TESLA

FLIGHT RESERVATION SYSTEM

A PROJECT REPORT

Submitted by

JOY FRANCESCA MACHADO
MOUNIKA DANTULURU
SHIVANI GOWRISHANKAR

COLLEGE OF COMPUTER AND INFORMATION SCIENCE

NORTHEASTERN UNIVERSITY, BOSTON

DECEMBER 2014

PROJECT STATEMENT

The problem that we have tried to solve is to build an airline reservation system. The Airline Reservation System project is an implementation of a general Airline Ticketing website like Kayak, where the customers can search the availability of flights offered by various airlines to desired location and their prices. The features to be covered by the system are user-profiles (the user can create an account with the website), where the user can view his/her personal details given during account creation, booking history, ‘Buy’ and ‘Cancel’ option to purchase tickets and cancel already purchased tickets. The user can input the source location, destination location, arrival date and destination date and view search results to book tickets according to his travel plans. The user can write reviews and rate the airlines about his experience during a journey. The user can also view other user’s comments and rating of an airline just before buying a ticket. The user also gets points for every purchase and can also purchase tickets using the points.

PROPOSED SOLUTION

We plan to implement the project by creating a profile page for the user. From the profile page of the user, the user can search the availability of tickets. This search query is sent to the web services and the data is fetched by using an external API which returns the data in JSON format. Once the data has been fetched it will be displayed in the search results page. The user can choose any one of the search results and can book a flight. While booking flight, the user will also be able to view the rating and comments on the airline the user has chosen to travel with.

The user also has the option of using either dollars or frequent flier miles while booking a flight. The frequent flier miles are accumulated for every user, when he/she purchases a ticket using dollars. The accumulated frequent flier miles can be redeemed whenever needed and used to book a flight.

The user can rate and review an airline that he/she had travelled with. The user can also change the rating and review after given, if he/she changes his/her mind.

The user profile page displays the current bookings made by a user and also allows the user to cancel a booking or view the itinerary of a booking.

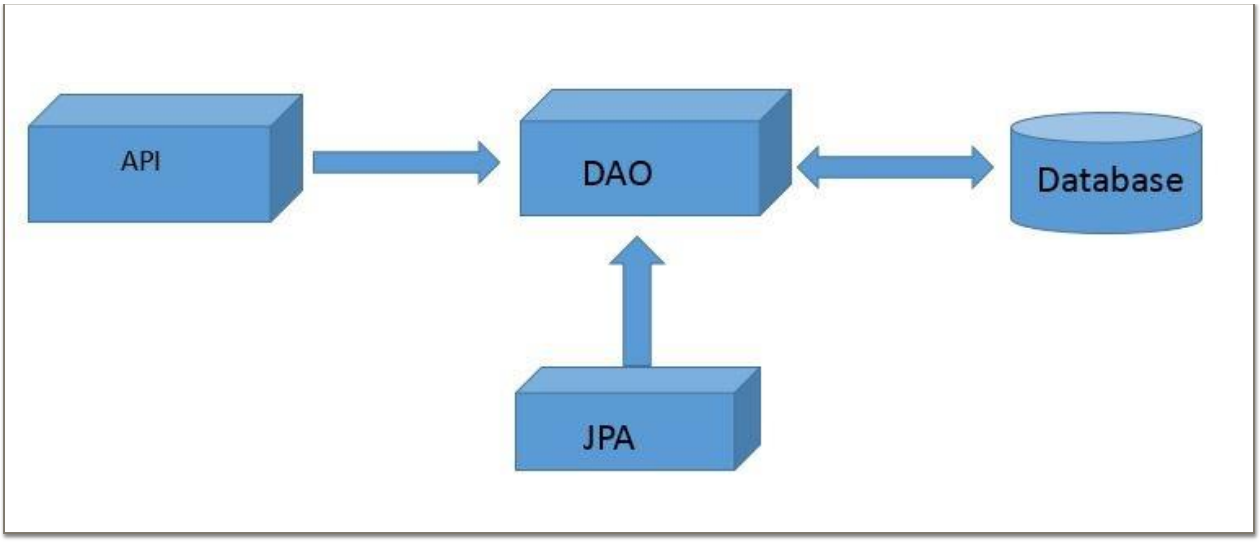
ARCHITECTURE

The initial part of the project involved fetching data from the external API. The API that was used was ‘Sabre Dev Studio’. The data obtained from the API was in JSON format and were parsed using a server side web service client to display the search results. The data were parsed using JSON.simple toolkit- a java toolkit for JSON to decode JSON text.

Once the user selects an itinerary to book and enters the passenger details, the itinerary and passenger details are persisted into the database using JPA.

The persisted data is displayed using JSP to the user.

The user admin is created and can create, delete, update and read all user profiles. JWS was implemented to handle the use cases of user admin.



API’s

The External API that we have used is ‘Sabre Dev Studio’ API to display the flight search results from departure location to destination location for that particular departure date and return departure date to travel with in the country live data of flights is shown.

Internal Java persistence API for the user admin implemented using JWS technology to Create a User, Delete a User, Update User Details and Read All the Users in the Database

TECHNOLOGIES USED

- JPA
- JWS
- Bootstrap
- CSS
- MYSQL
- JSON

USE CASES

- **Use Case:** Guest Registration [userRegistration]

  Description: Guest wants to register to the web application

  Actors: Customer

  Preconditions: A user with same credentials should not exist

  Steps:

  Actor Actions: Guest Submits Details

  System Responses: Guest successfully registered as user

  Post Condition: User directed to login page

  Alternate paths: Guest submits user name that already exists

  Error Message: User name already exists

- **Use Case:** User Login[userLogin]

  Description: User wants to login

  Actors: Customer

  Preconditions: User name should exist and password should match

  Steps:

  Actor Actions: User submits login details

  System Responses: User authentication passed

  Post Condition: User directed to his profile page

Alternate paths: User authentication fails

Error Message: The username or password you entered is incorrect

- **Use Case:** Search for Flights[flightSearch]

  Description: User wants to search for flights

  Actors: Customer

  Preconditions: Flights for the requested dates should have seats available

  Steps:

  Actor Actions: User selects arrival, departure dates and boarding,destination locations

  System Responses: Flight details displayed

  Post Condition: User should give the dates of travel and location of departure and arrival

  Alternate Paths: No flights available for the user entered details

  Error Message: Flights not available

- **Use Case:** Select a flight[selectFlight]

  Description: User selects a flight from the search results

  Actors: Customer

  Preconditions: Flights for the requested dates should have seats available

Steps:

Actor Actions: User selects a flight from the search results

System Responses: Gives the flight journey details

Post Condition: Displays the flight journey details like time of departure, arrival and stops etc.

- **Use Case:** Book Seats [seatsBooking]

  Description: User selects the number of seats(i.e passengers traveling) and books them

  Actors: Customer

  Preconditions: User should select the number of passengers traveling

  Steps:

  Actor Actions: User clicks the book button

  System Responses: booking details are displayed

  Post Condition: The tickets are purchased and are added to his booking list

  Alternate Paths: User booking fails as number of seats requested are no longer available

  Error Message: Booking failed

- **Use Case:** Enter Passenger Details

Description: Once the number of seats are selected the user should enter each passenger details in order to process the booking

Actors: Customer

Preconditions: User should select the number of passengers traveling

Steps:

Actor Actions: User clicks the book button

System Responses: booking details are displayed

Post Condition: The tickets are purchased and are added to his booking list

Alternate Paths: User booking fails as number of seats requested are no longer available

Error Message: Booking failed

- **Use Case:** View Booking History [viewBooking]

  Description: User wants to view an booking history and is displayed in descending order of date of booking (i.e latest transaction first)

  Actors: Customer

  Preconditions: Orders exists, customer authenticated

  Steps:

  Actor Actions: User requests booking history

  System Responses: All booking details are displayed

Post Condition: all orders are displayed in descending order of date of booking (i.e latest transaction first)

Alternate path: User requests booking history, no history found

Error message: No purchases made

- **Use Case:** View Travel Details [Itenary]

  Description: User selects travel details of a particular journey from his booking history

  Actors: Customer

  Preconditions: should be present in the booking history

  Steps:

  Actor Actions: User clicks on the travel ID from the booking history

  System Responses: Displays the journey details

  Post Condition: System displays all the journey details related to that booking i.e travel id

  Alternate Paths: Booking has been canceled

  Error Message: Shows all the journey details with the message "This is no longer available as the order has been canceled"

- **Use Case:** Cancel a Booking[cancelBooking]

Description: User selects a travel id of upcoming journey and selects the cancel button for that passenger or the complete travel

Actors: Customer

Preconditions: The journey should be upcoming

Steps:

Actor Actions: User selects either a particular passenger or the whole travel

System Responses: Cancels order

Post Condition: The journey details are still visible but the order is canceled for a selected passenger or the whole travel

Alternate Paths: If the travel id is of past journey

Error Message: Order cannot be canceled as it is a past journey

- **Use Case:** Comment on Airlines [Comments]

  Description: A user can comment about his experience during the journey with a particular airlines

  Actors: Customer

  Preconditions: the airlines should be displayed

  Steps:

  Actor Actions: User selects a particular airlines and comment there

  System Responses: the comment if it is posted

Post Condition: The posted comment is displayed under the airlines reviews

Alternate Paths: If there is no text

Error Message: Comment cannot be posted as there is no description

- **Use Case:** Rate an Airline [rating]

  Description: A user can rate a particular airlines

  Actors: Customer

  Preconditions: the airlines should be displayed

  Steps:

  Actor Actions: User can give a rating of 0 to 5 for a particular airlines

  System Responses: The rating of the user is posted

  Post Condition: The average rating of all customers is displayed

  Alternate Paths:

  Error Message:

- **Use Case**: View Frequent flier points

  Description: User views the frequent flier points

  Actors: Customer

  Precondition: User must have points in their profile

  Steps:

Actor actions– User views the profile

System Responses – frequent flier points are displayed for a user profile

Post condition: frequent flier points displayed to the user profile

- **Use Case:** Edit Comment [editComment]

  Description: A customer can edit a comment if it is necessary

  Actors: Customer

  Preconditions: customer should have created a comment

  Steps:

  Actor Actions: Customer selects a comment to edit

  System Responses: the comment is edited

  Post Condition: The comment is edited

  Alternate Paths:

  Error Message:

- **Use Case:** Edit Rating [edit Rating]

  Description: A customer can edit a rating if it is necessary

  Actors: Customer

  Preconditions: customer should have created a rating

Steps:

Actor Actions: Customer selects a rating to edit

System Responses: the rating is edited

Post Condition: The rating is edited

Alternate Paths:

Error Message:

- **Use Case:** Update Profile

  Description: A DBA can edit and update user profile

  Actors: Database Administrator

  Preconditions: the user should have admin rights

  Steps:

  Actor Actions: A DBA can edit previous data

  System Responses: The updated information is stored

  Post Condition: Displays the updated information on his profile

  Alternate Paths:

  Error Message:

- **Use Case:** View All Users [ViewAllUsers]

  Description: A DBA can view all users

Actors: Database Administrator

Preconditions: the user should have admin rights

Steps:

Actor Actions: A DBA will view all users' profile

System Responses: The profile of all users are displayed

Alternate Paths: The users not signed up.

Error Message: No user exist

- **Use Case:** Delete User

  Description: A DBA can delete user if the user wants to delete his/her profile

  Actors: Database Administrator

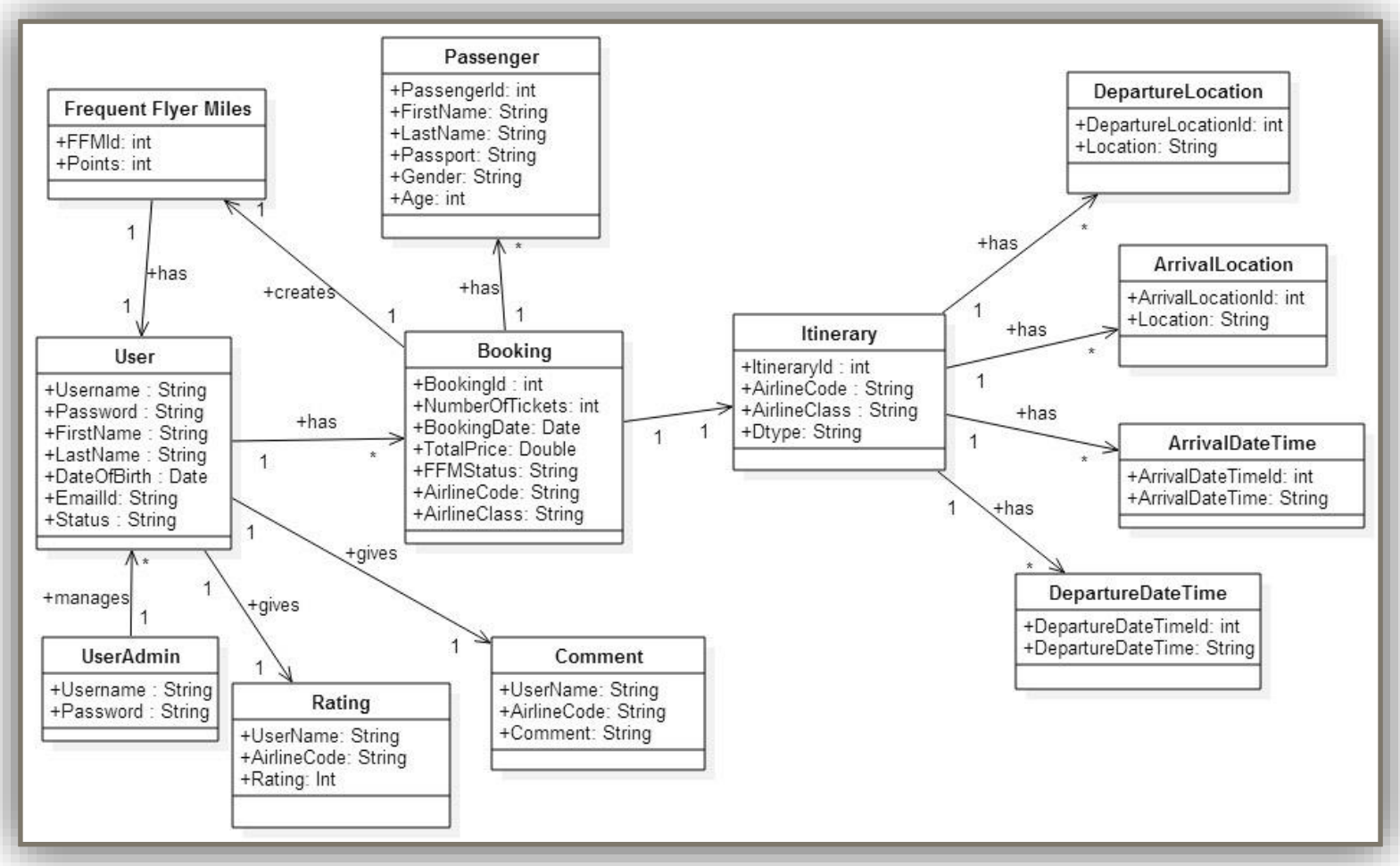
  Preconditions: the user should have admin rights

  Steps:

  Actor Actions: A DBA will check the details and delete the user

  System Responses: The user is denied permission to access

UML DIAGRAM



The Association classes are :

1. Booking and Frequent Flyer Miles
2. Frequent Flyer Miles and USER
3. Booking and Itinerary
4. Booking and Passengers

2-NF:

The table Itinerary after applying 2NF we get the tables Arrival Date, Arrival Location, Departure Date and Departure Location tables.

FUTURE SCOPE

- To fetch data from different API’s and give price comparison.
- Ability to suggest users travel packages and latest deals.
- Provide Hotel booking.
- Give email alerts about the upcoming journey.