PJ05 - Handout

Description

Project 5 continues the work you began in Project 4. If there are features that you were not able to complete by submission last time, you will have the opportunity to work on them here.

This project is worth 15% of your final grade. It is your final project and the capstone of your work in CS 18000!

The overall functionality requirements for Project 5 are the same as those of Project 4. However, there are three key differences:

- Project 5 requires a concurrent solution. That is, multiple users must be able to access the application at once.
- Project 5 also requires network IO. Users should not be limited to just one machine.
- Lastly, Project 5 must have a full GUI. Users must be able to interact with the application and utilize all the required features via the GUI.

Updating your Project 4 solution to meet these new requirements is your goal. At the same time, you may continue to develop optional features from Project 4 if you like.

This project is manually graded.

Note: 5 points of your grade is based on Coding Style. You will need to update the Starter Code to follow the standards described on Brightspace. Use the "Run" button to check your Coding Style without using a submission.

Team Work Expectations

There will be 3, 4, or 5 team members on each team. You will continue with the same team members from Project 4. We expect each member of every team to contribute to the project. You are permitted to divide the work in any way you see fit as long as responsibilities are evenly distributed and every team member contributes to the project source code. You will be required to document your contributions to the project in the final report.

We recommend using the Campuswire chat functionality to communicate with your team. Using this feature will make it easier for course staff to support your team (you can ask us to answer questions via the @ function) and helps us mediate any potential disputes.

Any team member who fails to contribute will receive a 0 on the project.

Be aware that team collaboration is limited to the members of your team. You should not be sharing code with individuals outside of your team. Remember to follow the course Academic Honesty policy. If you have concerns about whether or not something is okay, just ask us.

To simplify collaboration, you must make use of a Code Repository on Gitlab or Github. It will make sharing code, tracking changes, and debugging significantly easier. However, keep in mind that any repository you use must be private, with access limited only to members of your team. Code made available publicly is academic dishonesty. You will be required to submit a copy of your repository on Vocareum by cloning it into your work folder.

Note: Every team member must commit to the repository. A lack of commits may be used as evidence that a team member did not participate.

We reserve the right to modify your grade based on participation. You may receive a 0 for not contributing at all, or you may receive a 75% deduction on your team score for only contributing superficial content. If you wait until the last minute to work on the assignment, you will receive a 0 or a significant deduction as well. These cases will be judged on a case-by-case basis using evidence provided by teams.

Your team will share a workspace on Vocareum. Only one team member needs to submit.

In the event of a team disagreement, dispute, or lack of participation from any individual, you should contact the course coordinators as soon as possible. We can only help if we are aware of the situation.

Remember, this is a team project. Your project score will reflect your contribution to the team.

Implementation Restrictions

Before describing each of the options, we want to note several key restrictions on your project implementations.

- First, the use of the Intellij GUI Builder is not permitted for this project. That is, .form files present in your submission will immediately result in a 0 for the assignment. You must implement the GUI using the techniques described during lecture.
- Second, no data may be stored client-side. Additionally, all application data must persist even if there's a complete shutdown of the server.
- Third, we define real-time updates as those that occur automatically, without any action from the user. Requiring users to select a refresh button or navigate to a different page before updating content is not permitted.

All previously described restrictions from Project 4 are still in effect.

Keep these in mind while designing your solution!

Implementation Requirements

- The application must support simultaneous use by multiple users over a network. New content should appear in real-time as users add it.
- All user interactions must be GUI based.
    - Note: You are NOT permitted to keep the the keyboard (System.in) and the screen (System.out) implementation in your solution.

    - Functionality for the GUI must meet or exceed the functionality for the the keyboard (System.in) and the screen (System.out) interface implemented in Project 4.
- Data must persist regardless of whether or not a user is connected. If a user disconnects and reconnects, their data should still be present.
- Descriptive errors should appear as appropriate. The application should not crash under any circumstances.

These implementation requirements are in addition to those listed for Project 4.

Testing

There are no public test cases for this project. Each implementation will look different, and we do not want to restrict your creativity in any way.

However, you are expected to write your own custom test cases, specific to your team's implementation. You will expand upon the tests you created for Project 4. However, these tests will be in a different form.

Create a file called Tests.md in your repository. In it, create and document test cases that simulate user interactions with your application. You can use the following format:

Test 1: User log in

Steps:

1. User launches application.
2. User selects the username textbox.
3. User enters username via the keyboard.
4. User selects the password textbox.
5. User selects the "Log in" button.

Expected result: Application verifies the user's username and password and loads their homepage automatically.

Test Status: Passed.

You can customize the template to your needs, as long as the basic structure is maintained. Each test should be numbered.

Note: In addition to designing and documenting the tests, you need to actually perform them and verify your implementation works as expected. If your grader reviewing the application cannot match your results, you will lose points.

Presentation

Your team will record and submit a video presentation as part of this project. The requirements are as follows:

- Each team member must actively contribute to the presentation for at least two minutes.
- The overall presentation will be a minimum of 10 minutes and a maximum of 15 minutes.
- The presentation must include the following:
    - An overall pitch for the project (as in, try to sell the product to a prospective client).
    - A demo of the project's required functionality.
    - A demo of any optional features.

- An explanation of the testing done to ensure the project works reliably.
- An audio-visual component (Powerpoint, Google Slides, Prezi, movie, etc.)
- 2 minutes of time allotted to an FAQ section of questions about the project and your implementation. Answer 2-3 questions a recruiter or business manager would be interested in.

Nearly every CS job interview (and many actual jobs) involve presentations along these lines. Use this opportunity to practice the skills you will need in interviews for internships and post-graduation positions.

Report

In addition to your presentation, you must submit a project report. There will be two parts.

Part One

Part One is a minimum of 1000 words and requires the following:

- A minimum of 500 words describing your project and the functionality you implemented. Include both required and option features, along with descriptions of each.
- A minimum of 500 words documenting your design choices while implementing the project. Justify the project structure.

Part Two

Each team member must write a minimum of 350 words on the following:

- A minimum of 200 words describing your contributions to the project.
- A minimum of 150 words on what you would do differently, if anything, if you were given the opportunity to start over again. If you would not do anything differently, explain why.
- Each team member's section should be labelled with that individual's name.

Note: Be sure your document does not have any major grammar or structural errors. You are not required to adhere to any writing style guide (for example, APA), but your document should be organized following the guidelines listed above. You can use the following structure for a general outline:

- A cover page with the report title and all team member names listed
- A section labelled "Part 1" on a new page with the information described above (including as many pages as necessary).
- A section labelled "Part 2" on a new page with the information described above (including as many pages as necessary).
- We recommend using a 12 point font such as Times New Roman and double spacing. All section and subsection labels should be bolded.

Documentation

Your project must follow Coding Style (it will be 5 points of your solution grade). You should also document your code with comments explaining the functionality you implement. Not only will your teammates appreciate it if they have to debug, but you will also make it simpler to explain why you chose to implement it that way if asked during the presentation.

Additionally, your project will need to have a README document. This document will include the following:

- Instructions on how to compile and run your project.
- A list of who submitted which parts of the assignment on Brightspace and Vocareum.
    - For example: Student 1 - Submitted Report on Brightspace. Student 2 - Submitted Vocareum workspace.
- A detailed description of each class. This should include the functionality included in the class and its relationship to other classes in the project.

Submit

Here's a breakdown of grading:

- All solution code, test cases, and documentation must be submitted on Vocareum by cloning the repository. (60 points). All required files should be included in the repository.
- A written report must be submitted via Brightspace. (10 points).
- A ten minute, pre-recorded video presenting your work (20 points).
- The peer evaluation form must be submitted via CATME. (5 points).
- Coding Style (5 points)

Note: The three fundamental aspects of this project are concurrency, server-client interactions, and a GUI interface. Solutions that are lacking any of these features will not receive credit.