HW2 Use Case Diagram

Getting started

- Divide your project team into pairs. If your team has an odd number of members, one of the "pairs" may include three people.
- Your submission will be a single file for the whole group, and include each team's pair division (who are on which pair), use cases selected by each pair (please avoid overlapping of selected use cases across pairs):
    - The use-case diagram created by your team (including both pairs).
        - The diagram can be in any format. You can choose among the official toolkit (Rational Rose or Visio); the open sourced toolkit (ArgoUML); or even hand drawing. The purpose is to think thoroughly about your project.
    - Each pair to select one use case, and describe it in fully dressed fashion.

- The whole project team shall discuss and communicate on what use cases shall be included for the project so that the team has a consistent view on the project's requirements. However, each pair should work on their own chosen use case. You should definitely compare notes after you are done with the homework (to ensure that everyone is on the same page).
- Though this assignment is an exercise in getting you to write use cases, it is also really useful in helping you identify the main use cases for your system. Your team should do this assignment properly and use your answers to help you design your system and prioritize what needs to be done. The use cases that you identify will help with your planning game.

What you need to do

- You should do the reading and also read the lecture slides to get an idea of use cases.
- ***Use Case based Requirements*** (attached under */Readings/Requirements Engineering/UseCaseRequirements*) for further details.

Common mistakes

- Not identifying the important actors of a system.
- Choosing actors from inside the system. Use cases capture the frontier between the system and the outside world. (the exception is for video games where actors are the characters from inside the game world)
- Not expressing the use cases as clear actions (use active verbs).
- Focusing too much on the GUI instead of the actual underlying use case that the actors care about.
- Not linking related use cases properly to one another.
- Not describing the use cases clearly. Even though the person reading your homework might be familiar with your project, if you are using a specialized term (for instance in graphs, etc.) it helps to include an appendix/footnote to define what you mean.
- Not following the template for fully dressed use cases that we showed in class. Each header in the template is important and required.
- Not sufficiently identifying the main success scenario and the extensions. We want you to demonstrate that you have spent time thinking deeply about the use cases.
- It's easy to add too many extensions to the use cases. Don't do this. Identify those that are really pertinent and need to be addressed carefully.
- No labels

Tools

- You can either draw the diagrams by hand or use a UML diagraming tool.
- Drawing the diagrams by hand is easier but you will have to make sure you follow the UML notation. Using a tool will help by enforcing part of the notation for you, but you will probably learn better if you draw the diagrams by hand.
- If you choose to submit hand-drawn diagrams, make sure they are very easy to read. If the diagram or your writing is not easy to read, you may be awarded 0 points for your entire homework.
- If you choose to use a UML tool, we suggest the following:
    - draw.io: It is not specifically designed for UML. But it has basic shapes that appear in UML. In addition, it support creating links between shapes with different type of ends, such as arrow and dot. The link will move as the users drag the linked shapes.
    - Visio/Rational Rose, ArgoUML

While laying out your diagram:

- try to avoid making lines cross since this makes it very hard to follow.
- always follow the conventions of UML, e.g., if you want to show inheritance, the preferred way is that the arrow points upwards and not sideways.
- ensure that your text is readable.

Do not autogenerate UML diagrams from your software development IDE. The purpose of this homework is for you to use UML to express the design of the above scenarios conceptually. Autogenerated UML diagrams tend to be literal representations of code with an inappropriate level of detail that simply becomes noise. Directly creating your own diagrams is good practice for the exam as well.