Team 12

Food Truck Management Application

Final Report

Comments:

If anything is unclear as to how to run the code or where certain files are located, please view the readme file located on our team's github page.

Testing Approach:

The testing approach the we used was relatively simple, unit tests in the form of Junit and PHPunit were written to test that the app was performing the intended tasks. Once all the unit tests were past, the team would then meet every week and do a code review. An in-person code review was preferable to comments on github because all members of the group had many classes together and we would all basically finish class at the same time every day, so the team had no problems with finding a meeting time. It was more convenient because in person meetings can convey more than just an issue on a github page. We discussed how to rework functionality and the look of the app to create an user-friendly experience and to make sure that the code was clean and efficient. We also did a beta test by letting some of our friends use the app and report their opinions to us. They were a big reason the application ended up in its current completed state. Integration tests were also done in order to see how individual modules would work when tested as a group. Testing for the android application was only done by beta test. No unit tests were made for it since it got its functionality directly from the java application. Since we had extensive testing for java, we had confidence in the android portion as well.

Release Pipeline:

Our release pipeline does not use any time of automated build such as ant or Travis. This is because the initial java application had been completely very early, within the due date of deliverable 2. Because of this, the application had been thoroughly tested through our Junit tests and we got the code coverage that we deemed as acceptable and we gained confidence in the system. Since the testing phase of the application was done, there was no point to add any automated testing using TravisCI. Also, we had finished the PHP portion prior to learning about automated builds and testing using TravisCI and Apache Ant. Because of this, there was no real benefit to use these systems since the development of the application had been completed. So, our integration phase was the testing process discussed above and the editing of the code and writing new code. The build phase would use gradle to generate the android .apk file and the php files would just be updated as to run the newest version on the server and browser. The jar file from the java portion of the project was built using the built-in function in eclipse and then this would be exported and put on github for the team to see and for the iterative releases.