Raman Walwyn-Venugopal
CS 445: Final Project – Delectable
Memo

Status: The status of the project is complete and is capable of adding more features if necessary. It meets all the requirements stated in the description and the full API has been implemented.

Link to Repo: https://github.com/raman162/Delectable

Lines of Code: 838

Lines of Code (Unit Tests): 1036

Unit Test Coverage (Measured with Rspec): 100%

Cyclomatic Complexity: 5

Hours Taken To get Code Working: 60

Hours Taken To Prepare Submission: 2 - 3

Challenges Faced During this assignment:

- **Reports:** I was first unsure of what type of reports were to be generated and had them initially created in the admin class and not as an object themselves. I was also confused with whether there were to be multiple reports stored or just 4 reports to appear all the time. I decided that since orders can be placed throughout the day, a new set of reports were generated every time GET /report was called. If someone wanted to see yesterdays order report, they can just enter the start date and end date manually on the all orders report id. I also thought a server function can be later added which creates a report for each day at midnight.
- **Code Organization:** I wasn't too sure how to have the best organization for this project, this was something hard for me to do. I recognized that the admin had access to the menu, orders, and customers so I initially had them as variables for an admin class so they can be updated. However when it was time for me to develop the delivery (REST) side, I realized that I should have kept each item separate in the server. Since I realized that I needed a report class later during the project I was able to keep the reports separate from the admin class.
- **REST Implementation:** Sinatra is easy to work with in Ruby but my issue was trying to understand how to receive JSON objects, a solution was found via googling the issue once I realized it was something on Sinatra's end and not with the way I was sending the JSON objects. Another Issue I had was with writing server functions, I wasn't sure if I was doing the right thing writing some validations on the server versus writing them within the object. I realized I had to do server side validations to return errors when incorrect data was posted/putted.

Recommendations:

- **Keep TDD:** This was a wonderful part of the project, I think this needs to stay. I have developed good coding habits with writing test. I hope to improve on my unit tests with writing.
- **Release the API at the beginning:** I would have been able to write much cleaner code direct for the api if it was given earlier. Instead I spent a lot of time refactoring and writing extra code to match up with the api layer so everything can work.
- **Keep the description vague (what we got was really good):** This is very much like the real world when working with clients, it seems pretty detailed at the beginning but when actually diving into the code, there are so many edge cases that have to be considered. For not asking enough questions, I don't feel like my project is complete because I am not sure if I handled edge cases correctly.

N.B. This is absolutely my favorite project I have worked on while in school. The only way I can see something like this being better is where it involves people working in groups so they learn how to collaborate. Or people switching code at checkpoints, so people get accustomed to writing good code that they can leave to someone else and working on code that is not theirs. These are also real world scenarios that happen 99% of the time.