Assignment 1

Bookings and events can be successfully created within the system, allowing users to reserve rooms, manage events, and facilitate approvals according to their roles. This functionality was achieved through carefully designed RESTful controllers for each microservice, handling operations such as room availability checks, user role verification, and event approvals. By structuring the controllers with a clear separation of responsibilities, each service could independently handle its own resources and provide endpoints for streamlined inter-service communication.

The implementation of this system required thoughtful integration of both relational and non-relational databases, with PostgreSQL supporting structured data in the RoomService and UserService, and MongoDB handling more dynamic data in the BookingService and EventService. This hybrid approach enabled efficient data management and optimized performance. Inter-service requests, such as the verification of a user's role or room availability checks, were managed using synchronous RESTful calls, ensuring each action, such as booking confirmation, was validated in real-time. Containerizing each service with Docker and orchestrating them through Docker Compose allowed for consistent, reliable deployment.

Testing played a pivotal role in ensuring system reliability and resilience. Integration tests, conducted using TestContainers, validated that the basic services operated as expected and communicated effectively with others. However, implementation tests were unable to be successfully run on the Booking, Event and Approval-services at this time. Overall, the project highlighted the importance of well-designed controllers, modular implementation, and rigorous testing in creating a scalable and reliable microservices-based booking and event management system.