Deliverable 2

Introduction

In this stage of the project, our team wrote a BASH testing script that takes an input file (input.txt), then enters a while loop. During the loop, it parses the text file line by line, calls the java file and puts the parsed line as the argument during each iteration, calculates and compares the output value with the expected value, and finally prints the answer to the terminal. In later versions, the answer will be added to the output file.

Testing Plan Structure

Testing Process

Our tests will be a testing of individual classes and methods (unit testing) for the core operations of the API. This will be followed by component testing, by testing the interactions of these classes with others and their accompanying interfaces. This a major part of OpenMRS is its database (where medication, patient, and other records are stored), most of our testing will involve the input and processing of data, as well as the functioning of parent and child classes.

Requirements Being Tested

We are testing parts of the program that input and process data to confirm the reliability and functionality of the database portion of OpenMRS.

Tested Items

Operations from the API will be tested, such as Medication, Patient, Prescriptions, and other functionality related to the database. Components involving the API, webapp, and core parts of the program will be tested to confirm that their operations interact with each other as they should.

Testing Schedule

The tests that will be developed will go from small, basic unit tests to component tests. Once the classes are tested and pass, component tests will be run to test the integration of these operations. Once both of these are complete, a detailed test report can be processed and reviewed to determine the functionality of the software.

Testing Process

A master script will locate all available test files within the test file directory. The script assumes that all the files are formatted in a similar way. The script will obtain information about the test file's executable (and its location), what it is testing, the specific methods being used, the inputs, and the corresponding oracle.

Per test, after the information is read from the file, the executable driver file is compiled and then run with command line arguments specified in the test file. Upon completion of the test, the executable will output data to a file.

Test Recording Procedure
The master script will compare the output file with the oracle corresponding to the same test. A report will be processed to show how many cases failed, and which specific inputs caused the failure.

The team will review the reports and manage the failures. We will decide if the failure was an issue with the testing executable, or the program itself. If it is determined to be an issue with OpenMRS, this will be reported to them.

Hardware and Software Requirements
The software must be able to run with full functionality on Ubuntu 18.04. For our tests, we will be testing the software on an Ubuntu virtual machine. The master script must be able to be called from the top-level directory of our test repository.

Constraints
The sheer size of the program, as well as the complexity of the components themselves provide a challenge to testing. Since the software system has several components that connect to the API (such as a webapp), isolating test cases to certain classes and components may be challenging.

Specified Test Cases
Currently, we have 5 of our total 25 test methods specified.

/openmrs-core/web/src/main/java/org/openmrs/web/WebUtil.java
This class has a lot of good testable methods like: escapeQuotes, escapeNewlines, and stripFilename

/openmrs-core/api/src/main/java/org/openmrs/util/OpenmrsUtil.java
Another util class, these are easy to test and optimal for starting out our testing suite, here we have methods like: getFileAsString, validatePassword

The validatePassword is an interesting method to test as OpenMRS has default requirements for passwords and it would be interesting to see if we could set a password normally not allowed.

Because this project was built with testing in mind there are several tests that execute when building the program, we will seek to emulate this behavior in our driver class based on component and method being tested.

Conclusion

This will be our plan as we continue to develop our test cases, and then ultimately perform them on OpenMRS. Our goal is to test for the functionality of the API of the system, which is largely connected to the database components of the system. From there, we should be able to discern any faults, if found, with the system's operation.