ClinicFlow

Test Plan

Maxim Vasiliev #400043983 Susie Yu #000955758
Karl Knopf #001437217 Weilin Hu #001150873
Yunfeng Li #001335650

April 9, 2017

Contents

List of Tables

1 Introduction

1.1 Introduction

This document details our approach to testing ClinicFlow. This software will be used in a live clinic environment, so it is particularly important to ensure its robustness, reliability, and performance. We begin by testing individual components, modules and methods therein. This allows us to catch errors early in development, and help guide the design process. Next we will perform system testing, where we verify the correct operation of various user tasks. We must also perform user testing to make sure that the product is usable by clinic nurses and managers.

1.2 Definitions and Acronyms

Table 1: Definitions and Acronyms

Phrase	Definition
Module	A unit or section in a clinic
Provider	A health care worker (A doctor, nurse,technician)

2 Unit Testing

2.1 Overview

As the programs will be written in the Python 3 programming language, we can use the unittest framework. We will define test cases, with inputs and expected outputs and then be able to run this for each module in the system. We have not yet specified the names of each module and the functions contained, but we do have a general idea of what modules we would want and what they should do.

2.2 PatientSchedule.py

This is the module in the program that will take in a file containing patients in the clinic, and produce a schedule for them. First we must test if scheduling system can take appropriate inputs.

2.2.1 Test Inputs

Table 2: Unit Tests - `Patient Schedule-Input`

Test #	Inputs	Initial State
1	No patient dataset	No patient dataset present
2	Sample (valid) patient dataset	No patient dataset present
3	Incorrectly formatted patient dataset	No patient dataset present

We must the check the function (or functions) that create the patient schedule.

2.2.2 Test Inputs

Table 3: Unit Tests - `Patient Schedule- Scheduling`

Test #	Inputs	Initial State
4	Sample patient dataset	No patient dataset present
5	Sample patient dataset with many repeats	No patient dataset present

We must also check the function that will output the schedule in a useable form for the rest of the system.

2.2.3 Test Inputs

Table 4: Unit Tests - `Patient Schedule-Output`

Test #	Inputs	Initial State
6	Sample patient dataset	No patient dataset present
7	Sample patient dataset with many repeats	no patient dataset present

2.3 HealthcareSchedule.py

This is the module in the program that will take in a file containing the names and times available of the health care workers at the clinic, and it will be able to generate a schedule for them. We should first test to see if the module is able to accept the appropriate input.

2.3.1 Test Inputs

Table 5: Unit Tests - `HealthCare Schedule-Input`

Test #	Inputs	Initial State
8	No health care dataset	no health care dataset present
9	Sample health care dataset	no healthcare dataset present
10	Incorrectly formatted health care dataset	no healthcare dataset present

We must the check the function (or functions) that create the health care schedule.

2.3.2 Test Inputs

Table 6: Unit Tests - `Health Care Schedule- Scheduling`

Test #	Inputs	Initial State
11	Sample health care dataset	no health care dataset present
12	Sample health care dataset with many repeats	no health care data set present

We must also check the function that will output the schedule in a useable form for the rest of the system.

2.3.3 Test Inputs

Table 7: Unit Tests - `Health Care Schedule-Output`

Test #	Inputs	Initial State
13	Sample (valid) health care dataset	no health care dataset present
14	Sample health care dataset with many repeats	no health care data set present

2.4 Clinic.py

Clinics can have a variety of modules such as Blood taking or x-ray. Each section can be unique or have multiples. This program module uses clinic information from a data file, which will be read into the system when a simulation is run for that clinic.

2.4.1 Test Inputs

Table 8: Unit Tests - `Clinic Modules-Input`

Test #	Inputs	Initial State
15	No clinic dataset	no clinic dataset present
16	Sample (valid) clinic dataset	no clinic dataset present
17	Incorrectly formatted clinic dataset	no clinic dataset present

We must also check that the functions in this program module create a valid network of modules for the clinic.

2.4.2 Test Inputs

Table 9: Unit Tests - `Clinic Module-Output`

Test #	Inputs	Initial State
18	Sample (valid) clinic module system with no repeats	no clinic data set present.
19	Sample (valid) clinic module system with many repeats	no clinic data set present.

2.5 SimulationEngine.py

The system relies on the simulation engine to generate any results. This module must be able to take in the previous schedules and data, and use that to run a discrete event simulation using those schedules. We need to test both under normal conditions and under extraordinary conditions.

2.5.1 Test Inputs

Table 10: Unit Tests - `Simulation Engine`

Test #	Inputs	Initial State
20	Proper datasets under normal conditions. 25 patients in simulation.	no simulation data stored.
21	Proper datasets under normal conditions. 50 patients in simulation.	no simulation data stored.
22	Proper datasets under normal conditions. Only 2 flexable workers.	no simulation data stored

23	Proper datasets, but all breaks are simulataneous.	no simulation data stored
24	Changed datasets to larger numbers.	no simulation data stored.

2.6 Summary Output

The final goal of this product is to generate output which aids clinic employees in decision making. For this, the output must be formatted in a way which conveys a quick and thorough understanding of simulation results and recommended schedule. There will be multiple formats, including visualizations of components as well as formats in text.

2.6.1 Test Inputs

Table 11: Unit Tests - `Summary Output`

Test #	Inputs	Initial State
25	Genuine data	No previous output.
26	Skewed data	No previous output.
27	Invalid data	No previous output.
28	Missing Data	No previous output.

2.7 User Interface

The target users of this product includes clinic managers and/or other organizations with similar scheduling goals. The user interface should be clean and intuitive as some of these users might not have extensive knowledge in software products. When the user attempts to make illegal operations,the interface should respond accordingly and direct the user towards correct behaviour.

2.7.1 Test Inputs

Table 12: Unit Tests - `User Interface`

Test #	Inputs	Initial State
29	Base case	no previous interactions
30	Invalid interaction / value input	no previous interactions
31	Aberrant user behaviour	no previous interactions

2.8 Data Storage

It is important that the data is able to be saved and recalled. Detailed simulations take vital time, and users would wish to compare with previous results to evaluate the sensitivity of the system to variables. Storage errors should also not interfere with previously stored data.

2.8.1 Test Inputs

Table 13: Unit Tests - `Data Storage`

Test #	Inputs	Initial State
32	Valid simulation output data	no current data stored.
33	Missing data	no current data stored
34	Incorrectly formatted data input	no current data stored

3 System Testing

3.1 Login

The purpose of user login is to ensure that only valid users allowed to in access the system. There are two types of account. One is the administrator account which has full authority to manipulate the data and control all operable functions of the system. Another one is the viewer account which only can view the information. Testing retrieves the input account and password and match with the account information in an existing database to determine whether the user is valid and what the user can do.

Table 14: System Tests: Login

#	Initial State	Inputs	Pass Criteria
35	Login Page Empty input field of account and password.	Empty input of one of the input fields or both. Click login.	Stay at same page and error message of empty input.
36	Login Page Empty input field of account and password.	Valid input of account and password of administrator account. Click login.	Redirect to the main page of the application, and give user full authority of control.
37	Login Page Empty input field of account and password.	Valid input of account and password of viewer account . Click login.	Redirect to the main page of the application, user only allowed to view the data and schedule.
38	Login Page Empty input field of account and password.	Invalid of account or invalid password. Click login.	Stay at the same page and present an error message of the invalid login.

#	Initial State	Inputs	Pass Criteria
39	Application main page	Click logout.	Redirect to the login page.

3.2 Insertion and Storage of Data

The application should allow administrator account user to insert data such as patient procedure, doctor and nurse shift hours and other necessary data into corresponding databases. The testing will compare the inserted data with the data stored in database. There exists a checking module to validate the input values.

Table 15: System Tests: Data Storage

#	**Initial State**	**Inputs**	**Pass Criteria**
40	Application main page, admin account.	Click Add Data.	Redirect to the Application adding data page.
41	Application adding data page.	Add the inexistent data to target database with correct data type. Click submit.	Stay in same page and all inputs are cleaned. Data appear in the corresponding database.
42	Application adding data page.	Add the data to target database with incorrect data type or incorrect pattern. Click submit.	Stay in same page and save valid data. Error messages indicate the invalid inputs.

43	Application adding data page.	Add the data which already existed in target database. Click submit.	Stay in same page and save data. Error message indicates the redundancy of data.
44	Application adding data page.	Add data which out of domain (such as reservation date in past days) to target database. Click submit.	Stay in same page and save data. Error message indicates that the data out of valid range.
45	Application main page, viewer account.	Click Add Data.	Error message.

3.3 View, Modify and Delete Data

Administrator account user can view, modify and delete existed data in database. Viewer account user only can read the data in the database. According to the selection of user, retrieve corresponding database and display the content on user interface. Changes and deletion created by administrator account user should be stored into database. Testing checks whether the application display right data, and whether the changes synchronize with database. Test the validation of the database when unexpected actions happen.

Table 16: System Tests: Data Manipulation

#	**Initial State**	**Inputs**	**Pass Criteria**
46	Application main page, admin and viewer account .	Click View Data.	Redirect to the View Data page.

47	Application view data page, admin and viewer account.	Select target database, click view.	Stay at same page. Display the content of all data from corresponding database.
48	Application view data page, admin and view account.	Select target database and give specific condition, click view.	Stay at same page. Display the content of target data from corresponding database.
49	Application view data page, admin account.	Modify the displayed data and change the value by another valid value. Click save.	Stay at same page. The value in display area and database have been changed.
50	Application view data page, admin account.	Modify the displayed data and change the value by invalid value(empty for required value or out of valid range). Click save.	Stay at same page, no change happen in displayed data or database. Error messages indicate the unexpected changes.
51	Application view data page, admin account.	Delete whole one row of data by clicking deletion button at end of the row. Click save.	Stay at same page. The deleted row disappear and the data in database is deleted as well.

52	Application view data page, admin account.	Delete whole one row of data if the data is expected to use in future (Patient reservation in next few days). Click save.	Stay at same page. Pop up a deletion confirmation window. Confirming the deletion will remove the row from the list. Cancel deletion will save the data.
53	Application view data page, viewer account.	Modify the displayed data.	Stay at same page. Data is not editable.
54	Application view data page, viewer account.	Click delete button.	Stay at same page. Deletion button does not exist.
55	Application view data page, viewer account.	Click save.	Stay at same page. Save button does not exist .

3.4 View Schedule and Modify Schedule

The application allows administrator account user and viewer account user to view the schedules. Administrator account user also has the permission to adjust the schedule. Testing ensures that the changed schedule doesnt have conflicts.

Table 17: System Tests: View Schedule and Modify Schedule

#	Initial State	Inputs	Pass Criteria

56	Application main page, admin and viewer account.	Click view schedule.	Redirect to the view schedule page.
57	Application view schedule page, admin and viewer account.	Select type and date of the schedule. Click view.	Stay at same page. The page displays the existing schedule.
58	Application view schedule page, admin account.	Click adding button and add a new reserved time into blank area of schedule. Click save.	Stay at same page. The new time period is added into schedule.The new schedule is saved into database.
59	Application view schedule, page admin account.	Click on adding button on a reserved area. Click save	If no error happens , stay at same page. No adding button on a reserved area.
60	Application view schedule page, admin account	Click on moving button on a reserved area and choose an empty area. Click save.	If no error happens stay at same page. The selected time period is set at new area. The new schedule is saved into database.

61	Application view schedule page, admin account	Click on moving button on a reserved area and choose a reserved area. Switch the reservation. Click save.	If no error happens, stay at same page. The selected time period is switched with another one. The new schedule is saved into database.
62	Application view schedule page, admin account.	Application validates the changes of schedule.	Checks on the sequences of procedures in clinic. If adding, moving, and switching violate the the sequence, show error message.
63	Application view schedule page, admin account.	Click delete button to remove a reservation. Click save.	Pop up a window for deletion confirmation. If continue to delete, remove the reservation. Save the new schedule to database.

64	Application view schedule page, viewer account.	Click adding or click moving or click delete or click save.	Stay at same page. No adding button, no moving button, no deletion button, no save button.

4 Non-Functional Requirements Testing

4.1 Usability

Table 18: Usability Tests

#	**Description**	**Type**	**Pass Criterion**	Tester(s)
65	We will list the most frequently performed tasks, and the development team will use the product to complete them. We count the number of mouse clicks.	Functional (dynamic, manual)	The average number of mouse clicks should be less than five.	Development Team
66	We will invite five doctors and five nurses to use our product and give them a three minutes demonstration on how to complete a certain task. After that, the participants will try to complete the same task.	Functional (dynamic, manual)	The participants can complete the same task correctly in three minutes.	Testing Team

67	We will invite five doctors and five nurses to use our product, but we will not give a demonstration on how to complete a certain task. Instead, the participants will try to complete it based on their previous experience and the hints provided by the application.	Functional (dynamic, manual)	The participants can complete the required tasks in five minutes and they should encounter fewer than three errors.	Testing Team

4.2 Performance Testing

Table 19: Performance Tests

#	**Description**	**Type**	**Pass Criterion**	Tester(s)
68	The development team will use the product to complete the most frequently performed tasks. We will calculate the time from the start-up of the application to the completion of the work.	Functional (dynamic, manual)	The average time should be less than five minutes.	Development Team

69	The development team will generate 100000 data points, which is ten times more than the expected data points. We will input those data points into our application. Next, we will input 100 times more data points into the application.	Functional (dynamic, manual)	The application does not crash, and there are no obvious latencies (less than 5 seconds for each task) when performing the common tasks in the first case. The application can crash in the second case.	Development Team
70	The development team will generate random data points, which contain problems such as incorrect formatting data and illegal characters. We will input those data points into our application.	Functional (dynamic, manual)	The application does not crash and prompts the users that the input data is not correct and how they could correct it.	Development Team

5 Automated Testing

Automatic tests can be used to repeatedly compare results with previous test runs. It will be used to supplement the performance testing, by generating data sets including patients, arrival times, procedure stations, and employees. These tests can help find the limits on performance and possible unforeseen issues in certain combinations of data inputs.

5.1 Automated Unit Testing

There should be automated testing of the simulation engine. This testing should occur every time a simulation is run to determine if that trial is a valid trial. It should be applied to the SimulationEngine.py module, as that is where in the program the simulation occurs.

6 Schedule

Table 20: Testing Schedule

Date Finished	**Test Type**	**Event**	**Testers**
2016-11-20	Intial Tests	Proof of Concept Demonstration	Development Team
2017-02-12	Unit Tests	Demonstration Revision 0	Developement Team
2017-02-12	System Tests	Demonstration Revision 0	Developement Team
2017-03-22	Unit Tests	Test Report Revision 0	Developement Team
2017-03-22	System Tests	Test Report Revision 0	Developement Team
2017-03-22	Non-Functional Tests	Test Report Revision 0	Developement Team
2017-04-09	All Tests	Final Documentation	Developement Team