Sprint 3 Retrospective
Reservoir Planning Tool
Team 16: Clayton Marshall, Drew Atkinson, Jonah Heeren, Vritant Bhardwaj

What went well during sprint 3?

1. **User Interface**
   In collaboration with the project owners, the final user interface design was decided. After two renditions, the UI is designed to seem familiar to the target users, since it based on tools similar to this. At the same time, it is also designed to be responsive and work on tablets, phones and desktops.

2. **Linting Code**
   In order to pass off code that is both consistent and follows Google's style guide. We would frequently lint our code using JSHint and the base linter built into Atom. Before officially passing off our code to the project owners we ensured that in one last sweep, all of our code conformed to the standards we had worked to maintain.

3. **Documentation**
   During this sprint one of our main focuses was making this project easy to pass off to future developers. In order to do this we each needed to do our part in adding documentation on specific portions of code we wrote for the project. This included both adding additional docs as well as commenting the code we had already written. The team did a great job documenting what they had worked on over the semester and with these documents anyone with development experience should be able to pick up where we left off with ease.

What did not go well?

1. **Browser Compatibility**
   In our final round of testing, we needed to ensure that all components of our website functioned within all modern browsers. We decided to officially support IE 11+, latest and the previous major releases of Safari, Chrome, & Firefox. Internet Explorer naturally produced some issues both with not supporting the latest Javascript syntax (Arrow notation specifically) and loading in compatibility mode on the iTap machines. We also learned that OS X versions of Firefox have an issue causing the Google Maps search bar to display results ~400px below the actual search bar. All of these issues were found very near to the deadline and should have been tested and discovered sooner.

2. **Bad data in the database**

In the last 2 weeks, the project owners discovered that some of the modeled data was inaccurate and that was the data that we were using on the live site. Because they had to get the files again, we weren't able to re-do the database in time for the demo. We have been working with them in order to re-make the database and upload the new data to the server, but we will likely not have enough time to test it.

3. **Presentation**
   The final presentation for our project did not go exactly as planned, but no presentation ever does. We thought we rushed through the presentation and missed a few points that we would liked to have shared with the class, professors, and project owners. These points weren't extremely important to the project itself, but would have given it more credibility and maybe would have increased the interest in the project amongst students and professors.

What have we learned throughout 307?

1. **Communication with non-technical stakeholders**
   Having project partners to work with gave us a huge opportunity to discuss and work with people who aren't from a development or computer science background. We had to learn enough about the research project and the goals of the project owners in order to understand their needs and transfer that into the project itself. We've learned the importance of focusing on communication and we improved in this aspect throughout the semester.

2. **UI design**
   When designing tools, one has to always keep in mind the target users. In the beginning, we designed the website to be modern and follow the new standards. But our target users were not familiar with the new trends, rather they were comfortable with basic html form inputs on a plain white background. We had to go back and redesign the UI to adapt to that.

3. **Test. Test often. Test automated.**
   With any application of reasonable size, it's easy to create unintended bugs and break things. Before adding testing, we missed several bugs that should have been caught sooner. Once we automated testing on the server, we all had a lot more peace of mind about our code stability.