There are 4 Services in the project **Sweet-Home**.

1. api-gateway-master (API Gateway)
2. service_registry (Eureka Client)
3. Booking Service
4. Payment Service

Brief description of services:

1. Api-gateway-master(API Gateway):

   This service acts as a gateway for all user requests. Instead of exposing the Booking and Payment services, this gateway interacts with the users and re-routes the requests to the relevant service internally.

   Application.yml file is configured for running the server of this service on port no 9191 for including routers for Booking Service with service name BOOKING-SERVICE, for Payment Service with service name PAYMENT-SERVICE. Also Eureka Client is enabled for it.
2. Service-registry(Eureka Client):

   This service is the Eureka client running on its default port no 8761. All the other services have registered on it for communication.
3. Booking Service:
   - This Service has 2 api endpoints as

   a. http://localhost:9191/hotel/booking: This endpoint is responsible to collect information like fromDate, toDate,aadhar number,count from user and save it in its database and also returns the corresponding booking details. HTTP Method is Post and requestbody is like below:

```
{
  "fromDate": "2021-06-10",
  "toDate": "2021-06-15",
  "aadharNumber": "Sample-Aadhar-Number",
  "numOfRooms": 3
}
```

   Content type is JSon

   Reponse is like below:

```
{
  "id": 1,
  "fromDate": "2021-06-10",
  "toDate": "2021-06-15",
  "aadharNumber": "Sample-Aadhar-Number",
  "numOfRooms": 0,
  "roomNumbers": "[34, 9, 2]",
  "roomPrice": 15000,
  "transactionId": 0,
  "bookedOn": "2022-01-27T19:04:40.0965989"
}
```

This api is achieved in method name “searchRoomsInHotel” in BookingServiceController class inside controller package in Booking Service.

b. http://localhost:9191/hotel/booking/1/transaction : This endpoint is responsible to take the payment-related details from the user and send it to the payment service; get the transactionId and save the corresponding transactionId in the booking table. For the paymentMode, if the user provides any other input apart from the ’UPI’ or ‘CARD’, then it means that the user is not interested in the booking and wants to opt-out and this endpoint will throw a custom response with the message "Invalid mode of payment".

HTTP Method is Post and requestbody is like below:

```
{
  "paymentMode": "UPI",
  "bookingId": 1,
  "upiId": "upi details",
  "cardNumber": ""
}
```

Content type is JSon

Reponse is like below:

```
{
  "id": 1,
  "fromDate": "2021-06-10",
  "toDate": "2021-06-15",
  "aadharNumber": "Sample-Aadhar-Number",
  "numOfRooms": 0,
  "roomNumbers": "[34, 9, 2]",
  "roomPrice": 15000,
  "transactionId": 1,
  "bookedOn": "2022-01-27T19:04:40.096599"
}
```

This api is achieved in method name “bookRoomsInHotel” in BookingServiceController class inside controller package in Booking Service

- Booking service has 7 packages as below

a. Controller: This package have a class BookingServiceController which acts as a controller layer and accepts the apis and make us of injected bean of service layer to process the logic and sends back the response.

b. Dao: This package has BookingInfoEntityDao interface which extends JPArepository and acts as a DAO layer.

c. Dto: This package 2 classes which are 2 types of data transfer objects namely BookingInfoEntityDto and TransactionDetailsDto. TransactionDetailsDto is created to address the input request body for the api http://localhost:9191/hotel/booking/1/transaction .

d. Entities: This package has the BookingInfoEntity which is constructed as per database schema.

e. Feign : This package has a class PaymentServiceClient which is internally called in the method “bookRoomsInHotel” in BookingServiceController to get the transaction id with successful payment after booking. With the help of api-gateway service the feign client has achieved abstraction by using api-gateway as name parameter in @FeignClient annotation.

f. Service : This is the service layer package having interface BookingServiceService implemented by BookingServiceServiceImpl which has methods which are implemented as described below.

   i. acceptBookingDetails: This saves the entity into database and returns the saved entity.

   ii. updateTransactionForBooking: This updates the entity and returns the updated entity.

   iii. getBookingDetailsById: This returns the bookingInfoEntity based the Id.

g. Utils: This has a POJOConverter which has two methods for converting dto pojo to entity pojo and viceversa.

4. Payment Service:

   - This service has 2 api endpoints as

   a. http://localhost:9191/payment/transaction : The API to make transaction in the Booking service will interact with this endpoint to pass on the transaction details and get the transaction Id in return.
   HTTP Method is Post and request body is like below:
   ```
   {
     "paymentMode": "UPI",
     "bookingId": 1,
     "upiId": "upi details",
     "cardNumber": ""
   }
   ```
   This api is called in booking service controller layer class BookingServiceController in the method “booksRoomsInHotel”. Above Request Body is taken as input through Dto object “transactionDto”and passed through feign client of Payment service which is “PaymentServiceClient” which through the method “doTransaction” in PaymentServiceController saves the transaction entity in database and returns the transaction id. The response of this Api is the transaction id which is of Integer type. After getting the transaction id through this Api endpoint , using this transaction id booking service updates the booking infoentity by calculating the room price and setting/updating it for the entry as per the booking id.

   b. http://localhost:9191/payment/transaction/1 : This API is used to query the transaction details for a given transaction Id.
   HTTP Method is GET and path variable is booking id which is 1 in the above api. Response is the transaction details as below:
   ```
   {
     "transactionId": 1,
     "paymentMode": "UPI",
     "bookingId": 1,
   ```

```
  "upiId": "upi details",
  "cardNumber": ""
}
```

This api is called in PaymentServiceController method “getTransactionDetails”. It fetches the entity from database through service layer injected bean paymentServiceService and returns it after converting it to Dto object.

- Payment Service has 5 packages

a. Controller: This package have a class PaymentServiceController which acts as a controller layer and accepts the apis and make us of injected bean of service layer to process the logic and sends back the response.
b. Dao : This package has TransactionDetailsEntityDao interface which extends JPArepository and acts as a DAO layer.
c. Dto : This package has TransactionDetailsEntityDto class which is data transfer object for api. It imitates the entity class.
d. Entities : This package has the TransactionDetailsEntity which is constructed as per database schema.
e. Service: This is the service layer package having interface PaymentServiceService implemented by PaymentServiceServiceImpl which has methods which are implemented as described below.
    1. saveTransaction : This makes use of Dao layer and saves the entity into the database and returns it.
    2. getTransactionById : This makes use of Dao layer method findById and returns the TransactionDetailsEntity.