Guidelines for Final Project

Deadlines: Initial Design **(3%):** Due 4/17, Final Project **(7%):** Due 5/1
This is a group project (maximum 4 member teams)

Objective:

Design, develop and implement a working software system in Java. All code must be maintained on GitHub (See instructions on Canvas)

Problem Statement – Airline Ticket Booking System

Congrats! You're hired to create a project that lets users search for flights and book tickets. You can find more details about each perspective below

1. User's Perspective:

- User should be able to register by specifying username and password. Usernames should be unique
- User should be able to log in to the app with his/her own credentials (also log out)
- Once logged in, user can be presented with a menu asking for information like:
    - Round trip/One way
    - Dates for the trip
    - Source and destination
    - Number of tickets (Adult/Child)
- After entering appropriate information, user can hit search to view results of available flights (sorted in some default order)
- Note that you shouldn't show flights that are completely booked
- Can change sorting order according to different parameters (like time, cost, number of layovers, preferred airlines etc)
- Can also filter the results according to same parameters
- Can pick one flight and proceed to choose seats/class/additional baggage etc. Price of ticket changes accordingly
- Note that if a user books a seat for flight A, logs out and another user tries to book same seat for flight A, it should not be allowed.
- For long trips, ask for meal preferences
- After all options are chosen, user is shown total amount due
- Can pay with card and/or redeem miles (miles can only be gifted by airline and not earned as we do not want to track whether the person actually flew or not. If someone books a ticket, assume they travelled as well)
- You do not need to implement actual payment gateway. Just show appropriate messages and assume every user has adequate bank balance.

- Has option to cancel ticket only if within specific time limit (beyond which we will assume that the person took that flight & travelled) – You can decide how to reimburse the costs (points, free future flight, miles etc). You need to keep track of when a user booked a ticket and then compare it with a certain value to see if it's eligible for cancellation.
- Users can leave ratings and comments for each airline/provider. This can be viewed by any user of the system (the review should have user's name and date of last flight booked with that airline)
- Make sure that only users who have actually booked tickets with an airline is presented with the option to review
- Users should be able to view their booking history

2. Airline's perspective

- Airlines can also register and log in to the system (say, their representative)
- You should have some logic to determine whether current registration is a normal user or an airline
- Once logged in they can
    - Add/Remove flights
    - Note that two airlines cannot have a flight with same number
    - Provide points/miles to customers whose flights were cancelled
    - Blacklist certain customers from flying with them
    - Gift miles to certain customers
    - View all reviews (ratings and comments) Note that an airline cannot see customer reviews for other airlines.
- Note that you should add adequate information when creating new flights (like flight number, capacity per seating class, layovers etc)

Note: For all features, you are free to use your creativity and any logic to implement the same (e.g. any kind of menus, options etc). You are encouraged to add more features to make it as realistic as possible.

3. Optional Feature

- Generate order confirmation file
    - For each user, once they complete payment, create a properly formatted HTML file containing essential information. The generated confirmation can have information like the flight name, date, seat number etc. In essence, make the order confirmation as realistic as you can.

The software development cycle includes requirement specification, design and analysis, coding and testing phases. You will also prepare a report documenting each step used in the software design cycle and will use UML diagrams to properly represent

all classes used in your design. (You may use free software called StarUML which can be downloaded from http://staruml.sourceforge.net/en/download.php)

System Features:

The System that you are designing offers a variety of design choices. You are encouraged to design the software system that provides the most realistic user experience. Here are *some* of the required functionalities:

- Users should be able to register and login to their accounts using their unique credentials.
- User is presented with multiple options to pick flights
- Filtering and sorting the results
- Review system with comments
- Gifting miles to customers
- Airlines can add/remove flights, blacklist etc
- See features mentioned in each perspective above for detailed requirements
- Make your system user friendly by providing enough guidelines and help to use the system (For example: if your system is expecting any input in a specific format, be sure to specify that in the instructions as well as the Testing part of your report)
- **Note:** Your application must have **state persistence** by saving the necessary data to files. State persistence allows your system to outlive the process that created it. For example, if you add a few new flights to the system when you initially run it and then close the execution before running the system again, the system must be able to remember the previous flights and its state. This will allow you to use the previously entered information without having to create flights, users, etc. again and again. You can complete the entire project by storing data in files or any other persistent storage medium.

You have tremendous amount of freedom in designing this application. You are welcome to add any additional functionality to your system. Just imagine that you are an actual developer of the application.

Note: The design of a GUI is optional. You can also develop a command-line based system

System Design & Development:

The final project includes two major components: Project Design and Implementation.

Design (3%): The design component will consist of the software design for the project and will include the list of all classes (member variables and functions) to be used, their relationships & collaboration, as well as databases/files. The initial design must precede

any implementation. You will start by analyzing the requirements of the project and by identifying the classes required along with attributes and functionalities. A doc/docx document listing the design along with UML diagrams (Class diagrams, activity diagrams, etc.) must be submitted by April 17th (1159pm) on Canvas. Please include the following:

- Skeleton Code with all classes (with member attributes and member functions)
- What are class hierarchies and relationships? (in your report)
- All other data structures or files to be used (in your report)
- Peer evaluations are REQUIRED
- Also setup a private repository for your source code on github and add all instructors as collaborators (See instructions on Canvas).

It would be beneficial for you to analyze your system thoroughly and provide detailed UML diagrams. Here are some of the questions, answering those will help you put on the good path:

- How many classes will you have and how will they interact?
- How will you store user/airline ID, passwords, menu options?
- How will you represent the user options?
- How will you deal with uniqueness of flights?
- How will you link customer info to flight booking?

Implementation (7%): Once you complete the initial design, setup a *private* GitHub repository and start implementing the airline, and user functionalities separately. Test these separately with individual drivers. Be sure to add the appropriate user-friendly menu options that would allow the user to select various actions easily. Ensure that the menu options only work when correct input is provided by the user. Test each option individually to ensure that all available functionality is properly implemented. All source code must be shared amongst team members and instructors via github. Now imagine yourself as the user of the system (as User, Airline etc) and observe the behavior of the system (and see how it changes). Reflect on the efficiency of your choices. Review your initial design and make the necessary changes to remove any unnecessary attributes and functions. You may have to add some more functionality that you might have missed in your initial design. Implement the updated design and report the updates to the initial design along with a discussion regarding the efficiency of your choices. A doc/docx document listing the final design including UML diagrams must be submitted along with working Source Code and video demo by May 1st (11.59pm). See the **checklist** given below.

Testing: Develop and build unit tests for each operation. Create different drivers for testing each component of the system as a user/airline separately and then integrate the entire system and test with a separate main driver. Please ensure that the complete system is properly tested before submission.

Deliverables:
The following items must be submitted:

- **All source and header files** related to the system implementation (on Canvas as well as GitHub).
- **All drivers** that are part of the testing from all perspectives separated in different folders
- A **README** file with any information regarding compilation and testing I need to know in order to successfully compile and run your system. Include **any other files** needed to compile or test
- **Source code must be properly organized, readable, and must use proper best coding practices.**
- **The report file** with the final design with discussion, and details of testing activities. Feel free to show additional testing you performed in the report
- Video Demo (10 minutes long max) showing all features of your working system
- Peer evaluations are REQUIRED

Project Checklist:
You can use the following checklist to ensure that you have submitted everything required for the project:

- Did you submit evidence of successful compilation or testing (screen captures)?
- Did you submit the instructions required to compile your code?
- Did you add all instructors as collaborators to your private GitHub repository?
- Did you submit all files including any new header files used for compilation on Canvas (compressed as one file)?
- Did you implement user's features: password, unique ID, Order Entry, etc.?
- Did you submit a report with your designs (and UML diagrams), evidence of successful testing of all features?