Testing Policy and Report for ERP coin

Team Turing

May 2018

1 Policy

We have a policy that feature branches may only be merged into the main development branch if all tests pass, and similarly for merging the development branch into master.

Our back end is based on the Node.js framework and for this reason will be tested using Mocha. Due to the nature of our system, all requests will run through the central server, this allows us to create a central point of testing. All use cases can be expressed in terms of requests that must be sent to the server, hence our tests will be request-based and can be divided into modules that are formed by the APIs provided by our server.

As stated above any changes made to the server will be extensively tested before they are merged with the main branch. The smallest units of the server can be seen as features, hence we will perform unit tests on each feature to ensure that it works as intended before merging it into the main development branch. The features will be tested again as part of larger system in the development branch as part of integration testing, once these tests pass as well then only may the development branch be merged into the main branch and testing may start again on new features.

2 Tools

For this project the main testing tool used is Mocha which is a JavaScript testing framework running on Node.js. This framework allows us to extensively test our back end, which is our systems main point of computation through which all use cases run. Our server runs on Node.js making Mocha ideal for testing its functionality.

For more information on the Mocha framework visit: `https://mochajs.org/`

3 Github

TravisCI will also be used on GitHub, Travis CI is a hosted, distributed continuous integration service used to build and test software projects hosted on GitHub. TravisCI run tests every time a feature has been merged with the development branch and before the developement branch can be merged with the main branch.

For more information on TravisCI please visit: `https://travis-ci.org/`

4 Tests

This section will focus on the types of tests run on each of our sub-systems as well as describing how they are used to ensure our system meets the functional and non functional requirements.

Currently our system has 87 test passing with none failing. The following sub-sections will describe the different quality aspects that are tested. Reference is made to a screen shot of one of our subsystem tests that was run by TravisCI that can be found under the Figures section.

4.1 Security

Security tests are focused on the requirement that no unauthorized user may have access to the system. The system achieves this by testing that a user is logged in with every request. This creates the following two possible outcomes:

1. The user is logged in, their request may be processed
2. The user is not logged in, the request must be rejected

Examples of this can be found in figure 1, on line numbered 1652, where the coin balance of a user can not be retrieved if the request does not come from a logged in user.

4.2 Validation

Validation testing focuses that our system will correctly accept requests that contain valid data and will reject requests that contain invalid data. This prevents our system from entering an invalid state or storing invalid information on the database.
An example of this can be found in figure 1 on line 1625 where a request to add a new user will only pass if the information such as the email address of the user is in the correct format.

4.3 Correctness

Our correctness tests focus on ensuring that the server performs as expected when a request to is made by a user, this includes testing that data is actually inserted into the database when a request is made such as adding a new user or logging points of where users have been. An example of this can be found in figure 1, line 1650 where a test is made to see that password of the user has been updated in the database.
Another correctness test performed is to ensure that data returned by the server to the ERP Coin mobile application is valid, an example of this can be found on figure 1 line 1653 that ensures that the server correctly returns the number of coins that a user has earned so far.

To view our full set of test permormed by TravisCI please follow the following link to our GitHub:
`https://travis-ci.org/TeamTuringCOS301/back-end/builds/406071137?utm_source=github_status&utm_medium=notification`

5 Figures

Figure 1: TravisCI User API tests