Sprint 2 Retrospective
Reservoir Planning Tool

Team 16: Clayton Marshall, Drew Atkinson, Jonah Heeren, Vritant Bhardwaj

What went well during sprint 2?

1. **Organizing Individual Tasks**
   Tasks were assigned appropriately and no team member had problems understanding their tasks. As this was our second sprint we were all well acquainted with what was left to do for the project. This helped dramatically as we were able to spend more time implementing and less time figuring out how the application would work.

2. **Server Setup**
   After resolving the problems and differences with using and paying for AWS through through the ABE billing office, we decided to provision a server through ITAP. While we were initially skeptical about the stability and extensibility of a server hosted through ITAP, we were able to get a quality server with 2 virtual CPUs and 2GB of memory. This had the latest version of Ubuntu so we were able to run all of the software we need such as Node, Redis and MySQL.

3. **Improved Version Control and Communication**
   One of the things that we thought went well in the first sprint was version control and we only continued improving this during sprint 2. As each of us began further implementing individual modules we needed to heighten our attention toward versioning within project branches. Everyone on the team did a great job of consistently committing and making pull requests for the changes they made to the project. We hope to continue this behavior during the next sprint as the project comes to a close.

What did not go well?

1. **Debugging TDP Algorithm**

   A large priority for this sprint and more importantly the project as a whole was displaying accurate data. The team initially thought that this wouldn't be much of a task as a good majority of the algorithm pseudo code was given to us. We were wrong.
   Much of the time spent by our team members was working on debugging the values coming from the algorithm, meeting with the project owners to discuss algorithm inaccuracies, and working with other team members to retrieve correct values. In order to avoid this in the future everyone has downloaded a database on their local machine to allow for more effective debugging. We can also avoid this in the future by improving comments in modules that interact with one another.

2. **UI Challenges**
   The Google Maps API and the Google Charts API had various quirks that needed to be dealt with to prevent sizing errors. Fixing those quirks took longer than it should have, and so implementing a loading buffer animation and scroll animation was a challenge.

3. **Judging the duration of certain tasks**
   Some of our task durations were overestimated due to uncertainty with what the project owner's requirements would be. However, this ended up evening out as there were several tasks that we underestimated the duration of and were able to devote more time to.

How will we improve?

1. **Linting on all code**
   From now on we will use linting tools such as JSHint to lint all of our source code before we commit it to the repository. This will help us improve the quality and readability of our source, which will be important for the sustainability of the project and help any future project maintainers.

2. **Documentation**
   Currently we are documenting some of our code and have written a short overview of our project in the ReadMe. Going forward, we need to have method headers for every method that we have written and will write. We also need to add more documentation for how to run testing and how the server is set up in the ReadMe.