# Enterprise Middleware Coursework Report

Samuel Pillai Sathiyamoorthy
240503828

## Introduction:

The coursework required the development of a RESTful API for managing hotel bookings as part of a larger travel agency service. The project emphasized modularity, scalability, and adherence to enterprise-grade practices. Using modern middleware technologies like Quarkus, JPA, and JTA, the application was designed to be deployed on OpenShift, providing a cloud-native experience. The final implementation integrates with APIs developed by classmates to create a unified TravelAgent service for managing multiple commodities, including flights, taxis, and hotels.

## Service Composition and Architecture:

The hotel application was structured with a clear separation of concerns across multiple layers, ensuring maintainability and scalability:

1. **Entities**:
    - Hotel: Captures essential hotel details like name and address. A unique constraint ensures no duplicate hotel names.
    - Booking: Represents the relationship between Customer and Hotel, encapsulating booking details such as the bookingDate.
2. **Repositories**:
    - The HotelRepository and BookingRepository handle database operations for their respective entities, ensuring efficient data access through JPA.
3. **Services**:
    - HotelService: Contains business logic for managing hotel data.
    - BookingService: Validates and processes hotel booking requests.
4. **REST Endpoints**:
    - HotelRestService: Exposes endpoints for CRUD operations on hotels.
    - BookingRestService: Manages booking operations, including creation and cancellation.
5. **Integration**:
    - The TravelAgent API integrates with the hotel service and the services of classmates for flights and taxis, enabling aggregate bookings.
6. **Deployment**:

- Deployed on OpenShift using Maven and GitHub, ensuring a robust and cloud-native environment.

## JPA Annotations:

JPA annotations were extensively utilized to define entity relationships, constraints, and database operations:

1. **Core Annotations**:
   - **@Entity**: Marks Hotel and Booking as persistent entities.
   - **@Table**: Defines table-specific constraints, such as unique hotel names.
   - **@Id, @GeneratedValue**: Configure primary key generation strategies.
2. **Relationships**:
   - **@ManyToOne**: Establishes the link between Booking and Hotel or Customer.
   - **@OneToMany**: Used in reverse for associating multiple bookings with a single hotel or customer.
3. **Cascading**:
   - **CascadeType.ALL** ensures that operations like delete or persist on a parent entity (e.g., Hotel) propagate to associated entities (e.g., Booking).
4. **Advanced Features**:
   - Cascading deletions and orphan removal mechanisms were implemented to maintain referential integrity and prevent database inconsistencies.

## Technologies:

The hotel application leveraged cutting-edge middleware technologies to deliver a scalable, robust, and cloud-ready solution:

1. **Quarkus**:
   - A Kubernetes-native Java framework optimized for cloud deployment.
2. **JPA**:
   - Simplifies database interactions by abstracting the persistence layer.
3. **JTA**:
   - Ensures transactional consistency, especially in the GuestBooking API.
4. **OpenShift**:
   - Platform-as-a-Service for deploying and managing cloud applications.
5. **Swagger/OpenAPI**:

- Provides comprehensive API documentation and endpoint testing.

6. **REST Assured**:
   - Used for automated testing of RESTful endpoints.

## Validation:

Validation was implemented to ensure data consistency and compliance with business rules:

1. **Bean Validation**:
   - **@NotNull** and **@Size**: Validate fields like name, email, and address.
   - **@Pattern**: Enforces correct formatting for attributes such as email addresses.
2. **Custom Validation**:
   - HotelValidator:
     - Ensures unique hotel names during creation.
   - BookingValidator:
     - Prevents overlapping bookings for the same customer and hotel on the same date.
3. **Error Handling**:
   - Custom exceptions like HotelNameAlreadyExistsException and InvalidBookingDateException provide user-friendly error messages.

## Personal Experience and Project Reflection

Developing this application provided hands-on experience in enterprise middleware development. Key takeaways include:

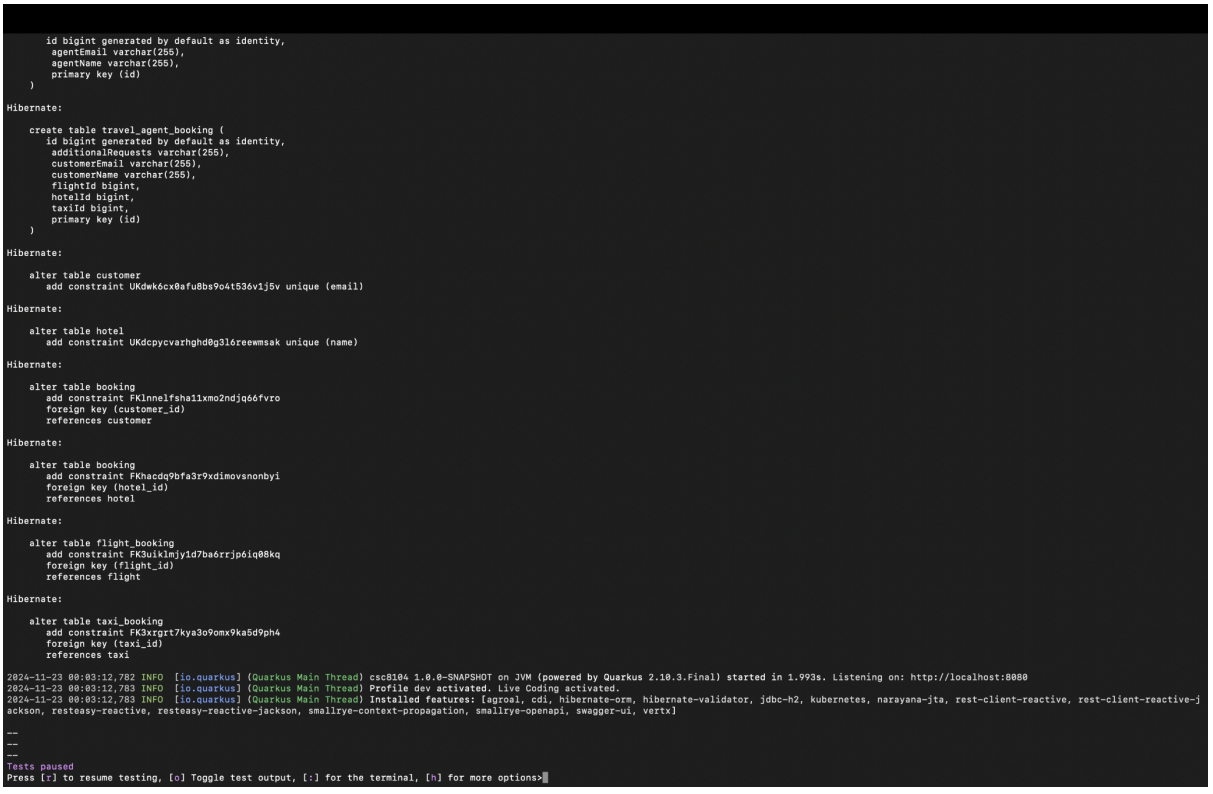
- **Service Integration**:
  - Integrating APIs for flights and taxis into the TravelAgent service highlighted the importance of standardized API contracts.
- **Transactional Consistency**:
  - Implementing JTA-based manual transaction management for GuestBooking was a challenging yet rewarding experience.
- **Cloud Deployment**:
  - Using OpenShift demonstrated the complexities of cloud deployment, including configuration management and testing for scalability.

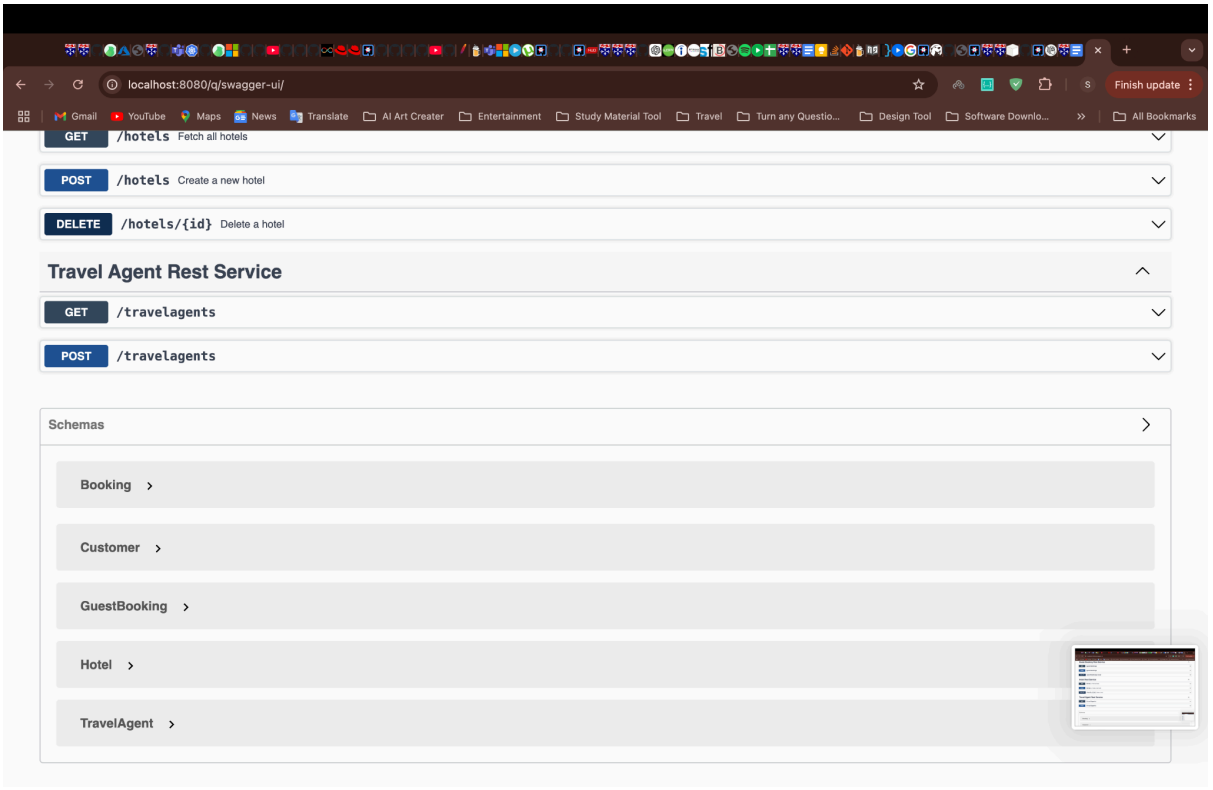
This project deepened my understanding of middleware technologies, RESTful principles, and cloud-native application development.

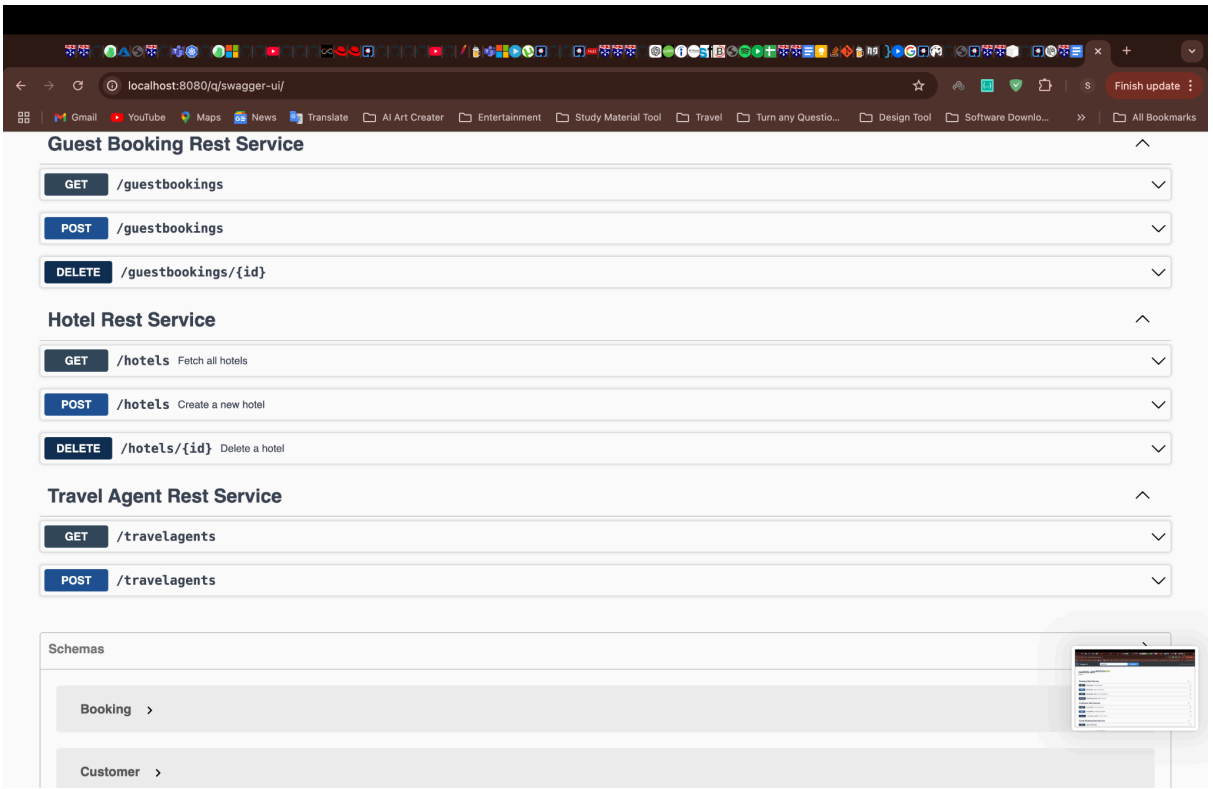
## References

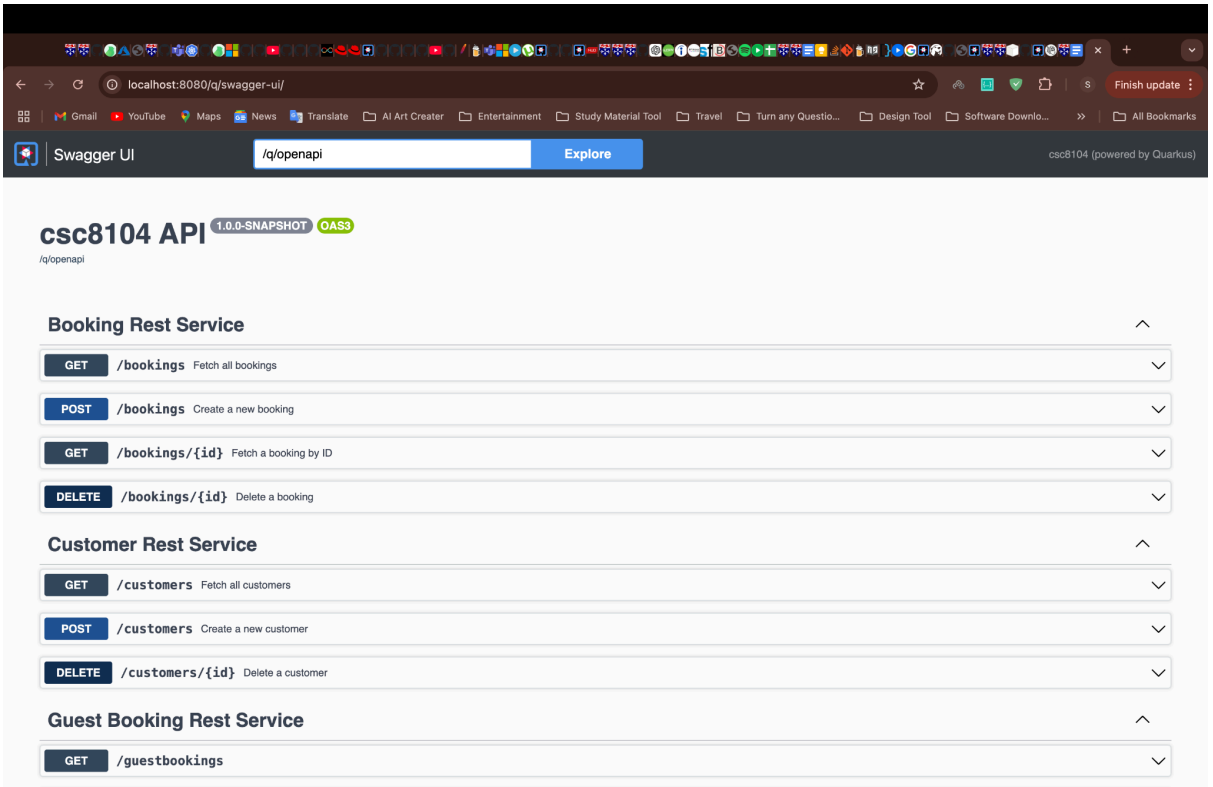

1. **Technologies**:
    - Quarkus, JPA, JTA, Maven, OpenShift, Swagger, REST Assured.
2. **Documentation**:
    - Official documentation for Quarkus, JPA, and OpenShift.
3. **Course Material**:
    - Coursework tutorials and lectures provided by Newcastle University.
4. **Additional Readings**:
    - RESTful Web APIs and Enterprise Integration Patterns for design insights.

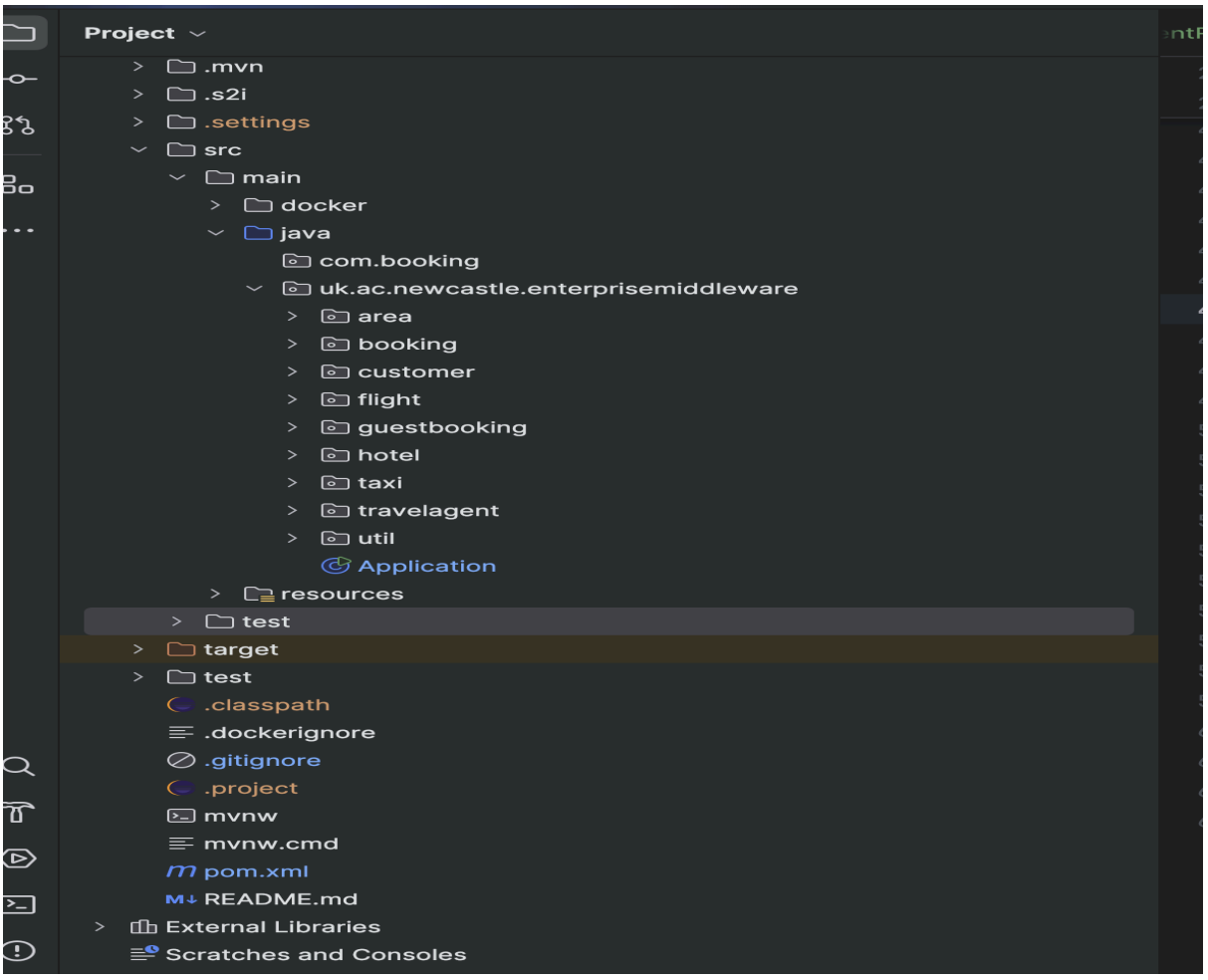

## Screenshots:




localhost:8080/q/swagger-ui/
Finish update
Gmail
YouTube
Maps
News
Translate
AI Art Creater
Entertainment
Study Material Tool
Travel
Turn any Questio...
Design Tool
Software Downlo...
All Bookmarks
GET /hotels Fetch all hotels
POST /hotels Create a new hotel
DELETE /hotels/{id} Delete a hotel
Travel Agent Rest Service
GET /travelagents
POST /travelagents
Schemas
Booking
Customer
GuestBooking
Hotel
TravelAgent


localhost:8080/q/swagger-ui/
Finish update
Gmail
YouTube
Maps
News
Translate
AI Art Creater
Entertainment
Study Material Tool
Travel
Turn any Questio...
Design Tool
Software Downlo...
All Bookmarks
Guest Booking Rest Service
GET /guestbookings
POST /guestbookings
DELETE /guestbookings/{id}
Hotel Rest Service
GET /hotels Fetch all hotels
POST /hotels Create a new hotel
DELETE /hotels/{id} Delete a hotel
Travel Agent Rest Service
GET /travelagents
POST /travelagents
Schemas
Booking
Customer


localhost:8080/q/swagger-ui/
Finish update
Gmail
YouTube
Maps
News
Translate
AI Art Creater
Entertainment
Study Material Tool
Travel
Turn any Questio...
Design Tool
Software Downlo...
All Bookmarks
Swagger UI
/q/openapi
Explore
csc8104 (powered by Quarkus)
csc8104 API
1.0.0-SNAPSHOT
OAS3
/q/openapi
Booking Rest Service
GET /bookings Fetch all bookings
POST /bookings Create a new booking
GET /bookings/{id} Fetch a booking by ID
DELETE /bookings/{id} Delete a booking
Customer Rest Service
GET /customers Fetch all customers
POST /customers Create a new customer
DELETE /customers/{id} Delete a customer
Guest Booking Rest Service
GET /guestbookings


Project
.mvn
.s2i
.settings
src
main
docker
java
com.booking
uk.ac.newcastle.enterprisemiddleware
area
booking
customer
flight
guestbooking
hotel
taxi
travelagent
util
Application
resources
test
target
test
.classpath
.dockerignore
.gitignore
.project
mvnw
mvnw.cmd
pom.xml
README.md
External Libraries
Scratches and Consoles