**CSE 5335-002 Web Data Management**

**Project 2**

**Design Document**

Seeram Likitha

1001363714

[likitha.seeram@mavs.uta.edu](mailto:likitha.seeram@mavs.uta.edu)

Table of Contents

[1. Project Description 3](#_Toc465283917)

[2. Requirement Analysis 3](#_Toc465283918)

[2.1 Software Requirements 3](#_Toc465283919)

[2.2 Hardware Requirements 3](#_Toc465283920)

[3. Assumptions 4](#_Toc465283921)

[3.1 Google Map 4](#_Toc465283922)

[3.2 Information in the Overlay Marker 4](#_Toc465283923)

[3.3 Text Display Area 4](#_Toc465283924)

[4. Implementation 5](#_Toc465283925)

[4.1 HTML File 5](#_Toc465283926)

[4.2 CSS File 5](#_Toc465283927)

[4.3 Javascript 5](#_Toc465283928)

[4.3.1 Initializing the Map 5](#_Toc465283929)

[4.3.2 Reverse Geocoding 6](#_Toc465283930)

[4.3.3 Asynchronous Calls to GeoNames API 6](#_Toc465283931)

[4.3.4 Click History 6](#_Toc465283932)

[4.3.5 Clear Button 7](#_Toc465283933)

[4.4 Running the application 7](#_Toc465283934)

[5. Conclusion 8](#_Toc465283935)

# Project Description

This project is about extracting Postal Address and Weather details of a place clicked on google maps. This is done by using two web services – **google maps API** and **geoNames API**.

Using the Reverse Geocoding process of google maps API, postal address of a place is fetched from the latitude and longitude values. And by using the same coordinate values, geoNames API is asynchronously called to fetch the weather details of the marked place.

The web page also stores the history of user actions (User clicks on google maps) in the history section. It also includes a ‘CLEAR’ button to clear the history section and set the map to the initial display.

# Requirement Analysis

This project works by using two web services. They are:

1. Google Maps API (Reverse Geocoding)
2. GeoNames API (Find Near By Weather)

We need to register for these two services and get the access enabled to use the services and get the output.

This project requires us to use the following technologies while implementing:

1. HTML5
2. JavaScript
3. AJAX
4. CSS

## 2.1 Software Requirements

Following software are required for the project:

* XAMP Control Panel (for windows users)
* Sublime Text Editor
* A Web browser

## 2.2 Hardware Requirements

A minimum of following hardware are necessary for this project:

* An Operating System to support the above software
* 2GB RAM

# Assumptions

A few assumptions are made while working on this project. They are listed below.

## 3.1 Google Map

As per the requirement google map is initialized as follows:

* Size: 600\*500 pixels
* Initial Marker position: 32.75, -97.13
* Zoom level: 17

## 3.2 Information in the Overlay Marker

The marked position on the google map fetches the information from both the web services. The data retrieved is appended and displayed in a single overlay marker.

Example:

Postal Address: Central Arlington, Arlington, TX, USA  
Temperature: 23.3, WindSpeed: 05, Clouds: broken clouds

The same appended information is displayed on the text area section too.

## 3.3 Text Display Area

The text display area is implemented as a <div> element which includes each marker information as a <p> element. This way of displaying the history makes the web page view more clear.

Therefore, the text display area can be viewed as (In HTML format) –

<div>

<p> information </p>

<p> information </p>

<p> information </p>

-----

</div>

# Implementation

This section includes the implementation details of HTML, javascript and CSS files.

## 4.1 HTML File

It includes 3 sections.

* Section 1: This section includes a **clear** button. The “**onclick**” attribute of this element calls a ‘**clearHistory**()’ method which clears the text area section and then sets the map to its initial location.
* Section 2: In this section, a google map is included with a marker pointing to the center. This map has a **click listener** enabled which changes the marker to the clicked location on the map.
* Section 3: This section displays the previous **history** of marker positions (a maximum of 10). The Information includes postal address and weather conditions of the place fetched by using the latitude and longitude values.

## 4.2 CSS File

A CSS file is linked to the HTML. This file includes different types of styles assigned to the selectors.

## 4.3 Javascript

The javascript file includes the following functionality:

### 4.3.1 Initializing the Map

The map initializing method “**initMap**()” initializes the map and marker. This method also includes a click listener on the map which sets the map’s marker to a new position and calls the “**reverseGeocoding**()” method to fetch the address of that position.

map.addListener('click', function(event) {

var latitude = event.latLng.lat();

var longitude = event.latLng.lng();

marker.setPosition(event.latLng);

reverseGeoCoding(geocoder, map, infowindow, latitude, longitude, marker);

});

### 4.3.2 Reverse Geocoding

Reverse Geocoding is a process of fetching address of a location using latitude and longitude values.

In this method, marker is set to the new position and map is zoomed out to level 11 as shown below:

map.setZoom(11);

marker.setPosition(loc);

In this method ‘**InfoWindow’**, overlay marker displaying information above the marker, is set to the position address returned by geocode method.

If the selected position does not return any address, then an alert has been setup to notify – ‘No results found’.

### 4.3.3 Asynchronous Calls to GeoNames API

An **AJAX** call is made to the web service ‘**findNearByWeather’** provided by geoNames API. This service supports the response in JSON and XML formats. Here we are working on the **XML** format of the **response**.

When the **readyState** is 4 i.e., when the response is ready, ‘**displayResult**()’ method is called. This method fetches the weather details from the response xml and appends it to the InfoWindow.

### Click History

History section is aligned to the right of the web page. This section displays the history of previous user clicks. It includes both the postal address and weather conditions of that location. Latest marker position appears at the last of the list.

Here an **Additional Functionality** has been implemented. To make the view in a more appreciable format, a **limit on the history log** has been imposed. This section displays a maximum of previous 10 user clicks. Each time the list gets to the size of more than 10, first element of the list is erased and the new latest marker information is appended to the bottom of the list.

Code snippet below:

var history = document.getElementById("history"); //Selecting the text area section

var oldHistory = history.innerHTML;

history.innerHTML = oldHistory + "<p>"+infowindow.getContent()+"</p>"; //appending the new information to the existing list

history.removeChild(history.children[0]); //Removing the first element of the list when the size reaches 10

### 4.3.5 Clear Button

The Clear button’s attribute “**onclick**” calls ‘**clearHistory**()’ method which overrides history information to a blank element. It also calls the initMap() method so that the map is set to its initial location.

## 4.4 Running the application

Make sure to include your project folder in the hotdocs folder in xampp. Apache and MySQL modules should be running in XAMPP control panel.

You should make sure your internet connection is enabled while working on this project as we retrieve data from web services

You need to hit the link **localhost/project2/webmash.html** to run the application in a browser.

# Conclusion

Project 2 has helped me learn concepts of web designing like:

* HTML5
* Advanced CSS
* AJAX calls in javascript
* Working with Web Services

This project has made me put in more efforts to analyze and work more on the coding part rather than designing.