Michael Grantham

GeoCloud: A Cloud Based Mapping and Data Collection Application

Marine Academy of Technology and Environmental Science

Author's Notes

I'd like to thank my research advisors Dr. John Wnek and Mr. Jason Kelsey for giving me the ability to do this project. I'd like to thank all of the contributors to the MySQL documentation as it has been invaluable for debugging my application. Lastly, I'd like to thank my father Michael P. Grantham for giving me assistance and advice when I was struggling with certain aspects of the development of the application.

Abstract

Data collection is an important factor in conducting research, researchers will often collect data at different locations, and later input Global Positioning System (GPS) coordinates into computer software to create maps. In some cases, there are multiple people collecting large amounts of data simultaneously; managing all of this data is a cumbersome task. The purpose of developing this application was to create a cloud based platform for location based data collection. This application had to be lightweight and could not require any installation and had to be able to run on any device with internet access and a web browser. This application had to allow multiple users to connect on one map, and add points and data. Data had to be able to be added to existing points based on the nature of the research that was conducted. This novel method of mapping streamlined the process of collection, mapping, and sharing of data. Unlike other applications, this application leverages open-source software which is lightweight and free. Based on development costs, running this application is estimated to cost only less than six dollars per month for over 150 users. This novel application provides users a low budget solution for cloud based mapping and data collection.

Introduction

Maps are the fundamental base of the study of geography; they are a way of recording information and presenting it to others (Foote 2015). Maps are important as methods of displaying data for the fields of ecology and geology, to political research, to archeology, and economic research (Foote 2015). There are online mapping platforms available, one such platform for mapping and data collection is ArcGIS Online. This is an application that can be purchased through a subscription service which costs anywhere from $38.00 per user per month for 100 users to $42.00 per user per month for five users (ArcGIS 2016). This subscription includes an application called Collector which allows users to add features and data to a map from their mobile device (ArcGIS 2016). This is a very useful ability to have for users who are collecting data in the field; however, costing up to $500.00 per year per user could be out of many user's price ranges if all the user wants is to be able to do is to create a map and collect data from a mobile device. The purpose of GeoCloud is to create a budget solution for those who wish to collect data and create a map from their mobile device.

The purpose of developing this application as a cloud-based application is so it accessible on any device, at any time. Cloud applications have existed for quite a while, but only in the last several years has the term "cloud" become popular (Carter 2012). Storing information in the "cloud" simply means it is kept on a central server that may either by operated by the application developer, or a third party service. There are some security concerns when running a cloud based application. Data breaches often occur because of weak passwords and authentication (Rashid 2016). This application's database is protected by a strong password created by a password generator to ensure maximum security. The details of this password will not be disclosed. Another threat is a data breach occurring and user personal information being compromised, which can cost thousands of dollars (Rashid 2016). By not storing any passwords through a system called Hashing and Salting, this application will not expose any passwords if a data breach does occur. This application will allow users to access all of their data anytime, anywhere, and still have a strong focus on security.

This application will meet the following specifications. This application will allow the user to create maps, add points to the maps, access their maps by logging into their user account. The user will be able to add data to points and export this data as a spreadsheet. The user will be

able to export the map for use in other mapping applications. The format for the map export will be Keyhole Markup Language (KML). This format can be opened in Google Earth, Google Earth Pro, and ArcGIS (Google 2016). The user will be able to share the map with other users and assign permissions to the different users. This application should work on both desktop and mobile platforms. All data will be stored in a database so it can be accessed on any device.

Methodology

The main languages used to develop this application were MySQL, PHP, and JavaScript. Hypertext Markup Language (HTML) and Cascading Style Sheets (CSS) were also used in the development. This application also leverages the features of the JavaScript library jQuery and the jQuery plugin FileSaver.js. Keyhole Markup Language (KML) is also used for the exporting of maps. This application was built iteratively in a series of five stages between July and December of 2016.

STAGE ONE

The first stage of the development was creating a basic application that would capture the user's location, and create a KML file using the user's geolocation. This application featured three buttons: "Add Point", "Finish", and "Save" (Figure 1). The "Add Point" button used the HTML5 geolocation feature to capture the user's current location. The user was then prompted to name the point. The information was then encoded into KML. When the user selected the "Finish" button the KML for each point was compiled in JavaScript and relayed via jQuery's Asynchronous JavaScript And XML (AJAX) functionality. The jQuery AJAX functionality allows data to be sent from the user's browser, also known as the client, to the server, and data to be sent back from the server, to the client, without reloading the page or opening a new page (jQuery 2016). The application used this to relay the KML to the server, where it was placed into a temporary KML file. The Uniform Resource Locator address (URL) of the KML that was created was sent back to the client. The user was then able to select the "Save" button which, using the Google Drive Application Program Interface (API) the user would be able to save the KML file into their Google Drive account.

STAGE TWO

The second stage of the development incorporated the Google Maps JavaScript API version three. Points that the user added were displayed on the map with a red placemark. The user could add points in two ways; adding a point by getting the current geolocation of the user's device, and inputting the latitude and longitude coordinates manually. All added points were displayed in a green box adjacent to the map display (Figure 2). When the application loaded, the user would be prompted to create a new map and name it. If the user chose not to create a new map they would be prompted to input the identification number of the map that they wanted to continue working on. This feature was achieved by storing all the data in a MySQL database. The database structure included a maps table and a points table. The maps table stored the map name and an auto-incrementing map identification number. This identification number was provided to the user so they could access the map again. The points table stored the point name, the point latitude, the point longitude, the map identification number which the point was in, and an auto-incrementing point identification number which was not provided to the user. When a map was loaded its points were loaded and displayed. The map could be exported to a KML file. The KML file was saved to the user's device rather than the Google Drive belonging to the user.

STAGE THREE

The third stage of development introduced the first working alpha version of the application. This version brought a variety of enhancements, but still only featured the basic functions to make it a working application. The first feature added was the database structure that would allow for expansion and secure data storage (Figure 3).

User accounts were implemented. With user accounts, comes a security risk. If the database were ever to get hacked, it is important to make sure that the security of the users' passwords are not compromised. The way this is accomplished is through a process known as hashing and salting. Hashing is the process of taking some set of data, for example, a password, and turning it into a string of characters that cannot be reversed back to the original set of data. The hash of the data will be the same every time that data goes through the hashing algorithm. Hackers get around hashes by using a rainbow table, which is a list of passwords and their

known hashes. This is stopped by adding a random string of characters, known as a salt, to the user's password, and then hashing the combination of the password and the salt, and storing the this hash and the salt used. When the user logs in the password will be combined with the stored salt and run through the hashing algorithm. If this resulting stored hash, matches the one stored in the database, the user has entered the correct password (Defuse Security 2016).

The login page provides the user the option to stay logged in. This feature provides users who use the same device the convenience of bypassing the login every time they use the application. To implement this system had to be created to store the user's information on the device without storing the password in such a way where it could be compromised. This was accomplished by storing two cookies upon user login. If the user selected to stay logged in, the cookies are stored for 30 days. If the user opts not to stay logged in, the cookies delete after the session closes. One of these cookies stores the user identifying number, and the other stores the hash that is stored for the user. Every time the user performs an operation that interacts with the database, the user id and user hash are checked with the ones stored in the database and the action will only be performed if the credentials are valid. This prevents users from accessing maps that do not belong to them, and it blocks users without accounts from using the application. All interactions between the client and the server are encrypted via Secure Sockets Layer (SSL).

Data can be added to points. When the map is created, the user is prompted to add parameters that are being assessed. For example, if the user was collecting temperature at different points, the user could input temperature as a parameter. Parameters added to a map apply to all points on the map. The values of the parameters, and new parameters can be added to each point after the points are created. A parameter added to one point will then apply to all other points on that map. Parameter values do not have to be filled on all points, only the points that the value applies.

The application uses stored procedures to communicate with the database. A stored procedure is a set of MySQL statements that can be executed by passing in different information (MySQL 2016). The application uses the following stored procedures.

- "AccessMap" takes the identification column value of the map that is being accessed and returns a list of points, including the point identification numbers, point names, the latitudes of the points, and the longitudes of the points.

- “AccessMapParameters” takes the identification number of the map being accessed and returns the parameter names and parameter identification numbers of the parameters assigned to that map.
- “AccessPointData” takes the point identification number as an input and returns the parameter names and the data that corresponds to that parameter and point if it exists.
- “AccessUserMaps” takes the user identification number and returns a list of maps that the user has created.
- “AddParameter” takes in the desired parameter name and the map identification number and will add the parameter into the database if the parameter does not already exist for the map. The procedure will return the parameter identification number for a success, or an error code for a fail.
- “AddPoint” takes the map identification number, the desired point name, the point latitude, and the point longitude and it will add the point information to the database.
- “AddValue” takes the parameter identification number and the point identification number and the desired data to add to the point and it will insert into the database that data and relate it to the point and the parameter.
- “CreateMap” takes the desired map name and the user identification number and will insert that map into the database and relate it to the user. On success the identification number of the map that was created will be returned.
- “CreateUser” takes the desired username, the hash, the salt, and the email and will insert into the database that user’s information if the username does not already exist. If the user does exist a value of true is returned for the variable “userexists” and a user identification number is not returned. If the user is created successfully the identification number of the user that was created is returned and a value of false is returned for the “userexists” value.
- The “Login” procedure takes the username and the password and will return the user identification number, and the hash for a successful login. A failed login will return a status of fail where a successful login will return a status of success. The

status variable is returned on most procedures and is intended for debugging purposes only.

Minor UI and design improvements were made that focused on mobile users. This development focused mainly on functionality over User Interface (UI) and User Experience (UX) (Figure 4).

STAGE FOUR:

In the fourth development stage, the rest of the features were added into the application. The ability to delete maps, points, and parameter values was added. This was achieved by adding a "deleted" column to the tables in the database. The "deleted" column value defaults to 0. When the object is deleted that value is set to one (1). All of the stored procedures were altered to only select for rows that have a "deleted" value of zero (0).

Sharing was implemented. The owner of a map can share the map with other users. The owner of a map selects the "share" button which prompts the owner to input the username of the user with whom wish to share their map. The application promts the owner to decide what level of permission they want to give the user with whom they are sharing the map. There are three levels of permissions: owner, editor, and collector. The owner is a user who creates a map. The owner can add points, add data, delete points, delete data, share the map, and delete the map. Editors can add points and data, but not delete points or share or delete the map. Collectors can only add and delete data; they cannot add any points, only collect data for points.

The ability to export the map data was added. The points could be exported as KML. The ability to export as a ".xls" file for use in Microsoft Excel and OpenOffice was added. In generated spreadsheet the data was pivoted so that statistical analysis could be performed in a third party application (Figure 5).

Other than the addition of the "deleted" columns, the database structure remained the same. The stored procedures that accessed data were altered to only select rows that had a deleted value of zero (0). The following procedures were created to support the added functionality.

- "AccessMapData" used the map identification number to gather all of the data in a map to be exported into a single spreadsheet.
- "DeleteData" uses the point identification number and parameter identification number to delete a specific data point by setting the deleted column to one (1).
- "DeletePoint" uses the point identification number to delete a specific point by setting the deleted column to 1.
- "DeleteMap" uses the map identification number to delete a specific map by setting the deleted column of that map to 1.
- "ShareMap" uses the map identification number, the username of the user who the map is being shared with, and the intended permissions to give that user to allow that user access to the map by adding a row into the map-user table. The map user table has had a permission column built into it from its creation in the third development stage in anticipation for sharing capabilities.

The UI was not updated in the fourth development stage (Figure 6).

STAGE FIVE:

During the fifth development stage style and user interface changes were made. This stage included the selection of colors, the implementation of buttons featuring descriptive icons to replace text-based links to perform actions, and the finalization of the layout of the application (Figures 7-10).

Results

Based on developer testing the application functions as it is intended. All functionality of the application was tested separately during development. The application was also tested once at the completion of development to ensure that all the components work properly. Any defects found during testing, were fixed. The first alpha version was tested by several users and the developer received no complaints from any of these users. The developer found that on different devices the geolocation accuracy and speed varied, but that cannot be addressed by the application. Based on the CPU utilization of the application on the server, and the payments made for the six months the application was in development (Table 1), the cost for supporting

177 users is $5.76 per month. This was determined by using the average CPU usage percent (0.588%) and the average cost per month ($5.76).

Discussion

Security is one of the most important factors that was considered during the development of this application. A variety of methods were employed to keep the data completely secure. First of which was the implementation of SSL. It is very important to keep all communications secure, especially when those communications are the geographical coordinates of the user. The Google Chrome browser requires a connection to be secured using SSL before it allows the application to access the user's location. No passwords are stored in the database, instead, the practice of hashing and salting is used. The process of hashing and salting involves the following. The user creates an account with a password. A random string is generated and attached to the password. The random string and password combination gets put through a hashing algorithm, in the case of this application, the MD5 algorithm. The hash, which is a string of characters produced by the hashing algorithm, is the same every time when the same data is input to the algorithm (Brookes 2011). The salt, which is the random string that was created when the user account was created, is stored in the database with the hash. The use of the salt prevents hackers from using a rainbow table, which associates values and hashes that the algorithm produces (Defuse Security 2016).

The approach of using a "deleted" column rather than actually deleting the rows allows for data restoration if the user deletes something by accident.

The different permission levels created were designed based on the developer's personal experience collecting data in a group setting. Generally, there were specific points at which many people were collecting data. Each person would be responsible for only a few parameters, and one person would be responsible for keeping track of the data. This application allows the points to be set out beforehand by the owner or the editor of the map. A collector can add data to the points as they collect it.

The map can be exported into a spreadsheet. When the data is exported, it is pivoted so that the points make up the columns, and the parameters make up the rows. This makes comparisons of the point to point data easier for the user (Figure 5).

The cost of running this application will likely vary from the estimate. The more users that use the application, the more processing power the server will need to have. The pricing for the Google Cloud Platform Compute Engine, the server that this application was developed on, does not increase linearly as the power of the server increases (Google Cloud 2016). The number of users that could use the application was determined based on the average CPU usage over a one month period, however all the users are not likely to use the application at the same time, so realistically many more users than 177 can use the application, just not at the same time on the current infrastructure. The variation in pricing per month is because of periodic down time where the virtual machine was restarted.

The cost of this application is much lower than the cost of the application ArcGIS Online, which costs up to $42.00 per user per month (ArcGIS 2016). The application ArcGIS Online is much more extensive than this application; however, GeoCloud is not intended to replace other mapping software. The purpose of this application is to supplement other mapping software, such as Google Earth Pro (which is a free application), to be able to use in some cases, as a replacement for expensive software such as ArcGIS Online. For a user who needs to collect data using a mobile device and then needs to be able to see that data in a spreadsheet and a map so they can interpret it, this application successfully fulfills that need.

Conclusion

This application successfully accomplishes the goal of making available a budget application that allows users to collect data based on geographical points. It successfully allows users to share maps with other users, and allows users to customize the abilities of the users with whom the application is shared. Users are able to export the data and maps to be used in other applications.

Future Work

The following features will be added in the future. The ability to import KML maps. To save maps as images. To do custom map stylings. To add the map features plains and lines. Owners will be able to revoke share priviliges. Users will be able to recover deleted points, data, and maps

Figures and Tables

Table 1. Cost per month of running the server which the application runs on. Cost differences are attributed to varying server downtime.

July	August	September	October	November	December	Average
$5.95	$5.95	$5.76	$5.95	$5.77	$5.18	$5.76

Figure 1. Screenshot of the user interface of the first stage of development.

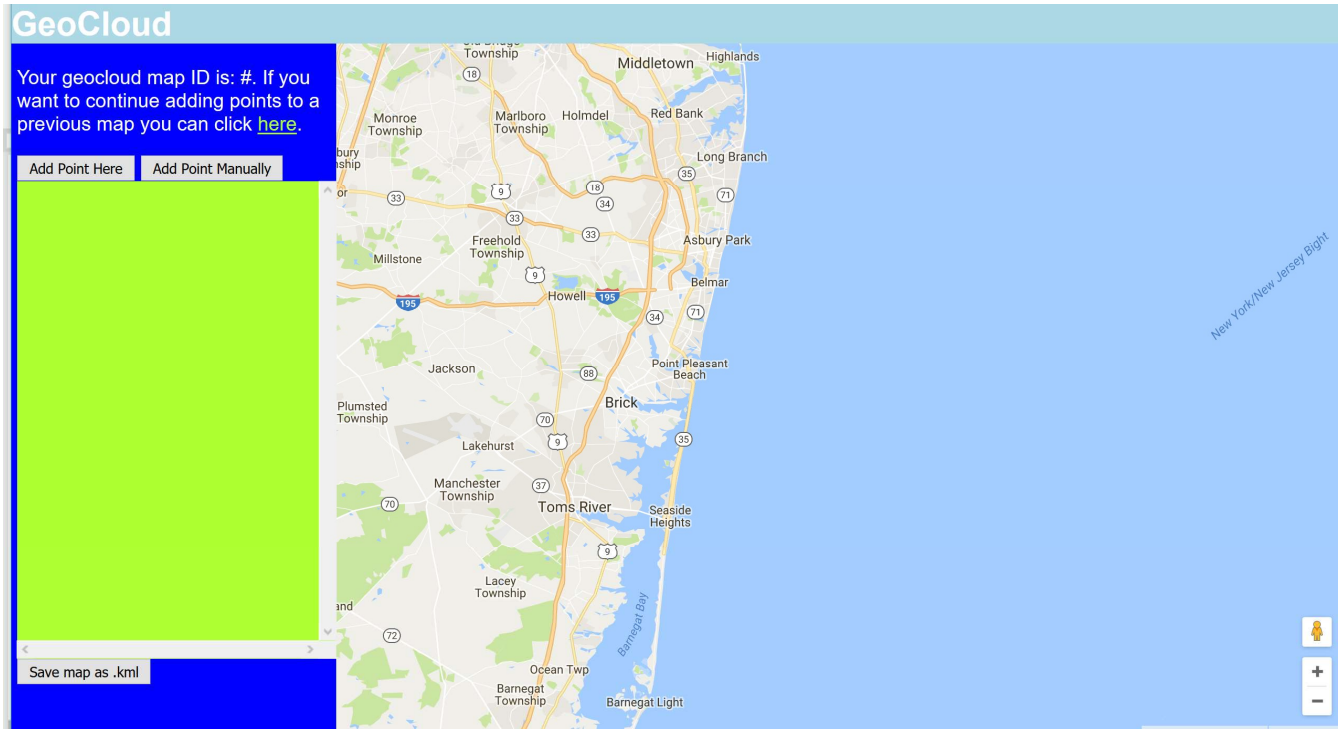


Figure 2. Screenshot of the user interface of the second stage of development. On the left is the navigation pane where the user can see the points they have added, and add more points. The map identification number is displayed so the user can access the map again at another time.

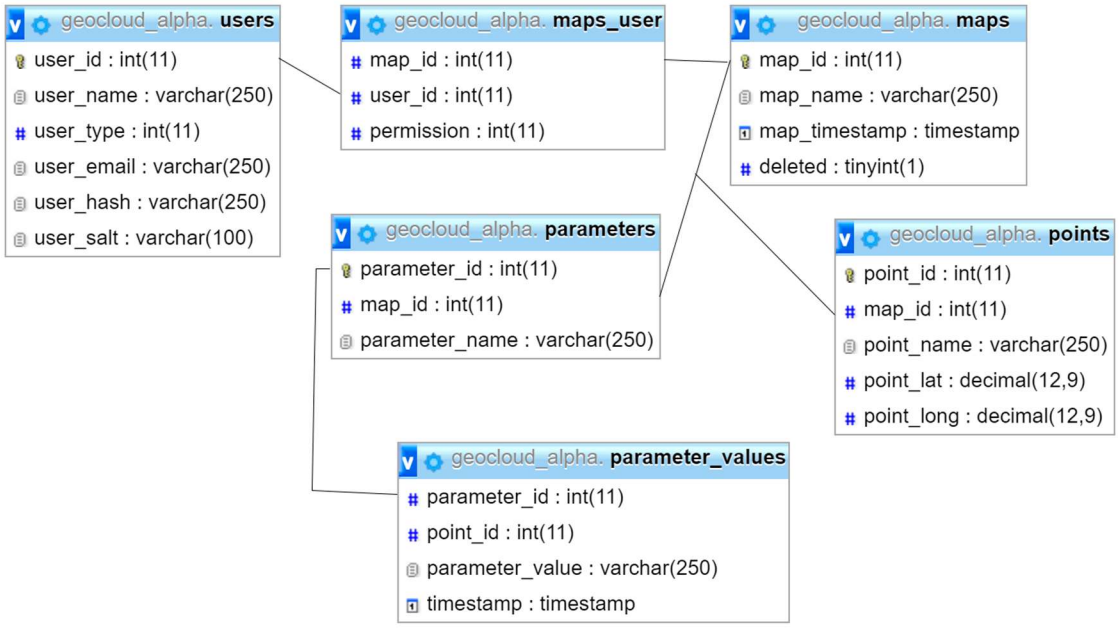


Figure 3. Diagram showing the relational database structure of the third development stage.

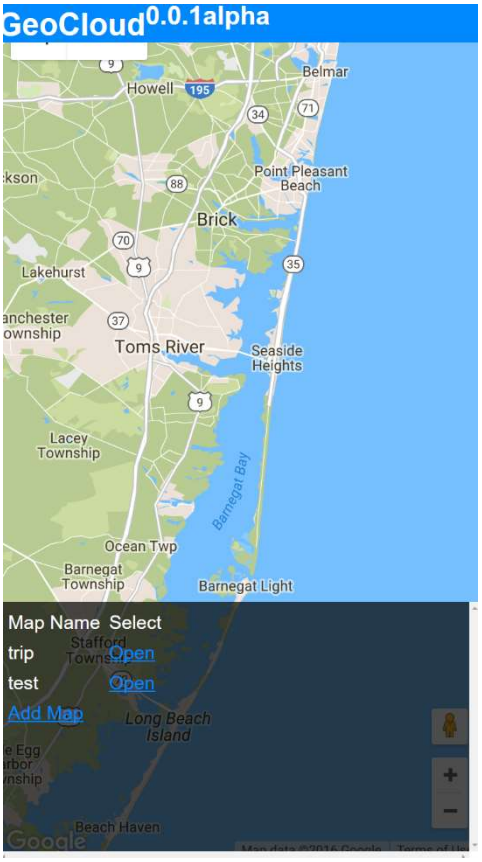
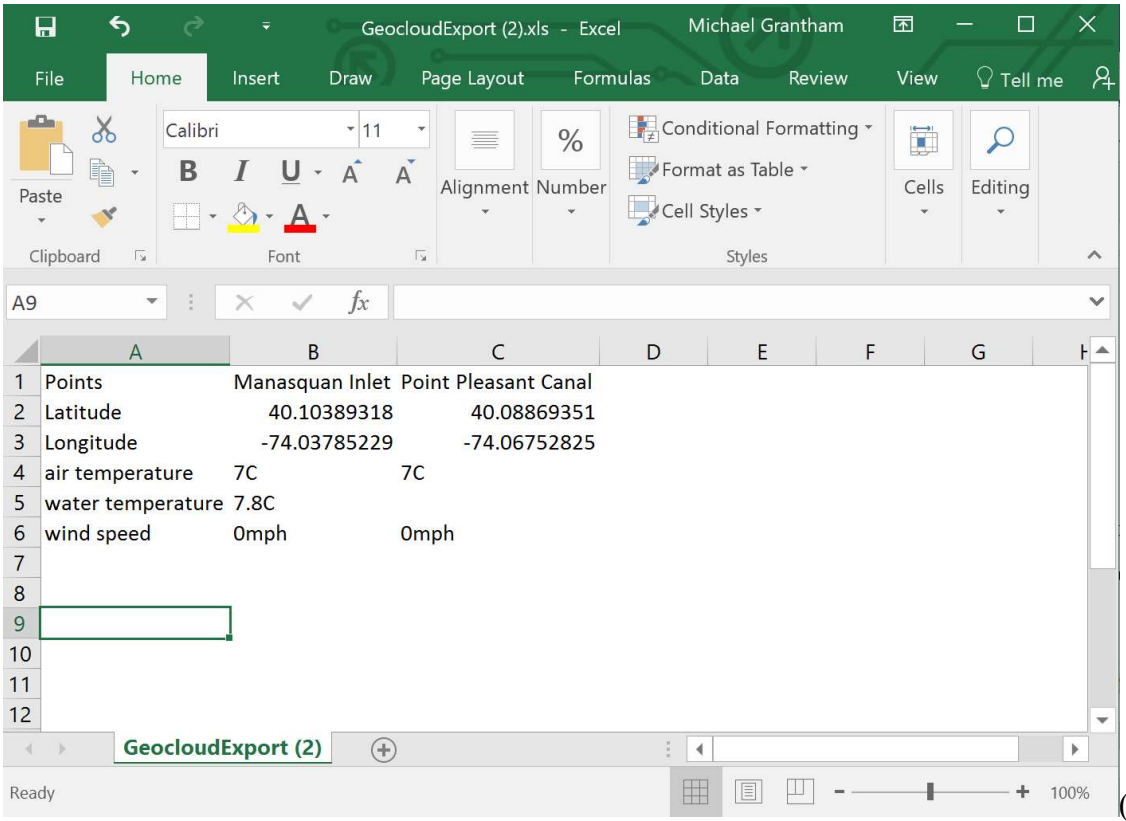


Figure 4. The third development stage focused mainly on functionality improvements, but focused all user interface (UI) improvements for mobile users.

point_id	point_name	point_lat	point_long	parameter_name	parameter_value	parameter_id
26	Manasquan Inlet	40.103893178	-74.037852287	water temperature	7.8C	15
26	Manasquan Inlet	40.103893178	-74.037852287	air temperature	7C	14
26	Manasquan Inlet	40.103893178	-74.037852287	wind speed	0mph	16
27	Point Pleasant Canal	40.088693511	-74.067528248	air temperature	7C	14
27	Point Pleasant Canal	40.088693511	-74.067528248	wind speed	0mph	16

(a)



(b)

Figure 5. The table shown above (a) is the result of running the MySQL stored procedure "AccessMapData". That data is manipulated in a PHP script to be displayed in a manner which makes more sense to a user which is shown immediately above this caption (b). Any data that was not input to the application leaves a blank cell in the spreadsheet.

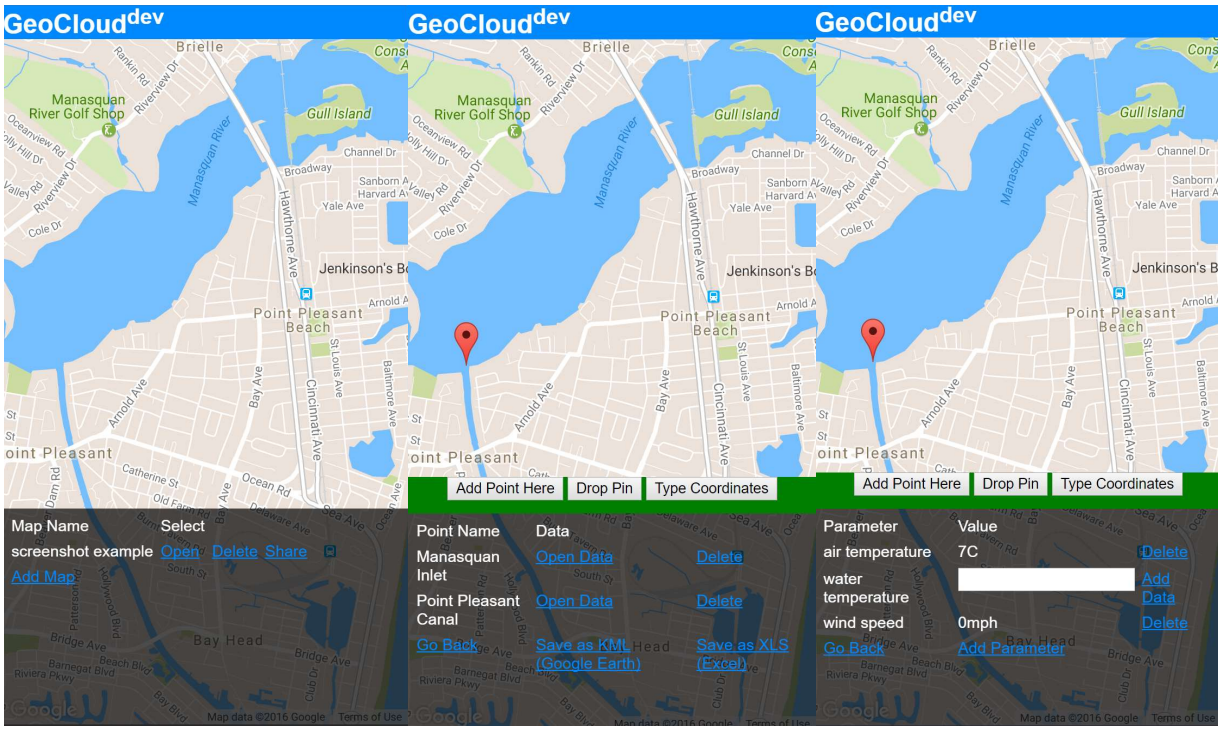


Figure 6. Screenshots of the fourth stage of development. No user interface (UI) changes were made between the third stage and the fourth stage, but the added functionality can be seen in these screenshots. The first one on the left shows the map can be deleted and shared. The second screenshot shows where the map can be exported to KML and XLS files. The second screenshot also shows where points can be deleted. The third screenshot shows where data can be added and deleted. The action bar that has the buttons for adding points will not be displayed for users with the "collector" permission.

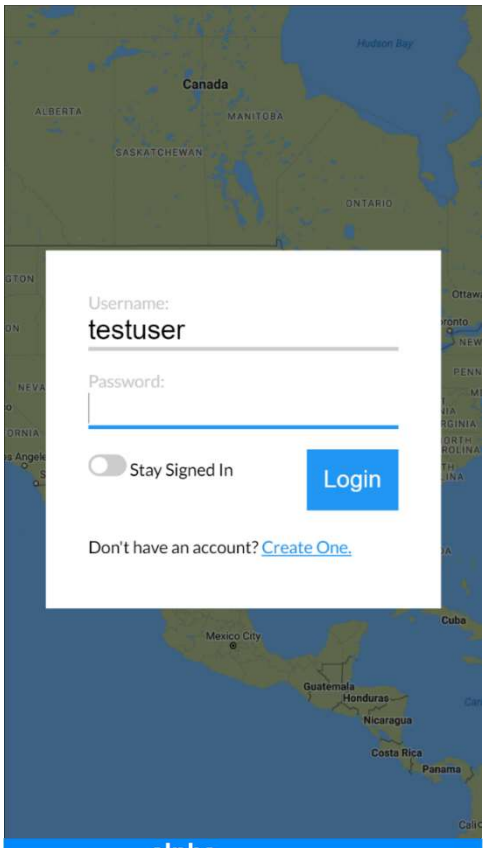


Figure 7. This screenshot shows the Sign In screen of the application on mobile.

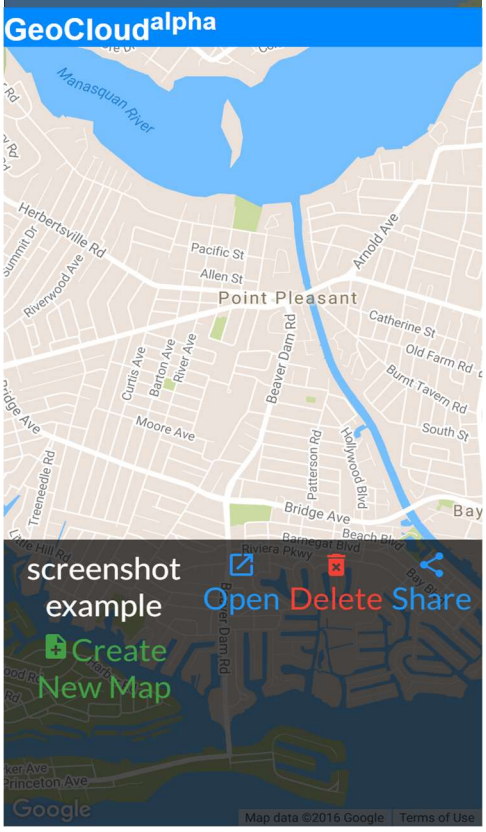


Figure 8. This screenshot shows the main screen of the app. This page allows users to manage their maps.

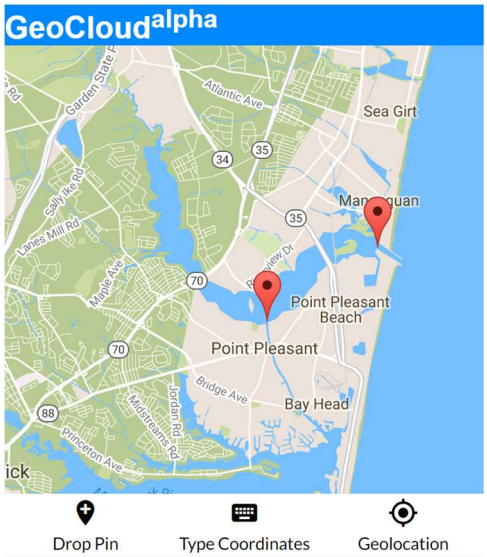


Figure 9. This screenshot shows the points screen of the app. Here users can add points, add data to the points with the pencil button, delete points with the trash can, and export the map.

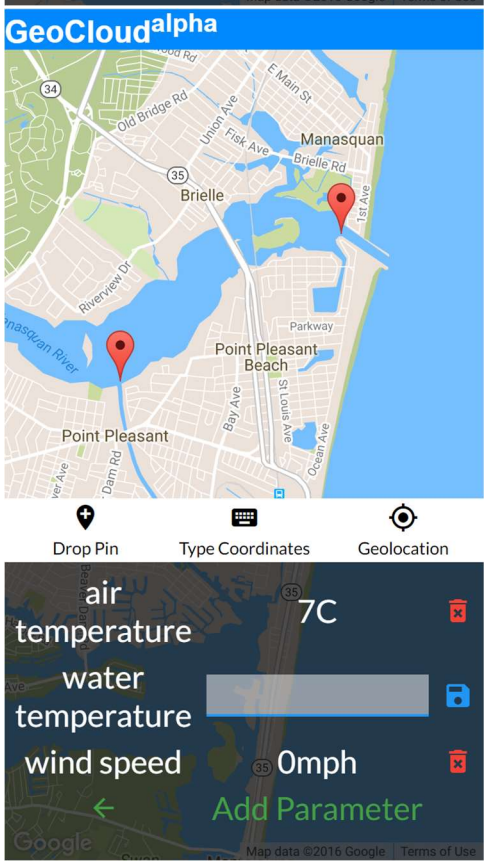


Figure 9. This screenshot shows the data screen of the app. Here users can view data, delete data with the trash can button, add data with the floppy disk button, and add parameters with the "Add Parameter" button.

References

Achour, M., Betz, F., Doval, A., et. al. (2014, December 26). PHP Manual Retrieved from http://php.net/manual/en/

Brookes, T (2011). What All This MD5 Hash Stuff Actually Means. Retrieved December, 2016, from http://www.makeuseof.com/tag/md5-hash-stuff-means-technology-explained/

Carter, J (2012, October). Cloud Computing Explained. Retrieved December, 2016, from http://www.techradar.com/news/internet/cloud-computing-explained-1105688

Defuse Security. (2016). Salted Password Hashing. Retrieved December, 2016 from https://crackstation.net/hashing-security.htm

Foote, K (2015). The Value of Maps. Retrieved December, 2016, from http://www.colorado.edu/geography/gcraft/notes/cartocom/cartocom.html

Google. (2016, February 9). KML Documentation. Retrieved June 13, 2016, from https://developers.google.com/kml/documentation/

Google Cloud (2016). Google Cloud Compute Engine Pricing. Retrieved December, 2016, from https://cloud.google.com/compute/pricing

Google Maps. (2016). Google Maps JavaScript API. Retrieved June 13, 2016, from https://developers.google.com/maps/documentation/javascript/

jQuery. (2016). jQuery Documentation. Retrieved June 13, 2016, from http://api.jquery.com/

MySQL. (2016). MySQL Documentation. Retrieved June 13, 2016, from http://dev.mysql.com/doc/

Rashid, F (2016, March). The dirty dozen: 12 cloud security threats. Retrieved December 2016, from http://www.infoworld.com/article/3041078/security/the-dirty-dozen-12-cloud-security-threats.html