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.

Development Sprint 3—Cloud Apps Test Bench

Test Case 1: We are going to make sure that the application deploys smoothly onto Carolina Cloud Apps. In order to test this, we will have several users log onto Cloud Apps to ensure that they are able to access the application. We will make sure that these users are able to select their availability and that this gets passed to the database. This should then get passed into the scheduling engine and the engine should handle scheduling all of the employees and then return the schedule to the application (the schedules will not be displayed yet).

Development Sprint 4—Displaying Schedules, Adding Minor Tweaks

Test Case 1: We are going to ensure that we can display a finalized schedule in full calendar. In order to test that this is correct we are going to compare the output that is given by full calendar to the output that is generated by the scheduling algorithm. If the two are the same then we know that we are correctly and accurately displaying this schedule. In addition we want to ensure that none of the features that we have previously built are broken so we will be testing each of the features again to ensure that the new additions do no break existing features. We do not anticipate that this will be a problem as displaying the schedule is fairly modular.

Test Case 2: We will be adding a few extra features that our client has requested. We will be using user testing in order to ensure that this does not break the existing features of the website. Additionally, we have users test to ensure that the features that we have added work as intended.

Development Sprint 5—Final Tweaks and Bug Fixes.

Test Case 1: For this we will be relying on users to point out any bugs in the user interface and the application. We will be fixing any bugs that users run into and changing the application based on user input. For this we will be using out previous testers to test the data. Additionally, we will be adding the test data that was provided by our client to ensure that the website is fully functional from beginning to end. We should be able to add users to the application and have them enter their availability, add locations and their times, allow the admin to edit the students, set the master schedule, run the scheduling algorithm and edit the master schedule. At this point we want a fully functional pipeline from users to admin. We will be testing this by entering all of her sample data of students and locations and building a schedule to ensure that the application preforms to specifications.

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

Development Sprint 3,4,5 Link

https://github.com/WritingCenterScheduler/application