Final project requirement version final (07/23/2022)

We must design and implement a flight reservation system for a flight agency using Java OOP. They have an admin that can modify master data. They may have up to 10 employees that can search, reserve, and print tickets for customers. And they may have an unlimited number of customers who can buy tickets from employees. Also, they have a cashier that finalizes tickets after receiving the whole amount of money in the form of cash or bank transfer.

The system consists of 3 modules: the flight module, the user module, and the accounting module.

1. User module:

This module contains login information, contact information, and activity history information for employees and customers. As this is not an online system, then each customer has a client id but cannot log in to the system. However, the rest of the 12 users (Admin, Employees, and Cashier) can log in with their user id and password. I expect that you search the internet and find the best way to implement session management in Java. You are allowed to use any framework of your choice if you need it. For example, you may say that our group needs to develop based on the Spring framework for some reasons (you must explain the reasons in your report).

Any activity like login, search, issuing a ticket, and logout must be recorded in a log table in the database.

2. Accounting module

A simple accounting system that keeps track of each ticket. When a ticket is issued, the price of the ticket must be paid in cash or via a bank transfer. When a ticket is issued, the first airline of the ticket will be credited for the price of the ticket minus 5%, which is the agency commission. The system must provide a report regarding the current balance of each customer, the cash register machine, the bank account, and finally, the business partner airlines.

When a ticket price is paid by the customer, the cashier will change the status of the ticket from reserved to booked after receiving the money. Each payment is linked to a ticket with a reference number that is the ticket(s) reservation number. Please note that a reservation may contain more than one ticket.

The cashier is responsible for entering the incoming balance into the cash desk or the bank account.

3. Flight module

The flight module contains at least the following tables: countries; airports; airlines; routes; planes; schedules; reservations. All this data except the schedule table have been collected from openflights.org. The reservation table must be designed by you and will be filled/searched by the system. The schedule table has a special design that must be filled by some random or test designed data. The explanation here regarding the tables collected from the openflights.org website.

3.1 Countries table

The country dataset contains a list of ISO 3166-1 country codes, which can be used to look up the human-readable country names for the codes used in the Airline and Airport tables. Each entry contains the following information:

name	Full name of the country or territory.
iso_code	Unique two-letter ISO 3166-1 code for the country or territory.
dafif_code	FIPS country codes as used in DAFIF. Obsolete and primarily of historical interested.

The data is UTF-8 encoded. The special value **\N** is used for "NULL" to indicate that no value is available and is understood automatically by MySQL if imported.

Notes:

- Some entries have DAFIF codes, but not ISO codes. These are primarily uninhabited islands without airports and can be ignored for most purposes.

Sample entries
`"Australia","AU","AS"`
`"Ashmore and Cartier Islands",\N,"AT"`

3.2 Airports table

Each entry contains the following information:

Airport ID	Unique OpenFlights identifier for this airport.
Name	Name of airport. May or may not contain the **City** name.
City	Main city served by airport. May be spelled differently from **Name**.
Country	Country or territory where airport is located. See Countries to cross-reference to ISO 3166-1 codes.
IATA	3-letter IATA code. Null if not assigned/unknown.
ICAO	4-letter ICAO code. Null if not assigned.
Latitude	Decimal degrees, usually to six significant digits. Negative is South, positive is North.
Longitude	Decimal degrees, usually to six significant digits. Negative is West, positive is East.
Altitude	In feet.
Timezone	Hours offset from UTC. Fractional hours are expressed as decimals, eg. India is 5.5.
DST	Daylight savings time. One of E (Europe), A (US/Canada), S (South America), O (Australia), Z (New Zealand), N (None) or U (Unknown). *See also:* Help: Time
Tz database time zone	Timezone in "tz" (Olson) format, eg. "America/Los_Angeles".
Type	Type of the airport. Value "airport" for air terminals, "station" for train stations, "port" for ferry terminals and "unknown" if not known. *In airports.csv, only type=airport is included.*
Source	Source of this data. "OurAirports" for data sourced from OurAirports, "Legacy" for old data not matched to OurAirports (mostly DAFIF), "User" for unverified user contributions. *In airports.csv, only source=OurAirports is included.*

The data is UTF-8 encoded.

Note: Rules for daylight savings time change from year to year and from country to country. The current data is an approximation for 2009, built on a country level. Most airports in DST-less regions in countries that generally observe DST (eg. AL, HI in the USA, NT, QL in Australia, parts of Canada) are marked incorrectly.

Sample entries
`507,"London Heathrow Airport","London","United Kingdom","LHR","EGLL",51.4706, -0.461941,83,0,"E","Europe/London","airport","OurAirports"`
`26,"Kugaaruk Airport","Pelly Bay","Canada","YBB","CYBB",68.534401,-89.808098, 56,-7,"A","America/Edmonton","airport","OurAirports"`
`3127,"Pokhara Airport","Pokhara","Nepal","PKR","VNPK",28.200899124145508, 83.98210144042969,2712,5.75,"N","Asia/Katmandu","airport","OurAirports"`
`8810,"Hamburg Hbf","Hamburg","Germany","ZMB",\N,53.552776,10.006683,30,1,"E", "Europe/Berlin","station","User"`

3.3 Airline table

Each entry contains the following information:

Airline ID	Unique OpenFlights identifier for this airline.
Name	Name of the airline.
Alias	Alias of the airline. For example, All Nippon Airways is commonly known as "ANA".
IATA	2-letter IATA code, if available.
ICAO	3-letter ICAO code, if available.
Callsign	Airline callsign.
Country	Country or territory where airport is located. See Countries to cross-reference to ISO 3166-1 codes.
Active	"Y" if the airline is or has until recently been operational, "N" if it is defunct. This field is *not* reliable: in particular, major airlines that stopped flying long ago, but have not had their IATA code reassigned (eg. Ansett/AN), will incorrectly show as "Y".

The data is UTF-8 encoded. The special value **\N** is used for "NULL" to indicate that no value is available and is understood automatically by MySQL if imported.

Notes: Airlines with null codes/callsigns/countries generally represent user-added airlines. Since the data is intended primarily for current flights, defunct IATA codes are generally not included. For example, "Sabena" is not listed with a SN IATA code, since "SN" is presently used by its successor Brussels Airlines.

Sample entries
`324,"All Nippon Airways","ANA All Nippon Airways","NH","ANA","ALL NIPPON","Japan","Y"`
`412,"Aerolineas Argentinas",\N,"AR","ARG","ARGENTINA","Argentina","Y"`
`413,"Arrowhead Airways",\N,"","ARH","ARROWHEAD","United States","N"`

3.4 Routes table

Each entry contains the following information:

Airline	2-letter (IATA) or 3-letter (ICAO) code of the airline.
Airline ID	Unique OpenFlights identifier for airline.
Source airport	3-letter (IATA) or 4-letter (ICAO) code of the source airport.
Source airport ID	Unique OpenFlights identifier for source airport.
Destination airport	3-letter (IATA) or 4-letter (ICAO) code of the destination airport.
Destination airport ID	Unique OpenFlights identifier for destination airport.
Codeshare	"Y" if this flight is a codeshare (that is, not operated by *Airline*, but another carrier), empty otherwise.
Stops	Number of stops on this flight ("0" for direct)
Equipment	letter codes for plane type(s) generally used on this flight, separated by spaces

The data is UTF-8 encoded. The special value **\N** is used for "NULL" to indicate that no value is available and is understood automatically by MySQL if imported.

Notes:

- Routes are directional: if an airline operates services from A to B and from B to A, both A-B and B-A are listed separately.
- Routes where one carrier operates both its own and codeshare flights are listed only once.

Sample entries
`BA,1355,SIN,3316,LHR,507,,0,744 777`
`BA,1355,SIN,3316,MEL,3339,Y,0,744`
`TOM,5013,ACE,1055,BFS,465,,0,320`

3.5 Planes table

Each entry contains the following information:

Name	Full name of the aircraft.
IATA code	Unique three-letter IATA identifier for the aircraft.
ICAO code	Unique four-letter ICAO identifier for the aircraft.
Total Capacity	Total amount of seats.
First Cap.	Number of 1st class seats.
Business Cap.	Number of business class seats.
Prem. Eco Cap.	Number of premium economy class seats.
Economy Cap.	Number of economy class seats.

The data is UTF-8 encoded. The special value **\N** is used for "NULL" to indicate that no value is available and is understood automatically by MySQL if imported.

Notes:

- Aircraft with IATA but without ICAO codes are generally aircraft classes: for example, IATA "747" can be any type of Boeing 747, whereas IATA "744"/ICAO "B744" is specifically a Boeing 747-400.

Sample entries
`"Boeing 787","787",\N,159,6,7,17,129`
`"Boeing 787-10","78J","B78X",193,7,9,21,156`
`"Boeing 787-8","788","B788",202,8,10,22,162`

3.6 Schedules table

The data would contain the following fields per row:

Source airport, destination airport, start date, end date, days of operation (days of week), departure time in GMT, arrival time in GMT, flight number, aircraft type, flight duration

Sample rows:
LCG,MAD,-,2013-07-19,1 2 3 4 5 6,13:25,14:45,IB513,320,01:20
LCG,MAD,2013-07-22,-,1 2 3 4 5,13:25,14:45,IB513,320,01:20
LCG,MAD,2013-07-20,-,6,13:30,14:50,IB513,321,01:20

This is an indicative sample, not the final format, you can extend it as you like. You must enter some sample data using some random data generator function or a bunch of designed test data. I recommend you design data for this table. For example, make sure you have a possible flight from A to B and from B to C some hours later. Then when you search for a flight from A to C your system must show there is a connection flight between A to C.

3.7 Reservation table

You need to design this table based on your own understanding. When an employee searches a flight between A and B, the number of available seats in each class must be provided separately.

Your system can offer possible flights with zero, one or two connection flights as soon as there is 4 hours gap between arrival and departure flights in a connection airport.

Deliverables

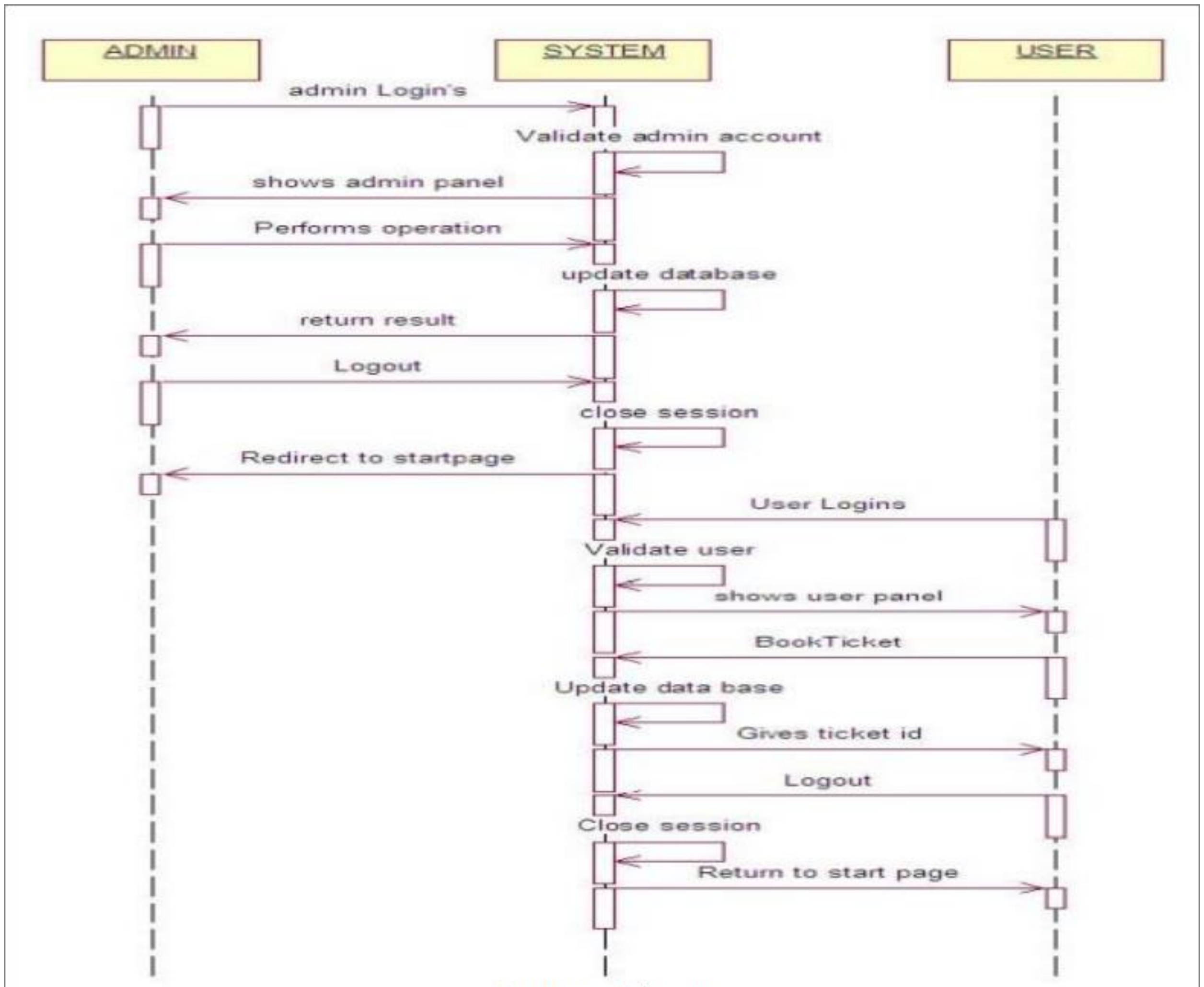
You must use a private Git. I can recommend Bitbucket. Invite both the instructor and TA to your repository. Use our ucalgary.ca emails. We will monitor your activity there. All users must have a clear name. After you submit your code on 08/18 you are allowed to publish your code on GitHub publicly. If we realized that a user has not participated enough, he or she may receive a different mark.

We do not need GUI for this project, try to have a robust implementation. Instead, when the user presses F1 in any step, it must show a nice help to the user to guide them through the process.

You can select any Database of your choice. I recommend using MariaDB or MySQL. All DDL for generating structures and CSV files for inserting data must be delivered in a folder call it as installation. In the installation, folder put a Readme.md file and explain the steps for running your program in a raw system.

Also, you need to deliver a report between 2 to 4 pages as a PDF that explains the architecture of your program. Please draw all related UMLs and try to keep the report as short as possible.

A sample sequence diagram[1] can be like this:



Sequence Diagram

Please do not forget to upload all deliverables as a Zip file inside the D2L also.

[1] https://ijisrt.com/assets/upload/files/IJISRT21JUN223.pdf