CST8002 Programming Language Research Project

# Practical Project Part 02 – Project Review I – See Brightspace for due date

* Refer to the Weekly Schedule document posted in Brightspace under Course Information for additional requirements common to all assessments as well as details on the required use of the data set specified for the course.
* Make reference to the documents in the dataset content area in Brightspace for the dataset to use, note that all columns need to be used unless your professor has indicated otherwise, the dataset column names must be used in your source code as variable names as well to verify you are using the dataset provided.
* The first record in the data set may contain the column names, if so, you may skip over this record when reading in and parsing the data set data.
* You may use your previous work in this course as a reference / starting-point but I expect modifications, i.e. passing in older work again with none to very small changes will not earn marks.
* **Important: Many previous students have found this to be the most challenging activity in the course, budget your time and start work early.**

## Tasks

### Version Control

* Using your GitHub account:
  + If you have not done so already create a private remote repository for your project development over the semester, give the repository a name like “CST8002\_PracticalProject\_Section\_YourName”
    - The section would be your lab section, e.g. 350, 360 and so on.
    - YourName would be your name as it appears in ACSIS or in Brightspace.
  + If you have not done so already, invite your professor to your repository as a collaborator, so they can review your commit history as well as clone your repository for marking purposes. See the contact information area in Brightspace for your professor’s name and email address.
* As you work on your project you are expected to follow an iterative development process, making commits that have brief, but detailed comments explaining what program feature or functionality was achieved. These will correspond to tasks / requirements below.
* For **Practical Project 2 tag your commits with “v2.0”**, see [1] the resource at the end of this document on tagging.

### Program Functionality

* Refactor or re-use your project started in Practical Project Part 1 to meet the following requirements:
  + Project has a layered design and implementation e.g. Presentation, Business, and Persistence with record-objects (also known as Model or Entity objects) or uses a Model View Controller framework. For example, all user interactions are via the Presentation Layer, the sequential data structure in memory is within the Business Layer, and the File-IO is within the Persistence Layer.
    - Use a separate commit for each part of your layered architecture, or each part of your MVC pattern as you develop them. The commit should briefly describe what part was created.
    - E.g., one commit for the model, one commit for the business layer (problem domain), one for the presentation layer etc.
  + Re-use your record object, or create one, (also known as entity object, data-transfer object) that uses the column names from the dataset as part of the source code, e.g. variable names, accessors/mutators names, or constants.
    - Make sure you use a commit for this part.
  + Use File-IO on startup to open and read the dataset, initializing one hundred record objects with data parsed from the first one hundred records in the csv file. If there are fewer than 100 records in the dataset, use them all. The record objects should be stored in a simple data structure (array or a list), use exception handling in case the file is missing or not available.
    - Use a commit after your file-IO logic is developed or updated.
  + Displays your full name on screen so it always remains visible, or after each user interaction.
  + Provide the user the interactive options and functionality to:
    - Reload the data from the dataset, replacing the in-memory data.
      * Create a commit when this task is completed.
    - Persist the data from memory to the disk as a comma-separated file, writing to a new file. Research using a GUID or UUID using an API to generate the file name for the output file. See [2].
      * Create a commit when this task is completed.
    - Select and display either one record or display multiple records from the in-memory data.
      * Create a commit when this task is completed.
    - Create a new record and store it in the simple data structure in memory
      * Create a commit when this task is completed.
    - Select and edit a record held in the simple data structure in memory
      * Create a commit when this task is completed.
    - Select and delete a record from the simple data structure in memory
      * Create a commit when this task is completed.
* Take a screen shot of your program performing each task above, ensuring your full name is within each screen shot. E.g., print “Program by Your Name” replacing Your Name with your ACSIS name every 10 records of output when displaying many records and / or as part of the menu system (or GUI or Web Page etc.)
* Write a single unit-test as proof of concept using a testing framework to test one part of your program. You should use a unit-testing framework with asserts or the equivalent.
  + Unit-Test Examples (you would only do one test, or a similar test):
* Does the program read in records, placing data into correct fields of record objects?
* Does the program add a new record into the sequential data structure?
* Does the program update a record in the sequential data structure as expected?
* Does the program remove a record from the sequential data structure as expected?
* Does the program catch any exceptions or errors if the file is missing?
* Etc.
  + Create a commit when this task is completed.
* Comment your source code file, and unit test file, using documentation comments (docstrings in Python, XML-document in C# or VB.Net, JSDoc for server-side JavaScript etc.)
  + Create a commit when this task is completed.
* Your program should use the following programming concepts: variable, methods, a loop structure, a decision structure, File-IO reading from the dataset, File-IO writing a csv file using GUID or UUID as file name, exception handling, use of an API library, an array (or similar data structure), unit testing, N-Layered or MVC architecture.
* Document your learning, demonstrate that your program runs and is unit tested with screen shots. (See detailed instructions below)

**Warning**

* **Your project must use separate source code files for each layer or part of the MVC.**
* **Organize your project with sub-folders having names similar to ‘presentation’, ‘business’, ‘persistence’, ‘model’ or ‘model’, ‘view’, ‘controller’ placing each source code file(s) in the appropriate sub-folder.**
* **If your programming language is Object Oriented or supports Object Oriented technology, you must use an Object-Oriented approach for this project.**
* **Simplistic projects with no package (sub-folder) structure, and / or a single source code file will not be awarded marks. i.e., you are expected to structure your projects code correctly.**

## Your single MS Word document for your Practical Project 2 report should have this general format

* Cover page with your full name within it, the content of your report should start on page 2.
* Heading with name “Evidence of Learning”
* Sub-headings:
  + variable, methods, a loop structure, a decision structure, File-IO reading from the dataset, File-IO writing a csv file using GUID or UUID as file name, exception handling, use of an API library, an array (or similar data structure), unit testing.
  + Then Either:
    - Use small code examples for each topic taken from your project with brief descriptive text, or
    - Indicate what line numbers in a larger code sample (including the file name) illustrate each concept within brief descriptive text (line numbers must be present in the code examples as well to match).
    - **You must indicate clearly to your professor that you can identify what parts of your code illustrate and match each programming concept.**
* Heading with name “Program Architecture”
  + Briefly outline how you organized your code files or classes into layers or similar architectural design.
  + Provide a UML package diagram demonstrating layers or MVC, UML class diagrams, and one UML sequence diagram for one program feature (e.g. saving to file). See [3], [4], [5], [6].

Note that a single source code file, with methods, is not a layered design. The program logic needs to be broken out into separate files / classes which are organized into source-code folders corresponding to the layers or parts of the MVC. See the example provided on Brightspace as part of your Hybrid Activity 03.

**A simple program that does not meet the architecture requirements will not earn full marks.**

* Heading with name “Program Demonstration via Screen Shots”
  + Include screen shots of your running program; I should see records from the data set displayed, as well as creating, updating, deleting records and re-loading and saving records via user interaction with the program. Your full name must appear within the screen shots.
* Heading with name “Unit Testing Demonstration via Screen Shots”
  + Include screen shot(s) of your running unit test(s), this may either be within an IDE or run from a console. Your full name must appear within the screen shot(s).
  + The screen shot must show a unit testing framework in use.
* Heading with name “Source Code Commenting Example”
  + Copy and paste all of the source code, including programmer comments, from one source code file from your project to demonstrate you can write **documentation-comments**. Use a font size of 10 point, with a monospaced font of your choosing.
  + Copy and paste your unit test source code file, including programmer comments into this section. Use a font size of 10 point, with a monospaced font of your choosing.
  + Note: Some frameworks generate many code files, which you never edit yourself. Only include a source code file, and unit test file, you created or edited directly.
* Do not copy and paste code from the web or AI agents into your demonstration program, it must be your own work. In other words, even properly cited and referenced code copied from a website will not earn marks, as you must provide your own work. Your professor while grading your submission will make this determination.

## Demonstration in Lab Period

* The week following the due date, you will be expected to demonstrate your program to your lab professor in the lab period.
* You will be asked to run the program to show that it works, then will be asked to answer two questions. One based on theory, and one related to making source code modification.
* Sample Theory Questions
  + Can you indicate what parts of your project correspond to the layered architecture or MVC design pattern parts?
  + Can you show where you use GUID or UUID to generate a unique filename for writing to file?
  + Can you overview how your unit testing code tests a program feature?
  + Similar questions on theory knowledge are possible at the discretion of the lab professor.
* Sample Code Modification Questions
  + Can you modify your program so that instead of using a(n) UUID or GUID it uses a different file name for writing to file?
  + Can you modify the order of inputs for creating a new file / updating a file?
  + Similar questions on modifying the code are possible at the discretion of the lab professor.

## Submission Requirements

* Upload your MS Word document by the due date to Brightspace
* Provide a plain-text link to your project GitHub repository in the comment section of your Brightspace submission. You should have already invited your professor as a collaborator when you started working on this assessment.
* Demonstrate your practical project to your lab professor in the lab period following the submission due date.
* **While the demonstration carries no direct marks, missing the demonstration will result in a zero for this assignment. Additionally, if you cannot answer simple questions to demonstrate concepts in this assignment to your lab professor you will lose up to 100% of the score on this assignment. Please refer to the Course Outline under Department Information.**

## Grading (Total 21 Points)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Criteria | Poor/Missing (0) | Below Expectations (1) | Meets Expectations (2) | Exceeds Expectations (3) |
| Evidence of Learning | Poor/Missing or program does not use data set column names as specified within the source code or program is not using correct data set. Program is not organized into separate source code files, and / or the source code files are not appropriately placed into sub-folders (packages) named after layers or MVC parts. | Many learning topics are missing, or the presentation is vague i.e. student does not indicate clearly at all what parts of their code match each learning topic. | Almost all of the learning topics are present in code samples and clearly indicated with brief explanatory write up. | Student documents that all learning topics are in use by presenting either small focused examples with brief explanatory write up or a large code sample with visible line numbers used to focus the discussion of program topics. |
| Program Architecture | Poor / Missing or not separated out into file-based layers or MVC. E.g. A program consisting of a single code file with methods or program is not using correct data set. Program is not organized into separate source code files, and / or the source code files are not appropriately placed into sub-folders (packages) named after layers or MVC parts. | Student vaguely mentions layers/MVC and/or program may not have at least 3 layers or all parts of MVC. No diagrams provided. | Student outlines the architecture with each file or class’s placement in a layer or as part of MVC indicated. Diagrams show a high-level view of the architecture, however not all diagrams were provided or are correct. | Student outlines and discusses the layered architecture or MVC in brief. Diagrams show the file / package structure and additional diagram(s) show components like classes. Sequence diagram is used to show communication between layers for one program feature e.g. user requests to save records to hard drive. Diagrams are correct. |
| Screen Shots, Running Program | Poor/Missing  E.g. missing full name from all images, image file not within the MS Word document. Screen shot does not demonstrate use of the dataset or program is not using correct data set. Program is not organized into separate source code files, and / or the source code files are not appropriately placed into sub-folders (packages) named after layers or MVC parts. | Screen shots are within MS Word document. No explanation of the image, headings or subheadings at most. Has partial name or nickname or but full name missing from some screen shots. Some project functionality missing or program crashes. | Screen shots are within MS Word document. Student provides a brief description of most images using one to two sentences. Has full name in nearly all screen shots. Most of the requested project functionality is working. | Screen shots are within MS Word document. Student provides brief yet detailed description of each image using one to two sentences. Has full name in nearly all screen shots. All project functionality is documented and working. |
| Screen Shots, Running Unit Test(s) | Poor/Missing  E.g. missing full name within images, image file not within the MS Word document. Screen shot does not demonstrate unit testing with a framework or program is not using correct data set. Program is not organized into separate source code files, and / or the source code files are not appropriately placed into sub-folders (packages) named after layers or MVC parts. | Screen shots are within MS Word document. No explanation of the image or only heading or subheading. Has partial name or nickname but full name missing from some screen shots. Test framework is run but does not actually test anything meaningful or crashes. | Screen shots are within MS Word document. Student provides brief generalized description of most images using one to two sentences. Has full name in nearly all screen shots. Test framework runs and performs a basic or rudimentary test but might not test a program feature. | Screen shots are within MS Word document. Student provides brief yet detailed description of each image using one to two sentences. Has full name in nearly all screen shots. Test framework runs test and tests at least one part of the program’s functionality. |
| Source Code, programmer comments. | Poor/Missing  E.g. missing full name as programmer comment at the top of the file as author of the file or program is not using correct data set. Program is not organized into separate source code files, and / or the source code files are not appropriately placed into sub-folders (packages) named after layers or MVC parts. | Student uses minimal comments in source code, e.g. the student does not comment (m)any class members. | Student comments some class and class members, however does not use documentation comments. | Student uses documentation comments in an accepted coding style specific to their language of study. If the language does not support documentation comments student provides evidence of this from reputable source, yet still comments code following best practices. |
| Version Control | Poor / Missing or there are no incremental commits in the commit history or program is not using correct data set. Program is not organized into separate source code files, and / or the source code files are not appropriately placed into sub-folders (packages) named after layers or MVC parts. | Student project development shows the use of a few commits; however, commits do not have a detailed description or no description. Not all the requested commits are present. | Student project has regular commits providing evidence of incremental builds, not all commits briefly overview what was changed. Missing at most one of the requested commits. | Student project has regular commits providing evidence of incremental builds, each commit details what was changed. Requested commits are present. |
| Source Code and Source Code files | Poor / Missing or program is not using correct data set. The professor was not given access to the remote GitHub repository or there is no repository. Program is not organized into separate source code files, and / or the source code files are not appropriately placed into sub-folders (packages) named after layers or MVC parts. | Not used for this criteria | Not used for this criteria | Student uploads both the MS Word document and provides a link to the remote repository. Professor was granted access to repository as a collaborator. |

# Recommended Resources / Sources Cited

[1] 2.6 Git Basics – Tagging. (n.d.). git-scm.com. [online] Available at: <https://git-scm.com/book/en/v2/Git-Basics-Tagging> [Accessed on Nov 10, 2024]

[2] Ronaldo Oliveira. (Jul 31, 2024). GUID vs UUID vs ULID: Understanding Unique Identifiers. medium.com. [online] Available at: <https://medium.com/@ronaldo.oliver7/guid-vs-uuid-vs-ulid-understanding-unique-identifiers-565c88cdca13> [Accessed on Nov 15, 2024]

[3] Kirill Fakhroutdinov. (2009-2024). UML Package Diagrams Overview. uml-diagrams.org. [online] Available at: <https://www.uml-diagrams.org/package-diagrams-overview.html> [Accessed on Nov 15, 2024]

[4] Kirill Fakhroutdinov. (2009-2024). Multi-Layered Application: UML Model Diagram Example. uml-diagrams.org. [online] Available at: <https://www.uml-diagrams.org/multi-layered-application-uml-model-diagram-example.html> [Accessed on Nov 15, 2024]

[5] Kirill Fakhroutdinov. (2009-2024). UML Class and Object Diagrams Overview. uml-diagrams.org. [online] Available at: <https://www.uml-diagrams.org/class-diagrams-overview.html> [Accessed on Nov 15, 2024]

- Use the links at the top to review multiple class diagram types.

[6] Kirill Fakhroutdinov. (2009-2024). UML Sequence Diagrams. uml-diagrams.org. [online] Available at: <https://www.uml-diagrams.org/sequence-diagrams.html> [Accessed on Nov 15, 2024]

Download GitHub Desktop. (n.d.). desktop.github.com. [online] Available at: <https://desktop.github.com/download/> [Accessed on Nov 10, 2024]

## Additional Notes

Video Game Software projects are not acceptable in this course.

Your source code within the MS Word document, should match the code in your source code files in your repository, this includes the programmer comments. If there are large or many differences, you will lose marks.

If you do not provide your GitHub link and include your professor as a collaborator so they can review your commit history and source code files you will receive a score of zero for this assessment.

Not using the correct data set for your course section will result in a zero for this assessment, note that a Canadian Cheeses dataset if present was for demonstration purposes only and is not the correct data set for your project.