EC 327 Fall 2021 Project

**Overview**

The purpose of this project is to create a GUI based application using C++ to keep track of personal health records and perform symptom surveys as is based on the Daily Symptom Survey required for all Boston University students to do using the Student Health Services (SHS) website. Besides providing daily symptom surveys, this program also plans to keep track of doctor’s appointments that are input to the program and all information is recorded on the user’s computer with an output .txt file.

Overall, this application is being built with the wxWidgets framework. wxWidgets is a C++ library that lets developers create applications for Windows, macOS, Linux and other platforms with a single code base. As a result of this our application can be built to run on multiple platforms, primarily Windows and macOS as that is what our team is running on our computers.

**Front End**

wxWidgets provides several different tools and easy to use GUI elements which can be placed anywhere on the screen. Our cMain class holds all the objects and subsequent member functions and variables to create the GUI. We used a system similar to a state space approach to display the correct window given a user's navigation through the app. Initializing a local integer variable to our cMain class, we would change its value (1 through 7) with respective member functions corresponding to which button was pressed or which option was selected from a drop down menu. That way every time the default constructor was called to display the next window as the user navigates, it would construct the correct one based on the value of our integer. This allowed us to efficiently create one class which powers the GUI.

Using WxWidgets allowed us to streamline the creation of this GUI with all the built in functions of the library. The first single window which opens for the user initially prompts them to answer a symptom survey with the ability to answer ‘Yes’/‘No’ through two button options *(using wxButton)*. Also using *wxTextCtrl* we included a text box for the user to report their symptoms should their answer be yes. The top of this page also includes a button to take the user to a scheduling page for their vaccine booster shot. Also at the top left of every window, a menu drop down option is available for seamless navigation throughout the app. After filling out the first page, or any of the subsequent pages where user input is required, a pop up is displayed either confirming the user’s selections/input, or prompting them again if there was an invalid or empty entry. The pops up were made using *wxMessageDialog.* Following the initial window, the user is taken to the next page which requires them to fill a series of text boxes and drop down menu selections (i.e appointment type, appointment day, location, name, BUID, etc.). The drop down menus were made using *wxStaticText* and *wxChoice*. While this is the main functionality of the app and where the user will be required to provide the most input, there are a multitude of other pages with additional information which can intuitively be navigated to.

**Back End**

As briefly mentioned before, the foundation of the application is built upon a state space related approach regarding the value of a local integer variable. With this foundation, we are easily able to call the correct constructor (which initializes and displays the correct window). Additionally this approach allows us to easily call the necessary member functions of the class given the window the user is on. For example, if they fill out the symptom survey, their information is correctly processed and saved in a .txt file with the results of that survey. If they move onto another page to schedule a COVID test, the information they have inputted, such as date of test, location, etc, is processed and saved in a .txt file. Some other back end data processing that occurs includes writing the current dates of output .txt file given the day that the user is using the app. A lot of the work in the back end includes this file IO and appending to the current .txt file if the user is scheduling multiple appointments and filling out surveys in one session.

Another major aspect on the back end involved managing memory leaks of the application. Given that calling our default constructor for cMain was the basis for creating the different windows of our GUI, and that within that constructor other objects such as buttons and text boxes were instantiated, it was important that the cMain destructor was able to clear the memory once certain objects were out of scope. In order to accomplish this task, we applied the same approach as when the default constructor was called–– a state space approach which through pointers would identify which frame/window was being moved on from (as the user navigated) so that the necessary objects (buttons, text boxes, drop down menus) of that page were deleted.