Learning Outcomes from the Project

Connor Sedwick

1 What technical information did you learn?

Over the course of the past nine months I have learned a great many things piecing together the documentation for this project as well as coding the graphical user-interface. From my experience working with a GUI API for the first time I have learned that time should be taken to learn and understand an API more in-depth in the early stages before its implementation is required. For instance, during the Winter term I ended up spending more time trying to understand the API we were using than I did actually implementing the widgets provided by it. I did learn much about dynamic widget creation with the Kivy API we used and have begun writing concepts for applications to which I can apply the Kivy API.

This being the first time I had done any programming in Python I also found it a very educational opportunity. Though programming with the language allowed me to learn the most, I would like to note that I also learned a great deal when I was writing the Tech Review section that focused on comparing programming languages. The most notable thing I learned regarding Python specifically is the way it handles objects and the way importing libraries is done.

Lastly, I really learned a lot regarding the use of LaTeX. I had some experience with LaTeXbefore when I was taking a previous course, but my experience in this course has improved on my skill with it. For the most part, I learned a lot regarding formatting and including figures in my documents.

2 What non-technical information did you learn?

Regarding non-technical information I learned a lot about presenting information. Near the beginning of the year I was very hesitant and poor at explaining parts of our project through speech. I tried to write scripts to read and numberings, but I still struggled. By the middle of Spring term I was finally getting more comfortable with speaking and as a result was much better at walking through my work and explaining it without requiring notes.

3 What have you learned about project work?

One of the most important things I have learned about project work is that planning is key before jumping into the actual development. Had we not come up with and settled on an idea for what softwares we were going to use we may not have made as much progress as we did.

I have also learned as a result of this project that not everything really does go to plan. We had many times where we had to rethink our implementation of some features as a result of the GUI API not being easy to work

with. Part of this was due to time constraints. Another bit of rethinking was how our interpreter would work, early on one of our ideas had to be scrapped due to it not being approved by our client.

The one really major lesson I took away from this was that it is very important to keep the client up-to-date on the state of the project. We had one incidence where we were unable to meet with the client do to no schedule overlap and as a result our client voiced some disapproval towards a private report he requested near the middle of Summer term. After making appropriate changes and arranging a meeting, we came to an understanding that the report we had provided did not cover all of our details thoroughly in the interest of document length. Consequently, the client was very pleased during the later arranged meeting that provided him with a live demonstration of our software. If this taught me anything it was that providing hard evidence of progress for the client is better than a written document.

4 What have you learned about project management?

On the topic of project work I have learned that it really helps to have people who specialize in a certain field. For instance, Behnam specialized in Python programming and machine learning which aided us in putting together the back-end as well as understand how we wanted to structure our software to support third-party libraries. Collin specialized in technical writing which really helped with editing the documentation leading up to the end of the project. My specialization was with the design and implementation of the GUI. As a result each of us took more responsibility depending on what we knew we could accomplish. We did of course split some of the work up, but because each of us had a strong background in the areas we specialized in we were able to effectively apply our talents.

I have also learned it is important to delegate a team leader to each aspect of the project. For instance, one of us was delegated for communicating with the client. Another person ended up being the project manager/lead in a way to make sure that our software was meeting its requirements. I do believe we could have done a better job with setting roles to each team member in the future.

5 What have you learned about working in teams?

From working with Behnam and Collin I have learned that it is not impossible to work well with people that I have not met before. Like most randomly assigned group projects I was nervous as to the character of the people I would be working with. After getting to know each other and work together I have learned that working in teams is really only effective if everyone understands each other. Even when we were struggling to get something completed we were able to work well together because we all had a great sense of humor and a close bond derived from our desires not to be held back another year.

As mentioned earlier, I also learned that it is a great idea to get a gauge on each team member's strengths in order to know how to delegate work and who to refer to in assistance is required on certain tasks. In all, I learned that good team can not be fragmented which is partially what occurred for us most of Winter term due to scheduling conflicts. This changed during Spring term as we began meeting regularly to work on development together rather than remotely.

6 If you could do it all over, what would you do differently?

If I could do it all over I would begin with testing out the GUI API much earlier to get a gauge on the difficulty of implementation. By far, the implementation of the Kivy API was the most time consuming part of developing the software. Sometimes taking time on the order of weeks to get a simple feature working.

What I would really like to do is to turn this software into a true Model-View-Controller layout that would greatly increase code readability and maintainability. As it is now the file organization is messy on the GUI

end and required a lot of refactoring in the code as well. I would also likely spend more time developing a file naming system that facilitated file and code organization.