Agri Drain Smart Irrigation Controller

Project Plan

Team	DEC1602
Email	DEC1602@iastate.edu
Adviser	Nicola Bowler
Client	Stephen Myers
Team Leader	Anne Ore
Webmaster	Griffen Clark
Key Concept Holder	Michael Parker
Key Concept Holder	Adam Wolter
Communications Liaison	Rodney Barto

Table of Contents

1. Introduction

1.1 Project Statement

Our project's aim is to create a web application to allow farmers to remotely control the amount of water that is being used to irrigate crops and to observe flow rates, nutrient levels, water levels, and rainfall totals from either a computer, tablet, or smartphone. The project will also save user preferences in terms of draining scheduling; for example, how much water is to be drained at what times of the day during different weekdays.

1.2 Project Purpose

The purpose of this project is to allow remote communication between the farmers and their irrigation systems that are mechanically implemented. It will allow more freedom and ease for the farmers to control and monitor water levels in their crops by using a user-intuitive graphical user interface. The web application that will stem from the success of this project will be have a low learning curve in order to make it convenient and easy to use. The purpose is also to bridge communication between data sent from the user interface to a server and how that data can be interpreted by the server, and vice-versa.

1.3 Goals

We aim to create several mockups of potential layouts of the web application and present them to the client for feedback. We also aim to create a proof of concept prior to the end of this term. Overall, we would like to achieve all statements previously outlined in parts 1A and 1B. In order to achieve these goals we will have a biweekly meeting with our advisor to ensure we are staying on track as well as frequent feedback from the client to eliminate misunderstanding or to account for change of desired outcomes.

2. Project Deliverables

The primary deliverable is an intuitive web application, accessible via computer and smartphone, where the user can control irrigation levels in their crops. The application will also allow users to monitor flow rates, nutrient levels, water levels, and rainfall total.

3. Design

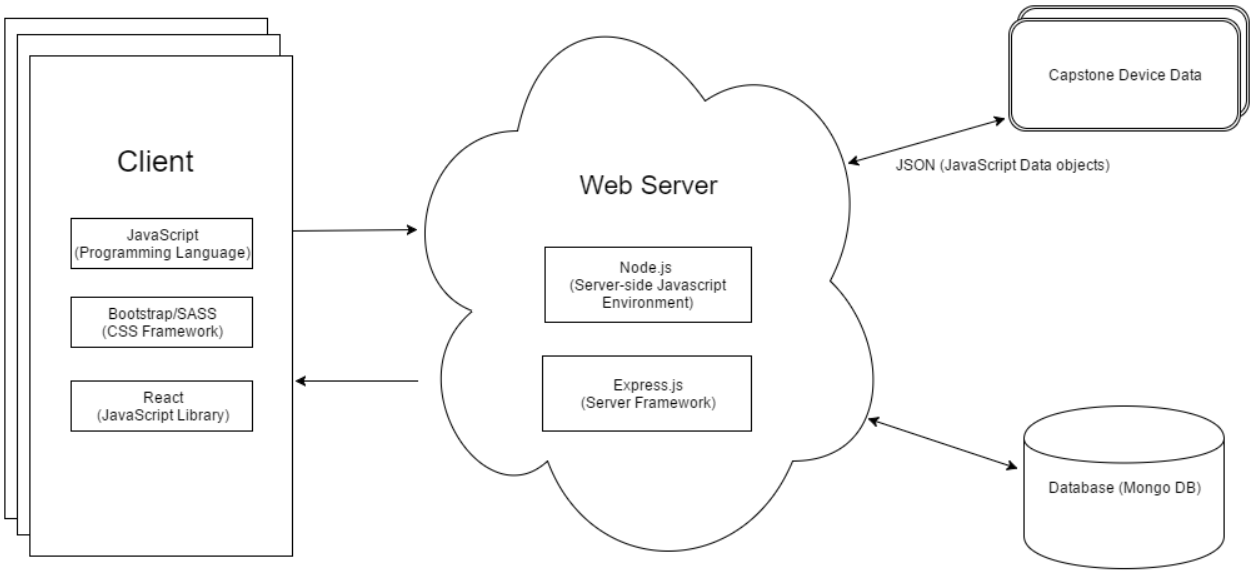
3.1 Previous Work

The tools, technologies, and frameworks that we have decided to use are widely used in the industry for web applications that are similar in nature. While JavaScript only recently became extremely popular, it is already heavily integrated in the workflow and business models of several large companies. For example, companies like Facebook, who are heavily invested into open source technology, see direct benefit by applying this investment to their most popular websites, Facebook and Instagram. This investment has manifested into the library now known as React[1] (one of the tools that we plan on using), which is now being used in the websites of hundreds of different companies, including Netflix, Dropbox, AirBnB, Yahoo, etc[2].

Another example of a company's investment into open source technology driven by JavaScript can be found in companies like Walmart, who invested in an open source server framework now known as Hapi. One of the main reasons for this investment was because they were unable to find a pre-existing framework, in any programming language, that could handle the amount of traffic they expected to receive on Black Friday[3].

The point that is being driven at here is simple: For all intents and purposes, our project is a simple web-based dashboard that communicates with a server, which in turn, communicates with several different drainage controllers. Since the hardware portion of this project is under the jurisdiction of the Agriculture Engineering senior design group, the main goal for our team is to create a web server with an accompanying website that can communicate with it, which realistically can be done in almost any programming language. The reason we chose JavaScript is because of how easy it is to use, coupled with the fact that there are several large companies that have enough confidence in the language to warrant the migration of entire platforms in order to use it.

3.2 Proposed System Block Diagram



3.3 Assessment of Proposed Methods

As mentioned in section 3.1, there are several different types of approaches that are available for designing a web application. There are a lot of popular programming languages that are used all around the industry, but none of them offer the same types of benefits as JavaScript, which is why we are choosing to go with that language.

One of the primary benefits that JavaScript offers is the lack of context switching between server-side and client-side code. As it stands, JavaScript is the only programming language that can run on both the server and the browser, meaning that by adopting it, we don't have to constantly switch between two different languages when developing our application. This allows us to spend more time focusing on one programming language instead of diverting our attention between two or three, and it makes for an overall cleaner codebase, which will benefit our client in the long run after our project has completed.

When using JavaScript, we have hundreds of different libraries and frameworks to choose from. A lot of them can be used for many different things, and while it's certainly possible for us to use React with several other frameworks, the technologies that we have decided upon were chosen simply because they were very easy to use and integrate with each other, and a few of our group members have pre-existing experience with them.

3.4 Validation

We plan on designing acceptance criteria for all modular aspects to this project. Ideally, we would like to set up a continuous integration server, such as Jenkins, to make deliverable builds

and schedule daily automated tests to ensure our product meets that acceptance criteria. These scheduled builds will be based on a strict repository workflow that we as a team will abide by.

This continuous integration will be done by using a git repository, with an integration branch that will merge all of our individual code changes. Once this integration branch meets our testing acceptance criteria, it will be merged into a master branch in which a deliverable build can be made.

The automated tests will be partitioned based on the modules of the total project. These tests will be based on user-story driven test case scenarios that will be developed earlier on in the process. These can be categorized by issuing black box testing (having testing criteria with specified user inputs and checking that the outputs are correct), and white box testing (testing on an design level to make sure these modules and interfaces communicate correctly and efficiently with each other).

We will appoint a quality assurance team member to continuously ensure that our code implementations align with our client's initial requirements and specifications.

4. Project Requirements/Specification

4.1 Functional

The following are functional requirements for the application:

- This project needs to allow users to remotely change the settings on an Agri Drain device.
- The users need to be able to see current and historical sensor data.

4.2 Non-functional

The following are non-functional requirements for the application:

- This project will be a web application, but it needs to be easily usable on mobile devices.
- We will need to have an intuitive user interface.
- The historical data should be in a format that is easy to back up.
- The code should be documented well enough that future developers can add to it.

5. Challenges

One of the biggest challenges is our lack of experience with irrigation or water drainage systems. It will be hard for us to put ourselves in an end-user's shoes; we don't know what

features should be prioritized, or how features should be arranged in the user interface. To reduce this risk, we will have Agri Drain approve our mockups before we write any code.

There is some risk associated with receiving timely information. Our client is not someone we frequently see, but we need to communicate with them frequently, for the reasons in the previous paragraph. We will mitigate this risk by arranging regular meetings with our client.

We will likely discover more challenges as we make more progress.

6. Timeline

6.1 First Semester

First semester will consist of moderate communication between the team/advisor and the client. Much of the first portion of the first semester will be continually refining requirements and specifications for verification. This will entail design mockups with feedback from client, architecture designs, a backlog of items to implement in the project, process tools for agile work process such as Jira, Crucible for code reviewing, and Confluence as a wiki for design and requirement documents so we will have everything prepared to start implementing the project. We also hope to be able to have a proof-of-concept that we can demonstrate to our client by the end of the first semester.

6.2 Second Semester

Second semester will mostly consist of the bulk of implementation work. We decided to use an agile work process. This will include bi-weekly sprints with a list of backlog items that will be a collection of minimum viable deliverables, bi-weekly sprint planning to determine how much work will be put into the sprint and who will be assigned what issues, bi-weekly sprint review and demos with client and advisor, bi-weekly backlog grooming to point estimate items in the backlog, quick 15-30 daily meetings of sprint updates among the team.

7. Conclusions

Our project's main deliverable is creating a web application to allow Agri Drain's customers to remotely control smart irrigation devices. This will need to have an intuitive user interface that is usable on both computers and mobile phones. Our team has the technical knowledge and design experience to make a good web application, but we aren't farmers, so we won't know what is best for them without speaking with our client extensively; this will likely be our biggest challenge.

8. References

1. https://facebook.github.io/react/
2. https://github.com/facebook/react/wiki/Sites-Using-React
3. http://www.infoworld.com/article/2608897/open-source-software/walmart-s-investment-in-open-source-isn-t-cheap.html