Product Requirements Document (PRD)

Title: Ticketer - Ticket Booking System

Date: 29th July 2025

1. Overview

This ticket booking system is a lightweight, full-stack web application that allows users to register, browse events, and book tickets online. It aims to simulate the experience of a modern ticketing platform, offering realistic user journeys and backend flows.

Designed primarily for training environments, this application provides a practical, hands-on context for exploring web application architecture and behavioral patterns in software systems.

Target Audience:

- General users to book flight and train tickets

---

2. Objectives

Functional Goals:

- Provide a user-friendly interface to search, view, and book event tickets.

- Implement core features common to ticketing platforms including user registration, login, booking history, and payment handling.

- Allow inspection and interaction with backend components for debugging.

3. Functional Requirements

3.1 User Management

- Users can sign up with a basic set of credentials.

- Passwords are transformed using a common hashing technique before being stored.

- Authentication maintains session data to provide access to user-specific features.

- Session token is the same as the user id.

3.2 Event Listing and Booking

- The system allows users to browse available events and view event details.

- Bookings are initiated by referencing event identifiers passed through query parameters.

- Seat selection and confirmation are available during the booking flow.

- Booking id is same as user_id+1 for first booking done by the user.

3.3 Booking History

- Logged-in users can view their past bookings via booking identifiers embedded in URL paths.

- Booking data is retrieved from backend storage by interpreting these identifiers.

3.4 Payment Simulation

- Integrates with a mock payment gateway to simulate the end-to-end ticket purchase flow.

- Users are guided through a simple checkout process.

3.5 Administrative Features

- Administrative dashboards and reporting endpoints are exposed under specific routes.

- Session presence allows access to advanced pages without enforcing rigid distinctions between user roles.

4. System Behavior and Design Notes

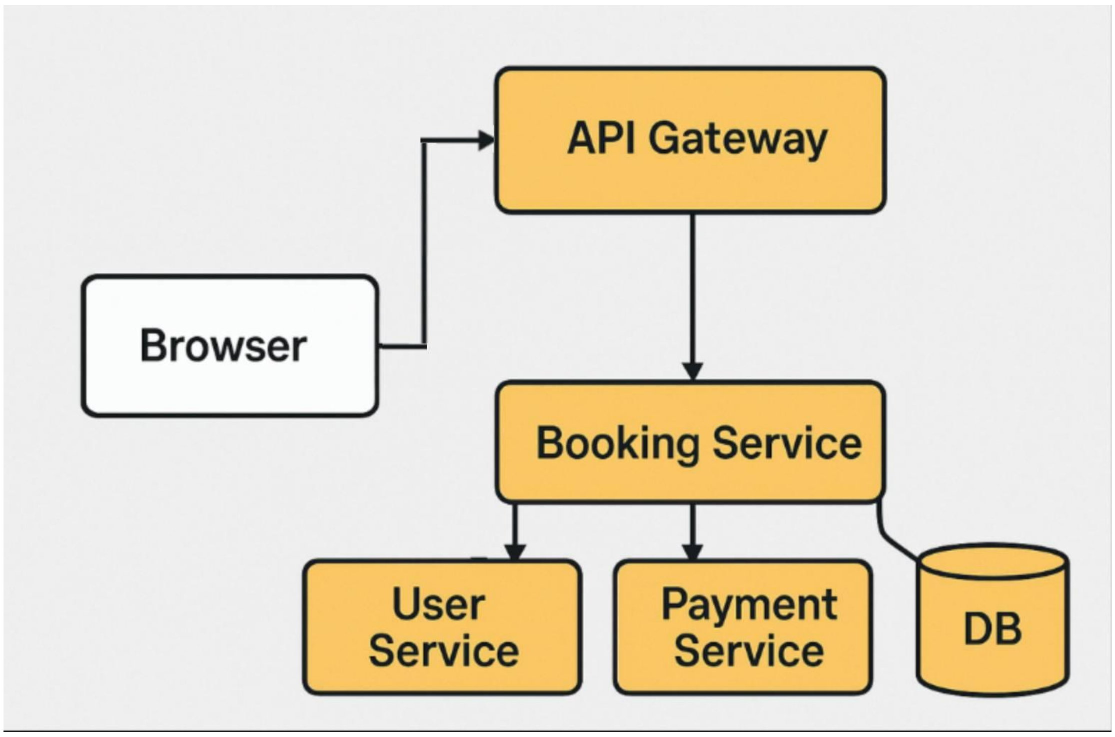
- Credential Handling: User credentials are protected using a basic transformation technique. This implementation is minimal by design, prioritizing accessibility and demonstration over cryptographic robustness such as SHA 1.

- Access Scope: Route-based access patterns are implemented uniformly across user and admin areas. This design encourages review of how access can be managed within web apps and allows comparison with more granular approaches.

- Data Access Patterns: The application uses simple integer-based identifiers for accessing resources such as events and bookings. These IDs are parsed directly from client inputs, enabling learners to assess the implications of such mechanisms.

- Client-Side Handling: The frontend includes transparent API usage and verbose console outputs to help learners observe interactions between the UI and backend.

5. Architecture Diagram



6. Non-Functional Requirements

Performance

- Optimized for demonstration environments with 30–50 concurrent users.

- Typical request response time under 2 seconds on local deployment.

Scalability

- Designed to be easily deployed on containers or virtual machines for labs.

- Backend components support easy instrumentation for inspection and logging.

7. Conclusion

This ticket booking system serves as a sandbox for exploring application design, development patterns, and backend workflows. While it offers a functional user journey, its implementation choices also make it a useful tool for evaluating how real-world decisions can impact maintainability, scalability, and control.