CSC 340 Final Project Rubric

Last Updated: Fall 2019

The final project for this class is worth 60 points of the course total 100 points. These 60 points are broken down into the following categories, with an adjustment for individual contributions as detailed below.

Documentation - 10 points

The following documents are required during submissions:

1. Requirements Documentation: The requirements document should include the system/platform requirements, a list of industry jargon (if any), software requirements, functional requirements, security requirements… etc. Basically all requirements relevant to your project.
2. Wireframe Diagram: A wireframe diagram of the GUI for all pages associated with the application.
3. Use Case Diagram (x 2). Draw a use-case diagram for all interactions with the user and any administrators. For two of the use-cases, create a page with the following: project name, use-case name, the actors, a brief description of the use-case, any pre-conditions, the procedure of the use-case from beginning to end, list any included and extended use-cases, and any post-conditions.
4. Architectural Diagram. A diagram detailing the overall architecture of your system. Connections between views, controllers, models, underlying persistent data storage, and external systems should be diagrammed. I will particularly be looking for accuracy (how well does the diagram represent the actual system you have created) and an understanding of using Connector Classes/Translator Classes to connect to external systems and persistent data.
5. UML Class Diagram. A complete diagram of all classes used in the system. You should include where classes extend other classes, or implement interfaces. I'll be looking for correct visibility markers, property and method declarations.

All documents should be added to the Git Repo under a subfolder labeled "docs." This is a writing intensive course, so you will be graded for proper grammar and spelling. All text and diagrams should be cleanly formatted, easy to read, and submitted in a PDF format. Each document should have the project name, team name, team members, and date clearly labeled.

External API - 5 points
Your project must connect to an external API or web-service. For full credit, you should have a Translator class that connects to an API class. All internal classes must connect to the Translator class and not connect directly to the API class.

Persistent Data Store - 5 points
Your project must save data to a persistent data store. This data store can be anything as simple as a plain text file, or as complex as a SQL database. For full credit, you should have a Translator class that connects to a Persistent Data class. All internal classes must connect to the Translator class and not connect directly to the Persistent Data class.

Coding Style Guide - 10 points
Your project must adhere to the Coding Style Guide posted on the course's Canvas site. You are allowed 3 abuses of the style guide before points will start being deducted. Any mistakes past 3 will result in 1 point off each.

Engineering 20 points
Your project must demonstrate adherence to proper software engineering techniques. Points will be deducted for each violation of the following principles:

1. MVC Architecture - Are view files disconnected from the internal logic? Do controllers pass information on to models? Remember: Dumb Views, Skinny Controllers, Fat Models.
2. Enumeration - Are numbers that define states, record default settings, or those that keep track of common property values or class types enumerated?
3. Encapsulation - Are classes mostly self-contained? Do you limit the number of dependencies on other classes by passing primitive data types when possible?
4. Re-useability - Are individual classes or packages generic enough that they could be re-used for other areas of your project or other projects?
5. Duplicate Code - Is your project free of duplicate code? Would a change in logic or behavior require redundant edits to multiple classes? Have you used base or abstract classes effectively?
6. Segmentation - Have you extracted helper methods from larger methods that can be easily overridden in extended classes? Have you broken down large classes

into smaller helper classes where doing so would make the system easier to understand and edit?
7. Functionality - Does your project function correctly? Is it useable? Is it easily broken? Does it account for user error?
8. Flexibility - How difficult would it be to add a feature? Can you easily identify the areas of edit for doing so?
9. Stability - Does your project run without crashing?
10. Fulfillment of Requirements Doc - Does your project do what you said it would do in your requirements document? Is it complete?

Testing - 10 points

Your project will receive 1 point, up to 10, for each built-in test method that effectively ensures performance of a class or method within a class. Each test case must test all edge cases, 1 error case, and at least 2 normal cases.

Individual Grades

Each project will be graded on its own merit. Grades will then be distributed to each student involved in the project based on their answers to their presentation and the results of their peer review, with a maximum grade being the overall project grade.

For example, Bob is a member of Team Foo. Team Foo's project scores a B+ overall. During final project presentation, Bob can clearly articulate his areas of responsibility. He can answer questions about the design of his system or subsystem, and he can demonstrate a clear understanding of how and why he used the design patterns that he used. In his peer review from his other team members, they indicated that Bob was an integral member of the team, responded to communication from other team members, was helpful and important to the final outcome of the project. Bob receives the maximum grade, which in this example is a B+.

In another example, George is a member of Team Bar. Team Bar's project scored an A+. However, during the defense of Team Bar's project, George mostly remained silent when asked questions directed toward him. He could not articulate how the code worked, nor could he explain why certain classes were written the way they were written. In his peer review, all of his teammates stated that George was largely absent from team meetings, did not work diligently on his assigned tasks, shirked his responsibilities and generally was of no help to the rest of the team. George will receive an F on his final project.

Submission

Your project, and all documentation, should be uploaded to GitHub. On the day of the final presentation, you must email me the link to your GitHub repo. After presentations have concluded, I will clone your git repo to my local machine so that I can review your code and documents in detail. No changes are allowed once presentations begin. Your projects ability to compile and function must be demonstrated during our final exam period. It is your responsibility to have any hardware necessary ready to run your project.