Second Creek Monitoring

April 25, 2016

Ben Parrott
Haley Whitaker
Aaron Young
Ryan Wagner
Saajid Haque
Richard Fagan

Customer: Dr. Jon Hathaway

Contents

1 Executive Summary **3**

2 Requirements **4**

3 Change Log **7**

4 Documentation and Design Process **8**

4.1 Allocation of Tasks 8

4.2 Research 8

4.3 Alternative Solutions 9

4.4 Selected Solution 10

4.4.1 Setting up Microsoft SQL Server 10

4.4.2 Pushing Data Through Flowlink 11

4.4.3 LoggerNet and LNDB 12

4.4.4 Processing Data in SQL Server 12

4.4.5 Hosting the Website through Django 12

4.4.6 Website Creation 13

4.4.7 Data Retrieval 13

4.4.8 Data Display 14

4.4.9 Explanations of Data 14

4.4.10 Aesthetic Appeal 15

4.5 Completion of Requirements 18

4.6 Results and Outcomes 18

5 Lessons Learned **19**

5.1 Overcoming Challenges 19

5.2 Communication 19

6 Statements of Team Members' Contributions 20

6.1 Ben Parrott 20

6.2 Aaron Young 20

6.3 Saajid Haque 20

6.4 Haley Whitaker 21

6.5 Ryan Wagner 21

6.6 Richard Fagan 21

7 Signature of all Team Members and Customer 23

1 Executive Summary

Dr. Hathaway from the Department of Civil and Environmental Engineering has an ongoing effort to collect high resolution data from Second Creek and a weather station located on the University of Tennessee-Knoxville campus. Dr. Hathaway would like to host a website which students and the general public can navigate to and see the conditions of the local environment and creek, thereby promoting water quality and watershed awareness. The website will provide a user-friendly, aesthetically pleasing interface and better access to real time information about the local environment. Another goal of this project is to ensure data security. It is critical to minimize any data loss that could potentially affect the ongoing research.

Several unique and interesting challenges have presented themselves over the course of this project. One such unexpected challenge was a communication issue between a new sensor placed in Second Creek and the software that we were using to send the data to the database. This error required a representative from ISCO (the company that manufactures the sensors) to try to troubleshoot the problem and develop a solution. The solution to this problem is still ongoing as company help is needed in order to fully solve this problem.

Another challenge involved pushing the data collected by the sensors in Second Creek into the database. A communication issue occurred when the ISCO device at the creek tried to push data to the computer in the hydraulics lab in the John D. Tickle Engineering Building. This communication issue only manifested itself on that particular computer and only when the computer was connected to the internet by means of an Ethernet cable. The root cause of this problem was eventually tracked down to the delay of the wireless bridge between the creek and University of Tennessee network. This delay resulted in the packets from the flow meter to timeout before they could reach the server. An interesting solution to this problem is to have the server consistently ping the flow meter, which causes the bridge to remain active.

Even though a few obstacles had to be overcome, this project has resulted in an aesthetically pleasing front-end website design with a robust back-end to support the website. A Microsoft SQL Server holds all of the data from the sensors in Second Creek and the weather station, which is then supplied to the website. The data is displayed in a mixture of current value tiles and week-long display tiles. Pop-up explanations appear if the upper right corner of a tile is clicked on, ensuring that the data is easily readable and understandable by users. The primary goal of this project is the promotion of watershed and environmental awareness; the created website, located at hydrolab02.engr.utk.edu, should certainly assist in this process with the hope being that people will continue to use this website for years to come.

2 Requirements

1 Back-end Requirements

1.1 Data Collection

1.1.1 An EXO Advanced Water Quality Monitoring Platform shall have sensors to collect the following information from Second Creek:

1.1.1.1 Blue-green Algae Phycocyanin (PC)

1.1.1.2 Blue-green Algae Phycoerythrin (PE)

1.1.1.3 Conductivity

1.1.1.4 Depth

1.1.1.5 Dissolved Oxygen

1.1.1.6 fDOM

1.1.1.7 pH

1.1.1.8 Temperature

1.1.1.9 Turbidity

1.1.2 An Isco Signature flow meter shall collect the following information from Second Creek:

1.1.2.1 Flow rate

1.1.2.2 Stage (height)

1.1.3 A weather station utilizing Campbell Scientific sensors shall collect the following information from the University of Tennessee - Knoxville agricultural campus:

1.1.3.1 Relative Humidity

1.1.3.2 Soil Temperature

1.1.3.3 Air Temperature

1.1.3.4 Wind Direction

1.1.3.5 Wind Speed

1.1.4 Data shall be collected from the EXO Advanced Water Quality Monitoring Platform at least every two minutes.

1.1.5 Data shall be collected from the Isco Signature flow meter at least every two minutes.

1.1.6 The creek data shall be transmitted from the creek monitoring box to the server via NanoStation.

1.1.7 The weather station data shall be collected at least every two minutes.

1.1.8 The weather station data shall be transmitted to the server via a Campbell Scientific CR6.

1.2 Data Storage

1.2.1 The data shall be stored in a SQL database.

1.2.2 The data shall be backed up to an additional SQL database.

2 Front-end Requirements

2.1 The data collected from the creek and the weather station shall be displayed on a website.

2.2 The following data values from the creek shall be displayed on a website:

2.2.1 Conductivity

2.2.2 Depth

2.2.3 Dissolved Oxygen

2.2.4 pH

2.2.5 Temperature

2.2.6 Turbidity

2.2.7 Flow rate

2.2.8 Stage (height)

2.3 The following data values from the weather station shall be displayed on a website:

2.3.1 Relative Humidity

2.3.2 Soil Temperature

2.3.3 Air Temperature

2.3.4 Wind Direction

2.3.5 Wind Speed

2.4 The creek data and weather station data specified in sections 2.2 and 2.3 shall be displayed in the following manner:

2.4.1 Each unique data value shall be presented with an accompanying graphic tailored for the nature of the data value.

2.4.2 Each unique data value shall have an accompanying, non-technical description explaining the nature of the data.

2.5 The website shall be compatible with Firefox 43, Chrome 48, Internet Explorer 11, and Safari 9.

2.6 The creek and weather station data website shall be linked from Dr. Hathaway's website (http://hathaway.utk.edu/).

3 Change Log

April 14, 2016 - Requirement 2.2.8 required the display of the "stage" data value on the website. However, after consulting with our customer, we learned that the stage measurement of Second Creek is the same as the depth measurement which is already being displayed on the website. Therefore, another tile will not be created for the stage data value.

April 14, 2016 - Requirement 2.4.1 said that "each unique data value shall be presented with an accompanying graphic tailored for the nature of the data value." After discussing the graphs with our customer, he specified that some of the data does not need both current and week-long displays on the website. For the creek information, the following data does not need to be shown in one week graphics: pH, conductivity, and dissolved oxygen. For the weather information, the wind direction does not need to displayed on a one week graphic.

4 Documentation and Design Process

4.1 Allocation of Tasks

In order to decompose this project into manageable pieces, the six person team was split into two groups with one group focused on the front-end of the website and one group focused on the back-end of the website. The back-end team is comprised of Aaron and Ben. The function of this group was to create a database to collect the data gathered by the sensors in Second Creek and at the weather station and also to host the website on the server with Django. Flowlink software interfaces with the sensors in Second Creek, and LoggerNet interfaces with the sensors at the weather station, both of which the back-end team has had to learn in order to utilize them effectively in the data retrieval process.

The front-end team is comprised of Ryan, Richard, Saajid, and Haley. Since Ryan has the most experience with web development, he has been leading the effort by initially creating the website which the rest of the team has helped design and optimize. The front-end team is focused on displaying the data from the database in a user-friendly and aesthetically pleasing format. Google Charts was chosen to display the week-long data and so this group has had to familiarize themselves with the functionality of Google Charts in order to display the information on the website effectively. In order to create the website, a combination of HTML, Javascript, and CSS was used which is what the front-end team was focused on. Once data was able to be displayed on the website, the front-end team shifted gears to focus more on the aesthetics of the website and making sure that everything will be easily understandable to anyone who navigates to the website.

4.2 Research

The main topic that needed to be researched in regards to this project involves the software that pulls in the data from the sensors. Flowlink is used to gather the data from all of the sensors that are in Second Creek. The Flowlink software pushes the data from the creek to the database, so making sure that this step works properly is essential to being able to display meaningful data on the website. Our research indicated that Flowlink is only compatible with a few databases. Our team decided to use Microsoft SQL Server because it was compatible with Flowlink, it works well on a Windows operating system, and our team has prior experience with it. The flow of data is shown in Figure 1 below.

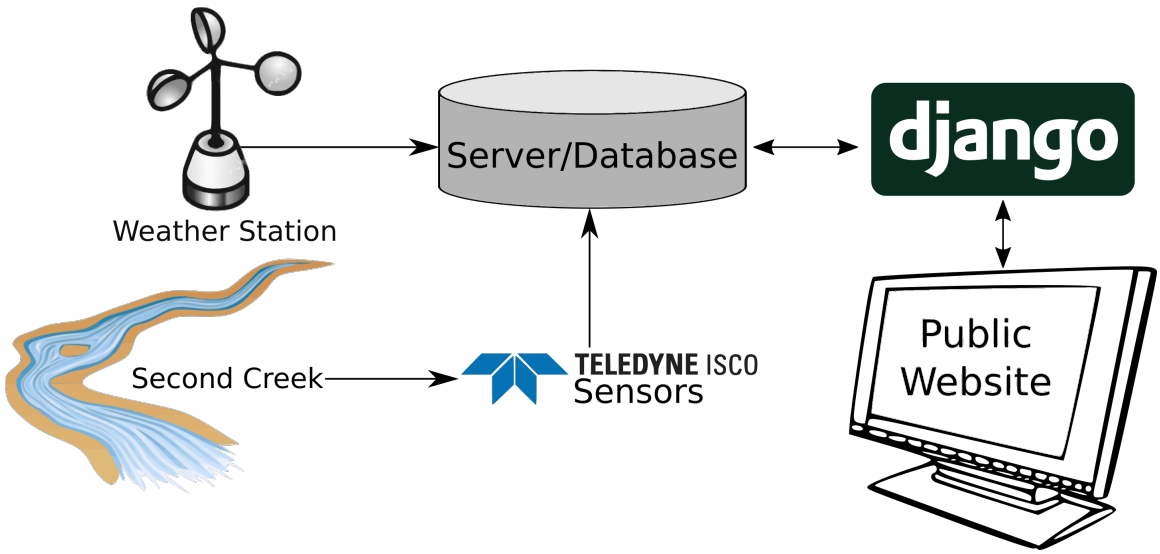

Figure 1: Data Flow Diagram

Initially the website was only going to display values on our website collected by the sensors in Second Creek. When the data from the weather station was added to the project, research into how the data from the weather station was already being collected was done. Dr. Wesley Wright, a professor in Bio-systems Engineering is currently managing the weather station that is located in the University of Tennessee Gardens. He provided information on the software, LoggerNet, that is used to gather the data. Later on, research was also done in order to determine the best way to push the weather station data into the database.

Lastly, research was done into how to display the data collected in a user-friendly and aesthetically pleasing way. Our customer recommended Google Charts so we looked into its capabilities and limits. Through this process, much information was gathered on the different types of charts that can be utilized from scatter plots, bar graphs, gauges, etc. In order to make the website interesting to look at, utilization of many different types of charts will be necessary. Google Charts provides much of the code describing how to make the charts along with examples of how to modify them. These modification include sizes of objects; colors of data points, backgrounds, and labels; legends and keys; and many more. This flexibility allows for us to have greater control over exactly how the design of the website looks.

4.3 Alternative Solutions

One major choice made in this project was the decision to use Microsoft SQL Server as the primary database. There were a couple of alternatives considered before this decision was made. MYSQL was the first consideration, since our team has had the most experience with it, and it is free to use. However, it was discovered that MYSQL is incompatible with

Flowlink which made it a poor choice. The other major consideration was Oracle. Oracle has many features in common with Microsoft SQL and is also compatible with Flowlink. Ultimately, Microsoft SQL Server was selected since it is the product with which our team was most familiar.

For the weather station component, Dr. Wright mentioned an issue with the LoggerNet software: he had to be logged into his computer in order to collect data continuously because the software has to be open and running to gather data. This was an issue as real time data is needed for the website, and due to security reasons, the team decided that the server computer should not be logged in at all times. One alternative that was explored was running the program as a Windows Service so that the software would run even when not logged in and then finding a way to move the data into the database. Unfortunately, this method proved challenging. Another idea that was discussed was asking Dr. Wright to create a data table for us from his computer because he was already collecting data continuously by keeping his desktop logged into in his locked office. Then we would have received the file from him via SharePoint. However, we still ran into the issue of putting this data into the database, and we preferred not to rely on someone else for the data for our website especially in case of any troubleshooting issues in the future. Consequently, neither of these options were selected because using LNDB was found to be the best solution.

We considered options other than Google Charts to display our data. One option we considered was D3.js, a JavaScript library for manipulating documents based on data. D3 provides a lot of potential to create impressive and appealing graphics and charts; however, it has a higher learning curve and would have increased development time significantly. We also considered using Bootstrap to create a responsive web page which reorganizes and re-sizes according to the user's screen size. We hoped that Bootstrap would simplify the implementation of our websites tiled theme. Unfortunately, we discovered it was not sufficient for our needs, and we had to create all of the necessary CSS manually.

4.4 Selected Solution

4.4.1 Setting up Microsoft SQL Server

The first task involved in getting the data collection to work is to set up a database in which to hold the information. Our team used Microsoft SQL Server. Installation of SQL Server is guided by the installer; in the installation process, it is important to remember to used mixed authentication because SQL Server logins are required for configuring Flowlink. In other respects, the defaults can be used during the installation process. During the installation process, the login, password, and Server instance name should be noted since they will be necessary to configure Flowlink and LoggerNet Database Software (LNDB). Additional information was also provided by the Flowlink documentation, which specified how to setup the database so that the Flowlink program would be able to add and modify the tables pertaining to the creek data [1].

4.4.2 Pushing Data Through Flowlink

In order to aggregate data from the creek sensors, ISCO provides proprietary software called Flowlink Pro. The Pro version allows for this data to be placed into a database. Although this task was initially considered to be relatively simple, it turned out to be quite challenging due to software and network problems encountered along the way.

The first step is to create the necessary tables in the database for Flowlink to fill. For this purpose, Flowlink provides a tool called Flowlink Server Configuration Utility. You will provide the SQL Server instance, the login information for the SQL Server, and the IP address of the computer. Additionally, a new user name and password for the server will need to be specified. Specific details concerning this process can be found in Flowlink Pro's installation guide [1].

After the database is set up, an IP Listener needs to be created. This can be configured again through the Flowlink Server Configuration Utility. The purpose of this action is to tell Flowlink to listen on a certain port for data packets from the flow meter. The IP listener is configured to add data received from packets directly to the Microsoft SQL Server database. More details on setting up this IP Listener can be found in the installation guide [1].

Additionally, to get data to push to the server, the flow meter needs to be configured with the IP address, port number, and pushing rate desired. The team used Flowlink to configure the flow meter to send packets to the server every 5 minutes. This configuration can also be done by interacting directly with the flow meter. It is important to note that due to a bug in the Flowlink software, the data push settings sometimes are not saved; entering and exiting the help page from the data push configuration window often causes the data push settings to save. Although this is not a documented feature of Flowlink, this trick nevertheless helps in such cases. When the settings are saved, the user should see a message from Flowlink indicating a successful configuration.

After doing the above actions our team experienced network issues. The problem encountered was that the flow meter never seemed to send packets to the server. When other computers where similarly configured, the above steps worked without issue and debugging with a wireless adapter also worked. The cause of this problem was determined to be the delay of the wireless bridge between the flow meter and the University of Tennessee's network. This delay resulted in the packets from the flow meter to timeout before they could reach the server. A interesting solution to this problem is to constantly ping the server, causing the bridge to remain active. Our team formalized this simple solution by writing a Windows Batch Script to constantly ping the server. Using Windows Task Scheduler, this script was made to run in the background when the computer is powered on. This keeps the flow meter communicating with the server and makes the data pushing successful.

4.4.3 LoggerNet and LNDB

LoggerNet is the program provided by Campbell Scientific to communicate with their weather station sensors. The University of Tennessee already has a license to the LoggerNet software and Dr. Wesley Wright installed and configured the LoggerNet software on our server so that it would be able to directly collect data from the weather station. He also setup the collection program that runs on the weather station so that the data we wanted to display on the website would be collected into a table. LoggerNet is designed to collect data into a plain text file, and it is unable to import the data directly to a SQL database. After research, our team found a companion program, provided by Campbell Scientific, that connects a running LoggerNet instance to a SQL database. This program is called LoggerNet Database Software—or LNDB for short—and through its use, data from the weather station is easily stored into tables in the database.

4.4.4 Processing Data in SQL Server

The system uses a set of recurring SQL queries process the data in the database. The purpose of this processing is to reduce the size of the data set such that it is feasible to send it to the client for display on the website. It was decided that hourly and daily averages would be appropriate for this purpose. These values are stored in each database under tables called Hourly and Daily.

The queries selectively average only the values which need to be updated. This involves averaging all data points from two hours before the most recent item in the Hourly table. Each query starts by deleting the "stale" values in the Hourly and Daily tables, re-averages them, and inserts them into the table again. There are two scripts, one for the second creek database and the other for the weather station database. In addition to these two scripts, two other scripts were developed which clear the Hourly and Daily tables and recalculate all the values. These scripts can be used to maintain the database, should the tables get into a bad state.

A Microsoft SQL Server feature called SQL Agent is used to schedule the queries. A job is created for each script and each job is set to run at a specified interval. At this point in time, the jobs are scheduled to run every 10 minutes, but this interval is trivial to configure. Running the jobs more frequently will increase the responsiveness of the history values but at the cost of computational resources.

4.4.5 Hosting the Website through Django

The website is hosted using Django with Windows IIS. Django is the web framework and IIS is the web server. In order for Django to communicate with the MSSQL server, a package called Django MSSQL was used. This package allowed Django to be configured to

communicate with MSSQL and use Django's database model interface to retrieve data from it. The Django MSSQL package has strict dependencies which limited the versions of Python and Django that could be used. Therefore Python 3.4 (64-bit), Django 1.7, and PyWin32 build 220 (64-bit) were installed. All of these versions work together and we were able to setup Django and connect to the MSSQL server by following the instructions listed in the quick-start guide for Django MSSQL [2].

Windows IIS is provided for windows as an optional feature that can be added. It was enabled on our computer through the "Turn Windows features on or off" control panel dialog. In order for IIS to host the Django website an intermediate protocol was also needed. Our team decided to use FastCGI since it has high performance and it is easier to setup. IIS and FastCGI were both setup by following the instructions provided by Microsoft Azure documentation [3]. Although Azure was not used for this project, this documentation adequately explained how IIS, FastCGI, and Django are all setup to work together.

4.4.6 Website Creation

To display our data, we decided to use a combination of Google Charts and our own graphics. The large tiles on our website are containers for Google Charts which show ranges of data, and the small tiles contain graphics showing the type of data displayed and a single data point. Because Bootstrap was insufficient for our tiling needs, we used it solely for the buttons at the top of the page. We designed the tiles in such a way that they will re-size to fit the users browser window, even if the user adjusts the size after the website has loaded. There are two different view modes depending on the browser window size which determine how many tiles will fit on a row of the screen. On smaller screens, two small tiles or one large tile will fit on a row. Larger screens can fit four small tiles, two large tiles, or two small tiles and a large tile on each row.

We tried to minimize the number of external dependencies required by our website. Our final list of dependencies not served from our own server is: Bootstrap, Google Charts, and jQuery. Google Charts is, of course, a necessary dependency, and jQuery can be expected to remain backwards compatible and available for our use. If there is ever an issue with Bootstrap, our dependence on it is minimal and it should not be difficult to replace.

The management of each individual chart or tile has been simplified as much as possible. As a result, it should be simple for the website to be modified after the end of this semester. Data management and positioning is handled automatically, and only chart specific options need to be specified to add or modify charts.

4.4.7 Data Retrieval

The website requests measurements from the server periodically in order to keep the data on the web page up to date. These requests take the form of http requests. The

requests contains information specifying the database to pull data from (either the creek data or the weather station data), the ID of the data to pull, and the range of data points to collect. If this range is -1, only the current value is returned. Otherwise, the range specifies the number of weeks from which to return data. For ranges which are not -1, a table is also specified indicating which table (hourly or daily) the data should be pulled from. Different measurements were assigned different ID's. The Flowlink software comes with it's own measurement ID system which this project adopted for the creek data. The weather station data uses ID's based roughly on the column index in the data base. Other special measurements, such as rainfall in the past 24 hours, were assigned their own ID and were handled specially in Django.

4.4.8 Data Display

Our customer has the desire to display a combination of measurements on the website. Both current values taken by the sensors and a full week of data will be displayed on the website. In order to achieve this, we used rectangular tiles to display the week-long data charts and smaller square tiles that are half of the size of the rectangular tiles for the display of singular data values. For the longer data graphs, Google Charts is used to display the data through a combination of bar graphs, line charts, and scatter plots. The single data values are displayed mostly by pulling a single data value and displaying it in the center of the square. For a select few graphs, a gauge or single value chart were used.

4.4.9 Explanations of Data

In order to assist viewers in understanding what the data displayed on the website means, pop-up boxes were added to display information including what the measurement determines and what a typical value for the measurement might be. For example, the pop-up explanation for flow rate as shown in Figure 2 below includes information about what flow rate measures, why measuring the flow rate of a creek is important, and how to quantify the information that is being shown. This information is provided in order to assist user's in understanding the data that is being presented.

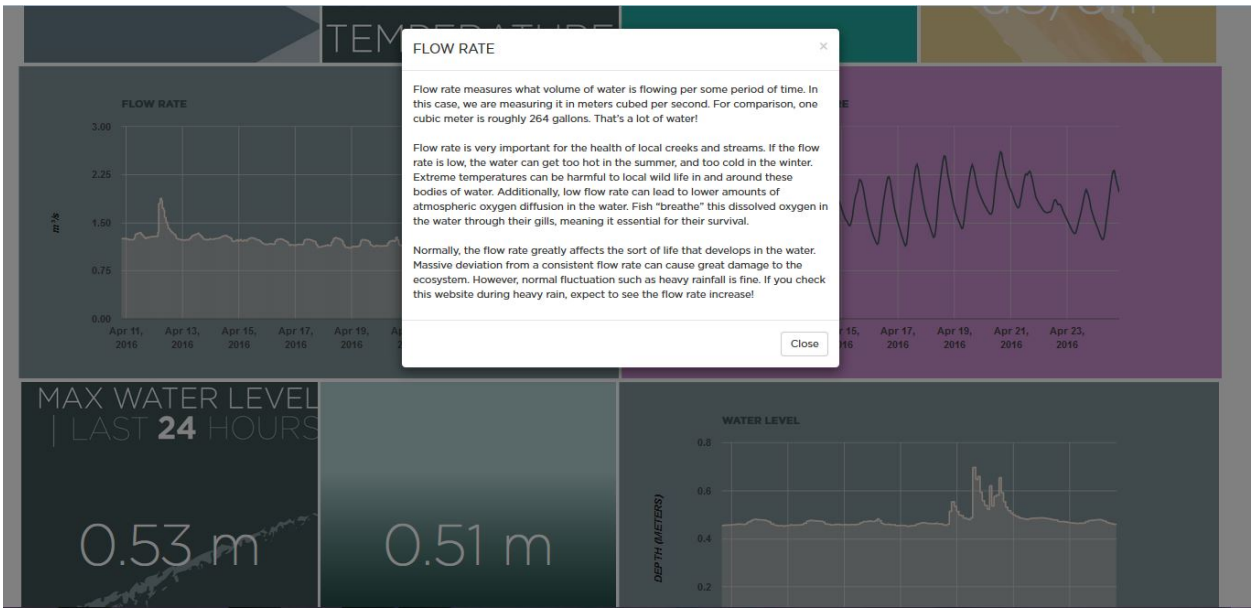

Figure 2: Pop-Up Explanation

4.4.10 Aesthetic Appeal

Our team conferred with design consultant FITZ + HEM over the aesthetic appeal of our website. Color palettes, fonts, and general design were discussed. We choose an abstract minimalism approach for the design of the website as can be seen in Figure 3 and Figure 4 below.

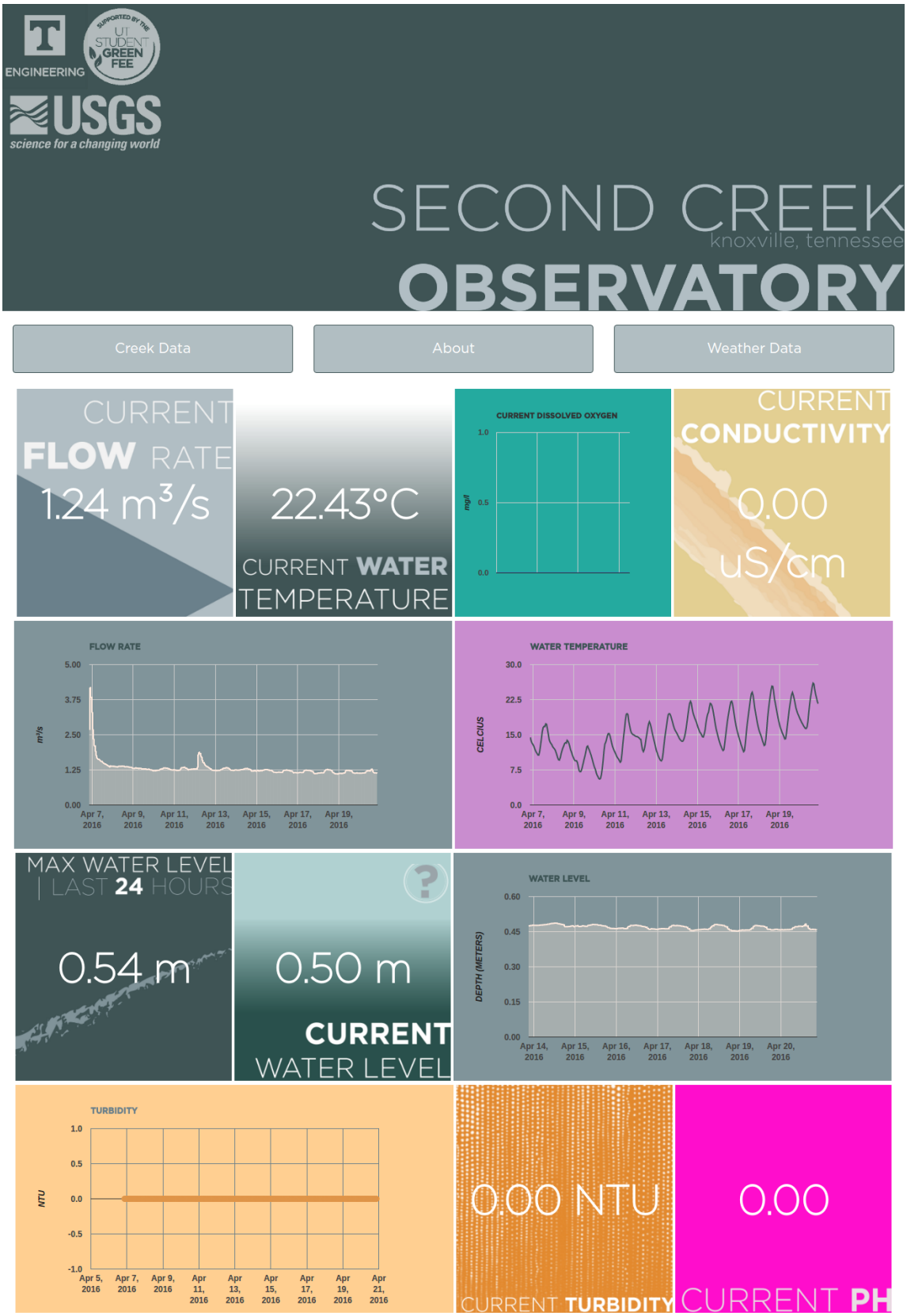

Figure 3: Creek Data Page

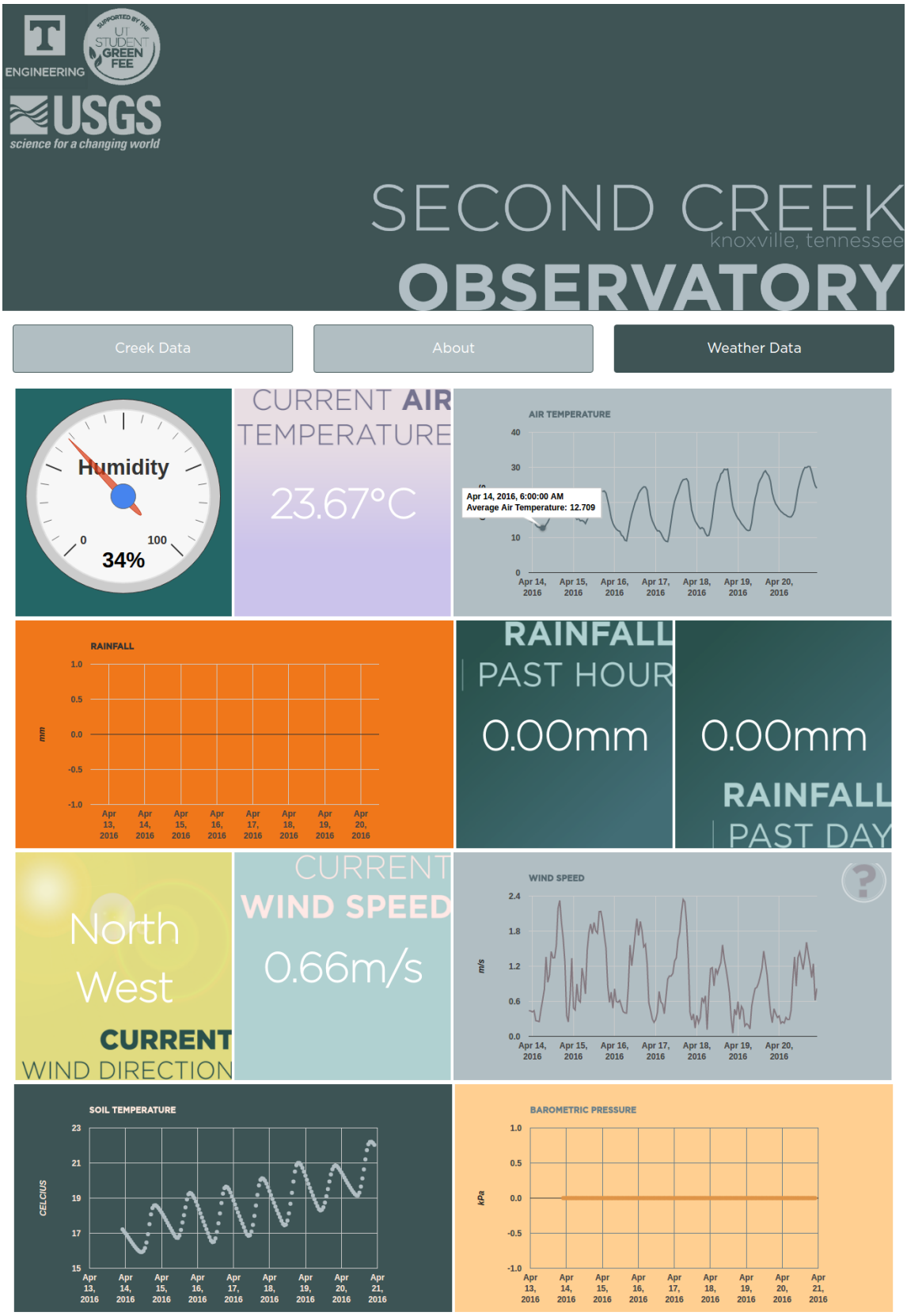

Figure 4: Weather Station Data Page

4.5 Completion of Requirements

In order to make sure that all agreed upon requirements were met, whenever discussing any deviations from our initial plan we consulted the requirements document and the customer. Since bi-weekly meetings with the customer were held, any time the group was thinking about changing or re-designing something, the customer was able to give his feedback early on in the process. In regards to the back-end, the work progressed in stages to ensure components worked together. The first task was to configure Flowlink and handle the pushing of data from the flow meter to the server's database. This functionality is a built-in feature to Flowlink Pro, but the team had challenges getting this method to work due to bugs in Flowlink and to network difficulties. With this component complete, the team turned to setting up Django to work with the database from the sensors in Second Creek. First, a plain webpage showing a single data point was developed and deployed on Django's development server. Then, this simple webpage was configured in Internet Information Services. Once this example was developed, hosting the actual website was just a matter of changing which files were served. Finally, once the group was able to make contact with Dr. Wright, the data from the weather station was configured to be added to the database through LoggerNet and LNDB.

After the requirements were created, the front-end was planned and designed with these requirements, implementation feasibility, and potential for issues in mind. During the implementation phase, features were chosen to be implemented based on the requirements and the design plan. Periodically, the website was checked on a variety of browsers to ensure compatibility. The website was primarily developed on the Google Chrome browser, so there were a few compatibility issues when testing on different browsers. For example, Internet Explorer does not support Promise objects, so a work-around was implemented using jQuery. We also had to add some Firefox specific CSS because text was overlapping in one of the small tiles. Once the website was nearly completed, we tested its features and browser compatibility thoroughly to ensure it met the requirements and expectations of the previous phases.

4.6 Results and Outcomes

This team has successfully created a functional website which meets the initial requirements of the project. The data displayed represents the most up-to-date requests of our customer and new data is automatically queried every five minutes. Informative and entertaining information about the project and the meaning of the different data types is available to the user. The data is displayed in a clear and visually appealing format, and the website fluidly re-sizes and re-organizes to fill the user's entire browser window. The data has been processed by the back-end so that it loads quickly, and no unnecessary data is sent. Loading animations are also employed to make any wait time more enjoyable. Logos of the projects sponsors are also included in the page header. Our customer has repeatedly expressed his satisfaction with our results.

5 Lessons Learned

5.1 Overcoming Challenges

During the course of this project, many unexpected challenges arose which provided many opportunities for learning. As mentioned above, the communication issue the sensor in Second Creek and the software was not a problem that we were initially expecting to have to overcome since this error was not caused by our group. This situation is representative of what working in the real world will be like though, because a problem will often occur due to an issue in someone else's product and a team will have to figure out how to work around the issue.

One issue we ran into with the front-end was with transitioning from the different data views. When making this transition, we need to hide the current charts, show the loading screen, get the data, draw the new charts, and then display these new charts. The hiding and showing of the charts takes time, so if the data arrived and the graphs were populated too quickly, the charts would not display correctly. To solve this issue we made use of JavaScript's Promise object, but, as was stated earlier, we had to replace this with jQuery's Deferred object because of compatibility issues with Internet Explorer.

5.2 Communication

Another lesson learned throughout this process was the importance of early and often communication with our customer. We had an initial meeting with our customer before winter break and then once we met again in January, we set up a bi-weekly meeting on Thursdays at 9 AM. Having at least an hour every two weeks to interact with our customer, update him on our progress, and gather any feedback from him was invaluable in making sure that we were staying on top of our project and satisfying the requirements that we agreed upon.

6 Statements of Team Members' Contributions

6.1 Ben Parrott

Responsibilities:

- Back-end Team Member
- Writer
- Presenter

Contributions:

- Set up Microsoft SQL Server, wrote SQL scripts for processing data, and handle SQL Server job scheduling.
- Assisted with the configuration of Flowlink and LoggerNet.

6.2 Aaron Young

Responsibilities:

- Back-end Team Member
- Writer
- Presenter

Contributions:

- Worked on the database.
- Assisted with the configuration of Flowlink and LoggerNet.
- Set-up Django and IIS.

6.3 Saajid Haque

Responsibilities:

- Front-end Team Member
- Writer
- Presenter

Contributions:

- Used Google Charts to create and edit how data is represented on website.

6.4 Haley Whitaker

Responsibilities:

- Team Leader
- Front-end Team Member
- Writer
- Presenter

Contributions:

- Used Google Charts to create and edit how data is represented on website.
- Wrote and turned in bi-weekly progress report documents.

6.5 Ryan Wagner

Responsibilities:

- Front-end Team Lead
- Writer
- Presenter

Contributions:

- Created initial website.
- Worked on optimization and design of website.

6.6 Richard Fagan

Responsibilities:

- Front-end Team Member
- Writer
- Presenter

Contributions:

- Used Google Charts to create and edit how data is represented on website.
- Wrote the pop-eup explanations for the website that describe what the data values actually mean.

References

[1] Teledyne ISCO. "Flowlink Pro Server Installation Guide." (n.d.): n. pag. Web.

[2] "Quickstart." *Quickstart—–Django MSSQL Dev Documentation.* Django MSSQL. Web. 21 Apr. 2016.

[3] Valois, Hugues. "Django Hello World Web Application on a Windows Server VM." *Python Web App with Django.* Microsoft, 04 Aug. 2015. Web. 21 Apr. 2016.

7 Signature of all Team Members and Customer

Haley Whitaker
Team Leader

April 25, 2016
Date

Aaron Young
Back-End Team Member

April 25, 2016
Date

Ben Parrott
Back-End Team Member

April 25, 2016
Date

Ryan Wagner
Front-End Team Leader

April 25, 2016
Date

Richard Fagan
Front-End Team Member

April 25, 2016
Date

Brayo Haque
Front-End Team Member

April 25, 2016
Date

Customer

4/26/16
Date