**Documentation: Group 7**

Project Name *: MakeMyTrip*

**Milestones:**

1. Offering Data as a Service

Description

The milestone starts with the server layer designing where we will be designing and implementing the DAO layers which are PassengerDao, FlightsDao and BookingsDao. All the Dao Classes will be deliverable and at the end we will be able to push records into the NOSQL database where we will be using Mongo Db. In the PassengerDao class we will first design a model for it where we will include all the necessary fields having one primary key emailId which will be generated by the passenger once he signs up. This model will gather all the necessary information from the passenger who intends to book a flight ticket. The PassengerDao class will have all the essential methods in it required to perform any CRUD operation in the business layer. We will have various necessary methods to extract data and push the data to the database. This will give the opportunity to the passenger to create his account by giving his details, delete his account, update his account and view his details. Secondly In FlightDao we will design it in a way that when the passenger hits the search button after entering all the basic requirements to book a flight, he/she gets all the flights details on that route along with their cost and time. This will enable the passenger to view better options in terms of cost and time in order to book the most suitable flight. Information like flight name, date of arrival, date of departure, source, destination, flight capacity, travel charges etc will be displayed to the passenger in order to compare the most convenient flight for the journey. We will also generate an auto-generated unique sequence for this flightDao class which will be the flightId to distinguish between the flights. Lastly in the BookingsDao, we will have all the booking details of the passengers who have successfully made the payment and booked a ticket. This collection will have a primary key which will be PNR of the ticket. This will contain all the details about the journey including the date of booking, date of arrival, date of departure, passenger name, passenger’s email-id, source, destination and cost of the travel.

This DAO class will have all the necessary methods to interact with the database and fetch all the details related to the booking whenever the passenger wants to view it. It will also enable the passenger to delete his/her already booked ticket.

The above mentioned Dao classes will form the data service layer of our application and will be deliverable solely in the vertical architecture.

Stories

1. Creating model classes for Passenger, Flights and Bookings.
2. Learning and establishing a connection with Mongo Db.
3. Creating PassengerController class:-

i)Create addPassenger() method which will take passenger object as an argument and return type is String, which will add the passenger details in collection.

ii)Create getAllPassenger() method of the return type List<Passenger> which will return the entire list of all the Passenger.

iii)Create getPassenger() method which will take emailid as an argument and it will return Passenger details based on that id.

iv)Create updatePassenger() method which will take emailid as an arguments and it will update details of user based on their requirement.

v) Create deletePassenger() method which will take emailid as an argumet and return type is Boolean through which we can eliminate the record of that passenger from the database.

1. Creating FlightController class:-

i)Create create() method which will take flight object as an argument and it will return one String message. This method will add flight details in list of flight.

Ii)Create FlightsList() method which has no argument but it will return whole flight list whatever FlightList conatins.

Iii)Create getFlight() method which will take flightId as an argument and it will return Flight details according to flightId.

Iv)Create updteFlight() method which will take flightId as an argument and it will update flight detail as per the admin requirement.

v)Create checkSource() method, it will take source as an argument and it will check that given city name is valid airport name or not, according to the result it will return either true or false.

vi)Create checkDestination() method this method will take destination as an argument and then it will check that given city is valid airport name or not.

vii)Create deleteFlight() method, this will return one string message that if flightid will match with flight list then it will delete that flight from list then it will return message else it will return unable to delete flight with id message.

Vii)Create deleteAllFlight() method it will delete all flight details and return message.

1. Creating BookingController() class:-

i)Create addBookings() method whose return type is String, which will return Booking done successfullymessage after adding bookings details of passenger in list.

ii)Create getAllBookings() this method will return list<Booking>. It will return all the details from BookingList for admin use only.

iii)Create getBookingByPNR(), this method will take PNR as an argument and it wil return Booking history according to PNR number to the passenger.

iv)Create deleteBookings(), this method will take PNR as an argument and it will return Deleted Successfully message after successful deletion else it will return unable deleted.

1. This project also contains Swagger2 just for the documentation of URL. We implements Swagger to check every URL on single page and together. In swagger we added some information about the method return type.

7. There are some method which need security so that except admin no one can access like updteFlightDetails,allPassengerDetails etc, so we implemets spring security only for authority purpose.

8. We checked each and every method manually using URL but to check it automatically we integrate Mockito Junit testing just to verify that in testing tools.