Web App Requirements

1. Functional Requirements

1.1 User Registration and Authentication

- **User Registration**: Allow users to sign up via email, social media accounts (Google, Facebook, etc.), or third-party services.
- **User Login/Logout**: Provide authentication mechanisms for logging in, logging out, and password recovery.
- **Role-based Access**: Different roles (e.g., Admin, Event Organizer, and Customer) with different permissions.

1.2 Event Management

- **Event Creation**: Event organizers can create events, add details (title, description, date, time, venue, etc.), and upload images.
- **Ticketing Options**: Provide ticket types (e.g., general admission, VIP, early bird), quantity, price, and other custom ticket attributes.
- **Event Categories**: Organize events into categories (e.g., music, conferences, sports, etc.).
- **Event Editing & Management**: Event organizers should be able to edit, update, and delete events after creation.
- **Event Drafts**: Allow saving events as drafts before publishing.

1.3 Ticket Management

- **Ticket Sales**: Enable users to purchase tickets for events. Offer options like single ticket, group tickets, and promo codes.
- **Ticket Transfer**: Allow ticket holders to transfer tickets to others (e.g., via email or a unique link).
- **Ticket Confirmation/QR Code**: Provide electronic tickets with a unique QR code for entry.
- **Discounts and Promo Codes**: Support for discounts, early-bird pricing, and promotional codes.

1.4 Payment Processing

- **Payment Gateway Integration**: Integrate with payment processors like Stripe, PayPal, or others for secure payments.
- **Multiple Currencies**: Support for multiple currencies depending on the location of the event.
- **Refunds & Cancellations**: Offer an easy way to handle refunds for canceled or rescheduled events.
- **Transaction History**: Allow both users and event organizers to view transaction histories.

1.5 Event Discovery

- **Event Search**: Provide search functionality to find events based on categories, keywords, location, and date.
- **Filters**: Allow users to filter events by date, price range, location, etc.
- **Event Recommendations**: Personalized event recommendations based on the user's interests, previous events, or browsing history.
- **Interactive Maps**: Display venue locations on a map (Google Maps, for instance).

1.6 User Profile & Dashboard

- **User Dashboard**: A personal dashboard for users to manage their profile, see their ticket purchases, and event registrations.
- **Event Organizer Dashboard**: Event organizers should have a dedicated dashboard to track ticket sales, manage events, and view analytics.
- **Event Statistics**: Provide event analytics (e.g., number of tickets sold, revenue, demographics).

1.7 Social Sharing & Engagement

- **Social Media Integration**: Allow event organizers and users to share events on social media platforms like Facebook, Twitter, Instagram, etc.
- **Event Invitations**: Allow users to invite friends to events via email, SMS, or social networks.
- **Reviews & Ratings**: Let users rate and review events after attending.

1.8 Mobile Optimization

- **Responsive Design**: Ensure the web app is fully responsive and optimized for mobile devices (smartphones and tablets).
- **Mobile App (Optional)**: we may consider developing a mobile app for both iOS and Android in the future.

1.9 Notifications

- **Email/SMS Notifications**: Notify users about event confirmations, ticket purchases, updates, and event reminders.
- **Push Notifications**: Send event reminders and promotions to users if using a mobile app.
- **Admin Notifications**: Notify admins about significant system events (e.g., high traffic, suspicious activity).

---

2. Non-Functional Requirements

2.1 Usability

- **User-Friendly Interface**: Simple and intuitive user experience. Minimize the number of steps for event creation and ticket purchasing.
- **Accessibility**: Ensure the app is accessible (WCAG 2.1 compliance), including text-to-speech support, color contrast, and keyboard navigation.
- **Localization**: Support multiple languages and regional formats (e.g., date, time, currency).

2.2 Performance

- **Fast Load Times**: Optimize the website for fast loading times, particularly for event pages and ticketing flows.
- **Scalability**: Design the system to handle spikes in traffic, especially during popular events.
- **High Availability**: Ensure the app is reliable and always accessible, especially during peak times.

2.3 Security

- **Data Encryption**: Use SSL/TLS for secure communication.
- **Payment Security**: PCI DSS-compliant payment processing.
- **Role-based Access Control**: Prevent unauthorized access by restricting sensitive data to authorized roles.
- **Two-factor Authentication**: For users with sensitive accounts or for the admin panel.
- **Data Backup**: Regular backup of all user and event data to prevent data loss.

2.4 Reliability

- **Error Handling**: Provide clear error messages and a fallback system for user actions that fail (e.g., payment failures, event creation issues).
- **Uptime Monitoring**: Ensure the system is actively monitored for uptime and performance metrics.

2.5 Analytics & Reporting

- **User Analytics**: Track user behavior on the platform, such as ticket purchases, event views, and search trends.
- **Event Analytics**: Provide event organizers with real-time analytics on ticket sales, revenue, demographics, and audience engagement.
- **Admin Reporting**: Admins should have access to site-wide statistics, including financials, user activity, and event performance.

---

3. Technical Requirements

3.1 Tech Stack

- **Frontend**:
    - HTML, CSS, JavaScript (React.js, Vue.js, or Angular for dynamic content)
    - Responsive design (Bootstrap, Tailwind CSS, etc.)
    - Server-side rendering (optional) for SEO and performance
- **Backend**:
    - Node.js with Express.js, Python with Django/Flask, Ruby on Rails, or Laravel for backend APIs
    - GraphQL or RESTful APIs for communication between frontend and backend
    - WebSockets for real-time updates (e.g., ticket sales)
- **Database**:
    - Relational Database (PostgreSQL or MySQL) for structured data (events, users, transactions)
    - NoSQL (MongoDB) for unstructured data or caching
- **Payment Gateway**:
    - Stripe, PayPal, or other services for payment processing
- **Cloud Hosting & CDN**:
    - AWS (Amazon Web Services), Google Cloud, or Azure for hosting
    - Content Delivery Network (CDN) like Cloudflare for fast static asset delivery
- **Search Engine**:
    - Elasticsearch for event search functionality
- **Authentication**:
    - OAuth for social logins (Google, Facebook, etc.)
    - JWT (JSON Web Tokens) for API authentication

3.2 Deployment & DevOps

- **Version Control**: Git with GitHub, GitLab, or Bitbucket for source code management.
- **CI/CD Pipelines**: Automate deployment using tools like Jenkins, CircleCI, or GitHub Actions.
- **Containerization**: Docker for containerized application deployment.
- **Monitoring**: Implement logging and monitoring tools like New Relic, Sentry, or Prometheus.

---

4. Business Requirements

- **Pricing Structure**:
    - Consider a commission model (per ticket sold), subscription model (for organizers), or a hybrid.
- **Customer Support**:
    - Offer customer support via live chat, email, or a help center with FAQs.
- **Marketing Features**:
    - Integrate email marketing tools (e.g., Mailchimp) for event promotions.

---

5. Future Considerations

- **Mobile App Development**: iOS and Android apps for native ticket purchase and event management.
- **Integration with Event Partners**: Allow third-party event organizers or platforms to list events on Ticketjar.
- **Virtual Events**: Support for virtual events (e.g., webinars, live streams).

This roadmap of features and technical requirements will give us a comprehensive framework to begin the development of Ticketjar. As the app grows, we can refine and add more sophisticated functionalities based on user feedback.