CSE325 Software Engineering Course Project Grading & Schedule

Project Grade Composed of

Project Plan	10%
Requirements Document	15%
Design Document	15%
Test Cases Document	15%
Review 1	10%
Review 2	10%
Final Project presentation and demonstration	15%
Retrospective	10%
Total	**100%**

Project Schedule

Jan 30	Last Day to Discuss Project Title with faculty and confirm team members
Jan 31	Code Repository Setup
Feb 3	Project Plan due
Feb 11	Requirements Document due
Feb 13	Last Day to Discuss Requirements Document with faculty
Feb 24	Design Document due
Feb 27	Last Day to Discuss Design Document with faculty
Mar 12	Test Plan & Test Case Document due
Mar 24	First Review & Demo
Apr 9	Second Review & Demo
Apr 30	Final Project Presentations and Demonstrations (Submission of Project Document)
May 1	Last Day to work on project (Any improvements)
May 6	Retrospective due

Code Repository

http://github.com

Project Plan

1. Problem Statement
    (a) Clear and well-defined problem statement (one or two sentences)
    (b) How will your project be different from any similar systems? (one or two sentences)
    (c) Be as detailed as you can in 2-4 sentences.
2. Project Objectives
    (a) Clear and well-defined list of project objectives (what things the project will do)
    (b) Be as detailed as you can
    (c) Also mention which team member is going to work on which objective
3. Feasibility Study
    (a) Evaluation of the potential of the proposed project
    (b) Justification that the project will be completed within the timeframe
    (c) Also study the technical & operational feasibility of the project
4. Process Model
    (a) Choose the appropriate process model for your project
    (b) Justify the why you have chosen it
5. Deliverables
    (a) Specify well-defined deliverables. (major results or services that will be produced, specific things the software will do)
    (b) Mention any platform(s) and/or framework(s) which you plan to use.
6. Project Scheduling
    (a) Plan and Schedule the project according to the process model chosen
    (b) Gantt chart & pert chart

Requirements Document **(IEEE format appropriate for your project)**

1. Problem Statement
    (a) Clear and well-defined problem statement (one or two sentences)
    (b) Be as detailed as you can in 2-4 sentences.
2. Background Information

(a) Explain background information about the problem, the domain, and targeted users.
(b) Mention whether there are any applications or systems that are similar to your planned work.
(c) Discuss the limitations of other solutions and how you address each limitation.

3. Stakeholders
4. Requirements
   (a) Divide this section into two subsections, Functional Requirements and Non-Functional Requirements.
   (b) Make sure to include as complete a list of requirements as you can right now.
   (c) Format the requirements into "user stories" (e.g. "As a-----, I would like to -----.")
   (d) Prioritize the requirements and provide unique IDs
   (e) Include use case diagrams
   (f) In the Non-Functional Requirements subsection, please discuss performance and platform requirements such as response time, scalability, usability, security, etc. This can be either in a "user stories" format with detailed explanations or just a discussion for each requirement.
   (g) All requirements should contain enough work for the semester so that each team member will have to spend around 10 hours/week for the project. If not, points will be deducted and the team will be asked to resubmit the requirements document with adequate amount of work.
   (h) Create as many user stories as you would like, it is better to have too many user stories than too few

Design Document **(IEEE Format)**

1. Purpose
   a. Briefly explain the system you are designing and its purpose.
2. Design Outline
   a. Outline your design decisions (for example client-server model), identify the components of your system, and describe the purpose of each component.
   b. Describe the interactions between individual system components.
   c. Include design diagrams that clearly show the high-level structure of your system. (Architectural, Context, Behavioural diagrams, ER diagram)

3. Design Issues
    a. Ensure you spend enough time thinking about the design issues. Only one or two design issues will not be sufficient to get full credit.
    b. Each design issue requires descriptive title, solution options for the issue, and justification of your choice.
    c. You may divide this section into two subsections, Functional Issues and Non-Functional Issues.
4. Design Details
    a. Include class level design of the system (i.e. class diagrams) and be as detailed as you can.
    b. Describe the classes and interactions between the classes.
    c. Add sequence diagrams for different activities in the system, which will be helpful at the later stages of your project.
    d. Include activity diagrams (or state diagrams) and UI mock-ups.
5. Discuss alternate design details

Test Cases

1. The test cases should be written to cover all the requirements
    a. Each test case should have the detailed steps to test the requirement
    b. The precondition and post condition of the tests should be clearly mentioned
    c. The desired output of the test case should be clearly defined
2. Traceability matrix should be created.
3. The test plan should contain which test cases will be executed when according to the process model

Reviews:

1. Version Control
    (a) All team members are required to consistently use version control system.
2. Implementation and Testing
    (a) Your implementation and demo should be based on the Planning Document. Please implement the tasks that you have planned in the document.
    (c) Each team member must present and explain the tasks they have worked on during the duration.

3. Overall Presentation
    (a) Ensure to prepare your demo before the meeting and do a dry-run.
4. Notes
    (a) Please arrange to meet with your day on the suggested meeting day or one day before it.
    (b) If you have some tasks based on machine learning or data mining algorithms, and cannot avoid certain error rates, it is understandable for your application not to always produce correct results. No points will be deducted for the tasks which work reasonably well.

Retrospective

1. What went well?
    (a) You may write this section in sentences and/or list successful user stories and tasks with detailed discussions.
2. What did not go well?
    (a) Include general retrospective review for this sprint.
    (b) Ensure to list ALL unsuccessful user stories and tasks with detailed discussions
3. How should you improve?
    (a) Mention at least two ways to improve your work in the next semester and be as detailed as you can in
    (b) 2-5 sentences..
4. Notes
    (a) For section 1 and 2, only listing out some user stories and tasks without any discussion will not get you full credits.

Sample Docs


IEEEXplore-SRS-template.pdf


SDD_Template.pdf


SQETestCaseSpecificationTemplate.pdf


CSE310_FinalReport.docx