Assignment Wk07 Web Services

Problem statement:-

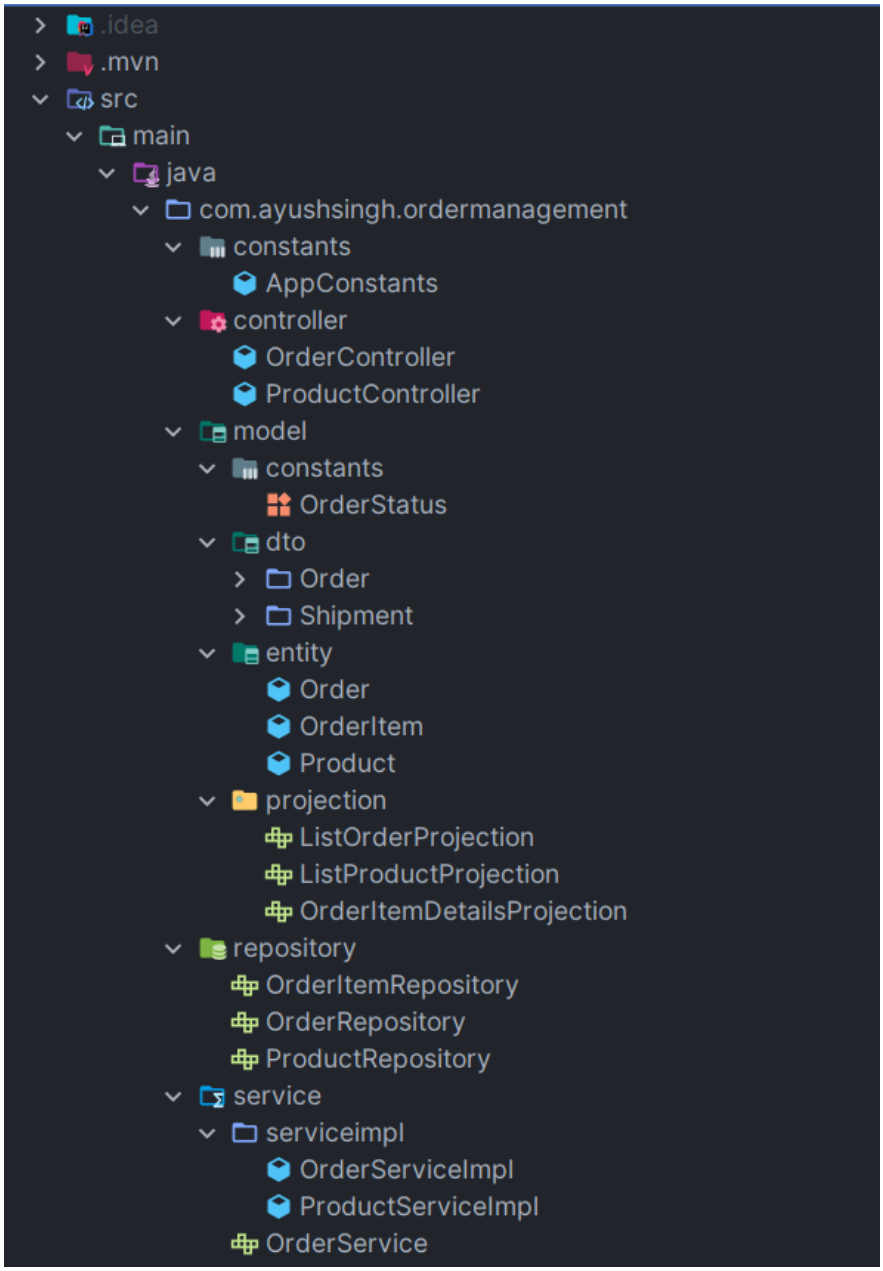
Use micro-service architecture to write rest services for Order management and Shipment.

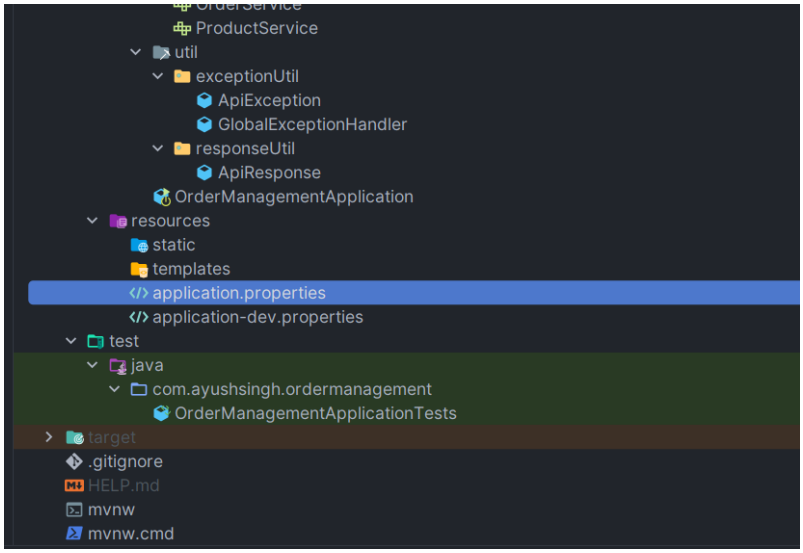
Requirement:

1) Write CRUD operations for Order Management
2) Write an end point in Order management service to place the order to shipment service.
3) Write rest assured test cases for Order Management service

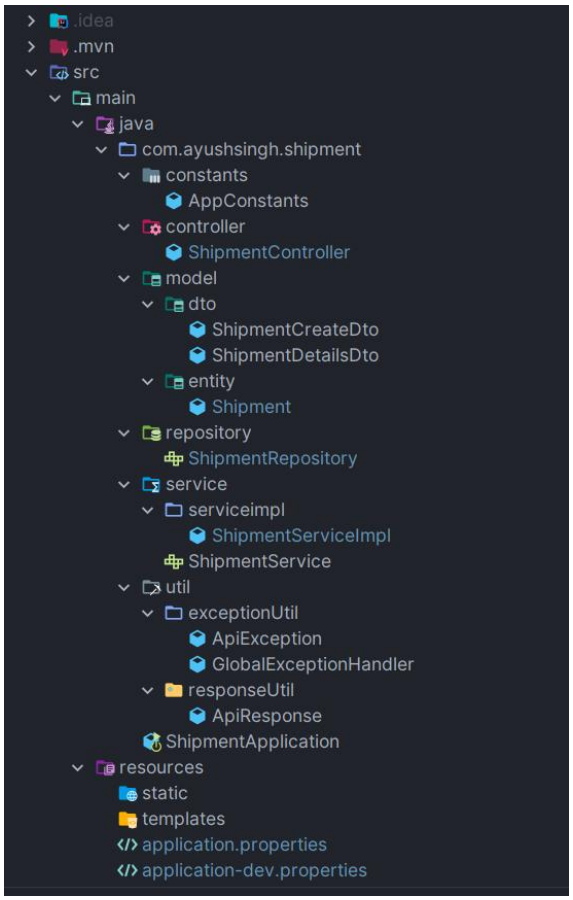
Project Structure

Order Management





Shipment



Order Management Project Details

Entity classes

```
package com.ayushsingh.ordermanagement.model.entity;

//---imports---

@Getter
@Setter
@NoArgsConstructor
@AllArgsConstructor
@Table(name = "ecom_product")
@Entity
public class Product {

    @Id
```

```
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long productId;

    @Column(name = "product_token", nullable = false, unique = true)
    private String productToken;

    @Column(name = "product_name", nullable = false, unique = true)
    private String productName;

    @Column(name = "stock_quantity", nullable = false)
    private Long stockQuantity;

    @OneToMany(mappedBy = "product", cascade =
{CascadeType.MERGE,CascadeType.PERSIST,CascadeType.DETACH,CascadeType.REFRESH},
orphanRemoval = true)
    private Set<OrderItem> orderItems = new HashSet<>();



    @PrePersist
    public void prePersist() {
        productToken = UUID.randomUUID().toString();
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        Product product = (Product) o;
        return Objects.equals(productToken, product.productToken);
    }

    @Override
    public int hashCode() {
        return Objects.hash(productToken);
    }
}
```

```
package com.ayushsingh.ordermanagement.model.entity;

//-imports

@Getter
@Setter
@NoArgsConstructor
@AllArgsConstructor
@Entity
@Table(name="ecom_order")
public class Order {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long orderId;

    @Column(name = "order_token", unique = true, nullable = false)
    private String orderToken;

    @Column(name="order_status",nullable = false)
    @Enumerated(value = EnumType.STRING)
    private OrderStatus orderStatus;

    @Column(name="address",nullable = false)
    private String address;

    @Column(name="customer_name",nullable = false)
    private String customerName;

    @Column(name="shipment_code",unique = true)
    private String shipmentCode;
```

```
    @OneToMany(mappedBy = "order", cascade =
{CascadeType.MERGE,CascadeType.PERSIST,CascadeType.DETACH,CascadeType.REFRESH},
orphanRemoval = true)
    private Set<OrderItem> orderItems = new HashSet<>();

    @PreRemove
    private void preRemove() {
        for (OrderItem orderItem : orderItems) {
            orderItem.setOrder(null);
        }
        orderItems.clear();
    }

    @CreatedDate
    @CreationTimestamp
    @Column(name = "created_at", nullable = false, updatable = false)
    private Date createdAt;

    @LastModifiedDate
    @UpdateTimestamp
    @Column(name = "updated_at")
    private Date updatedAt;



    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        Order order = (Order) o;
        return Objects.equals(orderToken, order.orderToken);
    }

    @Override
    public int hashCode() {
        return Objects.hash(orderToken);
    }
}
```

```
package com.ayushsingh.ordermanagement.model.entity;


@Getter
@Setter
@NoArgsConstructor
@AllArgsConstructor
@Entity
@Table(name="ecom_order_items")
public class OrderItem {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long orderItemId;

    @ManyToOne(fetch = FetchType.LAZY,cascade =
{CascadeType.MERGE,CascadeType.PERSIST,CascadeType.REFRESH})
    @JoinColumn(name = "order_id")
    private Order order;

    @ManyToOne(fetch = FetchType.LAZY,cascade =
{CascadeType.MERGE,CascadeType.PERSIST,CascadeType.REFRESH})
    @JoinColumn(name = "product_id")
    private Product product;

    @Column(name = "quantity", nullable = false)
    private Long quantity;
```

```
    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        OrderItem orderItem = (OrderItem) o;
        return Objects.equals(orderItemId, orderItem.orderItemId) && Objects.equals(order,
orderItem.order) && Objects.equals(product, orderItem.product);
    }

    @Override
    public int hashCode() {
        return Objects.hash(orderItemId, order, product);
    }

}
```

Repositories

```
public interface OrderItemRepository extends JpaRepository<OrderItem, Long> {


    @Query("""
            select
            oi.product.productToken as productToken,
            oi.product.productName as productName,
            oi.quantity as quantity
            from OrderItem oi
            where oi.order.orderToken = :orderToken
            """)
    List<OrderItemDetailsProjection> findOrderItemDetailsByOrderToken(String orderToken);

    @Modifying
    @Query("""
            delete from OrderItem oi where oi.order.orderToken =:orderToken
            """)
    void deleteByOrderToken(String orderToken);
}
```

```
public interface OrderRepository extends JpaRepository<Order, Long> {

    @Modifying
    @Query("delete from Order o where o.orderToken = ?1")
    void deleteByOrderToken(String orderToken);


    @Query("select o from Order o where o.orderToken = ?1")
    Optional<Order> findByOrderToken(String orderToken);

    @Query("""
            select
            o.address as address,
            o.customerName as customerName,
            o.orderToken as orderToken,
            o.orderStatus as orderStatus,
            o.createdAt as orderDate
            from Order o order by o.createdAt desc
            """)
    List<ListOrderProjection> findAllOrders();
}
```

```
public interface ProductRepository extends JpaRepository<Product, Long> {


    @Query("SELECT p FROM Product p WHERE p.productToken = ?1")
    Optional<Product> findByProductToken(String productToken);

    @Query("""
            select
            p.productToken as productToken,
            p.productName as productName
```

```
            from Product p
            """)
    List<ListProductProjection> getAllProducts();
}
```

Projections

```
public interface ListOrderProjection {

    String getAddress();

    String getCustomerName();

    OrderStatus getOrderStatus();

    Date getOrderDate();

    String getOrderToken();
}
```

```
public interface ListProductProjection {

    String getProductName();
    String getProductToken();
}
```

```
public interface OrderItemDetailsProjection {

    String getProductName();
    String getProductToken();
    String getQuantity();
}
```

Service

```
public interface ProductService {

    List<ListProductProjection> getAllProducts();
}
```

```
public interface OrderService {

    String placeNewOrder(OrderCreateDto orderCreateDto);

    OrderDetailsDto orderDetails(String orderToken);

    String cancelOrder(String orderToken);

    String updateOrder(String orderToken, OrderUpdateDto orderUpdateDto);

    List<ListOrderProjection> getAllOrders();
}
```

Implementations

```
@Service
@Slf4j
public class OrderServiceImpl implements OrderService {

    private final OrderRepository orderRepository;
    private final OrderItemRepository orderItemRepository;
    private final ProductRepository productRepository;
    private final RestClient restClient;

    public OrderServiceImpl(OrderRepository orderRepository, OrderItemRepository
orderItemRepository, ProductRepository productRepository) {
```

```
        this.orderRepository = orderRepository;
        this.orderItemRepository = orderItemRepository;
        this.productRepository = productRepository;
        this.restClient = RestClient.builder().baseUrl("http://localhost:8086").build();
    }

    @Override
    public String placeNewOrder(OrderCreateDto orderCreateDto) {

        Order order = new Order();
        order.setOrderStatus(OrderStatus.PLACED);
        order.setAddress(orderCreateDto.getAddress());
        order.setCustomerName(orderCreateDto.getCustomerName());
        order.setOrderToken(UUID.randomUUID().toString());
        Set<OrderItemDto> itemList = orderCreateDto.getProducts();
        Set<OrderItem> orderItems = new HashSet<>();
        if (itemList.isEmpty()) {
            throw new ApiException("Order cannot be empty!");
        }
        for (OrderItemDto orderItemDto : itemList) {
            OrderItem orderItem = new OrderItem();
            orderItem.setQuantity(orderItemDto.getQuantity());
            orderItem.setOrder(order);
            Product product =
productRepository.findByProductToken(orderItemDto.getProductToken()).orElseThrow(() -> new
ApiException("Product with id: " + orderItemDto.getProductToken() + " not found!"));
            if (product.getStockQuantity() < orderItemDto.getQuantity()) {
                throw new ApiException("Insufficient stock for product: " +
product.getProductName());
            }
            orderItem.setProduct(product);
            orderItems.add(orderItem);
        }
        order.setOrderItems(orderItems);
        ShipmentCreateDto shipmentCreateDto = new ShipmentCreateDto();
        shipmentCreateDto.setOrderToken(order.getOrderToken());
        Map<String, Object> response =
restClient.post().uri("/api/v1/shipment/create").contentType(MediaType.APPLICATION_JSON).b
ody(shipmentCreateDto).retrieve().body(Map.class);
        log.debug("Response: " + response);
        Integer code = (Integer) response.get("code");
        if (code != 2000) {
            throw new ApiException("Shipment could not be created!");
        }
        Map<String, Object> responseData = (Map<String, Object>) response.get("data");
        order.setShipmentCode((String) responseData.get("shipmentCode"));
        orderRepository.save(order);

        return "Order with id: " + order.getOrderToken() + " created successfully!";
    }

    @Override
    public OrderDetailsDto orderDetails(String orderToken) {
        Optional<Order> orderOptional = orderRepository.findByOrderToken(orderToken);
        if (orderOptional.isEmpty()) {
            throw new ApiException("Order with id: " + orderToken + " not found!");
        }
        Order order = orderOptional.get();
        OrderDetailsDto orderDetailsDto = new OrderDetailsDto();
        orderDetailsDto.setOrderToken(order.getOrderToken());
        orderDetailsDto.setCustomerName(order.getCustomerName());
        orderDetailsDto.setAddress(order.getAddress());
        orderDetailsDto.setOrderStatus(order.getOrderStatus());

orderDetailsDto.setOrderItems(orderItemRepository.findOrderItemDetailsByOrderToken(orderTo
ken));
        return orderDetailsDto;
    }

    @Transactional
    @Override
```

```
    public String cancelOrder(String orderToken) {
        orderItemRepository.deleteByOrderToken(orderToken);
        orderRepository.deleteByOrderToken(orderToken);
        Map<String, Object> response =
restClient.delete().uri("/api/v1/shipment/cancel/{orderToken}",
orderToken).retrieve().body(Map.class);
        Integer responseCode = (Integer) response.get("code");
        if (responseCode != 2000) {
            throw new ApiException("Shipment could not be deleted!");
        }
        return "Order with id: " + orderToken + " deleted successfully!";
    }

    @Override
    public String updateOrder(String orderToken, OrderUpdateDto orderUpdateDto) {
        Optional<Order> orderOptional = orderRepository.findByOrderToken(orderToken);

        if (orderOptional.isEmpty()) {
            throw new ApiException("Order with id: " + orderToken + " not found!");
        }
        Order order = orderOptional.get();
        order.setAddress(orderUpdateDto.getAddress());
        order.setCustomerName(orderUpdateDto.getCustomerName());
        orderRepository.save(order);
        return orderToken;
    }

    @Override
    public List<ListOrderProjection> getAllOrders() {
        return orderRepository.findAllOrders();
    }


}
```

```
@Service
@RequiredArgsConstructor
public class ProductServiceImpl implements ProductService {

    private final ProductRepository productRepository;
    @Override
    public List<ListProductProjection> getAllProducts() {
        return productRepository.getAllProducts();
    }
}
```

Utility classes

```
@Getter
@Setter
public class ApiResponse<T> {

    private String message;
    private T data;
    private Integer code;

    public ApiResponse(T data) {
        this.data = data;
        this.message= AppConstants.SUCCESS_MESSAGE;
        this.code=AppConstants.SUCCESS_CODE;
    }

    public ApiResponse(String message, T data, Integer code) {
        this.message = message;
        this.data = data;
        this.code = code;
    }
}
```

```java
public class ApiException extends RuntimeException {

    public ApiException(String message) {
        super(message);
    }
}
```

```java
@RestControllerAdvice
public class GlobalExceptionHandler {

    @ExceptionHandler(ApiException.class)
    public ResponseEntity<ApiResponse<String>> handleApiException(ApiException e) {
        e.printStackTrace();
        return new ResponseEntity<>(new ApiResponse<>(AppConstants.ERROR_RESPONSE,
e.getMessage(), AppConstants.ERROR_CODE), HttpStatus.OK);
    }
}
```

Controller

```java
@RestController
@RequiredArgsConstructor
@RequestMapping("/api/v1/product")
public class ProductController {

    private final ProductService productService;

    @RequestMapping("/all")
    public ResponseEntity<ApiResponse<List<ListProductProjection>>> getAllProducts() {
        return new ResponseEntity<>(new ApiResponse<>(productService.getAllProducts()),
HttpStatus.OK);
    }

}
```

```java
@RestController
@RequiredArgsConstructor
@RequestMapping("/api/v1/order")
public class OrderController {

    private final OrderService orderService;

    @PostMapping("/new")
    public ResponseEntity<ApiResponse<String>> createOrder(@RequestBody OrderCreateDto
orderCreateDto) {
        String token = orderService.placeNewOrder(orderCreateDto);
        return new ResponseEntity<>(new ApiResponse<>(token), HttpStatus.CREATED);
    }

    @GetMapping("/{orderToken}")
    public ResponseEntity<ApiResponse<OrderDetailsDto>> getOrderDetails(@PathVariable
String orderToken) {
        OrderDetailsDto orderDetailsDto = orderService.orderDetails(orderToken);
        return ResponseEntity.ok(new ApiResponse<>(orderDetailsDto));
    }

    @DeleteMapping("/{orderToken}")
    public ResponseEntity<ApiResponse<String>> cancelOrder(@PathVariable String
orderToken) {
        String message = orderService.cancelOrder(orderToken);
        return ResponseEntity.ok(new ApiResponse<>(message));
    }

    @PatchMapping("/{orderToken}")
    public ResponseEntity<ApiResponse<String>> updateOrder(@PathVariable String
orderToken, @RequestBody OrderUpdateDto orderUpdateDto) {
```

```
        String message = orderService.updateOrder(orderToken, orderUpdateDto);
        return ResponseEntity.ok(new ApiResponse<>(message));
    }

    @GetMapping("/all")
    public ResponseEntity<ApiResponse<List<ListOrderProjection>>> getAllOrders() {
        List<ListOrderProjection> orders = this.orderService.getAllOrders();
        return ResponseEntity.ok(new ApiResponse<>(orders));
    }
}
```

Shipment Project Details

Entity Class

```
@Getter
@Setter
@NoArgsConstructor
@AllArgsConstructor
@Entity
@Table(name = "ecom_shipment")
public class Shipment {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "shipment_id")
    private Long id;

    @Column(name = "shipment_token", unique = true, nullable = false)
    private String shipmentCode;

    @Column(name="current_shipment_location", nullable = false)
    private String currentShipmentLocation;

    @Column(name="order_token", nullable = false,unique = true)
    private String orderToken;


    @CreatedDate
    @CreationTimestamp
    @Column(name = "created_at", nullable = false, updatable = false)
    private Date createdAt;

    @LastModifiedDate
    @UpdateTimestamp
    @Column(name = "updated_at")
    private Date updatedAt;


    @PrePersist
    public void prePersist() {
        this.shipmentCode = UUID.randomUUID().toString();
    }


}
```

Repository

```
public interface ShipmentRepository extends JpaRepository<Shipment, Long> {



    @Modifying
    @Query("""
            delete from Shipment s where s.orderToken = :orderToken
            """)
    void deleteByOrderToken(@Param("orderToken") String orderToken);
}
```

Service

```
public interface ShipmentService {

    ShipmentDetailsDto createShipment(ShipmentCreateDto shipmentCreateDto);

    Boolean cancelShipment(String shipmentCode);
}
```

Implementation

```
@Service
@RequiredArgsConstructor
public class ShipmentServiceImpl implements ShipmentService {

    private final ShipmentRepository shipmentRepository;
    private final ModelMapper modelMapper;
    @Override
    public ShipmentDetailsDto createShipment(ShipmentCreateDto shipmentCreateDto) {
        Shipment shipment=new Shipment();
        shipment.setOrderToken(shipmentCreateDto.getOrderToken());
        shipment.setCurrentShipmentLocation("WAREHOUSE");
        shipment=shipmentRepository.save(shipment);
        return modelMapper.map(shipment,ShipmentDetailsDto.class);
    }

    @Transactional
    @Override
    public Boolean cancelShipment(String orderToken) {
       shipmentRepository.deleteByOrderToken(orderToken);
       return true;
    }
}
```

Controller

```
@RestController
@RequiredArgsConstructor
@RequestMapping("/api/v1/shipment")
public class ShipmentController {

    private final ShipmentService shipmentService;

    @PostMapping("/create")
    public ResponseEntity<ApiResponse<ShipmentDetailsDto>> createShipment(@RequestBody
ShipmentCreateDto shipmentCreateDto) {
        return new ResponseEntity<>(new
ApiResponse<>(shipmentService.createShipment(shipmentCreateDto)), HttpStatus.CREATED);
    }

    @DeleteMapping("/cancel/{orderToken}")
    public ResponseEntity<ApiResponse<Boolean>> cancelShipment(@PathVariable String
orderToken) {
        return ResponseEntity.ok(new ApiResponse<>(AppConstants.SUCCESS_MESSAGE,
shipmentService.cancelShipment(orderToken), AppConstants.SUCCESS_CODE));
    }
}
```

Note

For both the projects we provide the configurations in application-dev.properties file and activate the profile in application.properties.

application-dev.properties

```
server.port=8086 //different for different applications
spring.datasource.url=jdbc:mysql://localhost:3306/ecom
spring.datasource.username=hbstudent
spring.datasource.password=hbstudent
```

```
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.jpa.properties.hibernate.generate_statistics=true


spring.output.ansi.enabled=always


spring.jpa.generate-ddl=true
spring.jpa.hibernate.ddl-auto = update
spring.jpa.show-sql=true
spring.jpa.properties.hibernate.format_sql=true
spring.jpa.properties.hibernate.jdbc.time_zone=UTC


spring.config.import=optional:secrets.properties

logging.level.org.springframework.security=DEBUG
```

application.properties

```
spring.threads.virtual.enabled=true

spring.profiles.active=dev
```

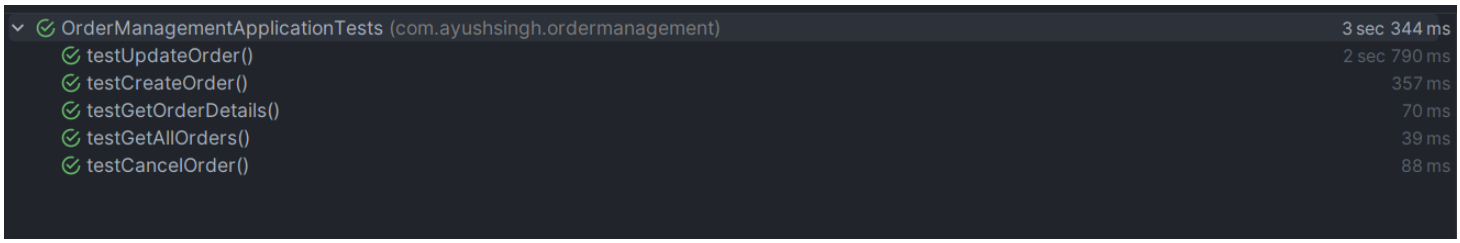
Tests for Order Management

```
package com.ayushsingh.ordermanagement;

import com.ayushsingh.ordermanagement.model.dto.Order.OrderCreateDto;
import com.ayushsingh.ordermanagement.model.dto.Order.OrderItemDto;
import com.ayushsingh.ordermanagement.model.dto.Order.OrderUpdateDto;
import io.restassured.RestAssured;
import io.restassured.http.ContentType;
import org.junit.jupiter.api.BeforeAll;
import org.junit.jupiter.api.Test;
import org.springframework.boot.test.context.SpringBootTest;

import java.util.HashSet;
import java.util.Set;

import static io.restassured.RestAssured.given;
import static org.hamcrest.Matchers.equalTo;

@SpringBootTest(webEnvironment = SpringBootTest.WebEnvironment.DEFINED_PORT)
class OrderManagementApplicationTests {

    private static final String BASE_URL = "http://localhost:8085/api/v1/order";
    private static final String NEW_ORDER_ENDPOINT = "/new";
    private static final String ALL_ORDERS_ENDPOINT = "/all";



    @BeforeAll
    public static void setUp() {
        RestAssured.baseURI = BASE_URL;
    }

    @Test
    void testCreateOrder() {
        OrderCreateDto orderCreateDto = new OrderCreateDto();
        orderCreateDto.setCustomerName("Ayush Singh");
        orderCreateDto.setAddress("Ayush's Address");
        Set<OrderItemDto> orderItems=new HashSet<>();
        String[] productsTokens={
                "a8152a2c-5a5b-40f4-8a16-ef6f70e81aae",
                "2f82b9d7-8029-4a9b-a59f-9a371cbdc68e",
                "6a7a3f9c-1f12-4c63-8bcb-75c12d8f8a17"
        };
        for(int i=0;i<productsTokens.length;i++){
            OrderItemDto orderItemDto = new OrderItemDto();
            orderItemDto.setProductToken(productsTokens[i]);
```

```
            orderItemDto.setQuantity((long) (i+1));
            orderItems.add(orderItemDto);
        }
        orderCreateDto.setProducts(orderItems);
        given()
                .contentType(ContentType.JSON)
                .body(orderCreateDto)
                .when()
                .post(NEW_ORDER_ENDPOINT)
                .then()
                .assertThat()
                .statusCode(201);
    }

    @Test
    void testGetOrderDetails() {
        String orderToken = "baa17db6-ee47-4fc3-ac83-055be028307a";

        given()
                .pathParam("orderToken", orderToken)
                .when()
                .get("/{orderToken}")
                .then()
                .assertThat()
                .statusCode(200)
                .body("data.customerName", equalTo("Ayush Singh"));
    }



    @Test
    void testUpdateOrder() {
        String orderToken = "baa17db6-ee47-4fc3-ac83-055be028307a";

        OrderUpdateDto orderUpdateDto = new OrderUpdateDto();
        orderUpdateDto.setCustomerName("Ayush Singh");
        orderUpdateDto.setAddress("Uttar Pradesh");

        given()
                .pathParam("orderToken", orderToken)
                .contentType(ContentType.JSON)
                .body(orderUpdateDto)
                .when()
                .patch("/{orderToken}")
                .then()
                .assertThat()
                .statusCode(200)
                .body("data", equalTo(orderToken));
    }

    @Test
    void testGetAllOrders() {
        given()
                .when()
                .get(ALL_ORDERS_ENDPOINT)
                .then()
                .assertThat()
                .statusCode(200);
    }

    @Test
    void testCancelOrder() {
        String orderToken = "baa17db6-ee47-4fc3-ac83-055be028307a";

        given()
                .pathParam("orderToken", orderToken)
                .when()
                .delete("/{orderToken}")
                .then()
                .assertThat()
                .statusCode(200)
```

```
                .body("data", equalTo("Order with id: " + orderToken + " deleted
successfully!"));
    }
}
```

Results



API

Create Order

POST: http://localhost:8085/api/v1/order/new

Request Body:

```
{
    "address": "Customer's address",
    "customerName": "Customer's name ",
    "products": [
        {
            "productToken": "6a7a3f9c-1f12-4c63-8bcb-75c12d8f8a17",
            "quantity": 3
        },
        {
            "productToken": "2f82b9d7-8029-4a9b-a59f-9a371cbdc68e",
            "quantity": 2
        }
    ]
}
```

Response Body:

```
{
    "message": "Success",
    "data": "Order with id: baa17db6-ee47-4fc3-ac83-055be028307a created successfully!",
    "code": 2000
}
```

List all products

GET: http://localhost:8085/api/v1/product/all

Response Body:

```
{
    "message": "Success",
    "data": [
        {
            "productName": "Product 1",
            "productToken": "6a7a3f9c-1f12-4c63-8bcb-75c12d8f8a17"
        },
        {
            "productName": "Product 2",
            "productToken": "2f82b9d7-8029-4a9b-a59f-9a371cbdc68e"
        },
        {
            "productName": "Product 3",
```

```
            "productToken": "a8152a2c-5a5b-40f4-8a16-ef6f70e81aae"
        },
        {
            "productName": "Product 4",
            "productToken": "f4f2b6b9-106d-478f-9a19-00c6cd7e23e5"
        },
        {
            "productName": "Product 5",
            "productToken": "c74a8631-99af-446f-a0f6-30ac3d4a85e2"
        },
        {
            "productName": "Product 6",
            "productToken": "63f7ee67-cc42-49b2-8d95-c7f562b508c9"
        }
    ],
    "code": 2000
}
```


Get order details

GET: http://localhost:8085/api/v1/order/baa17db6-ee47-4fc3-ac83-055be028307a

Response Body:


```
{
    "message": "Success",
    "data": {
        "orderToken": "baa17db6-ee47-4fc3-ac83-055be028307a",
        "orderStatus": "PLACED",
        "address": "Customer's address",
        "customerName": "Customer's name ",
        "orderItems": [
            {
                "productToken": "2f82b9d7-8029-4a9b-a59f-9a371cbdc68e",
                "quantity": "2",
                "productName": "Product 2"
            },
            {
                "productToken": "6a7a3f9c-1f12-4c63-8bcb-75c12d8f8a17",
                "quantity": "3",
                "productName": "Product 1"
            }
        ]
    },
    "code": 2000
}
```


Update Order

PATCH: http://localhost:8085/api/v1/order/baa17db6-ee47-4fc3-ac83-055be028307a

Request Body:


```
{
    "address": "New address 1",
    "customerName": "Customer name 2"
}
```


Response Body:


```
{
    "message": "Success",
```

```
    "data": "baa17db6-ee47-4fc3-ac83-055be028307a",
    "code": 2000
}
```


Delete Order

DELETE: http://localhost:8085/api/v1/order/00b3ffb4-f7d6-4e8f-b54d-a85c43267783

Response Body:


```
{
    "message": "Success",
    "data": "Order with id: 00b3ffb4-f7d6-4e8f-b54d-a85c43267783 deleted successfully!",
    "code": 2000
}
```


Source Code links

The complete source code can be found here:

https://github.com/singhayush20/Assignments/tree/main/Spring%20Boot/Wk07