Final Report Smart Ticketing System Online

Md Abdur Rahim

Id : 221-15-5915

Sec : 61_V

Project Title: **Designing for a Smart Ticketing System Online Website**

Overview

The Smart Ticketing System Online for Diu Paribahan utilizes a microservices architecture to ensure robustness, scalability, and high availability. This design allows independent scaling and management of services, enhancing fault tolerance and ease of updates. By leveraging modern cloud services and efficient data management practices, the system is well-equipped to handle current demands and future growth, ensuring a seamless user experience even during peak traffic periods.

Company Background

Diu Paribahan, a family-owned transport company, has been specializing in passenger bus services since 1990. The company operates over 60 buses, including luxurious models like VOLVO and SCANIA, and employs over 200 trained staff, safely transporting over a million passengers annually across Bangladesh and beyond, including routes to Kolkata, India.

- ## Research on Scalable Web Application Architectures

Monolithic Architecture

Description: A single, unified codebase where all components are interlinked.

Pros:

- Simple development and deployment
- Easier testing

Cons:

- Hard to scale
- Difficult to maintain with growing complexity
- Single point of failure

Microservices Architecture

Description: Breaks down the application into smaller, independent services.

Pros:

- Independent scaling
- Fault isolation
- Easier to manage and update individual components

Cons:

- Complex development and deployment
- Requires robust inter-service communication mechanisms

Serverless Architecture

Description: Runs code in response to events and automatically manages server resources.

Pros:

- No server management
- Automatic scaling
- Cost-effective for variable workloads

Cons:

- Cold start latency
- Limited execution duration
- Dependency on third-party services

Other Architectures

- **Container-based Architecture**: Uses containers for consistent environments across development, testing, and production.
- **Event-Driven Architecture**: Focuses on producing and consuming events for decoupled components.

Scalability Requirements Analysis for Online Ticketing System

- **Expected User Base**: Large and growing due to the popularity of events.
- **Traffic Patterns**: High traffic during event announcements and booking periods, with peaks and troughs.

- **Data Volume**: Significant volume of transactional data, including user details, bookings, and payment information.
- **Future Growth Projections**: Anticipated increase in users and events, requiring the system to handle larger loads without performance degradation.

- ## Architecture Design

Choosing the Architectural Pattern

Microservices Architecture is chosen for the following reasons:

- **Scalability**: Individual services can be scaled independently based on load.
- **Fault Isolation**: Issues in one service do not affect the entire system.
- **Agility**: Easier to develop, test, and deploy smaller, independent services.

Defining Components/Modules and Their Responsibilities

1. **User Interface (UI)**
    - **Responsibility**: Provides a responsive and user-friendly interface for users.
    - **Interaction**: Communicates with backend services through RESTful APIs.
2. **Authentication Service**
    - **Responsibility**: Manages user authentication and authorization.
    - **Interaction**: Provides JWT tokens to the UI for secure communication with other services.
3. **Ticket Management Service**
    - **Responsibility**: Handles ticket creation, updating, and deletion.
    - **Interaction**: Interacts with the Database Service for storing ticket information and the Notification Service for booking confirmations.
4. **Payment Gateway**
    - **Responsibility**: Processes user payments securely.
    - **Interaction**: Integrates with third-party payment processors and updates the Ticket Management Service upon successful transactions.
5. **Notification Service**
    - **Responsibility**: Sends notifications to users, such as booking confirmations and reminders.
    - **Interaction**: Triggered by the Ticket Management Service and Payment Gateway.
6. **Analytics Service**
    - **Responsibility**: Analyzes usage patterns and generates reports.
    - **Interaction**: Collects data from all services for comprehensive analytics.
7. **Database Service**
    - **Responsibility**: Stores user profiles, ticket details, and transaction records.

    - **Interaction**: Central repository accessed by all services needing persistent data storage.

Load Balancing, Caching, and Data Storage

- **Load Balancing**: Use AWS Elastic Load Balancer to distribute incoming traffic across multiple instances of services, ensuring even load distribution and high availability.
- **Caching Mechanisms**: Implement Redis for caching frequently accessed data, such as event listings and seat availability, to reduce database load and improve response times.
- **Data Storage**:
    - **Relational Database**: Use PostgreSQL for structured data, such as user profiles and ticket transactions.
    - **NoSQL Database**: Use MongoDB for flexible and scalable storage of event-related information and logs.

Communication Between Components

- **Internal Communication**: Use RESTful APIs over HTTP for simplicity and compatibility. Services communicate using lightweight JSON messages.
- **External Communication**: Secure communication using HTTPS for all interactions between the UI and backend services to protect user data.

- # Initial Implementation: Technologies and Frameworks

8. **HTML**: Structure of the web pages.
9. **CSS**: Styling of the web pages.
10. **JavaScript**: Client-side scripting for interactivity.
11. **Tailwind CSS**: Utility-first CSS framework for rapid UI development.
12. **Daisy UI**: Component library for Tailwind CSS to build accessible and customizable UI components.

- ## Future Development

To further enhance the scalability and functionality of the system, the following technologies will be integrated in the next phases:

- **React.js**: For a more dynamic and responsive user interface.
- **Node.js and Express.js**: For building scalable server-side applications.
- **MongoDB**: For flexible, document-oriented data storage.

Conclusion

By adopting a microservices architecture and leveraging modern web technologies, the smart ticketing system for Diu Paribahan is designed to be scalable, resilient, and user-friendly. The system is well-prepared to handle current demands and future growth, ensuring a seamless experience for users during peak traffic periods.