DEPARTMENT OF COMPUTER SCIENCE
COLLEGE OF ARTS AND SCIENCES
CSCI 4961/4962 Capstone Deliverable #4

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.

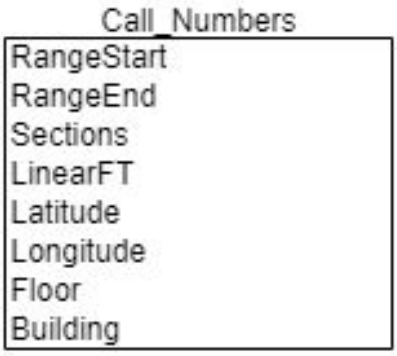
Basic Functionality

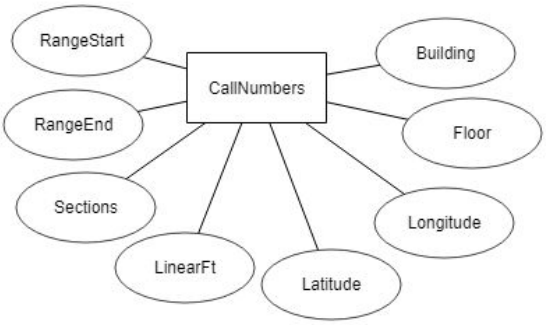
When an individual finds the book he or she would like to locate on the library website, there will be a button that prompts two options for users: Send a Text Message or Link to Google Maps. The "Send a Text Message" will allow the user to provide their phone number and get a text with the book's call 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 receives a call number from Library Catalog Button that is sent to DB to query. It receives coordinates from DB and output a marker on the shelf location of the call number. The map is 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 sends coordinates/floor location.
- Library Button
  - PHP
  - Holds the user's desired call number and link to our Google Maps API
  - The button sends the call number to the Maps API and receives a link to Google Maps. The link includes marker.
  - **Currently, the library has set up a separate button/link for testing purposes. It will be used to test the database queries.**

ER Diagram





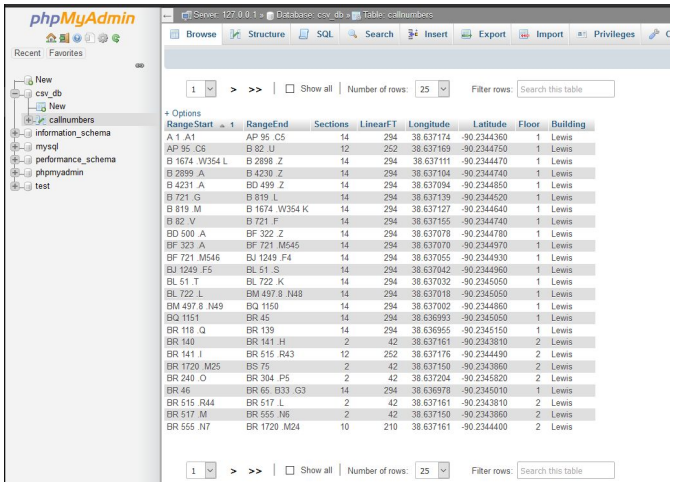
CallNumbers is the table, and each attribute is a column in the database. There are no foreign/primary keys. We only have one table containing all of the shelf information, so there are no relationships with other tables.

Google Maps Code

The code is within the Github library repository. The code mainly consists of PHP to use retrieved data from the database to create a link (index.php, which contains markerurl.php). The code also detects which device the user is using so the link will properly direct them to Google Maps or the link will take a mobile user to a separate HTML page that instructs them on changing the floor level manually (detectDeviceType.php, mobilePage.html)

Database

The .sql script to construct the database is also within the github, under the database folder.



Query

The current (newer) query goal is to split up a call number in PHP, and then update the database by creating columns that splits up the call number ranges as well. There will be a script containing the update that appropriately splits up the call number ranges in the database, and the query is what retrieves and returns data to the maps code.

Old query to locate coordinates

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

Current query to locate coordinates

```
$callNum = “HF5412.G45.7”
list ($first, $two, $three, $four, $five) = explode (“.”, $callNum)

$arrayFirst = preg_split(‘/(?<=[0-9])(?=[A-Z]+)i’,$first);
$arraySecond = preg_split(‘/(?<=[0-9])(?=[A-Z]+)i’,$second);
$arrayThird = preg_split(‘/(?<=[0-9])(?=[A-Z]+)i’,$third);
$arrayFourth = preg_split(‘/(?<=[0-9])(?=[A-Z]+)i’,$fourth);
$arrayFifth = preg_split(‘/(?<=[0-9])(?=[A-Z]+)i’,$fifth);


SET @valOne = "HF5412";
SET @valTwo = “G45";
SET @valThree = "7";

-- Adding one of the six columns to database
ALTER TABLE `call numbers` ADD SECOND_PART_START VARCHAR NOT NULL DEFAULT '0';

UPDATE `call numbers` SET SECOND_PART_START =SUBSTR(`start`, LOCATE('.',`start`, LOCATE('.',`start`)+1)+1)
FROM `call numbers`;

-- Splits range start using the delimiter
SELECT LENGTH(`start`)-LENGTH(REPLACE(`start`, '.', '')) FROM `call numbers`;

SELECT
   SUBSTR(`start`, 1,LOCATE('.',`start`)-1) as FIRST_PART_START,
SUBSTR(`start`, LOCATE('.',`start`)+1, LOCATE('.',`start`, LOCATE('.',`start`)+1)-LOCATE('.',`start`)-1) as
SECOND_PART_START,
SUBSTR(`start`, LOCATE('.',`start`, LOCATE('.',`start`)+1)+1) as THIRD_PART_START

FROM `call numbers`;

-- Splits range end using the delimiter
SELECT LENGTH(`end`)-LENGTH(REPLACE(`end`, '.', '')) FROM `call numbers`;

SELECT
   SUBSTR(`end`, 1,LOCATE('.',`end`)-1) as FIRST_PART_END,
SUBSTR(`end`, LOCATE('.',`end`)+1, LOCATE('.',`end`, LOCATE('.',`end`)+1)-LOCATE('.',`end`)-1) as
SECOND_PART_END,
 SUBSTR(`end`, LOCATE('.',`end`, LOCATE('.',`end`)+1)+1) as THIRD_PART_END

FROM `call numbers`;

-- Compares split call number with the split range start and range end values
SELECT `longit`, `lat` FROM `call numbers` WHERE (@valThree BETWEEN `start` AND `end`)
```

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	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**	Completed
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	In progress
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.	Removed
8A	Navigation will either consist of real-time movements of user to shelf or predetermined paths a user can take	Removed
8B	For predetermined paths, user inputs current floor and destination floor	Removed
9A	Send all database information over to the library	Completed
9B	Send our Maps API over to library and makes sure it	Completed

	outputs a link to Google Maps	

Requires 9 are completed because the basic setup of the call number database is completed and can easily be updated/changed with a script if necessary. A demo of the project is already sent and live, up and running on the library catalogue site at: libcat.slu.edu:2082 despite the query not being completed. This allows us to debug and test the code while it is actually on the library database, and not simply on our local database. This also eases the process of transferring files once the new, updated query and modified database is completed as the library already has a database set up for this project. They just need to import the new database (or run the script that modifies) and we will send a new index.php file that includes the new query.

Issues

- We cannot change the text on the buttons on the library website as we do not have access to that code.
- Current query is still work in progress. It will replace the old query that is currently in the JavaScript as a placeholder
    - Splitting the ranges based on the number of delimiters
        - Currently: For the following ranges, they are split in three parts even though some ranges only have two parts

start	end
A1.A1	AP95.C5
AP95.C6	B82.U
E840.8.K43	E999
HD1	HD30.23.B6
HD30.23.B7	HD 31 .B36
HF5410.M.45	HF5415.122.S4

FIRST_PART_START	SECOND_PART_START	THIRD_PART_START
A1		A1.A1
AP95		AP95.C6
E840	8	K43
		HD1
HD30	23	B7
HF5410	M	45

    - All six columns need to be added to the database using ALTER TABLE through the query
        - The code that adds the additional columns must be removed once they have been added **once** to the database
        - Data needs to be added for these new columns from row 1 → default value when columns are created is NOT NULL

start	end	section	feet	longit	lat	floor	building	variable	THIRD_PART_S
A1.A1	AP95.C5	14	15	30	-90	1	Lewis		*NULL*
AP95.C6	B82.U	12	5	30	-60	1	Lewis		*NULL*
E840.8.K43	E999	9	189	38	-90	2	Lewis		*NULL*
HD1	HD30.23.B6	5	554	32	-93	2	Lewis		*NULL*
HD30.23.B7	HD 31 .B36	43	345	34	-95	2	Lewis		*NULL*
HF5410.M.45	HF5415.122.S4	4	344	37	-99	2	Lewis		*NULL*
		0	0	0	0	0			4
		0	0	0	0	0			
		0	0	0	0	0			A1.A1
		0	0	0	0	0			AP95.C6
		0	0	0	0	0			K43
		0	0	0	0	0			HD1
		0	0	0	0	0			B7
		0	0	0	0	0			45

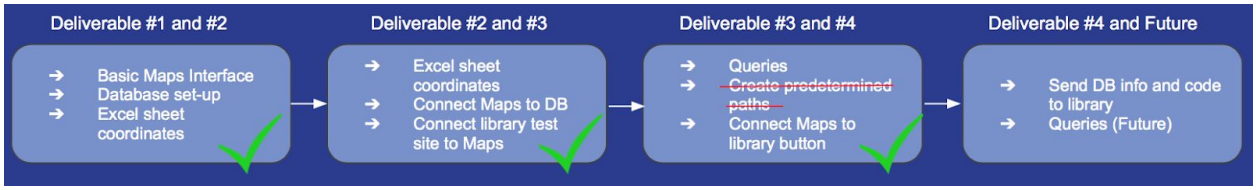
- For comparing call number to call number ranges, the splitted parts of the ranges will be used (hence the reason why they must be their own columns - it is easier to find coordinates for a row when the splitted range parts associate with the coordinates
  - Old comparison query:

```
SELECT `longit`, `lat` FROM `call numbers` WHERE (@valThree
BETWEEN `start` AND `end`)
```

  - New comparison query:

```
SELECT `longit`, `lat` FROM `call numbers` WHERE (@valOne
BETWEEN `FIRST_PART_START` AND `FIRST_PART_END`) AND (@valTwo
BETWEEN `SECOND_PART_START` AND `SECOND_PART_END`) AND
(@valThree BETWEEN `THIRD_PART_START` AND `THIRD_PART_END`)
LIMIT 0,1
```

Timeline:



Deliverable #4 and Future Plan After Capstone:

The completed tasks during Deliverable #4 were sending all database information to the library, ensuring that they are able to access it and use it as we are able to locally. We also completed the task of sending the Maps API to the library, verifying that a link to Google Maps is created and outputted.

The program works on the test library site due to the old query being used as a placeholder until the new query is completed. Any other changes to the database will also be exported and sent to be imported into the library database if necessary. This will continue after the deliverable #4 presentation.