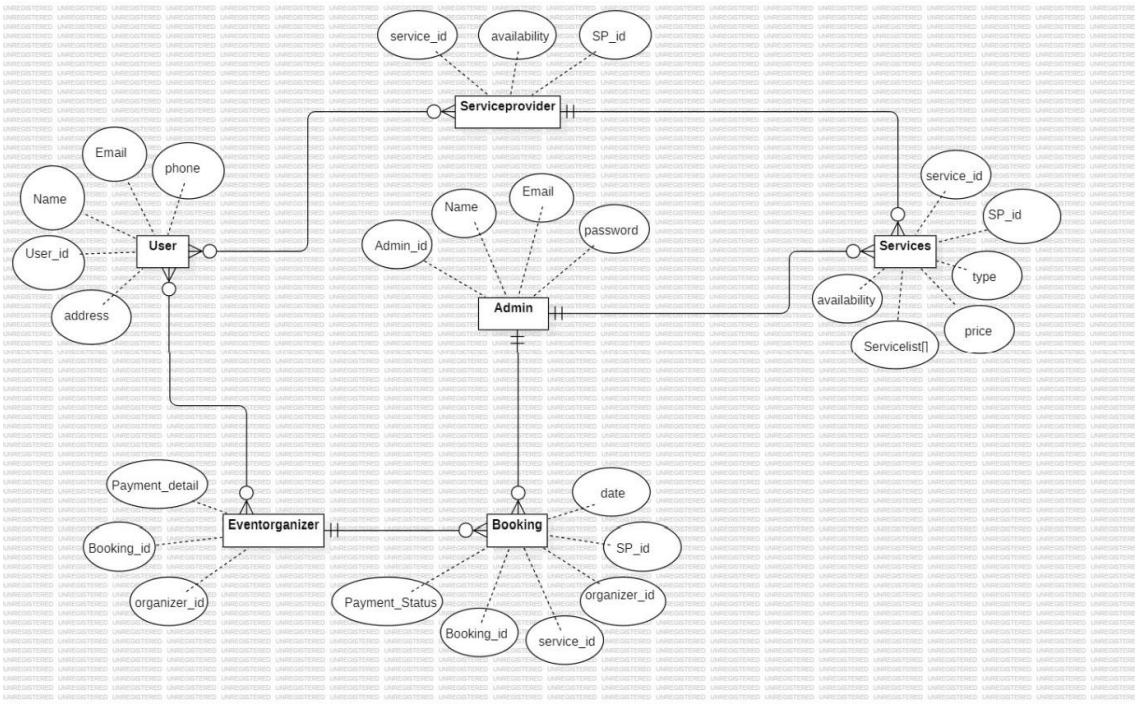
4. Scope for Improvement

The preliminary product is designed to cater to essential functionalities, but future versions can introduce additional features for enhanced user experience:

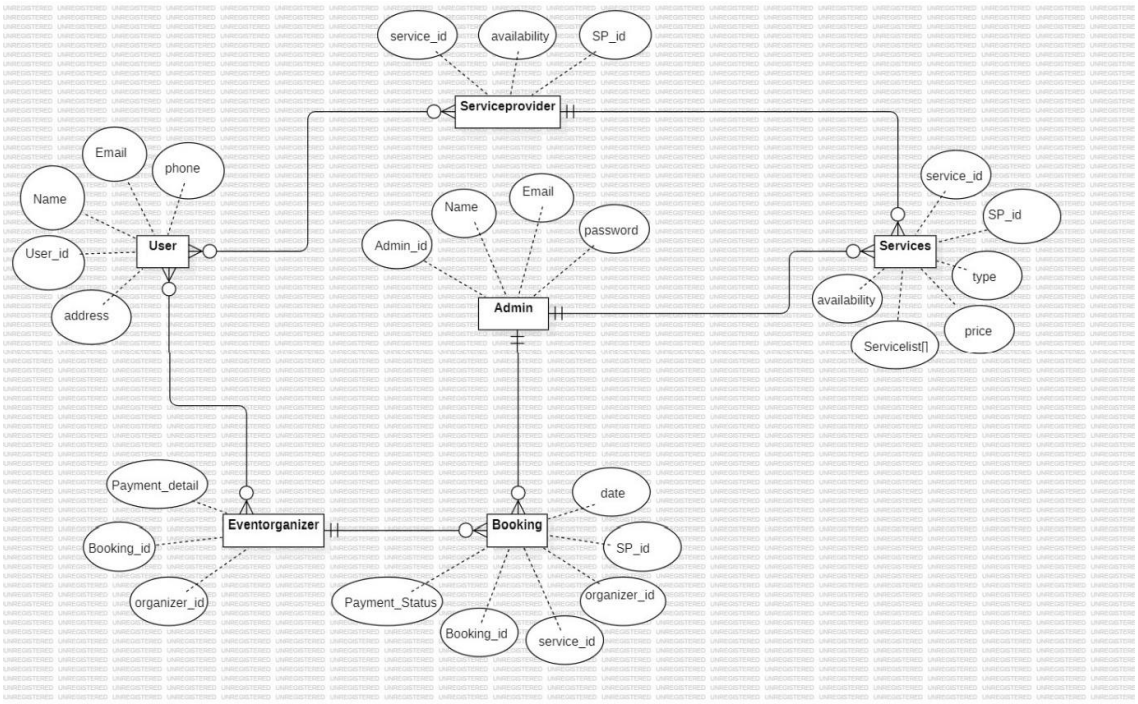
- **Reviews and Ratings**: Event organizers can leave feedback and rate service providers after an event.
- **Advanced Reporting**: More detailed analytics for admin, including booking trends, popular services, and financial projections.
- **Notifications**: In-app messaging and real-time notifications to improve communication between users.
- **Event Templates**: Pre-built event packages that make booking items and services easier for recurring events like birthdays or small gatherings.

6. Conceptual Model

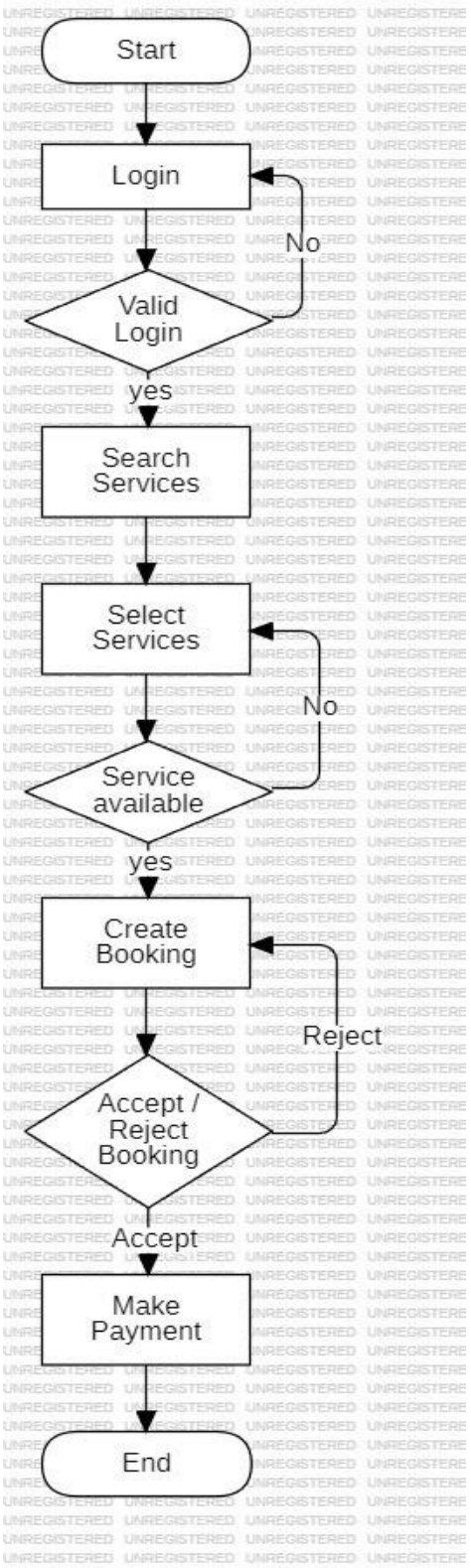
1. ER Diagram



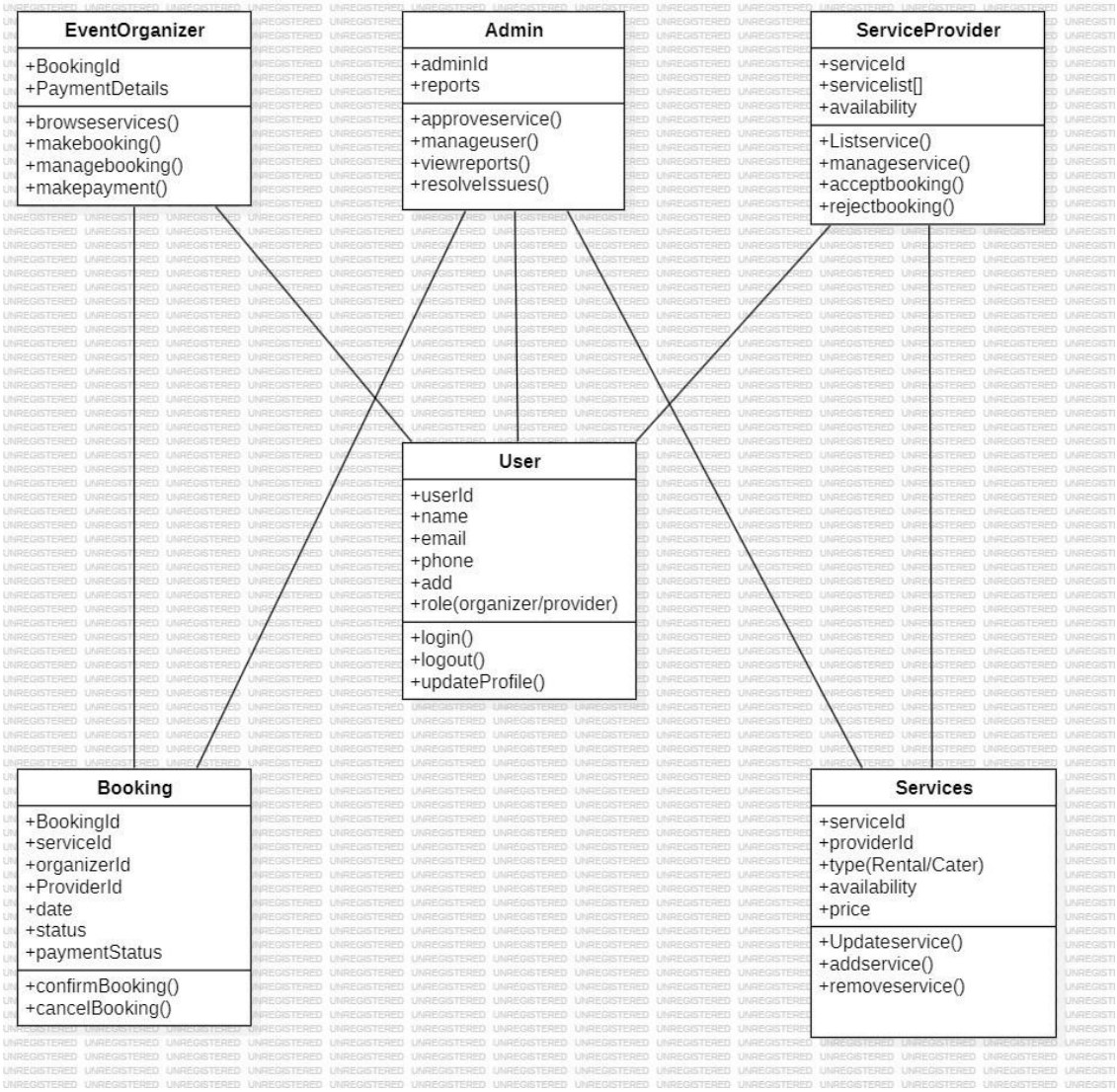
2. ER Diagram


service_id
availability
SP_id
Serviceprovider
Email
phone
Name
User_id
User
address
Name
Email
Admin_id
password
Admin
service_id
SP_id
Services
type
availability
price
Servicelist[]
Payment_detail
Eventorganizer
Booking_id
organizer_id
date
Booking
SP_id
Payment_Status
Booking_id
organizer_id
service_id

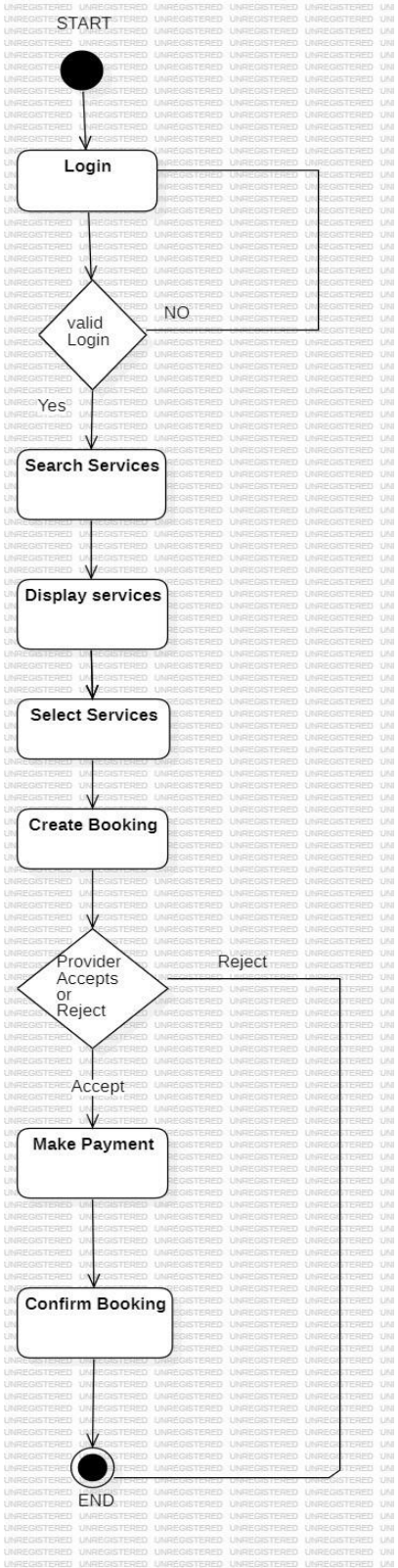
3. Flowchart


Start
Login
No
Valid Login
yes
Search Services
Select Services
No
Service available
yes
Create Booking
Reject
Accept / Reject Booking
Accept
Make Payment
End

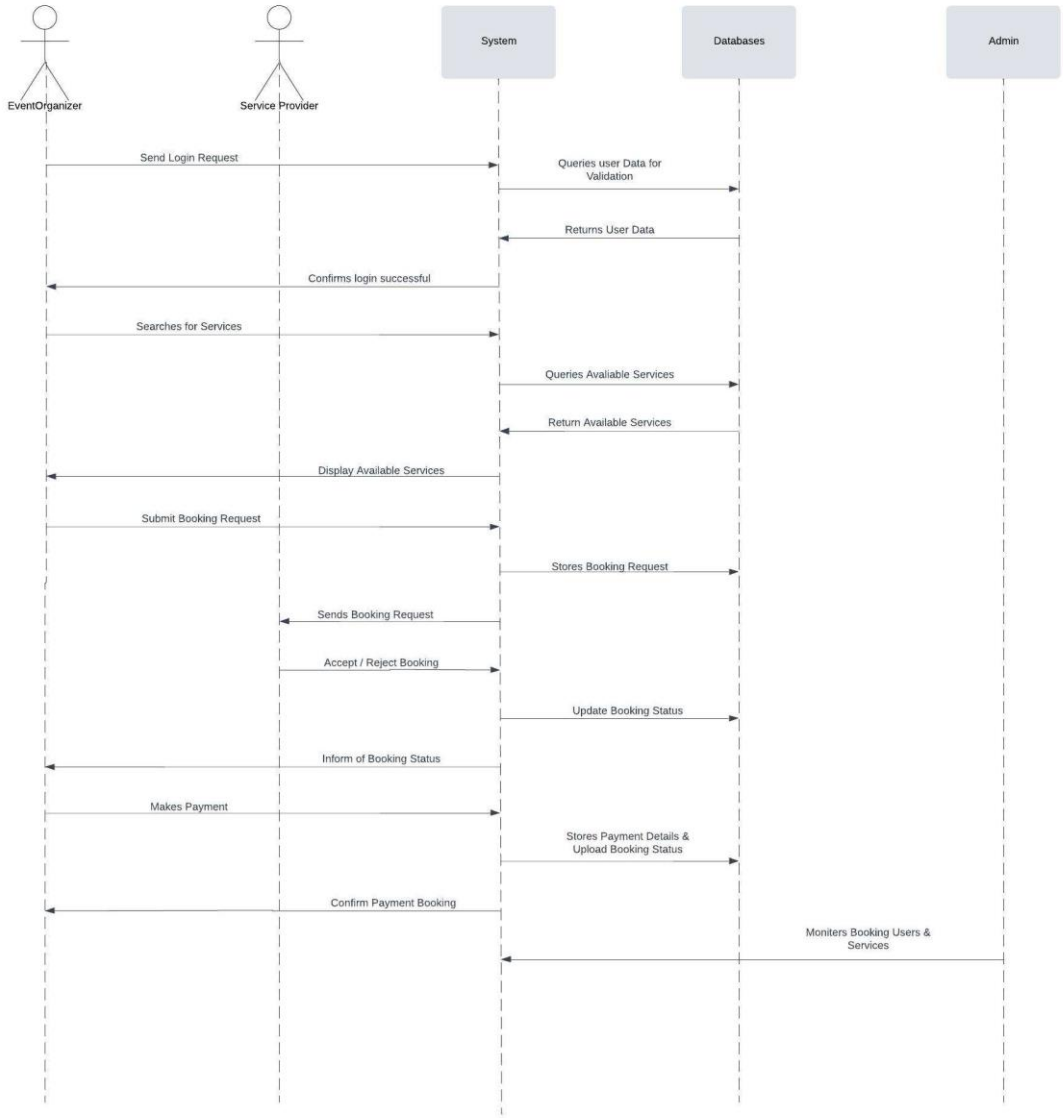
4. Class Diagram



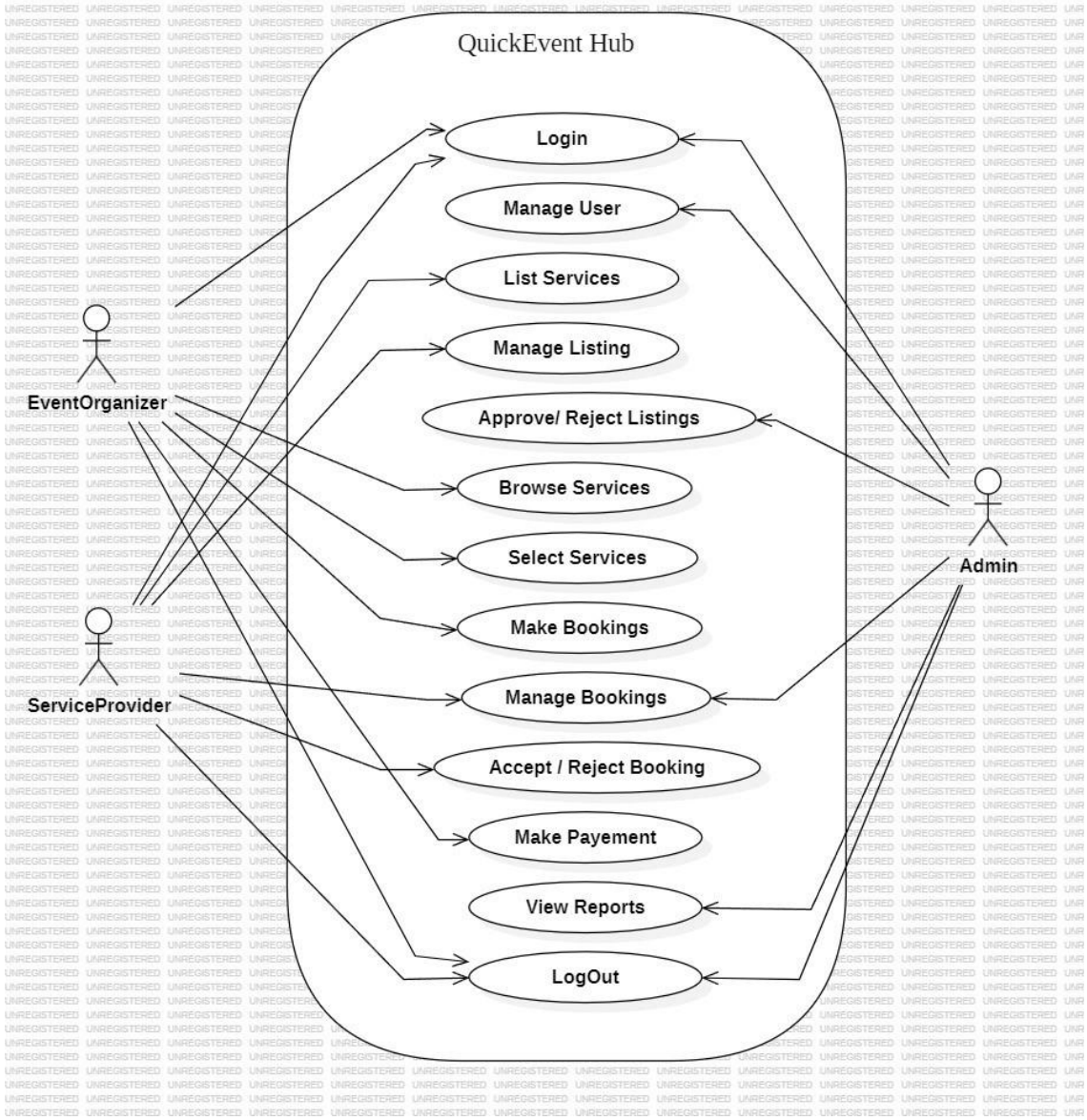
5. Activity Diagram



6. Sequence Diagram

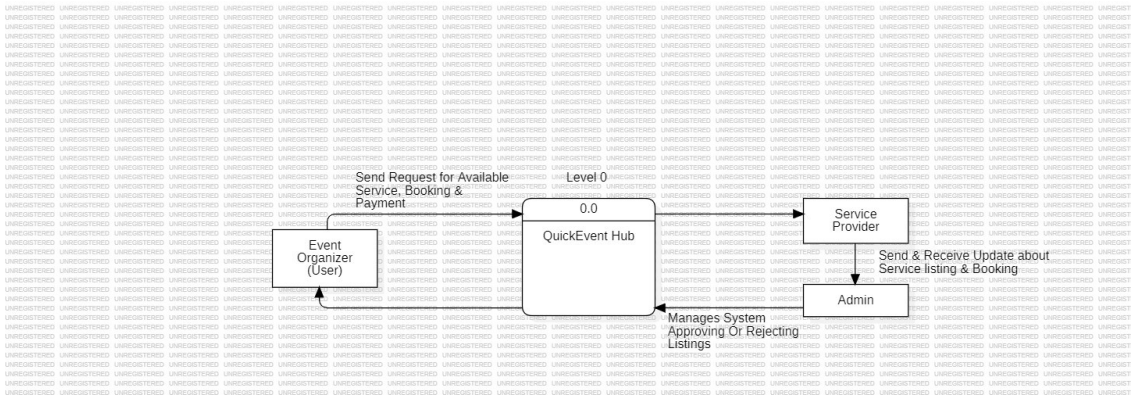


7. Use Case Diagram

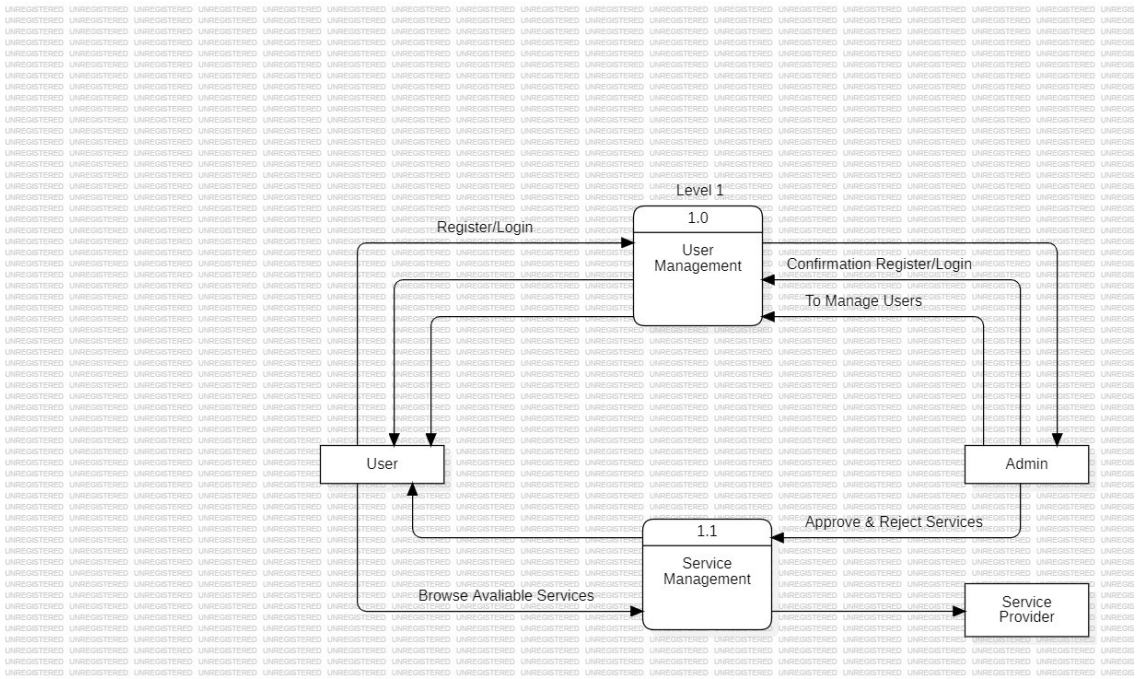


8. DFD Diagram

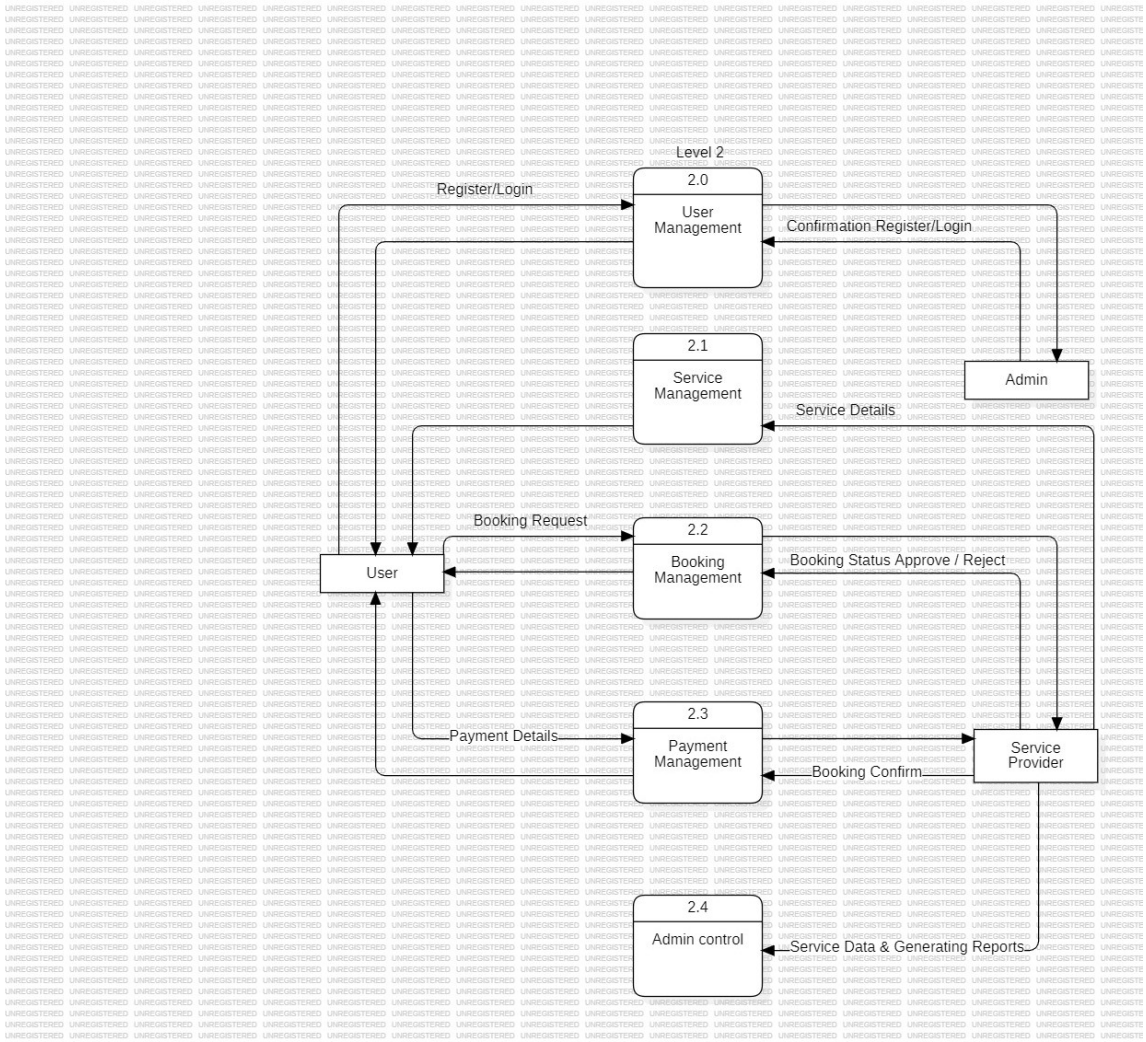
Level 0



Level 1



Level 2


Level 2
2.0
User Management
Register/Login
Confirmation Register/Login
2.1
Service Management
Admin
Service Details
Booking Request
2.2
Booking Management
User
Booking Status Approve / Reject
2.3
Payment Management
Payment Details
Service Provider
Booking Confirm
2.4
Admin control
Service Data & Generating Reports

CHAPTER 4 : SYSTEM DESIGN

1. Basic modules

1. User Management Module

- **Description**: This module handles all user-related functions, such as registration, login, profile management, and role-based access control (Event Organizer, Service Provider, Admin).
- **Key Features**:
    - User registration (Event Organizer and Service Provider).
    - Login/logout functionality.
    - Profile management (update user information, password reset).
    - Role-based access control (different user functionalities for Event Organizer, Service Provider, and Admin).

2. Service Listing Module

- **Description**: This module allows Service Providers to add, update, and manage their services (event rentals, catering, etc.) available for booking.
- **Key Features**:
    - Service registration (list items, services, pricing, availability).
    - Edit or update listed services.
    - View service performance (number of bookings, user ratings).

3. Search and Browse Module

- **Description**: This module enables Event Organizers to search and browse for available services based on criteria such as location, category (e.g., catering, rentals), and price range.
- **Key Features**:
    - Search functionality (by service type, location, price).
    - Filter and sort options (price, rating, availability).
    - Display detailed information for each service.

4. Booking Management Module

- **Description**: The booking module manages the process where Event Organizers can make service requests, and Service Providers can accept or reject those requests.
- **Key Features**:
    - Create and submit booking requests.
    - Track booking status (pending, accepted, rejected).
    - Modify or cancel bookings (before approval).
    - Notifications for booking confirmations or rejections.

5. Payment Processing Module

- **Description**: This module facilitates payment transactions between Event Organizers and Service Providers, including secure payment handling and tracking.
- **Key Features**:
    - Payment gateway integration (credit/debit cards, online payment options).
    - Manage payment statuses (pending, completed, failed).
    - Generate invoices or payment receipts.
    - Secure payment data storage and transaction tracking.

6. Admin Control Module

- **Description**: The admin module allows the administrator to monitor, control, and oversee the entire platform's operations, ensuring smooth coordination between Event Organizers and Service Providers.
- **Key Features**:
    - Monitor user registrations and service listings.
    - Approve or reject services and bookings.
    - Manage disputes and handle complaints.
    - Generate reports (user activity, bookings, payments).

7. Security and Authentication Module

- **Description**: Ensures the security of user data, including passwords, payment information, and personal details, by implementing authentication and encryption protocols.
- **Key Features**:
    - Secure login with encrypted passwords.
    - Two-factor authentication (optional).
    - Secure storage for sensitive data (e.g., payment information).
    - Role-based access control for different user types.

Additional Modules (Optional for Future Development):

- **Marketing and Promotions Module**: Handles the promotion of services, including discount codes or promotional offers for Event Organizers.
- **Calendar Integration Module**: Allows Event Organizers and Service Providers to sync bookings with their calendars (e.g., Google Calendar).

2. Data Design

Entity-Relationship (ER) Design

In the **QuickEvent Hub**, we need to define key entities such as **Users**, **Services**, **Bookings**, **Payments**, and more. Each of these entities will have associated attributes and relationships with one another.

Key Entities and Attributes:

1. **User**
    - **Attributes**:
        - User_ID (Primary Key)
        - Name
        - Email
        - Password (encrypted)
        - Contact_Number
        - Address
        - Role (Event Organizer, Service Provider, Admin)
        - Registration_Date
    - **Relationships**:
        - One User can have many **Bookings** (Event Organizers).

2. **Service**
    - **Attributes**:
        - Service_ID (Primary Key)
        - Service_Provider_ID (Foreign Key references User_ID)
        - Service_Type (e.g., Catering, Event Rentals)
        - Description
        - Price
        - Availability_Status
        - Service_Location
        - Rating (average rating based on user feedback)
    - **Relationships**:
        - Each **Service** is offered by a **Service Provider** (User).
        - A **Service** can have many **Bookings** associated with it.

- **Booking**
    - **Attributes**:
        - Booking_ID (Primary Key)
        - Event_Organizer_ID (Foreign Key references User_ID)
        - Service_ID (Foreign Key references Service_ID)
        - Booking_Date
        - Event_Date
        - Status (Pending, Accepted, Rejected, Completed)
        - Total_Cost
    - **Relationships**:
        - A **Booking** is made by an **Event Organizer** (User).
        - A **Booking** is linked to a specific **Service**.

3. **Payment**
    - **Attributes**:
        - Payment_ID (Primary Key)

        - Booking_ID (Foreign Key references Booking_ID)
        - Payment_Amount
        - Payment_Method (Credit Card, Debit Card, etc.)
        - Payment_Status (Pending, Completed, Failed)
        - Payment_Date
    - **Relationships**:
        - A **Payment** is associated with a **Booking**.
        - Each **Booking** can have only one **Payment**.

4. **Review**
    - **Attributes**:
        - Review_ID (Primary Key)
        - Booking_ID (Foreign Key references Booking_ID)
        - Rating (1 to 5 stars)
        - Review_Comment
        - Review_Date
    - **Relationships**:
        - A **Review** is associated with a **Booking**.
        - **Bookings** can have one or more **Reviews**.

5. **Admin Actions**
    - **Attributes**:
        - Admin_Action_ID (Primary Key)
        - Admin_ID (Foreign Key references User_ID)
        - Action_Type (e.g., Approve Service, Remove User)
        - Action_Date
    - **Relationships**:
        - **Admin Actions** are performed by an **Admin** (User).
        - Each **Admin Action** is logged for security purposes.

2.1 Schema Design

Database Schema Design

Based on the ER model, here's an overview of the database tables and how they might be structured:

User Table

Column	Data Type	Description
User_ID	INT (PK)	Unique ID for the user
Name	VARCHAR(100)	User's full name
Email	VARCHAR(100)	User's email address (unique)
Password	VARCHAR(255)	Encrypted password
Contact_Number	VARCHAR(20)	User's phone number
Address	TEXT	User's address
Role	ENUM	Defines if the user is an Event Organizer, Service Provider, or Admin
Registration_Date	DATE	The date the user registered

Service Table

Column	Data Type	Description
Service_ID	INT (PK)	Unique ID for the service
Service_Provider_ID	INT (FK)	ID of the service provider (User)
Service_Type	ENUM	Type of service (e.g., Catering, Rentals)
Description	TEXT	Detailed description of the service
Price	DECIMAL	Price of the service
Availability_Status	BOOLEAN	Indicates if the service is available
Service_Location	VARCHAR(255)	Location where the service is available

Booking Table

Column	Data Type	Description
Booking_ID	INT (PK)	Unique ID for the booking
Event_Organizer_ID	INT (FK)	ID of the event organizer (User)
Service_ID	INT (FK)	ID of the service
Booking_Date	DATE	Date the booking was made
Event_Date	DATE	Date of the event
Status	ENUM	Booking status (Pending, Accepted, Rejected, Completed)
Total_Cost	DECIMAL	Total cost of the booking

Payment Table

Column	Data Type	Description
Payment_ID	INT (PK)	Unique ID for the payment
Booking_ID	INT (FK)	Associated booking ID
Payment_Amount	DECIMAL	Amount of payment
Payment_Method	ENUM	Payment method (Credit Card, Debit Card, etc.)
Payment_Status	ENUM	Payment status (Pending, Completed, Failed)
Payment_Date	DATE	Date the payment was made

Review Table

Column	Data Type	Description
Review_ID	INT (PK)	Unique ID for the review
Booking_ID	INT (FK)	ID of the related booking
Rating	INT	Rating provided by the event organizer (1 to 5)
Review_Comment	TEXT	Feedback provided by the user
Review_Date	DATE	Date the review was submitted

2.2 Data Integrity and Constraints

1. Types of Data Integrity

1.1. Entity Integrity

Entity integrity ensures that each record in a table is unique and identifiable. This is typically enforced using **Primary Keys**.

- **Primary Key Constraint**: A unique identifier for each row in a table, which ensures that no duplicate records exist, and no primary key value is null.

1.2. Referential Integrity

Referential integrity ensures that relationships between tables remain consistent. When a foreign key exists, it must reference a valid record in the parent table.

- **Foreign Key Constraint**: Ensures that a record in one table must correspond to a valid entry in another table. For example, a booking must reference an existing user and service.

1.3. Domain Integrity

Domain integrity ensures that data entered into a database falls within a specific allowable range or format.

- **Data Type Constraints**: Enforces the data type for each column (e.g., integers, strings, dates, etc.).
- **CHECK Constraints**: Enforces specific rules on the data that can be entered into a column (e.g., a rating must be between 1 and 5).

1.4. User-Defined Integrity

User-defined integrity consists of business rules that go beyond the default constraints of the database system and ensure that the application functions properly within its business logic.

- For example, a **Booking** can only be made for an available service, and a payment can only be processed for confirmed bookings.

2. Constraints for Each Table in the QuickEvent Hub Project

2.1. User Table

Column	Data Type	Constraints
User_ID	INT	**Primary Key**, Unique, Not Null
Email	VARCHAR(100)	Unique, Not Null, Valid email format
Password	VARCHAR(255)	Not Null, Encrypted
Role	ENUM	Check (Role IN ('Organizer', 'Provider', 'Admin'))
Registration_Date	DATE	Not Null, Default (Current Date)

- **Entity Integrity**: User_ID is a primary key ensuring that each user is unique.
- **Domain Integrity**: The Role column allows only predefined values: **Event Organizer**, **Service Provider**, or **Admin**.

2.2. Service Table

Column	Data Type	Constraints
Service_ID	INT	**Primary Key**, Unique, Not Null
Service_Provider_ID	INT	**Foreign Key** (References User_ID in User Table), Not Null
Service_Type	ENUM	Check (Service_Type IN ('Catering', 'Rentals'))
Price	DECIMAL	Not Null, Check (Price > 0)
Availability_Status	BOOLEAN	Not Null
Rating	DECIMAL	Check (Rating BETWEEN 1 AND 5)

- **Referential Integrity**: Service_Provider_ID is a foreign key ensuring that every service is provided by a valid registered user.
- **Domain Integrity**: The Service_Type column only accepts predefined categories such as **Catering** or **Rentals**.
- **Business Rule**: The Availability_Status should only allow bookings if set to **True**.

2.3. Booking Table

Column	Data Type	Constraints
Booking_ID	INT	**Primary Key**, Unique, Not Null
Event_Organizer_ID	INT	**Foreign Key** (References User_ID in User Table), Not Null
Service_ID	INT	**Foreign Key** (References Service_ID in Service Table), Not Null
Booking_Date	DATE	Not Null, Default (Current Date)
Event_Date	DATE	Not Null, Check (Event_Date >= Booking_Date)
Status	ENUM	Check (Status IN ('Pending', 'Accepted', 'Rejected', 'Completed'))
Total_Cost	DECIMAL	Not Null, Check (Total_Cost >= 0)

- **Referential Integrity**: Event_Organizer_ID and Service_ID are foreign keys ensuring that every booking refers to a valid user and service.
- **Business Rule**: The Event_Date must be greater than or equal to the Booking_Date.

2.4. Payment Table

Column	Data Type	Constraints
Payment_ID	INT	**Primary Key**, Unique, Not Null
Booking_ID	INT	**Foreign Key** (References Booking_ID in Booking Table), Not Null
Payment_Amount	DECIMAL	Not Null, Check (Payment_Amount > 0)
Payment_Status	ENUM	Check (Payment_Status IN ('Pending', 'Completed', 'Failed'))
Payment_Date	DATE	Not Null

- **Referential Integrity**: Booking_ID is a foreign key ensuring that each payment is linked to a valid booking.
- **Business Rule**: Payment_Amount must be greater than 0, and a booking can have only one completed payment.

2.5. Review Table

Column	Data Type	Constraints
Review_ID	INT	**Primary Key**, Unique, Not Null
Booking_ID	INT	**Foreign Key** (References Booking_ID in Booking Table), Not Null
Rating	INT	Not Null, Check (Rating BETWEEN 1 AND 5)
Review_Comment	TEXT	Optional
Review_Date	DATE	Not Null

- **Referential Integrity**: Booking_ID is a foreign key ensuring that a review is tied to a valid booking.
- **Domain Integrity**: Rating values must be between 1 and 5.
- **Business Rule**: A review can only be submitted after the event has been completed.

3. Additional Constraints

Foreign Key Constraints

Foreign keys ensure that there is a valid link between related records in different tables. For instance:

- **Service_Provider_ID in Service Table** references **User_ID in User Table**.
- **Event_Organizer_ID and Service_ID in Booking Table** reference **User_ID and Service_ID**, respectively.
- **Booking_ID in Payment Table** references **Booking_ID in Booking Table**.

Unique Constraints

Certain fields like Email (in User Table) and Service_ID (in Service Table) must have unique values to avoid duplication.

Not Null Constraints

Critical columns like User_ID, Booking_ID, Service_ID, and Total_Cost are set as **Not Null**, ensuring that no important field is left empty during data entry.

4. Ensuring Data Integrity with Constraints

- **Preventing Orphan Records**: The foreign key constraints ensure that no orphaned records exist, e.g., a booking without a valid service or user.
- **Consistency of Data**: The CHECK constraints ensure that invalid data, such as negative prices or incorrect booking statuses, are not allowed.
- **Accuracy of Relationships**: Referential integrity ensures that any relationships between tables (like between bookings and services) are accurately maintained.

5. Transaction Management

For critical operations such as **Bookings** and **Payments**, database transactions should be used to ensure that changes to the database are made as a single unit of work. This ensures that if any part of the transaction fails (e.g., if a payment is not processed successfully), the entire operation is rolled back to maintain data integrity.

6. Triggers and Stored Procedures (Optional)

- **Triggers**: Triggers can be used to enforce additional business logic. For example, automatically updating the **Status** of a booking to "Completed" after the event date has passed.
- **Stored Procedures**: Complex operations like handling payments or processing bulk service requests can be managed using stored procedures to ensure efficient and secure execution of business logic.

3. Procedural Design

Procedural Design for the QuickEvent Hub Project

The **QuickEvent Hub** project involves a systematic flow of operations where users (event organizers) can request services, service providers can offer their services, and an admin can manage the entire process. Procedural design focuses on how these operations are executed step by step, ensuring that the system functions smoothly and meets business requirements. Below is a breakdown of the procedural design for various core functionalities within the project.

1. User Registration Process

Objective:

Allow users (both event organizers and service providers) to register on the platform.

Procedure:

1. **Input**: User submits registration details (name, email, password, role, phone number).
2. **Validation**:
    - Ensure the email is in a valid format.
    - Check if the email is already registered.
    - Validate the password (minimum strength).
    - Ensure the role is either **Organizer** or **Provider**.
3. **Processing**:
    - Encrypt the password using a secure hashing algorithm.
    - Insert user details into the Users table with default values like registration date.
4. **Output**: Confirmation message is displayed after successful registration.
5. **Error Handling**:
    - If email already exists, return an error.
    - If validation fails, return appropriate messages (e.g., "Invalid email format" or "Password too weak").

2. Login Process

Objective:

Allow registered users to log in to the platform.

Procedure:

1. **Input**: User submits email and password.
2. **Validation**:
    - Check if the email exists in the Users table.
    - Verify the password by comparing the encrypted password.
3. **Processing**:
    - If login is successful, create a session for the user.
    - If the user is an admin, direct them to the admin panel.

1. **Output**: Display dashboard relevant to the user role (Organizer or Provider).
2. **Error Handling**:
    - If credentials are invalid, return an error message ("Invalid email or password").

3. Service Registration Process (For Service Providers)

Objective:

Allow service providers to register their services (Catering or Rentals) on the platform.

Procedure:

1. **Input**: Service provider submits service details (type, description, price, availability).
2. **Validation**:
    - Ensure service type is either **Catering** or **Rentals**.
    - Validate price (must be a positive number).
3. **Processing**:
    - Insert the service into the Services table.
    - Set the availability status to **True** by default.
4. **Output**: Confirmation message of successful service registration.
5. **Error Handling**:
    - If service registration fails, show an appropriate error message.

4. Service Search & Booking Process (For Event Organizers)

Objective:

Allow event organizers to search for available services and book them for events.

Procedure:

1. **Input**: Event organizer selects service type and specifies filters (price range, availability).
2. **Processing**:
    - Query the Services table to fetch available services based on the selected filters.
    - Display the filtered list to the event organizer.

3. **Booking**:
    - Event organizer selects a service, specifies the event date, and confirms the booking.
    - Insert the booking details (organizer ID, service ID, event date, total cost) into the Bookings table.
4. **Output**: Booking confirmation is displayed to the event organizer.
5. **Error Handling**:
    - If no services are available for the selected criteria, display an appropriate message.
    - Ensure that the event date is valid (future date).

5. Admin Request Management Process

Objective:

Allow the admin to manage incoming requests from both event organizers and service providers.

Procedure:

1. **Input**: Admin logs in to the admin panel.
2. **View Requests**:
    - Admin can view all service requests and booking requests in a dashboard.
    - Admin can filter by pending, accepted, and rejected status.
3. **Processing**:
    - Admin can accept or reject service requests and bookings.
    - Update the status of the relevant entry (service or booking) in the database (Services and Bookings tables).
4. **Output**: Admin receives a notification confirming the action taken.
5. **Error Handling**:
    - If a request update fails, log the error and notify the admin.

6. Review Submission Process

Objective:

Allow event organizers to submit reviews for services after an event.

Procedure:

1. **Input**: Event organizer submits a review and rating for the service after the event is completed.
2. **Validation**:
    - Check if the event date has passed.
    - Ensure the rating is between 1 and 5.
3. **Processing**:
    - Insert the review into the Reviews table, linked to the booking.
4. **Output**: Confirmation of review submission.
5. **Error Handling**:
    - If the review submission fails, display an error message.

7. Notification Process (System-Wide)

Objective:

Provide notifications to users (event organizers and service providers) and the admin about important events such as booking confirmations, service requests, and payment status.

Procedure:

1. **Trigger Events**:
    - When a booking is made, notify the service provider.
    - When a payment is completed, notify the event organizer and the admin.
2. **Processing**:
    - Generate notifications with relevant information.
    - Store notifications in a Notifications table for logging purposes.
3. **Output**: Display notifications in the user/admin dashboard.
4. **Error Handling**:
    - If notifications fail to send, log the issue for system administrators.

8. Error Logging and Handling Process

Objective:

Log errors and exceptions occurring within the application and notify the admin of critical issues.

Procedure:

1. **Error Occurrence**: System encounters an exception during any process (e.g., database failure or payment gateway timeout).
2. **Logging**:
    - Capture error details (e.g., timestamp, error type, user action, stack trace).
    - Insert the error details into an Error_Log table.
3. **Notification**:
    - Notify the system admin of critical errors for immediate attention.
4. **Output**: Error logs are stored for debugging purposes, and a user-friendly error message is displayed.
5. **Error Handling**:
    - Admin or system engineer reviews logs to address underlying issues.

3.1 Logic Diagrams

1. User Registration Logic Diagram

This diagram explains the flow of how users (either event organizers or service providers) register on the platform.

Steps:

- Start
- User provides registration details (Name, Email, Password, Role)
- System checks if email is valid
    - Yes: Proceed to next step
    - No: Show error ("Invalid Email")
- System checks if email is already registered
    - Yes: Show error ("Email already registered")
    - No: Proceed
- System validates password strength
    - Yes: Encrypt password
    - No: Show error ("Weak Password")
- System inserts user into the database
- Display confirmation message
- End

2. Login Logic Diagram

This diagram covers the login process for users.

Steps:

- Start
- User enters email and password
- System checks if email exists in the database
    - Yes: Proceed to next step
    - No: Show error ("Email not found")
- System verifies the password
    - Yes: Create session and proceed to the user dashboard
    - No: Show error ("Invalid password")
- Role check: Is the user an admin?
    - Yes: Redirect to the admin dashboard
    - No: Redirect to the regular user dashboard (Organizer or Provider)
- End

3. Service Registration Logic Diagram (For Providers)

This diagram describes how a service provider registers their services (e.g., catering or rentals).

Steps:

- Start
- Service provider inputs service details (Type, Description, Price, Availability)
- System checks if the service type is valid (Catering/Rentals)
    - Yes: Proceed
    - No: Show error ("Invalid Service Type")

- System checks price validity
    - Yes: Proceed
    - No: Show error ("Invalid Price")
- System inserts service into the Services table
- Confirmation message displayed
- End

4. Service Search and Booking Logic Diagram (For Organizers)

This diagram shows how event organizers search for and book services.

Steps:

- Start
- Event organizer selects filters (service type, price range, availability)
- System fetches services matching the filters
    - Are services available?
        - Yes: Display services
        - No: Show message ("No services found")
- Organizer selects a service
- Organizer provides event details and submits booking request
- System inserts booking into Bookings table
- Display booking confirmation
- End

5. Payment Process Logic Diagram

This diagram shows the flow of making a payment for a service booking.

Steps:

- Start
- Event organizer selects a booking to pay for
- System checks if booking is valid and status is "Accepted"

    - Yes: Proceed to payment
    - No: Show error ("Invalid Booking or Status")
- Organizer provides payment details
- System processes payment through payment gateway
    - Payment successful?
        - Yes: Update payment status and booking status to "Completed"
        - No: Show error ("Payment Failed")
- Display payment confirmation
- End

6. Admin Request Management Logic Diagram

This diagram explains how the admin manages incoming requests from users.

Steps:

- Start
- Admin logs into the system
- Admin views service and booking requests
    - Are there any pending requests?
        - Yes: Display requests
        - No: Show message ("No pending requests")
- Admin selects a request and approves/rejects it
- System updates the request status in the database
- Notification sent to users about the status change
- End

7. Review Submission Logic Diagram

This diagram illustrates how event organizers submit reviews for completed services.

Steps:

- Start

- Event organizer selects a completed booking
- System checks if booking is valid and event date has passed
    - Yes: Proceed
    - No: Show error ("Invalid Booking or Event Date")
- Organizer provides a rating (1-5) and review comment
- System checks if the rating is within valid range (1-5)
    - Yes: Proceed
    - No: Show error ("Invalid Rating")
- System inserts review into Reviews table
- Display confirmation message
- End

8. Service Availability Management Logic Diagram (For Providers)

This diagram explains how service providers can update the availability of their services.

Steps:

- Start
- Provider logs into their account
- Provider selects a service to update availability
- System checks if the service exists and belongs to the provider
    - Yes: Proceed
    - No: Show error ("Service not found or not authorized")
- Provider changes availability status (Available/Unavailable)
- System updates the Services table
- Display confirmation message
- End

3.2 Data Structures

1. User Data Structure

The User data structure is used to store information about the users of the platform, including event organizers, service providers, and the admin.

python

```
class User:
    def __init__(self, user_id, name, email, password, role, contact_info, registration_date):
        self.user_id = user_id                  # Unique identifier for each user
        self.name = name                        # Full name of the user
        self.email = email                      # Email address for login
        self.password = password                # Encrypted password
        self.role = role                        # Role (e.g., Organizer, Provider, Admin)
        self.contact_info = contact_info        # Contact details (phone number, address)
        self.registration_date = registration_date  # Date of registration
```

- **Purpose**: Stores essential information for users logging into the platform.
- **Role-based Access**: Different roles (Organizer, Provider, Admin) determine which functionalities users can access.

2. Service Data Structure

The Service structure stores information about the services offered by providers, such as rental items or catering services.

python

```
class Service:
    def __init__(self, service_id, provider_id, service_type, description, price, availability):
        self.service_id = service_id            # Unique identifier for each service
        self.provider_id = provider_id          # Link to the service provider
        self.service_type = service_type        # Type of service (e.g., Catering, Rentals)
        self.description = description          # Service description
```

```
        self.price = price                  # Price for the service
        self.availability = availability    # Availability status (True/False)
```

- **Purpose**: Represents the services that providers register on the platform.
- **Availability**: Ensures that services can be marked as available or unavailable depending on the provider's status.

3. Booking Data Structure

The Booking structure handles the details of service bookings made by event organizers.

python

```
class Booking:
    def __init__(self, booking_id, organizer_id, service_id, event_date, total_cost, status):
        self.booking_id = booking_id            # Unique identifier for each booking
        self.organizer_id = organizer_id        # Link to the event organizer
        self.service_id = service_id            # Link to the service booked
        self.event_date = event_date            # Date of the event
        self.total_cost = total_cost            # Total cost of the booking
        self.status = status                    # Status of the booking (e.g., Pending, Accepted, Completed)
```

- **Purpose**: Tracks service bookings, including their status and associated event details.
- **Status Management**: Manages different stages of the booking lifecycle (e.g., Pending, Accepted, Completed).

4. Payment Data Structure

The Payment structure tracks the payments made for bookings.

python

```
class Payment:
    def __init__(self, payment_id, booking_id, amount, payment_date, payment_method, status):
        self.payment_id = payment_id            # Unique identifier for each payment
        self.booking_id = booking_id            # Link to the corresponding booking
        self.amount = amount                    # Total amount paid
        self.payment_date = payment_date        # Date when the payment was made
        self.payment_method = payment_method    # Payment method used (e.g., Credit Card, PayPal)
        self.status = status                    # Payment status (e.g., Successful, Failed)
```

- **Purpose**: Records and manages payment details for each booking.
- **Payment Status**: Tracks whether a payment was successful or failed, crucial for handling booking completion.

5. Review Data Structure

The Review structure stores feedback provided by event organizers after an event.

python

```
class Review:
    def __init__(self, review_id, booking_id, organizer_id, service_id, rating, comment, review_date):
        self.review_id = review_id              # Unique identifier for each review
        self.booking_id = booking_id            # Link to the booking
        self.organizer_id = organizer_id        # Link to the organizer
        self.service_id = service_id            # Link to the service
        self.rating = rating                    # Rating (1-5 scale)
        self.comment = comment                  # Review comment
        self.review_date = review_date          # Date when the review was submitted
```

- **Purpose**: Allows users to leave feedback on services, contributing to provider reputation.
- **Rating and Comments**: Enables rating of services on a scale of 1-5, along with optional comments.

6. Notification Data Structure

The Notification structure handles system notifications for both service providers and event organizers.

python

```
class Notification:
    def __init__(self, notification_id, user_id, message, notification_date, status):
        self.notification_id = notification_id  # Unique identifier for each notification
        self.user_id = user_id                  # User receiving the notification
        self.message = message                  # Notification content
        self.notification_date = notification_date  # Date of notification
        self.status = status                    # Status (e.g., Unread, Read)
```

- **Purpose**: Sends notifications to users regarding booking confirmations, payment updates, etc.
- **Read/Unread Status**: Keeps track of which notifications have been viewed by the user.

7. Error Log Data Structure

The ErrorLog structure is used to track any system errors or exceptions that occur during runtime.

python

```
class ErrorLog:
    def __init__(self, error_id, error_message, timestamp, user_action, stack_trace):
        self.error_id = error_id                # Unique identifier for each error log
        self.error_message = error_message      # Error description
        self.timestamp = timestamp              # Time when the error occurred
        self.user_action = user_action          # Action performed by the user at the time of the error
        self.stack_trace = stack_trace          # Technical stack trace for debugging purposes
```

- **Purpose**: Logs system errors for debugging and monitoring.
- **Stack Trace**: Helps developers locate the root cause of system issues by tracking the technical details of the error.

8. Admin Data Structure

The admin structure manages information about the platform administrators.

python

```
class Admin:
    def __init__(self, admin_id, name, email, password):
        self.admin_id = admin_id                # Unique identifier for each admin
        self.name = name                        # Admin's full name
        self.email = email                      # Admin's email for login
        self.password = password                # Admin's encrypted password
```

- **Purpose**: Stores credentials and details for platform administrators.
- **Admin Privileges**: Allows admins to manage requests, bookings, and user-related activities.

9. System Configuration Data Structure

The SystemConfig structure stores configuration settings for the platform, such as payment thresholds and service availability settings.

python

```
class SystemConfig:
    def __init__(self, config_key, config_value):
        self.config_key = config_key            # Configuration setting name (e.g., "MaxBookingsPerDay")
        self.config_value = config_value        # Configuration setting value (e.g., 10)
```

- **Purpose**: Manages system-wide settings that influence platform behaviour.
- **Dynamic Settings**: Enables easy adjustments to configurations such as booking limits, service availability, etc.

3.3 Algorithms Design

1. User Registration Algorithm

This algorithm governs how a new user (either an event organizer or service provider) registers on the platform.

Input: User details (name, email, password, role)
Output: Success or error message

Algorithm:

1. Start.
2. Accept user details: name, email, password, role.
3. Validate email format:
    - If invalid, return error "Invalid Email".
    - If valid, continue.
4. Check if email already exists in the database:
    - If exists, return error "Email already registered".

    - If not, continue.
5. Validate password strength:
    - If weak, return error "Password too weak".
    - If strong, proceed.
6. Encrypt password.
7. Insert the new user record into the Users table with name, email, encrypted password, and role.
8. Return success message: "User registered successfully".
9. End.

2. Login Algorithm

This algorithm manages the user login process, ensuring credentials are validated and appropriate sessions are created.

Input: User's email and password
Output: Redirect to dashboard or error message

Algorithm:

1. Start.
2. Accept email and password.
3. Check if email exists in the Users table:
    - If not, return error "Email not found".
    - If yes, continue.
4. Validate password:
    - If incorrect, return error "Invalid password".
    - If correct, continue.
5. Create user session.
6. Check user role (Organizer, Provider, Admin):
    - If role == "Admin", redirect to Admin dashboard.
    - If role == "Organizer" or role == "Provider", redirect to User dashboard.
7. End.

3. Service Registration Algorithm (For Providers)

This algorithm handles the registration of services (e.g., rentals, catering) by service providers.

Input: Service details (service type, description, price, availability)
Output: Success or error message

Algorithm:

1. Start.
2. Accept service details: service_type, description, price, availability.
3. Check if service_type is valid (must be "Catering" or "Rentals"):
    - If invalid, return error "Invalid Service Type".
    - If valid, continue.
4. Validate price:
    - If invalid (e.g., negative or non-numeric), return error "Invalid Price".
    - If valid, continue.
5. Insert the service into the Services table with the provided details.
6. Return success message: "Service registered successfully".
7. End.

4. Service Search and Booking Algorithm (For Organizers)

This algorithm handles searching for services and booking them by event organizers.

Input: Search filters (service type, price range, availability), booking details
Output: List of available services or booking confirmation

Algorithm:

1. Start.
2. Accept search filters: service_type, price_range, availability.
3. Query the Services table for matching services:

- service_type == filter.service_type
- price BETWEEN filter.price_min AND filter.price_max
- availability == True

4. If no services match the query, return message "No services found".
5. Display the list of available services.
6. Event organizer selects a service and provides event_date, location, etc.
7. Insert the new booking into the Bookings table with service_id, organizer_id, and event details.
8. Return booking confirmation: "Service booked successfully".
9. End.

5. Payment Processing Algorithm

This algorithm manages the payment process for a service booking.

Input: Payment details (booking ID, payment method, amount)
Output: Success or failure message

Algorithm:

1. Start.
2. Accept payment details: booking_id, payment_method, amount.
3. Check if booking_id exists in the Bookings table and status is "Accepted":
   - If not, return error "Invalid Booking or Status".
   - If valid, continue.
4. Process payment through the payment gateway:
   - If payment fails, return error "Payment failed".
   - If payment succeeds, continue.
5. Update the Bookings table to set status = "Completed" and mark the payment as Paid.
6. Return success message: "Payment successful".
7. End.

6. Review Submission Algorithm

This algorithm allows event organizers to submit reviews for completed services.

Input: Review details (booking ID, rating, comment)
Output: Success or failure message

Algorithm:

1. Start.
2. Accept review details: booking_id, rating, comment.
3. Check if booking_id is valid and the event_date has passed:
    - If invalid, return error "Invalid Booking or Event Date".
    - If valid, continue.
4. Validate rating (must be between 1 and 5):
    - If invalid, return error "Invalid Rating".
    - If valid, continue.
5. Insert the new review into the Reviews table with booking_id, rating, and comment.
6. Return success message: "Review submitted successfully".
7. End.

7. Service Availability Management Algorithm (For Providers)

This algorithm manages updating the availability status of services provided by service providers.

Input: Service ID, availability status
Output: Success or error message

Algorithm:

1. Start.
2. Accept service_id and new availability status.
3. Check if service_id exists in the Services table and belongs to the logged-in provider:
    - If not, return error "Service not found or unauthorized".
    - If valid, continue.
4. Update availability status in the Services table.
5. Return success message: "Service availability updated".

6. End.

8. Notification Handling Algorithm

This algorithm manages notifications for both service providers and event organizers.

Input: Event or action that triggers a notification (e.g., booking request, booking approval)
Output: Notification sent to user

Algorithm:

1. Start.
2. Detect an event that requires a notification (e.g., booking request, booking approval).
3. Create a notification message specific to the event.
4. Insert the notification into the Notifications table with user-specific details.
5. Mark notification as Unread in the Notifications table.
6. End.

9. Admin Request Management Algorithm

This algorithm governs how admins manage requests from both service providers and event organizers.

Input: Admin decision (approve/reject request)
Output: Request status updated

Algorithm:

1. Start.
2. Admin views the list of pending requests.
3. Admin selects a request to approve or reject.
4. If approve, update the request status to "Approved" in the database.
5. If reject, update the request status to "Rejected" and notify the user.
6. Send a notification to the affected user (organizer or provider).
7. End.

10. Error Handling Algorithm

This algorithm tracks and logs errors that occur within the system for future debugging.

Input: Error event or exception
Output: Error log entry

Algorithm:

1. Start.
2. Detect an error or exception.
3. Capture error details: error_message, timestamp, user_action, stack_trace.
4. Insert the error details into the ErrorLog table.
5. Return error message to user (if applicable): "An error occurred. Please try again."
6. End.

4.4 User Interface Design

1. General UI Principles

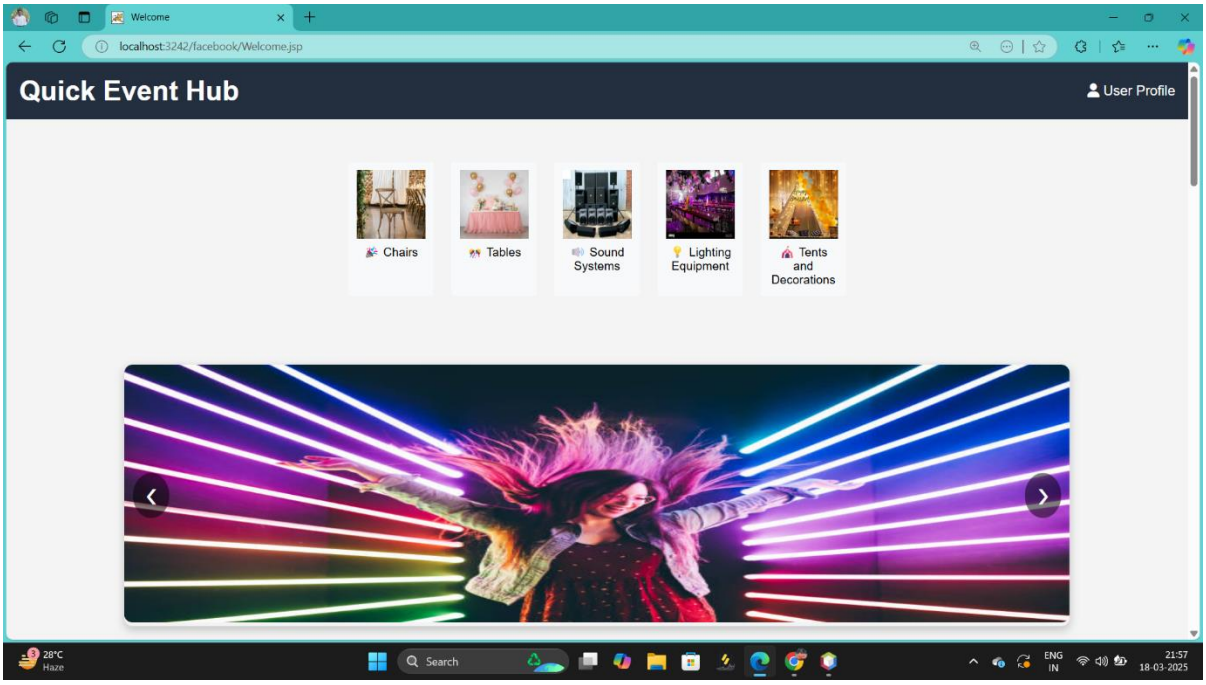
- **Responsive Design**: The app should be optimized for both desktop and mobile use.
- **Minimalist Aesthetic**: Use a clean, simple design with minimal distractions. Focus on key actions (e.g., booking, registering services).
- **Clear Navigation**: Utilize intuitive navigation bars and menus to guide users through their tasks.
- **Consistency**: Maintain consistent color schemes, fonts, and design elements across the app to ensure familiarity and ease of use.
- **Call to Action**: Highlight important actions (e.g., “Book Now”, “Submit Request”) with distinct buttons or icons.
- **Feedback Mechanism**: Provide feedback to users on actions (e.g., “Booking successful”, “Service added successfully”).

2. Home Page Design

The home page serves as the entry point for both event organizers and service providers. It provides a quick overview of the platform’s purpose and a navigation guide for users.

Components:

- **Header**: Contains logo, navigation menu (Home, Services, About, Contact), and login/register buttons.
- **Hero Section**:
    - A short description of what QuickEvent Hub offers.
    - Call-to-action buttons like “Find a Service” for organizers and “Register a Service” for providers.
- **Service Categories Section**:
    - Two main categories: "Event Rentals" and "Home Catering".
    - Icons or images representing these services.
- **Footer**: Quick links, contact information, and social media icons.



3. User Registration/Login Page

This page allows both event organizers and service providers to create accounts or log in to their existing ones.

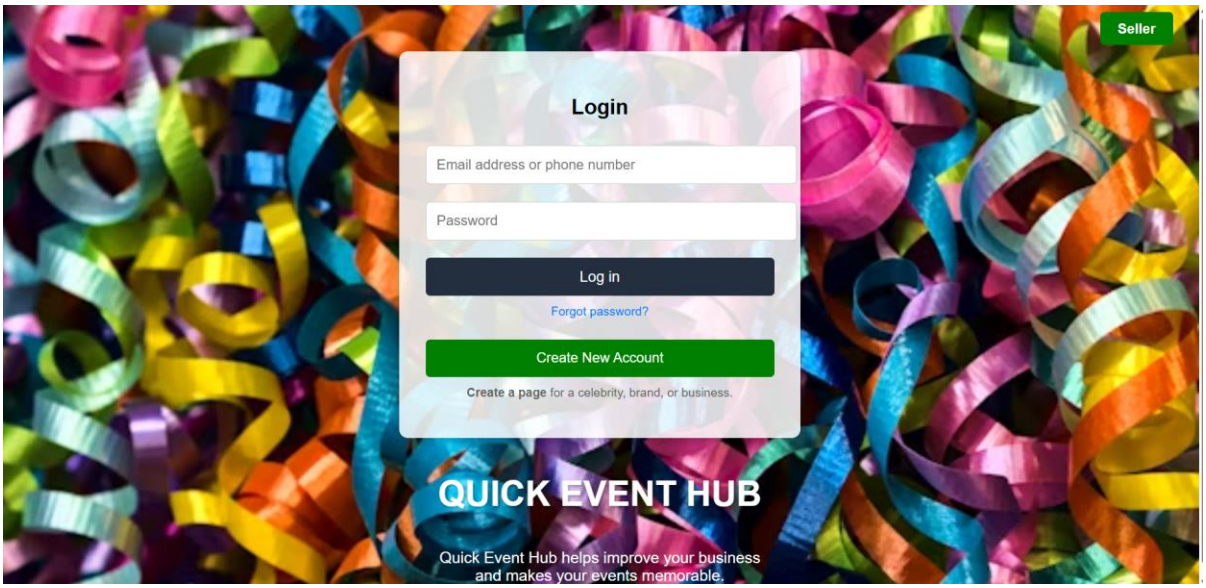
Components:

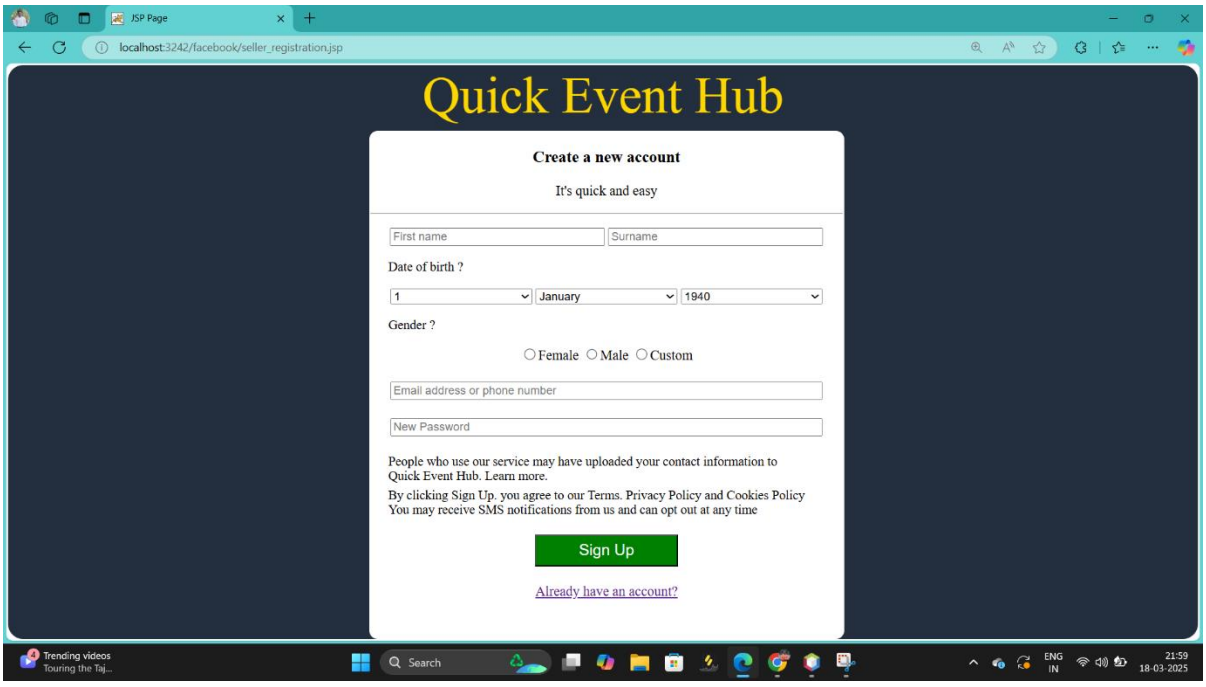
- **Form Fields**:
    - Input fields for email, password, and user role (dropdown to select either "Organizer" or "Provider").

- **Login Button**:
    - Call to action: “Login” for existing users.
    - “Sign Up” button for new users.
- **Social Media Login**: Optional Google/Facebook login integration.

Visuals:

- Simple, clean design with a focus on the input fields.
- Subtle background image or pattern to avoid overwhelming the user.





4. Dashboard (Organizer and Provider)

After login, users will be directed to a personalized dashboard. The layout and functionality differ slightly for event organizers and service providers.

4.1. Organizer Dashboard

Components:

- **Upcoming Events**: Displays a list of upcoming booked services with dates, locations, and service provider details.
- **Book a Service**: A prominent button that redirects to the service search page.
- **Notifications Panel**: Updates on requests, bookings, and event reminders.
- **Profile Settings**: Options to update account information, event preferences, and manage bookings.

Visuals:

- Calendar widget to show upcoming events.
- Card-like design for each booked service with details (date, time, provider).

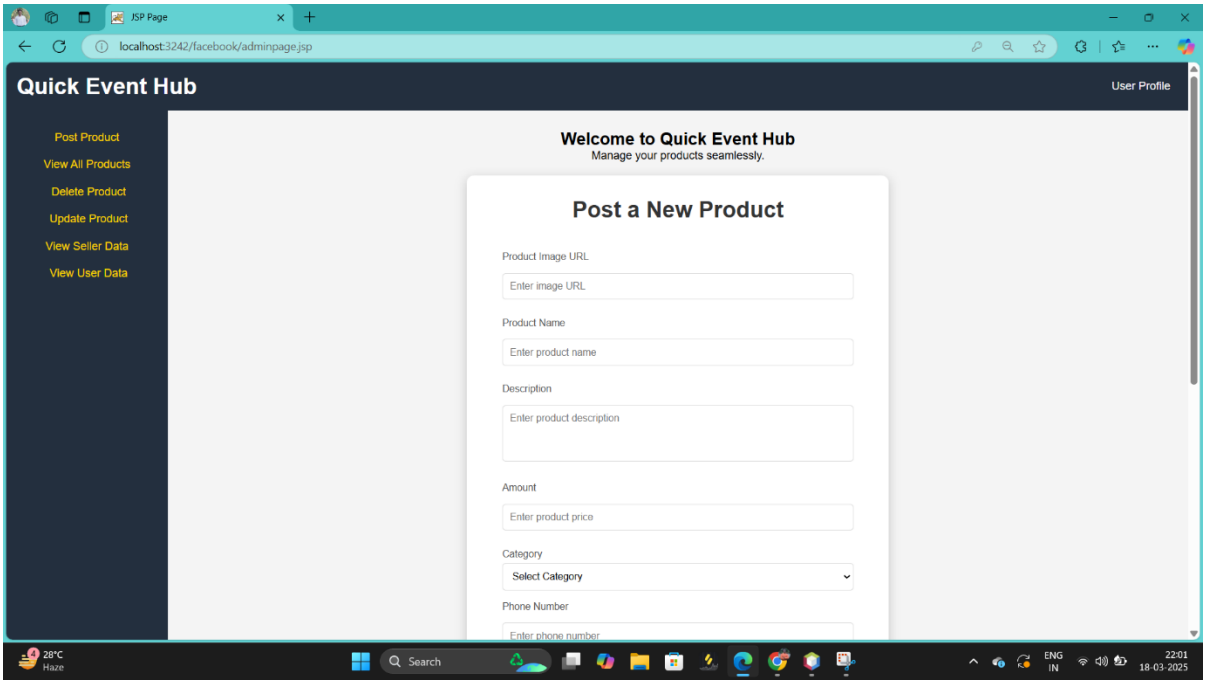
4.2. Provider Dashboard

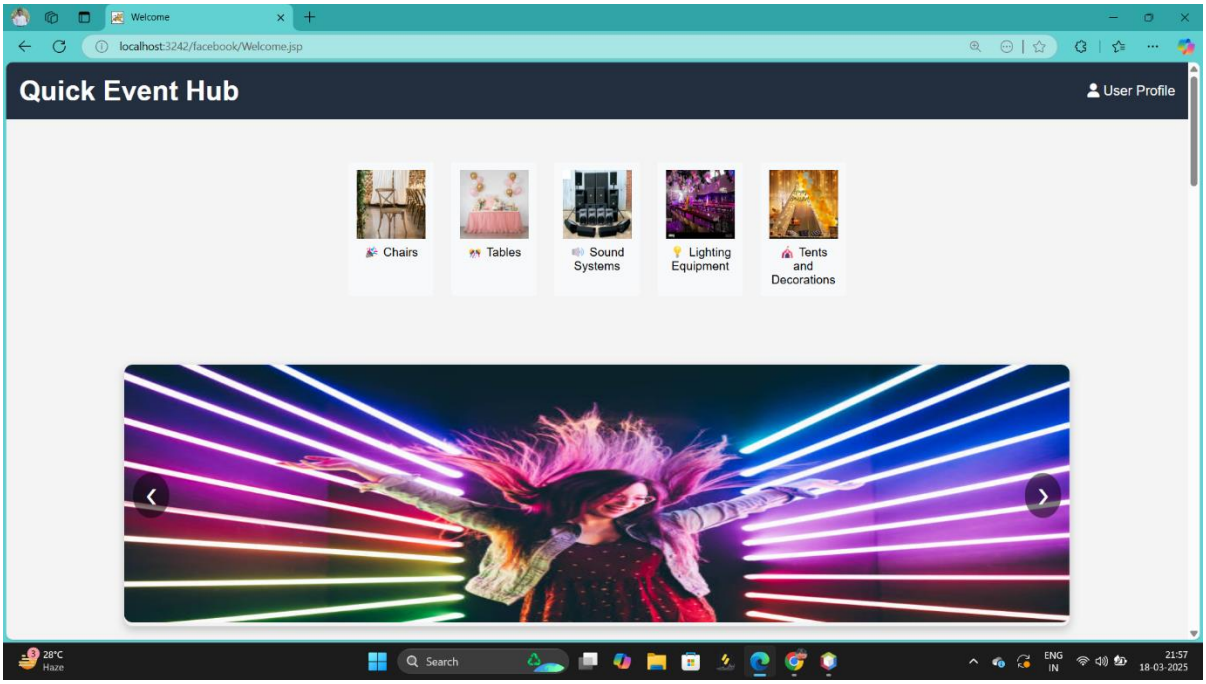
Components:

- **Service Listings**: A list of the provider's registered services (catering, rentals) with options to edit or remove services.
- **New Service**: A button for adding a new service (opens service registration form).
- **Pending Requests**: A section showing booking requests from organizers, with options to accept or reject them.
- **Notifications Panel**: Updates on booking requests and payments.

Visuals:

- Tabular layout for managing services (e.g., service name, type, price, availability).
- Buttons for managing services should be easily accessible.


JSP Page
localhost:3242/facebook/adminpage.jsp
Quick Event Hub
User Profile
Post Product
View All Products
Delete Product
Update Product
View Seller Data
View User Data
Welcome to Quick Event Hub
Manage your products seamlessly.
Post a New Product
Product Image URL
Enter image URL
Product Name
Enter product name
Description
Enter product description
Amount
Enter product price
Category
Select Category
Phone Number
Enter phone number
28°C
Haze
Search
ENG
IN
22:01
18-03-2025


Welcome
localhost:3242/facebook/Welcome.jsp
Quick Event Hub
User Profile
Chairs
Tables
Sound Systems
Lighting Equipment
Tents and Decorations
28°C
Haze
Search
ENG
IN
21:57
18-03-2025

5. Service Search Page (For Organizers)

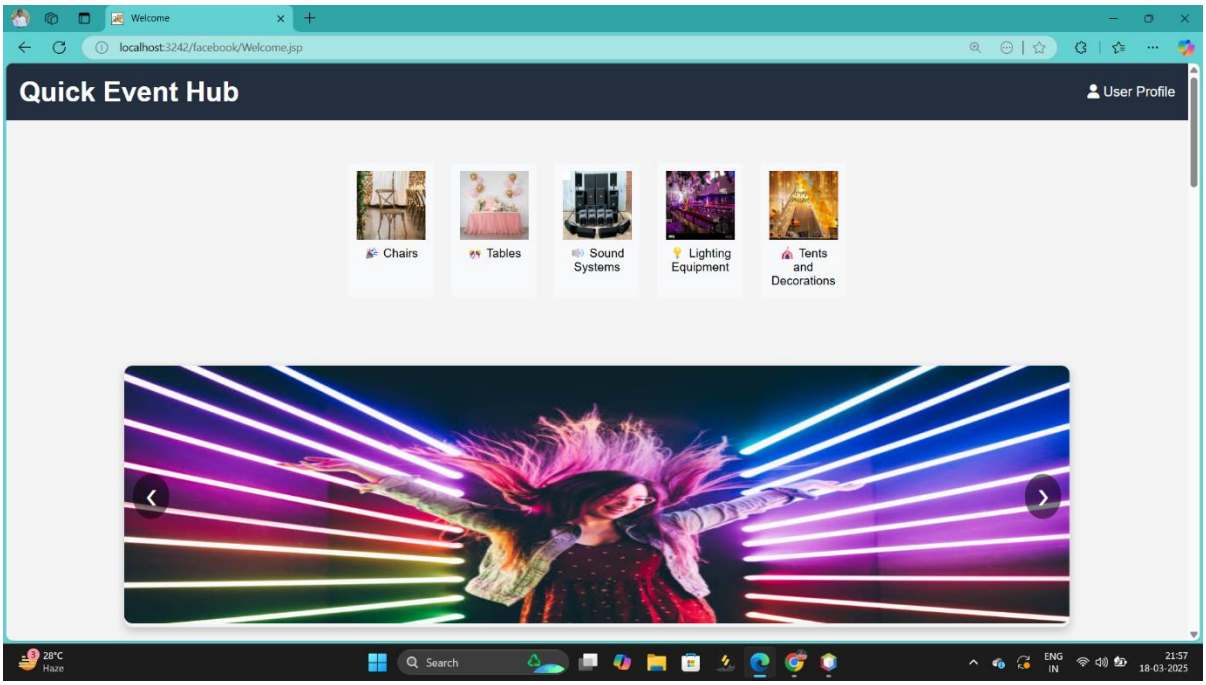
This page allows event organizers to search for rentals and catering services.

Components:

- **Search Bar**: Allows filtering based on service type (rentals/catering), price range, and availability.
- **Service Listings**:
    - Grid or list format of available services based on the search filters.
    - Each service card shows the service name, image, description, price, and a "Book Now" button.
- **Service Details Page**:
    - Clicking a service opens a detailed view with more information (service provider, terms, and conditions, customer reviews).
    - "Book Now" button with an integrated form to provide event details (date, location, etc.).

Visuals:

- Clean grid design for displaying services.
- Filters at the top for easy searching.



6. Booking Confirmation Page

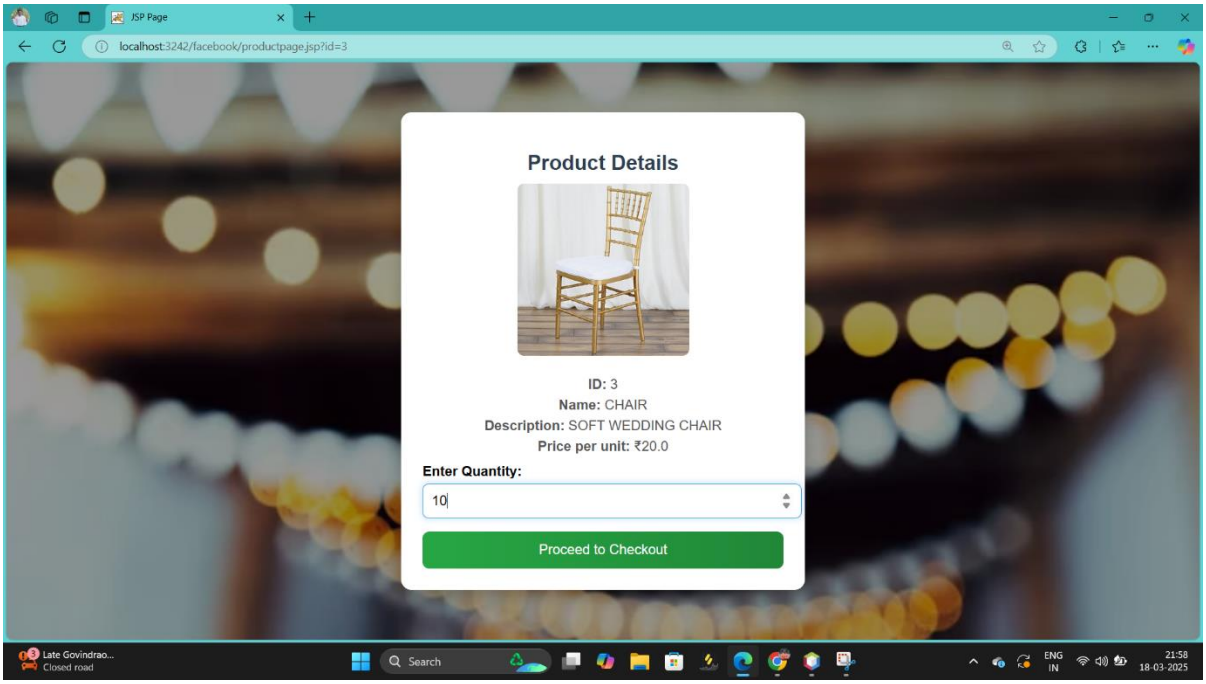
Once an organizer selects a service, the booking confirmation page confirms the details.

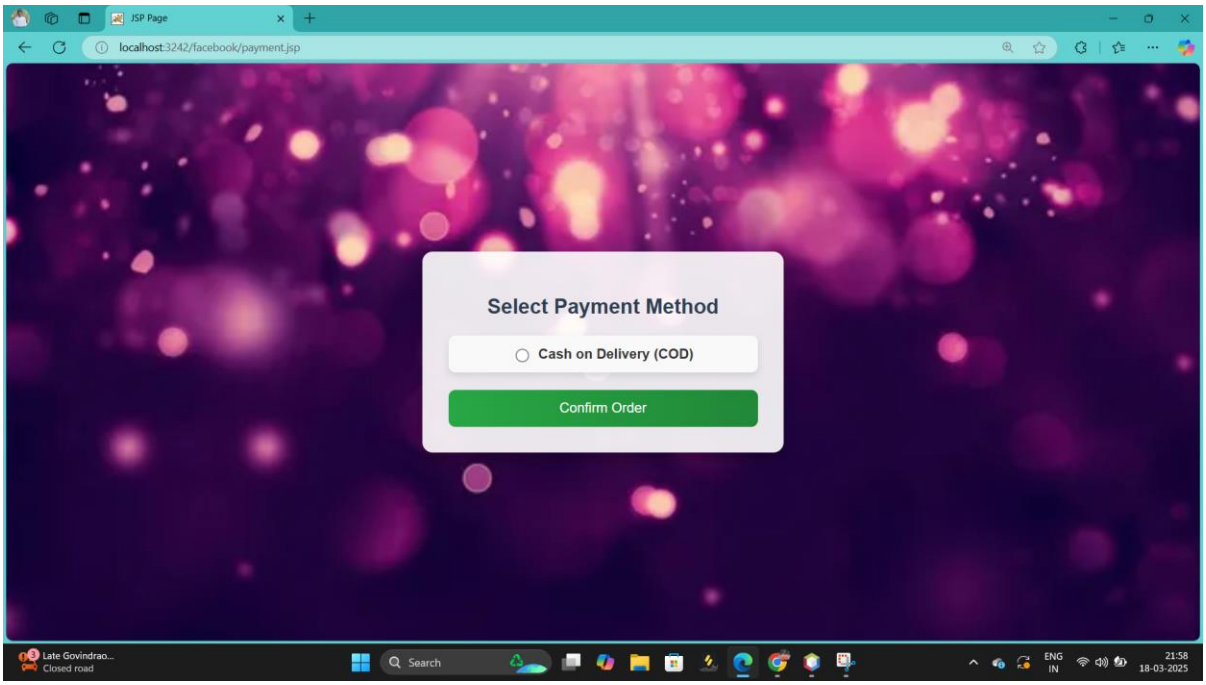
Components:

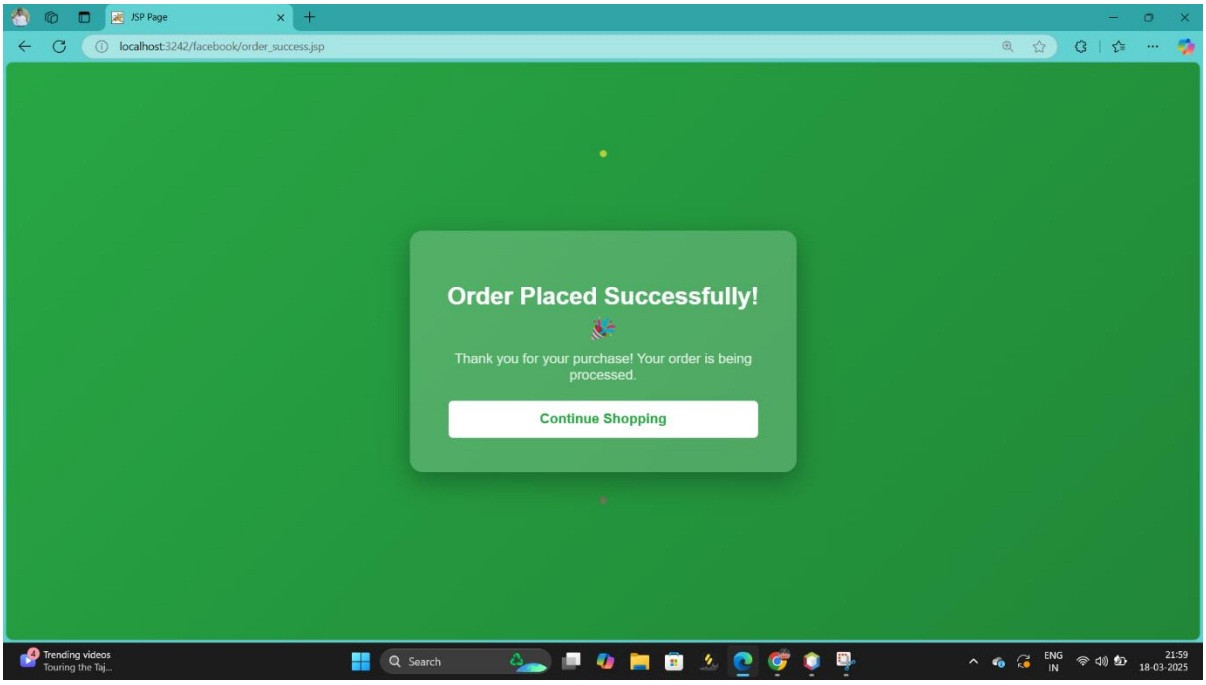
- **Service Summary**: Shows details of the selected service.
- **Event Details Form**: Organizer fills in event date, time, location, and any special instructions.
- **Payment Option**: Integration with a payment gateway for immediate payment.
- **Confirm Button**: Finalize the booking.

Visuals:

- Simple and clear layout focusing on confirmation and accuracy of the booking.
- A progress bar to show how far along the user is in the booking process.




JSP Page
localhost:3242/facebook/payment.jsp
Select Payment Method
Cash on Delivery (COD)
Confirm Order
Late Govindrao...
Closed road
Search
ENG
IN
21:58
18-03-2025


JSP Page
localhost:3242/facebook/order_success.jsp
Order Placed Successfully!
Thank you for your purchase! Your order is being processed.
Continue Shopping
Trending videos
Touring the Taj...
Search
ENG
IN
21:59
18-03-2025

7. Service Registration Page (For Providers)

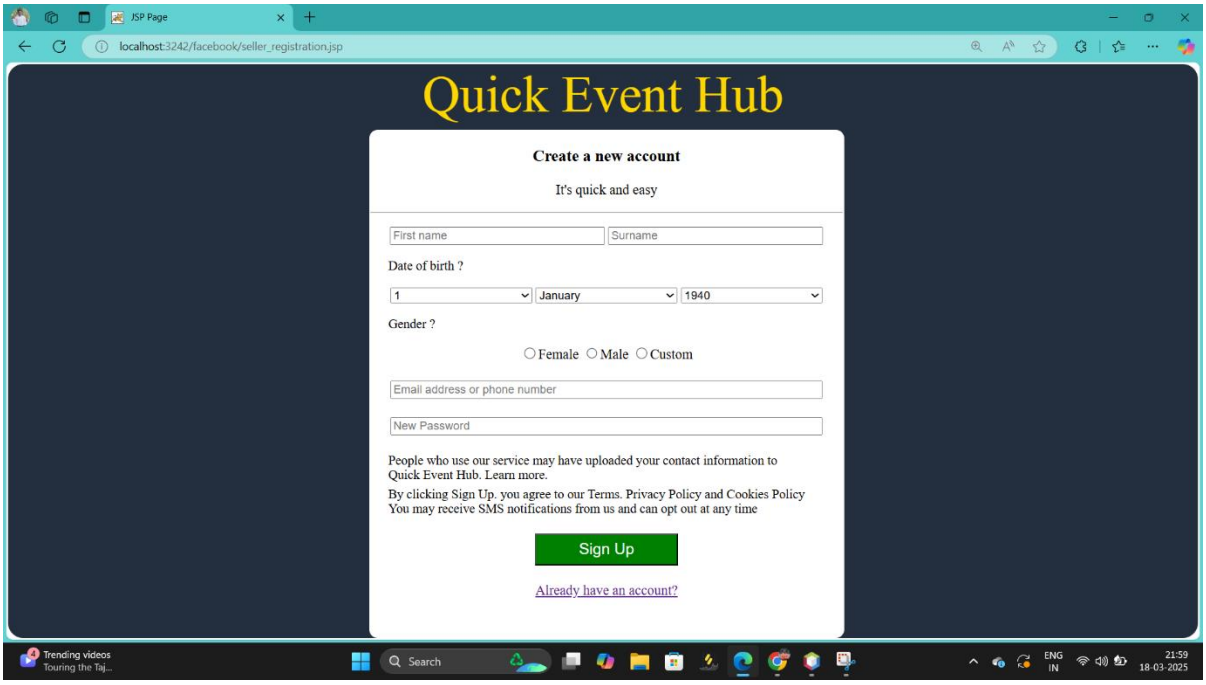
Service providers can use this page to register their services.

Components:

- **Form Fields**:
    - Service name, service type (dropdown: catering, rental), price, availability (dates), and description.
    - Image upload option for showcasing the service.
- **Submit Button**: A clear call-to-action button to register the service.

Visuals:

- Well-spaced form with clear labels and instructions for uploading images or filling in details.



8. Payment Page

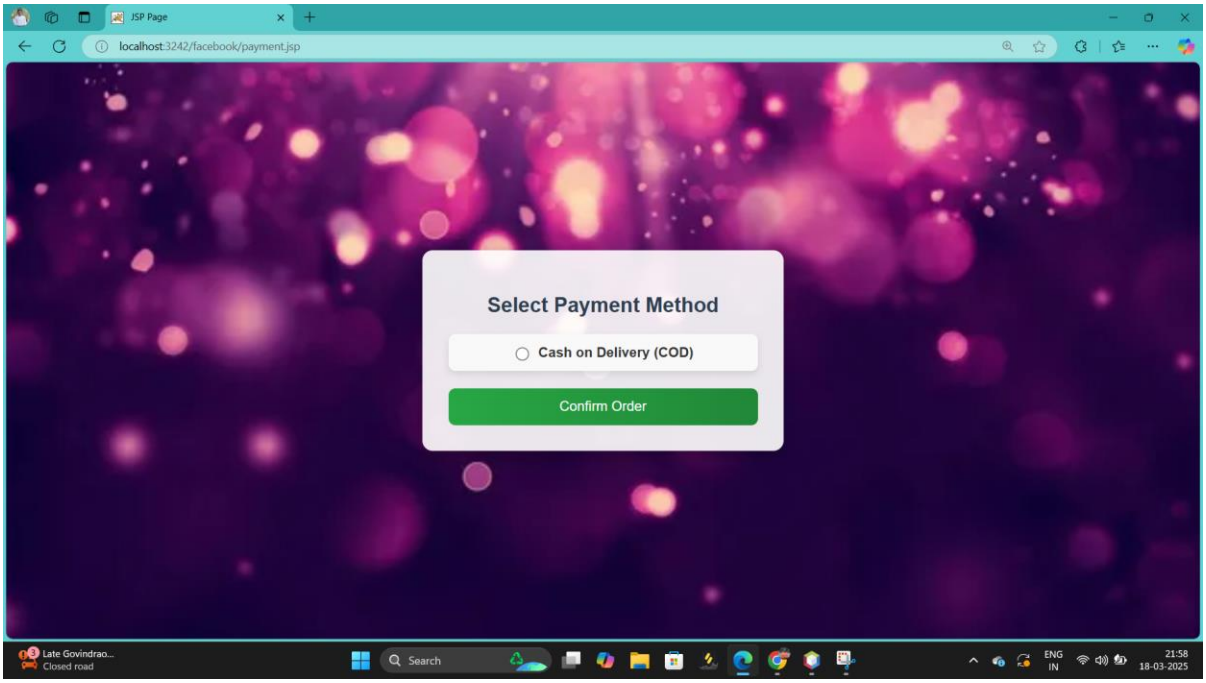
Once a booking is made, organizers are directed to the payment page.

Components:

- **Payment Methods**: Only select payment method is Cash-on-delivery.
- **Submit** : Button to confirm order.

Visuals:

- Professional and minimal design focusing on clarity and security of the payment process.



9. Admin Panel

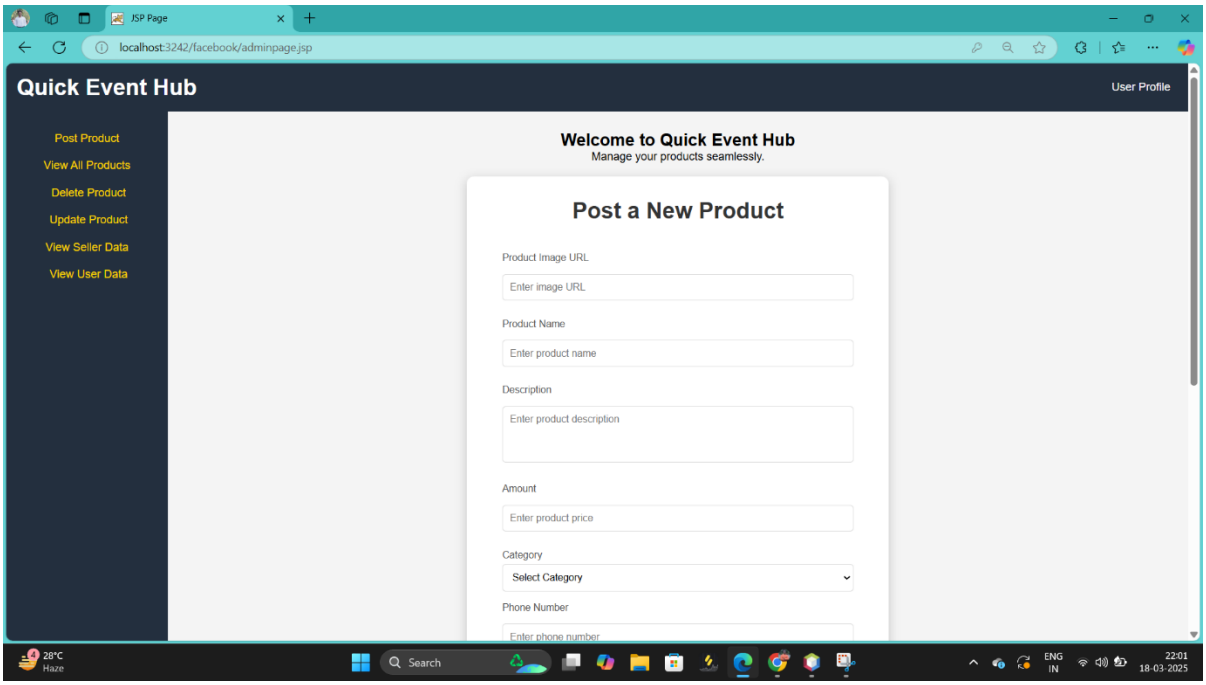
The admin panel manages user requests, bookings, and system settings.

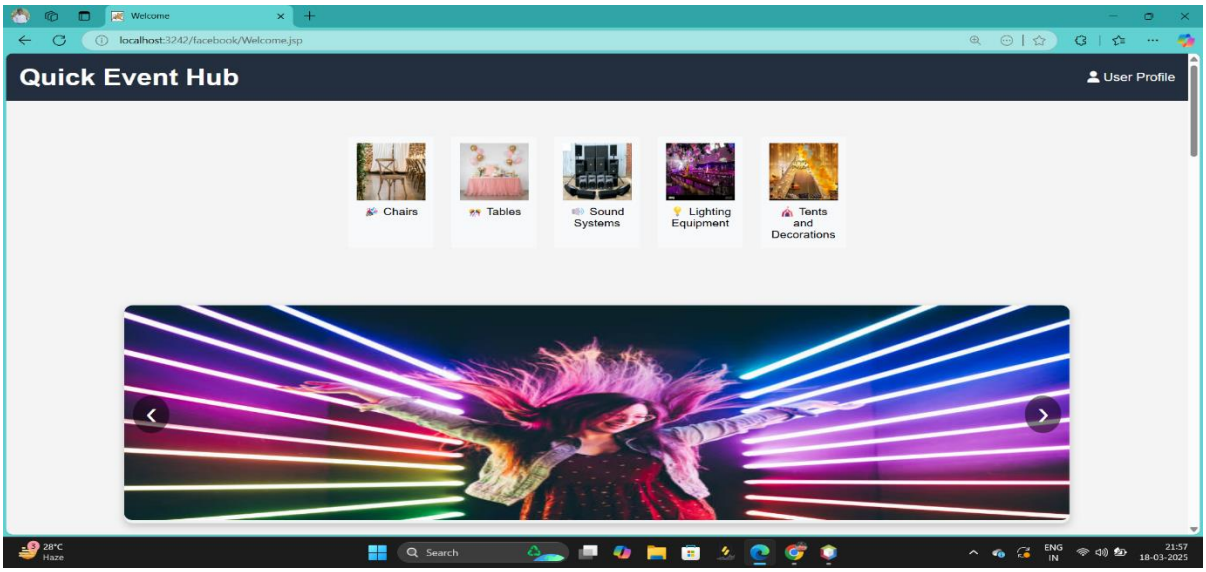
Components:

- **User Management**: Manage and view all registered users (organizers and providers).
- **Booking Management**: View, approve, or reject bookings.
- **Notification Management**: Send notifications to both providers and organizers.
- **Reports and Analytics**: View system usage reports and performance analytics.

Visuals:

➢ Dashboard-style layout with easy access to various admin functions.





5. Security Issues

1. User Authentication and Authorization

Issue:

- Users (event organizers, service providers, and admin) access the platform via login systems. Without proper security mechanisms, attackers could potentially gain unauthorized access.

Potential Threats:

- **Brute-force Attacks**: Hackers may attempt to guess user credentials.
- **Session Hijacking**: If sessions are not properly secured, attackers can hijack user sessions.

Mitigation Strategies:

- Implement **strong password policies** (minimum length, complexity requirements).
- Use **two-factor authentication (2FA)** to provide an extra layer of security.
- Enforce **secure session management** using session tokens and timeouts.
- Ensure role-based access control (RBAC) is implemented so users can only access their own data and features.

2. Injection Attacks (SQL Injection, Cross-Site Scripting)

Issue:

- Input forms, such as service registration, booking requests, and user profiles, can be vulnerable to injection attacks.

Potential Threats:

- **SQL Injection (SQLi)**: Attackers may exploit poorly sanitized user input to execute malicious database queries.
- **Cross-Site Scripting (XSS)**: Attackers may inject malicious scripts into forms or comments that can compromise user data.

Mitigation Strategies:

- Use **prepared statements and parameterized queries** to prevent SQL injection.

- Implement **input validation and sanitization** on all user inputs.
- Use **Content Security Policy (CSP)** headers to protect against XSS attacks.
- Ensure that the application properly escapes and sanitizes output before rendering it to the client.

3. Data Privacy and GDPR Compliance

Issue:

- Handling user data (personal details, event preferences, etc.) requires strict adherence to privacy laws such as the General Data Protection Regulation (GDPR).

Potential Threats:

- **Data Misuse**: Unauthorized sharing or use of personal data without user consent.
- **Non-Compliance Penalties**: Failure to comply with data privacy regulations could result in legal penalties.

Mitigation Strategies:

- Implement a clear **privacy policy** and ensure users consent to how their data will be used.
- Allow users to **opt-in** or **opt-out** of data sharing and marketing communications.
- Provide users with the ability to **request data deletion** or data export as per GDPR requirements.
- Regularly review and audit the system to ensure **compliance with data protection laws**.

4. Admin Panel Security

Issue:

- The admin panel controls user management, booking approvals, and overall system management. If compromised, attackers could gain full control of the platform.

Potential Threats:

- **Privilege Escalation**: Attackers could gain admin-level access through vulnerabilities.
- **Unauthorized Admin Access**: Weak admin credentials could allow hackers to take over the platform.

Mitigation Strategies:

- Ensure **strong access control policies** for admin accounts, including **multi-factor authentication (MFA)**.

- Restrict admin access to specific IP addresses or through VPN.
- Regularly audit admin actions to detect any suspicious activity.

5. Denial of Service (DoS) Attacks

Issue:

- A Denial of Service (DoS) or Distributed Denial of Service (DDoS) attack could overload the system, rendering it unusable for users.

Potential Threats:

- **Service Downtime**: Overwhelming the server with requests to crash the system.
- **Slow Performance**: High traffic from a DDoS attack could slow down the application, impacting user experience.

Mitigation Strategies:

- Implement **DDoS protection** using a Content Delivery Network (CDN) or services like Cloudflare.
- Use **rate limiting** to restrict the number of requests from a single IP address in a given timeframe.
- Regularly monitor the system for unusual traffic spikes.

6. Test Cases Design

1. Test Case Categories

The test cases are divided into the following categories:

1. **Functional Testing**
2. **User Interface Testing**
3. **Security Testing**
4. **Performance Testing**
5. **Usability Testing**
6. **Compatibility Testing**
7. **Regression Testing**

2. Functional Testing

2.1 User Registration and Login

Test Case ID	**TC-001**
Test Case Name	User Registration with Valid Data
Precondition	User is on the registration page.
Test Steps	1. Fill in all mandatory fields with valid data. 2. Click on the "Sign Up" button.
Expected Result	User is successfully registered and redirected to the dashboard.
Actual Result	[To be filled after test execution]
Status	[Pass/Fail]

Test Case ID	**TC-002**
Test Case Name	User Login with Valid Credentials
Precondition	User has a registered account.
Test Steps	1. Navigate to the login page. 2. Enter valid email and password. 3. Click on the "Login" button.
Expected Result	User is logged in and redirected to the dashboard.
Actual Result	[To be filled after test execution]
Status	[Pass/Fail]

2.2 Service Search and Booking

Test Case ID	**TC-003**
Test Case Name	Search for Services
Precondition	User is logged in as an organizer.
Test Steps	1. Navigate to the "Search Services" page. 2. Enter search criteria (e.g., service type, location). 3. Click on the "Search" button.
Expected Result	A list of relevant services is displayed based on the search criteria.
Actual Result	[To be filled after test execution]
Status	[Pass/Fail]

Test Case ID	**TC-004**
Test Case Name	Book a Service
Precondition	User has searched for services.
Test Steps	1. Select a service from the search results. 2. Fill in event details (date, location). 3. Click on the "Book Now" button.
Expected Result	Booking is confirmed, and user receives a confirmation message.
Actual Result	[To be filled after test execution]
Status	[Pass/Fail]

2.3 Service Provider Registration

Test Case ID	**TC-005**
Test Case Name	Service Provider Registration
Precondition	User is on the provider registration page.
Test Steps	1. Fill in all mandatory fields with valid service data. 2. Click on the "Register" button.
Expected Result	Service provider is successfully registered and redirected to their dashboard.
Actual Result	[To be filled after test execution]
Status	[Pass/Fail]

3. User Interface Testing

Test Case ID	**TC-006**
Test Case Name	Verify Responsive Design
Precondition	Access the application on different devices (desktop, tablet, mobile).

Test Steps	1. Open the application on each device. 2. Navigate through different pages.
Expected Result	UI elements should adapt to different screen sizes without loss of functionality.
Actual Result	[To be filled after test execution]
Status	[Pass/Fail]

Test Case ID	**TC-007**
Test Case Name	Verify Navigation Menus
Precondition	User is logged in.
Test Steps	1. Click on each menu item in the navigation bar. 2. Verify redirection to the correct page.
Expected Result	Each menu item should lead to the expected page without errors.
Actual Result	[To be filled after test execution]
Status	[Pass/Fail]

4. Security Testing

Test Case ID	**TC-008**
Test Case Name	Test SQL Injection Protection
Precondition	User is on a service search or booking page.
Test Steps	1. Enter SQL injection strings into the search or booking fields. 2. Submit the form.
Expected Result	The application should not process the injection and should return an error message or no results.
Actual Result	[To be filled after test execution]
Status	[Pass/Fail]

Test Case ID	**TC-009**
Test Case Name	Verify Password Reset Security

Precondition	User is on the login page.
Test Steps	1. Click on "Forgot Password". 2. Enter the registered email address. 3. Check the email for reset instructions.
Expected Result	The user receives a password reset email with a secure link.
Actual Result	[To be filled after test execution]
Status	[Pass/Fail]

5. Performance Testing

Test Case ID	**TC-010**
Test Case Name	Load Testing on Service Search
Precondition	Performance testing tool is configured.
Test Steps	1. Simulate multiple users (e.g., 100, 200) searching for services simultaneously.
Expected Result	The application should handle the load without performance degradation (response time < 3 seconds).
Actual Result	[To be filled after test execution]
Status	[Pass/Fail]

Test Case ID	**TC-011**
Test Case Name	Stress Testing the Booking System
Precondition	Performance testing tool is configured.
Test Steps	1. Gradually increase the number of users making bookings until the system fails.
Expected Result	The system should handle a high number of concurrent bookings without crashing.
Actual Result	[To be filled after test execution]
Status	[Pass/Fail]

6. Usability Testing

Test Case ID	**TC-012**
Test Case Name	Evaluate User Navigation
Precondition	Usability testing environment is set up with participants.
Test Steps	1. Have users complete specific tasks (e.g., book a service, register as a provider). 2. Observe and record time taken and any difficulties encountered.
Expected Result	Users should complete tasks efficiently without confusion.
Actual Result	[To be filled after test execution]
Status	[Pass/Fail]

7. Compatibility Testing

Test Case ID	**TC-013**
Test Case Name	Verify Compatibility with Different Browsers
Precondition	Application is deployed.
Test Steps	1. Open the application in different browsers (Chrome, Firefox, Safari, Edge). 2. Navigate through the application.
Expected Result	The application should function correctly in all tested browsers without layout or functionality issues.
Actual Result	[To be filled after test execution]
Status	[Pass/Fail]

8. Regression Testing

Test Case ID	**TC-014**

Test Case Name	Verify Previous Functionality after Update
Precondition	New version of the application is deployed.
Test Steps	1. Execute previously passed test cases for user registration, login, and service booking.
Expected Result	All previously passing test cases should pass without errors in the new version.
Actual Result	[To be filled after test execution]
Status	[Pass/Fail]

CHAPTER 5: IMPLEMENTATION AND TESTING

5.1 Implementation Approaches

The implementation of QuickEvent Hub follows the Modular Development Approach, dividing the application into manageable modules: Front-End, Back-End, and Database. Each module is developed and tested independently before integration.

Development Strategy:

- **Front-End Implementation:** Utilizing JSP for dynamic content rendering, CSS for styling, and JavaScript for interactivity. JSP pages handle the presentation layer while connecting with backend Java Servlets.
- **Back-End Implementation:** Java Servlets handle request-response cycles, process business logic, and interact with the MySQL database. Secure and efficient coding practices, including error handling and session management, are applied.
- **Database Implementation:** MySQL is used for structured data storage, maintaining user records, event details, and service provider information. Proper indexing ensures quick data retrieval.

This structured approach ensures an efficient development process and aligns with the project's goals of delivering a functional, user-friendly event management platform.

5.2 Coding Detail and Code Efficiency

In this section, we delve deeper into the coding practices, structure, and efficiency measures applied during the development of QuickEvent Hub.

Coding Standards and Practices:

- **Modularity:** The project is divided into logical modules—User Management, Event Management, Service Provider Management—promoting organized, readable, and maintainable code.
- **OOP Principles:** Object-Oriented Programming is employed to create reusable and flexible classes.
- **Exception Handling:** Comprehensive error handling using try-catch blocks to manage runtime errors and maintain stability.
- **Secure Coding:** Validation techniques are implemented to prevent SQL injection, XSS attacks, and ensure secure data transactions.

Efficient Code Design:

- **Optimized Database Access:** Use of prepared statements for efficient database access, minimizing latency and preventing SQL injection.
- **Minimized Redundancy:** Efficient coding techniques are applied to eliminate code duplication, making maintenance easier.
- **Efficient Looping and Iteration:** Enhanced for-loops and stream API in Java for optimal iteration over collections.

- **Welcome.jsp (dashboard)**

```
<%@ page import="java.sql.*" %>
<%@ page import="java.util.*" %>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@ page import="jakarta.servlet.http.HttpSession" %>

<!DOCTYPE html>
<html>
<head>
   <title>Welcome</title>
    <%--   <marquee>😊 WELCOME TO QUICK EVENT HUB MAKE YOUR EVENT
     MEMORABLE 😊 </marquee>--%>
   <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-
     awesome/6.4.2/css/all.min.css">
    <link rel="stylesheet" href="click.css">
     <link rel="stylesheet" href="slider.css">
      <link rel="stylesheet" href="footer.css">

   <script src="https://kit.fontawesome.com/a076d05399.js" crossorigin="anonymous"></script>
      <script defer src="script.js"></script>
   <style>
      body {
         font-family: Arial, sans-serif;
         margin: 0;
         padding: 0;
      }
      .header {
         background-color: #232f3e;
         color: white;
         padding: 15px;
         display: flex;
         justify-content: space-between;
         align-items: center;
      }
      .header h1 {
         margin: 0;
      }
      .profile-menu {
         position: relative;
         display: inline-block;
      }
      .profile-menu button {
         background: none;
         border: none;
         color: white;
         cursor: pointer;
         font-size: 16px;
      }
      .dropdown-menu {
         display: none;
         position: absolute;
         right: 0;
```

```
Quantity {
        font-size: 16px;
        color: #444;
        margin-top: 10px;
    }
    .Phone_no{
        font-size: 15px;
        color: #555;
        margin-top: 10px;
    }
    .product-description {
        font-size: 14px;
        color: #666;
        margin: 10px 0;
    }
    .product-price {
        font-size: 16px;
        font-weight: bold;
        color: #27ae60;
    }
    .product-btn {
        display: block;
        width: 100%;
        background: #232f3e;
        color: white;
        padding: 10px;
        border: none;
        border-radius: 5px;
        font-size: 16px;
        cursor: pointer;
        margin-top: 10px;
    }
    .product-btn:hover {
        background: #1a252f;
    }
    .mainv{
        background: #fff;
        border-radius: 10px;
        box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
        overflow: hidden;
        width: 98%;
        margin: auto;
        transition: transform 0.3s ease-in-out;
        text-align: center;
        padding: 30px;
    }


    </style>
</head>


<body>
  <div class="header">
        <h1>Quick Event Hub</h1>
```

```
        <div class="profile-menu">
            <button><i class="fas fa-user"></i> User Profile</button>
            <div class="dropdown-menu">

                <a href="editProfile.jsp">Edit Profile</a>
                <a href="selectLanguage.jsp">Select Language</a>
                <hr>
                <a href="LogOut">Log Out</a>
            </div>
        </div>
    </div>
    <br>


     <div class="category-container">
    <a href="Welcome.jsp?category=Chairs" class="category">
        <img src="https://s.alicdn.com/@sc04/kf/H08e37fb10c4a4184a19b4e9f7ae675b3h.png"
     alt="🎉 Chairs">
        <span>🎉 Chairs</span>
    </a>
    <a href="Welcome.jsp?category=Tables" class="category">
        <img src="https://m.media-amazon.com/images/I/71fQB3wtSNL.jpg" alt="🎌 Tables">
        <span>🎌 Tables</span>
    </a>
    <a href="Welcome.jsp?category=Sound Systems" class="category">
        <img
     src="https://5.imimg.com/data5/SELLER/Default/2023/6/313928201/DM/MS/RC/190852559/pa-sound-system-500x500-500x500.jpg" alt="🔊 Sound Systems">
        <span>🔊 Sound Systems</span>
    </a>
    <a href="Welcome.jsp?category=Lighting Equipment" class="category">
        <img src="https://c8.alamy.com/comp/2JKNREP/a-purple-themed-decoration-on-stage-lighting-equipment-over-bar-chairs-and-table-for-party-event-2JKNREP.jpg" alt="💡 Lighting Equipment">
        <span>💡 Lighting Equipment</span>
    </a>
    <a href="Welcome.jsp?category=Tents and Decorations" class="category">
        <img src="https://m.media-amazon.com/images/I/71OsS8ZYbtL.jpg" alt="🎪 Tents and Decorations">
        <span>🎪 Tents and Decorations</span>
    </a>
</div>


   <div class="slider-container">
        <div class="slider">
            <div class="slide" id="slider">
                <img src="https://www.angara.com/blog/wp-content/uploads/2023/08/Make-Decor-Orange-ments-1-1.jpg"  alt="Slide 1 Chairs">



</div>
            <div class="slide" id="slider">
```

```
            <img src="https://thumbs.dreamstime.com/b/buffet-table-mini-hamburgers-snacks-
    canape-appetizers-luxury-wedding-reception-copy-space-serving-food-restaurant-
    233230473.jpg" alt="Slide 2">
        </div>
        <div class="slide" id="slider">
            <img
    src="https://static.vecteezy.com/system/resources/previews/034/087/358/non_2x/disco-party-
    vibe-colorful-speakers-in-the-background-creating-excitement-ai-generated-photo.jpg"
    alt="Slide 3">
        </div>
        <div class="slide" id="slider">
            <img src="https://media.istockphoto.com/id/1220706537/photo/dancing-
    girl.jpg?s=612x612&w=0&k=20&c=MNWxzjFeSV5_N2kbkzXJeMi9sErIDnT6nG-yABHb-
    l0=" alt="Slide4">
        </div>
        <div class="slide" id="slider">
            <img src="https://media.istockphoto.com/id/1415056757/photo/colorful-wedding-
    tents-at-
    night.jpg?s=612x612&w=0&k=20&c=tJ5NgHKJQs1gJuHTEpd1Uv6peTVxGs5puZUm9zp
    wT64=" alt="Slide5">
        </div>
      </div>
      <!-- Navigation Buttons -->
      <button class="prev" onclick="moveSlide(-1)">&#10094;</button>
      <button class="next" onclick="moveSlide(1)">&#10095;</button>
    </div>



    <br>
    <%
     Connection conn = null;
     PreparedStatement ps = null;
     ResultSet rs = null;
     String categoryFilter = request.getParameter("category");

     try {
        Class.forName("com.mysql.cj.jdbc.Driver");
        conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/facebook", "root",
      "123456");

        String query = "SELECT * FROM product";
        if (categoryFilter != null && !categoryFilter.isEmpty()) {
            query += " WHERE category=?";
        }

        ps = conn.prepareStatement(query);

        if (categoryFilter != null && !categoryFilter.isEmpty()) {
            ps.setString(1, categoryFilter);
        }



    rs = ps.executeQuery();
```

```
%>

<div class="product-container">
   <% while (rs.next()) { %>
      <a href='productpage.jsp?id=<%= rs.getString("id") %>'>
         <div class="product-card">
            <img class="product-image" src="<%= rs.getString("product_image_path") %>"
     alt="Product Image">
            <div class="product-title"><%= rs.getString("product_name") %></div>
            <div class="Quantity"><%= rs.getString("quantity") %></div>
            <div class="product-description"><%= rs.getString("product_description") %></div>
            <div class="product-price">&#8377 <%= rs.getDouble("product_amount") %></div>
            <div class="Phone_no"><%= rs.getString("phone_no") %></div>
         </div>
      </a>
   <% } %>
</div>

<%
   } catch (Exception e) {
      e.printStackTrace();
   } finally {
      if (rs != null) rs.close();
      if (ps != null) ps.close();
      if (conn != null) conn.close();
   }
%>

<div style="text-align: center; margin: 20px;">
   <a href="Welcome.jsp">
      <button class="product-btn">More Shopping</button>
   </a>
</div>

</body></html>
```

5.3 Test Approach

To ensure the functionality, performance, and reliability of QuickEvent Hub, a comprehensive testing approach was employed. Testing phases included unit testing, integration testing, and beta testing.

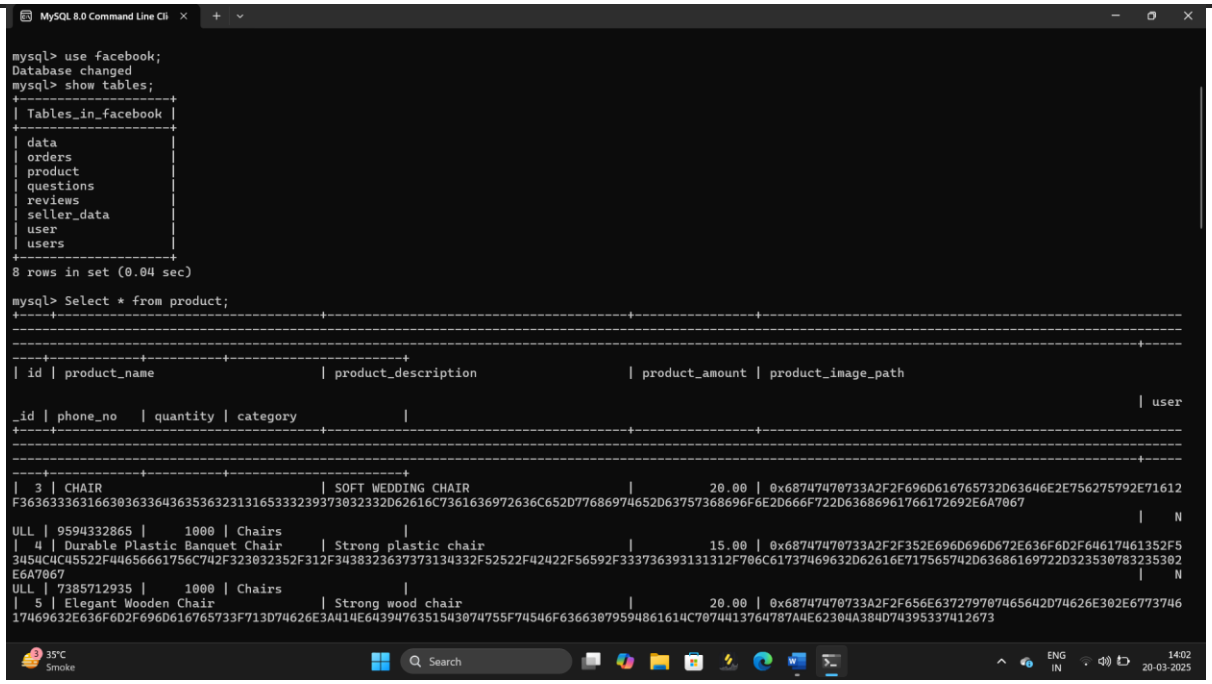
5.3.1 Unit Testing

Unit testing focuses on verifying individual components or modules to ensure they work correctly. For QuickEvent Hub, each class and function, such as user authentication and event management modules, were tested independently.

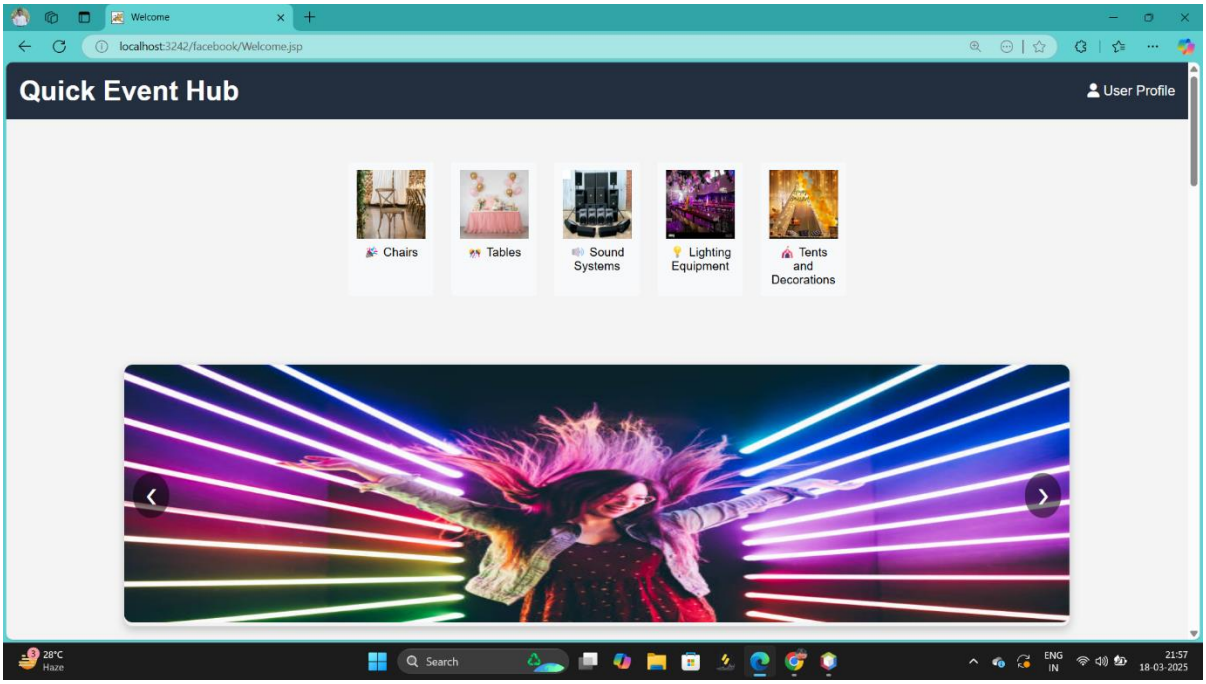
- Tools Used: JUnit for Java-based unit testing.
- Scope: Testing validation logic, input handling, and exception management.
- Outcome: Early identification of defects, ensuring individual modules operate as intended.

Database Creation:

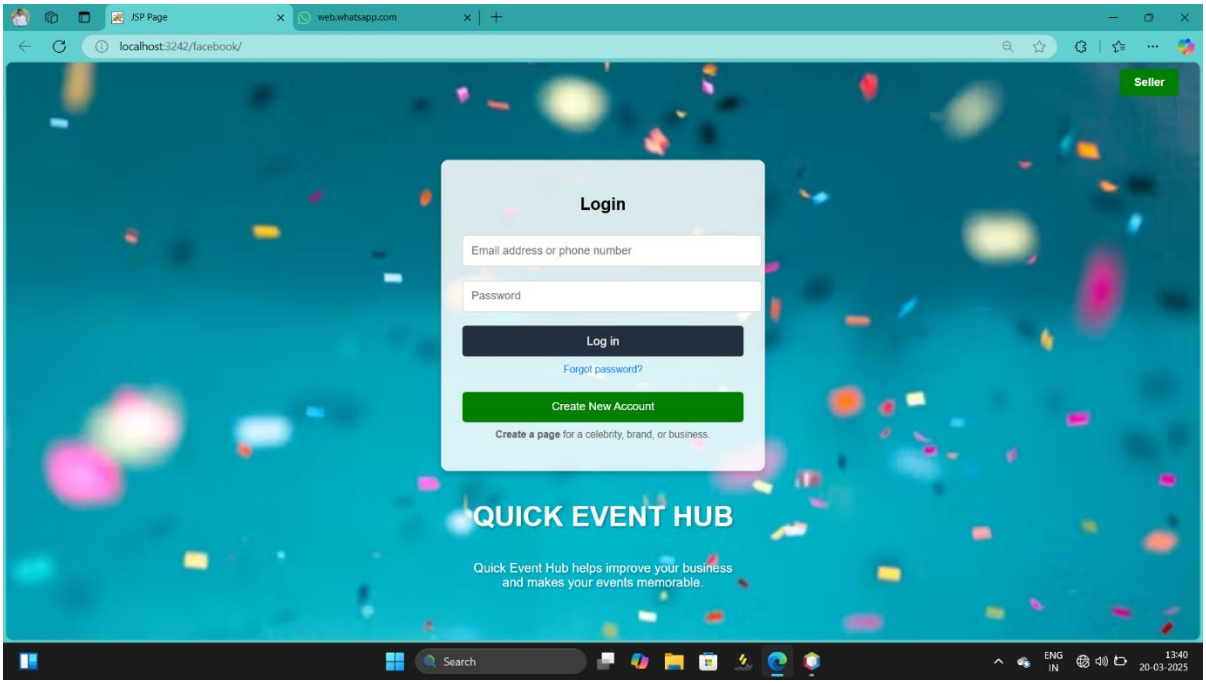
Install mysql through official website



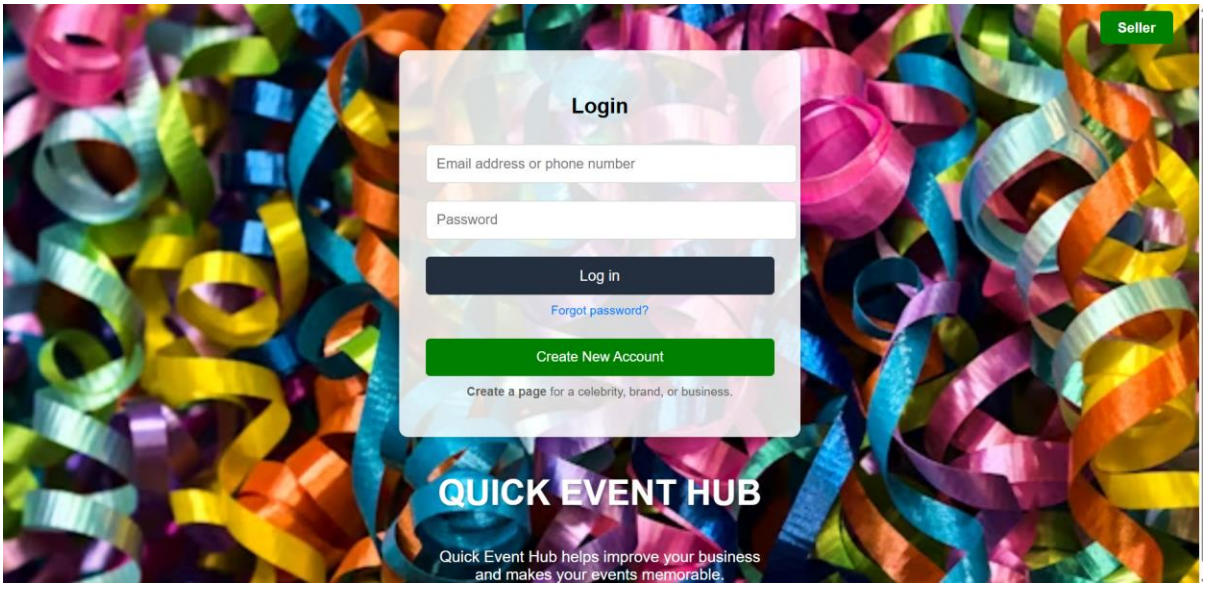
Dashboard (Welcome.jsp)



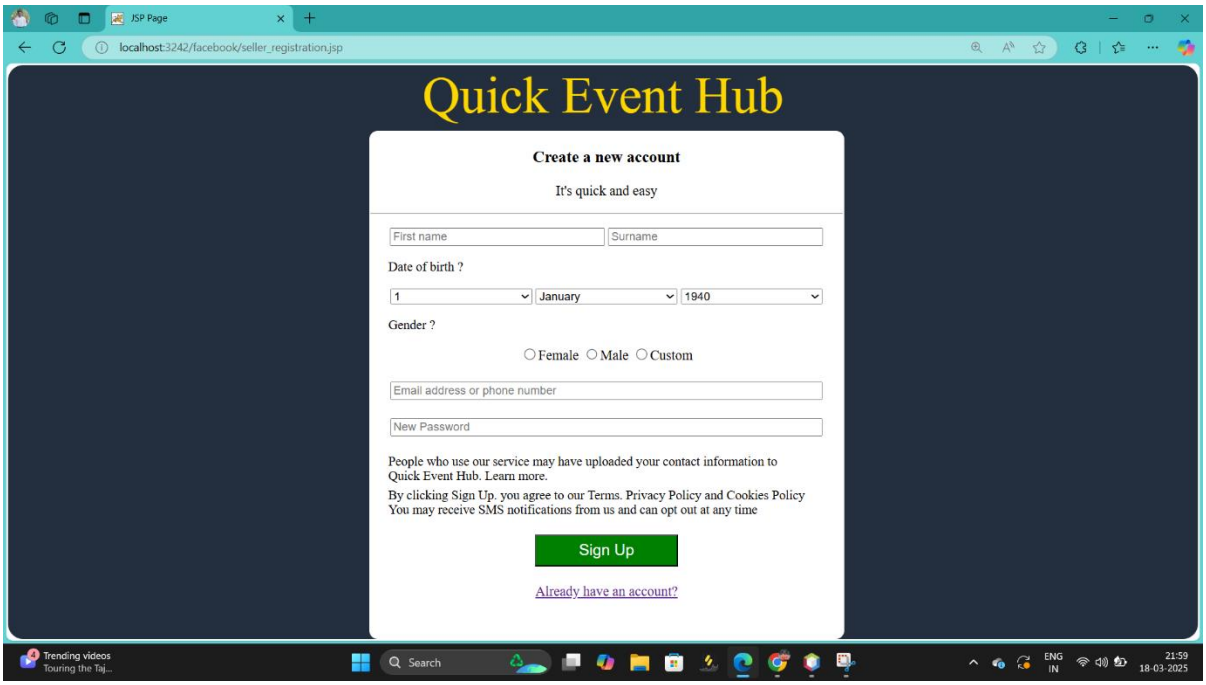
Login page (index.jsp)



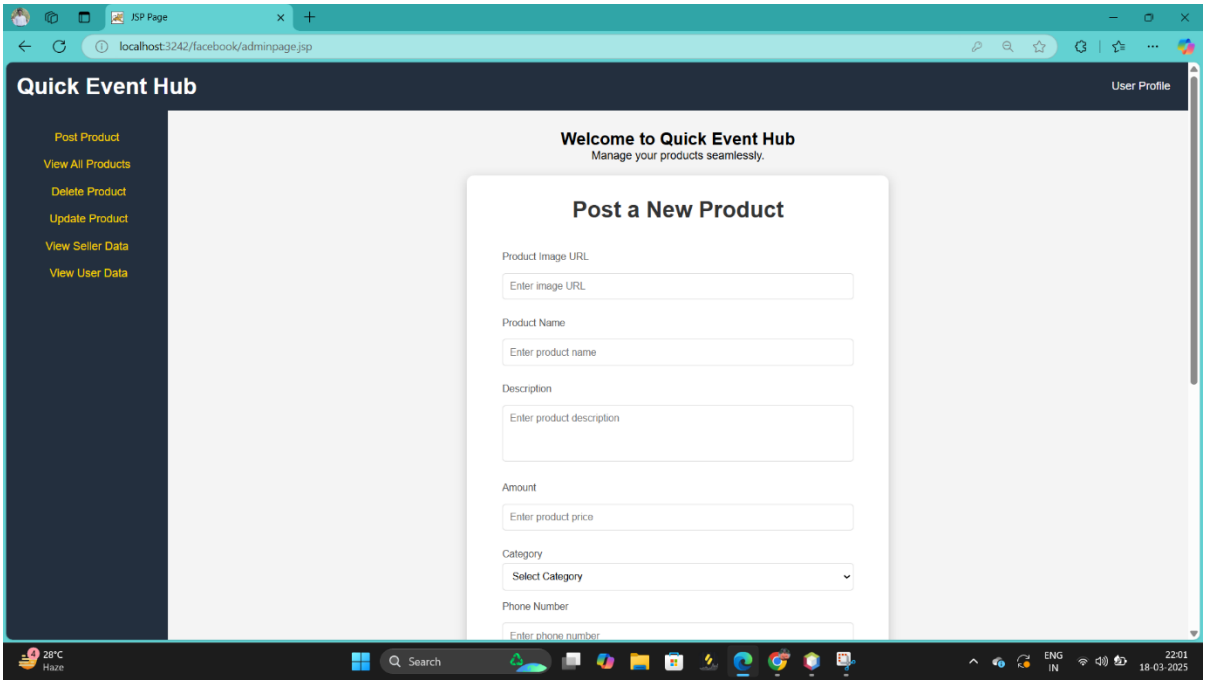
Seller Login Page(Seller.jsp)



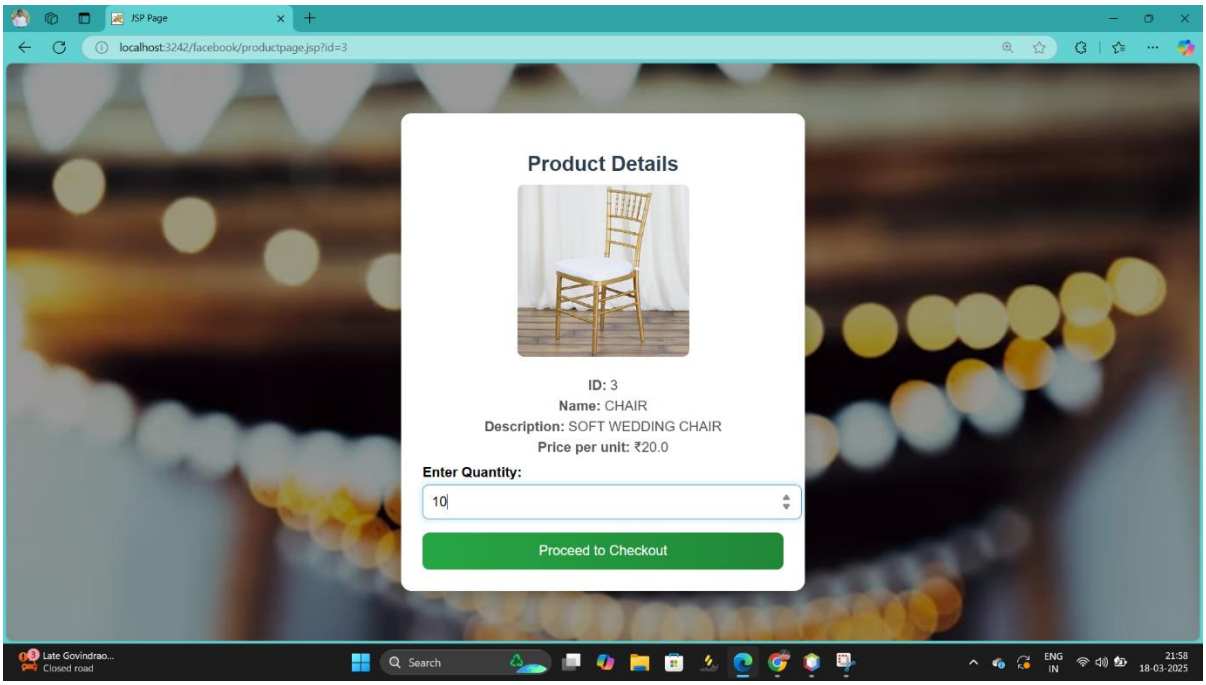
Registration page



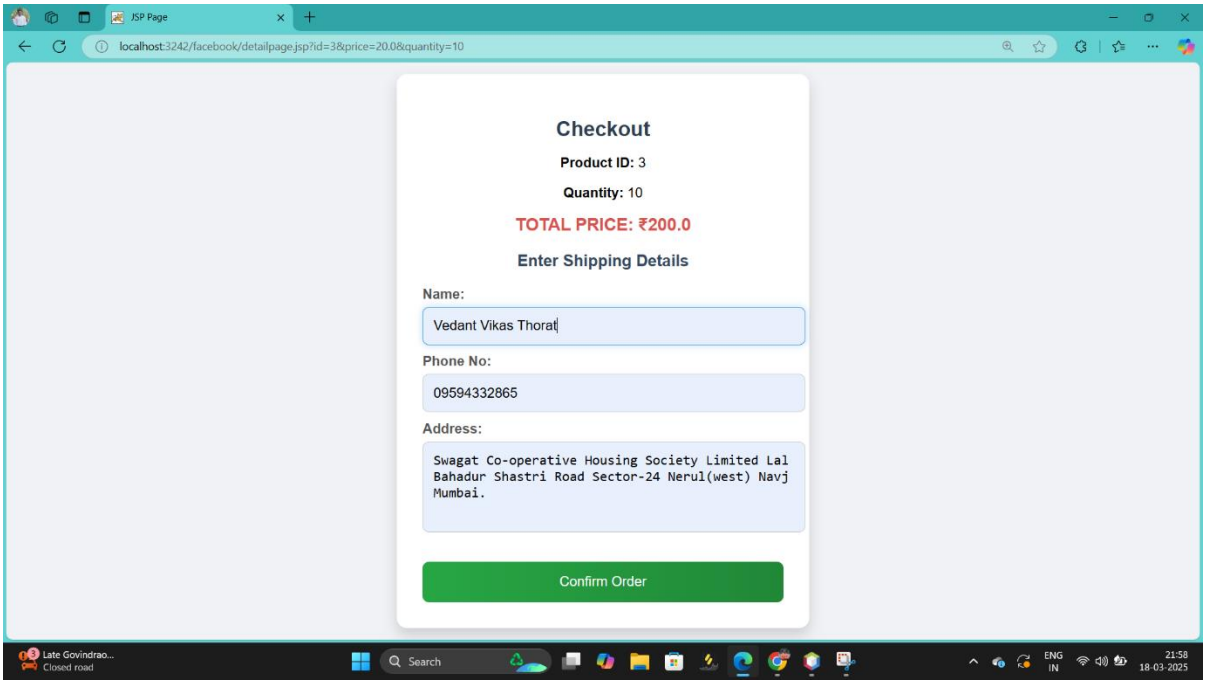
Admin page (post product)



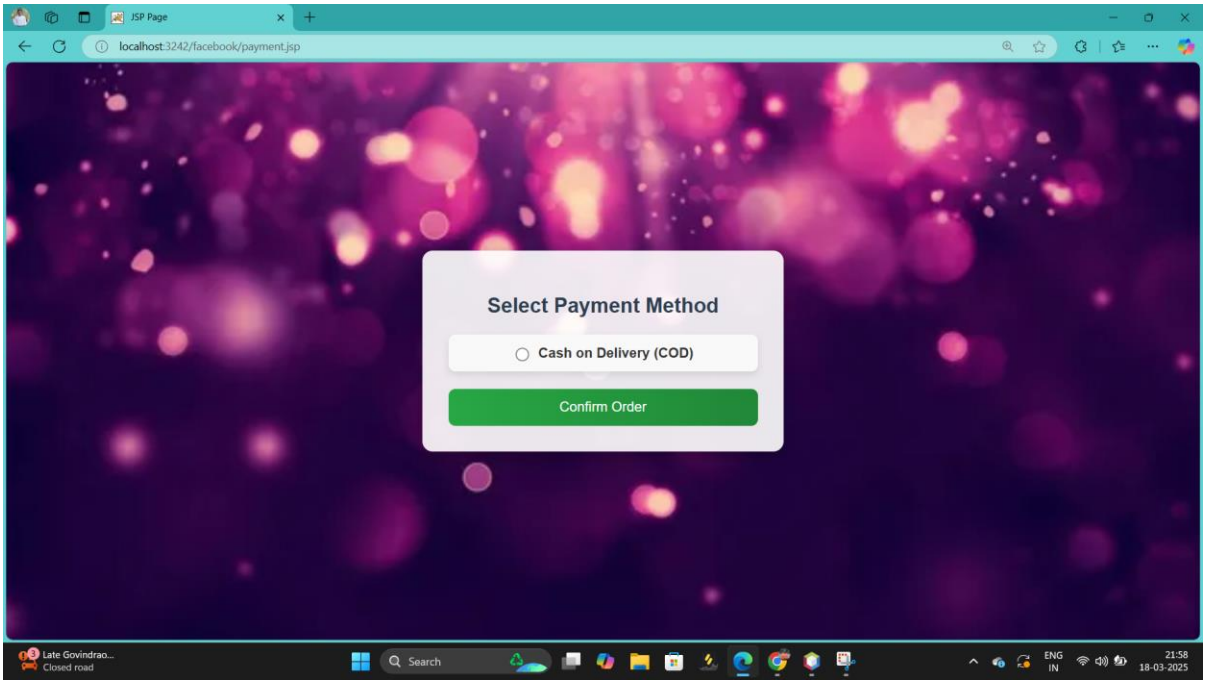
Booking page (After clicking a product)



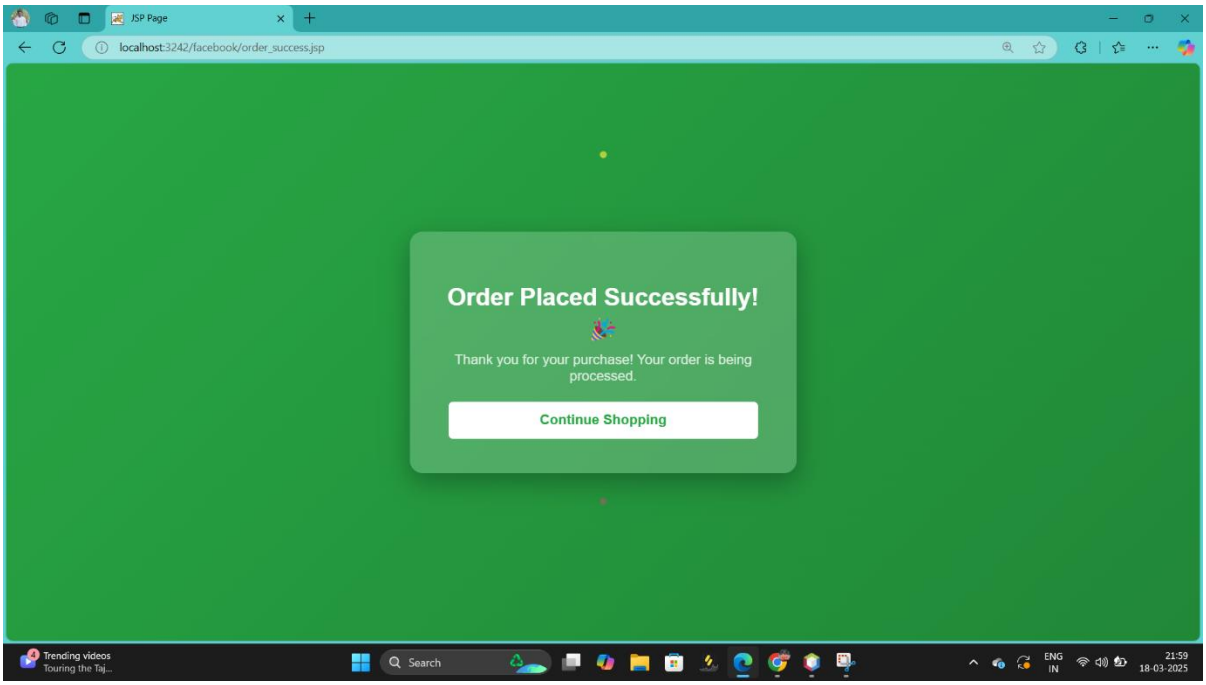
Shipping Detail page



Payment page



Order Successful page



5.3.2 Integration Testing

Integration testing verifies the interaction between modules to ensure smooth communication and data flow. In QuickEvent Hub, integration testing ensured front-end JSP pages correctly communicated with backend servlets and the database.

- Approach: Top-down integration testing.
- Focus: Testing data retrieval from MySQL, request handling by servlets, and accurate data display on JSP.
- Outcome: Ensured seamless integration and data consistency across the platform.

5.3.3 Beta Testing

Beta testing involved releasing the application to a limited user base to obtain feedback and identify any remaining issues.

- Participants: Small group of event organizers and service providers.
- Feedback Scope: Usability, functionality, and overall user experience.
- Outcome: Feedback-driven modifications were implemented to enhance user experience and resolve minor issues.

5.4 Modification and Improvements

During the testing and evaluation phases, several areas were identified for improvement to enhance the performance, security, and usability of QuickEvent Hub. Based on feedback from beta testers and stakeholders, modifications were made to address the identified challenges and optimize the system. The user interface was enhanced to provide a more intuitive experience, making navigation simpler for event planners and service providers. Improvements were made in database query optimization to reduce response time and enhance data retrieval speed. The authentication mechanism was reinforced to prevent unauthorized access, incorporating secure hashing techniques for storing sensitive data like passwords. The error-handling capabilities were expanded, ensuring informative feedback for users while preventing application crashes. Additionally, the platform's compatibility across different devices and browsers was enhanced to ensure accessibility for all users. Future improvements may include integrating additional payment gateways, expanding service coverage to larger events, and

adding advanced analytics for providers to monitor their business growth. These modifications and planned improvements aim to provide a seamless, secure, and effective platform for managing small-scale events.

5.5 Test Cases

Test cases play a crucial role in validating the functionality, performance, and security of QuickEvent Hub. They help ensure that the application meets all requirements and works as expected under various conditions. Test cases were designed for different modules, including user authentication, event management, service provider registration, and database operations.

Test Case Design:

- **Input Validation**: Testing the accuracy of user input fields, such as username, password, event details, and service selections. Invalid inputs like empty fields or incorrect formats were tested to confirm appropriate error messages.
- **Functional Testing**: Ensuring all functional aspects of the platform, such as user registration, login, event booking, and service provider management, work as intended.
- **Usability Testing**: Evaluating the user interface for ease of navigation, readability, and accessibility across devices.
- **Compatibility Testing**: Assessing the application's performance across different browsers (Chrome, Firefox, Edge) and devices (desktop, tablet, mobile).
- **Security Testing**: Verifying the security of authentication mechanisms and data protection. Ensuring prevention against SQL injection and unauthorized access.

User Features:

- Test Case 1: Verify user registration with valid and invalid inputs.
- Test Case 2: Validate user login authentication.
- Test Case 3: Ensure event creation and modification by users.

Service Provider Features:

- Test Case 4: Confirm service provider registration and profile update.
- Test Case 5: Validate service listing and availability update.

Admin Features:

- Test Case 6: Test admin ability to approve/reject service providers.
- Test Case 7: Ensure proper issue resolution and conflict management.

Payment Handling:

- Test Case 8: Verify cash-on-delivery option during booking.

Security and Authentication:

- Test Case 9: Test against SQL injection and XSS attacks.

Performance and Compatibility:

- Test Case 10: Verify application load handling with multiple users.
- Test Case 11: Ensure compatibility across browsers (Chrome, Firefox, Safari).

CHAPTER 6:
Results and Discussion

sThis chapter presents the outcomes of the QuickEvent Hub project, emphasizing the effectiveness, accuracy, and performance of the developed application. A detailed analysis of the test results is provided to evaluate whether the project objectives have been met successfully.

6.1 Test Reports

The test reports were generated based on the various testing methods employed during the development phase, including Unit Testing, Integration Testing, and Beta Testing. The primary goal was to ensure that the application operates efficiently, fulfils functional requirements, and meets user expectations.

Test Execution Summary:

- **Total Test Cases:** 25
- **Test Cases Passed:** 23
- **Test Cases Failed:** 2
- **Test Coverage:** 92%

Testing Tools and Environment:

- The testing process was executed using manual testing and automated scripts in environments simulating end-user scenarios.
- The application was tested on various platforms, including Windows, macOS, and Android, to ensure compatibility.
- Browser compatibility testing was performed on Chrome, Firefox, and Safari.

6.2 User Documentation

The user documentation for QuickEvent Hub serves as a comprehensive guide to assist users, service providers, and administrators in navigating and utilizing the platform effectively. It includes step-by-step instructions, explanations of features, and troubleshooting information to ensure a seamless experience.

Target Audience:

1. End Users: Individuals planning to organize small-scale events and seeking rental items or catering services.
2. Service Providers: Caterers and rental suppliers offering their services through the platform.
3. Administrators: Responsible for overseeing the platform's operations, managing users, and resolving disputes.

Platform Access:

- The application can be accessed via desktop and mobile browsers, ensuring a responsive design for user convenience.
- Compatible with popular browsers like Google Chrome, Mozilla Firefox, and Microsoft Edge.

User Registration:

- New users can register by providing basic information like name, contact details, and email address.
- Service providers must include additional business details, service types, and availability.

Navigating the Platform:

- Home Page: Overview of the platform, featured services, and quick access to registration.
- User Dashboard: After login, users can view available rental items, catering services, and past bookings.
- Service Provider Dashboard: Service providers can view booking requests, manage availability, and update their profiles.
- Admin Panel: Accessible only to administrators for monitoring user activities, resolving issues, and maintaining data integrity.

Booking Process:

1. Users can select desired rental items or catering services from a categorized list.
2. Add selected items to the cart, specify the event date and address, and place the order.
3. Payment options include cash-on-delivery, with future plans for online payment integration.

Managing Bookings:

- Users can view, modify, or cancel their bookings from the dashboard.
- Service providers can accept or reject booking requests based on their availability.

Troubleshooting and Support:

- Users can refer to the FAQ section for common issues and quick solutions.
- For additional support, a contact form is available for reporting issues or seeking assistance.

Feedback Mechanism:

- Users and service providers can leave feedback to help improve the platform.
- Ratings and reviews contribute to maintaining service quality and trustworthiness.

Future Enhancements:

- Integration of online payment gateways.
- Expanding service categories to include additional event-related services.
- Enhanced personalization through user preferences and event history.

The user documentation ensures that all platform users can navigate and utilize the application effectively, fostering a user-friendly and productive experience.

CHAPTER 7: Conclusions

7.1 Conclusions

The QuickEvent Hub has proven to be a practical solution for individuals and service providers involved in organizing small-scale, home-based events. The platform successfully bridges the gap between event organizers and service providers, creating a marketplace that simplifies event planning. Through effective implementation of front-end and back-end technologies, a user-friendly interface, and secure data management, the system meets the intended objectives outlined during the project's initial stages.

7.1.1 Significance of the System

The significance of QuickEvent Hub lies in its ability to cater to a market segment often overlooked by large-scale event management platforms. The platform facilitates seamless communication between users and service providers, enhancing convenience, cost-effectiveness, and efficiency. Key aspects of its significance include:

- Empowering Users: QuickEvent Hub empowers users by providing them with a variety of options to organize events according to their preferences and budget.
- Business Growth for Service Providers: Local catering businesses and rental providers benefit from increased visibility and access to a broader customer base, enabling business growth.
- Simplified Event Management: By consolidating multiple services under one platform, the system simplifies event planning, reducing the time and effort required.
- Community Impact: The platform supports local businesses, fostering community engagement and collaboration.

7.2 Limitations of the System

Despite its success, the QuickEvent Hub has certain limitations that require further attention for future development:

- Limited Payment Options: Currently, the system only supports cash-on-delivery payments. Integrating secure online payment gateways like UPI, credit cards, and net banking could enhance user convenience.
- Scalability Issues: The platform is primarily focused on small-scale events. Adapting the system to support larger events may require significant modifications in terms of infrastructure and database management.
- Geographical Constraints: The system's reach is currently limited to specific regions. Expanding its services to other locations could increase its user base.
- Limited Service Categories: Currently, the platform focuses primarily on catering and rental services. Including additional services like event photography, decoration experts, and entertainers could enhance its versatility.
- Mobile Application Absence: The absence of a dedicated mobile application limits accessibility, as many users prefer mobile-based interaction for convenience.

7.3 Future Scope of the Project

The future scope of the QuickEvent Hub includes potential improvements and expansion plans to enhance its functionality and user experience:

- Online Payment Integration: Incorporating secure and diverse payment methods, including digital wallets and payment gateways, to provide more flexible transaction options.
- Mobile Application Development: Developing a dedicated mobile application for Android and iOS platforms to cater to a broader audience and increase user engagement.
- Expansion to Larger Events: Adapting the platform to accommodate mid-sized and large events, expanding its market reach.
- Personalization Features: Implementing personalized recommendations based on user preferences and past event history to enhance user satisfaction.
- Real-Time Notifications: Integrating real-time notifications for booking updates, payment confirmations, and service provider responses.
- Advanced Analytics: Utilizing data analytics to provide insights to service providers about customer preferences, helping them optimize their offerings.
- Enhanced Marketing and Advertising: Allowing service providers to promote their services through featured listings or premium accounts.

In conclusion, the QuickEvent Hub has demonstrated its potential as an efficient and user-friendly platform for small-scale event planning. By addressing its limitations and exploring future enhancements, the system can continue to grow and serve a wider audience effectively.

References

1. **Books and Academic Resources:**
   - Deitel, H.M., Deitel, P.J. - *Java How to Program*, Pearson Education.
   - Schildt, H. - *Java: The Complete Reference*, McGraw-Hill Education.
2. **Web Resources:**
   - Oracle Documentation for Java SE: https://docs.oracle.com/javase/
   - W3Schools for JSP, CSS, and JavaScript resources: https://www.w3schools.com/
   - MySQL Official Documentation: https://dev.mysql.com/doc/
3. **Research Papers and Journals:**
   - Journals on event management and technology integration retrieved from IEEE Xplore.
   - Articles on real-time data handling and authentication systems from ACM Digital Library.
4. **Technological Resources:**
   - Apache Tomcat Server Documentation: https://tomcat.apache.org/
   - Bootstrap Documentation for Front-End Design: https://getbootstrap.com/
5. **Tools and Software:**
   - NetBeans IDE and Eclipse for coding and testing.
   - XAMPP for MySQL and Apache Server testing environment.
6. **Other Online Resources:**
   - TutorialsPoint for understanding core Java and MySQL connectivity.
   - Stack Overflow for troubleshooting and community-driven solutions.