Software Requirements Specification (SRS)
Party Booking System

Version: 1.0
Date: 8/10/2024

---

2. System Overview

2.1 System Description

The Party Booking System allows users to search, book, and manage party venues. Venue owners can list their spaces with availability, while customers can browse, book, and pay for services.

2.2 Major Components

- **Frontend**: User interface for customers and venue owners.
- **Backend**: API services that handle requests, manage business logic, and perform CRUD operations on data.
- **Database**: Stores user data, venue information, booking details, and transaction logs.

3. Functional Requirements

3.1 User Registration and Authentication

- Users can create an account using an email address or third-party login (e.g., Google, Facebook).
- Users can log in securely using JWT (JSON Web Tokens) for session management.
- Admins and venue owners will have access to a dashboard with additional controls.

3.2 Venue Listings

- Venue owners can list their venues with details such as name, location, pricing, photos, and available dates.
- Users can filter venues by location, date, price range, and capacity.
- The system will display real-time availability for venues.

3.3 Booking System

- Users can book available venues by selecting an event date, time, and optional services (e.g., catering, decoration).
- The system will confirm availability before finalizing a booking.

- Bookings will be stored in the database and marked as 'Pending', 'Confirmed', or 'Cancelled'.

3.4 Payment Gateway

- Users can make payments using integrated payment gateways (e.g., Stripe, PayPal).
- Payment status will be updated in the system upon successful transaction processing.
- The system will handle payment failures with appropriate error messaging.

3.5 Notifications and Alerts

- The system will send email or SMS notifications on booking confirmations and cancellations.
- Alerts will be generated for upcoming bookings 24 hours prior to the event.

3.6 Admin Panel

- Administrators can manage users, venues, and bookings.
- Admins can generate reports on booking statistics, revenue, and venue performance.

4. Non-Functional Requirements

4.1 Performance

- The system should handle up to 10,000 concurrent users.
- API response time should not exceed 500ms under normal load.

4.2 Security

- All sensitive data will be encrypted using AES-256.
- Role-based access control (RBAC) will be implemented for user, venue owner, and admin roles.
- The system will follow OWASP best practices to prevent vulnerabilities.

4.3 Scalability

- The system will be designed to scale horizontally.
- The database will be optimized to handle large volumes of transactions.

4.4 Availability

- The system should have an uptime of 99.9%.
- In case of failure, the system should automatically redirect to a fallback page.

5. System Architecture

5.1 Backend

- **Technology Stack**: Node.js, Express.js, MongoDB, MySQL
- **API endpoints** will follow RESTful principles.
- **Data Models**:
  - User (id, name, email, password, role)
  - Venue (id, name, location, description, price, availability)
  - Booking (id, userId, venueId, date, status, paymentInfo)

5.2 Database

- MongoDB will be used for storing venue data and bookings.
- MySQL will be used for transactional data.
- The database schema will be normalized for efficiency and performance.

6. Assumptions and Constraints

6.1 Assumptions

- Users will have internet access to use the system.
- Payment processing will rely on third-party services (Stripe, PayPal).

6.2 Constraints

- The system should comply with data privacy regulations such as GDPR.
- Availability of venues will depend on accurate data from venue owners.

7. Appendix

- User Interface Mockups (Refer to design document)
- API Documentation (Refer to API doc)
- Database Schema Diagram