This senior design project is about culminating my academic experiences while at the University into a project which showcases the many facets of knowledge which I've gained here. It is to put theory into practicality and challenge myself and my groupmates to work together with our combined skillsets to create something which each individual could not as easily do alone. The project also adheres to a schedule with regular check in points, usually once a week, such that we do not fall behind, which is similar to most projects already done in my academic time and also in my work experience. This is not a one-off project which will be forgotten about the moment it is completed, but instead a project which I hope I will be proud enough of to put on my resume. It should be a good capstone for the end of my undergraduate degree and a learning experience, like everything before it.

My curriculum at UC has been preparing me to work in groups and to create comprehensive computer science projects. Classes like Software Engineering provided the basis for group software development under a deadline, which taught many lessons that lectures and exams could not. Others, like Database Design And Development, taught the foundation for how data structures are effectively stored and organized such that building software architectures upon them are intuitive and natural. Classes like Python Programming and Data Structures make you work down in the dirt of programming and help you understand how to write organized and efficient code, something highly recommended for large projects.

My co-op experiences while at UC have prepared me to utilize both industry knowledge of best practices and soft skills for working under deadlines and with a group. Clear, concise communication is the most valuable skill I've learned from co-oping and that I am still working on. From experiences at KLH Engineers working as a software engineer in working with end users and fellow developers I've created a mental model of the correct way to interact with all involved parties of a software project. I understand much more than just how to do things to get the job done, but why we do what we do so that the job becomes as simple and easy as it can be. At HII Mission Technologies as a software engineer I worked with unfamiliar technologies and had to learn on the job, something I will have to do for this project and will continue to do for the rest of my career. These experiences have greatly influenced my knowledge of the domain of computer science and the practical skills needed to make robust software projects.

My personal motivation for the project is that I hate litter. While I can go outside and pick it up, which is the end solution no matter what software solutions people come up with, I can still

contribute my skills and knowledge to help people in the fight against litter. In creating a project which can help guide and encourage people to go outside and make a difference in the world, I hope my skills can affect the world around me and not exist in a personal bubble. I believe my work should not exist purely virtually, but should be grounded in the physical world, something we as a society become abstracted from more and more everyday by news, entertainment, and social experiences being moved online. The onus to pick up the litter is on the individual, volunteer group, or city, but the friction to help start that action will be helped by this application.

The preliminary approach for this project is to understand each part of this application that needs to be built and what tools we need to build it. The next step is to then choose how to represent the different parts and the best way to do that given our scope and requirements while factoring in the deadlines of the semester. After this preliminary approach is created, to review this, we will think about whether the entire application can be made without any additions or implicit instruction in order to prove that the plan is sound. Some accomplishments include decision matrices, design diagrams, UI mockups, architecture explanations, and user stories (requirements). To self evaluate, we will criticize the project in the view of a third party so that any and all flaws are considered and evaluated for severity. If we can see ourselves using the app and can see it being used by others, we will know if we've done a good job. If people get excited at the idea and it solves a real problem, then we can consider that we've done a good job.