Hydro Citizens

Technological Feasibility

November 7, 2017

Sponsor:
Dr. Benjamin L. Ruddell

Team Mentor:
Dr. Eck Doerry

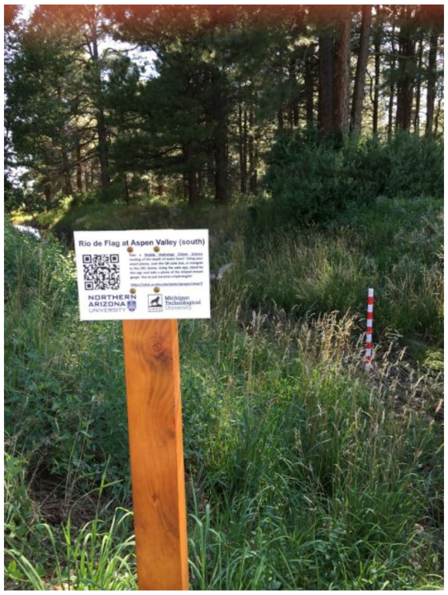
Team Members:
Luis Arroyo
Logan Brewer
Ryan Ladwig
Kelli Ruddy

1. Introduction

Our waterways and wetlands have an important role in our everyday lives and the more knowledge we have about them, the better. Hydrological data is important when it comes to flood prediction and prevention, water quality, and public education and knowledge. If we have more data, we can more fully understand how rainfall fluctuation affects a specific area. Flooding is one of the most destructive aspects of increased amounts of rainfall and can result in loss of homes and even death. Although we can never fully predict how severe a flood will be, we can better understand how increased rainfall will affect that area if we have the data. Right now the United States Geological Survey (USGS) collects most of the hydrological data in the U.S. and the problem with this is that they only collect data from large waterways and reservoirs. This is because the gauges they set up to collect this data are extremely expensive so they usually do not collect data on smaller waterways or ephemeral rivers, which are rivers without flowing water all the time. We need to understand how rainfall fluctuation affects certain areas not gauged by USGS. Without this knowledge, certain areas could be negatively affected by increased amounts of rainfall. Having more data is also important because then we can more fully educate the public on hydrological information and its importance. Having the public understand why we need to continue collecting this type of data is important, especially when it comes to our solution.

Our solution to this lack of data is to develop a mobile application that allows users to take pictures of a gauge which unlike USGS's expensive equipment will be a wooden post in the ground and a PVC pipe with red and white stripes as seen in Figure 1.1. The wooden post will have a QR code attached that can be used to uniquely identify the site and to provide more information to passersby.

Once the user has taken a photo of the wooden stake with the PVC pipe in the background, our image processing algorithm will add a horizontal line to where it believes the water height is in the photo. The user may adjust this line to get a more accurate water level.


Rio de Flag at Aspen Valley (south)
NORTHERN ARIZONA UNIVERSITY
Michigan Technological University
Figure 1.1 - Our gauging solution

We will use the user’s phone’s geolocation in order to plot exactly where the photo has been taken and this location will be stored with the photo upon submission. From there we will store the image, location, and water height data in our local database. We will then take the water height data and send this to our local HydroServer and then it will be joined with the main national HydroServer which is where a lot of national hydrological data is stored. Once the user has uploaded their information, we will give them feedback on their submission, which may include previous data points collected at that location or a graph showing how their contribution helps this data crisis. One key feature that separates our application from anything else will be the offline capabilities. Some of these gauges and waterways will be in areas without cell service or internet and this is where our application comes in. The user will be able to take a photo and store the information that goes along with the photo on our application and will be ready for submission when they are back in an area with coverage. The user should still be able to receive some confirmation that would have been stored on the application just for this.

As we proceed with the requirements development, we will now begin exploring the implementation options to ensure a feasible solution. This document will outline all of the major technological challenges that we will be facing while developing our project. In the Technology Analysis section, we will analyze methods for resolving these challenges, decide on a solution to each of our challenges, and explain our rationale for choosing our solutions. Finally, in our Technology Integration section, we will describe how all of our technologies work together and how they will resolve our technological challenges and needs.

2. Technological Challenges

The purpose of this section is to list and describe the technological challenges that we expect to encounter during this project. After analyzing the requirements of our project, we decided that we would need a platform that we could use to develop our mobile application, access and management of a database in order to store pictures and hydrological data, a manner in which to run image processing on images taken by the user, and we would need a way to present data to the user in a way that was easy to understand. From these needs we developed our expected technological challenges:

- Application Development Framework - We will need a platform that allows us to create this application.
- Database Management and Design - We will need a way to store large quantities of data and be able to access this data quickly.
- Computer Vision Framework - We will need to be able to use an image processing tool that connects with our application and allows us to plot a line indicating the water level.
- Data Visualization - We will need to be able to use a data visualization tool in order to present our collected data as well as other data in a more visually pleasing manner than simple text data.

In the following pages of this document we will be analyzing and describing these different challenges and finding the best possible solutions to each of them.

3. Technical Analysis

The purpose of this section is to fully describe and analyze each technological challenge for our project. We will then present alternatives and determine a solution that will work best in the context of our project.

3.1 Application Development Framework

The application framework is a software library that provides a structure to support the development of applications for a specific environment such as Android and iOS. Some features that app frameworks allows us to design GUIs for mobile applications and web application. Finding the best mobile app development framework to suit our needs is pivotal to our success when developing this project. In considering suitable app framework for this project, we will need the following features:

- Stable connection to the database - Allow access the database add a picture and hydrological data.
- Efficient image processing - Allow the user to take a picture of the PVC pipe and adjust the water level in the image.
- Implement data visualization - Allow the user should see a graph of the hydrological data that has been collected from a particular body of water.

The application should allow the user take a picture of a water gage and adjust a line of the water level. The user will be given the option, and encouraged, to write about their observations at each site such as the weather condition or extreme water levels. Once they have completed their submission, the application will send the image and the information they wrote to a local database. The picture and any special observations will stay in the local database, but the water level data will be transferred to the Hydroserver. If the user does not have internet connection, the information the user collected should be stored on the device until they gain access to a stable internet connection. The user is also able to see where they are located on the map and if there are any gages close by.

3.1.1 Alternatives:

After researching on Google the several app frameworks and talking to other classmates that had experience with mobile apps, the possible mobile application development framework options that will help us integrate a database, image processing, and data visualization features are Android Studio, Meteor, Visual Studio. When analyzing each alternative, we looked at which applications could help us implement the features easily, which ones would allow for the potential of cross platform development, and what languages each alternative uses. We will compare each of the following alternatives and determine which one we will use for our project based on their capabilities and known compatibility.

3.1.1.1 Android Studio

Android Studio is an IDE platform that provides tools for building apps on any type of Android device. The mobile applications are developed in Java and XML programming languages. Android Studio also allows the inclusion of APIs which allows us to implement the features needed for this capstone. With these APIs, we are able to connect to a database, integrate image processing using OpenCV, and data visualization. However, the problem with these APIs are that they only target a certain Android version, which can lead to conflicts using the APIs. Another problem is that if we want to work with iOS, then this would require rewriting the code using the Swift programming language. The last issue is that Android Studio mostly supports the local databases. If multiple users want to access the same database, it can lead to conflicts such as the database being corrupted.

3.1.1.2 Meteor

Meteor is a JavaScript web framework that is used to work with web and cross-platform mobile applications. Meteor integrates APIs from Angular, React Native, MongoDB, npm, and Cordova. With the APIs that it provides, we can manage to integrate most of features we are required to implement. With these APIs, we can connect to a database using MongoDB, integrate image processing using OpenCV, and data visualization using D3.js. However, the disadvantages of this framework is that if the local database continues to expand, the application will start to become slow when writing and accessing a table. The phone itself will also have a high CPU usage, high memory usage, and a slow performance.

3.1.1.3 Visual Studio

Visual Studio is a cross-platform IDE that is used to develop web, mobile, and desktop applications. The mobile applications are developed in C# and F# languages. Visual studio integrates APIs from Xamarin and React Native. With these APIs, we can connect to the database using SQLite and SQL Server database management systems. We can also integrate image processing and data visualization from Xamarin. However, the disadvantages of using Visual Studio is that we are very limited in how much coding we are allowed outside of this framework. When developing a new application, it would take 2GB of space without implementing anything on the app. If we use the API's, they would take more space into the application. Visual Studio is considered a cross-platform, however some of the code would be required to be rewritten from the front-end and back-end of the application, leading to time consuming.

3.1.2 Chosen Approach:

When choosing an application framework, we tried to keep in mind the features that need to be implemented for this capstone project. We have outlined the key features in Figure 3.1 seen below.

	Android Studio	**Meteor**	**Visual Studio**
Image Processing	4	4	2
Database	4	4	3
Data Visualization	5	4	3
Cross-Platform	1	4	4
Total	**14**	**16**	**12**

Figure 3.1 - Based on a 1-5 scale, 5 being a built-in feature and 1 not being feasible with the framework.

The table demonstrates each application framework (top of table) and the features that would fulfill our requirements (left-most column). For each requirement, we use a scale from 1 to 5, 5 being a built-in function and 1 not being possible to implement in the framework. The ratings were assigned based on the documentation for each alternative, as well as information found on blogs and example projects. The best framework option is Meteor because it will allow us to implement most of the features required for this capstone project in an easy and efficient way by using existing packages. Meteor score a 4 in image processing, database integration, data visualization, and cross-platform compatibility, because there are existing packages that are compatible with the framework.

Based on the research and looking at the challenges that we will be facing with each app framework, the best option is to use Meteor. Meteor is the best option because it will allow us to implement most of the features required for this capstone project in an easy and efficient way.

3.1.3 Proving Feasibility:

Meteor offers tutorials on YouTube on how to implement some of the features that would be necessary for this capstone project. If we want to know more about a method and how it is used, Meteor also offers an API Documentation Guide that states about how to use a package and what functions it includes. We can use APIs that the platform offers in order to implement the requirements. For our demonstration we will be installing the Meteor Development Framework and implement basic features that are required for this capstone. We plan on improving the application next semester by implementing a better user interface app and improve the existing required features.

3.2 Database Management and Design

This is important for our application because we need to be able to store a large amount of images and data points that are collected by our user. We may also need to store information from the National Weather Service to provide to the users as feedback. Our database needs to be scalable to account for user submissions of information and photos. We also need connection reliability in order to be able to access a user's submitted data whenever requested. Our database needs to access this submitted data in a timely and efficient manner.

3.2.1 Alternatives

Options to solve our database problem are MySQL, MongoDB and Apache Cassandra. We chose these three options after looking into many other database options that did not fit specific criteria.

When considering a suitable database tool for this project, we will need it to be:

- Scalable
- Reliable in terms of the connection
- Fast when accessing data
- Accessible for many devices

These options were the closest to what was needed for this project and in this document we will more fully investigate which is the best option and why.

3.2.1.1 MySQL

MySQL is an open source relational database management system so data is not just stored in one big table. It is also advertised as fast, reliable and scalable, which tackle each of our main concerns with a database. MySQL has a PHPMyAdmin built in support API which would be helpful for our project, it works well with geographical information systems and has great data security. It is also a good database if creating a web application. Some cons of using MySQL are you need multiple add one, it is not good with storing qualitative data such as images which is a key part of our project and frequent updates are required for large amounts of data. One of the biggest drawbacks to MySQL for this specific project is the storing of images. MySQL does not handle the storing of images very well.

3.2.1.2 MongoDB

MongoDB is an open source and NoSQL database. MongoDB uses collections and documents rather than rows and tables. It is a fairly new solution to the database problem emerging in the early to mid 2000's. It utilizes something called BSON which is a binary representation of JSON documents. MongoDB offers dynamic schemas, is accessible from many devices, works very well with large amounts of data and has in-memory speed. To store images, MongoDB utilizes GridFS which will allow for the storage of user submitted images. Since this is a newer product, there is less information and support. It also scales horizontally, however our team had initially discussed a need for vertical scaling.

3.2.1.3 Apache Cassandra

Apache Cassandra is an open source NoSQL database and functions best with unstructured data. Similar to MongoDB, it has dynamic schemas and can handle large amounts of data.Given the large amounts of data, Apache Cassandra is very reliable. It also offers something called CQL which is similar to SQL. It can have some unpredictable performance.

3.2.2 Chosen Approach

When choosing a database we tried to keep in mind our most important features for the application. We have outlines these key features in Figure 3.2 seen below. Scalability referred to if the database would be able to scale easily given large amounts of data which is something our application could face due to large amounts of possible user submissions. All three of our options scored a 5 for this category on the 1-5 scaling, where 5 is the best and 1 is the worst. MySQL scored a 3 for connection reliability while the other two received a 5 after we did some research. MongoDB and Apache Cassandra were good at handling large amounts of data and for this reason they were also efficient when it came to data lookups. All three options were accessible from most devices.

	MySQL	**MongoDB**	**Apache Cassandra**
Scalability	5	5	5
Connection Reliability	3	5	5
Data Lookup Efficiency	3	4	4
Device Accessibility	5	5	5
Handles large data amount	3	5	5
Total	**19**	**24**	**24**

Figure 3.2 - Based on a 1-5 scale, 5 being the best and 1 being the worst.

After some discussion, we realized we decided a NoSQL option would be the best because of how NoSQL supports more dynamic data. Our final decision was between MongoDB and Apache Cassandra. Based off of the research done and looking at the challenges we will face, we have chosen to work with MongoDB. Both MongoDB and Apache Cassandra were good for storing large amounts of data. Although MongoDB and Apache Cassandra scored the same in Figure 3.2, MongoDB is great for real-time analytics and image storing which are important factors in our project. MongoDB also offers GridFS which is a big plus when it comes to the storing of images. We feel as though we have fully explored each of the three options and have come to this decision fully informed.

3.2.3 Proving Feasibility
Based on the pros and cons of each of the researched database frameworks, MongoDB is the best solution. Our main concerns with a database are that it can store images, that it is scalable, and that it has connection reliability. We will further develop this by testing many images and data points to analyze how it works under these conditions. We will also test by continuing to add and delete data.

3.3 Computer Vision Framework
A computer vision framework (CV framework) is a system that allows for image processing, image analysis, and object tracking. The framework takes an image as input and analyzes the image, returning the desired output as defined by a set of algorithms. CV frameworks are typically used for facial recognition, but we will be using a CV framework in order to determine the water level of a body of water from a picture of a pole that is secured into the bed of that body of water.

3.3.1 Alternatives
Possible alternatives for our Computer Vision Framework include Tracking JS, which is a relatively new system that runs with JavaScript and is primarily used for facial tracking and recognition, OpenCV, which is a complete computer vision framework that is written in C/C++ and has wrappers for Python and Java, and finally JS Feat which is another CV framework written in Javascript.
The computer vision framework that we will use will need to:

- Distinguish red stripes from white stripes on a PVC pole
- Determine the height of the water that is covering the pole
- Use minimal resources on a user's device - RAM, CPU, local storage
- Run natively on Android and iOS
- Run the image processing algorithms offline

Tracking JS, OpenCV, and JSFeat were the three most common CV frameworks that we discovered when researching potential CV frameworks for our project.

3.3.1.1 Tracking JS
Tracking JS is a Javascript package for computer vision that would allow us to run our algorithm(s) in Javascript, which would aid in the cross-platform compatibility of our project. The focus of Tracking.js is in facial recognition, something that we would not require. There are a few tutorials on how to use Tracking JS, but we would need to spend a reasonable amount of time determining how we can learn Tracking JS from existing projects and examples. Tutorials for Tracking JS include color recognition and feature recognition.

3.3.1.2 OpenCV

OpenCV is a very popular computer vision framework with many examples and projects, that can be used as a reference, readily available on the internet. This would allow us to jump into programming with OpenCV without already having extensive knowledge about Computer Vision. There are a few projects, that use OpenCV, on the web that bear strong similarities to the computer vision framework requirements that we have to fulfill (that we will need for our project).

3.3.1.3 JSFeat

JSFeat was recommended by several resources that dealt with image processing and computer vision. We were unable to find an extensive collection of examples or tutorials for JSFeat, meaning that we would likely need to have a deep understanding of computer vision if we were to use this framework. Because there are not very many tutorials for using JSFeat, we would need to extensively research all facets of computer vision in order to understand the capabilities of JSFeat.

3.3.2 Chosen Approach

For this project, we will be moving forward with using OpenCV as our computer vision framework.

	Tracking JS	OpenCV	Feat JS
Offline	✔	✔	✔
JS	✔	✔ (With reduced functionality)	✔
Python		✔	
Android	✔	✔	✔
iOS	✔	✔	✔
Full Computer Vision Support (1-5)	4	5	5
Support / Tutorials	3	5	1
Totals	**7**	**10**	**6**

Table 3.3 - Check marks indicate that a feature is present, and numbered cells are scaled from 1 to 5, with 5 being the most desirable.

All of the aforementioned frameworks would be suitable for our project. In Table 3.3, you can see that each computer vision framework (top of table) would fulfill our requirements (left-most column) of a CV framework. OpenCV does have the added flexibility of allowing us to use a Python wrapper for the existing C/C++ source files, but we don't know yet if this will be helpful or not. Also in Table 3.3, you can see that both OpenCV and FeatJS have full CV Framework support, however FeatJS only scores a 1 out of 5 for "Support / Tutorials" due to a lack of existing example projects and tutorials currently available.

In summary, the flexibility of OpenCV (JS, Java, C++/C, Python), along with its widespread-use and available examples/tutorials, is what drove us to choose it as our computer vision framework. Once we gained access to the client's existing project, we realized that it also uses OpenCV as its Computer Vision Framework, so we feel comfortable with our choice.

3.3.3 Proving Feasibility

Stanford university offers a course on Computer Vision Frameworks / Image Processing and has posted all of their tutorials and projects online. The assignments in the course use OpenCV and one of the chapters deals with using OpenCV with Android. The projects that would be of most help to us deal with shape and color recognition, but we can also use these tutorials when we try to run OpenCV on Android, as the course website includes instructions on how to eliminate the need to have the device connect to a server. For our demonstration, we will be installing OpenCV on an Android device and implementing the existing algorithms on that device. We plan on improving the algorithm in the coming semester, but we will be using the existing algorithm for our demonstrations this semester.

3.4 Data Visualization

Data Visualization is a way to display data through graphs and charts. This is important because we need to present the data we are collecting in a way that is visually pleasing and easier to digest than simple text data. When considering suitable data visualization tools for this project, we will need them to be:

- Highly customizable
- Able to handle massive quantities of data
- Useable by us

3.4.1 Alternatives

We are going to be taking hydrologic data and displaying it in different ways that are easier for the user to see and understand, such as interactive scatter plots. Because of this reason, data visualization is very necessary for our final product. This is going to be used to display the information in a way that isn't simple text. We need to take all of the data we will be using and collecting and display it in a more visually pleasing manner. After searching through many websites and forums, we have found some of the best possible alternatives to be:

- D3.js
- Google Charts
- Chart.js

These three visualization options were the most commonly used when we researched these tools. In the following sections will look deeper into all three tools and explain which we found to be the best for project.

3.4.1.1 D3.js

D3.js is a JavaScript library that allows the creator to make dynamic, interactive data visualizations in web browsers. It is commonly used when you want to display data in a way that allows the user to interact with it. One specific way that we may be able to use this is by displaying the depth data that is collected by the user and allow them to see how it compares to the national water model in an interactive bar graph.

3.4.1.2 Google Charts

Google charts is an API that lets you make charts from data and display this in a web page. This is most commonly used when you have a lot of data collected from online sources that may change often. This also brings with it a package called GeoChart that allows the usage of maps. This would allow us to plot the locations of the hydrology stations easily. However, Google Charts requires a direct connection to the internet in order to load the visualizations which is a serious downside.

3.4.1.3 Chart.js

Chart.js is the most simple visualization tool that is used when trying to plot and compare data in a very simple manner. It has eight very easy to use and implement charts. However, this simplicity comes at the cost of customizability. Also Chart.js creates canvas images which are static and non-interactable. The simplicity and ease of implementation aren't worth the loss in customizability for this project since we may be needing to display large amounts of data in very different ways.

	D3.js	**Google Charts**	**Chart.js**
Charts rendered in	SVG	HTML5 using SVG and VML	canvas
Input data format	JSON and XML	JavaScript API	JavaScript API
Browsers supported	All modern web and mobile browsers	All modern web and mobile browsers	All modern web and mobile browsers
Chart and map types	No pre built charts	13 2D charts, maps available	6 chart types
Licensing	BSD-3	Free for all usage	Free under MIT license

Figure 3.4

	D3.js	**Google Charts**	**Chart.js**
Customizability	5	3	1
Scalability	5	3	1
Ease of use	1	4	5
Total	11	10	7

Figure 3.5 - Based on a 1-5 scale, 5 being the best and 1 being the worst.

When looking at table 3.5 above, you can see our rankings for each visualization tool. D3 is by far the best choice for customizability. Google Charts and Chart.js both have limited amounts of prebuilt charts that you can choose from, while D3 allows you to create hundreds of different kinds of charts with as much customization as you need. When looking at the scalability, D3 again wins. It is able to handle very large amounts of data with fast creation times. Google Charts was close in this category but the requirement to be online means we would only be able to create charts with a network connection which is very limiting. Charts.js comes in last because it is unable to handle very large amounts of data without resulting in slow load times. The final category was ease of use. Charts.js was the best choice for this category, with Google Charts being a close second.The reason for this is the simplicity of Charts.js, but Google Charts impressive documentation is what brings it close. D3 gets the lowest possible score due to its lack of prebuilt charts and complex code.

<u>3.4.2 Chosen Approach</u>

Our final choice weighed customizability and scalability much higher than ease of use. Due to this, D3.js was the clear choice. It has significantly more customizability than Google Charts and Chart.js. It doesn't come with any pre-built charts which attributes to the very low ease of use but this difficulty is overshadowed by all the interactive charts and graphs that we will be able to create with it. We will be able to take D3.js and implement it along with our collected water depth data and other data from the National Weather Service and the National Water Model in order to display this data to the users in a visual representation that is pleasing to the view and easier to digest in comparison to plain text data.

<u>3.4.3 Proving Feasibility</u>

For proving feasibility, we will be taking different chunks of data and creating a wide range of charts that will show our ability to use D3 anyway that we want to. We will also be directly testing the charting of example data that would be gathered from our app in order to show possible direct applications. Finally we will be implementing this in our app development tool and see how that interaction works within the app that we are creating for our demo.

4. Technology Integration

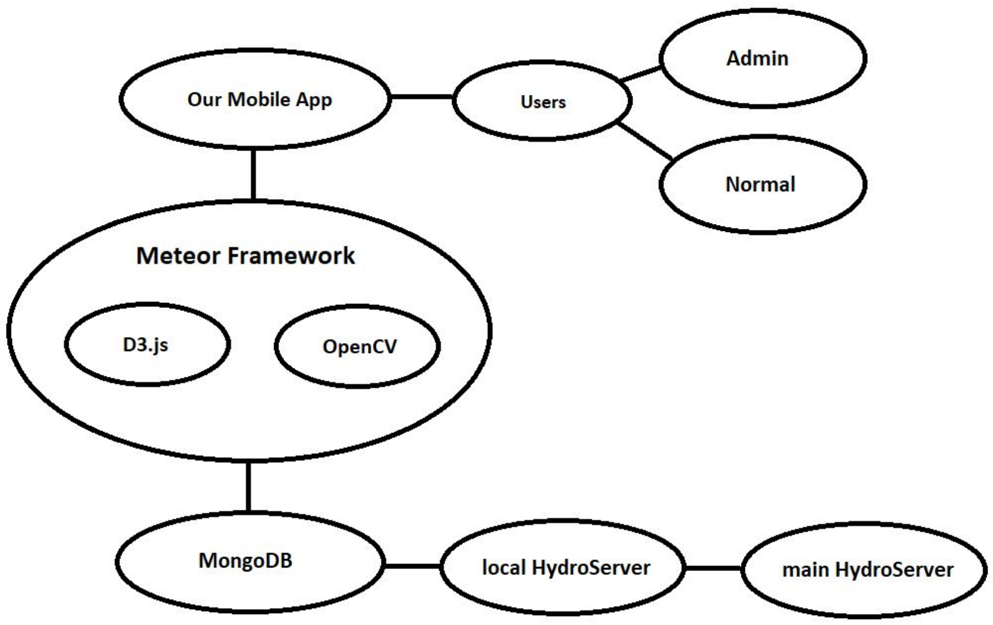


Figure 4.1 System Diagram

Now that we have seen all of our solutions, we will be doing our best to combine them as seen in the above figure 4.1. The Meteor development tool will be our base for our app. We will have users that are connected to our app and have different privileges based on if their role is admin or if they are a normal user. In connection with our app, we will be attaching a MongoDB database to the app in order to solve our storage issue. We will also be creating a local HydroServer database that will connect to our MongoDB and store the relevant hydrologic data. From our local HydroServer this data will be federated to the main HydroServer. We will be using D3.js in order to display the data we store in our database in a more meaningful manner than plain text. We will also be using OpenCV in connection to Meteor for our computer vision framework. Part of our testing for each individual part will be to see how they will come together and interact. Above you can see our system diagram, showing D3 and OpenCV running within our Meteor application, and MongoDB connecting to our Meteor application from the outside.

5. Conclusion

The collection of hydrological data is important when it comes to flood prevention, water quality and public education and knowledge. In order to have more accurate data, we must collect more. Through our application we hope to get the public more involved with this data collection and by doing so, they will understand this data's importance. This document has thoroughly discussed every technological issue we will face and what technologies will best help our team to produce a the best solution.

We plan on testing these more fully as noted in the proving feasibility section of each technological issue analysis. After much discussion and analysis, as a team we have decided these four technologies will best support our problems we will face with our project this year.