W4156 Second Iteration Demo Report

Team: Dutchers

IA: Sara Samuel

0 Our team

Team name: Dutchers

Team members:

Zhihao Li (zl2695)
Qianrui Zhao (qz2338)
Yushi Lu (yl3974)
Qian Zheng (qz2348)

IA: Sara Samuel

1 Demo review

Our demo happened at 4:00pm on Dec 3rd, 2018 at 6LE1 Cepsr.

Challenges: the logic of division of countable and uncountable items is very difficult to implement. So in this iteration, we temporarily offer two split options, evenly splitting and collecting all, which greatly simplified the development difficulty and is practical in real use cases.

2 Progress review

2.1 What we demonstrated:

(the underlined entries are new changes from the 1st iteration)

a. The content of a receipt can be recognized and shown in the user interface.
b. The title of the receipt and name of each item can be customized.
c. Receipt updates can be shared with receipt sharers in real time.
d. Both ongoing receipts and past receipts can be displayed in the receipt history.
e. Each item of a receipt can be either split or collected all by the receipt creator, and the total price will be recalculated according to the operation.
f. The user can sign up and log in to the application.
g. The logged-in user can add friends and create chatting groups.
h. Receipts can be shared with friends.

2.2 Progress since the first iteration:

All of the use cases described in 2.1 except a and f are new functionalities added after the first iteration. And the UI of receipt page is drastically optimized.

3 CI server

For this iteration, we failed to deploy our test for the mobile part on our CI server due to some reasons. Firstly, our server, an EC2 instance running under the Amazon free trial, only has less than 8GB disk space and limited RAM space. Currently, without installing any dependency to run tests for our mobile app, there is only nearly 2GB left. We are afraid that installing all dependency (including OpenSSL, nodejs, npm, react-cli, jest, enzyme and so on) would leave no room for our server to process requests from the front-end. Also, to run tests successfully requires some manual configurations, which we have not figured out how to achieve them by running a script on our CI server.

Even though we did not have our CI server configured, we still got the coverage report successfully shown on our CI server. The reason for that reports which can be read by the CI server is not necessarily generated by tests running on the CI server. So it reads reports we generated locally and pushed to the git repository. The report is shown below.

Our unit test reports and coverage reports are supported by JUnit and Cobertura respectively.

4 Github link

https://github.com/DoDutchAoA/Do-Dutch

5 Detailed report

https://github.com/DoDutchAoA/Do-Dutch/blob/master/README.md