Discovr - Reflection Statement

Working on Discovr over the last three months has given us many opportunities to grow as developers. Through this project, we learnt how to use IDE's, coded in new languages, worked with unfamiliar frameworks, and implemented software to run analysis on our project. But more importantly, it gave us insight on how software development works in the real world with regards to workflow and teamwork within a development team.

One of the main takeaways from this project is the importance of in depth planning needed prior to the implementation phase. During normal schoolwork and projects there wasn't a need for a pre-coding planning since the "end goal" of a project is already defined by the instructor and you can immediately work towards completing it. But for an open-ended project like this, we had to consider how to design our own "end goal" and plan steps to ensure that we are meeting this criteria. While we did have preliminary designs of our app, with our requirements and features outlined, we did not have an in-depth planning or design process for each of these requirements. Without this in-depth planning, there was no consensus among the members on how the app would function and be integrated as a whole. Thus, when we reached the integration stage, there were many conflicts to be solved before code could be merged together. Overall, this led to a less effective workflow due to the lack of understanding amongst the team members.

Discovr also taught us about the realities of feature-creep. Our initial design had a lot of functionalities which we thought would be good for an "All-in-one" UBC application. This led to discussions around functions which sounded useful (such as "busyness of buildings", "event tags", or "user profiles") but provided little utility to the user compared to the complexity of its implementation. This also led us to incorrectly estimate our original development schedule. Thus, we had to readjust our schedule by reassessing the features and removing them to complete the application on time. The focus on these features also pushed us away from what the original vision was for Discovr. We feel that for future projects, we should always refer back to the original outline when proposing additional features to prevent things like feature-creep to happen.

Another takeaway is the importance of formal testing. This is often neglected or nonexistent for the sake of implementing features to satisfy the stakeholders. We found that to reach our promised milestones we were time-strapped to implement even our basic functions. As such, we did not put much thought and effort into testing what we implemented other than making sure it works during the day of the demo. A more focused effort on testing was put forth later in the project, but since most of the codebase was already written without testing in mind, it was very difficult to create tests for our methods and classes. This required us to refactor code for logic separation and learn new frameworks late in the process in order to perform even the most basic of tests. If testing was more prominent early in the project, we wouldn't have unexpected behaviour from untested code.

In conclusion, self defined projects are a learning curve for developers - especially if it's within a team. It is important to have experience with these types of projects since most of the real world work are - to a degree - self defined. It is vital to have a solid vision of the product in order to plan requirements, features, and testing that will be implemented during the development period.