Group Project Assignment

***Important note:** The purpose of the class project is to exercise the skills and knowledge you gain in the class. Don't think of the project as a programming project (although there is programming involved). Think of it as a software engineering project. You will be applying software engineering approaches. Your grade will depend on how well you do with the software engineering. Coding is just a small part of the entire project. In fact, it is possible for a reasonable skillful programmer to hack a solution in a weekend or two, so the programming part of the project shouldn't be your main concern. Once again, you must show evidence of how you have applied software engineering skills and knowledge to the project.*

- You will work on the class project as a member of a self-selected team. Your team should consist of 3 to 5 people.
- The problem statement can be found in the file, “ProjectDescription.”
- Your team will produce five documents and three presentations.
- The documents will be submitted using the Discussion tool for the rest of the class to view. If you want to revise your documents, this may be accomplished using the Discussion tool as well. We prefer you to post your documents in .pdf format.
    - Team charter
    - Vision document
    - Project plan
    - Requirements document
    - Design document
- There will be three presentations to the class during the term. Since we intend this to be an incremental development, we will expect you do deliver the capability in three increments. We call them the skeletal, minimal, and target increments. Your team can use any available tool to create a video of your presentation, the link to which you will then submit using the Discussions tool. The rest of the class will be able to then view your presentations. The presentations will demonstrate working capability that your team has implemented. The demos should be as long as necessary to show that each increment works as required. (Probably no more than 15 minutes for the skeletal and minimal systems). The final target system demo may be a bit longer since we would like to see validation of all of your requirements in this video.
- At the end of each document or video, please include a list of credits – who did what on the deliverable.
- Each of the documents and presentations will be graded on a scale of 1 to 10. The grade is based on quality: correctness and completeness as well as formatting and notation. This amounts to 80 percent of your project grade. Everyone on the team receives the same grade on these documents and presentation. Here is the grading rubric for these documents and presentations:
    - 10 – Exceptional effort. Far exceeds expectations (A+)
    - 9 - Meets expectations (A)
    - 8 – Meets most expectations, but misses some (B)
    - 7 – Misses most expectations (C)
    - 6 – Fail (F)
- 20 percent of your project grade will be based on a peer review. You will have the opportunity to provide feedback on your team partners. (See the rubric in the "TeammateEvaluationForm.pdf" file.) Peer evaluations are due the last week of the course. You will submit a pdf file containing the evaluations tables for each of your teammates. We will average together the feedback from your teammates for this peer evaluation.
- You will have to code and test the project.
    - You may use any programming language(s) you want.
    - You may use any CASE (Computer Aided Software Engineering) tools that are available to you.
- Feel free to use any tools at your team’s disposal for collaboration, document production, and code management.

The project is described in the section, "ProjectDescription pdf."

The development approach will be incremental, with each increment building upon the previous one. There are five documents and three video presentations:

1. Team Charter
2. Vision Document
3. Project Plan
4. Software Requirements Specification (SRS)
5. Skeletal System demo
6. Software Design
7. Minimal System demo
8. Target System demo

The Team Charter

This document describes your team. You should choose a team name and produce a team logo. (Team motto, team flag, team song, secret handshake, etc. are optional.) The team charter document should contain the team name and logo, and biographies of the team members. It would be a good idea to include photos of the members, since we won't be seeing you in person. It might a bit early to decide on these things, but your charter should include a first cut at who is going to do what on the project. Jobs include project manager, lead architect, lead programmer, lead tester, lead SQA (Software Quality Assurance) engineer, lead CM (Configuration Management) engineer, etc. Of course these people just lead. Everybody works. This project divides into several subsystems, so you may want to divide your team into sub-teams, each of which is responsible for one of the subsystems. You might want to decide who will be the editor for each of the deliverables you have to produce. You might also consider who will be doing the presentations. One person can do the whole presentation, or you can use the "tag team" approach.

An important part of your charter should be a description of how the team will make decisions, and how it will handle and resolve conflicts and issues. You should consider how you will determine whether a teammate will be dropped from the team for failure to participate.

Vision Document

- Documents the vision of the product from the stakeholder viewpoint.
- Documents the problem.
- Identifies stakeholder groups and their needs.
- You can modify the template used in the lecture and remove any items that aren't really applicable to this project.
- Feel free to "make stuff up" on this document. You may pretend you are making a real product for real customers. We want you to demonstrate that you know how to write a vision document. This only applies to the vision document. The rest of the documents should reflect the project you are working on in this class.

The Project Plan

Your plan should have at least the following sections:

- Work-breakdown structure (WBS).
- The features of the following four increments:
    - Skeletal — architecture.
    - Minimal — most important functionality.

    - Target — what you can expect to produce in one semester.
    - Dream — what you could do if you had enough time. This will not be a deliverable increment, but rather a wish list of possible future features. If you finish the target system early, you may start working on features from the dream system. (This is one way to achieve the "exceeds expectations" grade on the final project demo.)
- Schedule.
- Deliverables list.
- Risk assessment plan.
- Quality plan.
    - Quality Assurance.
    - Testing.
    - Configuration Management.

Software Requirements Specification (SRS)

- This sets out the requirements for the target system.
- Describe the following:
    - Glossary - define key terms from the problem domain.
    - Software architecture — identify the subsystems. What are the functions of each subsystem? What information does the subsystem encapsulate? What are the interfaces required? This may be a separate document.
    - Document applicable functional requirements with use cases in a use case document
    - Document any functional requirements that are not amenable to use cases in a supplementary specification document.
    - Non-functional requirements and design/implementation constraints shall be documented in a supplementary specification document.
    - GUI "paper prototype" (flows and layouts) for all user interfaces.

The Skeletal System Increment

- This increment validates the architecture. This is the framework upon which you will build your application.
- All subsystems are defined and their requirements are specified in SRS (Software Requirements Specification) document.
- Executable GUI prototypes for all user interfaces.
- All messages between subsystems are defined in the SRS.
- We don't expect much application logic to be included in this increment.
- Stubs and drivers may be needed for testing.
- Demonstration of working architecture may be accomplished by sending chat messages between subsystems.
- From here, the individual subsystems may be developed independently.

Software Design Document

- This is the design of the target system.
- Restrict the focus to the server subsystem: entity classes and essential use cases involving these entity classes.
- Class diagram — show the entity classes and their connections (associations, aggregations, specializations). You don't need to fill in attributes and operations in the boxes in the class diagram. Put this detail in a separate section.
- Using a text format (pseudocode), for each class, document its attributes, operations, and connections to other classes. Include behavior specifications for the operations.

- For each of the major scenarios, document the interactions between objects of these classes using a dynamic model (e.g., sequence or activity diagram).

The Minimal System Increment

- Purpose of the Minimal system:
  - Minimize risk.
  - Prove out the architecture.
  - All remaining requirements are fleshed out.
  - Update the Project Plan.
  - Revise the Software Requirements Specification.
  - Continue the Design.
  - This can serve as a fall-back position in case you can't get the target increment working properly
- Capabilities delivered in minimal increment:
  - All essential functionality of main use cases.
  - The demonstration video should show an actual working application.

The Target System Increment Project Demo

- This is the final increment of capability for the semester.
- It will be demonstrated in the last week of the class.
- Your team should produce a "lessons learned" document as part of the final demo package.
- Peer evaluations are also due the final week of the class.