**EVENT DISCOVERY MANAGEMENT**

**INTRODUCTION**

Planning an event involves coordinating multiple service providers, such as **event halls, beauty parlors, and catering services**, which can often be time-consuming and challenging. Searching for reliable service providers, comparing prices, and checking availability manually can result in inefficiencies and delays. This project aims to develop a **Local Event Finder Platform** that enables users to **search, compare, and book** essential event services seamlessly in one place.

The platform offers a **centralized solution** where users can browse various service providers based on key factors such as **capacity, location, pricing, and customer reviews**. By providing **real-time availability, transparent pricing, and user-generated feedback**, the system helps customers make informed decisions quickly and efficiently.

Users can **book marriage halls**, find **beauty parlors offering bridal makeup, grooming, and spa services**, and explore **catering options based on cuisine, budget, and customer ratings**. Service providers can list their offerings, manage bookings, update availability, and connect directly with customers. The system ensures a secure and efficient booking process, benefiting both users and vendors.

By eliminating intermediaries and offering a **direct booking mechanism**, the platform enhances **convenience, reliability, and accessibility** for event planning. Customers experience a streamlined booking process, while service providers gain greater visibility and a broader customer reach. The integration of **user ratings, reviews, and real-time updates** ensures a seamless and user-friendly experience.

This project is designed to be **completely free**, making it an ideal solution for individuals and businesses seeking a cost-effective way to plan and manage events. By leveraging a **user-friendly interface and efficient service management**, the **Local Event Finder Platform** simplifies event planning and improves the overall experience for both customers and service providers.

**2. LITERATURE CHRONICLE**

The evolution of event planning and service booking has undergone significant transformations over the years, shifting from traditional manual processes to advanced online platforms. In the early stages, event planning relied heavily on word-of-mouth recommendations, phone calls, and physical visits to service providers, making the process time-consuming and inefficient. With the advent of online business directories in the early 2000s, users gained access to basic information about event halls, catering services, and beauty parlors. However, these platforms lacked essential features such as real-time availability tracking and price comparisons. The growth of online booking platforms in the 2010s introduced user reviews, ratings, and digital reservations, making service selection more informed.

Despite these advancements, many platforms operated as intermediaries, increasing costs and limiting direct interaction between customers and service providers. Recent developments in AI-driven and real-time booking systems have revolutionized the industry by offering automated recommendations, live availability updates, and seamless direct bookings. Future trends indicate the integration of blockchain for secure transactions, voice-enabled assistants for easy bookings, and AR-based virtual previews of venues and services. The Local Event Finder Platform builds upon these advancements by offering a free, transparent, and efficient solution that connects users directly with service providers, ensuring a hassle-free and cost-effective event planning experience.

**PROBLEM DEFINITIONS**

Event planning requires the coordination of multiple service providers, such as event halls, beauty parlors, and catering services. However, the existing process of finding, comparing, and booking these services is highly inefficient and time-consuming. Customers often rely on word-of-mouth recommendations, scattered online directories, or in-person visits, making it difficult to obtain accurate information about pricing, availability, and quality. Many existing platforms act as intermediaries, leading to higher service costs and limited direct interaction between customers and vendors. Additionally, the absence of a centralized platform results in fragmented bookings, where users must visit multiple websites or contact individual service providers separately. The lack of real-time availability updates further complicates the process, often leading to booking conflicts or last-minute cancellations. Service providers also face challenges in reaching potential customers due to limited digital presence and inefficient booking management systems. To address these issues, a comprehensive Local Event Finder Platform is needed to streamline the process by offering a centralized system where users can search, compare, and book event-related services efficiently. This platform should ensure real-time availability tracking, transparent pricing, user reviews, and direct communication between customers and service providers, ultimately simplifying event planning and enhancing customer satisfaction.

**Existing System**

**Introduction**

The existing event discovery systems rely heavily on traditional methods such as social media, manual searches, and fragmented platforms. Users struggle to find relevant events efficiently due to the lack of a centralized platform, leading to difficulties in event planning and participation.

**Challenges in the Existing System**

1. **Manual Event Search**: Users have to search across multiple websites or rely on word-of-mouth to find events.
2. **Lack of Centralized Database**: Event details are scattered, making it difficult to get complete and accurate information.
3. **Limited Personalization**: Most systems do not offer intelligent recommendations based on user preferences.
4. **Static Event Listings**: Many existing platforms do not update real-time event statuses, causing confusion among users.
5. **No Automated Notifications**: Users must manually check for updates rather than receiving real-time alerts.
6. **Ineffective User Engagement**: Many systems do not allow event organizers to interact effectively with potential attendees.
7. **Lack of Integration**: There is little to no integration with maps, ticketing services, or social media for a seamless experience.

**Conclusion**

The current event discovery systems are inefficient and require manual intervention. A new approach is needed to provide real-time event updates, personalization, and better engagement for both users and event organizers.

**Proposed System**

**Introduction**

The proposed **Event Discovery Management System** aims to provide a centralized platform for users to search, discover, and manage events efficiently. Using Python and MySQL, this system enhances event discovery by integrating AI-based recommendations, real-time updates, and automation.

**Key Features of the Proposed System**

1. **Centralized Event Management**: A single database stores all event-related information, ensuring accuracy and completeness.
2. **Advanced Search & Filters**: Users can filter events based on location, date, category, and interests.
3. **AI-Based Personalized Recommendations**: The system suggests events based on users’ previous interactions and preferences.
4. **Real-Time Updates**: Organizers can modify event details dynamically, keeping the data up to date.
5. **Automated Notifications & Alerts**: Users receive emails or in-app notifications about upcoming events, ticket availability, or last-minute changes.
6. **User & Organizer Dashboard**:
   * **Users**: Can track favorite events, manage bookings, and receive recommendations.
   * **Organizers**: Can create, modify, and promote events with analytics insights.
7. **Integration with Maps & Ticketing Services**: Provides location-based event discovery and enables direct ticket purchases.
8. **Secure Registration & Booking**: Implements secure authentication and online transactions for a smooth experience.
9. **Community Engagement Features**: Allows users to rate, review, and interact with event organizers.

**Advantages Over the Existing System**

* **Efficient Event Discovery**: Users can find events faster with intelligent search and recommendation features.
* **Enhanced User Experience**: The platform provides a seamless and interactive interface.
* **Better Engagement for Organizers**: Enables direct communication with attendees, increasing participation.
* **Improved Data Security**: Ensures safe transactions and user data protection.
* **Real-Time and Automated Functionalities**: Reduces manual intervention and enhances efficiency.

**Conclusion**

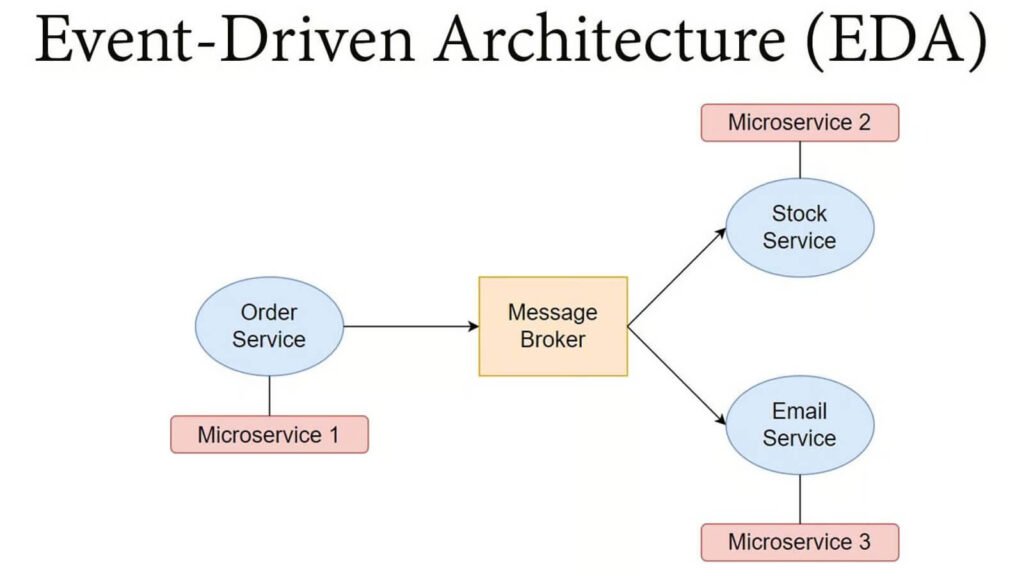
The proposed system provides a robust and scalable solution for event discovery, enhancing user experience and engagement. By leveraging Python and MySQL, this system will streamline event management, making it more accessible, interactive, and data-driven.

**TOOLS AND TECHNIQUES**

**METHODOLOGY**

The project involves developing a comprehensive platform with distinct user roles, Mysql database design, backend, Django frontend, user registration and login using JWT, event management, notifications, feedback and polls, security and access control, testing and quality assurance, deployment and scaling, and documentation and training. This multifaceted approach ensures a userfriendly interface, efficient data management, secure authentication, and scalable deployment, catering to both students and admins with role-specific functionalities.

**4.1 Technical Overview Fig -1: Technical Architecture**



The web browser sends an HTTPS request to the server. Express.js listens for incoming requests and directs them to the appropriate API endpoint. Middleware functions may intercept the request to perform tasks like authentication or data validation. If the request involves interacting with the database, Mongoose helps define the data structure and performs operations on the Mysql database using the Mysql driver. The Mysql driver communicates with the Mysql database to retrieve or manipulate data as needed. Finally, the server processesthe request, generates a response (In our case in JSON format), and sends it back to the web browser.

The below DFD provides a high-level overview of the data flow within the college event announcement platform. In this diagram, the main components are:

a. User Interface: This represents the interface through which users interact with the college event announcement platform. It can be a web-based interface, mobile application, or any other means of user interaction.

b. User Accounts Database: This database stores user account information, such as usernames, passwords, and personal details.

c. Admin Panel: This component provides administrative functionality to manage the event announcements. It allows authorized administrators to create, update, and delete events.

d. Event Management System: This system handles the management of events. It interacts with the user accounts database to authenticate administrators and with the event database to perform operations on events.

e. Event Database: This database stores event information, including event details, dates, locations, and associated metadata.

f. Email Service: This component is responsible for sending email notifications to users regarding event announcements. It may interact with the event management system to retrieve relevant event information.

The data flow in this example follows a typical pattern:

1. Users interact with the user interface to browse, search, and view event announcements.

2. User account-related data flows between the user interface and the user accounts database for user authentication and personalized experiences.

3. Authorized administrators access the admin panel to manage events, creating new event announcements or modifying existing ones.

4. The event management system interacts with the event database to store and retrieve event-related information.

5. The event management system may trigger notifications to users by utilizing the email service, which retrieves relevant event information from the event database and sends email announcements.

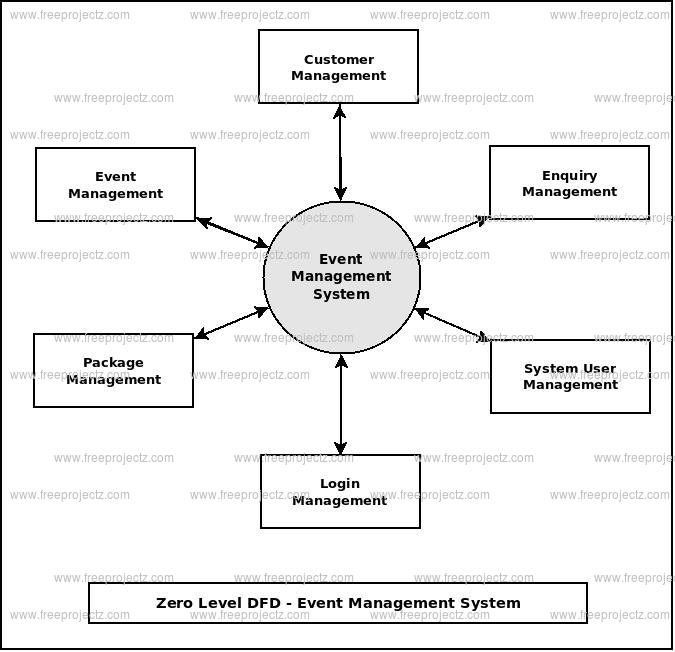
**Design and Implementation Define User Roles**:

Identify and define two user roles: students and admins. Determine their specific permissions and access levels within the platform. Database Design (Mysql): Create a Mysql database to store user data, event details, registration information, feedback, and other relevant data. Design schemas for users, events, notifications, feedback, and polls Implement user authentication and authorization mechanisms for both students and admins. Create API endpoints to manage user profiles, event data, registration forms, notifications, feedback, and polls. Implement email and WhatsApp notification services for users. Set up data retrieval and storage for past event details. Frontend (Django): Build anDjango frontend for a user-friendly and interactive interface. Create separate views and components for students and admins. Develop pages for viewing event information, registration forms, past event details, notifications, feedback submission, and poll participation. Implement user authentication and route protection. Design user dashboards for both students and admins with role specific features.

**User Registration and Login:** Create registration and login forms for both students and admins. Implement user authentication using for secure access.

**Event Management:** Develop admin-specific interfaces to create, edit, and manage event details. Enable event posting, including descriptions, dates, times, locations, andregistration forms. Store and retrieve event data from the database.

**Data Flow Diagram**



**Notifications:** Integrateemail for sending notifications to students. Set up automated notifications for event updates, registration confirmations, and other important information.

**Feedback and Polls:** Create forms for students to submit feedback on events and platform usability. Implement poll questions and capture user responses. Store feedback and poll data in the database for analysis.

**Security and Access Control:** Implement role-based access control to restrict actions and data access based on user roles. Ensure data security and privacy through proper authentication and authorization mechanisms.

**Testing and Quality Assurance:** Perform thorough testing to validate the functionality and user experience. Test for security vulnerabilities and data integrity. Conduct user acceptance testing to gather feedback for improvements.

**Deployment and Scaling:** Deploy the web application on a production server. Configure server infrastructure, hosting, and security measures. Ensure scalability to accommodate growing user numbers.

**Documentation and Training:** Create documentation for the platform’s functionality, API endpoints, and deployment instructions. Provide training for administrators on managing events and user data.

**Project Plan**

**USE CASE DIAGRAM**

**ER Diagram**

User

Status Tracking

Book

Confirmed

Search

Login

Status\_id

Booking\_id

Status

Create\_at

Sategories

Booking\_id

Service\_id

User\_id

Data

time

Bookings

Category\_id name

Catezies

Services

Category\_id name

**Class Diagram**

**Service**

Service\_id(int)

Category\_id(int)

Name(String)

Description(String

Price(decimal)

**User**

User\_id(int)

Name(String)

Email(String)

Passoword(String)

Login()

Category\_Id(int)

Name(String)

**Category**

**Booking Status**

Status\_id(int)

Booking\_id(int)

Status(String created\_at(timestamp)

**Review**

Review\_id(int)

User\_id(booking\_id)

Rating(int)

Comment(String)

**User**

User\_id(int)

Name(String)

Email(String)

Passoword(String)

Login()

**User Flow Diagram**

User

Status Tracking

View Servi ces

Book

Search

Login

**Servicer Flow Diagram**

Login

Search

Booking Updation

View Servi ces

User

Add /Del /Update Services

**DataBase Schema**

# bookings

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field | Type | Null | Key | Default | Extra |
| booking\_id | int(11) | NO | PRI | NULL | auto\_increment |
| service\_id | int(11) | NO | MUL | NULL |  |
| servicer\_user\_id | int(11) | YES |  | NULL |  |
| user\_id | int(11) | NO | MUL | NULL |  |
| booker\_name | varchar(100) | NO |  | NULL |  |
| email | varchar(100) | NO |  | NULL |  |
| mobile | varchar(15) | NO |  | NULL |  |
| address | text | NO |  | NULL |  |
| category\_name | varchar(255) | NO |  | NULL |  |
| date | date | NO |  | NULL |  |
| time | time | NO |  | NULL |  |
| guests | int(11) | YES |  | NULL |  |
| dj\_name | varchar(100) | YES |  | NULL |  |
| theme | varchar(255) | YES |  | NULL |  |
| birthday\_age | int(11) | YES |  | NULL |  |

# booking\_status

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field | Type | Null | Key | Default | Extra |
| status\_id | int(11) | NO | PRI | NULL | auto\_increment |
| booking\_id | int(11) | NO | MUL | NULL |  |
| status | varchar(255) | YES |  | PENDING |  |
| created\_at | timestamp | NO |  | CURRENT\_TIMESTAMP |  |

# reviews

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field | Type | Null | Key | Default | Extra |
| id | int(11) | NO | PRI | NULL | auto\_increment |
| service\_id | int(11) | NO | MUL | NULL |  |
| user\_id | int(11) | NO | MUL | NULL |  |
| rating | int(11) | NO |  | NULL |  |
| review | text | NO |  | NULL |  |
| created\_at | timestamp | NO |  | CURRENT\_TIMESTAMP |  |

**Database Table Structures - Services Module**

**Service**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field | Type | Null | Key | Default | Extra |
| service\_id | int(11) | NO | PRI | NULL | auto\_increment |
| service\_name | varchar(255) | NO |  | NULL |  |
| service\_description | text | NO |  | NULL |  |
| service\_price | decimal(10,2) | NO |  | NULL |  |
| service\_image | text | NO |  | NULL |  |
| city\_pincode | varchar(10) | NO | MUL | NULL |  |
| service\_address | text | NO |  | NULL |  |
| category\_id | int(11) | YES |  | NULL |  |
| servicers\_user\_id | int(11) | NO | MUL | NULL |  |
| created\_at | timestamp | NO |  | CURRENT\_TIMESTAMP |  |

**Services Details**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field | Type | Null | Key | Default | Extra |
| servicers\_user\_id | int(11) | NO | PRI | NULL | auto\_increment |
| servicers\_name | varchar(255) | NO |  | NULL |  |
| servicers\_mobile | varchar(20) | NO | UNI | NULL |  |
| servicers\_email | varchar(255) | NO | UNI | NULL |  |
| servicers\_password | varchar(50) | YES |  | NULL |  |
| servicers\_address | varchar(255) | YES |  | NULL |  |
| created\_at | timestamp | NO |  | CURRENT\_TIMESTAMP |  |

**Services Category**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field | Type | Null | Key | Default | Extra |
| category\_id | int(11) | NO | PRI | NULL | auto\_increment |
| category\_name | varchar(255) | NO | UNI | NULL |  |
| created\_at | timestamp | NO |  | CURRENT\_TIMESTAMP |  |

**Service City**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field | Type | Null | Key | Default | Extra |
| service\_city\_id | int(11) | NO | PRI | NULL | auto\_increment |
| city\_pincode | varchar(10) | NO | UNI | NULL |  |
| city\_name | varchar(255) | NO |  | NULL |  |
| created\_at | timestamp | NO |  | CURRENT\_TIMESTAMP |  |

**Service Details**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field | Type | Null | Key | Default | Extra |
| servicers\_id | int(11) | NO | PRI | NULL | auto\_increment |
| servicers\_user\_id | int(11) | NO | MUL | NULL |  |
| category\_id | int(11) | NO | MUL | NULL |  |
| created\_at | timestamp | NO |  | CURRENT\_TIMESTAMP |  |

**Users**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field | Type | Null | Key | Default | Extra |
| user\_id | int(11) | NO | PRI | NULL | auto\_increment |
| user\_name | varchar(25) | NO |  | NULL |  |
| user\_email | varchar(25) | NO | UNI | NULL |  |
| user\_password | varchar(25) | NO |  | NULL |  |
| user\_mobile | varchar(50) | NO |  | NULL |  |
| reg\_date | timestamp | NO |  | CURRENT\_TIMESTAMP |  |

**Project Accomplishment**

**Module Description**

The Event Discovery Management System is divided into several key modules, each playing a crucial role in its overall functionality.

**User Management Module**

* Allows users to register, log in, and manage their profiles.
* Implements role-based access control for users and event organizers.
* Provides secure authentication using Django.

**Event Management Module**

* Enables event organizers to create, update, and delete events.
* Stores event details such as date, time, location, category, and description.
* Provides real-time updates and modifications.

**Search and Filtering Module**

* Allows users to search for events based on various filters (location, category, date, etc.).
* Implements AI-based personalized recommendations for better user experience.

**Notification and Alerts Module**

* Sends automated email and SMS notifications for event updates and reminders.
* Notifies users about ticket availability, cancellations, and rescheduled events.

**Booking and Ticketing Module**

* Allows users to register for events and book tickets securely.
* Integrates with payment gateways for ticket purchases.
* Sends booking confirmations and generates digital tickets.

**Review and Rating Module**

* Enables users to rate and review events.
* Helps improve event quality based on user feedback.

**Data Analytics and Reports Module**

* Provides insights on user engagement and event popularity.
* Generates reports for event organizers to analyze performance and attendance.

**Integration Module**

* Connects the system with third-party services like Google Maps and social media platforms.
* Enables seamless event promotion and location-based services.

**SYSTEM REQUIRMENTS**

**SOFTWARE REQUIREMENTS**

Operating System : Microsoft windows10

Front End : PYTHON (Django)

Back End : MySQL Server

Documentation : Microsoft Word

**HARDWARE REQUIREMENTS**

Processor : Intel i3 core

Speed : 3.10 GHz

RAM : 8 GB

Hard Disk : 500 GB

Monitor : 15 SVGA color

Mouse : Wired mouse

Keyboard : wired keyboard

Printer : Laser Printer

**Project Outcome and Analysis**

**a)Results**

**b)Analysis**

**1. System Design & Architecture**

* Modular Structure: The project follows a modular structure with clear separation between frontend templates, backend logic, and database handling. This makes it maintainable and scalable.
* MVT Pattern: The system loosely follows the MVT(Model-View-Template) design. The views handle user interfaces, the controller (views.py) processes business logic, and models are implicitly managed via MySQL.

#### **2. User Experience (UX)**

* **Responsive UI**: Designed using Bootstrap 5.3, the interface adapts seamlessly to desktops, tablets, and mobile screens.
* **Form Automation**: Based on the selected category, the booking form dynamically loads relevant fields using jQuery, which simplifies the booking process and reduces confusion.
* **Feedback Loop**: Users receive confirmation of bookings and can track booking statuses, creating a full-cycle service experience.

**3. Database Design & Optimization**

* **Normalization**: The database schema is normalized to eliminate redundancy. Lookup tables (e.g., categories) and relationship tables (e.g., bookings, booking\_status) ensure efficiency.
* **Referential Integrity**: Enforced via foreign key constraints, ensuring that deletions in parent tables (like services) cascade properly to dependent records.
* **Trigger Usage**: A SQL trigger ensures that every new booking automatically gets an initial status record without requiring extra application logic.

#### **4. Data Handling & Validation**

* **Client-side Validation**: HTML5 validation is used for fields like email, time, and required fields.
* **Server-side Validation**: On the backend, all input is validated, and errors are handled gracefully.
* **Safe Queries**: Use of parameterized queries prevents SQL injection and ensures safer interaction with the database.

**5. Admin & Control Panel Readiness**

* Although a complete admin module isn't implemented yet, the database and design are ready to support features like:
  + Approving or rejecting bookings
  + Adding/editing/deleting services
  + Monitoring service usage and user analytics

**6. Future Expandability**

* **Machine Learning Integration**: The structure supports the addition of ML-based recommendation engines to personalize services.
* **Payment Gateway**: Easy to integrate Stripe, Razorpay, or PayPal due to modular design.
* **Notification System**: Can be extended to send email/SMS notifications on status changes.

**7. Technology Stack Usage**

* **Frontend**: HTML, Bootstrap, jQuery for dynamic UI
* **Backend**: Python with Django-like pattern handling
* **Database**: MySQL with relational integrity, triggers, and indexing
* **Recommendation Logic**: Currently rule-based, with plans for AI-enhanced personalization

**8. Performance Monitoring**

* Page load time and form submissions are optimized for speed.
* Minimal redundant SQL queries; only necessary data is fetched.
* Use of LIMIT, indexes, and proper JOINs keeps the system responsive.

**Conclusion**

The development of a **Local Event Finder Platform** addresses the challenges associated with event planning by providing a **centralized, user-friendly, and efficient** solution for booking event-related services. Traditional methods of searching for event halls, beauty parlors, and catering services are time-consuming, fragmented, and often unreliable. Existing platforms either lack comprehensive features or act as intermediaries, increasing costs and limiting direct customer-vendor interaction. By integrating **real-time availability tracking, user reviews, price comparisons, and direct booking capabilities**, the proposed system ensures a **seamless and hassle-free** experience for both customers and service providers. This platform enhances accessibility, reduces manual effort, and improves decision-making through transparent and up-to-date service listings. As the demand for **digitized and automated event management solutions** continues to grow, this system represents a **modern and cost-effective** approach to streamlining event planning, ultimately benefiting both users and businesses by ensuring a **smooth, efficient, and well-coordinated** service booking process.

**Future Enhancement**

As the project currently handles core booking functionality effectively, several features and improvements can be incorporated to make the system more robust, user-friendly, and scalable:

**Payment Gateway Integration**

* Allow users to make online payments at the time of booking via:
  + Razorpay / Stripe / PayPal integration
  + Send payment confirmation receipts
  + Handle refunds and cancellations with transaction tracking

**Recommendation System**

* Use machine learning to suggest services based on:
  + User preferences
  + Past bookings
  + Popularity trends
  + Time of year or event types

**Mobile Application**

* **Develop a mobile app version for:**
  + **Easier user access on Android/iOS**
  + **Push notifications**
  + **Location-based services for nearby event providers**