CSCE 361 Capstone Project Design Documents

In this assignment, you will work on a team of 4-5 students to apply the principles and practices you learned in CSCE 361 on a multi-week project. Project choices can be found here: Project Description.docx

Download Project Description.docx

Objectives

Students will:

- Gain experience developing software on a team
- Use the Scrum development process
- Demonstrate good software engineering practices

Instructions

This assignment is to be completed in assigned teams; **no collaboration outside of your teams is permitted**. Twice during the assignment, you will need to complete a peer assessment. Your teamwork grade will be based on the peer assessments and on the git history.

Commit material that you worked on individually under your own name using the defaults that you set. *When (and only when) you commit material that was developed using pair programming, override the default commit author to reflect both authors* so that we can properly credit both authors for their contribution grades. When you override the default commit author list both students' names, and for the email address use a fake email address that is unique to the pair of students by concatenating your Canvas login IDs (the angle brackets around the email address are required):

```
git commit --author="Herbie Husker and Lil Red <hhusker20lred19@dev.null>"
```

You can use this same technique for the rare circumstance in which your partner is briefly unable to commit code themselves:

```
git commit --author="Herbie Husker <herbie@huskers.unl.edu>"
```

Setup

1. Your team will need to create a GitHub repository with a C# .NET project. Each of you should clone the repository to your local computer and import the C# .NET project into your IDE. Create the repo as a PRIVATE repository on GitHub and include the instructor and TAs as collaborators. You will find the GitHub IDs of the instructor and TAs in the syllabus.

2. *If **at any time** your repository is public or has internal visibility then you will receive a 10% penalty. Further, if another student accesses your non-private repository and copies your solution then I will assume that you are complicit in their academic dishonesty.*

Assignment

You and your team have already been assigned your project. The initial requirements for the project are in an earlier document and are also in the README.md file in your repository.

Requirements Elicitation

The requirements are incomplete. This is both accidental and intentional. It is accidental in that, like all requirements, the customer has expectations that they didn't realize think to articulate. You will need to elicit the missing requirements. You may do that by asking questions to clarify vague requirements, by having me "use" a paper prototype to get feedback, or any of the other techniques described in the requirements engineering lesson.

The incompleteness of the requirements is also intentional in that I expect each team to participate in requirements elicitation. *Teams that do not elicit requirements will be responsible for at least one more requirement than what is included in that thread. Please work with your team and flush out the requirements to render a working product with a robust set of features.*

NOTE: A key Scrum practice is that you do not add scope to a sprint once it has started. I will allow you to add to sprint 1 any new requirements discovered by Nov. 8. Any new requirements discovered after April 9 must not be added to sprint 1's backlog.

Development Process

You will use a modified Scrum process.

- You will have two sprints, each lasting nearly 2 weeks.
- You will need to meet with your team over Zoom and conduct a kickoff meeting and sprint 1 planning meeting.
    - You may have an informal kickoff meeting at any time before the due date.
- You will have a sprint 1 review & retrospective and sprint 2 planning meeting.
- Your project retrospective will be part of the preparation for your presentation.

- You will hold scrum meetings on *at least three the other days* for each sprint. You may choose to set up a Zoom meeting or meet through any other medium your team likes.
    - You may hold additional scrum meetings. (Normally you would hold one per day - typically, your standup meeting.)
- You will not need to provide a burndown chart. Instead, during your daily scrums, make a note of which Issues have been closed since the last scrum (as well as any new issues that were opened since the last scrum).
- You will record a short presentation and demonstrate your program. We will provide details at a later date.

Backlogs and Issues

Consider the stated requirements (and the game rules, if applicable) to be the principal source of your product backlog. There will be other sources for your product backlog, such as any additional requirements you come up with to make sure that the product functions reasonably.

In each sprint planning meeting, decide on the goal for the sprint and formulate the sprint backlog.

Following the Scrum practice of self-organizing teams, each team member should assign Issues to themselves. Equally dividing Issues at the start of a sprint is *not* a Scrum practice. Instead, when a team member has no Issues to work on, they assign an Issue (or Issues) to themselves in our case, you can do that through GitHub issues.

Note: it may be that some issues will require more than one team member, but GitHub's Issue Tacker will only let you assign it to one person. That's okay. The purpose of self-assigning Issues is to make sure nobody's duplicating effort. The Issue Tracker will *not* be used to judge contribution levels.

Issue Flow

Your repository has two Milestones (from the web interface left-side menu, select `Issues` -> `Milestones` to see them). When you create an Issue, you can leave it unattached to any Milestone, or you can assign it to a Milestone. When you decide that an Issue is part of your sprint 1 backlog or part of your sprint 2 backlog, attach it to the relevant Milestone (from the web interface, select the Issue, and on the right-side menu there is an option to attach the Issue to a Milestone).

The Milestone page conveniently shows which Issues attached to the Milestone are open and not assigned to anyone, which are open and assigned, and which are closed. This is a convenient way to observe your progress in the absence of a burndown chart. (The Enterprise Edition of GitLab has a built-in tool that creates burndown charts based on Issue status. The Community Edition that UNL uses does not.)

The Issues attached to the `Sprint 1` Milestone and to the `Sprint 2` Milestone will constitute your sprint backlogs.

Adding Additional Issues Mid-Sprint

A key Scrum practice is not to add scope to a sprint after the sprint has begun. If you realize there is something to be done that you hadn't thought of before, it should be placed in a future sprint.

You can, however, add new Issues to the current sprint's backlog in two circumstances:

- If you realize that completing an Issue that is part of the current sprint requires that you first complete another task, you can create an Issue for that other task and place it in the current sprint's backlog. (Please indicate in the Issue's comments that it is a prerequisite for the other Issue.)
- If you have exhausted the current sprint's backlog then you may, only with the professor's concurrence, add scope to the current sprint.

Tests

Use your best judgment when deciding when to create unit tests.

If you find that you need to debug code, I strongly encourage you to create a Visual Studio Unit test demonstrating the bug, and then use the debugger to run the test and step-through its execution to see where the internal values and/or the behavior deviates from what you expected.

Design

Use your best judgment when designing your program. We will look for evidence of applying the design principles covered in class. Specifically, we will look for proper Separation of Concerns, either through MVC or iDesign.

Persistent Data

If you need to store persistent data, you may do so using a CSV file, a JSON file, a SQL Server database (or if you choose to do the project in Java, using either Hibernate or JDBC), or some other reasonable means. If you are using a database and store your authentication credentials in a resources file (such as `resources/database.properties` or `resources/hibernate.cfg.xml`), be sure to place that file is listed in your `resources/.gitignore` file. Also be sure to include a template for us to create our own resources file with our authentication credentials.

External Libraries

If you need external libraries, find them and get them through NuGet. **(Refer lecture slides on NuGet and package managers)**

Model

Place a class diagram in your repository at the end of each sprint. (You may update it more frequently if you wish.) We do not require sequence diagrams nor any other UML diagrams. Using a tool to auto-generate this diagram from your source code is acceptable; a hand-prepared class diagram is also acceptable. The tradeoff is that an auto-generated diagram is guaranteed to be consistent with the code, but a hand-crafted diagram is going to be able to depict the useful information and only the useful information.

If you wish to break your class diagram into multiple diagrams (and if your tool supports this) then this is acceptable if it improves readability/ understandability.

The diagram(s) must be in pdf, jpg, or png format. Before saving the diagram, please arrange the classes so the structure of the system is clear.

Instructions to build and run the program

Add to `README.md` any special instructions we'll need to build and run your program, such as any dependencies that aren't in the source code or the pom.xml file. If we cannot compile and run your program, we cannot grade the functionality.

Miscellanea

- **Copying models or code off of the internet and placing it into your deliverables is completely out of the question.** You may look at examples to understand something you're trying to teach yourself, but *you must indicate these sources in the deliverables.*
- (If you implement a GUI) You may copy artwork from outside sources to include in your project if and only if you have the license to do so. If the artwork is in the public domain or is licensed under a Creative Commons license then you likely do not need to take explicit action to obtain the license. Otherwise, you probably will have to pay for the license (be sure that the license permits redistribution). *Be sure to comply with the terms of the license, whether it's a Creative Commons license or a commercial license, and be sure to indicate the source in the deliverables*.
- **Underperforming team members** If a student is habitually failing to be a contributing member of your team beyond your ability to resolve, you may ask your TA or the professor for help resolving the problem. In extreme circumstances, the professor may remove that student from the team. This can occur only *after* we've discussed your team's dilemma and concurred that no better option is available. A fired team member may complete the project on his/her/their own, or join a group with other fired team members (if any exist).

- **Toxic team members** If a student's behavior is harming the other team members' ability to learn and perform, the professor may remove that student from the team. This can occur only *after* we've discussed your team's dilemma and concurred that no better option is available. A fired team member may complete the project on his/her own, or join a group with other fired team members (if any exist). If necessary, the case will be referred to Student Affairs for appropriate action.

Deliverables

- MVC / iDesign Design document clearly indicating interfaces, classes and methods. This should be a detailed architecture diagram of your final product.
- Link to GitHub repo
- Database Design document clearly indicating the database schema and information on primary/foreign key relationships between database entities.
- Updated `README.md` (as necessary)

COMBINE ALL THE 4 ABOVE DOCUMENTS INTO ONE SINGLE PDF DOCUMENT AND UPLOAD IT TO THIS ASSIGNMENT. ONE SUBMISSION PER GROUP.

*It is your responsibility to ensure that your work is in the **correct repository** and that we can access the repository at the **time each deliverable is due**. We will grade what we can retrieve from the repository at the time it is due. Any work that is not in the correct repository, or that we cannot access, will not be graded.*