Introduction

Done Well:

- Our scheduler is designed pretty well, and works as intended.
- Our GUIs look quite clean and is rather intuitive
- Our early ECB diagrams helped us quite a bit with the implementation of various features throughout the project
- Our initial class diagram was well constructed early on which helped us identify relationships between classes easily throughout the development process
- The communication between team members was quite clear and concise through the duration of the project

Done Poorly:

- Although most of our design diagrams were constructed pretty well in the interim phase, as development progressed we made some design changes which could have been planned better
- Imitating student study habits turned out to be quite difficult, although we put a lot of time in planning this process for the scheduler, maybe more time would have helped
- The GUI was difficult to implement properly mostly due to lack of experience with a variety of layout managers, which left us with limited choices in what we can do to implement a flexible but robust GUI that acts like a calendar. The ideal case would have been to have more experience with different layout managers so that we could have more quickly made a decision on what was robust but flexible enough, and more robust than what we currently have implemented.
- Estimating completion dates was rather difficult for us early on as some unexpected difficulties occurred in development
- Researching libraries early on could have helped mitigate some implementation issues

Lessons Learned/Improvement:

- We learned a lot about GUIs and Java's layout managers throughout the project which will help with future UI related projects whether in Java or not
- Designing software can be difficult as things change throughout development, so we definitely learned about the process a lot
- We learned quite a bit about using GIT and how important it is in software development
    - We could definitely improve our usage of GIT in the future, specifically with branch and merge management. Ideally we should have designated a repository manager
- We could have looked more into design patterns before starting development as some may have been useful if used early on

Conclusion:

By developing a calendar application in Java, we learned a lot not only about Java since most of us were new to it, but the entire software development process. In addition, we learned about the importance of requirements engineering and the effect shortcomings in the requirements can have on the overall quality of the final software.