Assignment 2
Software Requirements and High-level Design
Total – 50 pts.

1. (4 pts.) What are the generic activities that take place in the requirements engineering process? What are the inputs/outputs of the RE process? Point out at least two problems you may come across in this process.

Response: Generally speaking, the requirement engineering process is the process of creating the software requirements that all stakeholders and project members understand and agree with. The inputs for requirements engineering are the high-level needs of the client and the output is a description of each requirement as understood by stakeholders and project members as well as various graphs and diagrams used to help each the various members understand the requirements more effectively. Problems that can crop up include stakeholders not actually knowing what they want, stakeholders expressing what they want in their terms which can be misunderstood by project members, different stakeholders can have conflicting requirements, organizational and political factors may influence the system requirements, and requirements can change during the analysis process. The requirement process is involves interviewing clients/stakeholders, running feasibility studies on proposed requirements, discovering various requirements, validating requirements, and generating use case diagrams, ER diagrams, and sequence diagrams.

2. (3 pts.) Describe two different ways that requirements can be verified.

Response: Requirements can be verified through prototyping and reviews with client, as well as the more formal model validation with statistical analysis and test-case generation.

3. (4 pts.) What is main difference between the requirements and the design aspects of a software development project? Describe a certain task that would be done as part of requirements engineering but not during design and another task that would be done as part of design but not during requirements collection.

Response: The main difference between requirements and design aspects of software development is that the requirements process provides a description of what the system should be able to do and the design process should provide an explanation of how the system will do a given requirement. In the requirements process a use-case model would be generated, while in the design process a context model would be developed.

4. (4 pts.) Provide at least two user interface design guidelines for an interactive system (kiosk) that assists airline travelers check-in and print boarding passes by themselves. Why are these guideline important for this particular system?

Response: The two most important user interface design guidelines for a kiosk system for printing out boarding passes would be for it to be Simple and Efficient. Simplicity refers to principle of limiting the interface of the user to only what is truly needed, excess sounds and images could only serve to distract and confuse the user; in a time-sensitive situation that printing

out a boarding pass could be, it needs to be as distraction free as possible. The interface should also be efficient, reducing the amount of user interaction needed to successfully print out a boarding pass, again because of the potential time-sensitivity of printing out boarding passes.

5. (4 pts.) Write a user documentation for the interactive system in question 4. You can pick any form of documentation listed in the handout titled, "User Documentation". Include at least four distinct steps or entries in your documentation.
Response:

Tutorial:

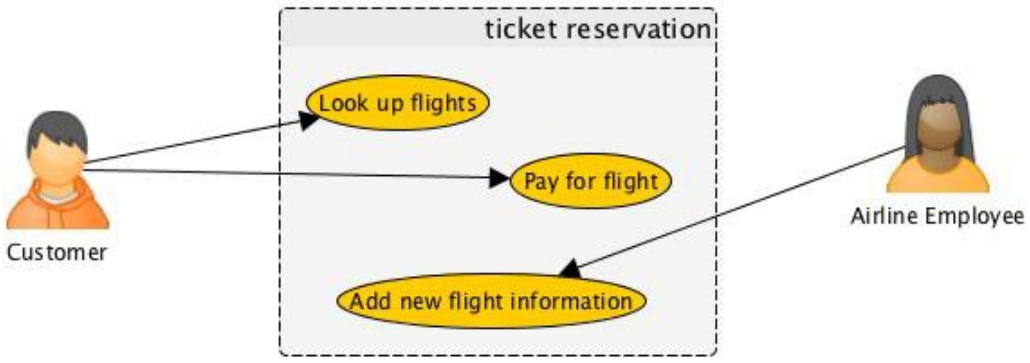
Step 1: From the main screen, select login and enter your user SuperFlyer® number, which will take you to a list of your upcoming flights.

Step 2: From the upcoming flights you have, select the flight you wish to print the boarding pass for, which will take you to the boarding pass confirmation screen.

Step 3: On the boarding pass confirmation screen, review and confirm that the data being displayed is correct, then press the "Print Boarding Pass" button.

Step 4: Collect boarding pass from attached tray, if you need to print out more boarding passes, click the "Return to Passes" button and proceed from Step 2. If you are finished, press logout. If nothing is pressed within 20 seconds, you will be automatically logged out.

6. (20 pts.) Provide the following for an online flight ticket reservation system:
a. A use case diagram with at least two actors and three use cases.
b. A complete scenario for one of the use cases – this should include all necessary details (refer the powerpoint presentation on use cases).
c. One user-level functional requirement (complete sentence).
d. Two or more design-level requirements for the above functional requirement (complete sentence).
e. One verifiable, non-functional requirement (complete sentence).
f. A context model including at least three other systems in the environment.
g. An architectural model with at least four sub-systems.
Response:
a.



b. Pay for Flight

The customer needs to be able to pay for a given ticket to confirm reservation of space on a flight.

Precondition: The customer is logged in with their credentials and the reservation has been added to their cart.

Main flow of events:

1. The customer selects a flight from their cart.
2. The customer clicks “Pay for flight”.
3. The customer fills out billing information.
4. A receipt is displayed and e-mailed to the customer.

Postcondition: The customer's billing account will be billed and the ticket will be reserved for the customer.

Exceptions:

1a. The customer selects a flight that is full.
System will tell customer the flight is full
System will alert admin if there is much interest in a full flight

3a. Card is rejected by authentication server.
System will allow customer to re-enter billing information

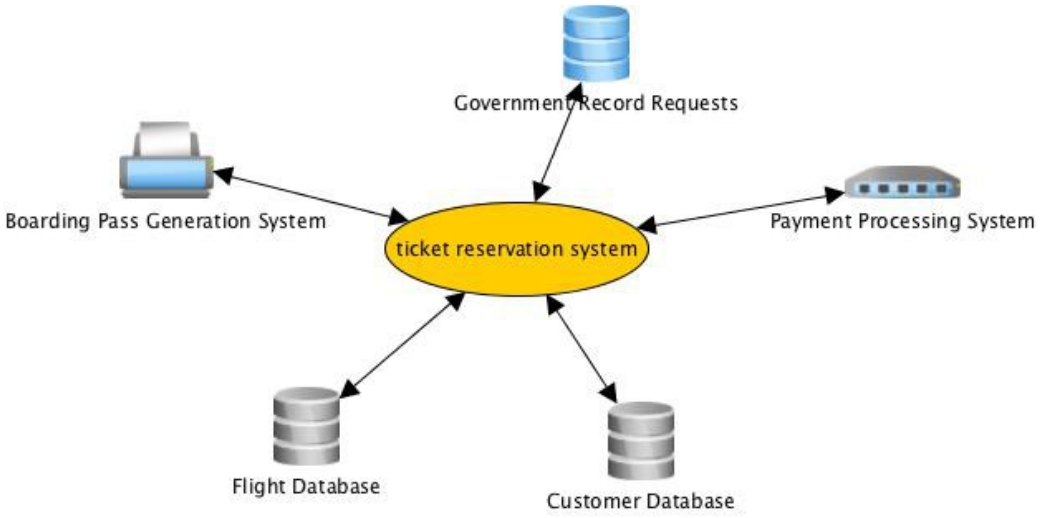
c. The customer needs to be able to pay for a given flight reservation.

d.

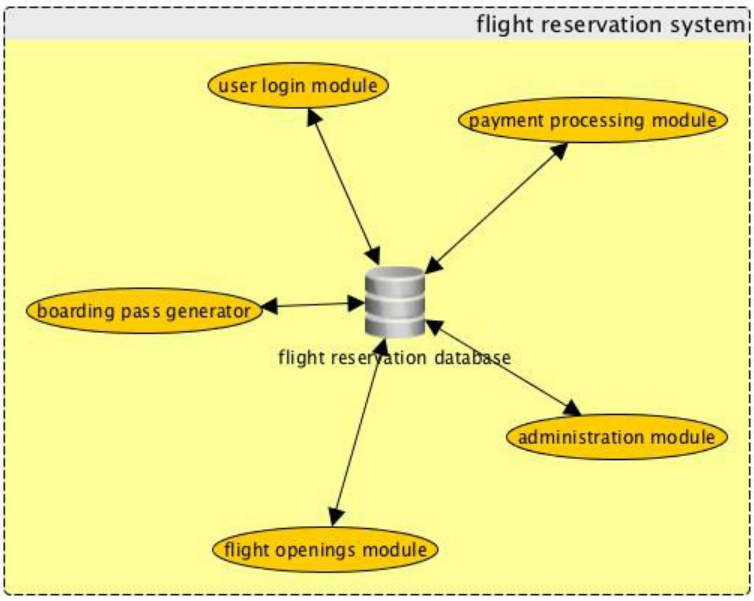
1. When the customer attempts to pay for a flight, the system will verify that the space is available.
2. Credit card information will be validated before reserving a ticket on a flight.

e. A customer should be able to search for and discover if there is space available on a given flight in under 5 seconds.

f.



g.



7. (4 pts.) What are coupling and cohesion? Use the airline reservation system above to provide specific examples of bad coupling and bad cohesion.
Response: Coupling is the degree to which a module is connected to other modules, while cohesion is the degree to which as module performs only the one specific function it was designed to. An example of bad (high) coupling in the airline reservation system would be if something that the payment processing module required to run existed in the user login module, while an example of bad (low) cohesion would be if the user login module also took care processing payments.

8. (3 pts.) What is refactoring? Describe a specific situation in which you would decide to do refactoring in a software development project?
Response: Refactoring is rewriting a part or whole of the underlying code of a system without making changes to the effects of the code, either for the purpose of increased readability in code or for improved efficiency of the system. As you are 75% finished with your project, you realize that everything you've learned since starting the project makes the first work you did is a total mess and should be completely fixed; that is the point that refactoring must be done. Also, if you inherit a pile of code that “works” but there are obvious unneeded clutter in the code, refactoring should be done before building on top of it.

9. (2 pts.) Compare thin and fat clients.
Response: Thin clients are unsophisticated clients that simply pass information back and forth to a server/connection without doing any computation, while fat clients are complex clients that can do local processing before passing on data. PuTTY and ssh are examples of a thin client, while a web browser is a fat client.

10. (2 pts.) What is the system boundary? What is an actor?

Response: The system boundary defines what is in 'scope' of the system, what the system will be composed of and responsible for. An actor is a specific role a person interacting with a system will portray, used in use-case diagrams to convey the different situations various people can find themselves in while interacting with the system.