Steps To start the application:

- **Start Eureka server first which will get started at port number 8761**
- **Start bookingService which will get started at port 8081**
- **Start paymentService which will get started at port 8083**

NOTE: Have not created API Gateway

Booking Service:

BookingService structure-

Made an embedded h2 database and added relevant dependency
Made a package entity and made an entity class BookingInfoEntity which will get save into booking database.
Similarly, made another packages dao,dto,controller and service and BookingDAO, BookingDTO, BookingController, BookingService and BookingServiceImpl as their respective java classes and interface.

BookingService endpoint 1-
Api- “http://localhost:8081/hotel/booking”

A post mapping end point createBooking which will save the passed DTO into database after converting into database entity by using modelmapper(relevant dependency added).
acceptBookingDetails method is added in BookingService interface and its implementation is provided by BookingServiceImpl

As part of acceptBookingDetails totalDays are calculated based on toDate and fromDate parameters and then roomPrice is set by the logic - 1000*totalRooms*totalDays.
A function randomRoomNumbers is written which will randomly generate numbers between 1 and 100 and convert them to String.
After we get all these details , save method from **BookingDAO** which extends JPA repository is called and Booking is saved into database.
After booking is saved a savedBookingDTO object is returned by createBooking method(Post mapping endpoint) with HTTP status as created.

BookingService endpoint 2-

Api-"http://localhost:8081/hotel/booking/{bookingId}/transaction"

A post mapping end point createTransaction which will save the passed PaymentDTO into database after converting into database entity by using modelmapper(relevant dependency added).As part of paymentDTO it contains transactionId which was 0(default) while booking entity was saved
Now, there are certain condition as part of this endpoint,

1. If there is no entity saved in database with this booking id we return a response entity with message "invalid booking id" and HTTP status as BadRequest.
2. If the payment method of paymentDTO is neither "UPI" not "CARD" then we return a response entity with message "invalid mode of payment" and HTTP status as BadRequest.

acceptTransactionDetails method is added in BookingService interface and its implementation is provided by BookingServiceImpl.
It finds an Optional bookingInfoEntity based on bookingId by using findById method of JPA repository and if no booking is present it returns null otherwise .get() method is used to retrieve booking entity.
PaymentServiceURL through which the transaction entity is saved in database is hardcoded and then rest template is used to communicate with payment service

(synchronus communication) to retrieve transactionId and that Id is saved into bookingService entity using setTransactionId method.Then bookingService entity which consist transactionId is saved into BookingDatabase.

In the controller class BookingCompleted message is printed into console and savedBookingDTOWithTransactionId object is returned as response entity (converted through modal mapper) and HTTP status as created.

Payment Service:

PaymentService Structure-

Made an embedded h2 database and added relevant dependency
Made a package entity and made an entity class TransactionDetailsEntity which will get save into payment database.
Similarly, made another packages dao,dto,controller and service and PaymentDAO, PaymentDTO, PaymentController, PaymentService and PaymentServiceImpl as their respective java classes and interface.

PaymentService endpoint1-

Api-“http://localhost:8083/payment/transaction”

A post mapping end point createBooking which will save the passed DTO into database after converting into database entity by using modelmapper(relevant dependency added).
acceptTransactionDetails method is added in PaymentService interface and its implementation is provided by PaymentServiceImpl.
As part of this function a booking after payment transaction is saved in paymentService database using inbuilt save method of JPA repository

As part of Api response transactionId of savedTransactionDTO is returned as response entity with HTTP status created.

PaymentService endpoint2-

Api-"http://localhost:8083/payment/transaction/{transactionId}"
A get mapping endpoint getTransactionDetailsBasedOnId ,which will give transaction based on transactionId.
getTransactionDetails method is added in PaymentService interface and its implementation is provided by PaymentServiceImpl.
This method uses inbuilt method findById of JPA repository to fetch a transaction based on Id.

Eureka Service -

Added relevant dependency of Eureka Server in pom.xml
Few changes incorporated in application.properties-
1.Server post is set to 8761(as given in instruction)
2.Eureka client is set to false(it is set to true in other two application)

Main class is annotated with @EnableEurekaServer

NOTE-
1.PaymentService and BookingService are given their names in their respective application.properties as PAYMENT-SERVICE and BOOKING-SERVICE
2.Default port of payment service is set to 8083 and booking service is 8081