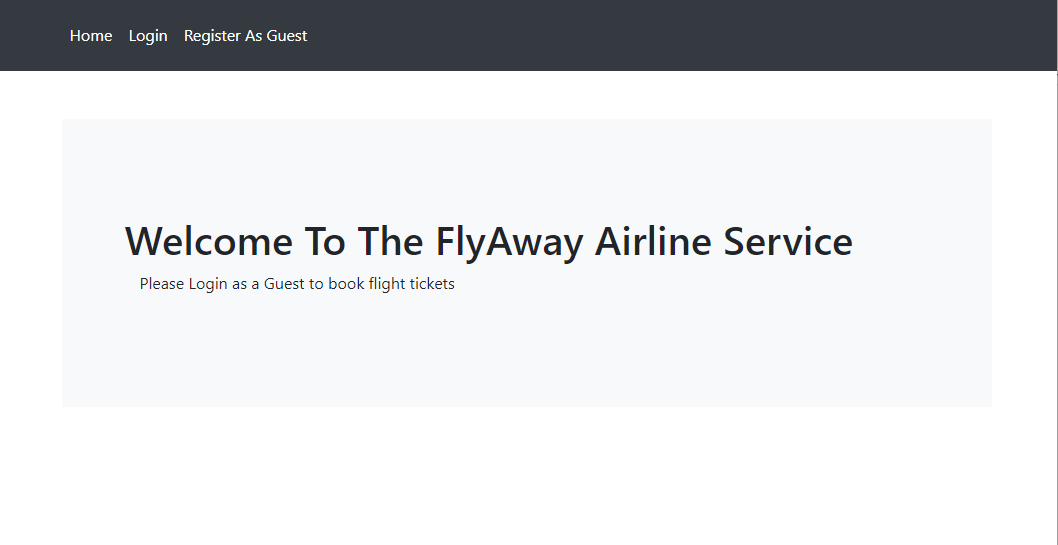
**FlyAway AirLine Ticket Booking Web Portal**

**Index:**

**Git URL:** **https://github.com/RajaRajeswari12/Phase2FlyAway.git**

1. **Jsp Pages:**
   1. HomePage.jsp
   2. AdminLogin.jsp
   3. RegisterGuest.jsp
   4. MainMenu.jsp
   5. UserMenu.jsp
   6. AdminMenu.jsp
   7. SearchForm.jsp
   8. SearchResult.jsp
   9. BookTicket.jsp
   10. PassengerDetails.jsp
   11. PaymentSite.jsp
   12. Ticket.jsp
   13. BookingFailed.jsp
   14. getBookedTicket.jsp
   15. RegisterAdminUser.jsp
   16. RegisterFlight.jsp
   17. FlightList.jsp
   18. EditFlightDetail.jsp
   19. PopulateFlightDtl.jsp
   20. ListFlightTicketsAvailability.jsp
   21. setBeanProperty.jsp
2. **Servlets [Controller]**
   1. com/controller/FlightDetailServlet.java
   2. com/controller/FlightScheduleServlet.java
   3. com/controller/FlightTicketSearchServlet.java
   4. /com/controller/UserRegisterServlet.java
3. **DataAccessObject [Dao]**
   1. com/dao/FlightDetailDao.java
   2. com/dao/FlightScheduleDao.java
   3. com/dao/FlightTicketSearchDao.java
   4. com/dao/UserDao.java
4. **Models[Hibernate]**
   1. com/model/Airline.java
   2. com/model/FlightAvailabilityByDate.java
   3. com/model/FlightDetail.java
   4. com/model/FlightRunningDays.java
   5. com/model/FlightTicket.java
   6. com/model/PassengersDetail.java
   7. com/model/SearchFlightDetailPojo.java
   8. com/model/TripSourceDestination.java
   9. com/model/User.java
5. **StoredProcedure**
6. **Database Table Creation Queries**
7. Jsp Pages
8. **HomePage.jsp**

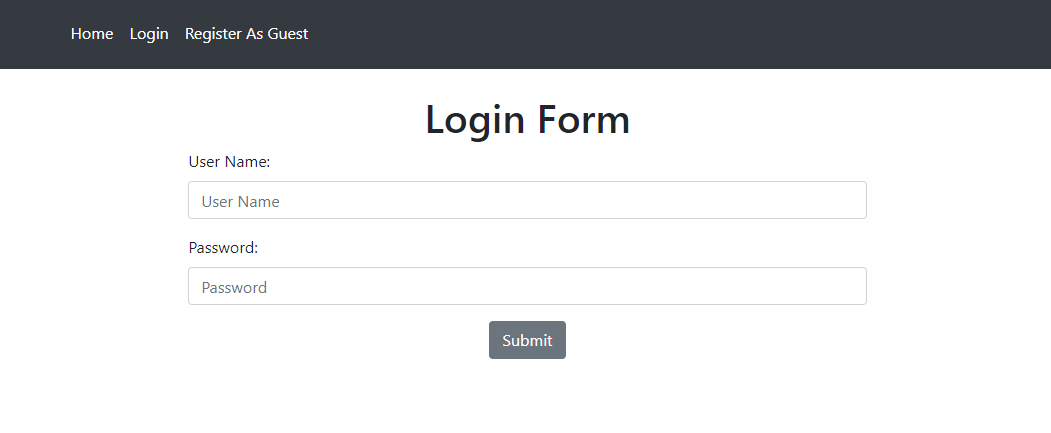
Home Page display the Welcome Note and has link to Login either as a Guest or Admin. It has a link to register a new user.



1. **AdminLogin.jsp**

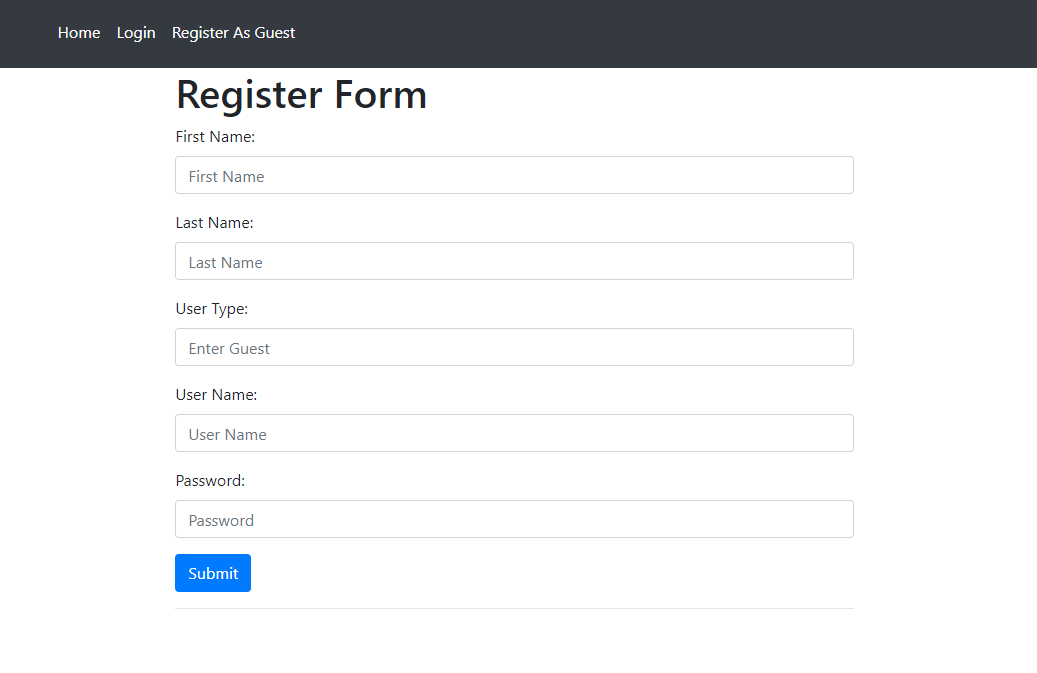
Login Page is used by both the User and Admin to Login into FlyAway portal.

If the User logs in with his/her credentials then it is navigated to the ticket search page. If the Admin logins with his/her credentials then it is navigated to the Admin page where the admin can see and add the flight details.



1. **RegisterGuest.jsp**

RegisterGuest jsp page is used to register new user as guest. Only a registered user can search for the available flights and book a flight ticket.



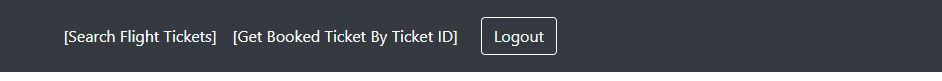
1. **MainMenu.jsp**

MainMenu jsp contains the menu list link to be displayed before Login or Register as a New User.



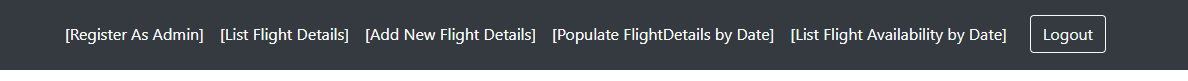
1. **UserMenu.jsp**

UserMenu jsp contains the menu list link to be displayed after the User login as a Guest.



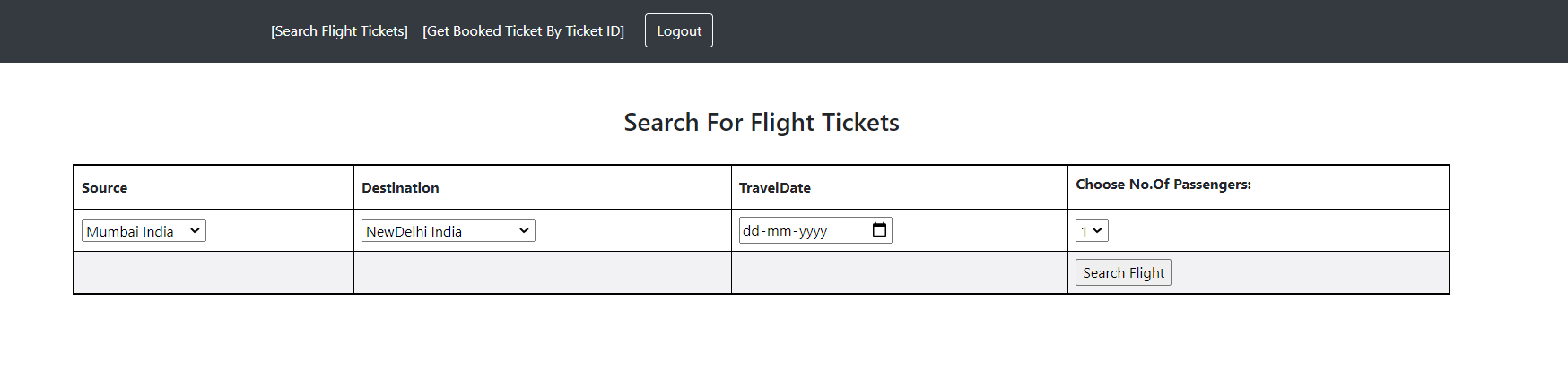
1. **AdminMenu.jsp**

AdminMenu jsp contains the menu list link to be displayed after the User login as a Admin.



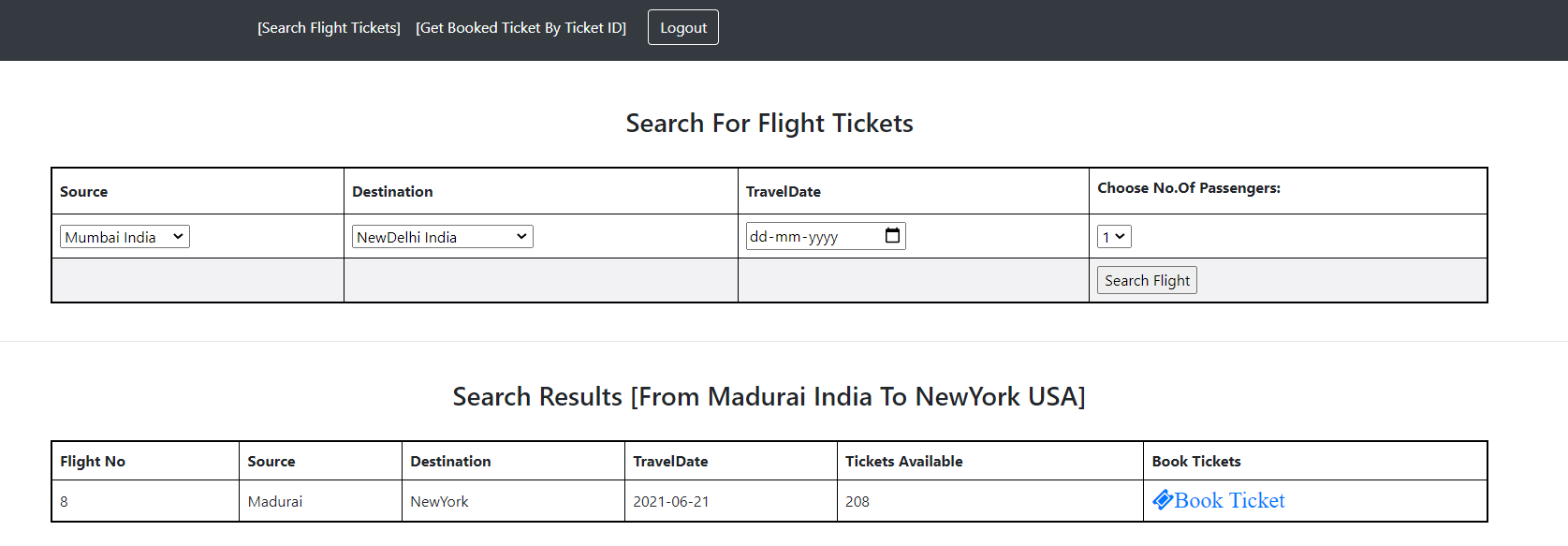
1. **SearchForm.jsp**

SearchForm is available to the User logging in as a **Guest.** Display the form with the Source city drop down, Destination city drop down,Travel Date and No Of Passengers[Max of 6 tickets can be booked]. The user can enter the details of his/her interest and search for the available flight tickets.



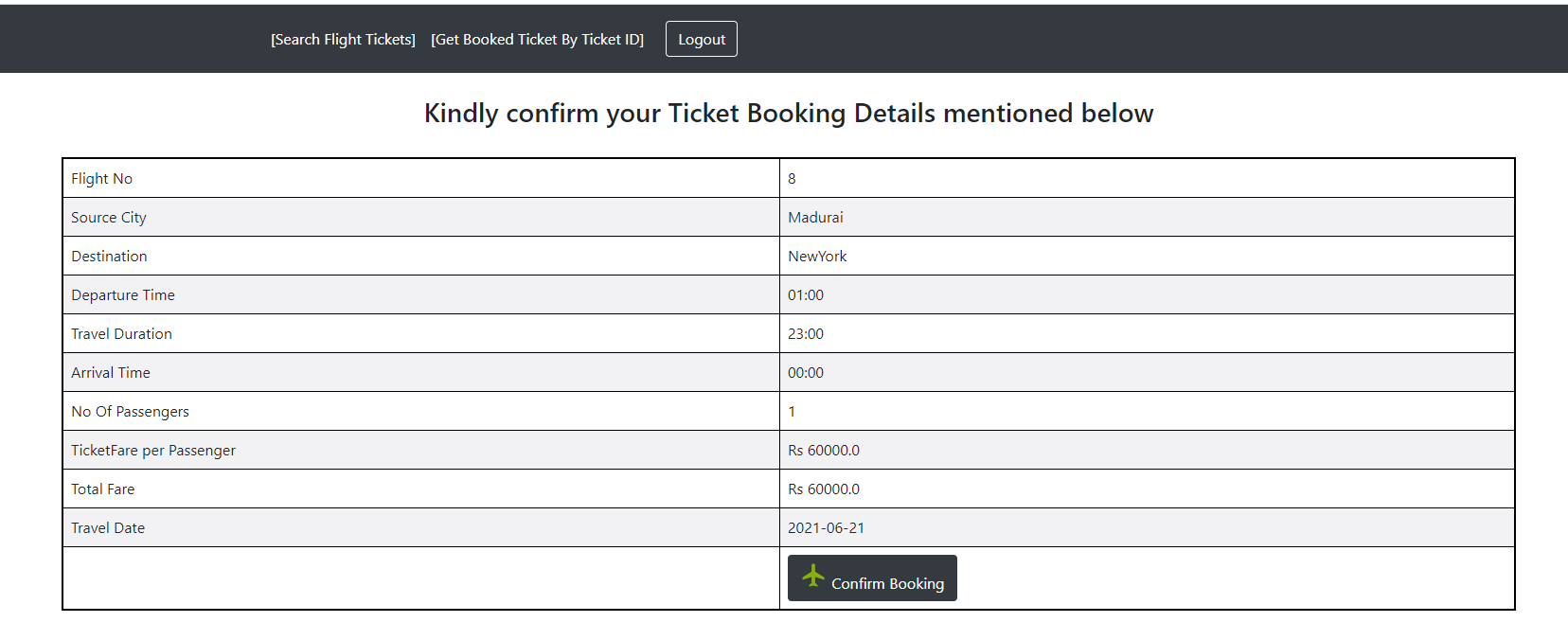
1. **SearchResult.jsp**

SearchResult is available to the User logging in as a **Guest.** SearchResult jsp page displays the available flights depending upon the details entered by the user.



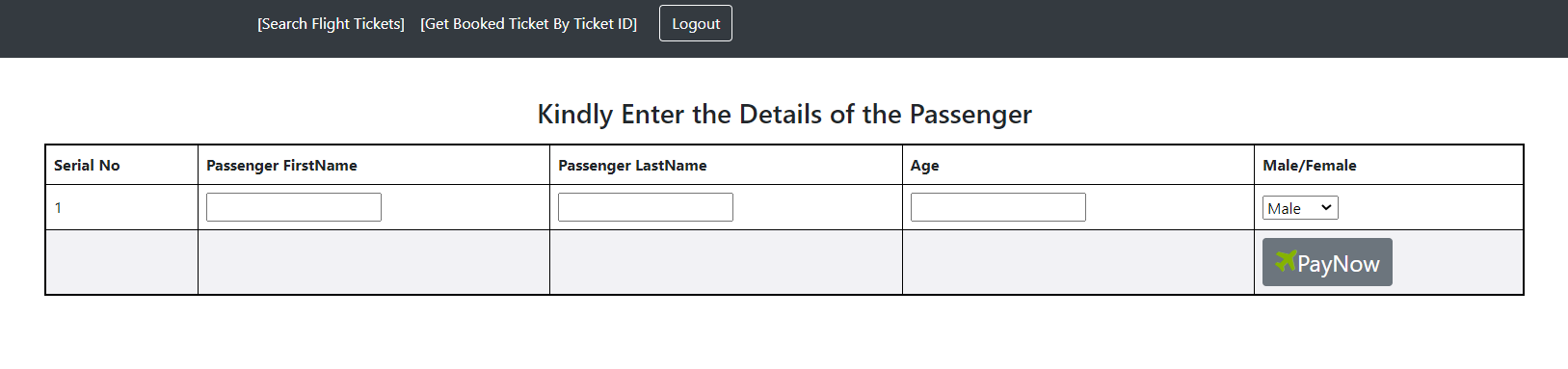
1. **BookTicket.jsp**

BookTicket is available to the User logging in as a **Guest.** BookTicket jsp is used to display the details entered by the User for confirmation.



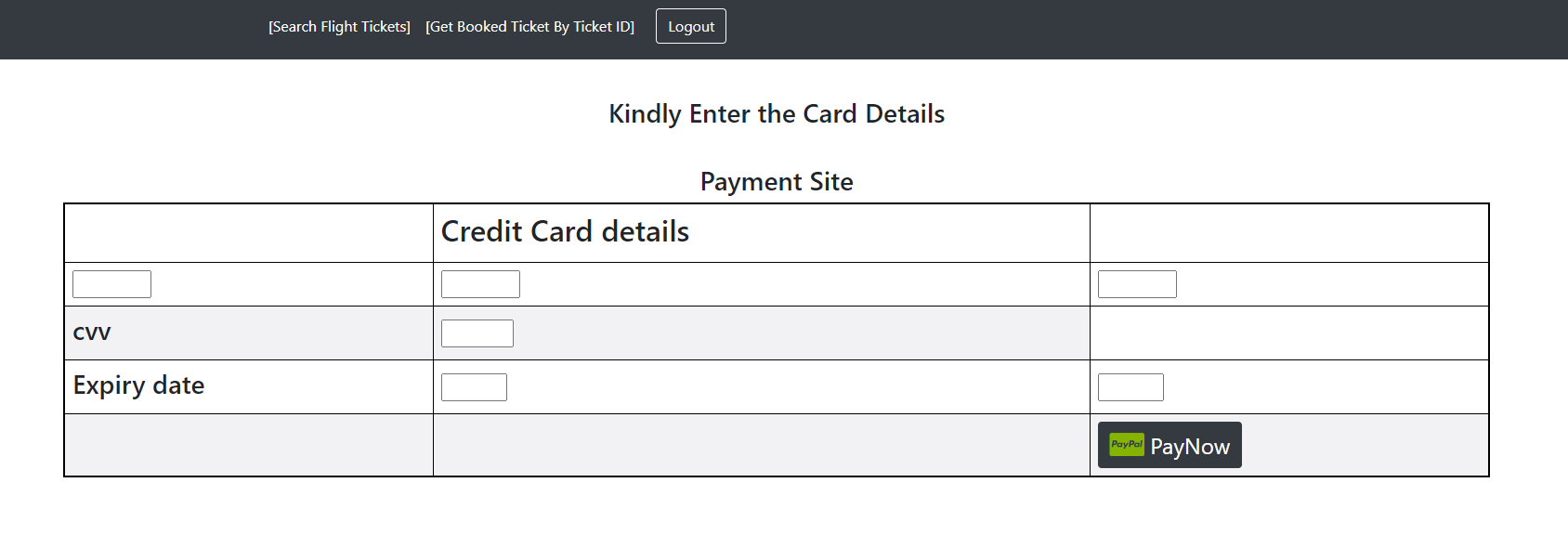
1. **PassengerDetails.jsp**

PassengerDetails is available to the User logging in as a **Guest.** PassengerDetails jsp is used to get the details of the passengers.

* + 1. 

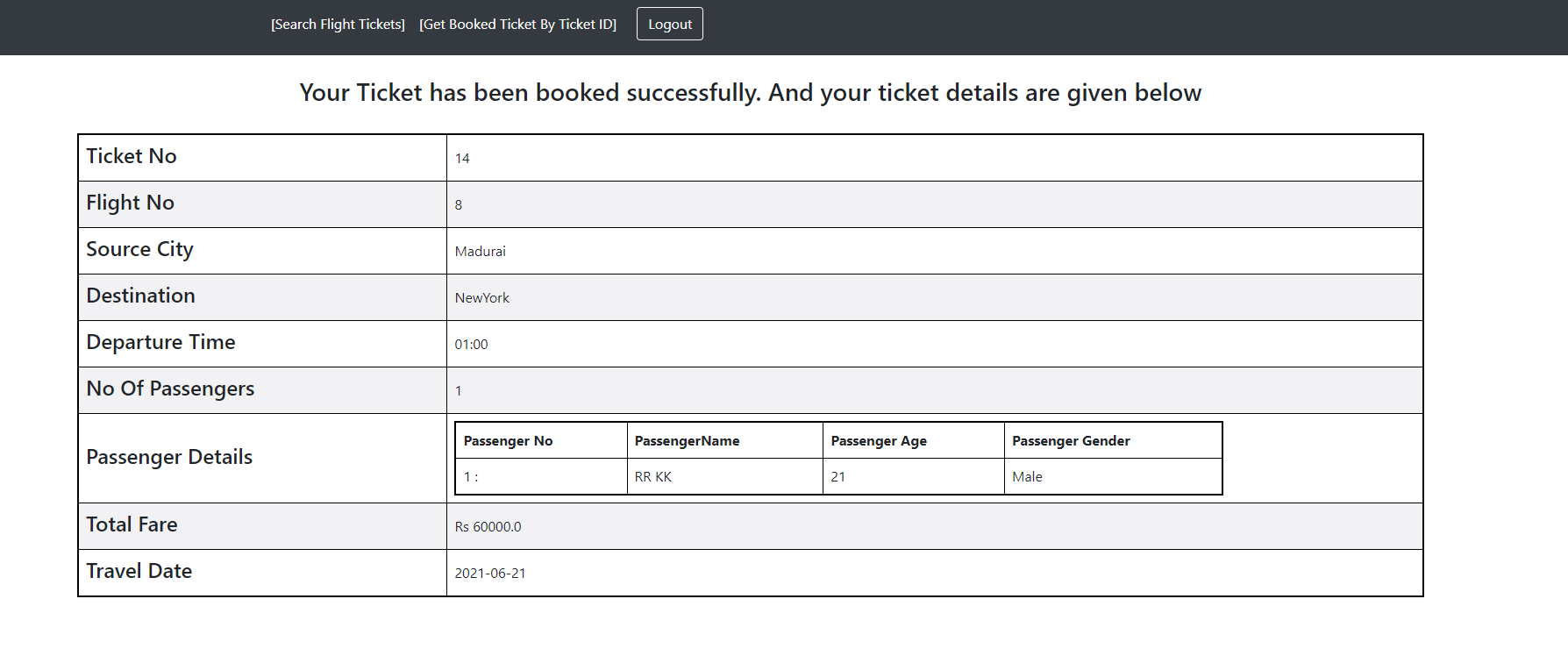
1. **PaymentSite.jsp**

PassengerDetails is available to the User logging in as a **Guest.** Payment jsp is a dummy payment page which is designed to get the credit card details of the User to book tickets.



1. **Ticket.jsp**

Ticket is available to the User logging in as a **Guest.** Ticket jsp is used to display the successfully booked flight details.

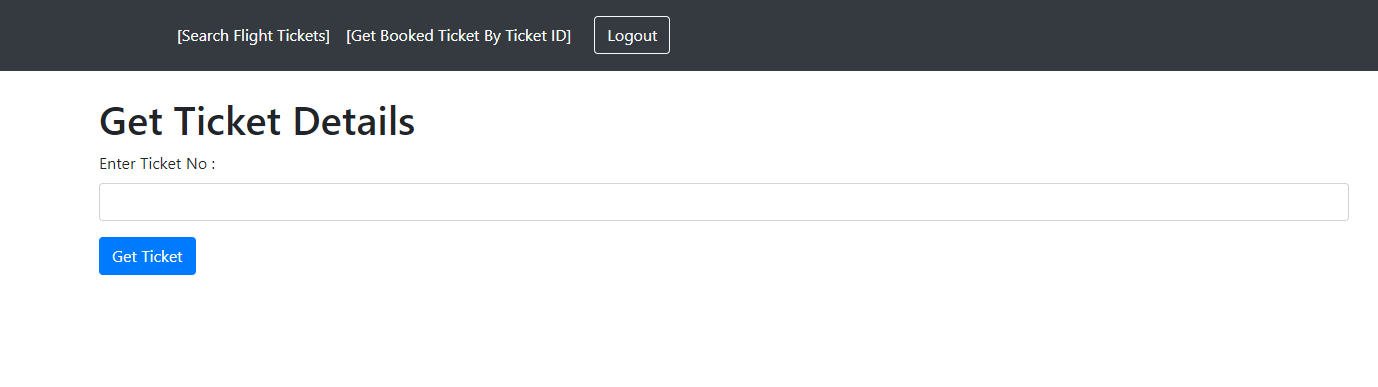


1. **BookingFailed.jsp**

BookingFailed is available to the User logging in as a **Guest.** BookingFailed jsp display unsuccessful flight ticket booking.

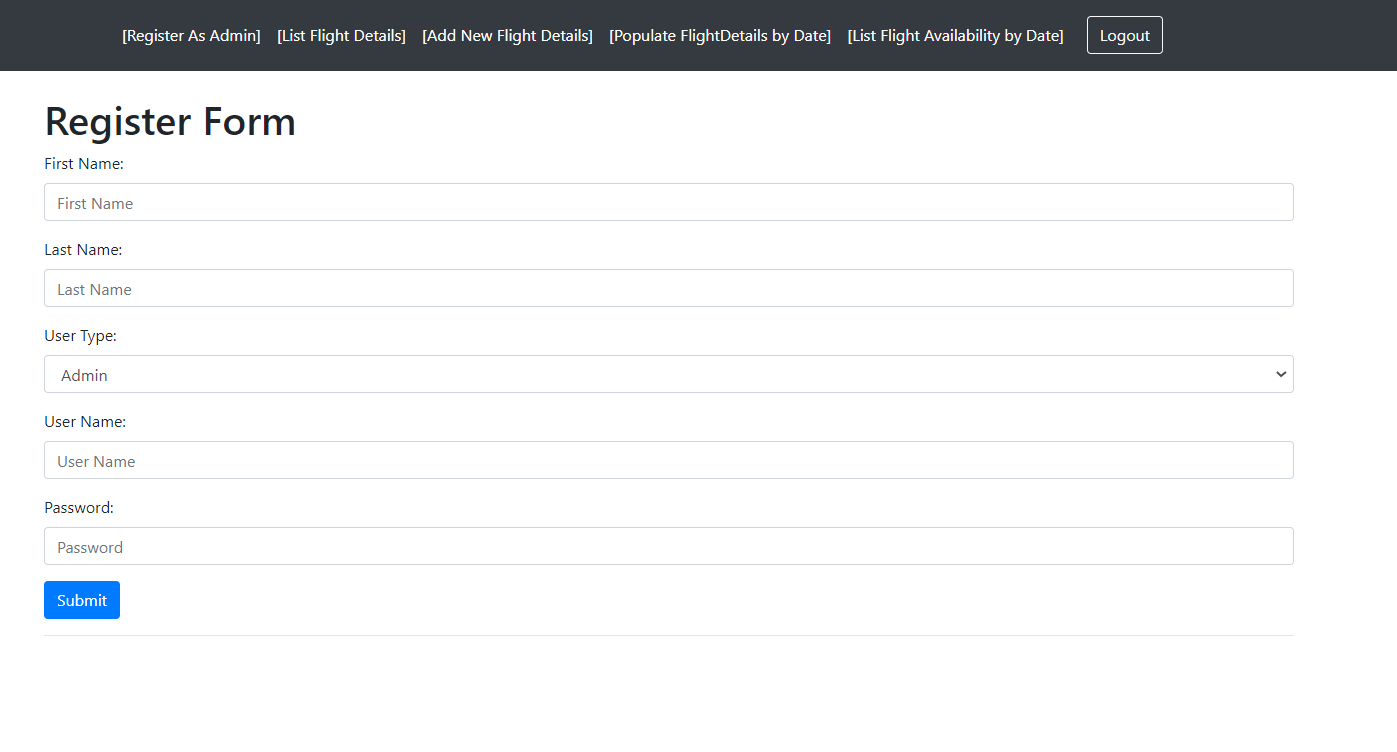
1. **getBookedTicket.jsp**

getBookedTicket is available to the User logging in as a **Guest.** getBookedTicket jsp page gets the ticket number from the user and displays the ticket details.



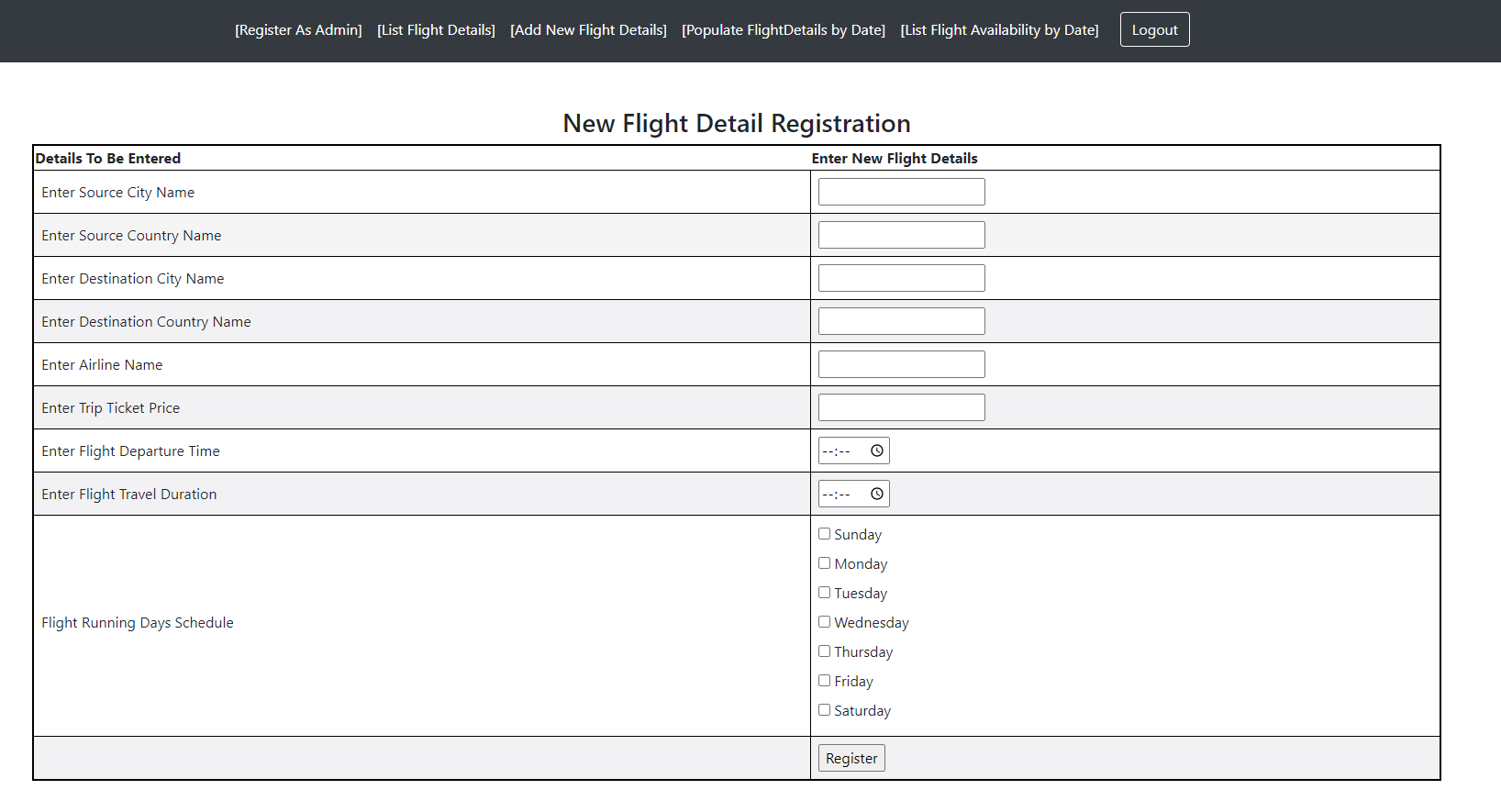
1. **RegisterAdminUser.jsp**

RegisterAdminUser jsp page is used to register new user as Admin. Only a registered Admin can check and update flight details. RegisterAdminUser page can be accessed by the admin only. One Admin user has to be registered through backend, then with the help of that admin login credentials we can register new admin login credentials.



1. **RegisterFlight.jsp**

RegisterFlight is available to the User logging in as a **Admin.** RegisterFlight jsp is used to add new flight details to the backend.

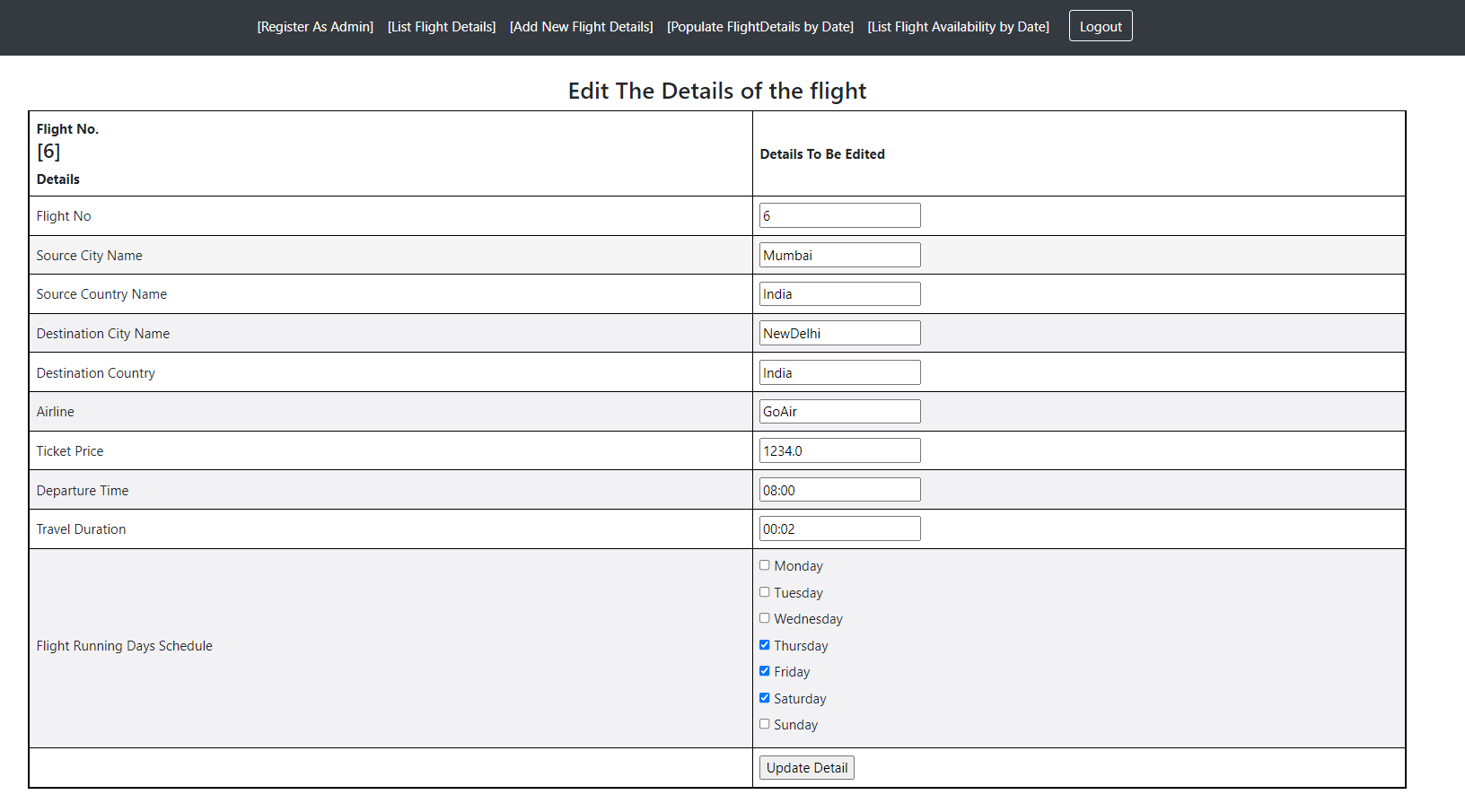


1. **FlightList.jsp**
   1. FlightList is available to the User logging in as a **Admin.** FlightList jsp is used to list all the available flight details from the backend. This page has an option to update the flight details by clicking the edit button, Delete the flight details by either clicking the delete link or by clicking deleteSelectedFlight button.



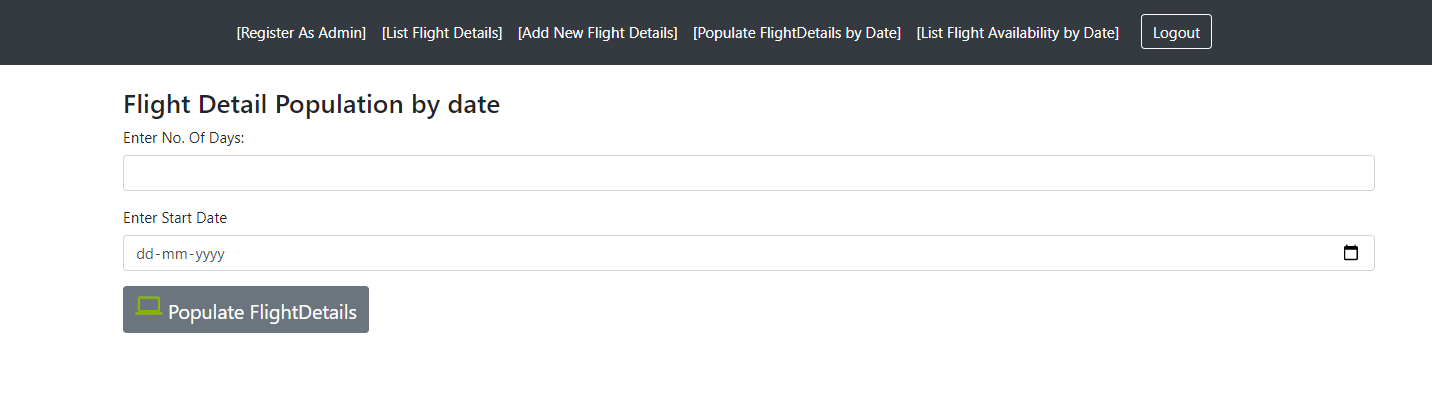
1. **EditFlightDetail.jsp**

EditFlightDetail is available to the User logging in as a **Admin.** EditFlightDetail jsp is used to update the selected flight details.



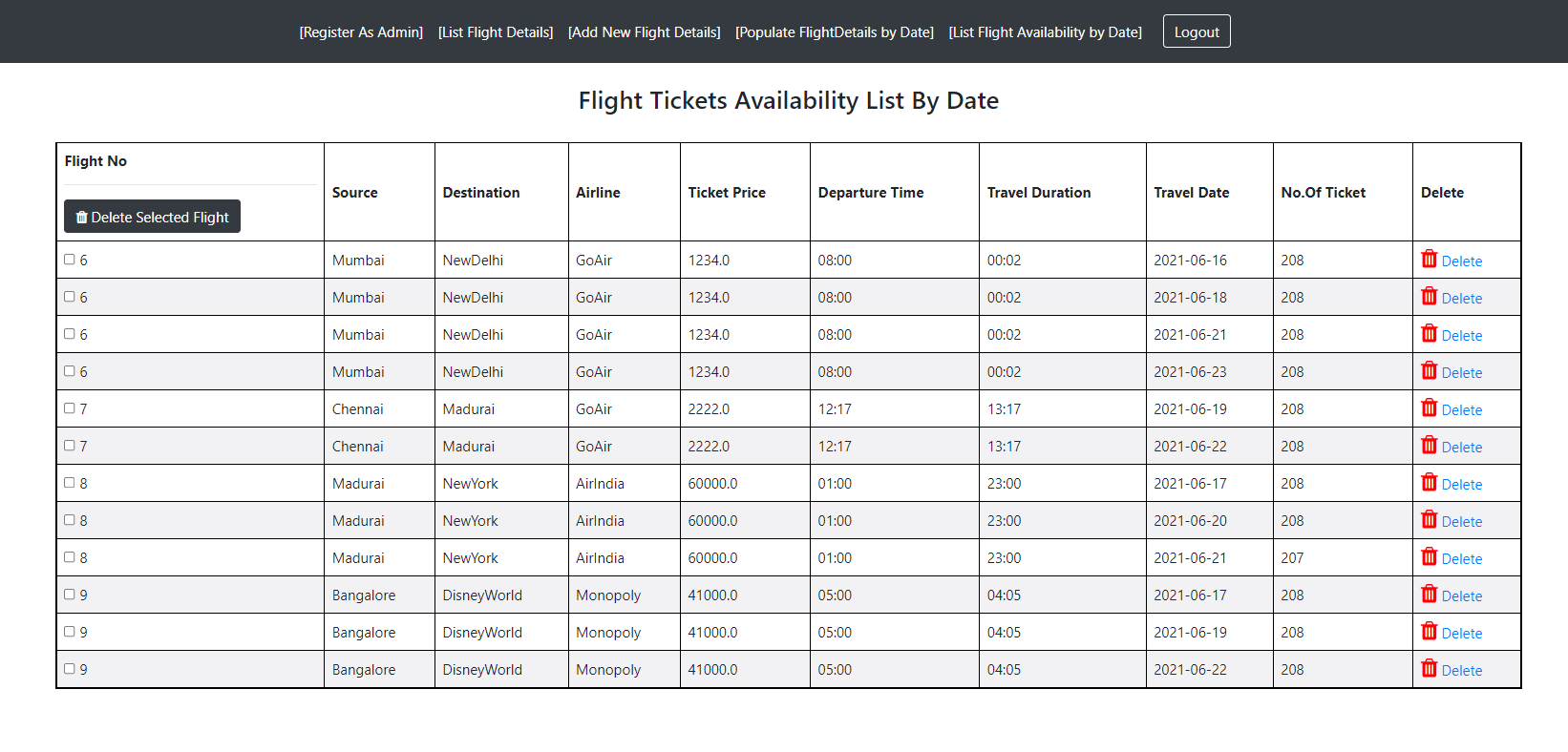
1. **PopulateFlightDtl.jsp**

PopulateFlightDtl is available to the User logging in as a **Admin.** PopulateFlightDtl jsp is used to populate the flight tickets for the upcoming dates. The Admin enters only the flight running days(Monday,Tuesday,…) . The flight details specific to the date won’t be available. In order to populate the flight tickets specific to the date based on the input updated by the Admin the stored procedure is called. Two details has to be entered in this page.They are DayCount[Upto How many days Flight Tickets are needed] and Date[From which date the flight tickets has to be generated].



1. **ListFlightTicketsAvailability.jsp**

ListFlightTicketsAvailability is available to the User logging in as a **Admin.** ListFlightTicketsAvailability jsp is used to list the flight tickets available by date. This page has the option to delete the flight details.



1. **setBeanProperty.jsp**

setBeanProperty jsp used to set the input given in the Add new Flight Detail page and editFlightDetail page to the POJOs.

1. **Servlets [Controller]**
   1. com/controller/FlightDetailServlet.java

FlightDetailServlet controller is used to process the data according to the request. The methods used in the controller are

* + 1. **addNewFlightDetail**

Receive the flightDetail through the session and insert it in the backend (Database). Finally after successful creation of the record redirect the request to the FlightList.jsp

* + 1. **deleteFlightByFlightId**

Retrieve the Id of the flight to be deleted from the request URL and delete the flight record in the Database.

* + 1. **updateFlightDetail**

Receive the updated flight Detail through the session and update it in the backend (Database). After successful updation of the record redirect the request to the FlightList.jsp

* + 1. **editFlightDetail**

Receive the edit request along with the details of the flight to be edited and redirect it to the EditFlightDetail.jsp with that flight details.

* + 1. **deleteListedFlight**

Retrieve the Id List of the flights record to be deleted from the request parameter and delete all the flights record in the Database.

* + 1. **getFlightDetailList**

Receive the listing request and connect to the Database to retrieve all the flight records and redirect to the FlightList.jsp

* 1. com/controller/FlightScheduleServlet.java
     1. **getFlightDetailsByTripDateList**

Receive the listing request and connect to the Database to retrieve all the flight details records available by Date and redirect to the ListFlightTicketsAvailability.jsp

* + 1. **populateFlightDetail**

Receives the request along with two inputs and directs the stored procedure to populate the data for the specified days from the given date.

* + 1. **deleteListedFlightByTripDate**

Retrieve the Id of the flightTrip to be deleted from the request URL and delete the flightTrip in the Database.

* + 1. **deleteListedFlightByTripDateById**

Retrieve the Id List of the flightTrip records to be deleted from the request parameter and delete all the flightTrip records in the Database.

* 1. com/controller/FlightTicketSearchServlet.java
     1. **searchForm**

Contact DAO and get the source, destination list , set it in the session attribute and redirect to the SearchForm.jsp

* + 1. **searchFlight**

Receives the request with the details given by the user and redirect it to the corresponding DAO.

* + 1. **bookFlightTicket**

Retrieves the detail of the flight in which the user is going to book the flight ticket and redirect it to the PassengersDetail.jsp

* + 1. **payAndBookTicket**

Retrieves the detail required for booking the flight ticket and redirect to the Dao and then direct to the Ticket.jsp page.

* + 1. **getBookedTicketDetail**

Retrieve the ticket id from the request, redirects to the Dao and with the records returned by the dao , it redirect to the Ticket.jsp page.

* 1. /com/controller/UserRegisterServlet.java
     1. validateUser

Receive username and password from the user, redirects to the DAO and with result returned by the dao it redirects to the respective pages and set the session attributes.

* + 1. registerUser

Receives the details through the request and redirects to the DAO to insert new user record.

* + 1. logoutUser

Receives the request and removes the session attributes and set session invalidate.

1. **DataAccessObject [Dao]**
   1. **com/dao/FlightDetailDao.java**
      1. **registerFlightDetail**

Insert new flightDetails into the table. Before inserting into FlightDetail table checks whether the airline and Source,Destination already exist in their respective table with the help of the checkFlightDetail() method.

* + 1. **checkFlightDetail**

checks whether the airline and Source,Destination already exist in their respective table.If it exist then update the flightDetail with the respective record details.

* + 1. **deleteFlightDetails**

Deletes the record first from FlightAvailabilityByDate, then FlightDetail and finally from FlightRunningDays.

* + 1. **getFlightDetail**

Retrieves all the records available in the FLightDetail Table along with the respective records from airline and TripSourceDestination Table

* + 1. **getFlightDetailById**

Retrieves the record in the FLightDetail Table along with the respective records from airline and TripSourceDestination Table for the given flightId.

* + 1. **updateFlightDetail**

Update the record in the FLightDetail Table along with the respective records in airline and TripSourceDestination Table .

* 1. **com/dao/FlightScheduleDao.java**
     1. **deleteFlightDetailsByTripIds**

Deletes the record from the FlightAvailabilityByDate Table for the given FlightScheduleIds.

* + 1. **populateFlightDetail**

Populate the data by calling the stored procedure with the given data. [specified No. Of days, given start date]

* + 1. **getFlightAvailabilityByDate**

Retrieves all the records available in the FlightAvailabilityByDate Table.

* 1. **com/dao/FlightTicketSearchDao.java**
     1. **getSourceList**

Retrieves all the distinct sourceCityName available in the TripSourceDestination Table.

* + 1. **getDestinationList**

Retrieves all the distinct destinationCityName available in the TripSourceDestination Table.

* + 1. **getAvailableFlight**

Retrieves all the flights available on the given date for the given source destination.

* + 1. **bookTicket**

Deduct the no.Of seats on that flight trip[FlightAvailabilityByDate] and insert the records in the FlightTicket Table and corresponding passengers detail in the PassengersDetail

* + 1. **deductFlightSeat**

Simple method to Deduct the no. of seats on the FlightAvailabilityByDate table.

* + 1. **getFlightTicket**

Retrieves the ticket details from the FlightTicket table for the given ticket Number.

* 1. **com/dao/UserDao.java**
     1. **saveUser**

This method saves the new user details in the user table.

* + 1. **validate**

This method checks whether the given username and password exists in the table.And returns their respective user type.

1. **Models[Hibernate]**
   1. com/model/Airline.java
   2. com/model/FlightAvailabilityByDate.java
   3. com/model/FlightDetail.java
   4. com/model/FlightRunningDays.java
   5. com/model/FlightTicket.java
   6. com/model/PassengersDetail.java
   7. com/model/SearchFlightDetailPojo.java
   8. com/model/TripSourceDestination.java
   9. com/model/User.java
2. Stored Procedure:[ `populateFlightDetailByDate]

This Stored procedure is created to populate the flight details and tickets availability for the specified days. This procedure reads the given input **[For how many number of days the flight detail has to be generated and from which date]** and data from the table **flightRunningDays** and **flightdetail**, after that generate data and insert into the table FLIGHTAVAILABILITYBYDATE. This procedure has to be run by the Admin.

CREATE DEFINER=`root`@`localhost` PROCEDURE `populateFlightDetailByDate`(in noOfDays int,in givenFromDate date)

BEGIN

DECLARE FlightRunningId INT DEFAULT NULL;

DECLARE Monday tinyint DEFAULT NULL;

DECLARE Tuesday tinyint DEFAULT NULL;

DECLARE Wednesday tinyint DEFAULT NULL;

DECLARE Thursday tinyint DEFAULT NULL;

DECLARE Friday tinyint DEFAULT NULL;

DECLARE Saturday tinyint DEFAULT NULL;

DECLARE Sunday tinyint DEFAULT NULL;

DECLARE FlightId INT DEFAULT NULL;

DECLARE done TINYINT DEFAULT FALSE;

Declare dayCount int;

Declare fromDateToPopulate date;

DECLARE COLCURSOR cursor for

SELECT fd.flightId,frd.\*

FROM flightrunningdays frd,flightDetail fd where fd.FlightScheduleId = frd.FlightScheduleId;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

set daycount = 1;

OPEN COLCURSOR;

my\_loop:

LOOP

FETCH NEXT FROM COLCURSOR INTO FlightId,FlightRunningId,Monday,Tuesday,Wednesday,Thursday,Friday,Saturday,Sunday;

select concat(FlightId,Monday,Tuesday,Wednesday,Thursday,Friday,Saturday,Sunday);

SET fromDateToPopulate = givenFromDate;

set dayCount = 0;

IF done THEN

LEAVE my\_loop;

end if;

while(dayCount <= noOfDays) do

if Monday = 1 and weekday(fromDateToPopulate) = 0 THEN

INSERT ignore INTO FLIGHTAVAILABILITYBYDATE(flightId,TravelDate,NoOfTickets) values(FlightId,fromDateToPopulate,208) ;

END IF;

if Tuesday = 1 and weekday(fromDateToPopulate) = 1 THEN

INSERT ignore INTO FLIGHTAVAILABILITYBYDATE(flightId,TravelDate,NoOfTickets) values(FlightId,fromDateToPopulate,208) ;

END IF;

if Wednesday = 1 and weekday(fromDateToPopulate) = 2 THEN

INSERT ignore INTO FLIGHTAVAILABILITYBYDATE(flightId,TravelDate,NoOfTickets) values(FlightId,fromDateToPopulate,208) ;

END IF;

if Thursday = 1 and weekday(fromDateToPopulate) = 3 THEN

INSERT ignore INTO FLIGHTAVAILABILITYBYDATE(flightId,TravelDate,NoOfTickets) values(FlightId,fromDateToPopulate,208) ;

END IF;

if Friday = 1 and weekday(fromDateToPopulate) = 4 THEN

INSERT ignore INTO FLIGHTAVAILABILITYBYDATE(flightId,TravelDate,NoOfTickets) values(FlightId,fromDateToPopulate,208) ;

END IF;

if Saturday = 1 and weekday(fromDateToPopulate) = 5 THEN

INSERT ignore INTO FLIGHTAVAILABILITYBYDATE(flightId,TravelDate,NoOfTickets) values(FlightId,fromDateToPopulate,208) ;

END IF;

if Sunday = 1 and weekday(fromDateToPopulate) = 6 THEN

INSERT ignore INTO FLIGHTAVAILABILITYBYDATE(flightId,TravelDate,NoOfTickets) values(FlightId,fromDateToPopulate,208) ;

END IF;

SET fromDateToPopulate = date\_add(fromDateToPopulate, interval 1 DAY);

set dayCount = dayCount+1;

end while;

END LOOP;

CLOSE COLCURSOR;

END

1. **Database Table Creation Query:**

create database AirlineTicket;

use AirlineTicket;

create table Trip\_Source\_Destination (

TripId int primary key not null auto\_increment,

sourceCityName varchar(45) not null,

sourceCountry varchar(45) not null,

DestCityName varchar(45) not null,

DestCountry varchar(45) not null,

constraint uk\_Src\_Dest\_pair unique(sourceCityName,DestCityName)

);

create Table Airline(

AirlineId int primary key not null auto\_increment,

AirlineName varchar(45),

constraint uk\_airline\_nm unique(AirlineName)

);

create table FlightRunningDays(

FlightScheduleId int primary key not null auto\_increment,

Monday boolean not null default 0,

Tuesday boolean not null default 0,

Wednesday boolean not null default 0,

Thursday boolean not null default 0,

Friday boolean not null default 0,

Saturday boolean not null default 0,

Sunday boolean not null default 0

);

create table FlightDetail(

FlightId int primary key not null auto\_increment,

TripId int not null,

AirlineId int not null,

TicketPrice double not null,

FlightScheduleId int not null,

departureTime time not null,

TravelDuration time not null,

foreign key (TripId) references Trip\_Source\_Destination(TripId),

foreign key (AirLineId) references AirLine(AirLineId),

foreign key (FlightScheduleId) references FlightRunningDays(FlightScheduleId)

);

create Table FlightAvailabilityByDate(

FlightTripId int primary key not null auto\_increment,

FlightId int not null,foreign key(FlightId) references FlightDetail(FlightId),

TravelDate date not null,

NoOfTickets int not null,

constraint uk\_duplicate unique(flightid,travelDate));