Steps to run the project: make sure that the `spring.jpa.hibernate.ddl-auto=create`
Is not commented for running the first time, first run the Eureka server and then run the payment service and the booking service. Run the API-GATEWAY in the last. In this project I did not use the API-GATEWAY to make the post call from booking service to payment service. The payment service is configured to run on the port 8083 and Booking service is confiugred to run on the port number 8081. To check wether the services are registered in Eureka server use the localhost:8761 to verify it.

Booking service logic explanation:

BookingController:
Handles the end points for booking
/hotel/booking is handled by the method InitiateBooking it's return type is BookingInfoEntity it calls the RoomBookingService.bookRoom method and returns the result

/hotel/{bookingId}/transaction is handled by the method getBookingInformation
This first validates the payment mode and throws an exception if the payment mode is not card or upi and then it will fetch the booking information from the booking id that is given and it sends http request through the restTemplate to payment service and receive the response from the request and returns the result. If the result is successful then it calls the RoomBookingUpdateService and update the status of the booking.

BookingDAO:
It inherits JPARepository<BookingInfoEntity,Integer> which can be used instead of writing sql queries.

BookingInfoEntity:
BookingInfoEntity is a entity class that represents the database table. It has id has its key.

RandomRoomNumberService:
RandomRoomNumberService contains a method getRoomNumbers which accepts integer as a parameter and returns the random room numbers as a string with comma's seperated.

```
RoomBookingQueryService:
It is a interface for defining the methods for querying the data

RoomBookingQueryServiceImpl
```

It contains the implementation of RoomBookingQueryService in getBookingInformation, it is used to fetch the booking information when the booking id is given, if no record is found it will throw InvalidBookingIdException, else it will return the booking information

RoomBookingService:
It is a interface for defining the methods for booking room

RoomBookingServiceImpl
It contains the implementation for booking the room in bookRoom method, it takes the booking information and calculate the number of days required and the price to pay and makes the transaction id as 0 initially and stores all the data into database
Note: if you want to disable booking on a date before today uncomment
The block mentioned in bookRoom method

RoomBookingUpdateService:
It is a interface the methods for updating the booking status

RoomBookingUpdateServiceImpl
It contains the logic to update the booking information in updateBookingStatus.

Validators
Validators class contains a static method paymentModeValidator which is used to validate the payment mode if it is card or upi and it will throw InvalidPaymentException if payment mode is something other than card or upi

Validator:
Validator class contains the method paymentModeValidator which validates the payment details sent and if the details are incorrect i.e if the payment mode is not card or upi or if empty card number is sent when the payment mode is card or if empty upi id is sent when upi id is payment mode it will throw a InvalidPaymentCredentialsException

AOPExceptionHandlers:

InvalidBookingExceptionHandler:
InavlidBookingExceptionHandler is used to handle the InvalidBookingIdException it creates the object for ErrorResponse using the message from exception and gives HttpStatus.BAD_REQUEST.

InvalidDatesPassedExceptionHandler
InvalidDatesPassedExceptionHandler is used to handle the InvalidDatesPassedException. it creates the object for ErrorResponse using the message from exception and gives HttpStatus.BAD_REQUEST.

InvalidPaymentModeExceptionHandler

InvalidPaymentModeExceptionHandler is used to handle the InvalidPaymentModeExcpetion ,InvalidDatesPassedException. it creates the object for ErrorResponse using the message from exception and gives HttpStatus.BAD_REQUEST.

Payment Service:

PaymentController:

recievePayment method of the PaymentController handles the /payment/transaction post method which is used to make the payment it first validates the payment mode and if the payment mode is not card or upi it will throw an exception and then it passes data to the PaymentService.makePayment and receive the payment response and returns it.

getPaymentDetails method of the PaymentController handles the /payment/transaction/{id} post method which returns the transaction by passing the id if no transaction with the given id is found TransactionNotFoundException is thrown.

TransactionDAO
It inherits JpaRepository<TransactionDetailsEntity,Integer>, it is used to access the database and perform crud operations

TransactionDetailsEntity:
It is a entity class which represents the table payment in database

PaymentQueryService
PaymentQueryService is a interface that contains methods that are used to query transaction data

PaymentQueryServiceImpl:
getTransaction method in PaymentQueryServiceImpl contains the logic which fetches the transaction details with given transaction id if no transactions are present with the given id it will return null.

PaymentService
PaymentService is a interface that contains the methods that are used to implement payment module

PaymentServiceImpl
makePayment method in PaymentServiceImpl contains the logic for implementing the payment module it will receive the payment details and make use of TransactionDao and store the data in database and returns the object of TransactionDetailsEntity.

Validator:
Validator class contains the method paymentModeValidator which validates the payment details sent and if the details are incorrect i.e if the payment mode is not card or upi or if empty card number is sent when the payment mode is card or if empty upi id is sent when upi id is payment mode it will throw a InvalidPaymentCredentialsException

AOPExceptionHandlers:

InvalidPaymentCredentialExceptionHandler :
InvalidPaymentCredentialExceptionHandler is used to handle the InvalidPaymentCredentialsException,it creates the object for ErrorResponse using the message from exception and gives HttpStatus.BAD_REQUEST.

TransactionNotFoundExceptionHandler :
InvalidPaymentCredentialExceptionHandler is used to handle the TransactionNotFoundException,it creates the object for ErrorResponse using the message from exception and gives HttpStatus.BAD_REQUEST.