CST8333 Programming Language Research Project

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

* Refer to the Course Section Information (CSI) document posted in Blackboard 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.
* Refer to the documents in the dataset area in Brightspace for the list of dataset columns to use, note that all listed columns need to be used, and need to be present within your source code 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.
* **Caution: Many previous students have found this to be the most challenging activity in the course, budget your time and start work early.**

## Tasks

* 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.
  + 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.
  + 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.
  + Displays your full name on screen so it remains visible at all times, 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.
    - Persist the data from memory to the disk as a comma-separated file, writing to a new file.
    - Select and display either one record, or display multiple records from the in-memory data.
    - Create a new record and store it in the simple data structure in memory
    - Select and edit a record held in the simple data structure in memory
    - Select and delete a record from the simple data structure in memory
* 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.
  + 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.
* 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.)
* 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, 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, and include the source code with your programmer comments within a single MS Word document. Also, include the original source code files as part of your submission. See the expected format, submission requirements, and grading guide below.

## Your single MS Word document should have this general format

* Cover page with your full name within it.
* Heading with name “Evidence of Learning”
  + Code figures to demonstrate use of variables, a loop structure, a decision structure, File-IO against the dataset, exception handling, use of an API library, an array (or similar data structure) and unit testing.
  + Either:
    - Use small code examples for each topic, or
    - Indicate what line numbers in a larger code sample illustrate each concept.
    - **You must indicate clearly to your professor that you can identify what parts of your code illustrate and match to each programming concept.**
* Heading with name “Program Architecture”
  + Briefly outline how you organized your code files or classes into layers or similar architectural design. You must provide UML Diagrams as part of your write up. See this website as a quick example / review as well as see notes inside the rubric as well.

<https://www.uml-diagrams.org/multi-layered-application-uml-model-diagram-example.html>

Note that a single source code file, with methods, is not a layered design. It needs to be broken out into separate files / classes. See the example provided on Brightspace.

* 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).
* 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 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.

## Submission Requirements

* Upload your MS Word document and original source code files by the due date
  + One MS Word document
  + One zip file containing your original source code file(s)

\*\*Submitting any other format other than .doc or docx for your MS-Word document will result in zero for this assessment. Libra-Office users save-as… MS Word.

Do not bundle your MS Word document inside the source code zip-archive keep them separate instead.

* Ensure your full name is included in all materials as asked.

## Grading (Total 18 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. | 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. E.g. A program consisting of a single code file with methods. | Student vaguely mentions layers and/or program may not have at least 3 layers. No diagrams provided. | Student outlines the architecture with each file or class’s placement in a layer indicated. Diagrams show a high-level view of the architecture. | Student outlines, and discusses the architecture 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. |
| 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. | Screen shots are within MS Word document. No explanation of the image. 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 brief generalized description of each image. 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. 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. | Screen shots are within MS Word document. No explanation of the image. 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 each image. Has full name in nearly all screen shots. Test framework runs and performs a basic or rudimentary test. | Screen shots are within MS Word document. Student provides brief yet detailed description of each image. Has full name in nearly all screen shots. Test framework runs test and tests at least one part of the programs 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. | 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. |
| Source Code and Source Code files | Poor / Missing or MS Word document is bundled inside the code zip archive. | Not used for this criteria | Not used for this criteria | Student uploads both MS Word document, and source-code zip file keeping them separate, i.e. MS Word document is not inside zip file. |

## 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, this includes the programmer comments. If there are large or many differences, you will lose marks.

**If you do not submit your full source code in a zip archive to so your code figures can be verified this entire assessment will be awarded zero points. It is your responsibility to submit the correct files.**

Name your project something similar to CST8333ProjectByYourName, where ‘Your Name’ is your ACSIS name. Where you develop the project through Practical Project Parts 1 through 4, naming your project “Project 1” (etc.) will not look professional for future submissions.