Smart NFC Card Application

Testing Policy

Vast Expanse (Group 7)

COS 301 - 2019

Wian du Plooy	*u17237263*
Duncan Vodden	*u17037400*
Tjaart Booyens	*u17021775*
Savvas Panagiotou	*u17215286*
Jared O'Reilly	*u17051429*

Table of Contents

Definition of Testing

The purpose of testing is to ensure that the software does what it intends to do. We need to make sure that all functions of the service behave as expected when certain parameters are given as input to ensure that expected output is given to the end user. This section provides definitions of the different types of tests we have implemented for our software.

Unit Testing is when individual components (units) of the system gets tested to ensure that the basic logic of the unit works. The tests use stubs/mock data as parameters and/or outputs. Unit tests can be run at different granularities ranging from testing the program/class to testing individual functions.

Integration Testing is when different components (units) gets tested together to see if they integrate as expected. Integration Testing takes place after all related unit tests are done, to ensure that the individual units aren't causing the problems.

Test Process

After a specific function is coded, the person responsible for the function oversees writing several unit tests for that function and ensuring that the test passes.

All tests that will be written will comply with a strict and uniform structure that will give enough detail of what the test is testing. All tests should be short, concise and to the point, while still providing enough information in the report so that people running the tests are sure which tests are run and on what part of the software it is run.

Testing will be done through the Jasmine testing framework since jasmine has a wide variety of available tests and is well known with the JavaScript language. Tests will also be run automatically by Travis CI on pull requests to ensure that the software is in an acceptable state before merging into a different branch. This will ensure that bugs are identified before merging it into a branch that should always be in a deployable state.

The logs of Travis CI tests can be found here Travis CI.

When doing tests, it should be clear that the test is either a Unit or an Integration test.

Test Reporter

Jasmine provides a console reporter to show the results of all the tests in a structured way once it is run. It also gives a summary of the tests that run and how many have passed and failed. If a test failed it also shows you a stack trace on where the test failed as well as what the actual value from the server response is.

Structure of Tests

All files that will be used to run tests on the software will be named “*.spec.js” and will be stored under the “spec” folder in the root directory. These files will be recognized by Jasmine when the tests are being run.

Testing will be structured the way it is done on Jasmine.

The following is used in the Jasmine testing framework:

- Describe (Suite) - Describes your tests.
- It (Spec) - Tests that are run inside of a Suite.
- Expect - Defines what you expect from the server given your input.
- Matcher - Jasmine provides various matches to enable a rich testing suite, the matcher is used to compare the actual response from the server to the value in expected.

  Different matchers are explained by Jasmine here.

An example of a test is given below:

```
describe("A suite is just a function", function() {
  var a;
  it("and so is a spec", function() {
    a = true;
    expect(a).toBe(true);
  });
});
```

Tests should be written in a uniform structure that is explained below:

- All tests belonging to the same area of the software should be encapsulated by a root description, specifying what is to be tested. It should specify which class is going to be tested and if it is a Unit or Integration test.

  I.e. Server Unit Testing

- In this description, there will be multiple describes testing different functions of the software. These functions all have a different endpoint, so the endpoint needs to be specified. It should also be specified whether the request was made via “GET” or “POST”.

  I.e. POST localhost:3000/admin/login

- In this inner description, there will be multiple “its”, which will test various aspects of the response received from the server. It should make sure that the instance of the server should be running to ensure that all other tests should pass.

  I.e. server.run should be called

- Inside of every it, the expected result should be typed out to make sure that anyone looking at the report after testing knows what the expected output was and if it was received or not (this will be known when a test passes).