DEPARTMENT OF COMPUTER SCIENCE
COLLEGE OF ARTS AND SCIENCES
CSCI 4961/4962 Capstone Deliverable #3

Title of Project:	Library Book Finder
Client:	Lee Cummings
Supervisor:	Dr. Michael Goldwasser
Student(s):	Sara Jain, Lorna Xiao

Project Overview

Our project, "Library Book Finder", will allow students and other individuals to locate a book in the library. It will primarily focus on enabling user-friendly accessibility to shelves that correspond to library books through Google Maps or a text message with a link to Google Maps.

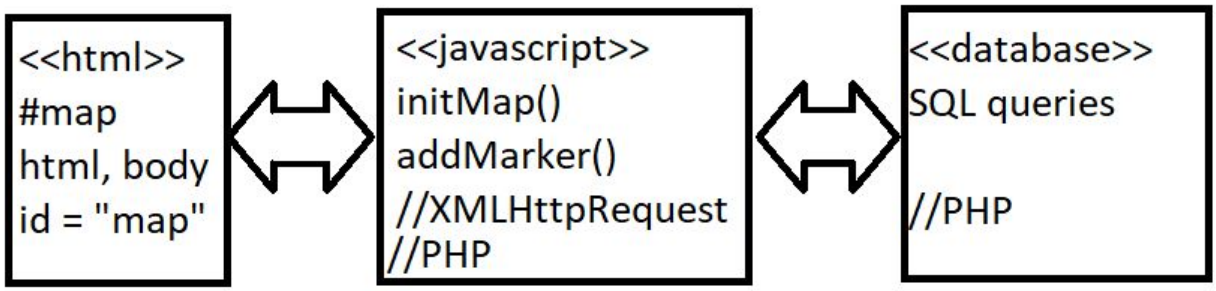
Basic Functionality

When an individual finds the book he or she would like to locate on the library website, there will be a button that prompts two options for users: Send a Text Message or Link to Google Maps. The "Send a Text Message" will allow the user to provide their phone number and get a text with the book's call number and link to Google Maps. Maps will indicate a given shelf location and the user will be able to navigate through the Maps app to that shelf (with their real time location or predetermined paths). With the call number, they can find the location of the book on the shelf. The other option, "Link to Google Maps" will take the user directly to Google Maps on their current device and show him or her exactly where the shelf (containing the book) is located. The library has these buttons ready or they can be quickly completed. Our project focuses on creating the Google Maps portion and connecting it to the Library site.

Major Components

- Google Maps API
    - HTML/JavaScript/PHP/Google Maps API/Directions API
    - Basic interface on Google Maps, used to create markers
    - The Maps API will receive a call number from Library Catalog Button. It then sends that call number over to DB to query. It will receive coordinates from DB and output a marker on the shelf location of the call number. Then, it directs user to that shelf based on a predetermined path using Directions API. The map will be a link that is sent over to the library site.
- Call Number Database
    - SQL/PHP
    - Holds all call number and shelf location information.
    - The database receives a call number from the client side, queries it for a shelf range. Then, it will send coordinates/floor location to the client side.
- Library Button
    - PHP
    - Holds the user's desired call number and link to our Google Maps API
    - The button will send the call number to the Maps API and receive a link to Google Maps. The link will include marker and path information.
    - **Currently, the library has set up a separate button/link for testing purposes. It will be used to test the database queries.**

UML Diagram

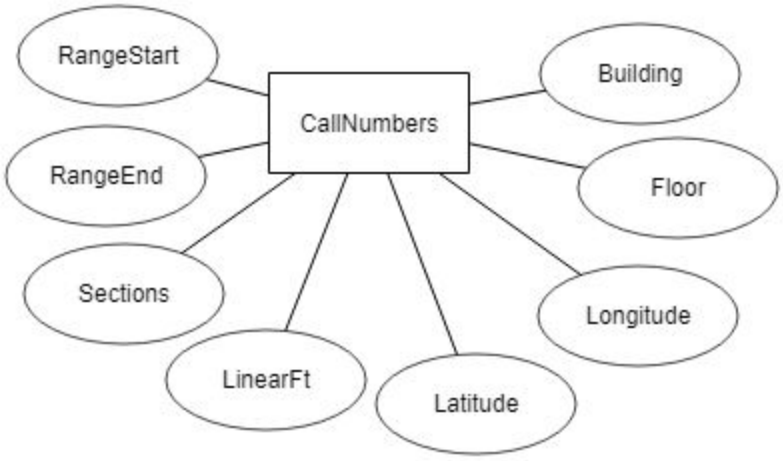


<<html» simply creates a basic webpage and sets the size of the map. We have given an id to our map called “map”.

<<javascript>> holds all of the information to initialize and to interact with the map. It also has the link to the Google Maps API with our API key. We are currently writing the PHP code to send/receive data from the database.

<<database>> holds a table called CallNumbers with all call number ranges and coordinates. We are currently writing the SQL queries and the PHP to get/send information.

ER Diagram



CallNumbers is the table, and each attribute is a column in the database. There are no foreign/primary keys. We only have one table containing all of the shelf information, so there are no relationships with other tables.

Google Maps Code

The code mainly consists of PHP to use retrieved data to create a link, which is currently outputted in HTML. Here are the main snippets of code.

```
$query = mysqli_query($conn,$sql);
```

```
if (!$query){
        die("Bad query.");
}
else{
        echo "Successfully retrieved data. ";
        if (mysqli_num_rows($query) > 0) {
                //This just gets the first array because query should return (1) result
                $row = mysqli_fetch_array($query);
        }
}
```

```
//Extract the book's coordinates and floor data
//$row is from query.php
$booklat = $row['Latitude'];
$booklng = $row['Longitude'];
$floor = $row['Floor'];
```

```
//Create a marker on the map
$markerurl = 'https://www.google.com/maps/place/' . $coords . '/@' . $coords . ',' . $zoomlevel
. 'z/' . $flooraddr . $latitude . '!4d' . $longitude;
```

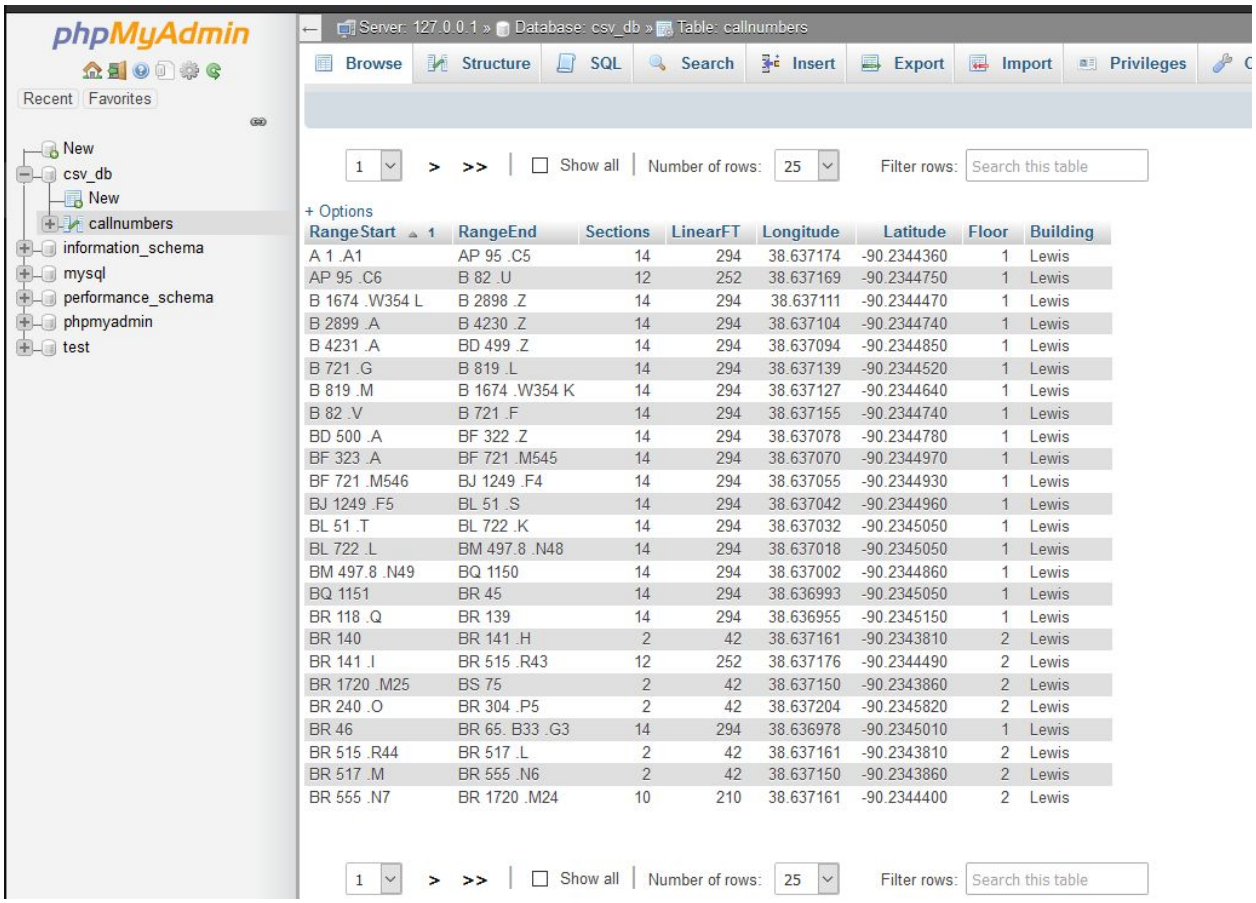
```
<a href = '<?php echo $markerurl; ?>' ><h3>This is a link to Google Maps with a marker on
this coordinate</h3></a>
```

Creating paths is currently in progress. To make a basic path URL, it is:

```
$dirurl = 'https://www.google.com/maps/dir/?api=1&origin=' .$fromcoords.'&destination='
.$tocoords. '&travelmode=walking';
```

However, since latitude and longitude coordinates do not consider height (as in different floors), it does not properly display path from the origin to destination. Unlike the marker's link, the $flooraddr (hexadecimal code) that properly shows the floor, is not as straightforward. This is because we must consider displaying the first floor first, and then the different floor plan at the user's destination.

Database



RangeStart	RangeEnd	Sections	LinearFT	Longitude	Latitude	Floor	Building
A 1 .A1	AP 95 .C5	14	294	38.637174	-90.2344360	1	Lewis
AP 95 .C6	B 82 .U	12	252	38.637169	-90.2344750	1	Lewis
B 1674 .W354 L	B 2898 .Z	14	294	38.637111	-90.2344470	1	Lewis
B 2899 .A	B 4230 .Z	14	294	38.637104	-90.2344740	1	Lewis
B 4231 .A	BD 499 .Z	14	294	38.637094	-90.2344850	1	Lewis
B 721 .G	B 819 .L	14	294	38.637139	-90.2344520	1	Lewis
B 819 .M	B 1674 .W354 K	14	294	38.637127	-90.2344640	1	Lewis
B 82 .V	B 721 .F	14	294	38.637155	-90.2344740	1	Lewis
BD 500 .A	BF 322 .Z	14	294	38.637078	-90.2344780	1	Lewis
BF 323 .A	BF 721 .M545	14	294	38.637070	-90.2344970	1	Lewis
BF 721 .M546	BJ 1249 .F4	14	294	38.637055	-90.2344930	1	Lewis
BJ 1249 .F5	BL 51 .S	14	294	38.637042	-90.2344960	1	Lewis
BL 51 .T	BL 722 .K	14	294	38.637032	-90.2345050	1	Lewis
BL 722 .L	BM 497.8 .N48	14	294	38.637018	-90.2345050	1	Lewis
BM 497.8 .N49	BQ 1150	14	294	38.637002	-90.2344860	1	Lewis
BQ 1151	BR 45	14	294	38.636993	-90.2345050	1	Lewis
BR 118 .Q	BR 139	14	294	38.636955	-90.2345150	1	Lewis
BR 140	BR 141 .H	2	42	38.637161	-90.2343810	2	Lewis
BR 141 .I	BR 515 .R43	12	252	38.637176	-90.2344490	2	Lewis
BR 1720 .M25	BS 75	2	42	38.637150	-90.2343860	2	Lewis
BR 240 .O	BR 304 .P5	2	42	38.637204	-90.2345820	2	Lewis
BR 46	BR 65. B33 .G3	14	294	38.636978	-90.2345010	1	Lewis
BR 515 .R44	BR 517 .L	2	42	38.637161	-90.2343810	2	Lewis
BR 517 .M	BR 555 .N6	2	42	38.637150	-90.2343860	2	Lewis
BR 555 .N7	BR 1720 .M24	10	210	38.637161	-90.2344400	2	Lewis

Query to locate coordinates

SELECT * FROM `callnumbers` WHERE (‘QB801.7 .523 1985’ BETWEEN `RangeStart` AND `RangeEnd`)

Formal Requirements

The following list shows the necessary requirements to create a successful program that meets the goals of the project.

Requirement	Description	Progress
1A	Overlay on top of the bird’s eye view of the library floor plans on Google Maps	Completed
1B	Overlay must be clickable (interactive) so that the user can see the call number ranges on each shelf Unnecessary from a user’s perspective. Only need a marker and a path.	Removed
1C	Overlay must permit zoom in/out feature	Completed
1D	Overlay must function for each of the floors with shelves	Removed

2	The program will be written in Javascript using the Google Maps API **and PHP**	In Progress
3A	For every call number range (shelf) in the Excel sheet, longitudes and latitudes of the shelf must be included in two separate columns	Completed
3B	Coordinates used as longitude and latitude are matched with the proper shelves as to be used to create markers on Google Maps	Completed
4	Excel sheet converted into a database that allows the clients to modify and update library information content (call number ranges, latitude and longitude incase they move the shelves)	Completed
5	Javascript program should use the database to find (query) the range of call numbers (shelf) based on an inputted call number	Completed
6A	When a user requests book info, the call number must be extracted from the button on the library site through a link	Completed
6B	The call number is then parsed through the database to check which call range number it belongs to	Completed
6C	The corresponding coordinates for that range (shelf) is outputted and sent to Google Maps to pinpoint location.	In Progress
7A	A marker must be placed at that location	Completed
7B	User should be able to navigate to reach destination.	In Progress
8A	Navigation will either consist of real-time movements of user to shelf or predetermined paths a user can take	In Progress
8B	For predetermined paths, user inputs current floor and destination floor It would be simpler to use the different library entrances	Removed
9A	Send all database information over to the library	Not started
9B	Send our Maps API over to library and makes sure it outputs a link to Google Maps	Not started

Issues

- Creating URL links for paths on different floors is not as straightforward as the markers
    - URL links has hexadecimal headers that change based on location and floor number; they are not constant
    - May be a dropped feature
- Query does not output only one result. Possibly due to the combination of punctuation, numbers, and letters.
- We may not have access to the official button on the library site
    - We will have to send the database and our Maps code over to the clients

Changes to Original Plan and Plan for Deliverable #4:

Instead of using the Google Maps API, we will directly use the actual Google Maps -- www.google.com/maps. Making markers that display on the correct floor plan is much simpler, whereas with the API, it requires us obtaining the different library floor plans and create overlays. Creating paths from library entrances are more useful to student and faculty/staff because it creates a frame of reference. Thus, we will work towards that instead of using user input, especially considering the fact that GPS and Wifi signals in the library are scattered. However, due to how creating paths on Google Maps is very unstable, there is a possibility it cannot be completed.

After that has been completed, we will provide all current code to the library staff and make sure that they are able to get the correct results after integrating our code into theirs.