Conclusion

a) Lessons Learned

The biggest lesson that the group learned was the importance of getting the nitty gritty details pushed out early. This was one of the bigger factors regarding issues early in the project, as the group realized once the project was underway that there were some areas overlooked, some assumed to be easier than in reality, and other small issues between the code and design. This was not a huge surprise, as with all projects some things come out of nowhere.

Another issue was that for a lot of the group, we were using new technology. Amazon Web Services and the database was new for almost everyone. This took a bit of a learning curve, but once the Session Manager was finished, there were no major issues regarding this. The GUI also forced the group to learn some new things, such as creating a custom icon and transitioning windows.

A huge lesson learned for all of the group was effectively using GitHub and version control. Effectively using GitHub, the branches, and checking things in and out took some getting used to. Going forward, the group would be using this tool more effectively, as well as tracking work spent on a document. This was something that was lacking, and instead, the group relied upon the document’s revision chart to track this stat.

b) Design Strengths

The design that has been developed is a great application of the strengths of object-oriented coding. Each object and each part of the platform are independent of each other, as they should be in a true object-oriented world. They can be independently altered without affecting the rest of the project, making it very lightweight and flexible. If the group wants to make changes to how the interest is calculated, it can easily be done, and there is no need to potentially mess up anything else. If we want to migrate database types, that is very easy as well as only a few lines of code would change in the Session Manager.

Another strength is that the group made design choices that rendered this application able to run on almost anything, anywhere. By choosing to program in Java and use Amazon Web Services to host the database, there are no consumer computers sold anywhere in the world that cannot download this program and run it. Because Java is so flexible and because Amazon Web Services can be accessed anywhere, the application is truly one of the more accessible applications out there.

c) Limitations

The biggest limitation of this project is that there are no outside connections except to the database. In normal banking software, you can transfer money almost anywhere, pay bills online, deposit checks, or do so much more. By the fact that this is an academic application and not used in the business world, the application does not have any access to a bank’s database, bill payment systems, or any other kind of real life financial systems. If the group had more time, it would be possible to simulate some of these things, but as of now, they are not included within the project.

Another limitation is that the system assumes that you already have a username and password. There is no way to create a new account and as such forces the group to go in and add data in manually so that the application can use it. This is not a problem for an academic solution, but in the real world, it would make business sense to have a very low barrier of entry for your customers.

d) Suggestions for Future Improvement

One of the more complicated things to implement in this sort of sense would be a way to manage stocks, mutual funds, index funds, or similar markets. All of that information is easily available online and can potentially be scrubbed from the website’s public API. However, as the bank that is being simulated represents a commercial bank and not an investment bank, this would be a huge scope change. However, that would really increase the usefulness of the app. Simply amending the database, adding some new methods to the session manager and coming up with a new screen for the GUI would make this a reality.

The easiest improvement, as noted above, would be to let users create their own account. Since the system is closed as of now, it would be relatively simple to let a user choose a username, password and then stock their account with a default balance. This would allow more flexibility in creating and testing accounts.

Another improvement would be to allow transfers between the banks own customers. If for example, member A knew the account name and type of another member, they could transfer money to that other customer. This would only require a new GUI screen as well as minor changes to the session manager.

An additional improvement would be to have a variable interest rate based off of the balance of the account. Currently, there is a distinction between the interest rate of a saving and checking account, but the group could come up with a formula that would incentivize customers to put more money in their accounts. This would only require a change to the interest calculator.

Ken’s lessons learned,

The version control between even a small group of programmers is a big issue. There were times when the one team member used older versions of the code for documentation and fixed things that had already been corrected. Waiting for another team member to do their part of the code or process was difficult at times because each piece depended on the others. Having a working interface or GUI was valuable in knowing that the back end components were functioning properly. Having a GUI without the backend was difficult also, and some of the requirements such as verifying login information got duplicated.

We also learned that it is very helpful to know and be able to rely on each team member’s strengths. The group stayed motivated and built on each other’s successes and helped each other with new coding techniques and different technologies.