RoomEase SRS Postmortem

Alex Vrhel (avrhel), Sid Gorti (sgorti3) Jakob Sunde (jsunde), Omar Alsughayer (oasugher), Cheryl Wang (cwang7), Matthew Mans (mans1626), Weijia Dai (weijid)

***Note: We updated our SRS one last time so the schedule more accurately reflects the order each task was done and the approximate workload for each team member. Developer days listed on the schedule are approximate.**

Features and Cuts

We completed four of the five features we originally listed in our SRS. We chose not finish the chores module because we didn't think it contributed enough novel functionality, not covered by the other modules and we we didn't have enough time to implement it to a point where we would feel it was complete. We didn't implement any of the stretch goals we originally had because of time constraints. However, we added a couple of additional features we did not initially enumerate in our System Requirements Specification. We implemented an auto fill feature for adding a fridge item; as the user starts typing, suggestions will pop up below the input field based on items they have added previously. Furthermore, if they enter an item that they have entered before, the expiration date field is filled in automatically for them. We also added a quickadd feature, providing the user with a quick way to get from the home page of the feed view to adding an item to one of the other modules.

Cutting the chore module as a feature allowed us to more completely finish the rest of the app to a level of polish that we are proud of. We felt that completing the chores module would have stretched our group very thin, since we implemented 4 main modules over the quarter, and the chores would have been at least as much work as the reservation module, if not more. Furthermore, we thought that the chores module was the least novel of our unimplemented modules, since roommates could use shared lists to achieve a similar effect of dividing responsibilities between themselves. We estimate about 10 developer days were saved by not implementing the chore module, but this is a rough estimate.

Task assignments

Originally, we didn't have very definitive ideas of who would work on what tasks, aside from the backend/frontend split of our team. The actual roles of our team members were as follows:

Frontend: As the PM, ***Alex*** was primarily concerned with organizing and running meetings, coordinating responsibilities, helping with front end implementation, drafting documents, and debugging/building the application at each milestone. He also worked to ensure code quality, chasing down bugs and improving code structure throughout the quarter. ***Jakob*** worked primarily on frontend implementation and UI, and was our primary CSS stylist. He implemented much of the fridge and feed modules, and fixed a large number of bugs and helped to refactor code later in the quarter. ***Sidd*** was responsible for creating and maintaining the developer and product websites, as well as working on front end implementation (primarily for the fridge module and the group login flow). He also helped with chasing down bugs and fixing UI look and feel throughout the quarter. ***Cheryl*** worked primarily on front end implementation, especially on the list module. She implemented front end functionality, as

well as styling and fixing UI problems for multiple modules. She was also in charge of writing tests for a few modules. Most of the frontend team did indeed end up working primarily on frontend tasks.

Backend: For the first portion of the project, ***Matt*** worked on the back end code. He wrote the loginHandler, which handled the database communication that was required to create a group and add a user to the database, and the requestHandler, which dealt with adding, removing and getting data from the database. Later in the quarter, he worked on implementing the reservations module and helped write tests. ***Corie*** did a lot of research on PouchDB offline synchronization and optimizing our database design. She also was our primary test writer, and helped the team test the UI and write many unit tests. Finally, she helped the front end team with some of the fridge implementation. ***Omar*** did backend work for figuring out how to locally store information, and backend work on Facebook integration and group login/logout. Omar also helped write tests for multiple modules and contributed a lot to various pieces of documentation throughout the quarter. All of the team members originally assigned to backend did do backend work, but there was a larger load of frontend work than originally anticipated, so they shifted to work on other tasks, largely testing and frontend implementation, partway through the project.

Time Management

We felt that there were a number of tasks we spent too much time on. Setting up automated testing took a very long time to get right, but was not a tool that we paid much attention to or used, due to version differences in the testing framework between our automated and manual tests. Some members of our team also spent a long time trying various Facebook plugins for PhoneGap before eventually finding one that worked, so integrating Facebook took longer than we would have liked. We had a few members spend too much time working on polishing things before we were late enough in the process, and a few hours were lost when team members would polish a certain UI element or feature that was changed later on in the project development. Finally, one of the major areas where we felt we spend too much time was debugging issues with building our application. Due to rather poor documentation from Adobe PhoneGap, we encountered a number of bugs and silent failures when we built our applications for various milestones. Differences between the served versions of the application (which we used while working) and the packaged application gave us many headaches during development, and we should have been more rigorous about regularly packaging our application to check for issues.

There were also a few tasks that we didn't spend enough time on. We could have spent more time developing tests, since they are an important way to verify correctness of our product, but we had many headaches with the testing framework set up. Furthermore, many of our test implementers were not the ones who implemented the modules they were testing, which made our test-writing procedure harder than it needed to be. Also, we didn't spend as much time user testing our application as we had originally planned. This was largely a result of us being too busy implementing features and trying to meet deadlines, and feeling like we didn't have long periods of stability on the application which we could use for user testing.