**Product Backlog - Team 16**

**Reservoir Planning Tool**

**Members:** Clayton Marshall, Drew Atkinson, Jonah Heeren, Vritant Bhardwaj

**Problem Statement**

There is a need to provide more secure water for crops throughout the growing season while maintaining adequate drainage during wet periods and limiting nutrient losses from drained agricultural landscapes. We will help build a web based solution by providing a visual representation of years of data, to plan out the construction of more efficient reservoirs. This planning can increase crop yield by 20%.

**Background Information**

**Audience:**

Our targeted user for this tool would be any farmers, drainage contractors, or researchers, looking to get an estimate on the needed reservoir size to increase annual crop yield and reduce irrigation deficit.

**Similar Applications:**

TheParadigm suite includes commercial options to plan the creation of reservoirs but it primarily addresses engineering concerns. Beyond that, there do not appear to be any other applications that fall into the scope of our project.

**Project Limitations:**

Accuracy of user uploaded data must be verified if it will be used to create aggregate data assumptions.

**Backlog**

**Functional Requirements:**

As a general user, I would like to…

* Open the website
* Be able to use Transforming Drainage Project’s data
* Select a location on a map
* See graphical representations of data
* See the calculated data of a region on a graph
* Upload my own data to use
* I would like my uploaded data to be calculated and shown on a graph
* Have my own data interlaced with Transforming Drainage Project’s data
* Download the results of the calculated data
* Have as less bugs as possible
* Interact with the graphs
* Have instructions on how to use the website, and how to upload my data
* Switch amongst various data sets
* Have good user experience on the website
* Have the website be responsive and adaptive

As a project owner, I would like to…

* Review data being used by other researchers
* Initialize the database
* Deploy the website
* The server to be set up
* Get feedback on the functioning of the website
* View the history and output of data provided by all users
* Be sure that there are no security-related vulnerabilities.
* Make sure that the website to be secure
* Have database to be hosted on AWS

**Non-Functional Requirements:**

*Hosting and Deployment Requirements*

Since we will deliver the application to the Transforming Drainage Project, it is important to ensure that the project can be hosted, maintained, and further developed into the future. We will have to coordinate with the project owners a way to host the project in accordance with their budget. Possible solutions for this would be Purdue web servers, which would pose limitations and restrictions on languages and frameworks we could use, but have the advantage of already being paid for by the university. Another option would be a virtual or dedicated Linux server, which would be more costly but provide more flexibility in the development. For hosting our database, Purdue ITAP will provide access to a MySQL server.

*Architecture Requirements*

We will be working with the Transforming Drainage Project leaders in order to determine an architecture that can be hosted in accordance with the deployment requirements and allow for further development by other people after the semester ends. When deciding on architecture, key requirements are that it is easily maintained, portable to different machines, and easy to learn by other developers. Since this is a web application, the client would consist of HTML, CSS and Javascript, and use standard libraries like jQuery as opposed to a complex framework like Angular or React.

*Usability Requirements*

The aim Reservoir Planning Tool is for people who have never used it before, but have knowledge of the domain, will be able to easily pick up the tool and use it. It is important that the front end of the application support many different browsers in order to maintain cross-platform compatibility. Due to the nature of our application, we will specifically target users on desktop screens, since that’s where the user is likely to have their data stored. In addition to supporting many browsers, we must also have an attractive and easy to use user interface. On the back end, the system needs to be robust enough to handle multiple users sending data at a time, and still be able to output the user’s data quickly.

*Documentation*

We must provide documentation to the user on how to use the platform, but we also must provide documentation to the project owners so that they may continue developing the platform in years to come. The documentation for the project owners would include information about what technologies we used to make the product and how we went about putting it all together.