Justin Tran
CS 5001
9/13/2024

Capstone Assessment

My group's senior design project will revolve around an inventory management system for the common home. It will be usable for any area in the home ranging from the kitchen, the bathroom, the garage, the basement, or even other homes or buildings. Besides the basic functionality of acting as a catalogue of items, users should be able to scan documents such as receipts or QR codes to automatically update the catalogue with ideally minimal user input or editing. We'd like our app to be accessible on both desktop and mobile platforms, such as Windows and Android, so we would need for its data to be handled on a server and accessed by clients. We are still in the ideating phase of the project, but we'd like for this app or program to be intuitive, hassle-free, and beneficial to its users in terms of not wasting perfectly good resources that they have, when they think about buying or collecting new things. Although the surface-level concept, we believe, is basic as a checklist is, we would like to incorporate more features such as automation into it to modernize it.

In terms of my curriculum, I can think of a few classes that will be relevantly helpful to this project. Particularly: Engineering Design Thinking (ENED-1120), Data Structures (CS-2028C), Programming Languages (CS-3003), AI Principles and Applications (4033), and Software Engineering (EECE-3093C) are all directly applicable to our program. Engineering Design Thinking and Software Engineering, for self-evident reasons, are helpful in helping us structure the development of our application, helping us organize our group responsibilities, and compartmentalizing the goals of a project of this scale. Although I have not completed a database design class at UC yet, my partner has; in conjunction with my partner's database experience and Data Structures class, we should easily be able to create forms of data management, both in and between the frontend/backend and databases. Similarly, although I have not completed an algorithms class, my partner has; with my partner's algorithms exposure and AI Principles and Applications class knowledge, we should be able to dip our toes into automation for our app. Last, but not least, what my partner and I both learned from Programming Languages has always served as a good basis for knowing which languages to use for our program's purposes.

Because I transferred into the College of Engineering and Applied Sciences from the College of Allied Health Sciences from within UC, my time on co-op for my Computer Science degree was shorter than my other classmates. I have only had three co-op semesters, as opposed to my partner's full five semesters; however, that does not make the experiences I gathered from my internships any less valuable. Overall, I interned at two companies: KLH Engineers for my first two rotations, and Medpace for my last (although I continued part-time for another semester afterward). At KLH, I was assigned to the software development team as a co-op. I first learned about the practical applications of iterative software development, specifically with agile and scrum there. KLH used Visual Studio, Git, C#, VB, and XAML. At Medpace, the development principles and processes were more or less the same, except I had exposure to a more diverse team, with multiple departments that contribute to a single project as opposed to only the software development team. Medpace was particularly influential to me, because at that point I had discovered that frontend, UI, and UX development was what I wanted to pursue in my future. At Medpace I mostly used Visual Studio Code, GitHub, and Angular (HTML, SCSS, and TypeScript). All

of my supervisors throughout all of my professional career were all wonderful and taught me invaluable skills through practical and real-world tasks, including, but not limited to the ones that I listed above. Besides the hard skills I developed on co-op, such as learning new languages or frameworks, I believe the most important skills I learned were in software development, like with the structure of iterative development, agile, project structure, the use of Git/GitHub with Visual Studio/Code. To me, I feel like those skills are the backbone of product development in software.

My motivation for this project stems from the fact that it is something that a few friends and I were talking about creating before this class, but never had the push to get started, as a personal project. Before senior design class, it always seemed like a faraway dream, especially since I had never worked on a project of my friends' and my own doing. But with the guidance of class, professors, advisors, and fellow classmates, I believe that it is now something a lot closer to being achievable than I could have thought before. And so, to begin with this project, my partner and I have not thought of anything concrete that we can put on paper, but we have a few ideas floating around as to what, generally, the application could be able to do. As I've been learning in my current UI class: development, especially in design or functionality, tends to go from a word cloud of ideas to more and more specific and solid concepts or models. At this point, we have not been able to enlist the help of a faculty advisor yet, so we don't really have any third-party opinions on our plans.

However, in terms of the plans we do have so far, we are hoping that by the near end of development for this project that our app should be an easy-to-use seamless manager for any kind of items of the household. We'd like to have a database set up to handle data inputted by any user of the application, a backend up and running to handle logic between the frontend and database, and finally a satisfying, user-friendly user interface that will hopefully be functional, at least on Windows computers and/or android devices. To evaluate our progress on this project, DevOps, Jira, or any other kind of task/project management software or techniques should easily be able to track our accomplishments. As for the quality of our work, I believe that to avoid the echo chamber that is just the two members of our group, or us and our advisor(s), we should be checking with other people that haven't been constantly and consistently involved in the development of our application to test things. As I've learned in previous co-op experiences and professional development classes, a program application is never truly done being developed. An application can be released or ready to use, but there will inevitably be either bugs, constructive criticism, or the application could become outdated. With that being said, to me, being "done" with this project is a goal or deadline that will have to be determined by around the time this project is ready to start being developed with code.