The Lyons Flood Photography Project

Forrest J. Bowlick, Alec Kelly, Florita Rodriguez, and William Zoeller

GEOG 489/689, Texas A&M University, Spring 2014



**The St. Vrain River cuts through major infrastructure, isolating the town of Lyons, in this September 13, 2013 photo. Photo Credit: AP, John Wark.**

**Abstract**

This paper describes the steps necessary to create the Lyons Flood Photography Project website, a component of the course grade in GEOG 489/689 (Web GIS) at Texas A&M University in Spring semester, 2014. This project built a website that allows users to upload photos of Lyons before, during, and after the major flooding in September, 2013. We used a combination of Google Fusion Tables, Google Documents, and an interactive web form to obtain data from users, and then displayed the images on a Google Maps interface. While we made significant progress with SQL and JSON using leaflet APIs, ultimately, the issues encountered with uploading and retrieving image files proved untenable, and we initiated a backup Google product option. This paper describes the components, which composed the composition of the website, offers screenshots and descriptions of the general construction of the site, and discusses potential future uses for the website or similar versions.



**The rising floodwaters covered the entire Loukonen Stone business in Lyons. Photo Credit: AP, Brennan Linsley**

**Background**

Severe flooding across the Colorado Front Range in September of 2013 resulted in over $1 billion in damages across the state. Floods severely damaged many mountain towns, including Estes Park, Jamestown, and Lyons. The widespread flooding destroyed extensive stretches of infrastructure, isolated thousands of residents, and destroyed numerous dams, ditches, and other agricultural and municipal water supplies in the region. Further downstream, the South Platte River, normally a minor tributary of the Missouri River, reached peak flood stage at a flow rate outpacing the Mississippi River at St. Louis. With the immense destruction across the Front Range, efforts to preserve the history of the event will form a long lasting connection with the human aspect of this disaster.

**Project Purpose**

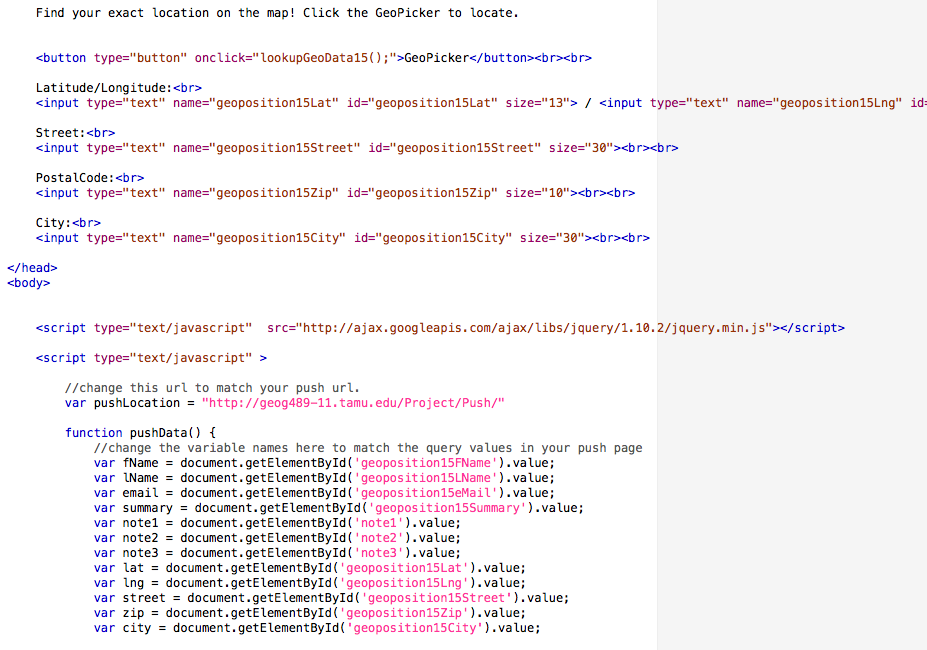
The overall purpose of the project was to provide a stable web platform where users could upload imagery of the Lyons flood for display on a map interface. Our primary objectives were to create a website that allowed users to upload photos and see them displayed on a map interface. The next sections below describe the essential components, which build this website: the database, the web design, and the server connections. Also below is a description of the Google components that formed the eventual backbone of the functioning website.

**Database**

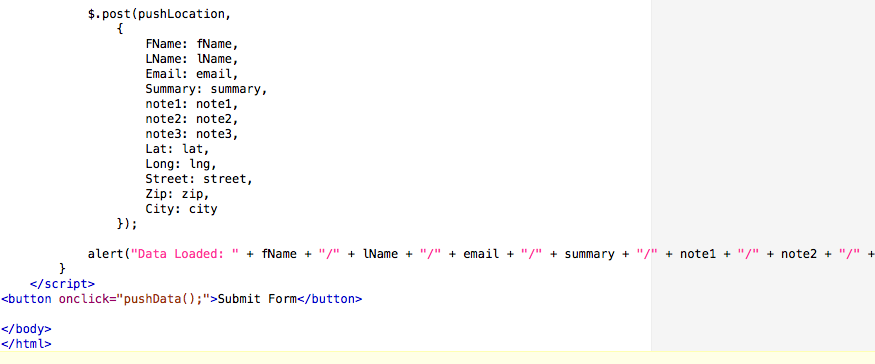
The original database form used SQL to establish connections between the user input from the web form and the SQL Database. Since the entire project centered on the idea of providing a service for the town of Lyons, Colorado, it would make sense to allow the citizens to the ability to submit their picture and information. The structure of the web form allows users to submit their name (first and last), address, images, and notes concerning the image. An initially empty table created in SQL received this information submitted through the web form. Code from a previous lab allowed for us to implement JQuery in order to extract the user input. The variables then push to the C# page. The JQuery values insert into the SQL database for storage. Figure 1 shows the gathering of document elements, and Figure 2 shows the transmission to the push location. Figure 3 describes how the variables input into the database. Finally, Figure 4 shows an image of the populated database.

From the screen shots, it is apparent that this does not deal with the upload of the images. Our plan was to have the images submitted from the user saved in a folder created on the server. While this method is not ideal for security reasons and architectural reasons, we decided upon uploading images through this method for simplicity’s sake. Another group provided the direction needed to set up the image uploads to the server. However, when trying to format the functions and code for our use, complications arose and rendered this option obsolete. Figures 5 and 6 show our attempt at creating a folder on the server and uploading an image provided by the user to the folder. We believe the main mix up for this method arises the call from each function. We were never able to establish the correct functionality for the functions, and their performance when the user hit the ‘Submit Form’ button on the web form. Although we were not able to get the image upload aspect of our project to work, the alternative method implementing fusion tables worked well for our result.

See Figure One below.

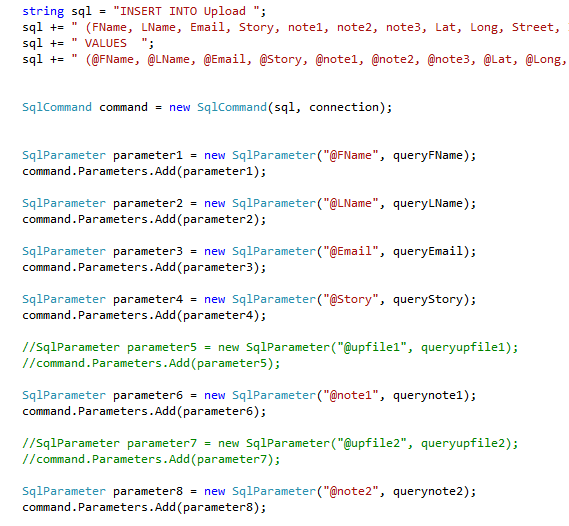


**Figure One. Gathering elements from the web form.**

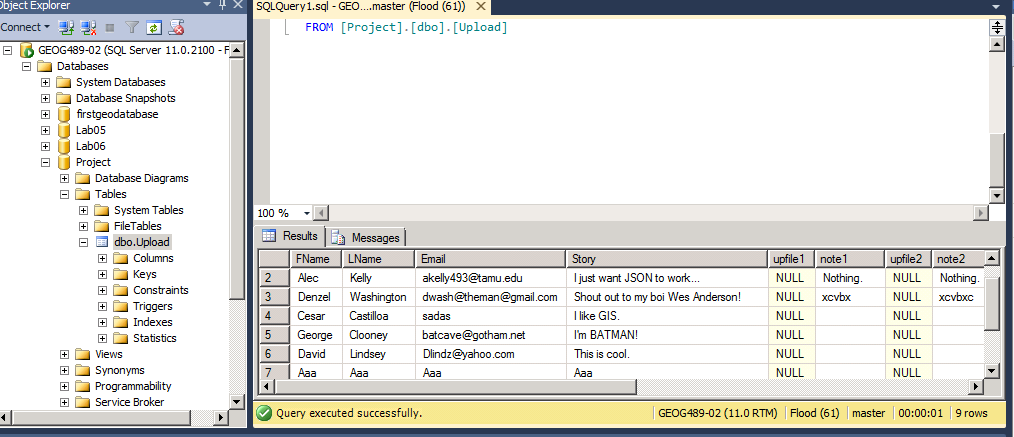


**Figure Two. Transmission to push location.**

See Figure Three on the next page, below.



**Figure Three. Input of variables into database.**



**Figure Four. The populated database.**



**Figure Five. Server folder attempt, part one.**



**Figure Six. Server folder attempt, part two.**

**Web Services**

The process of creating a functioning web page with GIS mapping capabilities is a multi-tiered project with different parts that require connection to function properly in order to accomplish the result. In the attempt to develop a web mapping application for the city of Lyons Colorado, each project piece was broken up among group members. This portion describes the hosting of the website along with the task of generating JSON and linking documents within the index page on the server.

To begin, we knew as a group that the client would want a website where they could actively search and link to other members within the community. We did not want to purchase a domain name because of the cost and we were not sure whether the city wanted us to email the HTML and code pages or if they were expecting us to fully operate and support the map. In order to host it publicly to the client during the course of the project we had to request an opening of the virtual machine port, which would allow the client to access the web page information on our server. This provided an easy way of access for the client and allowed members of the group to access the page from other computers not connected to the TAMU Geography servers. This allowed multi-platform review, in order to think of potential changes and steps that necessary for accomplishment before completion of the final product.

With the page accessible by anyone with knowledge of the document pathway within the server, it was important that we set up a file hierarchy so only the actual map page popped up when the server URL was entered into the query string on the client side. This required establishing the main web page as the index file within the server director. Additionally, setting up the other html and aspx pages within separate file folders required connection through an “<a href>” command in the main index page. Doing this serves a couple of purposes for the overall project design. Primarily, it condenses all of the information from all of the individual pages into one location. Therefore, instead of having to add all of the code within the submission page, Facebook and email links, city website, and TAMU Department of Geography site, we could create hyperlinks within the index code that simply references those pages making the code smaller and more manageable. Lastly, linking these documents makes it easier for the users who may interact with the page as the “<a href>” creates a sort of hyperlink that will take users to the correct page after a simple click instead of rerouting them through numerous pages not important to them. With the proper setup in order, the process of developing the web mapping application and generating the page with information becomes an accomplishable goal.

Generating JSON is crucial for any web GIS application because it is the JSON that proves the underlying database is correct and connected to web server. The JSON also produces the discrete data types such as points, lines, and polygons used within web mapping applications. In order to get this fundamental piece of the project to function properly required significant outside work and discovery. Using information and pieces of code from earlier in the semester, we hoped to replicate previous labs and get the JSON to ‘spit’ and generate points on a map. However, this task ended up being much more difficult than copying and pasting lines of code. In order to get JSON to output properly onto the webpage you have to connect your database information containing user submissions to the “Get” code that pulls from the server and pushes the extracted data to the webpage. This required the adjusting of code within the existing code fragments to match the access information of our database, such as My SQL Server Username, Password, Database, and Virtual Machine. Furthermore, information such as column names, table names, and WKT had to match within the “Get” pages of code attached to the website. Once this process was complete and the “Get” page connected to the database, using Visual Studio we were able to run the page code and spit JSON, which would be a big step towards generating the map feature points for the project.

**Web Design**

It was important to make the layout of the website simple and user friendly while incorporating elements of the main Lyons Colorado website. The website user interface went through three iterations before reaching a final product. The various versions were due to use of varying APIs. Markers within the map element contained info windows with images of the area and a brief description of the images and anecdotal information of the scene.

The first version of the website contained two web pages: Index and an Input page (Figure 7). The index page consisted of six div elements, a header, a navigation bar, two content columns on the left, a main content page for the map element, and a footer. The architecture of the page used Google API, SQL Server. The map display pulled data from a SQL through C# aspx files to populate the page with markers. The site’s second page was an input page used to allow user input into the SQL server, for populating new marks on the map in the index page. The Input page opened in a new window upon link click from the index file. The input page has forms for input with the inclusion of a second Google map where the user could click their location and an ‘on click’ command would populate the longitude and latitude in the form for the SQL server submission. Upon submitting the form data, the input page closes and returns the user to the index file.

A second version of the site used less div elements. This was to make the page simpler and more user friendly. The site included theme elements from the Lyons Colorado website. The color scheme and many of the city image elements copied over so the user would feel as if the project webpage was an extension of the town’s existing site. See Figure Eight.

The second version differed architecturally from the previous version by rejecting Google Map API and adopting Leaflet API for the map element. This approach arose due to problems with uploading file content into SQL server from the initial input page. The functionality of the second page was the same as the first version with added user friendliness and simplicity.

The third version of the page did not differ in organization from the second. The user interface remained the same. Additional elements were included such as hyperlink to the Lyon’s Recovery Facebook group, an email link to the city management, a link to Texas A&M University Geography Department, and copy write elements in the footer.

The architecture for the third version used Google Fusion tables and the map element returned to Google Map API. We rejected the original Input page and replaced it with a link to a Fusion Form for data submission. This was the best, most efficient means of populating data into the server and displaying the information in a simple and understandable manner.

Problems that arose while developing the interface consisted of properly displaying the map inside the respected div elements. In the first version the map would not extend out the length of the frame, leaving approximately 30% empty space on the right margin. We corrected this bug in the third version. However, the new div elements conflicted with approximately 25% of the map element. This was problematic because users could not use the aspects of the map such as switching from map view to satellite view. We introduced z-index elements to create priority. We gave greater priority to the map element.

The final site gave users a simple interface to input their personal data, read about the history of the city and the effects of the flood, and allow for user input into the map for populating new markers for the map. The site’s simplicity would allow most users the ease of performing all functions without requiring complicated instructions (Figure Nine). The color scheme matched that of the existing site, which helps streamline the sites and suggest that it is merely an extension of the original site. The incorporation of all of these components would allow an intuitive, user-friendly, and simple method for learning and archiving the city’s flood history.

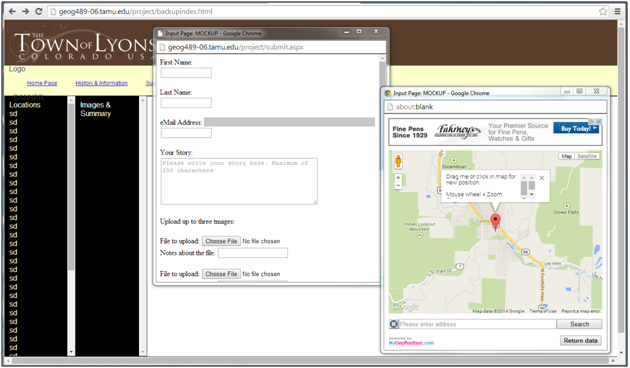
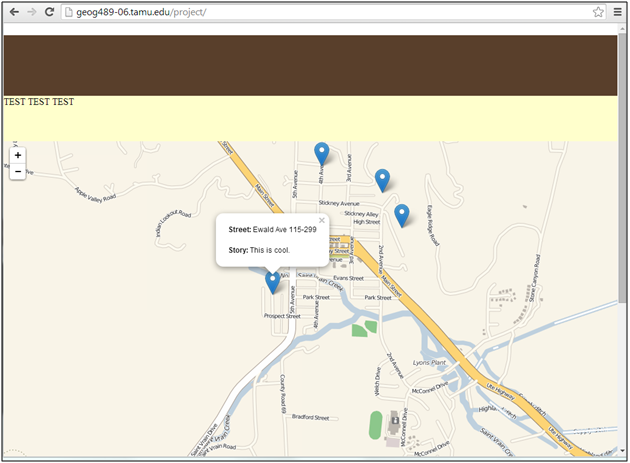
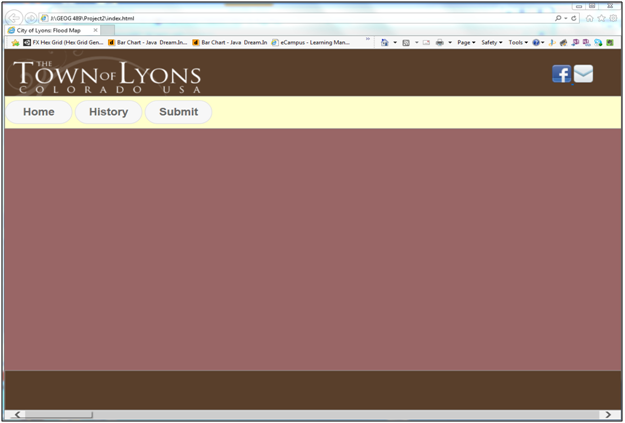


Figure Seven. Original website layout.

See Figure Eight on the next page, below.



**Figure Eight. Second site design.**

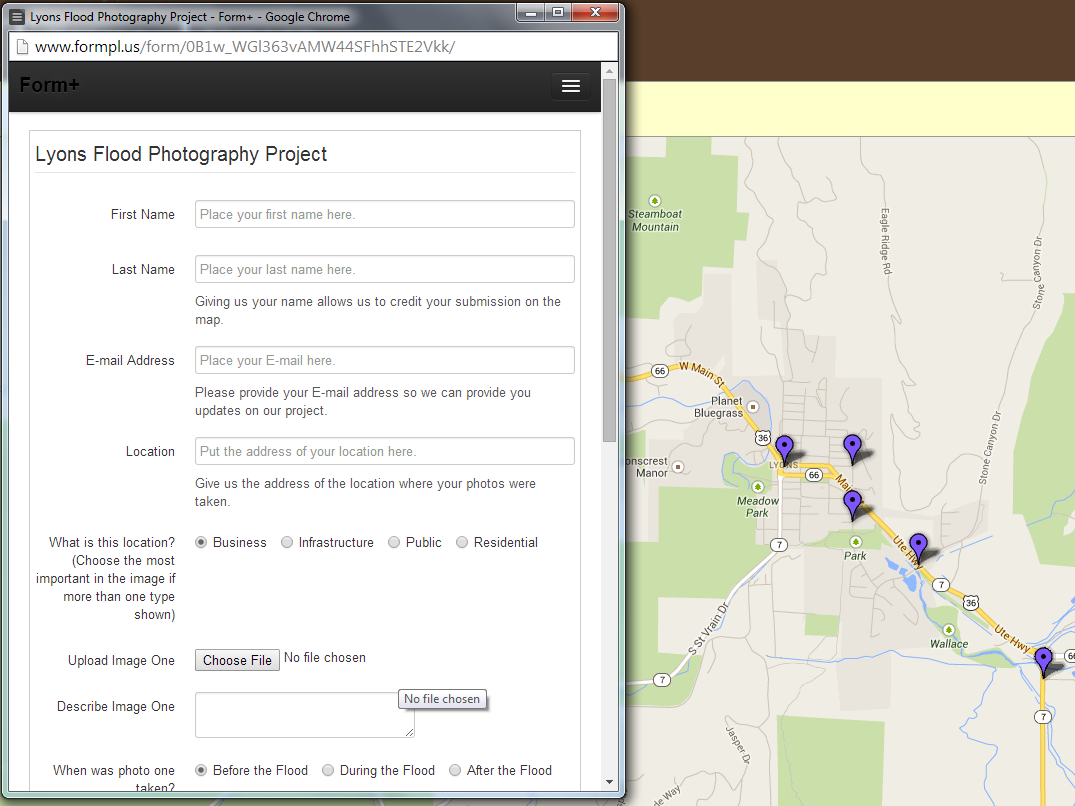
****

**Figure Nine. Final design shell.**

**Google Services**

After encountering severe difficulties with the SQL file upload, we completed a ‘back-up plan’ to use Google services to allow users to submit files. Then, using Google Fusion tables, we were able to populate a Google Map with the submitted points.

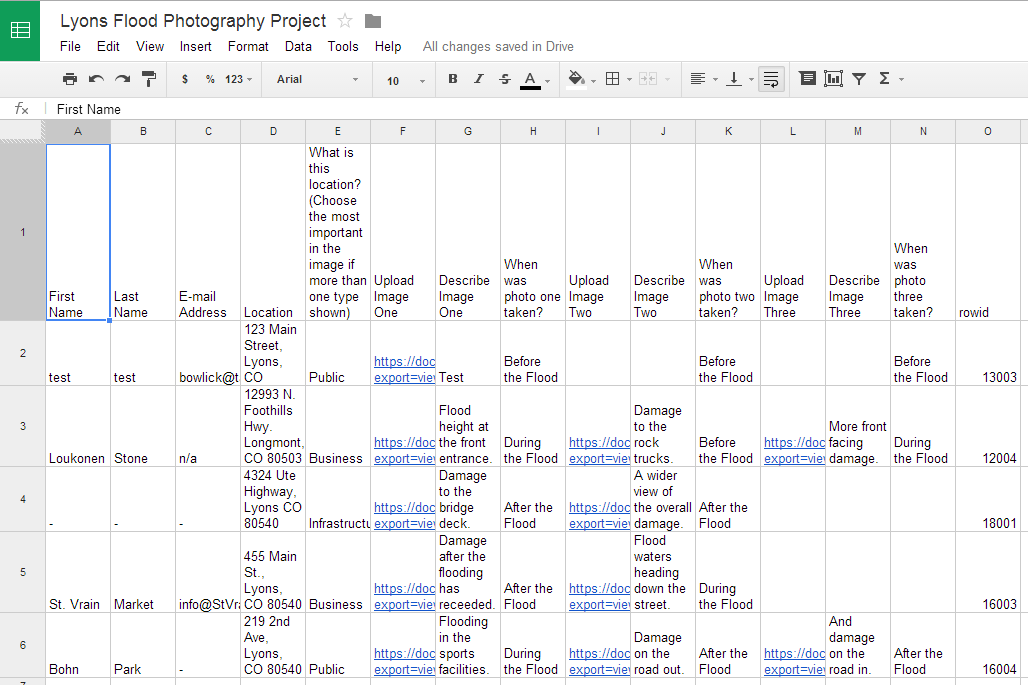
First, we used the ‘Form +’ extension for Google Forms. This allows users to upload files. The interface for our Form + submission is Figure Ten below.



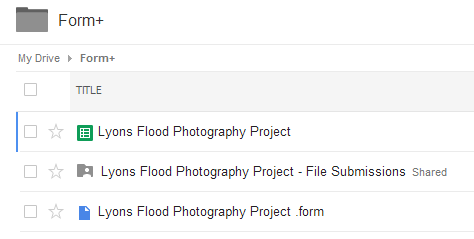
**Figure Ten. Form + submission with website in background.**

This form submits information to a Google Spreadsheet within a personal Google Drive. The images upload to a separate folder within the Google Drive. This set-up is easy, and does not require database knowledge or other technical and precise training to perform. Any web-familiar user could set up this form, making it easy for quick use or activation.

The connections between the form and the spreadsheet are in Figures Eleven and Twelve, below.

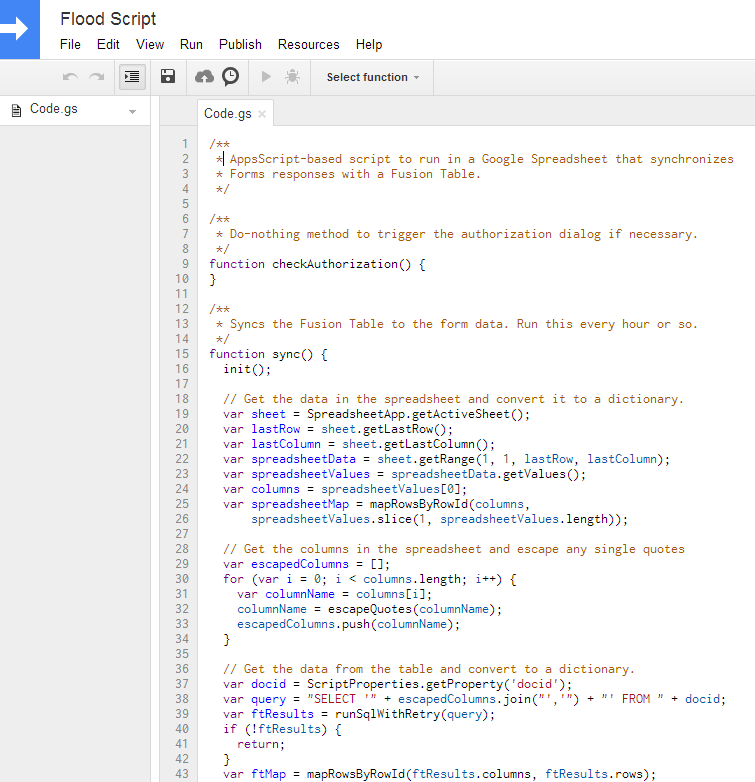


**Figure Eleven. The spreadsheet of inputs.**

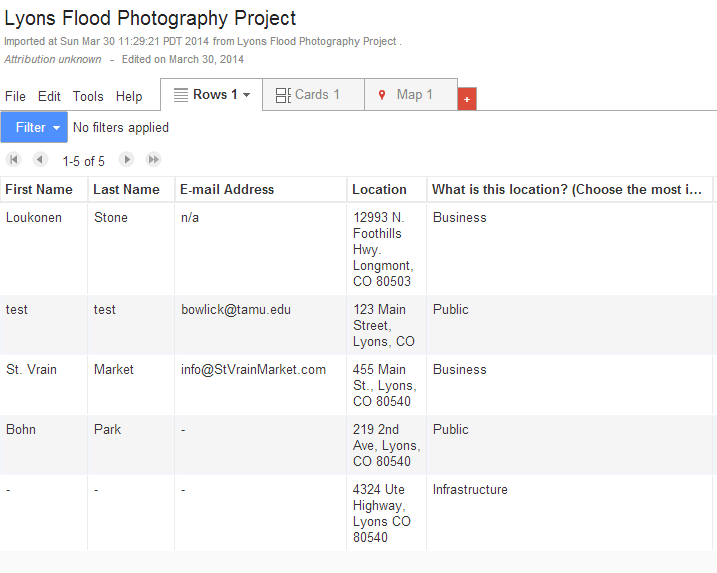
****

**Figure Twelve. Folder Hierarchy.**

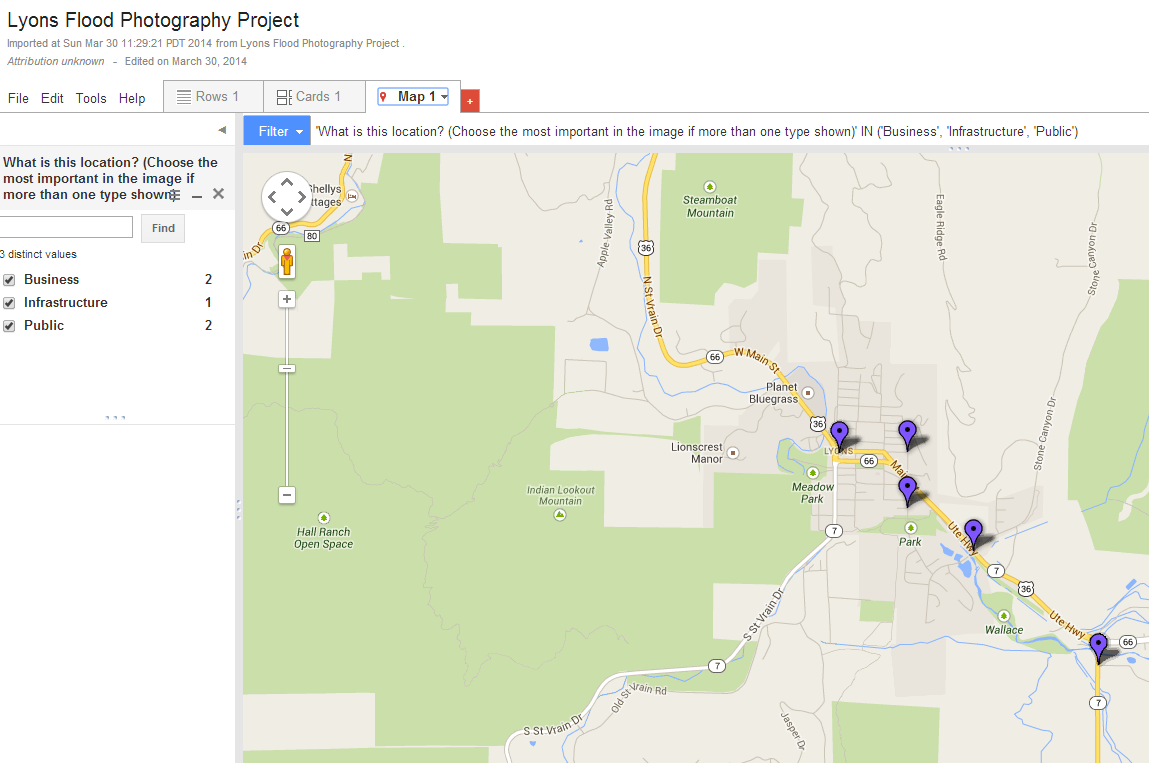
Within the spreadsheet, a small script updates a corresponding Google Fusion table. Since Google Fusion Tables are a current experimental form of Google Document, they do not currently allow direct input. However, a vibrant scripting community develops relevant scripts and add-ons to increase functionality. Since Google Fusion Tables contain mapping capabilities within its general framework, mapping is a simple task, as marker icons, marker display, and other tasks are completable within the table. Figures Thirteen, Fourteen, and Fifteen show the scripting, Fusion Table format, and mapping capabilities within this service.



**Figure Thirteen. Google Fusion Table update script from Google Spreadsheet (portion).**

****

**Figure Fourteen. Google Fusion Table portion.**

****

**Figure Fifteen. The Google Map interface within Google Fusion Tables.**

These services provide the functionality for the website.

**Discussion and Conclusion**

This project managed to complete its primary objectives, of creating a usable webpage where interested users could upload photos, which displayed on a web map. However, the completion of this page occurred with significant difficulties in file upload using traditional database programs. The Google Fusion Table approach is functional, and may be easily replicable for other entities that wish to set up a public forum for discussion. All file upload and display interfaces have risk of malice, but this seems an easy way to access files.

Overall, this project managed to complete a workaround to complete the objectives of the project. While not optimal in database format, it is something that can be handed off to the city for later adaptation or use.