Talia Attar, Rachael Adelson, Tewodros Mitiku, Eric Osband
Professor Clarkson
CS 3110
20 December 2020

MS3 Progress Report

Vision: Our vision for cBay—rebranded from Cornell Marketplace—really came to fruition in this final sprint. Looking back at our anecdote from MS2, which featured Eric selling his airpod, Talia making an offer on it, and then Tedi making a higher offer that is accepting, we are now able to carry this out through the use of our server. In fact, after deploying successfully (the result of a many long hours battling between OCaml and Docker), each team member, all in different states, now can sell, buy, and make offers on each others' items, all in real time!

Summary of progress: In this sprint, we focused our energy on developing a server. To do this, we partitioned our files into a client side (including main and server other files main needs) and a server side (including our backend and database files). We examined the communication between these sides and developed an API specification for the information they pass back and forth. We implemented a server and these routes on the server side and implemented making these requests and processing the body information on the client side. After having a server successfully working, we dockerized our server side and pulled the docker image onto a Google virtual machine instance. Once we had cBay fully up and running, we added some more features such as viewing the purchases a user has made, viewing the items a user has sold, and viewing overall transaction history. Finally, we added some extra fun features such as a brand logo and color palette for the client interface.

Activity breakdown:

Talia:

- Attempted to deploy using Heroku until a wall regarding OCaml compiling across systems was hit
- Worked on making the Docker image and pushing the image to DockerHub
- Implemented some server routes (both in the API specification and the server file)
- Designed logo and OCaml display of logo
- Researched creating and implemented the server in OCaml

Tedi:

- Researched creating and implemented the server in OCaml
- Met with Sequoia Capital, YC Combinator, and Peter Thiel to pitch cBay
- Raised $250 million dollars in Seed A round of funding for cBay
- Implemented some server routes (both in the API specification and the server file)
- Researched and did the bulk of deployment using Docker
- Set up virtual machine for server and pulled and ran Docker image on machine
- Once again, resolved team conflicts by acting as CEO and CTO of cBay, leveraging years of experience in industry and academia

Rachael:

- Implemented some server routes (both in the API specification and the server file)
- Added marketplace functionality such as viewing items by user, finding offers, and finding items by user.
- Thoroughly tested marketplace, itemstate, userstate, and transactionstate functions, debugging where errors arose
- Cleaned up marketplace functions and specifications

Eric:

- Implemented the totality of marketplace functionality in our main file

- Implemented the client side of the server, making requests to and unpacking responses from the server

Productivity analysis: This sprint posed an interesting challenge to our work structure as we transitioned from the lovely private rooms in Upson to inter-state calls. However, through using zoom work sessions, we were able to mimic the feel of working and supporting each other that is critical to our overall productivity as a team. We split up internally into a backend team and a frontend team, allowing us to maximize efficiency and streamline communication. Overall, we accomplished what we set out to do and were even able to overshoot our goals and deploy our server. If terms of accuracy in predicting our capacity for MS3, these accomplishments reflect that we initially underestimated the work we would be able to get done.

Scope grade: Looking at our goals from MS2, we completed satisfactory scope in implementing the entirety of the marketplace functionality into main and learning how servers are made in OCaml through parsing the package documentation for Cohttp and Lwt (the bonus lecture from week 14 in promises also aided this understanding). In dividing our files and successfully implementing our API specification on both the server side and the client side to result in a fully-functioning server, we also completed our good scope. In regards to our excellent scope, pertaining to read/write locks, we achieved the intended functionality of read/write locks in concurrent client usage through reading and writing from the json for each request made. Overall, we believe we achieved excellent scope because we met our goals as well as successfully completing an extra step of deployment which was quite a challenge.