Project Topic: Design of a database system for a frequent flier program
Deliver the .ZIP file (other formats won't be accepted) containing: (1) your ***frequentflier web folder*** (refer to part 1), (2) your ***FreqFlier Android Studio project folder*** (refer to part 2), (3) a ***vlink.txt*** file containing a link to a 15-min recorded video demonstrating the operation of your application (refer to part 3)

Part 1 (60%)

The server-side code is comprised of 1 Java Servlet and 7 Java Server Pages. You should abide by the following guidelines in detail to ensure a smooth grading process for the GTAs.
Create a folder named ***frequentflier*** in the webapps folder of your Apache Tomcat Server. This folder should contain all your JSP files and any resources you might need when developing the client application. It also contains the ***WEB-INF/classes*** folder containing the .class and .java files of the Login Servlet. You are required to develop the JDBC code to implement the following Java Server-side components. Your JDBC code typcially follows the same exact steps as we did in the lectures. What changes is the query to send to the database. Accordingly start by developing the SQL queries needed in each server component and then build the JDBC code and test it from a browser session. A good convention to follow when you have a response output from an SQL statement is to separtate the columns by a particular character and the rows by another character to facilitate the process of parsing the response in the Android mobile app later. Use a ',' to separate the columns and a '#' to separate the rows.

Java Servlets

Login.java (5% for correct code, 2.5% for successful execution)
Annotation reference name: /login
Request input parameters: passenger username and password Response output:

- The String "Yes:pid" if the username and password combination is valid and exists in the LOGIN table in the database. pid is the passenger id.
- The String "No" if the username or password are incorrect

URL format: http://127.0.0.1:8080/frequentflier/login?user=certainuser&pass=certainpass

Java Server Pages

Info.jsp (5% for correct code, 2.5% for successful execution)
Request input parameters: A passenger id
Response output: The passenger name, the number of points collected by the passenger. URL format: http://127.0.0.1:8080/frequentflier/Info.jsp?pid=certainpid

Flights.jsp (5% for correct code, 2.5% for successful execution)
Request input parameters: A passenger id
Response output: the flight id, flight miles, and destination for all the flights belonging to the specified passenger id.
URL format: http://127.0.0.1:8080/ frequentflier /Flights.jsp?pid=certainpid

FlightDetails.jsp (5% for correct code, 2.5% for successful execution) Request input parameters: The flight id.
Response output: the flight dept_datetime, flight arrival_datetime, flight miles, trip ids, and trip miles belonging to the specified flight id.
URL format: http://127.0.0.1:8080/frequentflier/FlightDetails.jsp?flightid=certainflightid

AwardIds.jsp (5% for correct code, 2.5% for successful execution)
Request input parameters: A passenger id
Response output: the distinct award ids belonging to the specified passenger id. URL format: http://127.0.0.1:8080/frequentflier/AwardIds.jsp?pid=certainpid

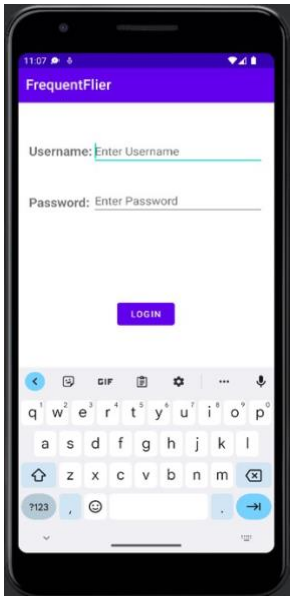
RedemptionDetails.jsp (5% for correct code, 2.5% for successful execution)
Request input parameters: an award id and a passenger id
Response output: the award description, the number of points needed, redemption date, and exchange center name for the specified (award id, passenger id) pair.
URL format:
http://127.0.0.1:8080/frequentflier/RedemptionDetails.jsp?awardid=certainawardid&pid=certainpid

GetPassengerids.jsp (5% for correct code, 2.5% for successful execution)
Request input parameters: the current passenger id
Response output: the passenger ids in the database excluding the current passenger id URL format: http://127.0.0.1:8080/frequentflier/GetPassengerids.jsp?pid=currentpid

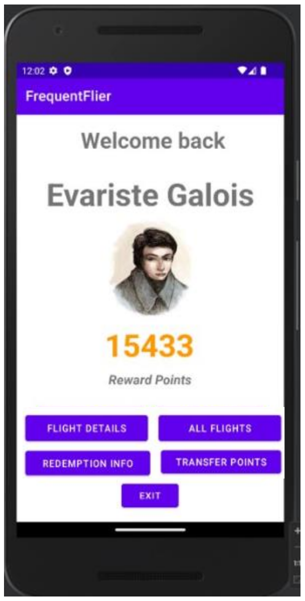
TransferPoints.jsp (5% for correct code, 2.5% for successful execution)
Use 2 UPDATE SQL DML statement in this JSP file. The first one to deduct the number of points from the current passenger point account and the second one to add the specified number of points to a particular passenger id.
Request input parameters: a source passenger id, a destination passenger id and the number of points to be transferred from the point account of the source passenger id to the point account of the destination passenger id.
Response output: A message indicating that the transfer is successful.
URL format: http://127.0.0.1:8080/frequentflier/TransferPoints.jsp?spid=source pid&dpid=destination pid&npoints=numberofpointstotransfer

Part 2 (30 %) [5% per activity]

In this part you will develop an Android client application project (name it FreqFlier) to demonstrate a small portion of your project operation. Due to the limited amount of time available, your app will consist of few activities for testing the interaction of your server-side code with the database. The Figures provide an idea of the UI anticipated for the activities and a specification of the server-side components that may be used to support the implementation of those activities.



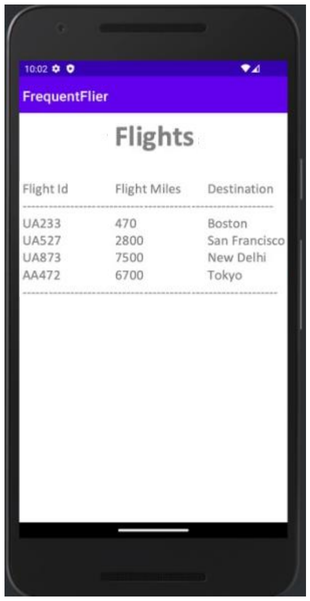
MainActivity Relies on the ***Login*** Servlet



MainActivity2

Relies on ***Info.jsp*** to fill the passenger name and number of points.

You should have .jpeg photos for 5 sample passengers (you can pick any sample photos of individuals from the Web). Name each photo according to the following scheme: pid.jpeg (pid is the passenger id). Store the images in an images folder under the frequentflier application folder on Apache Tomcat. MainActivity2 should request the photo of the specified passenger id to display it.

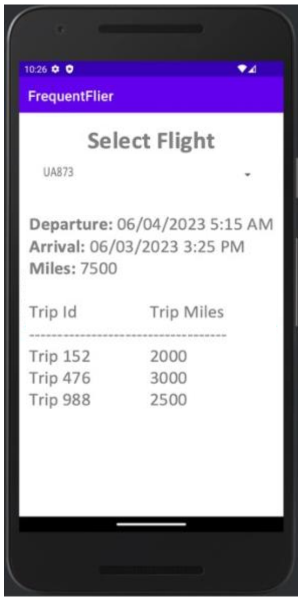


Flights

Flight Id	Flight Miles	Destination
UA233	470	Boston
UA527	2800	San Francisco
UA873	7500	New Delhi
AA472	6700	Tokyo

MainActivity3

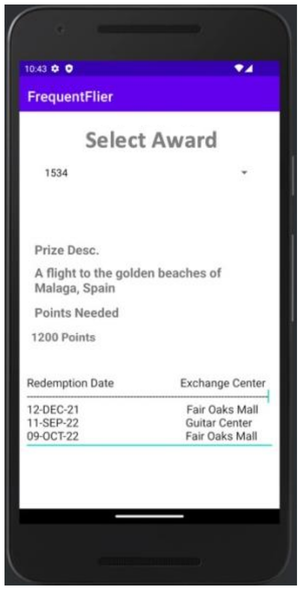
Relies on ***Flights.jsp*** to fill the flights information.



UA873

Departure: 06/04/2023 5:15 AM
Arrival: 06/03/2023 3:25 PM
Miles: 7500

Trip Id	Trip Miles
Trip 152	2000
Trip 476	3000
Trip 988	2500



1534

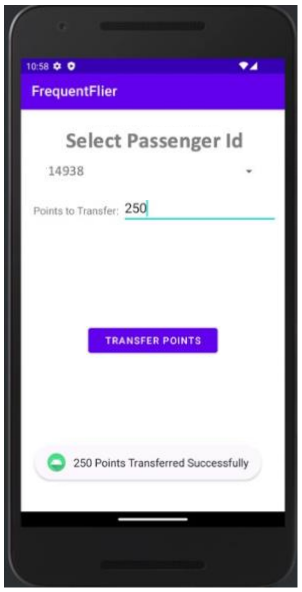
Prize Desc.

A flight to the golden beaches of Malaga, Spain

Points Needed

1200 Points

Redemption Date	Exchange Center
12-DEC-21	Fair Oaks Mall
11-SEP-22	Guitar Center
09-OCT-22	Fair Oaks Mall



14938

Points to Transfer: 250

250 Points Transferred Successfully

MainActivity4 Relies on:
(1) ***Flights.jsp*** to fill the spinner with the flight ids belonging to the passenger id
(2) ***FlightDetails.jsp*** to fill the flight departure date/time, flight arrival date/time, flight miles and trip ids and miles.

MainActivity5 Relies on:
(1) ***AwardIds.jsp*** to fill the spinner with the award ids of the passenger id
(2) ***RedemptionDetails.jsp*** to fill the award description and points needed for the selected award id. Moreover, it fills the redemption dates and exchange center names belonging to the (award id, passenger id) redemption information.

MainActivity6 Relies on:
(1) ***GetPassengerids.jsp*** to fill the spinner with passenger ids excluding the current passenger.
(2) ***TransferPoints.jsp*** to increase the point accounts of the passenger id selected in the spinner with the number of points entered. This JSP should decrease the amount of points from the current passenger id.

Part 3 (10 %)

Each group will record a 15-min video demonstration of their application and post the video to YouTube (or any video hosting service). A **link** to the video should be included in the ***vlink.txt*** file. Any broken link or a link that is not accessible will result in a ZERO grade. **Don't upload the whole video file to Blackboard**. All the group members should participate in the project demonstration. The following is what should be emphasized on in your video:

1. The code of the different Server-side components (Servlet and JSPs) (2.5%)
2. The Java code of the Android Activities (2.5%)
3. Testing each of the Server-side components using a browser session (2.5%)
4. Running the Android activities with the Android Studio emulator and demonstrating their functionality and interaction with the Servlet and JSPs (2.5%)