Ideal Plan

Ideally, we would like to test our system on every browser including Safari, Firefox, Chrome, Internet Explorer, Netscape, and Opera. In addition, we would want to test for each individual operating system including Linux, Windows, and Mac OS. We would like to test our scheduling module on a large variety of inputs to ensure that it works over a wide variety of students and test cases. In addition we would test every separately constraint to ensure that they function as expected in different conditions.

Intended Plan:

We will be testing our application on Windows and Mac OS in Chrome and Firefox as these are the most popular browsers and operating systems. We are going to ensure that each page of our application runs smoothly and functions without bugs. In addition, we will test the scheduling engine of our application separately using python unittest framework.

Platforms to be tested:

Windows, Mac OS, Chrome, and Firefox

Test Cases:

Development Sprint 1 – Scheduling Module Test Bench

Test Case 1: We are going to test the creation of the data needed to make a schedule. We will test being able to add candidates and locations to the scheduling module. We will then test the ability of the module to populate arrays containing the greatest need and to find the greatest need of the array. This will be done using the Python 3.4.5 unittest framework.

Test Case 2: We will test the ability of our schedule to take the constraint given and schedule a test data set consisting of one location and four students open for several hours a day. We expect to generate a suboptimal but usable schedule. This will be done using the Python 3.4.5 unittest framework.

Development Sprint 2—Database Test Bench

Test Case 1: We are going to make sure that we can input data into the database. We are going to do this in several different ways. Data is entered into the database via an HTTP POST request when the user enters their availability. We will visually inspect the contents of the front end and compare it to the back end database in order to ensure that changes made to the front end HTML table propagate into the database.

Test Case 2: We are going to make sure that the scheduling engine can receive data from the database and that it can communicate the schedule back to the database where it will be stored as a mongo document. This will be tested using a set of unit test benches written using the Python 3.4.5 unittest framework. These unit test will make sure that what is outputted by the engine is being put into the database and that the schedule stored in the database is what was outputted by the scheduler.

Links:

Development Sprint 1 Links

https://github.com/WritingCenterScheduler/Engine/tree/development/tests

Development Sprint 2 Link

https://github.com/WritingCenterScheduler/Engine/tree/database/tests