UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA
Faculty of Engineering, Built Environment and Information Technology

Tests and Reports
Cerebero

Frederick Ehlers	11061112
Jacobus Marais	15188397
Rikard Schouwstra	15012299
Victor Twigge	10376802

Stakeholders

Computer Science Department of University of Pretoria:	Vreda Pieterse
eCivix	Daniël Eloff Chairperson

Contents

1 Introduction **2**

2 Testing **2**
- 2.1 Front End . . . 2
  - 2.1.1 Mocha . . . 2
- 2.2 Back End . . . 2
  - 2.2.1 unittest (Python) . . . 2
- 2.3 Unity . . . 3
  - 2.3.1 Unity Test Tools . . . 3
- 2.4 Integration testing . . . 3
  - 2.4.1 Travis Continueous Integration . . . 3

3 Reports **4**
- 3.1 Reporting of Tests . . . 4
- 3.2 API documentation . . . 4
  - 3.2.1 Swagger . . . 4

4 Appendix **4**
- 4.1 Github . . . 4
- 4.2 Trello . . . 4
- 4.3 Swagger . . . 4
- 4.4 Heroku . . . 4
- 4.5 Travis CI . . . 4

1 Introduction

We Cerebero are working together with eCivix to create a new web based game. The idea of the game is to create an election simulator to teach High school students how elections work and what their vote essentially means in the greater scheme. The user will create party that they will control. The game will revolve around the party gaining funds and man power to do campaigns and gain more funds and man power to run bigger and more effective campaigns. The user with the score at the end of the game wins. The user will be playing against an AI (Artificial Intelligence) player that we will program. The AI will try be more effective/ more successful than the user. There will be a leader-board with all the users' scores and at the end of the client's event a winner will be chosen for a prize.

2 Testing

2.1 Front End

2.1.1 Mocha

1. Testing the AngularJS front end we will be making use of the testing framework called Mocha. We choose Mocha as it runs on NodeJS and uses JavaScript to do the tests, which to our benefit because of the fact that AngularJS also works with JavaScript.

2. This will then allow us to test our database Create, Read, Update and Delete operations that are carried out from the front end to ensure they are successfull and fail when they are supposed to.

3. Mocha is strong but also deverse since we can utilise Mocha to do code coverage as well. Thus allowing us to see how well our tests cover all our code, only then we will truly know if we are testing for everything.

4. There are numerous other reasons you want to use Mocha to improve our code:

   (a) It works with Continues Integration. Therefore our CI will not crash with Mocha tests.

   (b) Supports node debugging

   (c) Support for extensible reporting

   (d) Has copious amounts of documentation and support available

   (e) Much more benefits, feel free to see Mocha for more benefits.

This will provide us with a high quality product due to powerful technology of Mocha and thus quality of the test and their results will pass all quality expectations.

2.2 Back End

2.2.1 unittest (Python)

1. The bonus of using unittest for our Python code is that it comes standard with Python from version 2.1 or greater. Thus one less dependency to handle since this one is built into Python

2. It allows us to write our own classes as long as they derived from unittest.TestCase. Allowing us flexibility to test our code thoroughly and extensively.

3. It makes testing easy regardless if it is a small test or a very very large test and has functionality to assist with the larger test to ensure they run quickly and efficiently.

4. Provides a lot of detail when a test fails.

This will provide us with a high quality product due to powerful technology of Mocha and thus quality of the test and their results will pass all quality expectations.

2.3 Unity

2.3.1 Unity Test Tools

1. We are making use of Unity Test Tools since the framework is provided by the same company and can be easily integrated and used to do all the different tests we need to run.

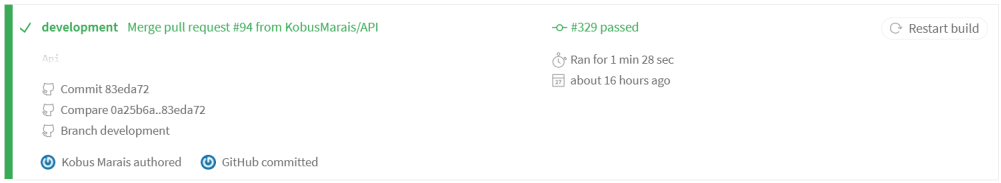
2.4 Integration testing

2.4.1 Travis Continueous Integration

1. We will use Travis CI to do our integration tests.
2. It will run our unit tests and gives us feedback on their successful/ failures. The bonus being that if Travis fails it will not build the project leaving the game on a last working copy and will only overwrite and do the update once the Travis build succeeds. Allowing for a great user experience since they will not experience any downtime of the game.
3. Please see how Travis builds here.
4. Travis.CI testing is integration testing that runs our integration tests to ensure that all parts still work when connected to one another, below is an example of these tests:

```
820
821   register
822     ✓ Returns a string of test data run by register calls
823 John
824 Doe
825 John.Doe@gmail.com
826 POST /api/register 200 21.987 ms - 25
827 Sum of numbers=
828 GET /AI 200 62.583 ms - 0
829
830   login
831     ✓ Returns a string of test data run by login calls
832 Johndoe
833 1234abcd
834 POST /api/login 200 1.551 ms - 25
835
836   collectFunds
837     ✓ Returns a string of test data run by collectFunds calls
838 123abc
839 Gauteng
840 POST /api/collectFunds 200 1.467 ms - 163
841
842   pollProvince
843     ✓ Returns a string of test data run by pollProvince calls
844 123abc
845 Gauteng
846 POST /api/pollProvince 200 0.959 ms - 177
847
848   getFunds
849     ✓ Returns a string of test data run by getFunds calls
850 123abc
851 POST /api/getFunds 200 0.658 ms - 17
852
853   getProfile
854     ✓ Returns a string of test data run by getProfile calls
855 123abc
856 POST /api/getProfile 200 0.647 ms - 54
857
858
859   7 passing (148ms)
```

5. This is the summary that is displayed every time a new feature is pushed to a branch.



6. We have setup our integration tests to also do unit testing, this ensures that every time code is pushed to a branch, that all of the functions still work as they are supposed to work, this does take a bit longer as displayed, but it ensures that we know the moment when something breaks.

3 Reports

3.1 Reporting of Tests

1. We will be using the reporting from the Mocha tests for reporting and quality control.
2. The feedback from the Unity Test Tools will be used as reporting.
3. unittest from Python will be utilised fully to provide detailed reports on tests that pass, fail and are expected to fail.

3.2 API documentation

3.2.1 Swagger

1. Made use of swagger to document and report on or API calls so there is clear requirements and details about the various API calls and make them manageble.

4 Appendix

4.1 Github

Github

4.2 Trello

Trello

4.3 Swagger

Swagger

4.4 Heroku

Heroku

4.5 Travis CI

Travis CI