**CSC 493 Senior project**

**Test Plan:**

In this phase, we will be creating a detailed testing plan for our programs. We will need to create a testing plan document, describing the test cases that will be implemented.

The document should begin with a brief description of the purpose of the testing. For example, if your program has a GUI component and a backend calculation component, you will need to concisely describe the functionality of each you will be testing. Be sure and mention what your program will do in the best case (i.e., if it passes all the tests).

For each function to be tested:

* **Description**: What is the function meant to do.
* **Input / Output**: What are the arguments to the function, their types, and their purpose.
* **Test Cases**: What sorts of tests will you run? For example, in a function to divide one number by another, one of your tests should be to check that the divisor isn’t zero.
* **Possible Errors and Handlers**: Describe the errors you’re testing to prevent, and how you avoid them in your code.
* **Acceptance Criteria**: For features that are not modular functions (e.g. live webpage), describe the criteria under which you would accept the feature as “done.”
* **Schedule:** Set deadline dates for the features to be tested.

**Example:**

* **Purpose**: I’m creating a website that enables the user to write and save notes, and I’m testing the core features of the website to make sure it’s up to date and working. The website is designed to provide an interface for creating, editing, and saving notes.
* **Features to be tested**:
  + The website is live and accessible from network-connected machines.
  + The buttons on the webpage perform the proper functions (create, save, edit).
  + Users can retrieve saved notes to edit.
* **Test Plan:**
  + *T1* – Ensure that website is live and accessible. To test this, I will attempt to access the webpage from multiple physical machines, as well as from a virtual machine.
  + *T2* – Edit button callback
    - *Input*: Mouse click event on edit button
    - *Output*: None
    - *Behavior*: Should open a file dialogue asking the user to choose a file to edit.
  + *T3* – Save button callback
    - *Input*: Mouse click event on save button
    - *Output*: None
    - *Behavior*: Should open a file dialogue asking the user to choose a name for the file.
  + *T4* – Save function
    - *Input*: file name (string), contents (array of strings)
    - *Output*: Contents written to file with given name
    - *Potential errors*: If the file name already exists, the program will provide the user with a popup asking if they wish to overwrite. If the file name input is not a string, it must be converted to one or throw an exception.
* **Schedule:**
  + Test the website availability on October 9, 2019
  + Test the format/interface on October 10, 2019
  + Test functionality on October 11, 2019