Instructions:

1. Deadline - Will be shared over email. Please attach the code as a GIT repo URL or zip/rar file to the mail once done.
2. **Partial submissions are allowed.**

Question 1: Develop and Dockerize a Java Spring Boot Application with PostgreSQL and RabbitMQ

Problem Statement:

You need to create a REST API for managing orders in a retail system. The API should support the following operations:

1. **Add a new order**: `POST /orders`
2. **Get details of an order by its ID**: `GET /orders/{id}`
3. **Update the details of an order**: `PUT /orders/{id}`
4. **Delete an order by its ID**: `DELETE /orders/{id}`
5. **List all orders**: `GET /orders`

Each order should have the following fields:

- `id`: Long (Primary Key)
- `customerName`: String
- `orderDate`: Date
- `totalAmount`: Double
- `status`: String

In addition to the REST API, you need to implement a messaging system using RabbitMQ to handle order processing. When a new order is created, an order confirmation message should be sent to a RabbitMQ queue.

Requirements:

1. Use Java Spring Boot to create the REST API.
2. Use PostgreSQL as the database.
3. Implement proper exception handling and return appropriate HTTP status codes.
4. Use JPA (Java Persistence API) for ORM (Object-Relational Mapping).
5. Write unit tests for the service layer.
6. Dockerize the application.
7. Use RabbitMQ for message queuing.
8. Write a message listener to process the order confirmation messages.

Evaluation Criteria:

1. Correctness and completeness of the API implementation.
2. Code quality and best practices (e.g., proper use of Spring annotations, error handling).
3. Correctness of the Docker setup (Dockerfile).
4. Correctness and functionality of the RabbitMQ integration.
5. Test coverage and quality of unit tests.
6. Ability to follow instructions and meet the requirements.

Question 2: Create a Java Spring Boot Application with MongoDB and Docker

Problem Statement:

You need to create a REST API for managing a product catalog in an online store. The API should support the following operations:

1. **Add a new product**: `POST /products`
2. **Get details of a product by its ID**: `GET /products/{id}`
3. **Update the details of a product**: `PUT /products/{id}`
4. **Delete a product by its ID**: `DELETE /products/{id}`
5. **List all products**: `GET /products`

Each product should have the following fields:

- `id`: String (Primary Key)
- `name`: String
- `description`: String
- `price`: Double
- `category`: String

In addition to the REST API, you need to implement a RabbitMQ message queue to handle product updates. When a product is updated, a message should be sent to a RabbitMQ queue to notify other services of the update.

Requirements:

1. Use Java Spring Boot to create the REST API.

2. Use MongoDB as the database.
3. Implement proper exception handling and return appropriate HTTP status codes.
4. Write unit tests for the service layer.
5. Dockerize the application.
6. Use RabbitMQ for message queuing.
7. Write a message producer to send product update messages to a RabbitMQ queue.

Evaluation Criteria:

1. Correctness and completeness of the API implementation.
2. Code quality and best practices (e.g., proper use of Spring annotations, error handling).
3. Correctness of the Docker setup (Dockerfile).
4. Correctness and functionality of the RabbitMQ integration.
5. Test coverage and quality of unit tests.
6. Ability to follow instructions and meet the requirements.