Lab 1

I have worked retail for the better part of the last 10 years so naturally I was thrown into many different groups of people. The people that I worked best with are the ones willing to show/teach me in areas I haven’t been exposed to and I do the same for them. Getting to know each member enough to know which member may know the answer of a lingering question has also worked out well when working in a team setting. What didn’t work was ignoring or avoiding members who you didn’t get along with. That usually resulted in instability in the workflow down the line. Also piss people off.

Let’s call our group: Group A. Everybody had questions related to different topics. Obviously we are all in the same boat so it was more or less a discussion giving rise to more questions. I suppose in an engineering setting, appointing a lead speaker/the experienced guy would of really helped settle things down initially. The objective was to simply “design an app” in a team setting so I suggested a problem from the study guide itself:

“Develop an algorithm that outputs your current location and a list of ATMs locations in that area. Get you the closest K ATMs to your location.”

I thought it was a good problem since you can hypothetically use java for it. And obviously we have all been exposed to java. So I pasted the problem on the chat. At this point we were evaluating the usefulness of solving this problem. One of the members suggested something really good actually and that is a method to find the ATMs by fees. This is definitely something an end user would like to be able to do and in general we should allow the ability to filter as the user wishes. This also led to the idea of composing documentation. So an example:

Given: Wants to see the ATMs by fees around user’s location

When: Search for ATMs

Then: Returns a list of ATMs from lowest to highest fees

So of course the app will consist of many of these types of methods. Of the group 2 of us were exposed to web app development and we proposed a design composed of the client request, a back end for our response and front end for an interactive UI the user can use to send a request.

Problem to solve: ATM

Client side:

Design search filter index by query for fees, distance, bank type

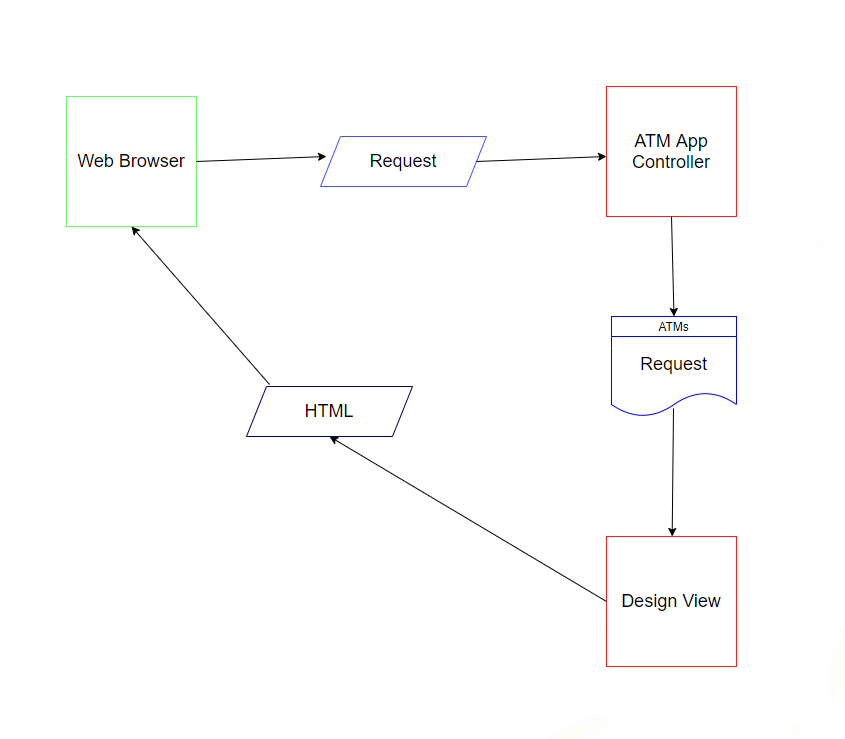
Back end:

Cloud repository with DAO to perform our queries

Front end:

Dynamic UI allowing users to select from a number of options

At this point we ran out of time actually but we did want to continue writing essential documentation like a general Model view of our app:



And discuss further about DTO that will define an ATM, a DAO that will fetch a particular ATM according to the given when then style documentation we have. After that draw up a class diagram that simply lays out our UI, Service layer and DAO.

Since we did begin from writing our methods from English text, naturally we would need to translate that into java somehow. So developing in behavior driven design would have been my preferred approach that way we have the feasibility to test whatever methods we are able to translate into our IDE and test them using a service layer stub or something. Ultimately each of us would be tasked to think of an important method relevant to the app, write the code, test it, and study its behavior ideally on a separate branch from main, write any comments in the commits and push it into github so others can review it.

As for how this would translate into a team setting, I think it would have definitely been tough since all of us don’t have experience in web based development. So we would need multiple sessions to cover what we know, what we don’t know, and along the way complete some tasks since realistically we cannot just pause development then start again in the real world. Personally, I don’t really know much front end besides your google copy paste HTML/CSS but there was one guy who seemed to know javascript so he for example would start with the UI while the others are actively working on the tasks they can do.

Not much more to say here since it’s just me explaining how I would prefer it to be like. But overall I suppose documentation followed by BDD is ideal since I suck at algorithms and BDD enables us to experiment a lot more without having to write excess code.