DESIGN DOCUMENT

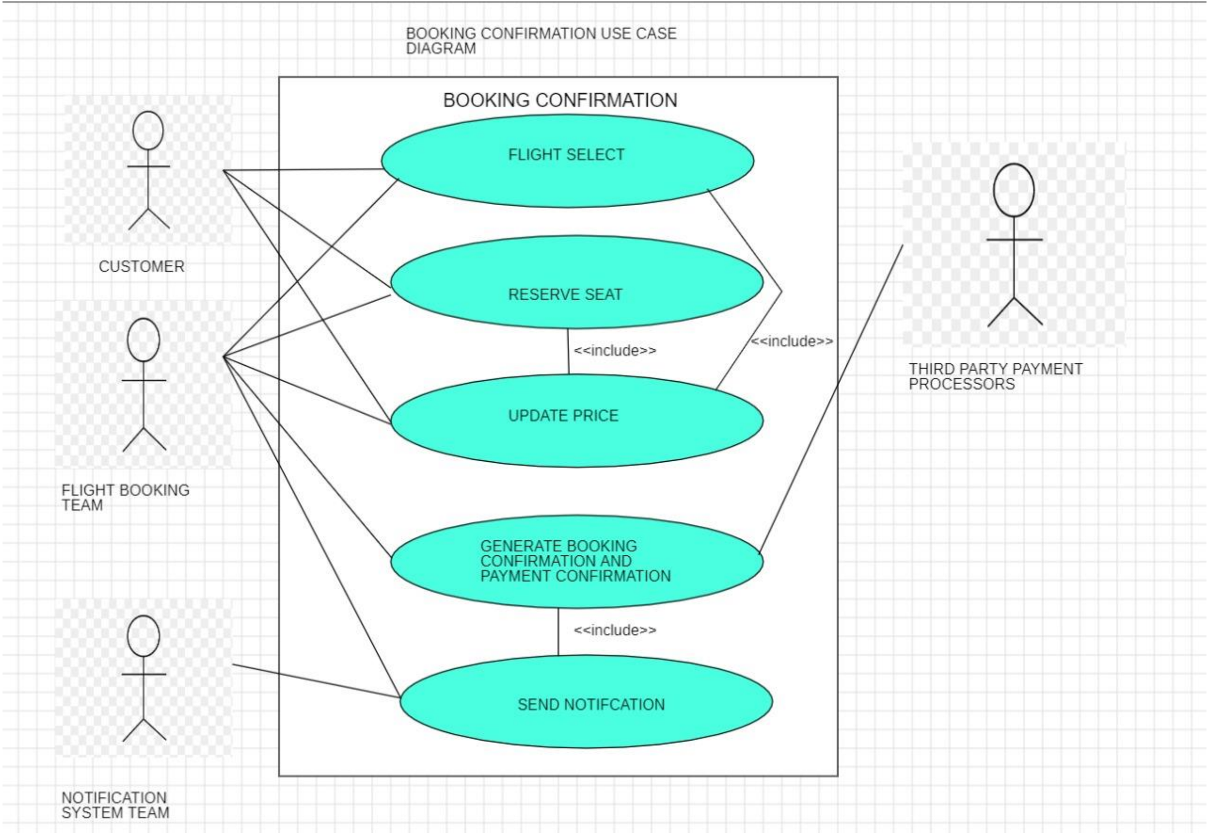
Sai Lithish Degapudi(PES2UG21CS456)

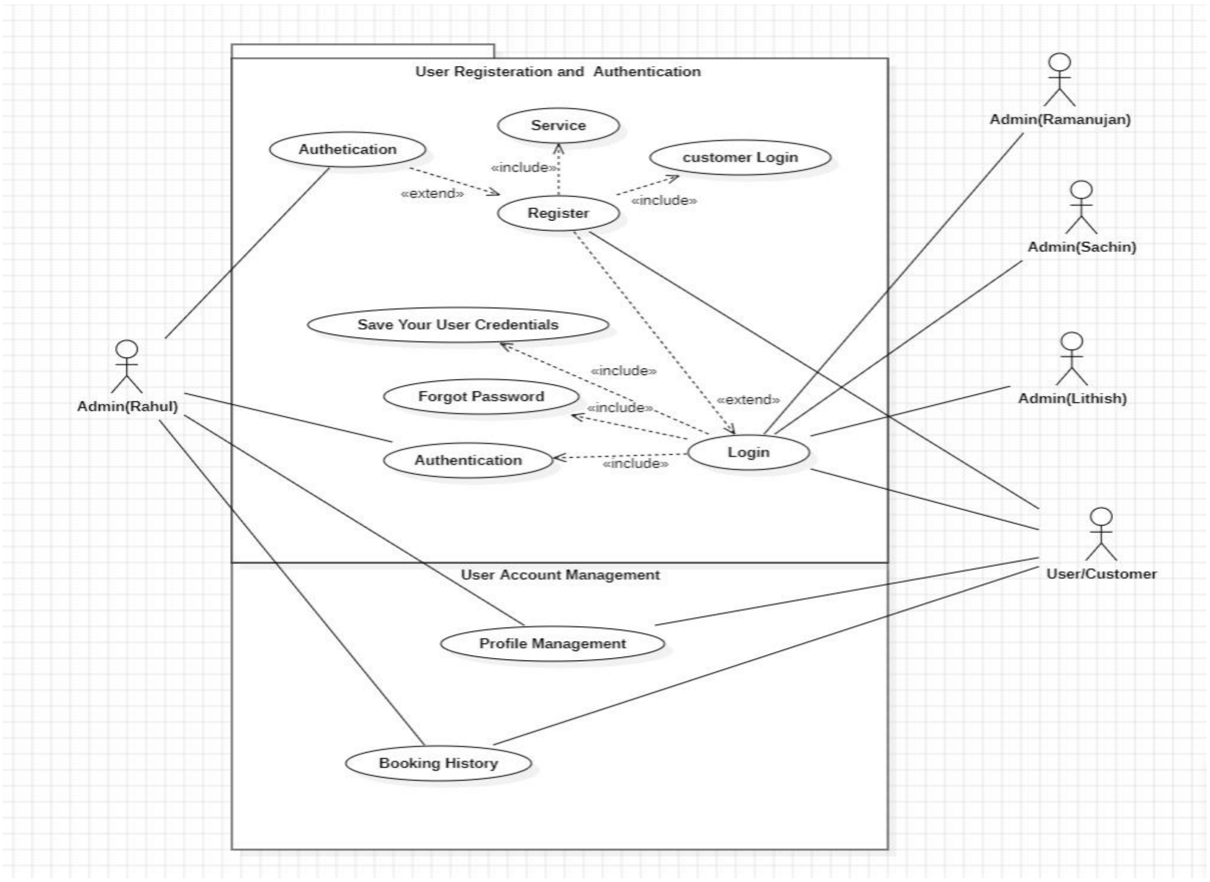
Sachin Ramesh Kulkarni(PES2UG21CS449)

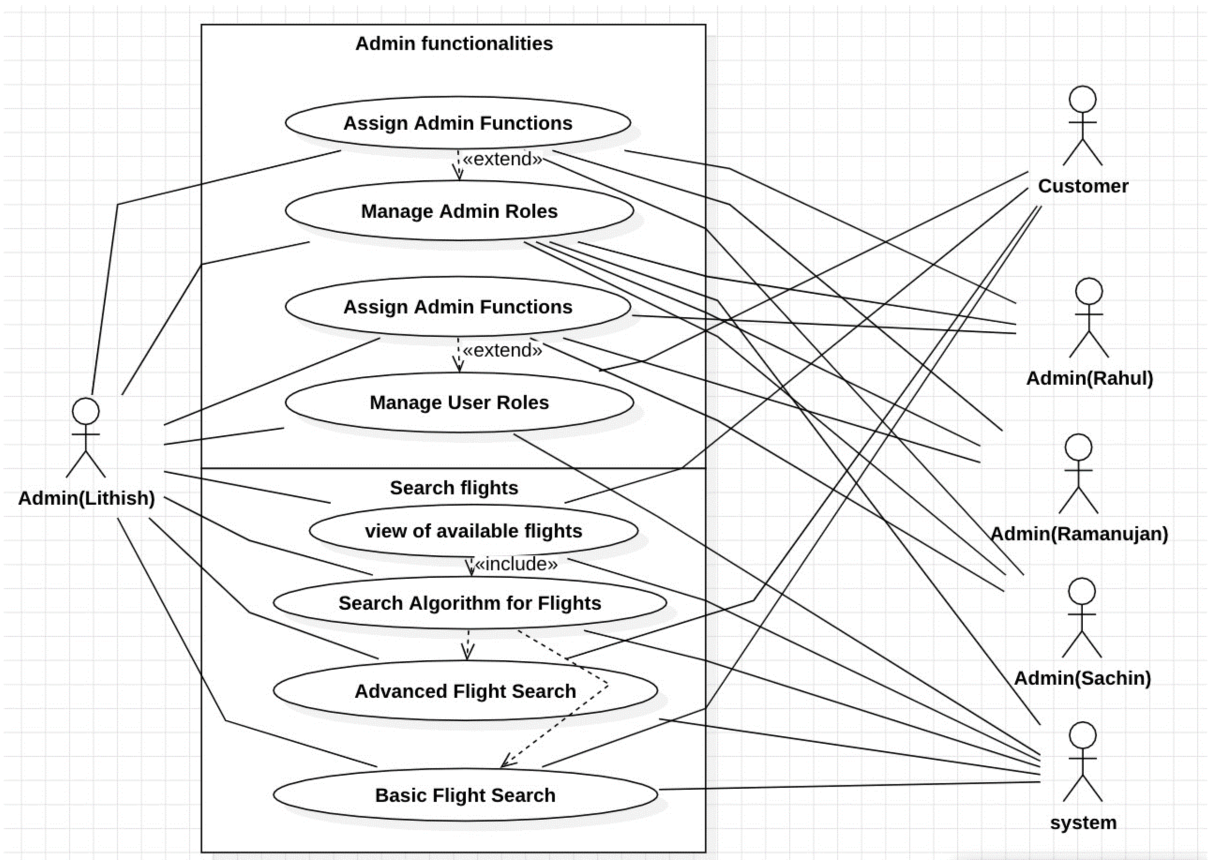
Sai Rahul Reddy Kona(PES2UG21CS458)

S Ramanujan (PES2UG21CS441) ------------> TEAM - 7

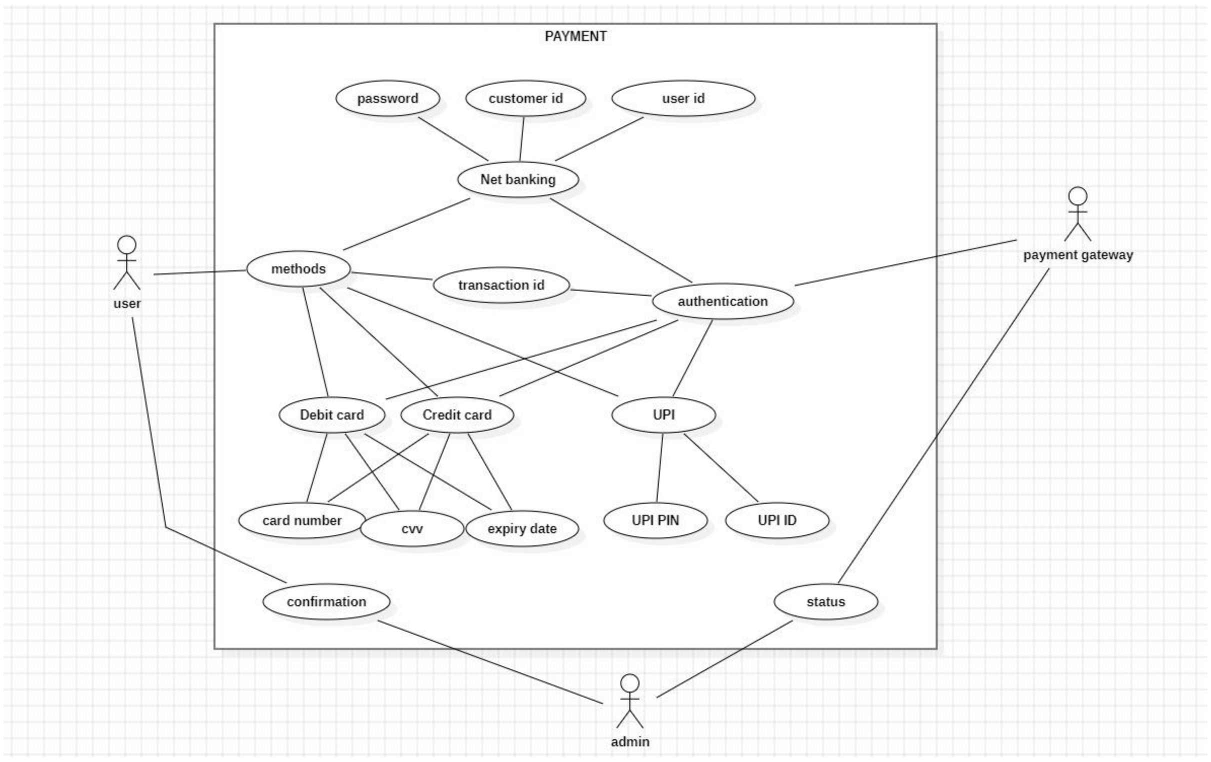
Use case diagrams





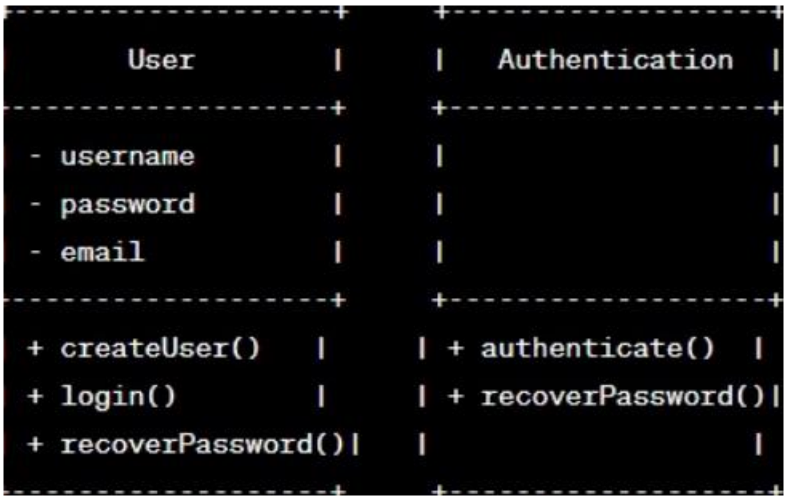


Admin functionalities
Assign Admin Functions
«extend»
Manage Admin Roles
Assign Admin Functions
«extend»
Manage User Roles
Search flights
view of available flights
«include»
Search Algorithm for Flights
Advanced Flight Search
Basic Flight Search
Admin(Lithish)
Customer
Admin(Rahul)
Admin(Ramanujan)
Admin(Sachin)
system

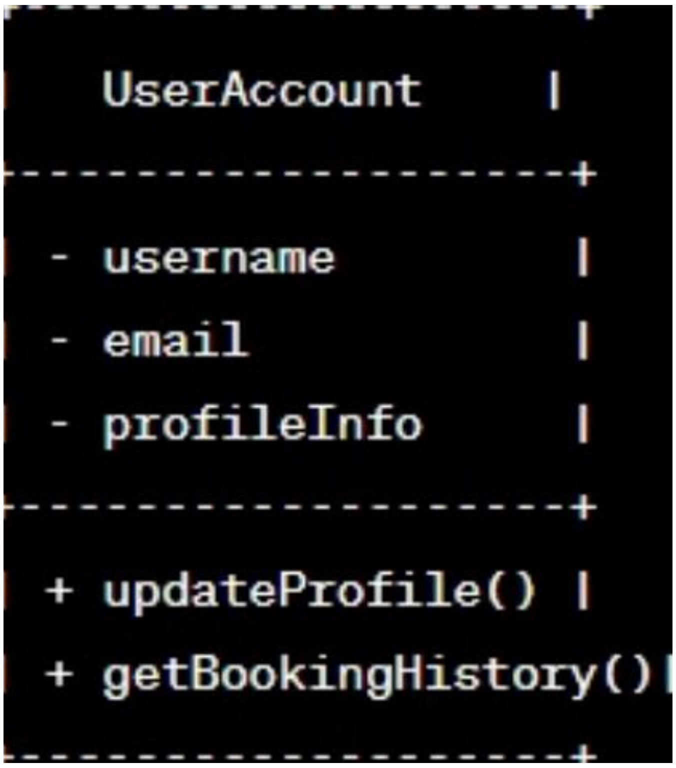


PAYMENT
password
customer id
user id
Net banking
methods
transaction id
authentication
Debit card
Credit card
UPI
card number
cvv
expiry date
UPI PIN
UPI ID
confirmation
status
user
payment gateway
admin

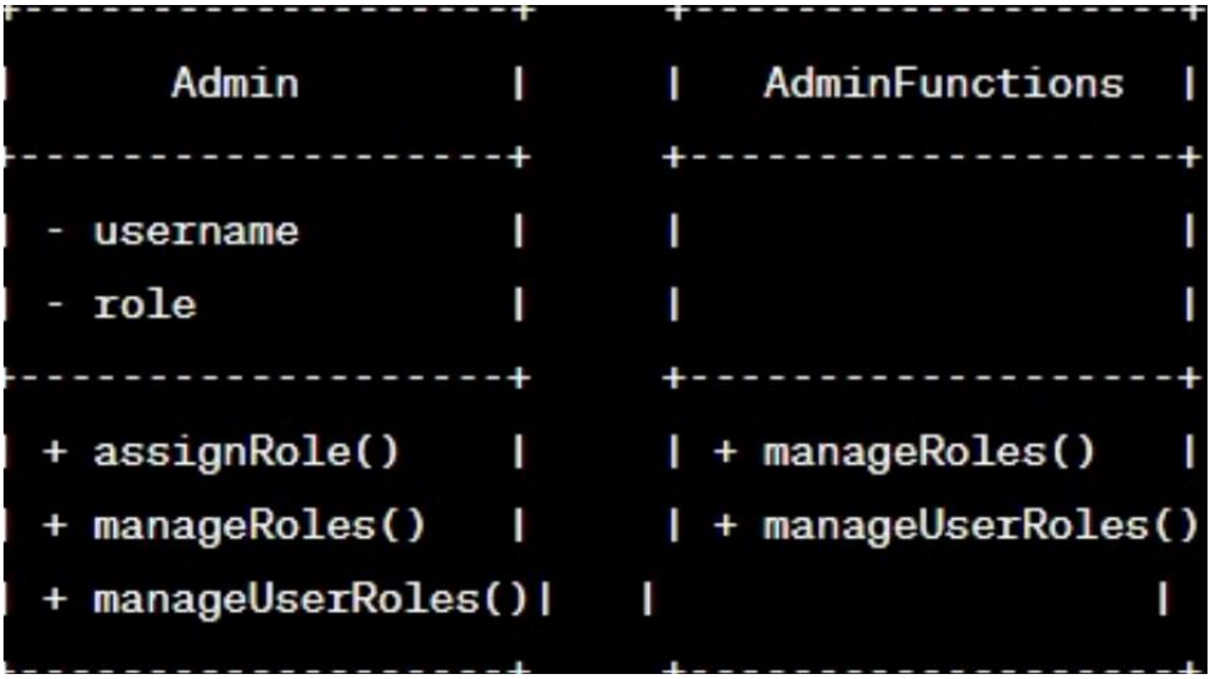
Class diagrams



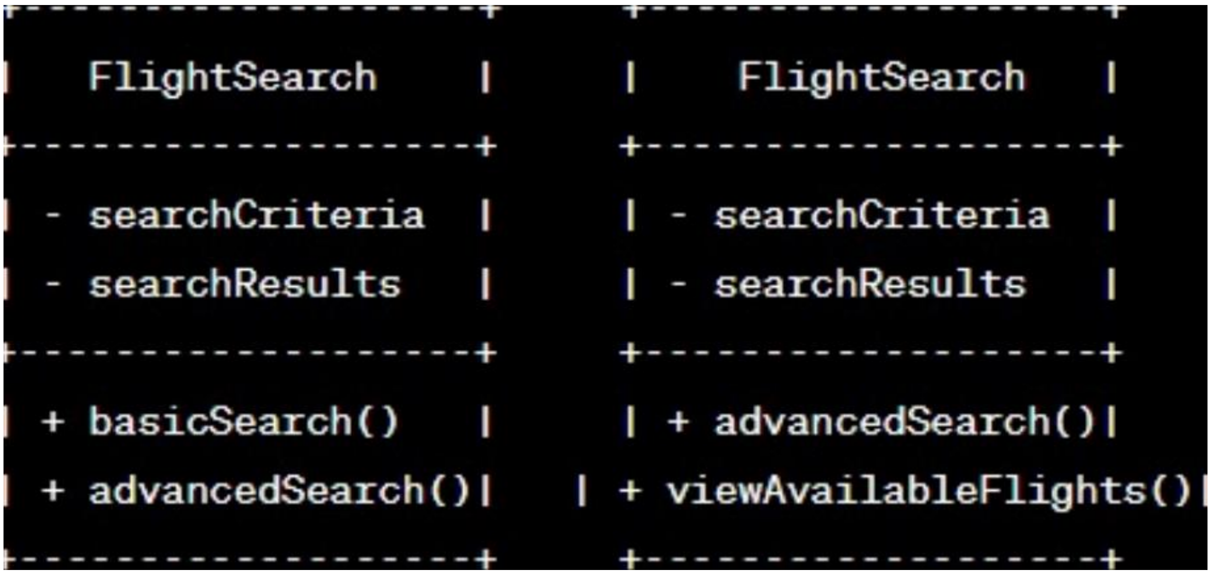
```
+-----------------+      +-----------------+
       User       |      |  Authentication |
+-----------------+      +-----------------+
 - username       |      |                 |
 - password       |      |                 |
 - email          |      |                 |
+-----------------+      +-----------------+
 + createUser()   |     | + authenticate()  |
 + login()        |     | + recoverPassword()|
 + recoverPassword()|   |                   |
+-----------------+      +-----------------+
```



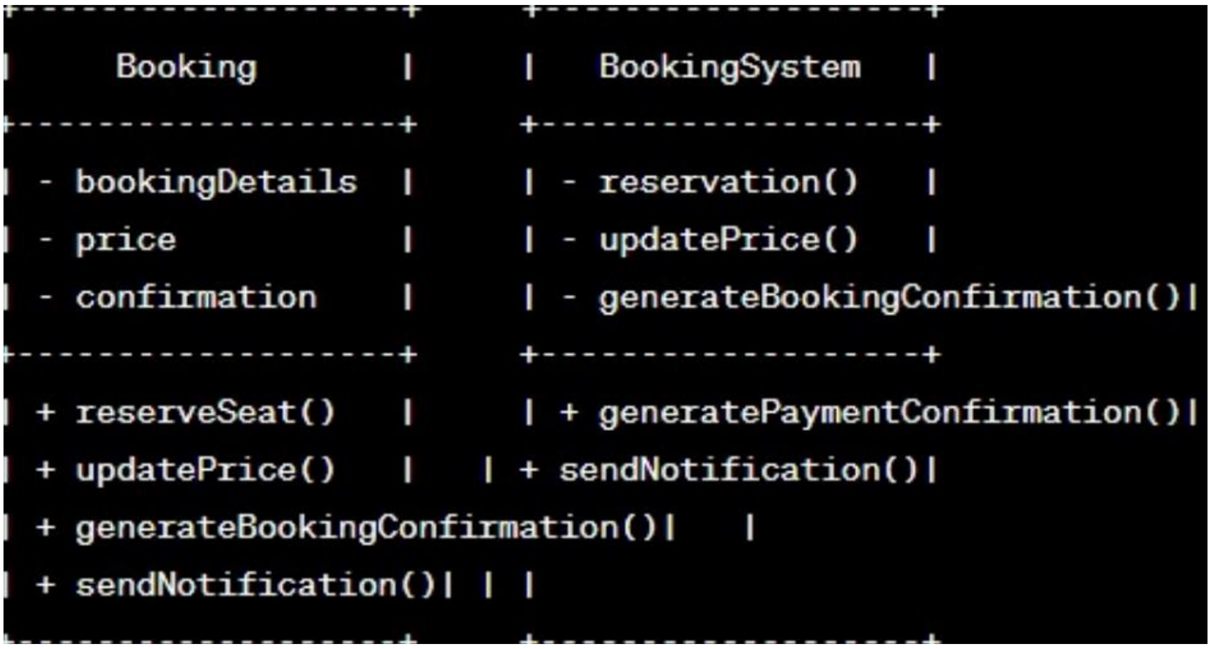
```
+--------------------+
   UserAccount     |
--------------------+
 - username         |
 - email            |
 - profileInfo      |
--------------------+
 + updateProfile()  |
 + getBookingHistory()|
--------------------+
```



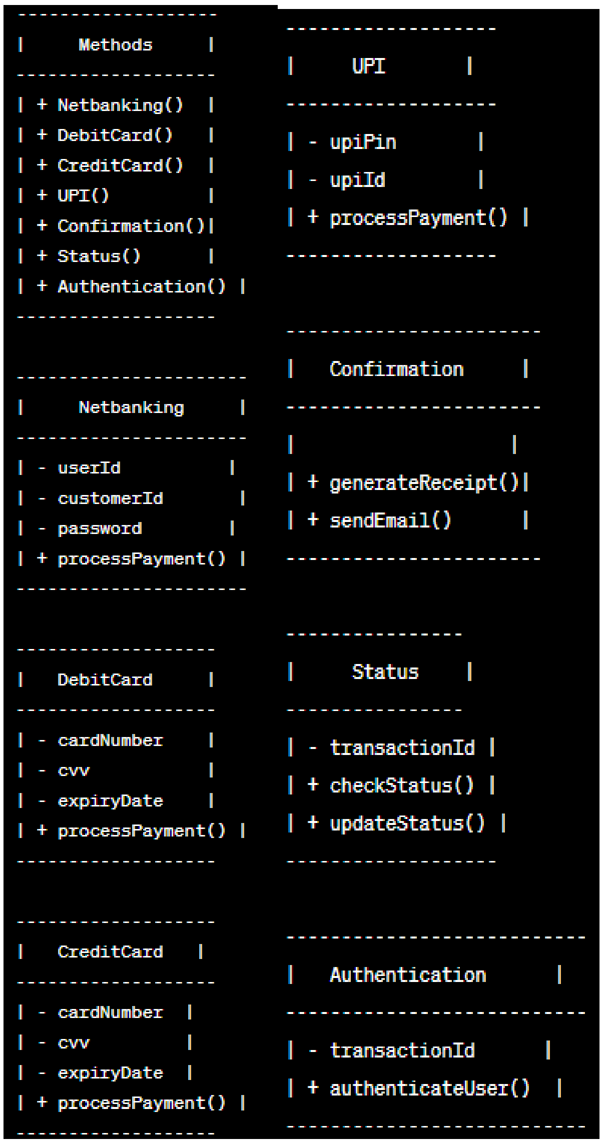
```
+-------------------+     +-------------------+
|      Admin        |     |  AdminFunctions   |
+-------------------+     +-------------------+
| - username        |     |                   |
| - role            |     |                   |
+-------------------+     +-------------------+
| + assignRole()    |     | + manageRoles()   |
| + manageRoles()   |     | + manageUserRoles()
| + manageUserRoles()|    |                   |
+-------------------+     +-------------------+
```



```
+-------------------+     +-------------------+
|   FlightSearch    |     |   FlightSearch    |
+-------------------+     +-------------------+
| - searchCriteria  |     | - searchCriteria  |
| - searchResults   |     | - searchResults   |
+-------------------+     +-------------------+
| + basicSearch()   |     | + advancedSearch()|
| + advancedSearch()|   | + viewAvailableFlights()|
+-------------------+     +-------------------+
```



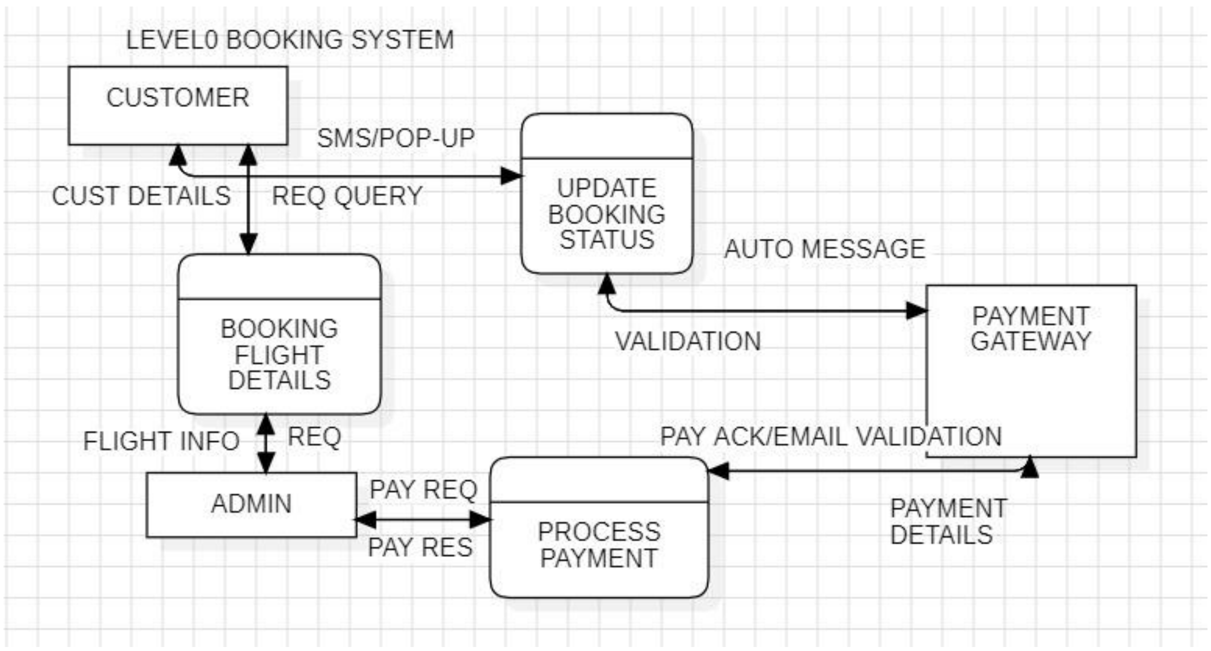
```
+-------------------+      +-------------------+
|     Booking       |      |   BookingSystem   |
+-------------------+      +-------------------+
| - bookingDetails  |      | - reservation()   |
| - price           |      | - updatePrice()   |
| - confirmation    |      | - generateBookingConfirmation()|
+-------------------+      +-------------------+
| + reserveSeat()   |      | + generatePaymentConfirmation()|
| + updatePrice()   |    | + sendNotification()|
| + generateBookingConfirmation()|    |
| + sendNotification()| | |
+-------------------+      +-------------------+
```



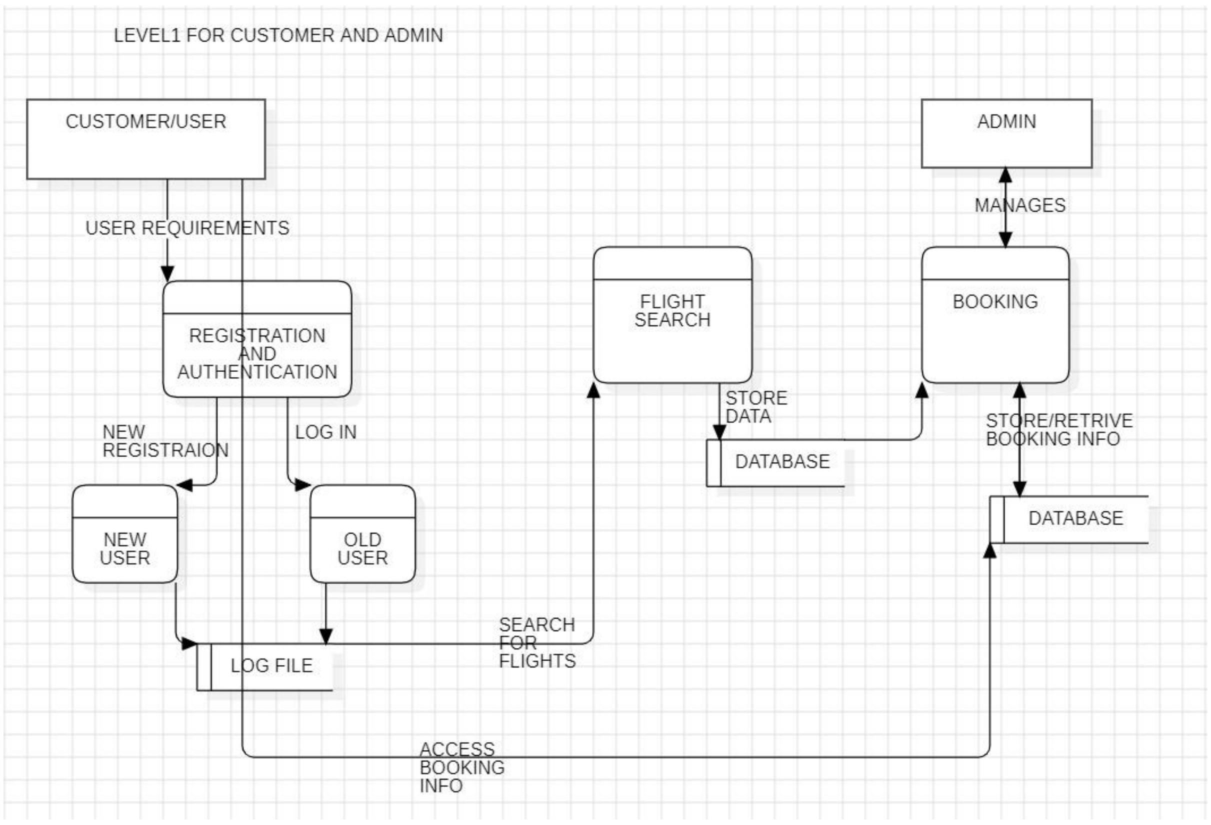
```
------------------
|     Methods     |
------------------
| + Netbanking()  |
| + DebitCard()   |
| + CreditCard()  |
| + UPI()         |
| + Confirmation()|
| + Status()      |
| + Authentication() |
------------------

---------------------
|     Netbanking     |
---------------------
| - userId          |
| - customerId       |
| - password        |
| + processPayment() |
---------------------

------------------
|   DebitCard     |
------------------
| - cardNumber    |
| - cvv           |
| - expiryDate    |
| + processPayment() |
------------------

------------------
|   CreditCard   |
------------------
| - cardNumber  |
| - cvv         |
| - expiryDate  |
| + processPayment() |
------------------
```

```
------------------
|     UPI       |
------------------
| - upiPin       |
| - upiId        |
| + processPayment() |
------------------

----------------------
|   Confirmation     |
----------------------
|                   |
| + generateReceipt()|
| + sendEmail()      |
----------------------

---------------
|     Status    |
---------------
| - transactionId |
| + checkStatus() |
| + updateStatus() |
------------------

--------------------------
|   Authentication      |
--------------------------
| - transactionId       |
| + authenticateUser()  |
--------------------------
```

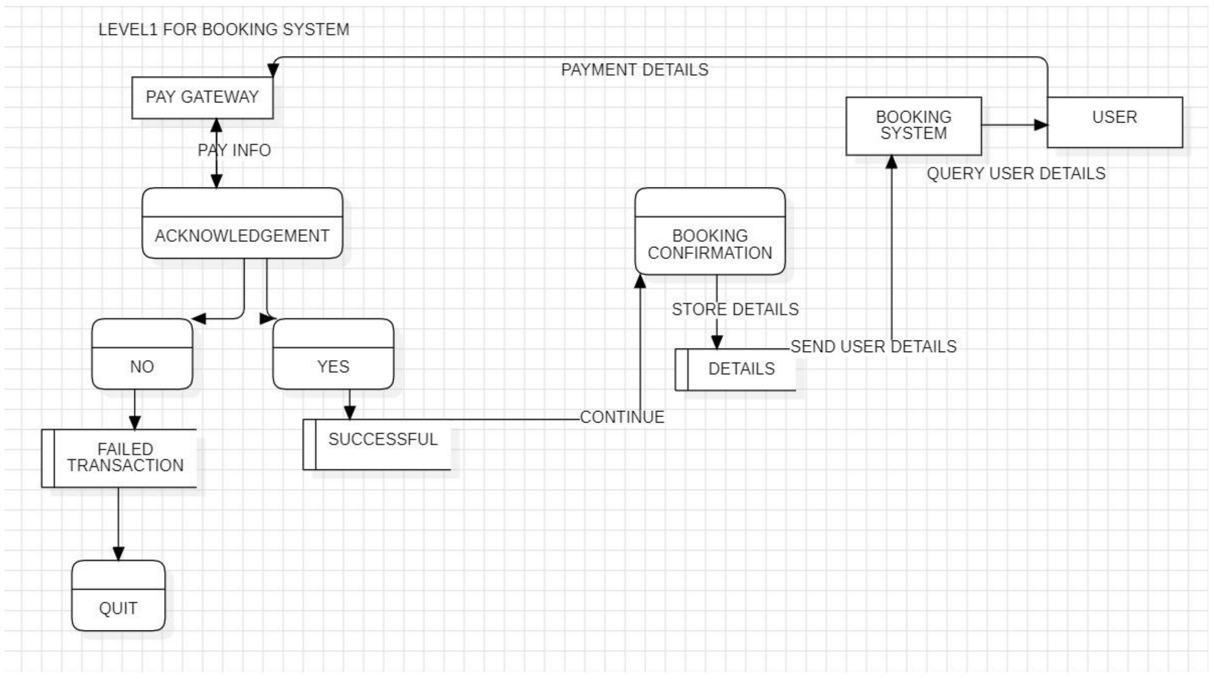
DFD'S

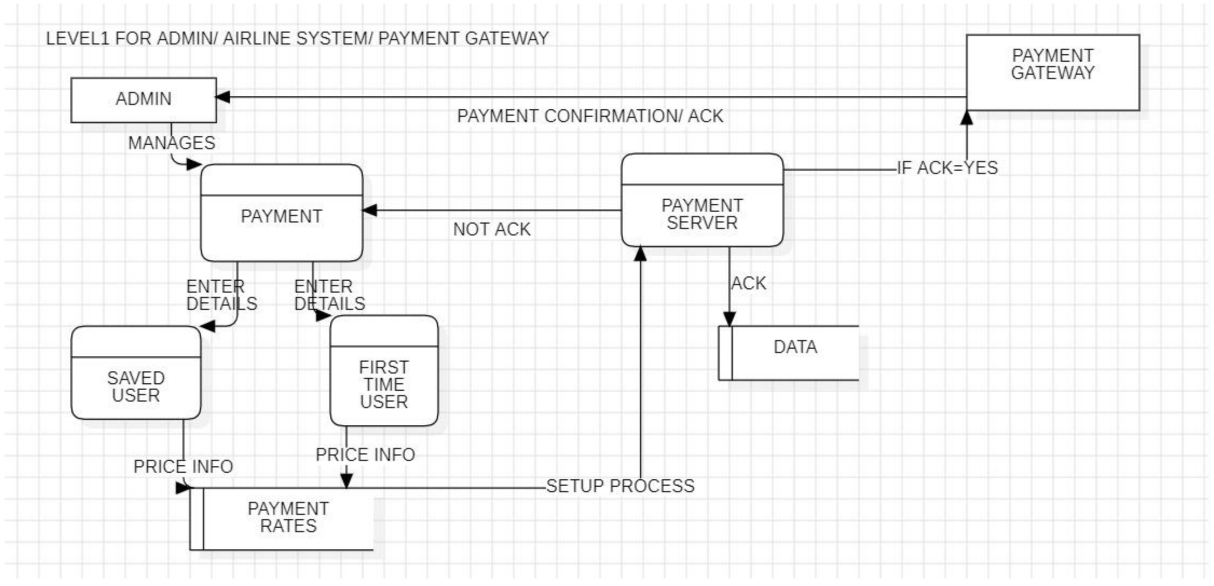


LEVEL0 BOOKING SYSTEM
CUSTOMER
SMS/POP-UP
CUST DETAILS
REQ QUERY
UPDATE BOOKING STATUS
AUTO MESSAGE
BOOKING FLIGHT DETAILS
VALIDATION
PAYMENT GATEWAY
FLIGHT INFO
REQ
PAY ACK/EMAIL VALIDATION
ADMIN
PAY REQ
PAY RES
PROCESS PAYMENT
PAYMENT DETAILS



LEVEL1 FOR CUSTOMER AND ADMIN
CUSTOMER/USER
ADMIN
USER REQUIREMENTS
MANAGES
REGISTRATION AND AUTHENTICATION
FLIGHT SEARCH
BOOKING
STORE DATA
STORE/RETRIVE BOOKING INFO
NEW REGISTRAION
LOG IN
DATABASE
DATABASE
NEW USER
OLD USER
SEARCH FOR FLIGHTS
LOG FILE
ACCESS BOOKING INFO





ARCHITECTURAL STYLE INTEGRATION

Model-View-Controller (MVC):

1. Model: Represents the application's data and business logic. In this context, the model can represent the flight data, user profiles, payment processing, and other business entities.

2. View: Represents the user interface and the presentation layer. Views display data to the users and receive user inputs, such as flight search criteria or payment details.

3. Controller: Acts as an intermediary between the Model and View. Controllers receive user inputs from the View, process them, and interact with the Model to update data and business logic. In an SOA-MVC hybrid, the Controller can communicate with various services to fulfill user requests.

Justification

1. Scalability:In an MVC architecture, the Controller can delegate tasks to different services within the SOA. This separation allows for the independent scaling of services, such as flight search, booking, or payment processing, based on their specific resource demands.

2. Maintainability: The Model represents the application's core logic, which can be structured as a set of services in the SOA. This separation of concerns makes it easier to maintain and update specific functionalities without affecting the entire system.

3. Flexibility: You can introduce new services or replace existing ones without disrupting the user interface (View) or the application's core logic (Model). This flexibility enables the addition of new features or integrations with external systems seamlessly.

4. Performance: Critical operations can be optimized within the individual services, ensuring that, for example, flight availability checks or payment processing are fine-tuned for performance without affecting the overall user interface.

5. Interoperability: The Controller in the MVC pattern can communicate with various services in the SOA to handle tasks such as interfacing with airline databases or payment gateways, leveraging SOA's interoperability features.

6. Fault Isolation: In the SOA-MVC hybrid, if a service fails, the Controller can handle the failure gracefully and possibly provide alternative services or functionality to the user, ensuring the application remains available.

7. Parallel Development: Development teams can work on different services within the SOA, including the Model's services, simultaneously, speeding up the development process.