CSCI 362
Team Isn’tThisFun - Deliverable #2
Test Plan H/FOSS Project: Beets

Introduction

Our team has decided to switch from our original choice, Open Trip Planner, to our other choice, the project Beets. Open Trip Planner contained too many bugs and problems and would be a problem to actually test. Beets is a python program and runs straight from command line. Beets also provides a fair number of testable classes. Our goal for the project is to create an automatic testing framework for the open source project Beets through collaboration. Ultimately, the test results, success or failed, will be displayed on a HTML page. In this deliverable, we will outline five of our tests cases and detailed specifications of our test plan.

The Testing Process

Team Isn’tThisFun will create a script to run 5 automated tests on 5 methods within the project Beets, for a total of 25 tests. These tests will compare the output of each method with an oracle for that method. The script will then display the results in a web browser. The way our team will do this is running executable files, which create output files to be compared to the corresponding oracle files. The results of this comparison between the output file and oracle file will determine whether the test passes, and then the results will be recorded and displayed on an HTML page within a web browser.

Requirements Traceability

Users are most interested in the system meeting its requirements and individually testing all requirements. Requirements of methods:

1. human_bytes(size) - The method should take in an integer corresponding to file size and output a human readable string displaying the file size.
2. human_seconds(interval) - The method should take in an integer corresponding to a number of seconds and output a human readable string displaying the time interval using English words.
3. human_seconds_short(interval) - The method should take in an integer corresponding to a number of seconds and output a short human-readable M:SS string displaying the time.
4. _sc_decode(soundcheck) - The method should take soundcheck files as input and output a decoded full
   scale value.
5. _string_dist_basic(str1, str2) - The method should take in two strings as input and output basic string distance normalized by string length formed by a Levenshtein calculation.

Tested Items

The methods of the project, Beets, which are to be tested are as follows.

Method and its Location:

human_bytes(size) → beets/ui/*init*.py

human_seconds(interval) → beets/ui/*init*.py

human_seconds_short (interval)→ beets/ui/*init*.py

_sc_decode(soundcheck) → beets/mediafile.py

_string_dist_basic(str1, str2) → beets/autotag/hooks.py

Testing Schedule

Overall Project Testing Schedule Based on Syllabus

9/29: Deliverable #2 - Create Test Plan with 5 of 25 test cases

10/22: Deliverable #3 - Design and build an automated testing framework including architectural description (with at least 5 test cases)

11/12: Deliverable #4 - Complete design and implementation of testing framework and create 25 test cases

11/24: Deliverable #5 - Design and inject five faults into code

12/1,3: Team Project Presentations and Final Report

Detailed by Week Schedule

9/28 - 10/4: Revise Test Plan if necessary

10/5 - 10/11: Driver frame working (Oracle output, etc) / Basic script working (call driver, read from file)

Designing automated testing framework /scripting / 5 Test Cases

10/12 - 10/18: Driver more functional / Script more functional

Designing automated testing framework / scripting / 5 Test Cases

10/17-20: Fall Break / Driver/Oracle/Script working for at least 5 test cases

10/21- 25: 5 Test Cases

10/26 - 11/1: Scripting framework / 5 Test Cases

11/2 -11/8: Scripting framework

11/9 - 11/15: Scripting framework / Finish Deliv #4 / Injecting faults into code

11/16 - 11/22: Inject 5 faults into code & working on Final Presentation

11/23 - 11/29 - Finish Deliverable #5 / Thanksgiving!

12/1, 3 - Final Presentations

Test Recording Procedures

Test cases will be recorded on a table with columns, test ID, method name, inputs, expected outcomes, actual outputs, and results(pass or fail). Once the tests are completed, they will be outputted html text file and reviewed and checked by all team members.

Hardware and Software Requirements

- Linux
- Python 2.7
- Levenshtein (optional)
- Python IDE

Constraints

Scheduling is a major hindrance affecting the testing process. Since the team members have different schedules that usually do not align, unfortunately only half of the team members were able to meet and work on the project in person. The rest of the collaboration had to be done virtually.

Test Cases

Test ID: 1a
Requirements being tested: The method should format file size in a human readable way.
Component being tested: beets/ui/__ init __
Method being tested: human_bytes(size) // int size Test inputs containing command-line arguments: human_bytes(1023)
Expected Outcome: 1023.0 B // String size

Test ID: 1b
Requirements being tested: The method should format file size in a human readable way.
Component being tested: beets/ui/__ init __
Method being tested: human_bytes(size) // int size Test inputs containing command-line arguments: human_bytes(-1)
Expected Outcome: -1.0 B // String size

Test ID: 1c
Requirements being tested: The method should format file size in a human readable way.
Component being tested: beets/ui/__ init __
Method being tested: human_bytes(size) // int size Test inputs containing command-line arguments: human_bytes(0)
Expected Outcome: 0.0 B // String size

Test ID: 1d
Requirements being tested: The method should format file size in a human readable way.
Component being tested: beets/ui/__ init __
Method being tested: human_bytes(size) // int size Test inputs containing command-line arguments: human_bytes(2000000000)
Expected Outcome: 1.9 GB // String size

Test ID: 1e
Requirements being tested: The method should format file size in a human readable way.
Component being tested: beets/ui/ __ init __
Method being tested: human_bytes(size) // int size Test inputs containing command-line arguments: human_bytes(None)
Expected Outcome: TypeError

Experiences

Beets was extremely well documented and had great instructions for cloning and building the source code into Ubuntu. All that was necessary was cloning the GitHub repository and installing pip. All team members were able to build beets correctly. Beets became frustrating once we started looking deeply at the code because a lot of the methods modify and use databases. After analyzing each method and deciding we should make sure there were several testable ones, we finally found five methods, three fairly easy and two more complex. The challenging part of this deliverable was to decide that we were definitely changing projects. Initially, we were unsure of where to begin this deliverable; however, in researching and looking through the website provided on the syllabus, it became clear how to format correctly a test plan. This was a definitely a learning experience. It has taught us that, when pressed for time to meet a deadline, making good, fast decisions such as changing from Open Trip Planner to Beets was necessary in order to complete the deliverable and move on in our team project.

References

Resources used to complete the test plan: http://iansommerville.com/software-engineering-book/web/test-planning/