Matt Riopelle, Mark Haley, David Banda

May 3, 2023

CSCI 4448

Final Project Report

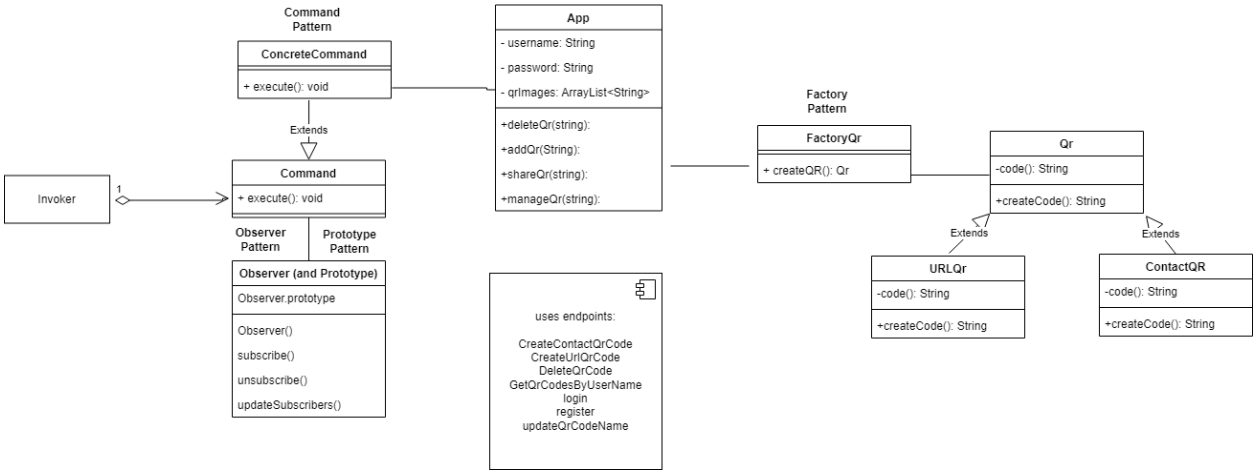
1. Name: QrCodeSocialApp

   Team: Matt Riopelle, Mark Haley, David Banda

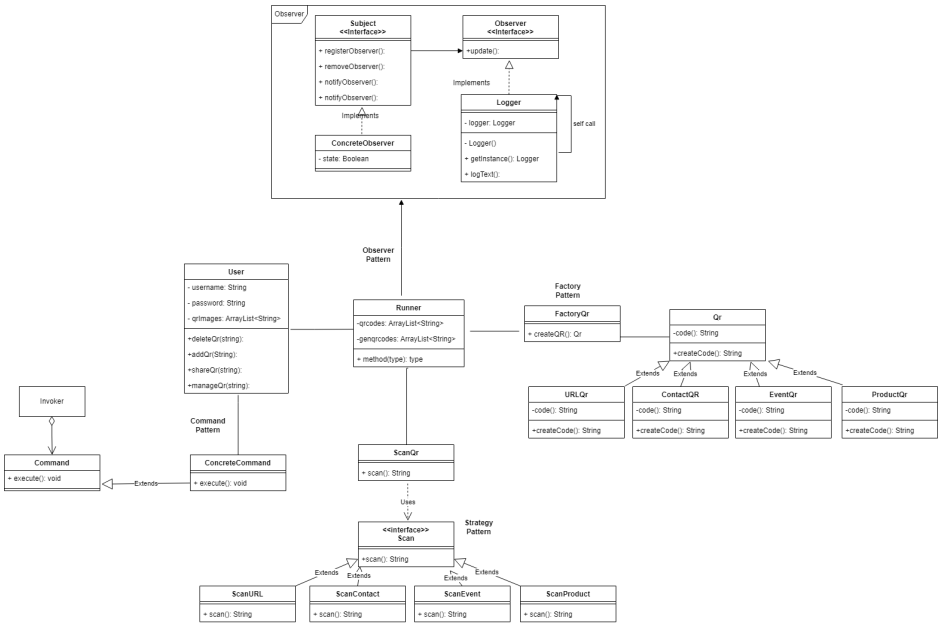
2. Final State

   Our project features a qrCode app in which users can register and login through a mongo atlas database. From here, users have the ability to scan qr codes, and to generate their own (either a URL or shareContact QrCode). It utilizes reactNative in order to generate the qrCodes, and stores the codes generated by each user in the mongo database. It allows the user to update their qrcodes as well: you can of course add new Qr codes, delete old qrCodes, modify existing qrcodes, and gather qrCodes by username. We used mongoose as our primary tool for connection to the database. Our frontend utilized xCode in order to generate a simple, yet user friendly interface. We decided to take out a couple of the Qr codes we initially had in mind, and instead just focused on the URL and shareContact qrCodes. Our scan feature no longer has a strategy pattern included with it; instead, we allocated a different pattern elsewhere (noted below).

3.



Above is the finished UML class diagram. Below is the original class diagram from project 5.



We decided to simplify our design by quite a bit. The first noticeable difference would be the combination of the User and Runner classes into app.js. This works as the centerpiece of our code design, and is associated with both command patterns, and factory patterns. The factory now only includes two types of URL's instead of 4. We also took out the strategy class, and instead added a prototype to our observer class. This all works in

tandem with the command pattern, as a prototype is used to instantiate an observer object, and notify all its subscribers when a command is executed. We felt that a more clean and simple design would be more friendly to a user this way.

4. The third party sources we used were ReactNative, Node, and MongoDB, each of which are listed below. We also used dependencies like mongoose to connect to our database, and react navigation to work with ReactNative.

   https://www.mongodb.com/atlas/database

   https://nodejs.org/en

   https://reactnative.dev/

   https://reactnavigation.org/

   https://mongoosejs.com/docs/

   The implementation of the rest of the code is original, but we used this tools heavily throughout to accomplish our final product.

5. The first key design element that I would like to highlight is our planning portion. With graduation coming up, finals for other classes, and other life duties, we found it very difficult to come together and plan out our final solution. As you can see is evident by the updated UML diagram, our initial planning process was definitely skewed pretty far away from our final product. Another key design element that was crucial would definitely be the research portion of the project. As we had learned about these patterns and code implementation using Java all semester, there was definitely some research required in order to learn how to port our ideas over to javascript. This was everything from key concepts like class and object instantiation, to learning more specific topics like pattern implementation in javascript. Although this definitely wasn't too difficult, and we

were able to learn with ease, it was crucial towards our understanding of being able to apply what we have learned towards other languages. Lastly, we found the testing process to be a bit difficult. Especially so because our frontend runs on an ios simulator through xcode, and one of our team members didn’t have a mac. So, we simply streamlined our processes, added bits and pieces of functionality, and tested, until we arrived at a final product. A lot of the issues that we ran into are easily solvable within the actual industry; however, with the complexity of managing each other's classes, work schedules, and other life duties, it certainly was a bit more difficult. That is simply because college offers quite a hectic schedule, but we were all able to work around it and come together in the end.