Name: Hammad Shabbir, Iqrash Qureshi  
Roll no: i221140-i221174  
Section: CS-B  
 **Devop’s Assignment no 1**

**Event Booking System Documentation**

## Overview

The **Event Booking System** is a microservices-based platform designed to facilitate seamless event discovery, ticket booking, and management. This system is structured into multiple services, each handling a specific responsibility to ensure scalability, reliability, and maintainability. Users can browse various events, book tickets, receive notifications, and manage their bookings efficiently.

## System Architecture

The system is built using a microservices architecture, ensuring flexibility and easy scalability. Each microservice is responsible for a distinct function and communicates with others via REST APIs and asynchronous messaging using **RabbitMQ**. The system comprises the following microservices:

**Key Microservices**

**1. User Service**

* Manages user authentication and profiles.
* Ensures secure login and registration mechanisms.
* Uses **FastAPI** or **Express.js** for backend implementation.
* Stores user data in **PostgreSQL**.
* Communicates synchronously via **REST API**.
* Provides APIs for:
  + Fetching all users
  + Fetching a specific user by ID
  + Creating a new user
  + Handling user login

**2. Event Service**

* Handles event listings and their details.
* Ensures availability status of events is accessible.
* Uses **Spring Boot** or **Node.js** as the backend framework.
* Stores event-related data in **MongoDB**.
* Communicates synchronously via **REST API**.
* Provides APIs for:
  + Fetching all events
  + Fetching a specific event by ID
  + Checking the availability of an event

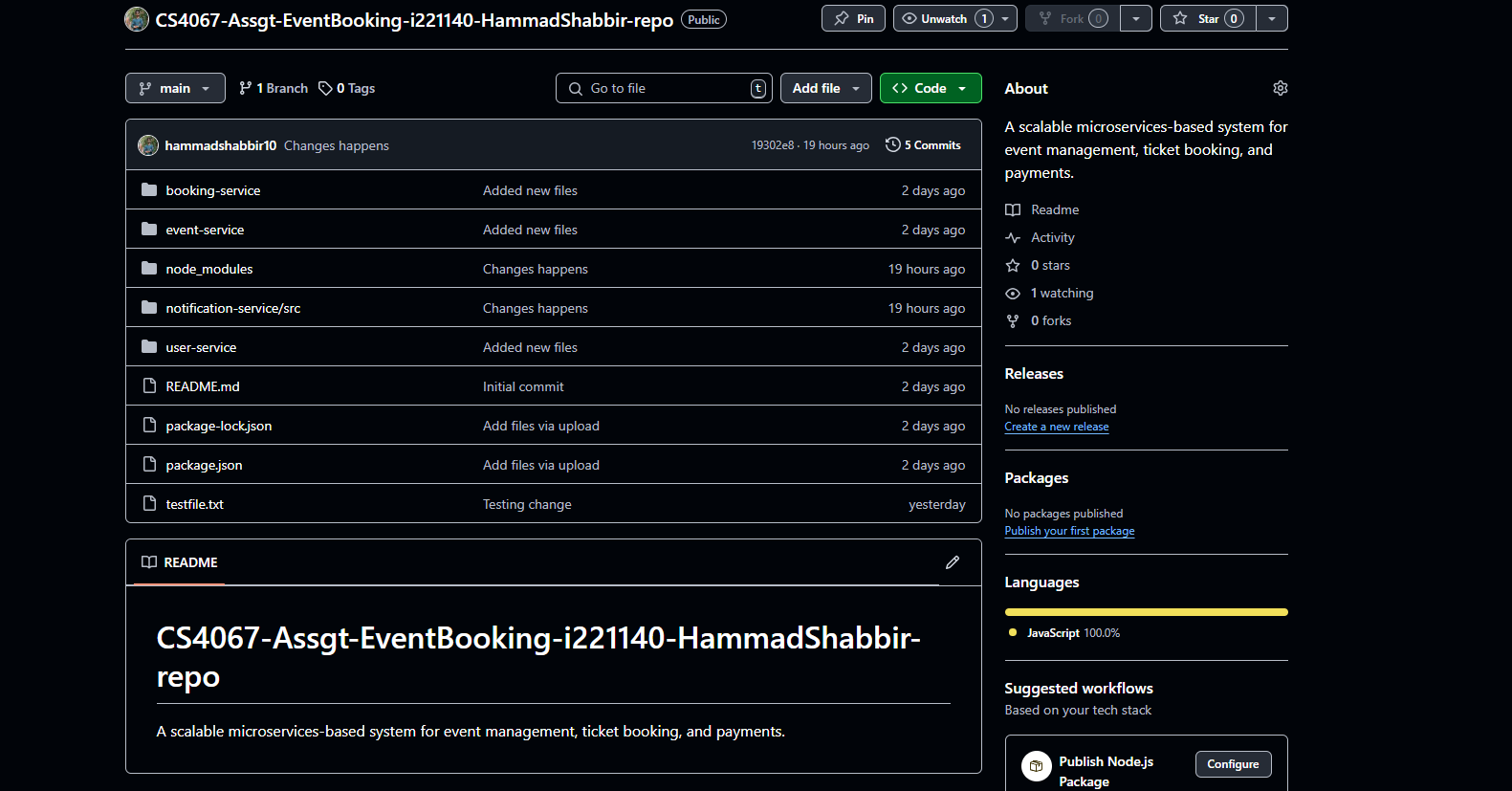
**3. Booking Service**

* Manages ticket bookings, payments, and status updates.
* Ensures smooth integration with payment gateways.
* Uses **Flask** or **Express.js** for backend development.
* Stores booking details in **PostgreSQL**.
* Communicates via **REST API** for synchronous operations and **RabbitMQ** for asynchronous messaging.
* Provides APIs for:
  + Creating new bookings
  + Fetching booking details
  + Processing payments

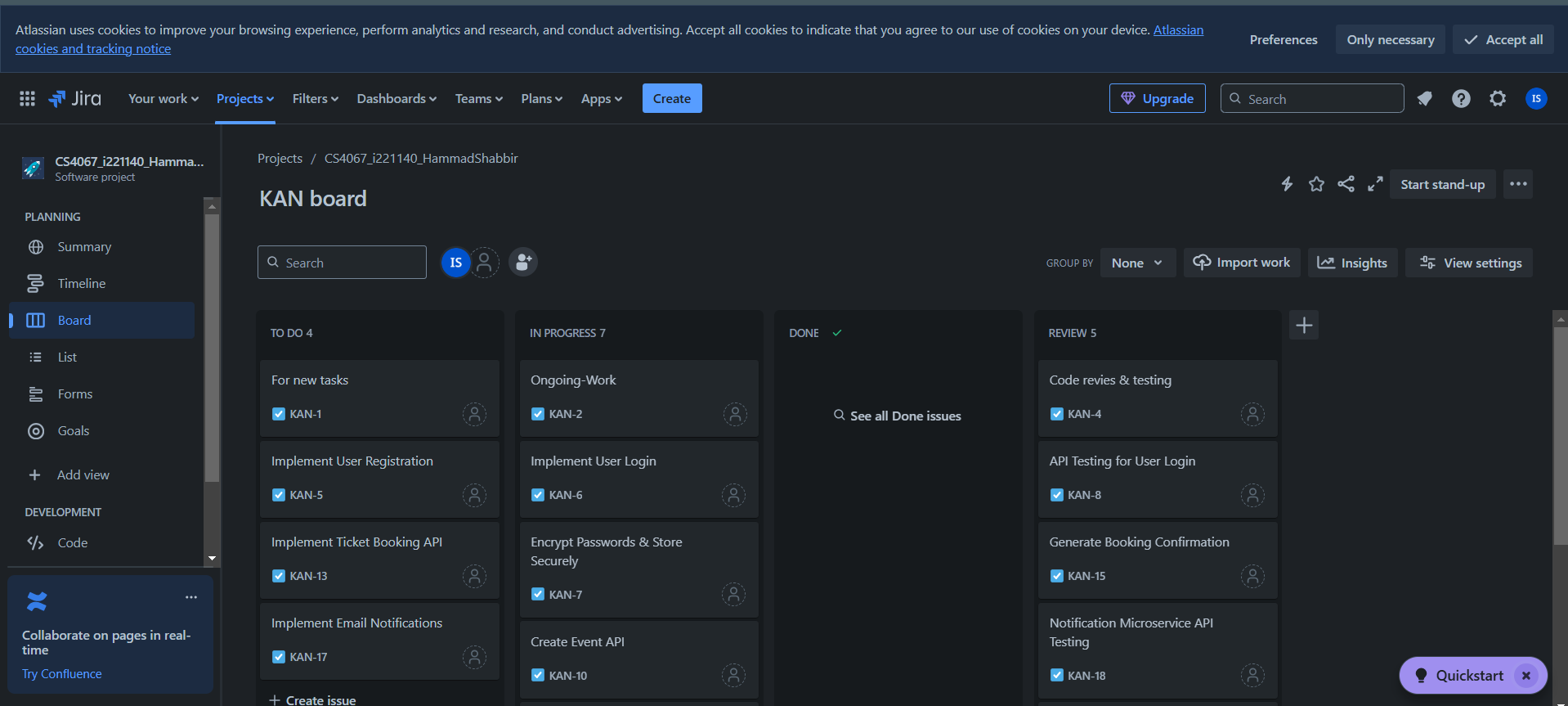
**4. Notification Service**

* Sends email notifications for booking confirmations and updates.
* Uses **Flask** or **Express.js** for backend processing.
* Stores logs and notifications in **MongoDB**.
* Communicates asynchronously via **RabbitMQ**.
* Triggers email notifications based on booking events.

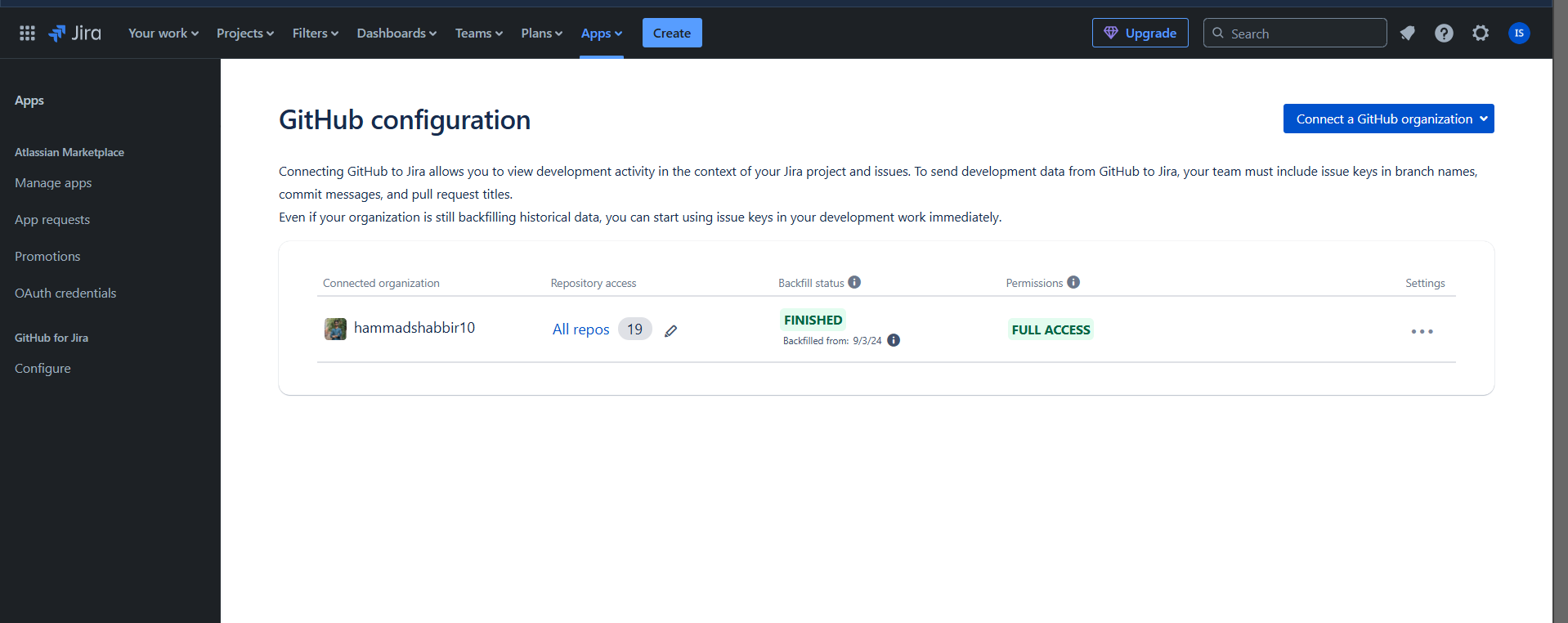
**Jira & Git-hub Integration**



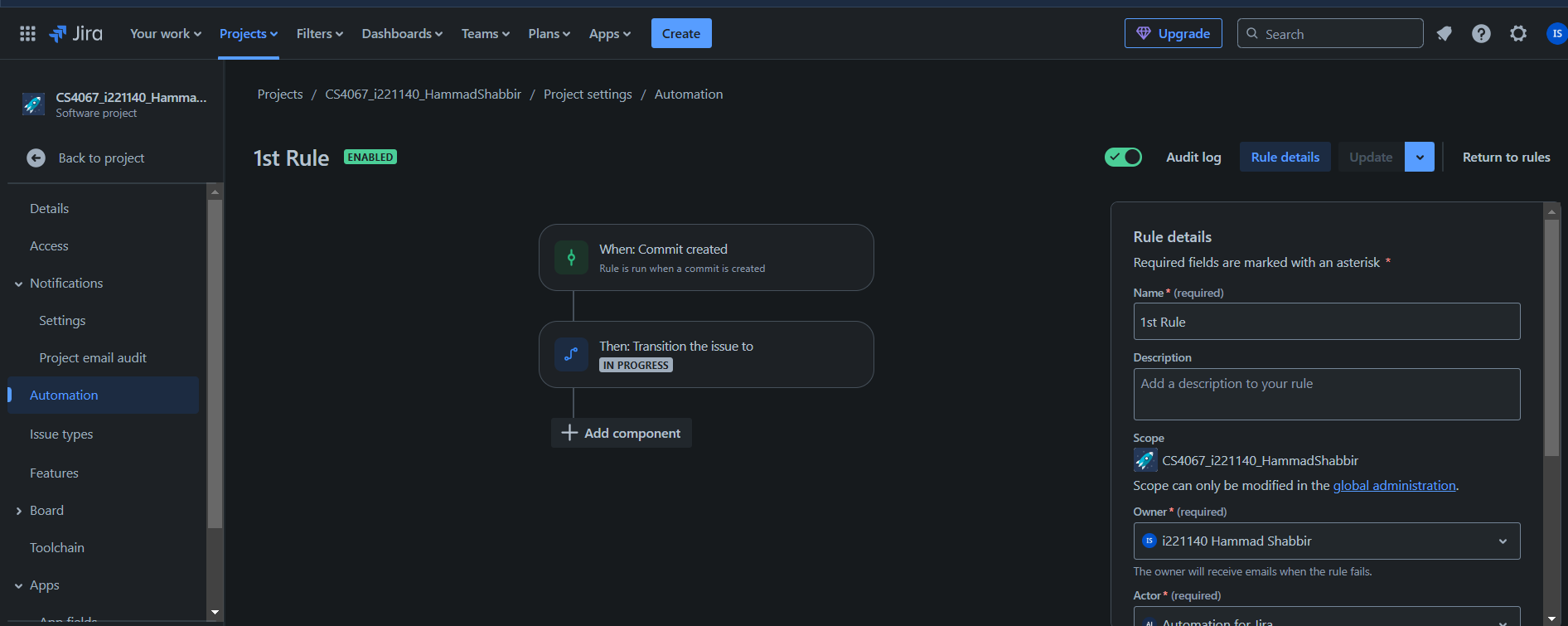
Kan-BAN setup



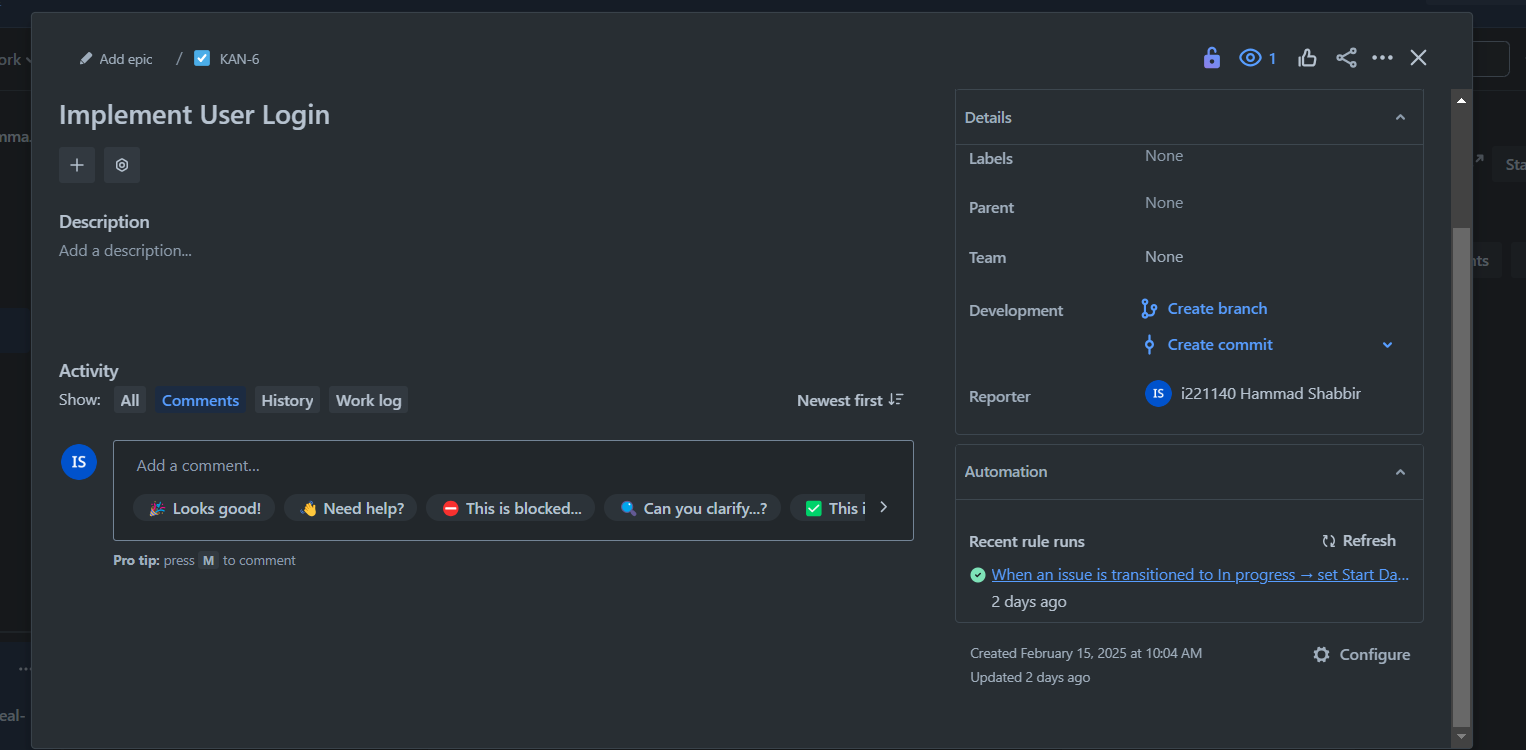
Integration Successful



My Rules has been designed



I have transfer TO-Do -> In Progress



**Frontend Implementation**

* Built with **HTML, CSS, and JavaScript**.
* Displays system details and microservice descriptions dynamically.
* Uses DOM manipulation to create service overview cards.

## Code Quality

Ensuring high-quality code and proper documentation is essential for the maintainability of the system. The project follows best coding practices, including:

* **Modular Code Structure:** Each microservice is isolated and self-contained.
* **Consistent API Design:** RESTful endpoints follow a structured naming convention.
* **Error Handling & Logging:** Proper logging mechanisms are in place for debugging.
* **Scalability Considerations:** Database choices and message queues enhance scalability.

## Conclusion

The Event Booking System is a well-structured microservices platform that enables smooth event ticketing experiences. By leveraging technologies like **FastAPI, Express.js, Spring Boot, MongoDB, PostgreSQL, and RabbitMQ**, the system ensures reliability, flexibility, and performance. Future enhancements could include additional features like payment gateway integration, real-time notifications, and AI-powered event recommendations.