C346 Project

Table of Contents

For the duration of the semester, the class will be participating in group projects as a way of researching new technologies and methodologies for developing software. The goal of this project is to develop a piece of software the same way any real, collaborative software project would be developed. To this end, you will present a project plan, assign yourselves roles within your groups, develop with source control, and present your work as a group.

You will have a public Git repository assigned to you at the IUS GitHub Organization. All work will be stored in this repository. You may fork the repository elsewhere, but the original repository will be graded.

Here are sample projects that did good work in the past:

- IUS-CS Github Full of recent projects
- Older projects (Requires IU login):
    - C346-Roguelike: Legend of *Cheryl
    - TeamROM
    - MiddleOne
    - ID-40k
    - Magma

Structure

You will submit a project survey as a group. This survey will have room for multiple project ideas in case a fallback is needed. Feedback will be given for each project to ensure that the project fits the scope of the class.

The project will be completed in multiple "sprints" that subdivide effort. The first sprint will be a prototype, and each sprint further will show some new amount of work added to the previous sprint.

Proposal/Project management

In the proposal packet, you will be deciding on your group of 3 (possibly 4) group members and a project goal. Keep the size of your group and the amount of time left in the semester in mind when selecting a project. A good project should not be a demo, but have some usefulness.

Your proposal should be in its own document under a "docs" directory in your repository.

Estimations

In addition to your project proposal, you should include a broken down list of high level goals that describe the work you will be doing during the project. Each goal should have an estimate attached. You may choose to estimate your goals with Agile story poirts (1,2,3,5,8,13,20) or just attach your estimated hours.

Your estimations will be living documentation in the Projects tab of your GitHub repository. You will update this as you progress. For example,

Task	Estimate (points)	Actual Effort (hours)	Developer
Write good software	8	10	Chris
Document bad software	2	3	Phil
Create database	1	1	Kyle
Write templates	2	5	Chris
Test	1	20	Kyle

This example table has several high-level goals estimated in story points and actual time spent attached with who completed the task.

> Your "estimation" and project planning should be on the external tool, GitHub Projects (a tab at the top of your repository page). Be sure to track people, tasks, and time.

Organization/Roles

The final document needed before starting your project is an organizational document. This will detail how much effort each group member is going to put towards each task. Record what roles each group member will have within your group. For example, you may choose to have one developer, one tester, and one documenter. You may also choose to split all tasks evenly amongst your group. Any configuration is fine for this task as long as every group member agrees.

> Each group member should have a stake in the project. Group grades are *not* assigned as a whole, but for individual members. It's okay if a particular member has more work than others, but all must participate.

Work out who is going to do what and store it in an "organization" document in your "docs" directory.

Code

As stated, this project should teach you something new. If you do not find yourself searching for answers and using a new technology, then you may have chosen too small a project. Each group member will commit their code and documentation to the group's Git repository, so a log of who has done what is public. Code will be scored on readability, embedded documentation (comments), and effort.

Testing

Well tested code is essential for any project. You will adopt a testing framework and unit test your code. There should be many unit tests. It may help to adopt a Test Driven Development plan by writing tests before you write sections of code. **YOU WILL NOT BE ABLE TO DO ALL OF YOUR TESTING LAST MINUTE.** If any functionality is untested, it should be recorded in a document with a solid reason for being skipped. If manual testing is necessary (for a UI), then those steps should be recorded in the same document. Manual testing is not a replacement for unit tests.

Create a "testing" document in the "docs" directory to record these issues.

YOU WILL NOT BE ABLE TO DO ADEQUATE TESTING LAST MINUTE. Think about testing before, during, and after writing source code. Set up a testing framework before writing source code.

Documentation

Documentation on how the program functions, how to use the program, and what limitations the program has should be provided. This can be informal, but generated API docs would be great as well. There are many API doc generators for various languages including:

- JSDoc, YUIDoc, doxx, and docco for Javascript (Link about these generators)
- JavaDoc for Java
- Sandcastle for Microsoft projects
- Pydoc for python
- Doxygen for general documentation (supports all languages above and more)

Create documents in the "docs" directory to satisfy this requirement. Create an "api" directory for any generated API documentation.

Presentation

Finally, show your project off! The last few days of class will be dedicated to you showing your projects to the class. The presentation should include a demonstration of your project, an expose of your code and tests, and a talk on why you chose to do this project along with any stories you've made along the way. You will each have a form to rate your peers and the instructor will take these into account when tallying up final grades.

Grading

Each item of project documents, code, testing, documentation, and presentation will contribute to the overall score. Note that code is only part of this grade. Be sure to contribute adequate resources to the other parts of the project to ensure a good score. The percentages here are rough estimates of weight which will be scored over all of the sprints and final presentation. Keep in mind, particular sprints may not focus on a given aspect of the project.

- 20% Project documents
    - Should be legible and show the timeline of your project. If your project document is empty, I will assume that you have not been working on the project.
    - Should be in markdown format if stored in GitHub, or a link to documentation if using some other tool like a wiki, project management suite, or other tool.
- 30% Code
    - Should be legible and formatted according to the idiomatic style of the tools used.
    - Should be well documented. (Hint: this makes the documentation section much easier!)
    - There should be a signifigant effort here. Projects with little source code will get little credit for this category.
- 15% Testing
    - All code written must be tested in some way.
    - Automated testing is manditory.
        - Anything that cannot be automated must have an explanation about why this is the case.
        - Anything that cannot be automated must have a testing plan written in asciidoc in the "docs" directory.
- 15% Documentation
    - At a minimum, API documents are simple to generate. Please ensure that you have documentation.

    - Other forms of documentation are encouraged as well. Write a manual! Make a tutorial! Etc!
- 20% Presentation
    - Presentation guidelines will be given near the end of the semester.
    - Presentations should be concise.
    - All presentations must include a technical demo.
        - If you are creating a web site, ensure that you have a way to host that site before the demo.
        - If you are going to present another way, ensure that your laptop works with the projector or your project can run on the instructor's machine.

Table 1. Rubric

Area	Subarea	Points	Notes
Documents		40	
	Formatting	8	Legability and english matter!
	Readme	8	Should include a summary and setup/build instructions
	Proposal	8	Detailed description of project and goals
	Estimation	8	Detailed and trackable
	Organization	8	Percentages match actual work tracked in estimation and git
Code		60	
	Reasonable size	20	Expect to write a few thousand lines of code

	Formatting	20	Use a linter, write idiomatic code
	Best practices	20	Again, write idiomatic code
Testing		30	
	Coverage	10	Testing : Code ratio should be 1:1
	Automation	10	Testing should be automated
	Best practices	10	Is injection used properly? Are tests well designed?
Documentation		30	
	Minimum API docs	10	Percentage of methods documented
	Build/install instructions	10	Documents describing use of the software external to the code
	Formatting	10	Professionalism of the documentation
Presentation		40	
	Description	10	Do we understand what the project is?
	Demo	10	Is there a live demonstration?
	Testing	10	Can tests be run, can we see the testing effort
	Project Management	10	Detailed description of the project organization

	Reflection	10	Post-project thoughts and learning experiences
Total		200	

This project represents the bulk of the work to be turned in during this semester. IUS guidelines state that you can expect to spend around 3 hours outside of class for every 1 hour in class. We meet for 4 hours per week. **A project that does not show due effort will not receive a passing score.**