Social\_Media\_Exam 3 – Documentation File

Christopher Newby

CS 350 – Fall 2017

Professor Mark Seaman

**Classes**

1. **Person** – 7 public methods, 1 private method. This is a class for the users of the system. It contains all variables and methods necessary for the creation and maintenance of person objects.
2. **Friend** – 3 public methods, 3 private methods. This class is for the tracking of friendships between users, it’s the equivalent of a many to many database table because its used for tracking many different users that have many different friends.
3. **Post** – 10 public methods, 0 private methods. This class handles all aspects of creation of a post. This contains all the variables and the methods necessary for the creation and maintenance of post objects.
4. **System\_Mod** – 4 public methods, 9 private methods. This class is the system functions such as System IO functions, etc. Originally it was named “System” but calling a class “System” in C# caused some unforeseeable issues for the program itself, thus it was renamed to be “System\_Mod” as short for System Module.
5. **Data\_Store** – 10 public static methods. The class contains static lists used for local storage of data, all of the public static lists on this class were used for the storage of local data for testing purposes until the IO methods were written very last in the chain of events.
6. **Test** – 1 public method, 14 private methods. This is the test module that is written as its own class. It was necessary to write the test as part of the program to test other methods.

**Design**

The design phase of the project was to take the exam instructions and create a set of instructions that would provide the individual functionality of the desired outcome. When I started out, I knew that doing each one of the pieces as individual functions was not going to give me the outcome that I wanted because with C# it’s strongly typed nature and the way that it’s set up meant that I would need several individual supporting functions aimed at getting the data that I required in the form that I required it. I started out using the basic function names but then in designing for each of the individual functions themselves it became apparent that I could easily do each of the desired functions in one method a piece, but I knew that it would not only look terrible, but there would be repeated code everywhere and that the individual methods themselves would be very long. This would seem to violate the idea of using good OOP and so I elected to design whatever pieces I needed to support the main functions as need be. I kept the functions that I needed to be public in the API portion of my class code, any of the other ones that I knew would only be for use within the class and that would not be used outside the class for any discernable reason would be given the private label and pushed to the non-API portion of the class. This allowed me to separate my API functions from the rest of the functions that were merely supporting functions. The code had several refactor phases for each module. First there was the “what are the pieces that I will need to make this thing work” phase, followed by the “get it turned into code of some sort” phase, then came the part where I refactored each method to accomplish the original task, but in some parts of the code I redesigned the functions to approach the needs in different ways such as passing an array or a list of generic objects instead of a list of defined objects, but then casting them into their perspective types based on input. Ultimately there was the “does this thing do what I really want or need it to do?” phase where I was able to establish within the larger picture of my design if the parts I had spent so much time working on to derive the ultimate answer of whether or not they would be useful and functional. Doing this allowed me to make one generic Write\_To\_CSV and Read\_From\_CSV functions one time that would work for any list of given objects, rather than creating individual functions for each of the different types of list objects for example. It started out as individual functions for each, but with much repeating code it forced me to ask myself “isn’t there a better way of doing this?” and convincing myself there must be!

**Interface**

I do have a small, partially functioning interface that I used for testing purposes. It was a simple form with a button that would trigger the master test function “Run\_Master\_Test()” function, which in turn simply activated all the individual functions in their own right to test them in an order that made sense and one that followed the general pattern of the design layout of the functions set forth in the exam instructions as being desired. It does contain the basic functional capability of creating users, posts, and friendships, but that is the limit of its ability. There is no data validation anywhere in my code, as would normally be the case but I felt with the scope of the project to be completed and the short amount of time to do it in, that it would be better to get it all working than to make sure every tiny little detail was accounted for. This is my source of technical debt on the project to this point. The methods for converting to a CSV from an object were also not my favorite, and in face I had a design in my head for casting them to string arrays and then passing the array itself, but the methods seemed code-heavy and somewhat cumbersome compared to relative simplicity of the methods that I used in the end. My design for my interface will not be included as part of this submission, but will be shown in the documentations of my screenshots of my running program. As you may know, visual studio does not run a console window by default, so logging things to a console window doesn’t work the same as it does for programs such as Python IDLE.

**Checklist**

1. 6 object classes – 52 total methods --- CHECK
2. All code performs as desired --- CHECK
3. 3 files (Test, Product, Design) --- CHECK
4. Tests for each use case (12 in total 11 required + one extra) --- CHECK
5. Test will run automatically when master test called --- CHECK
6. Module Interface that can be used by front end app --- CHECK

**Runtime Log (Screenshots in my case)**

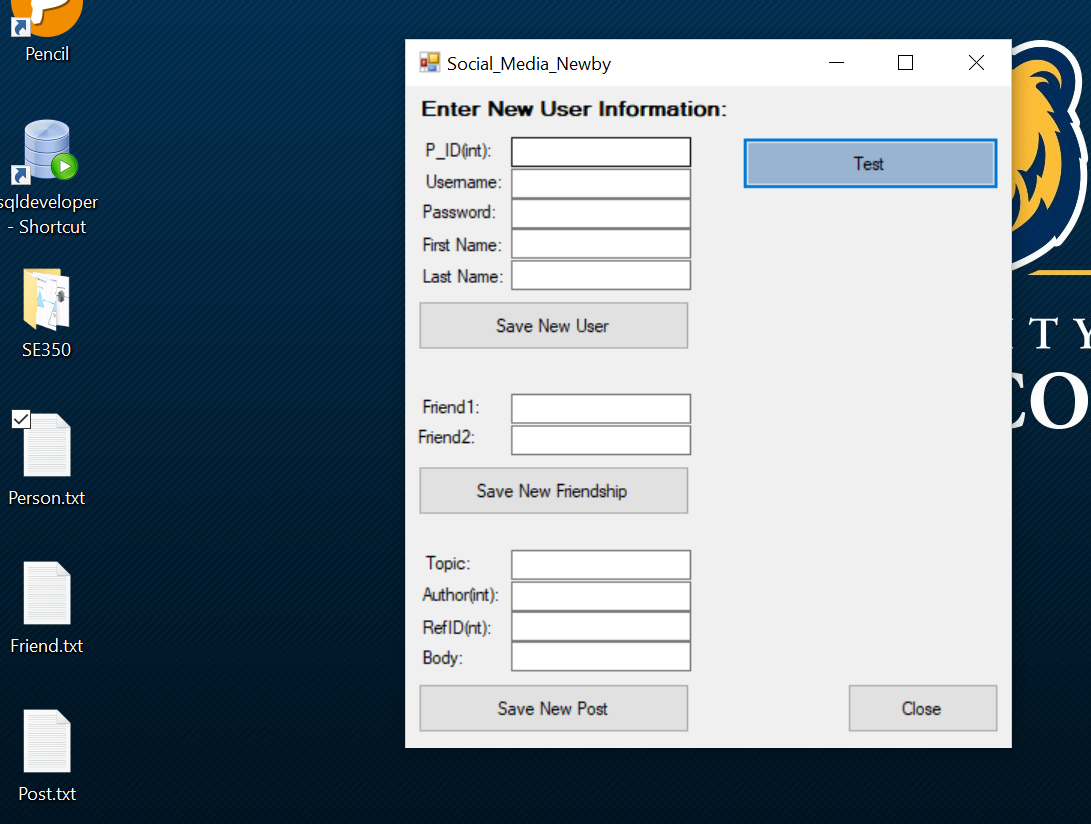
****

Figure Shows Files Created on Desktop After "Test" Clicked

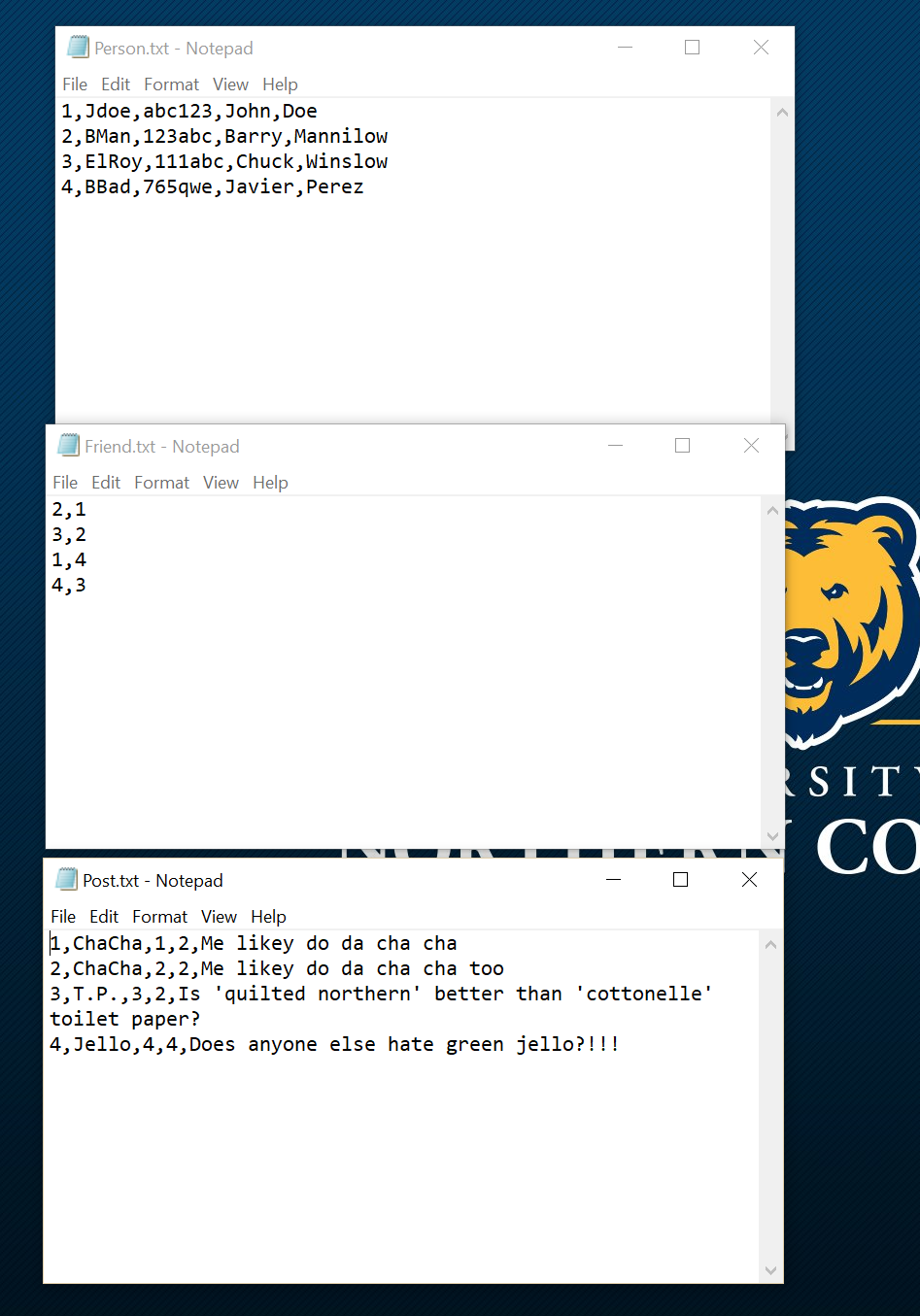
****

Figure Shows Content of Files on Desktop (Peristent Data)

**Output from Debug Window**

'Social\_Media\_Newby.exe' (CLR v4.0.30319: DefaultDomain): Loaded 'C:\WINDOWS\Microsoft.Net\assembly\GAC\_32\mscorlib\v4.0\_4.0.0.0\_\_b77a5c561934e089\mscorlib.dll'. Skipped loading symbols. Module is optimized and the debugger option 'Just My Code' is enabled.

'Social\_Media\_Newby.exe' (CLR v4.0.30319: DefaultDomain): Loaded 'C:\Users\chris\source\repos\Social\_Media\_Newby\Social\_Media\_Newby\bin\Debug\Social\_Media\_Newby.exe'. Symbols loaded.

'Social\_Media\_Newby.exe' (CLR v4.0.30319: Social\_Media\_Newby.exe): Loaded 'C:\WINDOWS\Microsoft.Net\assembly\GAC\_MSIL\System.Windows.Forms\v4.0\_4.0.0.0\_\_b77a5c561934e089\System.Windows.Forms.dll'. Skipped loading symbols. Module is optimized and the debugger option 'Just My Code' is enabled.

'Social\_Media\_Newby.exe' (CLR v4.0.30319: Social\_Media\_Newby.exe): Loaded 'C:\WINDOWS\Microsoft.Net\assembly\GAC\_MSIL\System\v4.0\_4.0.0.0\_\_b77a5c561934e089\System.dll'. Skipped loading symbols. Module is optimized and the debugger option 'Just My Code' is enabled.

'Social\_Media\_Newby.exe' (CLR v4.0.30319: Social\_Media\_Newby.exe): Loaded 'C:\WINDOWS\Microsoft.Net\assembly\GAC\_MSIL\System.Drawing\v4.0\_4.0.0.0\_\_b03f5f7f11d50a3a\System.Drawing.dll'. Skipped loading symbols. Module is optimized and the debugger option 'Just My Code' is enabled.

'Social\_Media\_Newby.exe' (CLR v4.0.30319: Social\_Media\_Newby.exe): Loaded 'C:\WINDOWS\Microsoft.Net\assembly\GAC\_MSIL\System.Configuration\v4.0\_4.0.0.0\_\_b03f5f7f11d50a3a\System.Configuration.dll'. Skipped loading symbols. Module is optimized and the debugger option 'Just My Code' is enabled.

'Social\_Media\_Newby.exe' (CLR v4.0.30319: Social\_Media\_Newby.exe): Loaded 'C:\WINDOWS\Microsoft.Net\assembly\GAC\_MSIL\System.Core\v4.0\_4.0.0.0\_\_b77a5c561934e089\System.Core.dll'. Skipped loading symbols. Module is optimized and the debugger option 'Just My Code' is enabled.

'Social\_Media\_Newby.exe' (CLR v4.0.30319: Social\_Media\_Newby.exe): Loaded 'C:\WINDOWS\Microsoft.Net\assembly\GAC\_MSIL\System.Xml\v4.0\_4.0.0.0\_\_b77a5c561934e089\System.Xml.dll'. Skipped loading symbols. Module is optimized and the debugger option 'Just My Code' is enabled.

'Social\_Media\_Newby.exe' (CLR v4.0.30319: Social\_Media\_Newby.exe): Loaded 'C:\WINDOWS\Microsoft.Net\assembly\GAC\_MSIL\Accessibility\v4.0\_4.0.0.0\_\_b03f5f7f11d50a3a\Accessibility.dll'. Cannot find or open the PDB file.

The program '[7080] Social\_Media\_Newby.exe: Program Trace' has exited with code 0 (0x0).

The program '[7080] Social\_Media\_Newby.exe' has exited with code 0 (0x0).