

Testing Report for ERP coin

Team Turing

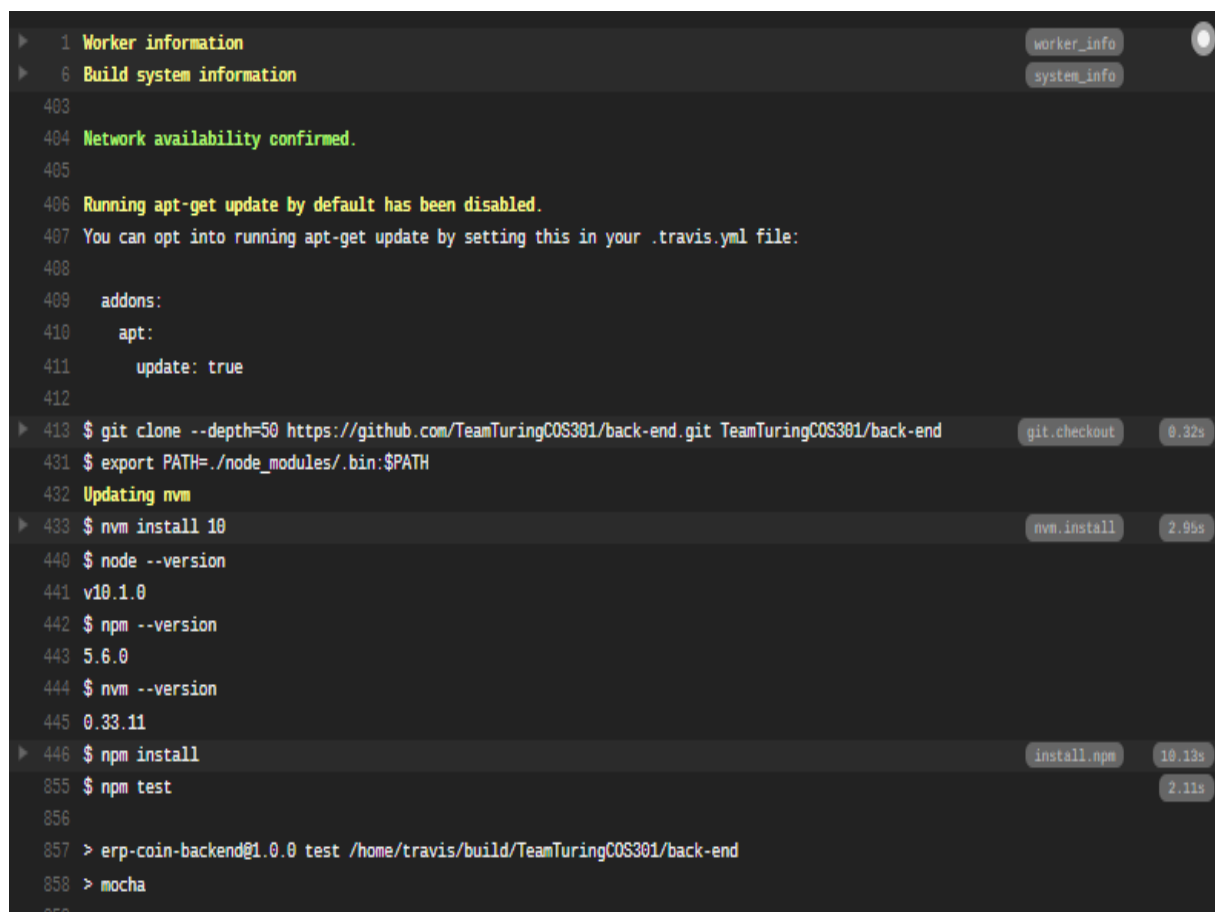
May 2018

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. The back end is tested using Mocha and SuperTest. Every implemented request has multiple unit tests to ensure that it works correctly and only works for authorized user and that our system is secure and reliable.

2 Tests

Below are the versions of the Node package manager and Node version manager that are used in our system as well as the setup of our testing environment.



```
▶ 1 Worker information worker_info
▶ 6 Build system information system_info
403
404 Network availability confirmed.
405
406 Running apt-get update by default has been disabled.
407 You can opt into running apt-get update by setting this in your .travis.yml file:
408
409 addons:
410   apt:
411     update: true
412
▶ 413 $ git clone --depth=50 https://github.com/TeamTuringCOS381/back-end.git TeamTuringCOS381/back-end git.checkout 0.32s
431 $ export PATH=./node_modules/.bin:$PATH
432 Updating nvm
▶ 433 $ nvm install 10 nvm.install 2.95s
440 $ node --version
441 v10.1.0
442 $ npm --version
443 5.6.0
444 $ nvm --version
445 0.33.11
▶ 446 $ npm install install.npm 10.13s
855 $ npm test 2.11s
856
857 > erp-coin-backend@1.0.0 test /home/travis/build/TeamTuringCOS381/back-end
858 > mocha
859
```

The tests that have been conducted on the back end are to ensure that no users have unauthorized access to any functionality of our system. Users must be logged in by use of the provided interfaces before the servers will respond to any requests. These tests also ensure that data is handled correctly and that incorrect data will be rejected to prevent an unwanted state in the server. The two interfaces provided by the server that are tested in this report are the User API and the Admin API will be used by the ERP coin app used by the general public.

The following report is for the User API where the following GET and POST requests correspond the actions:

- GET /user/add - Adding a new user to the database.
- GET /user/logout - A user logging out of the app.
- GET /user/login - A user logging into the system.
- GET /user/info - Get information about current user from database.
- POST /user/info - Used to update user information in the database.
- POST /user/password - Updates the password of the current user.

```
907
908 User API
909 GET /user/add
910   ✓ fails on missing data
911   ✓ succeeds with valid data (76ms)
912   ✓ adds the user to the database
913   ✓ hashes the password
914   ✓ sets the session cookie
915   ✓ fails with an existing username
916 GET /user/logout
917   ✓ clears the session cookie
918 GET /user/login
919   ✓ fails on missing data
920   ✓ fails for a nonexistent user
921   ✓ fails with an incorrect password (76ms)
922   ✓ succeeds with correct credentials (78ms)
923   ✓ sets the session cookie
924 GET /user/info
925   ✓ fails without a login session
926   ✓ returns the correct information
927 POST /user/info
928   ✓ fails without a login session
929   ✓ updates the user information
930 POST /user/password
931   ✓ fails without a login session
932   ✓ fails on missing data
933   ✓ fails with an incorrect password (75ms)
934   ✓ succeeds with the correct password (151ms)
935   ✓ updates the password (76ms)
936
937
938 56 passing (1s)
939
940
941
942 The command "npm test" exited with 0.
943
944 Done. Your build exited with 0.
```

The following report is for the Admin API where the following GET and POST requests correspond the actions:

- GET /admin/login - An admin logging into the system.
- GET /admin/add - Adding a new conservation admin to the database.
- GET /admin/super - Checks if the current admin is a super admin.
- GET /admin/info - Gets information of an admin from the database.
- POST /admin/info - Used to update information of an admin in the database.
- POST /admin/password - Updates the password of the current admin.
- GET /admin/list - Getting a list of all registered admins
- POST /admin/remove - Removing an admin from the database.
- GET /admin/logout - An admin logging out of the web portal.

```
862 Admin API
863 GET /admin/login
864   ✓ fails on missing data (42ms)
865   ✓ fails for a nonexistent admin
866   ✓ fails with an incorrect password (80ms)
867   ✓ succeeds with correct credentials (78ms)
868   ✓ sets the session cookie
869 GET /admin/add
870   ✓ fails without a login session
871   ✓ fails on missing data
872   ✓ fails with an existing username
873   ✓ succeeds with valid data (76ms)
874   ✓ returns the generated password (75ms)
875   ✓ hashes the password
876   ✓ fails for a regular admin
877 GET /admin/super
878   ✓ fails without a login session
879   ✓ returns true for a super admin
880   ✓ returns false for a regular admin
881 GET /admin/info
882   ✓ fails without a login session
883   ✓ returns the correct information
884 POST /admin/info
885   ✓ fails without a login session
886   ✓ updates the admin information
887 POST /admin/password
888   ✓ fails without a login session
889   ✓ fails on missing data
890   ✓ fails with an incorrect password (76ms)
891   ✓ succeeds with the correct password (150ms)
892   ✓ updates the password (77ms)
893 GET /admin/list
894   ✓ fails without a login session
895   ✓ fails for a regular admin
896   ✓ returns a list of admins
897 POST /admin/remove
898   ✓ fails without a login session
899   ✓ fails for a regular admin
900   ✓ fails on missing data
901   ✓ fails for a nonexistent username
902   ✓ fails for a super username
903   ✓ succeeds for a regular username
904   ✓ removes the admin from the database
905 GET /admin/logout
906   ✓ clears the session cookie
907
```

Top