Real-time Geospatial Activity Tracker

Final Project Report

Kasra Eslami, Conor O’Neill, Dakota Douglass
CS4624, Multimedia, Hypertext, and Information Access
Professor Edward A. Fox
Client: Dr. Claire Gilbert

May 2, 2018
Blacksburg, Virginia 24061
# Table of Contents

1. Executive Summary ................................................................................................................................. 4  
2. Introduction .............................................................................................................................................. 5  
   2.1 Goals and Objectives .......................................................................................................................... 5  
   2.2 Statement of Scope .............................................................................................................................. 5  
   2.3 Software Context ................................................................................................................................. 6  
   2.4 Report Outline .................................................................................................................................. 6  
3. User Manual ............................................................................................................................................ 8  
   3.1 Website Description ............................................................................................................................ 8  
   3.2 User Roles and Responsibilities .......................................................................................................... 8  
   3.3 Website Navigation ............................................................................................................................ 8  
4. Developer Manual .................................................................................................................................... 18  
   4.1 Client-side Technologies ..................................................................................................................... 18  
   4.2 Server-side Technologies ...................................................................................................................... 18  
   4.3 Future Technologies (yet to be Implemented) ................................................................................... 19  
5. Progress and Reflection ............................................................................................................................ 20  
   5.1 Initial Projected Timeline ....................................................................................................................... 20  
   5.2 Problems and Solutions ....................................................................................................................... 21  
   5.3 Final Timeline .................................................................................................................................... 22  
6. Future Work .............................................................................................................................................. 23  
   6.1 Putting it All Together .......................................................................................................................... 23  
   6.2 Beyond the Semester ............................................................................................................................ 23  
7. Acknowledgments ..................................................................................................................................... 25  
8. References ................................................................................................................................................ 26  
9. Appendix A: Technologies ........................................................................................................................ 27  
   A.1 Amazon Web Services .......................................................................................................................... 27  
   A.2 React.js .............................................................................................................................................. 27  
   A.3 React-Router ..................................................................................................................................... 27  
   A.4 React-Bootstrap .................................................................................................................................. 27  
   A.5 Datamaps .......................................................................................................................................... 28  
   A.6 Node.js ............................................................................................................................................. 28
A.7 Express.js .................................................................................................................. 28
A.8 Socket.IO .................................................................................................................. 29
Appendix B: Project Development & Progression ............................................................ 30
B.1 Design & Planning ..................................................................................................... 30
B.1.1 Project Workflow - Client/Server ........................................................................ 30

**Table of Figures**

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1.1</td>
<td>Maptivity World Live Map</td>
<td>9</td>
</tr>
<tr>
<td>3.1.2</td>
<td>Maptivity Playback Options</td>
<td>10</td>
</tr>
<tr>
<td>3.1.3</td>
<td>Maptivity Playback Simulation for Selected Data Set</td>
<td>11</td>
</tr>
<tr>
<td>3.1.4</td>
<td>Maptivity Zoom on Mouse Scroll</td>
<td>11</td>
</tr>
<tr>
<td>3.1.5</td>
<td>Maptivity Bottom Drawer</td>
<td>12</td>
</tr>
<tr>
<td>3.1.6</td>
<td>Maptivity Landing Page</td>
<td>13</td>
</tr>
<tr>
<td>3.1.7</td>
<td>Maptivity Login Modal</td>
<td>14</td>
</tr>
<tr>
<td>3.1.8</td>
<td>Maptivity About Us</td>
<td>15</td>
</tr>
<tr>
<td>3.1.9</td>
<td>Maptivity Services</td>
<td>16</td>
</tr>
<tr>
<td>3.1.10</td>
<td>Maptivity Contact</td>
<td>17</td>
</tr>
<tr>
<td>A.7.1</td>
<td>Server.js - Create Express Object</td>
<td>28</td>
</tr>
<tr>
<td>A.7.2</td>
<td>Server.js - Express app.get() Method</td>
<td>28</td>
</tr>
<tr>
<td>A.8.1</td>
<td>Server.js With Socket.io</td>
<td>29</td>
</tr>
<tr>
<td>B.1.1</td>
<td>Client/Server Flow Diagram</td>
<td>30</td>
</tr>
<tr>
<td>B.1.2</td>
<td>Maptivity GUI Design Plan</td>
<td>31</td>
</tr>
</tbody>
</table>
1. Executive Summary

In today’s business environment, companies should be empowered to explore their data through a simple, visual medium, rather than overwhelming and time-expensive tabular reports. Virginia Tech’s IT department provides services to supply international and domestic students, faculty, staff, parents, and alumni with IT and computer solutions. Maptivity is a tool that our group built for the IT department. Our focus was to provide them with a way to more effectively see connections and patterns among data collected from these services. This was accomplished by developing a software framework that displays a world map and shows pings for data events.

Our client, Claire Gilbert, executive director of IT experience and engagement at Virginia Tech, supplied us data of incoming and outgoing phone call services. We designed this project with the intention of receiving a real-time data feed; however, due to the permissions required and timeframe given, we instead worked with previously logged data as a proof of concept. The end result was a unique playback simulation feature. This feature provides customizable playback and analysis of any data set.

We currently have the entirety of the call center’s phone data for the years 2016 and 2017 stored on our server. Our server parses the phone data into a common object format first, then the client can run simulations on this data at up to 100,000x speed. After the data points ping on the map, the points are then retained on the map as translucent bubbles, providing a heatmap effect over time.
2. Introduction

We created an extensible distributed software framework that provides geospatial visualizations of data sets. Our client, Claire Gilbert, the executive director of IT experience and engagement at Virginia Tech, requires the ability to display call data trends during high traffic hours and as an aesthetically pleasing addition to the call center. Additionally, we were able to utilize this project as a baseline for building a simple data visualization and analytics platform that can be employed to influence intelligence and productivity decisions.

2.1 Goals and Objectives

The overall goal of this project was to provide a means of visualizing data geographically. To do this, we developed an extensible distributed software framework from scratch that utilizes the highly customizable Datamaps library in order to provide a means to visualize data on a geographic map. We designed the framework with our stretch goals in mind of extending the geographic data visualizations to a data analytics platform that can be utilized by small companies and businesses.

2.2 Statement of Scope

We set out with several required features which made up our metric for project completion. We also had a multitude of stretch goals and features for future product development outside the scope of the semester.

Requirements:
- Must have fully working website running on remote server (AWS instance or similar)
- Must have the ability to view pings of data events on a geographic map
- Must integrate server to client communication to push the data events to the web client from an external data feed

Stretch Goals:
- Playback visualizations of data in specific geographic regions
- Differentiate between various data events
- View activity logs over time specific to various data events
- View activity logs over time for the whole application or service
- Incorporate specific geographic data for other parts of the world besides the USA
2.3 Software Context

The goal of the Maptivity software is to provide the Division of Information Technology at Virginia Tech with a means of visualizing their call data geographically. Achieving the main goals for the project will provide our group with a solid foundation for implementing a fully fledged data analysis platform for the Division of IT.

Our client provided us with all their phone data for the years 2016 and 2017. This data set was at the center of our attention for the scope of this course. The IT department provides a number of services to supply international and domestic students, faculty, staff, parents, and alumni with IT and computer solutions. 4Help is the call center here at Virginia Tech which accepts calls and provides guidance for those seeking IT assistance. We built a playback feature to prove our concept to be useful for the IT department. The playback feature allows our client to run a simulation of incoming calls for an entire year’s worth of data at up to 100,000x speed. The reason for a playback feature was to allow our client to visualize calls over an extended period of time to better understand how their global reach is expanding. Visualizing the difference between two years of incoming call data in a fast-played simulation ignites a number of unique questions, concerns, and ideas. One of the main reasons this is more effective than tabular data is because the concept of time is still in tact while viewing the simulation whereas tabular data is presented all at once.

2.4 Report Outline

This report is focused on how users will utilize the framework, user roles/responsibilities, and documentation of the technologies used in the development of the project. After reading this report, readers should be informed of the motivation behind this project, the decisions made during its development, and how to utilize the application in its’ current state.

Section 3: User Manual
This section focuses on the website description and how clients and users can expect to interact with the service. It begins with a description of the framework, the features implemented, and how it can be utilized by clients. It also brings attention to the clients’ and users’ responsibilities and the roles they play in utilizing our service.

Section 4: Developer Manual
The Developer Manual explains the technology stack that was utilized during development.

Section 5: Progress and Reflection
This section covers our lessons learned throughout the development of the project as well as the future work we have planned for Maptivity outside the scope of the semester.
Section 6: Acknowledgements
This section acknowledges the people we interacted with throughout the development of the project during the semester, and how they contributed to the project.

Section 7: References
This section identifies the technologies and resources we utilized in the making of this project and its documentation.

Appendices
Lastly, the appendices document our project as we progressed with development throughout the semester. It includes documentation relating to brainstorming, workflow, and project specifics. Appendix A relates to both the current and future technologies chosen to implement the vision for this project. Appendix B refers to the requirements and design portion of the project. Furthermore, it highlights the development of the requirements and the decision-making processes we used to design the structure of the project.

The following section gives a brief description and overview of the website, some of the expected roles and responsibilities of our prospective clients, and how users can navigate on the Maptivity website.

3.1 Website Description

Maptivity can be accessed at https://maptivetech.com. This leads to a homepage featuring a login module and several different tabs, one of which allows access to the site’s main service for clients. The Map tab redirects signed in clients to the data visualization portal meant to display data in an interesting and useful visualization tailored to the client’s needs. The simplicity of the map will allow clients to continually be able to monitor activity without being overwhelmed or having to pause to absorb and comprehend data.

3.2 User Roles and Responsibilities

Maptivity users will generally be companies or small businesses seeking real-time data visualization for business intelligence and productivity. The following is a break-down of client responsibilities:

Data
Each client is responsible for supplying Maptivity with a data feed. The purpose of Maptivity is to be able to serve a wide domain of companies and their personal data needs. Every client visits our landing page, then must sign in before gaining access to their data. Once the client has signed in, they will be brought to a map page that allows them to visualize their data feed and run playback simulations with previous data.

Security
We understand that data is the lifeblood of nearly every one of our prospective clients. Different companies have different protocols when it comes to third parties accessing their data. Each company is responsible for providing us with these protocols along with any digital certifications necessary to continuously employ their data feeds.

3.3 Website Navigation

The Maptivity website navigation should be intuitive from the moment a user types in the URL, but we would like to specify particular use cases and scenarios in this subsection. We will
detail how a user may access each of the sections of our website and the purpose that section of the website serves.

**Map Visualization Access**
Click on the “Map” tab, after logging in. Users will be redirected to a map which would display real-time pings as shown in Figure 3.3.1.

*Figure 3.3.1: Maptivity World Live*

The live map page is meant to serve as a display for incoming data, meaning the user doesn’t have to do anything else to set up the map once they navigate to this page. Additionally, users can also hover their mouse to the top of the page to play back a simulation of data from a specific time period. See Figure 3.3.2.
Users can choose a data set from a specific year, and the map will begin a simulation from the given year with a time scale option, simulation time information, and pause/play control. See Figure 3.3.3.

Note: For now, we have retrieved phone call data from 2015 and 2016 from our client. This playback feature has brought us a step closer to a data analytics platform and has served as a viable proof of concept for real-time data visualization.
While on the map page the user can use the scroll wheel on their mouse to zoom in on specific geographic area. The result of this feature can be seen in Figure 3.3.4.
Users can hover their mouse to the bottom of the page to show another utility drawer that slides up from the bottom of the page. The drawer at the bottom of the page is intended to provide a space for the development team to design and put new features relating to the map.
Users seeking out specific information about Maptivity will find all necessary information on our scrollable landing page.
Users can sign in by clicking the “Sign In” button. A modal will popup where the user can provide a company name, a specific username, and a password to login. This feature is intended to allow companies to access their data but also give companies the ability to manage permissions for their data amongst the employees of the company.
Users seeking more information about the goals of Maptivity and its creators would go to the “About” page. This page gives the user a brief understanding of Maptivity.
Users seeking more information about the services that Maptivity has to offer, can click on the “Services” tab on the homepage. This section has four subsections which highlight the current and future features of the application:

- **Real-Time Data Visualization** - Geographic visualization of data events in real-time to facilitate pattern recognition and data analysis.
- **Playback Data Visualization** - Re-visualization of geographic data events.
- **Application Event Tracking** - Analyze specific application events over time.
- **Comparative Event Tracking** - Find the distribution of feature usage in your application.
Users seeking contact information, in the case of issues or questions unanswered on the site itself can click on the “Contact” tab on the homepage.
4. Developer Manual

This section goes over the technology stack utilized during the development of the project along with future technologies we plan to add in the future. For further information on the technologies mentioned in this section please refer to Appendix A.

4.1 Client-side Technologies

- React.js
  - Javascript library for building user interfaces.
- React-Router
  - Declarative routing for React JS.
- React-Bootstrap
  - A re-implementation of Bootstrap components using React.
- Datamaps
  - Customizable SVG map visualizations for the web in a single JavaScript file using D3.js.

4.2 Server-side Technologies

- Amazon Web Services
  - Scalable and high performance services geared for modern web applications.
- Node.js
  - A Javascript runtime built on Chrome’s V8 Javascript engine.
- Express.js
  - A minimal and flexible Node.js web application framework that provides a robust set of features for web and mobile applications.
- Socket.io
  - Enables real-time bidirectional event-based communication.
- Google’s Geocoding API
  - A geocoding API utilized to translate international addresses into latitude and longitude values

At the current stage of development we have implemented a playback feature that allows the retrieval and geospatial visualization of client data. Once the tool becomes real-time, we will have a fully employed server-client application. Specifically, we will have an entry point for our client (IT) to send us data as it occurs. The server will parse the data and send it to the front-end in a formatted JSON as a single entry to ultimately display a ping. This server-client interaction will be achieved using Socket.io.
Upon receipt of the data, the data for each year was merged into a single comma-separated-value document and parsed using Node.js to extract the area code and timestamp of each entry. The parser has been built to handle different formats of data. An online database was utilized to translate a specific area code into a geolocation, but this was solely for the United States and Canada. Unfortunately, there was only a database that mapped international area codes to a specific city and country. The geolocations for over 12 thousand rows of international data were obtained using Google’s Geocoding API. The addresses from the database were fed into a geocoding function which would return a large response object with geo-information in the form of a JSON that was utilized by Datamaps to produce data pings.

A dictionary with area code and geolocations was created to serve as an internal database. This made the geolocation lookup via an area code simple from the phone data we received from our client. From the 2016 and 2017 csvs, each area code was extracted, a geolocation was found, and date object was created. Eventually, we had two large JSON files, one for each year, with each entry in the form of “Timestamp, Latitude, Longitude.” This is used on the front-end to display pings and ultimately run the timed simulations.

Maptivity is currently hosted on an EC2 instance by Amazon Web Services. This instance is linked to the maptivetech.com domain on GoDaddy, which means that Maptivity can be accessed by simply typing in https://maptivetech.com.

### 4.3 Future Technologies (yet to be Implemented)

- **MongoDB**
  - A database which stores data in flexible, JSON-like documents.
- **Redux**
  - A predictable state container for JavaScript applications.
5. Progress and Reflection

This section specifies the projected timeline from the beginning of the project, the problems we faced along the way, how they were addressed, and a final timeline laying out the progress in response to every problem encountered throughout the semester. Furthermore, we detail lessons learned through the course of the semester in this section. For further information, see Appendix B.

5.1 Initial Projected Timeline

We started this project by creating a timeline of important milestones to be completed. This timeline was the initial guidance for our project milestones but was modified slightly after facing trouble with completing our data access milestones.

January:

Milestone 1: Client to comport with our project

Milestone 2: Begin planning, decide on specific implementations, test out different map visualisation libraries

February:

Milestone 1 Feb 10th: Set up for development, set up git, all necessary libraries chosen

Milestone 2 Feb 22nd: Service should be able to retrieve and buffer user access/use information from client's database

March:

Milestone 1 March 22nd: User interface finished

Milestone 2 March 29th: Service should be able to retrieve and buffer user access/use information from client's database

April:

Milestone 1 April 5th: Begin considering use cases and thorough testing, ensure real-time occurrences, tweak bugs

Milestone 2 April 26th: All pieces tied together, completed prototype
5.2 Problems and Solutions

One of the first problems that we encountered during development was the learning curve associated with React JS and web development in general. Prior to this project, two out of three group members had very limited experience with web technology stacks. This made it difficult to get traction when beginning development and continued to affect project velocity throughout the semester. We started out by delegating tasks during meetings, then splitting off to work on the tasks individually. After the first month of limited progress on the User Interface, we adopted a paired programming approach to development so that we could work together on problems we ran into. This greatly improved our developer workflow and allowed us to utilize the knowledge base of all group members.

Another issue we encountered was obtaining access to the data feed from the Division of IT. Our initial client meeting with Dr. Gilbert seemed very promising as she referred us to Ian Heflin to get us in touch with the source of appropriate data. Once we were working with Mr. Heflin, it took longer than expected to determine the correct source of data and arrange the meetings. This was partially due to scheduling conflicts and clarifications on the data we were seeking. We eventually did meet with the person in charge of all data logs and learned about the process to gain access to the data. We pushed back our data access milestones in order to compensate for this. In the future, we will be sure to dedicate more time towards effectively communicating with the client.

The latter part of our project implementation also presented the group with some issues. In order to display pings in real-time, we needed an open server to client communication to push geolocation data to the client. Initially, we had to weigh the advantages and disadvantages of different technologies in the context of the application. After some research, we concluded that WebSockets was the most appropriate solution that used two-way client-server communication protocol. We set up a small client-server application with Socket.IO (a bidirectional event-based communication library) to validate that information could be sent by the server and received by the client simultaneously.

In order to integrate this socket.io server/client communication into the scope of our project there was a small learning curve associated with how to wire the server up with Socket.IO (utilizing Express.js). Once that was configured, simple messages could be sent from the server to the console of the Map page (the client). This served as an important prelude to sending geolocation events from the server to the client.

At this point we hit another hurdle involving our client. It took Claire, Ian and us a a lot of discourse spanned over a couple weeks before we decided on the most appropriate data set to employ. We were ready to implement functionality to retrieve and display data in real-time, but we mutually agreed that implementing a playback feature as proof of concept was a more appropriate step of action before accessing a live data feed. We gained access to two sets of phone call data from different years. As a result, the focus shifted from real-time pings to
implementing a playback feature that could sequentially display pings from the two separate data sets.

In order to translate a phone number into a geolocation, the area code must be extracted and looked up in a database or external API. This is easy when dealing solely with US and Canadian phone numbers. However, to be able to translate international phone calls into geolocations, the country code must be provided. This is because different areas within different countries can share the same area code. We were eager to highlight an important use case for our client: to illustrate that the call center is getting international calls more than before. However, our client was unable to clarify the exact format of the phone calls. We had to implement a naive approach to parsing and looking up international phone calls.

Unfortunately, we were unable to make the tool occur in real-time for the final cut of the course. However, the playback feature has pushed us farther into making the tool a data analytics platform in addition to a real-time data visualization. Making the tool real-time simply requires wiring the working websockets framework to work with a live data feed and to provide an API endpoint to retrieve data from our client.

Ultimately, we did not face too many unexpected or completely exhaustive challenges, and the challenges we did face were handled appropriately. There will always be a learning curve when working with new technologies. In addition, there will always be unexpected delays when working with clients, as scheduling meetings across various groups is always a challenge. The important takeaway is that our team was able to respond to the challenges we faced and modify the project timeline accordingly.

## 5.3 Final Timeline

After encountering some challenges in the first half of the semester, the project trajectory for the second half took a slight turn. Fortunately, we planned for fluctuation and were able to modify the timeline such that we were only behind from the initial timeline with respect to the data access.

**January:**

*Milestone 1:* Client to comport with our project

*Milestone 2:* Begin planning, decide on specific implementations, test out different map visualisation libraries

**February:**

*Milestone 1 Feb 10th:* Set up for development, set up git, all necessary libraries chosen

*Milestone 2 Feb 22nd:* Discuss with client what kind of data set would serve most appropriate
for demo, seal project details

**March:**

*Milestone 1 March 5th:* Go through necessary protocols to get access for call center data from Information Technology Experience and Engagement

*Milestone 1 March 29th:* Access to data granted and given, plan to make use of the data for a playback simulation to serve as proof of concept before going real-time with live data feed

**April:**

*Milestone 1 April 12th:* Data is cleaned up, parsed, beautified JSON to feed to data map

*Milestone 2 April 26th:* Finish playback feature, show client

6. Future Work

The baseline functionality of providing geospatial data visualization has been completed. Though we do not have an implemented real-time data visualization tool, we have met all of our goals and even more. This section will discuss the finishing touches necessary to have a full-rounded real-time data visualization tool, and some future work not aligned with the intentions for this capstone course.

6.1 Putting it All Together

The next step is to take all of the pieces and put them together into a fully functional application. This means implementing functionality to retrieve a live data feed, integrating our socket.io implementation to push this data from the server to the client, and implementing a live feature on the front-end.

6.2 Beyond the Semester

We currently do not have a database implementation. This means that the only data we can currently use is that which gets pushed to the client during the user’s visit. This is obviously not ideal for the eventual implementation of the data analytics aspect of Maptivity, so we will implement a simple MongoDB database to store each data event. This is the second step of work yet to be completed and will serve as the basis for the expansion of this project.
Outside of the scope of this class, the potential to expand this project is massive. The goal is to eventually extend Maptivity into a data analysis platform that can be customized for specific data sets. We have ideas for a broad variety of features that can assist clients in achieving organizational success. Some of those features are:

**Application Event Tracking**
Analyze specific application events over time with the ability to distinguish between usage in different geographic regions.

**Comparative Event Tracking**
Compare and contrast what features of your application are being utilized more or less than others in different geographic regions and the world.

**Data Event Notifications**
Receive notifications when specific data events occur or when certain usage thresholds are reached. Implement the ability to customize and configure this notification system for different geographic regions.

**Data Event Predictions**
The ability to predict future events based on past data through the use of machine learning.
7. Acknowledgments

This section outlines personal acknowledgements for individuals who provided assistance and feedback throughout the development of this project. This is a small token of appreciation for the guidance provided by our client and our professor throughout the semester. Without these individuals, we could not have successfully completed this project.

**Claire Gilbert**
gilbertc@vt.edu
540-231-7492
Dr. Gilbert is the Executive Director of IT Experience and Engagement and was our first Point of Contact for this project. She was the person who originally saw the intent behind our project and directed us to Mr. Heflin to clarify the path to gain access to the data we need. She is also the main point of contact due to her interest in utilizing our real-time activity tracker for displaying call center data.

**Ian Heflin**
iheflin@vt.edu
Mr. Heflin is an Operations Analyst for IT Experience and Engagement. He has been helpful in providing project direction and getting us in contact with subject matter experts who provided feedback to our project.

**Edward A. Fox**
fox@vt.edu
540-552-8667
Dr. Fox is the professor for CS 4624: Multimedia, Hypertext, and Information Access, and was a resource throughout the semester. He provided insight into project development and client interaction.
8. References


React. Retrieved March 26, 2018, from https://reactjs.org/


Socket.IO. Retrieved March 26, 2018, from https://socket.io/
9. Appendix A: Technologies

A.1 Amazon Web Services

Link: https://aws.amazon.com/
Details: In order to get our code up and running globally, we created a t2 micro instance on Amazon EC2. This allows our code to be accessible from anywhere at any time. We may have to expand our AWS structure when we start storing and managing more data, but for now it serves as a good foundation for hosting our website.

A.2 React.js

Link: https://reactjs.org/
Details: React.js is a popular front-end framework to speed up the development and access of websites. React is founded upon the concept of creating reusable components then managing their state with a unidirectional data flow. This is great for our application because the state of the data needs to be passed around to many pages, and React is able to manage this without the user ever having to refresh the page.

A.3 React-Router

Link: https://reactjs.org/
Details: React Router is a library used to manage our URL structure and route the user to the correct pages. This allows us to keep semantic URLs and link pages with custom React components.

A.4 React-Bootstrap

Link: https://react-bootstrap.github.io/
Details: React bootstrap is a library that ports Twitter’s bootstrap into the popular front-end framework React.js. Bootstrap is a library that makes styling our website easy and intuitive. It uses a grid system and css themes to give us a good foundation to build onto.
A.5 Datamaps

Link: http://datamaps.github.io/

Details: Datamaps is the Javascript mapping library we chose to use for our project. The datamaps allows us to dynamically change out maps from world maps to specific country maps. Each of the maps have json files which designate the geo locations into map coordinates, and this is helpful for pinging data events in real time.

A.6 Node.js

Link: https://nodejs.org/en/

Details: Node serves as a single threaded web server that we spin up to send our index file to the client and data content we have stored on the server filesystem. We chose to use Node.js because this keeps our entire tech stack in Javascript and Node can effectively handle async requests with libraries like Express.

A.7 Express.js

Link: https://expressjs.com/

Details: We use Express, a popular Node web framework, to provide us with a mechanism to route each HTTP request. Once we create an Express application, we store it in an object called app. See Figure A.7.1.

Figure A.7.1: Server.js - Create Express Object

```javascript
const app = express();
app.use(express.static(DIST_DIR));
const server = http.createServer(app);
```

We have an app.get() method which specifies a callback function that will be invoked whenever there is an HTTP GET request with a path ('*') relative to the site's root. This endpoint is always placed at the end of your server.js file to catch all requests with the wildcard. It simply serves up our html file and is only used to send the homepage file to the user.

Figure A.7.2: Server.js - Express app.get() Method

```javascript
app.get('*', function(req, res) {
    res.sendFile(path.join(DIST_DIR, "index.html"));
});
```

In the future, more endpoints will be added for retrieving specific sets of data, and to manage the authentication of users.
A.8 Socket.IO

**Link:** https://socket.io/

**Details:** Socket.IO is a websockets Javascript library for real-time bi-directional communication between a client/server application. This ubiquitous library was the most appropriate in the context of our application as it would allow us to send and receive geolocations from server to client simultaneously.

The most important method with Socket.IO was `on()`. The `on()` method takes two arguments: the name of the event, in this case “testEvent” and a callback which will be executed after every connection event. The `on()` method is a core Node.js method tied to the EventEmitter class.

*Figure A.8.1: Server.js With Socket.IO*

```javascript
io.on("connection", (socket) => {
    console.log("Connected to client"), setInterval(
        () => socket.emit('testEvent', JSON.stringify(data.phoneData)),
        5000
    );
    socket.on("disconnect", () => console.log("Client disconnected"));
});
```

The connection event returns a socket object which will be passed to the callback function. By using said socket we are able to send geolocation data to the client in real-time. The client will be listening for event ‘testEvent.’ Whenever ‘testEvent’ has been triggered, the function `socket.emit()` will be executed, sending the JSON of geolocations and corresponding timestamps over the duplex channel. At that point we feed the data into the map and display the ping! We are also listening for the disconnect event. It will be fired as soon as a client disconnects itself from the server.
Appendix B: Project Development & Progression

B.1 Design & Planning

This section documents our design and the planning phase of the project. The diagrams below highlight a few key brainstorming sessions which guided the direction of our project. We had many sessions for designing and planning before we began writing any of the code. We also held whiteboarding sessions throughout the semester to update structure designs and the website UI.

B.1.1 Project Workflow - Client/Server

*Figure B.1.1: Client/Server Flow Diagram*
B.1.2 GUI Design - Login to Datamap

*Figure B.1.2: Maptivity GUI Design Plan*