# Air Ticket Reservation Application

## PROGRAMMER MANUAL

**PRESENTED BY**

**VINEETH ARAVELLI (800837218)**

**DHILIP RAMESH (800835837)**

The various .jsp files and Servlets used for the Assignment 3 modules are displayed below:

**Registration.java (Servlet)**

Registration.java servlet takes the input parameters entered by the user in the Registration page. It takes the user details from the registration page and saves the data in the database.

**Login.java (Servlet)**

This servlet takes the input parameters from the Login.jsp and checks whether the username and password already exists. If it matches the user will see the search page.

**Logout.java (Servlet)**

The Logout.java servlet has the code for killing all the sessions. The session.invalidate is used for this purpose and it routes to the Login.java page where a new user or an existing user can login for a new session.

**FlightsSearch.jsp**

This page has the link to the booking history page and all the parameters required to search for a flight. The various text fields in this page are – Source, Destination, Date of Travel, No of seats and Class. A place holder value box for the user to select his class is also provided. Once the user fills all the required fields and hits the Search will take the user to the Search Results page using the FlightSearchServlet.java.

**FlightSearchServlet.java**

Upon a successful login the user is redirected to the flight search page named FlightsSearch.jsp. This servlet queries the database based on the user inputs. The inputs given by the User like: Source, Destination, Date of Travel, Class is stored in the string variables and queries the database and stores them and which in turn is sent to the FlightsSearch.jsp.

**FlightSearchResult.jsp:**

This page will display the results of the Search query given by the user in the previous page. It will display the various Flights with their Flight No., Flight Date, Departure Time, Arrival Time, No. of Stops, Class and Cost. Also there is a link to View and Book page.

**FlightSearchResultServlet.java:**

The FlightSearchResultServlet.java page gets the various Flights, their Flight Nos, Flight Date, Departure Time, Arrival Time, No. of Stops, Class and Cost. It sends it to the MysqlDatabaseConnection.java to retrieve the required fields to be displayed on the FlightSearchResult.jsp. It will direct the user to the ViewandBook.jsp page

**ViewandBook.jsp:**

This page will display the detailed information of the flight selected by the user with the Number of stops, Duration, Jet type and the Number of seats. Also links to select the displayed flight, Back button and Home button are provided.

**ViewandBookServlet.java:**

The flight parameters are passed to the ViewandBookServlet.java by the ViewandBook.jsp. It also verifies whether the user requested seats are

available and throws an error if it is not available and goes back to the FlightSearchResult.jsp. The total cost is displayed if the searched flight is available and is redirected to the TransactionPage.jsp.

**Transaction.jsp**

When you click select in the view and book it forwards to the transaction page which the user has all the information about the selected flight. The transaction page contains the text fields for user bank account information such as the A/C holder name, Routing number and the A/C number. Confirm transaction button and Cancel buttons are provided.

**Transactioncnfrm.java:**

The Transaction.jsp provides the bank details to the Transaction Confirmation Servlet. The transaction.java model is used to verify the bank account details and the balance of the user. On success the booking history is updated with the user bookings using the Bookings.java. It redirects the user to the Transactioncnfrm.jsp with the flight details and transaction status.

**Transactioncnfrm.jsp:**

On Success: The Transaction Confirmation page displays all the messages from the previous page along with Successful transaction message. It also displays a form for the user to enter the passenger information. A Print button is present so that the user can print the ticket showing all the flight and passenger information entered.
On Failure: When there is a failure a failure message is displayed saying that the “Transaction was not successful”. Along with the message the reason for the failure is also mentioned along with it (Incorrect details, insufficient details etc.)

**BookingHistory.java:**

BookingHistory uses the username to search all the previous bookings for that user. It fetches the user values from the Bookings model and then redirects the user to the BookingHistory.jsp with the results.

**BookingHistory.jsp:**

The Booking history page displays all the previous bookings done by the user and also a link to the Home page.

**User.java:**

The session begins from the servlet User.java, where the username is used as the session attribute. This is used to prevent the users from accessing the pages in the flight reservation system unless they have the valid user credentials (username and password). If the username is null, then the user is redirected to the login page. It has the connect method which has the connection string to the database, username and the password to the login.

**Flight.java:**

This session has the getters and setters methods for the flight variables. The Getflight method is use to get the flight objects. It gives the getflight results as array objects. It also has the FLightSearchQuery and the FLightSearchResult which connects to the database to retrieve the values to the required servlets that in turn sends the .jsp files.

**Transactions.java:**

This session is used to validate the bank account details of the user who is booking the flight ticket. If there is sufficient balance then the transaction will be successful else if there is insufficient balance in the user's account then it will post an error saying insufficient funds. On successful transaction the user will get redirected to the Transaction Confirmation page. Thus the new balance for the user account is update based on the transaction amount.

**Bookings.java:**

Bookings.java will read the bookings history for the user from the database. Once the transaction is completed the booking history is stored in the database.

**MysqlDatabaseConnection.java:**

It is the model part of the flight search reservation system. It holds all the methods to query the Database and return them to the required servlets that sends the .jsp files.

**AddToCart.java:**

The AddToCart.java servlet is used to add the flights for booking. It also has the feature to add the selected flight. It displays all flights that have been added by the user for further transaction.

**Display_cart.jsp:**

It is used to display all the flight details in the Cart. We have option to delete and checkout the flights. The delete option is used to delete the flight details from the cart. The checkout option will redirect the user to Transaction page.

**Transaction model** (Banking model)**:**

This file is used as a banking model. It is used to check whether the user account has enough balance. The booking amount should be lesser than the account balance. If there is enough account balance for the transaction to succeed it returns a success status. If there is no enough balance then there will be a failure status is generated. On success the booking amount is deducted from the account balance.

**UpdateHistory servlet:**

This servlet is used to update the booking history using the booking model by adding the user booking details.

**BankServlet:**

The ConfirmBooking.jsp will generate the user bank account details to the BankServlet using AJAX. The bank account standing is checked using the Bank model – Transaction model. We use the GSON library which will take java objects and gives the responses as JSON values.

**ConfirmBooking.jsp**

We have used the AJAX calls using the Jquery. On success status the data is refreshed and the latest confirmations values are generated. The JSON values are parsed and displayed dynamically on success status. On failure the user remains in the same page and an alert box displaying the failure message is shown to the user. Also there is a second AJAX call which takes places between ConfirmBooking.jsp and the UpdateHistory servlet. It updates the Booking history with the successful transaction details.

# WEB CONTENT FILES:

File Name: AravelliRamesh-HW4.war

Objective: To develop an Air Ticket Reservation Application and to demonstrate the control flow of all the stages involved in the process

# INPUT/OUTPUT:

**Login.jsp**

INPUT: User Name, Password

OUTPUT: Registration.jsp(New users)/FlightsSearch.jsp(Existing users)

**Registration.jsp**

INPUT: User Name, E-mail id, Date of Birth, Password, Confirm Password

OUTPUT: Login.jsp

**FlightsSearch.jsp**

INPUT: Source, Destination, Date of Travel, No of seats, Class

OUTPUT: FlightSearchResult.jsp

Booking OUTPUT: BookingHistory.jsp

**FlightSearchResult.jsp**

When clicked View and Book,

OUTPUT: ViewandBook.jsp

**ViewandBook.jsp**

When selected:

Select: TransactionPage.jsp

Back: FlightSearchResult.jsp

Home: FlightSearch.jsp

Add more Flights: Maps to FlightSearch.jsp

**ConfirmBooking.jsp**

INPUT: A/C Holder Name, Routing No, A/C No

OUTPUT: Ajax calls Confirm/Update function

**Transconfrm.jsp**

**INPUT:** On success the Transconfrm.jsp is a part of ConfirmBooking.jsp

**BookingHistory.jsp**

Displays the Booked flights and Home button.

**Display_Cart.jsp**

INPUT: Delete, Checkout

OUTPUT:

Delete: FlightSearch.jsp

Checkout: Transaction.jsp

**Login.java (Servlet)**

**doPost()**

**INPUT:** The login.jsp calls the doPost method of Login.java for user registration.

**OUTPUT:** On valid credentials of user the user is directed to the Flight Search query page. For an invalid user the user is directed to the new user registration page.

**Registration.java (Servlet)**

doPost()

INPUT: The post method is called by the Registration.jsp page.

OUTPUT: After valid registration the user is directed to the Login.jsp page. checkUserName() calls the Insert() to return true or false.

**Users.java**

**Check():**

**INPUT:** The Check() is used to check the property file for an existing user using the User object.

**OUTPUT:** The exceptions are written to handle the property file errors.

**Insert():**

**INPUT:** The user credentials are stored in the property file.

**OUTPUT:** returns true or false based on user presence in the database.

**CheckUsername()**

**INPUT:** Checks in the property file if the username exists or not.

**OUTPUT:** It returns true or false if username is present in database or not.

**FlightSearchServlet.java**

**doPost()**

**INPUT:** When the search button is clicked on the FlightsSearch.jsp.

**OUTPUT:** redirects to the FlightSearchResults page. All the flight details are retrieved from the FlightSearch.jsp and the information is stored in the session object.

**AddToCart.java:**

doGet()

INPUT: User clicks on the shopping cart after selecting his flight.

OUTPUT: It will delete the object from the session variable holding a Shopping cart flight or an array of Shopping cart flights.

**TransactionConfirmation.java:**

doPost():

INPUT: The user bank details are entered. The selected flights are viewed by the user.

OUTPUT: The isValidAccount() is used to verify the users bank information. The user is redirected to Transactioncnfrm.jsp on success else back to the Transaction.jsp. The transaction Controller is used to populate the flight details. isSufficientBalance() returns true if the user has the amount to book the ticket else return false.

**ViewandBookServlet.java:**

doPost():

INPUT: ViewandBook.jsp passes the parameters and calls this servlet.

OUTPUT: redirects to ViewandBook.jsp when the requested seats are selected. If seats are available the Shopping cart is populated if the operation equals addtocart. If the operation equals the checkout it maps to the confirm booking JSP.

**BookingHistory.java**

doPost():

INPUT: clicking on the Booking History will call this servlet.

OUTPUT: redirected to BookingHistory.jsp. Using the Flight Controller and User bean the servlet calls the getFlights() of the Flight Controller to retrieve the flight details and the user bean to get the user who booked the flight. The Booking history calls the booking bean and insert the data into the table in Booking JSP.

**GetPackageInfo.java**

doGet():

INPUT: The FlightSearch.jsp calls this servlet.
OUTPUT: If the targetflight is not null then check for packages available for that destination and populate that travel packages also populate the Flight Search Result servlet. This calls the FlightPackage Bean using the getFlightByID() retrieves the available flight packages to the Destination.

**BankServlet.Java**

Dopost():

Input: The Confirm Booking Calls this servlet

Output: This is done using the GSON library which takes the java objects as input returns the JSON Value. The isValidAccount() is used to verify the users bank information. The user is redirected to confirmbooking.jsp on success else back to the confirmbooking.jsp. The transaction Controller is used to populate the flight details. isSufficientBalance() returns true if the user has the amount to book the ticket else return false.

**FlightSearchResultServlet.Java:**

**doGet():**

**Input:** The Flight search page calls this servlet.

**Output:** This gives the Flight Search Result Jsp. It takes the Flight Bean and Flightpackage bean as the input returns the values by the getFlight() and getFlightByID().

**UpdateHistory.Java:**

doPost():

INPUT: clicking on the Booking History will call this servlet.

OUTPUT: redirected to BookingHistory.jsp. Using the Flight Controller and User bean the servlet calls the getFlights() of the Flight Controller to retrieve the flight details and the user bean to get the user who booked the flight. The Booking history calls the booking bean and insert the data into the table in Booking JSP.

**Beans:**

**Flight.java:**

INPUT: Flight.java is used when the getter and setter methods for flight details are obtained.

OUTPUT: Flight.java populates the bean with the obtained flight details.

**User.java:**

INPUT: When the request for getter and setter methods for login details are obtained.

OUTPUT: The login details of the registered user are filled in the beans.

**Bookings.java:**

INPUT: Bookings.java is used when the getter and setter methods for flight booking details are obtained.

OUTPUT: The bean is populated with the flight booking details.

**FlightPackage.java:**

INPUT: During the request for getter and setter methods for flight booking details are obtained.
OUTPUT: The bean is populated with the travel package details.

**Transactions.java:**

INPUT: The bank details getter and setter methods are obtained.

OUTPUT: The banks details are populated in the bean.