ERP Ranger Application

Testing Policy Document
Developed for EPI-USE ERP by The Tenacious Technicians

de Villiers, Charles	Kirsten, Eric	Nel, Johan	Ross, Justin	Schwikkard, Dylan
u16056559	u16020431	u16354029	u17080526	u16120206

18 July 2019

1 Introduction

To ensure we deliver a product of the highest possible quality we put a great emphasis on testing each and every part of our system. To ensure testing is done correctly and thoroughly this document was created.

2 Testing Process

To ensure testing is done and that it is done properly, effectively and efficiently we have automated all of our tests. The goal of our testing process is to detect defective code, classes and interfaces to help reduce the amount of failures and errors the user might encounter during production.

2.1 Mobile Application

Testing our mobile application is performed with Flutter's built in testing functionality. There are 3 different types of tests being run on the mobile application. All the tests can be run once with the following command "flutter test test/full_test.dart"

2.1.1 Unit Testing

Unit testing is used to ensure that every function and class inside our app functions correctly. Unit tests can be performed by using the following command "flutter test test/unit_test.dart"

2.1.2 Widget Testing

Widget testing is used to ensure that all the application's screen are functioning and rendering correctly. This is very useful because testing each and every screen by hand would be very time consuming. Widget tests can be performed by using the following command "flutter test test/widget_test.dart"

2.1.3 Integration Testing

Integration testing is used to ensure that the application as a whole is working and functioning correctly. Integration tests can be performed by using the following command "flutter test test/integration_test.dart"

2.2 Web Application

Testing our mobile application is performed with Karma and Jasmine that are packaged along with Angular 7. There are 2 different types of tests being run on the mobile application.

2.2.1 Unit Testing

Unit testing is used to ensure that every function, class and service inside our web application functions correctly.

2.2.2 Integration Testing

Integration testing is used to ensure that the application as a whole is working and functioning correctly. This is done by testing components as a whole to ensure they are rendering and retrieving data correctly.

3 Approach to Test Process Improvement

Tests should be reviewed alongside code reviews after a sprint. Tests should be on the same quality standard as the production code released to end-users.

4 Testing Tools

4.1 Manual Testing

It is recommend that developers run the tests manually before committing code to Github.

4.1.1 Android Studio

Flutter tests can be run from inside Android studio.

4.2 Automated Testing

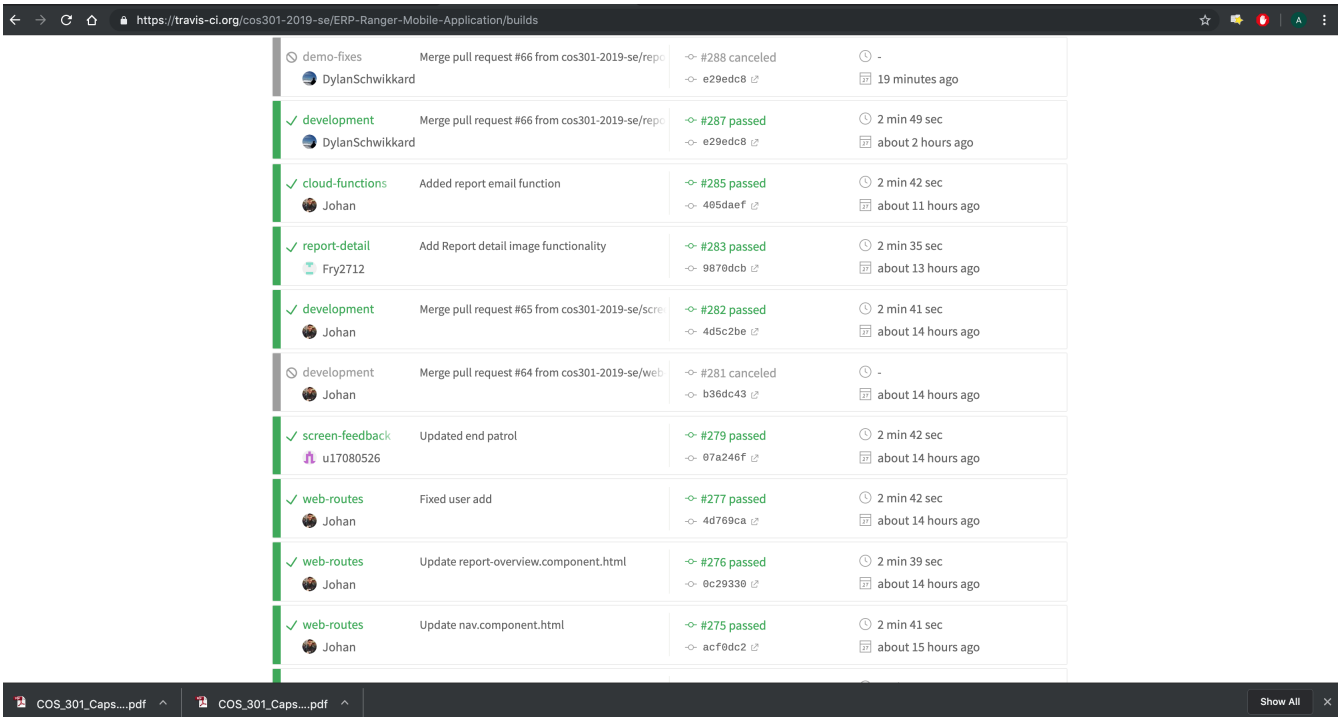
To ensure that tests are performed whenever changes are made we decided to use Github and TravisCI.

4.2.1 Github

Github does more than just managing our code base, it also provides us with an area to report issues missed by tests which were then encountered by users and it then allows us to effectively identify, track and resolve these errors. It also automatically invokes our next tool Travis CI to automatically run tests on each and every commit as well as each pull request.

4.2.2 Travis CI

Travis CI is our main testing tool. There is a testing file located in the root directory of our repository. Upon each commit and pull request this testing file is invoked. The testing file creates a small virtual machine and installs all the necessary dependencies to test our code. It then runs each and every test and sends the results to Github as well as our emails.



5 More Testing Information

- Flutter: https://flutter.dev/docs/testing
- Angular: https://angular.io/guide/testing