Location-Based Services Using HTML5 Geolocation and Google Maps APIs

Wen-Chen Hu
Department of Computer Science
University of North Dakota
Grand Forks, ND 58202
wenchen@cs.und.edu

Naima Kaabouch
Department of Electrical Engineering
University of North Dakota
Grand Forks, ND 58202
naima.kaabouch@engr.und.edu

Hung-Jen Yang
Department of Industrial Technology Education
National Kaohsiung Normal University
Kaohsiung City, Taiwan
hjyang@nknucc.nknu.edu.tw

Xiwei Wang
Department of Computer Science
University of North Dakota
Grand Forks, ND 58202
xiwei.wang2012@gmail.com

Abstract

A location-based service (LBS) is a service based on the geographical positions of mobile handheld devices such as smartphones or tablet computers. Though location-based services are popular and useful, most developers are not familiar with its development because it used to be complicated and difficult. However, the introduction of HTML5 Geolocation and various map APIs (Application Programming Interfaces) has made the LBS construction easy. Geolocation, a feature of the HTML5 specification and API, is used to pinpoint the user's location without needing the GPS support directly. Additionally, one of the popular map APIs is Google Maps APIs, which provide a powerful and convenient tool for map applications. This article introduces the HTML5 Geolocation and Google Maps APIs by constructing a simple LBS. LBS developers quickly learn how to build an LBS application by studying this article.

1 Introduction

Mobile computing research evolves constantly and swiftly. New mobile devices, technologies, methods, or applications are introduced every day. One of the mobile applications, location-based service (LBS), has attracted great attention recently. A location-based service is a service based on the geographical positions of mobile handheld devices such as smartphones and tablet computers. Two of the LBS examples are finding a nearby ethnic restaurant and comparing prices of a product from different stores. The future of the LBS is bright according to the following market research:

- Gartner (2012, October 11) expected almost 800 million location-based service users worldwide by the end of the year 2012 and revenue generated by consumer location-based services is forecast to reach $13.5 billion in 2015.
- Owing to different factors such as increasing smartphone adoption, continued growth of mobile advertising, and the wider coverage and higher speeds of mobile networks, the location-based services revenue is expected to reach $10.3 billion in 2015 up from $2.8 billion in 2010 according to Pyramid Research (2012, May).
- Federal Communications Commission (2012, May 25) reported that as of May 2011, 28 percent of adult Americans used mobile LBS and LBS are expected to deliver $700 billion value to consumers and business users over the next decade.

Though location-based services are popular, most developers are not familiar with their development because it used to be complicated and difficult. However, since the introduction of HTML5 Geolocation and Google Maps APIs, LBS development becomes a kind of Web development and many IT developers are already familiar with Web development. This research builds a simple LBS application by using HTML5 Geolocation and Google Maps APIs. Detailed design and implementation are described in this paper. IT workers could quickly join the LBS development after studying this article.

The rest of this paper is organized as follows. Section 2 gives a generic LBS architecture and an LBS example showing how LBS work. The proposed LBS application is given in Section 3, which includes two sub-sections: the system structure and the simple geographical database of the proposed system. Section 4 explains how to retrieve and update locations by using HTML5 Geolocation and PHP (Hypertext Preprocessor). The query processing is described in Section 5 including code of applying Google Maps APIs. The final section summarizes this study and gives possible LBS projects.

2 Location-Based Services (LBS)

This section introduces location-based services including a generic LBS architecture and an LBS example.

2.1 An LBS Architecture

A location-based service is a service based on the geographical position of a mobile handheld device (Kupper, 2005; Kolodziej & Hjelm, 2006). Two of the LBS examples are (i) finding a nearby ethnic restaurants and (ii) locating a nearby store with the best price of a product. A system structure of location-based services, shown in Figure 1, includes five major components (Steiniger, Neun, & Edwardes, 2006):

(a) Mobile handheld devices, which are small computers that can be held in one hand. For most cases, they are smartphones.
(b) Positioning system, which is a navigation satellite system that provides location and time information to anyone with a receiver.
(c) Mobile and wireless networks, which relay the query and location information from devices to service providers and send the results from the providers to devices.
(d) Service providers, which provide the location-based services.
(e) Geographical data providers, which are databases storing a huge amount of geographical data such as information about restaurants and gas stations.

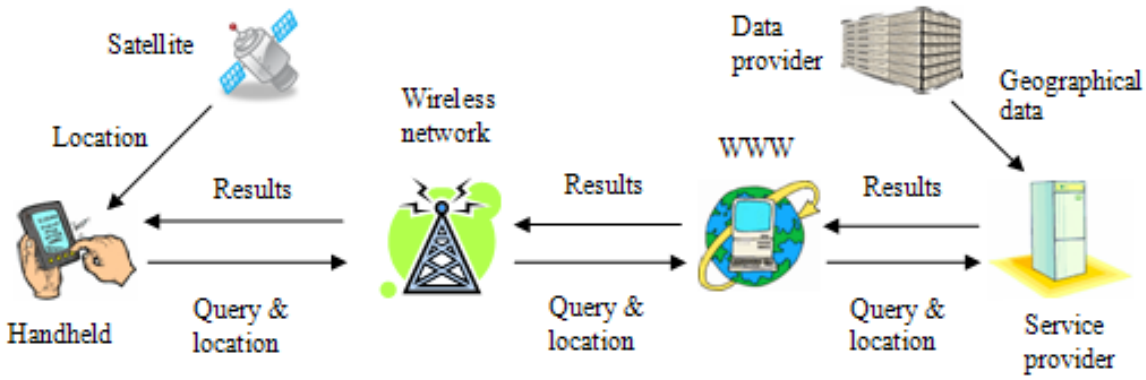


Figure 1: A System Structure of Generic Location-Based Services.

2.2 An LBS Example

The previous descriptions gave a generic LBS system structure. To help readers understand LBS, an example of our proposed system is given step by step as follows:

1. The tablet computer (a) includes our application of finding classmates on the go.
2. The user submits a query of finding his/her classmates of CSci457 along with the location information from a positioning system (b), HTML5 Geolocation in this case, from the application program.
3. The application program calls the server-side programs (d) located at the Aerospace School of the University of North Dakota along with the location information via mobile or wireless networks (c).

4. The programs at the servers perform the searches using a MySQL database (e) which stores geographical data.
5. The results such as a map with markers or directions are sent back to the user.

A nice introduction of LBS technologies and standards is given by Wang, Min, & Yi (2008, May 19-23).

3 The Proposed Location-Based Service

The main purpose of this article is to explain how to implement an LBS application by using HTML5 Geolocation and Google Maps APIs, so the proposed LBS application is made simple on purpose. It is to find classmates, who are constantly moving. For example, an assignment is due tomorrow and a student is urgent to find a classmate to help solving his/her assignment problems. The proposed LBS application keeps track of the whereabouts of the classmates, who have signed up the application, by using the methods `watchPosition` and `clearWatch` of HTML5 Geolocation. Once the classmates' locations are located, the direction between the student and the selected classmate is then given by using Google Maps APIs.

3.1 The System Structure

Figure 2 shows the system structure of the proposed system, which can be shared by many users. In order to use this system, users have to sign up first by entering their information such as names and classes. Once the application is running on a user's browser, he/she can start querying the whereabouts of his/her classmates. At the same time, his/her location information will be updated from time to time in the server.

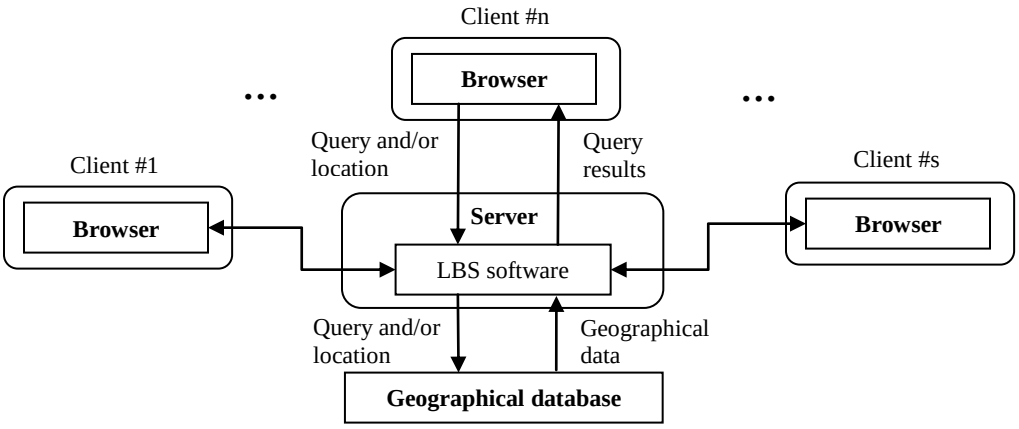


Figure 2: System Structure of the Proposed System.

Figure 3 shows the workflows of the proposed method. The users/clients communicate with the server via two paths:

- If the user does not submit a query, his/her location information is sent to the server automatically from time to time as long as the application is active.
- If the user submits a query along with his/her location information, the query results such as walking directions will be sent back from the server.

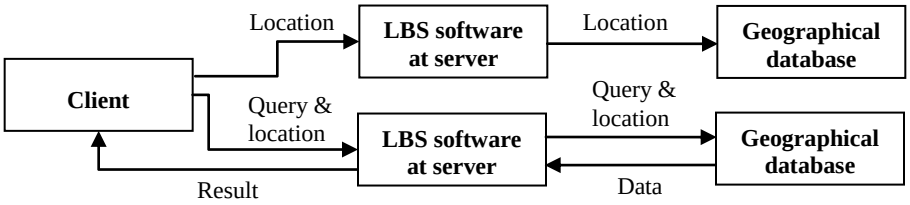


Figure 3: Workflows of the Proposed Method.

3.2 The Geographical Database

Geographical databases are usually provided by a third party such as GeoNames (n.d.). This project creates its own simple geographical database, GeoDB, as shown in Figure 4, which includes three tables: (a) `Users` table, (b) `Courses` table, and (c) `Enrollments` table. The attribute `time` of the `Users` table is used to validate the location, e.g., the location is valid only if it was updated within one hour.

PID	Name	Location and Time		
		Latitude	Longitude	Time
1	Poke Mon	47.9252569	-97.032855	20:59:11, 03/17/2013
…	…	…	…	…
n	Digi Mon	48.2956123	-96.903403	12:34:29, 03/10/2013

(a)

CID	Name	Credit Hours
CSci260	Web Programming	3
…	…	…
CSci515	Data Engineering	3

(b)

PID	CID	Semester
1	CSci457	13S
…	…	…
n	CSci370	12F

(c)

Figure 4: Geographical Database, GeoDB, Used in This Project Including (a) `Users` Table, (b) `Courses` Table, and (c) `Enrollments` Table.

The `Users` table is created by using the following SQL (Structured Query Language) commands (Oracle, n.d.):

```
mysql> CREATE TABLE  Users (
    >   PID        int NOT NULL AUTO_INCREMENT,
    >   name       VARCHAR(64),
    >   latitude   DOUBLE,
    >   longitude  DOUBLE,
    >   time       DATETIME,
    >   PRIMARY KEY (PID) );
```

Other two tables can be created by similar commands. Users have to sign up for the system and enter their information such as names and classes taking before using the application. The attributes of `latitude`, `longitude`, and `time` of the `Users` table will be updated automatically from time to time while the application is active. The following SQL command is used to enter the user data when signing up:

```
mysql> INSERT INTO  Users ( name, latitude, longitude, time )
    >   VALUES ( 'Poke Mon', 47.9252569, -97.032855, NOW() );
```

The HTML5 Geolocation `watchPosition` method invokes the update command whenever user position changes. To update the location information, the following SQL command is embedded in a PHP script:

```
mysql> UPDATE  Users  SET  latitude=$lat, longitude=$long,
    >   time=NOW( )  WHERE  PID=$id;
```

where `$lat` and `$long` are found by the HTML5 Geolocation method `watchPosition` and `$id` is the user id sent with the latitude and longitude.

4 Location Information Updating

There are a couple of versions of HTML (HyperText Markup Language) such as (i) HTML, each version of HTML adding more tags, (ii) HTML 4.0, trying to clean up the standard by marking some of the tags as deprecated, and (iii) XHTML (eXtensible HTML), a reformulation of HTML 4.0 in XML. Though HTML was well established and defined for most of the requirements of static Web pages, there were several areas that were missing from the previous HTML specifications. Without those missing specifications, each browser or software had its unique way of specifying certain objects. HTML5 is therefore proposed to improve and standardize the HTML specifications. Geolocation is a feature of the HTML5 specification and API. It is used to find users' locations, though the location accuracy is not guaranteed.

4.1 HTML5 Geolocation APIs

HTML5 Geolocation (W3C, 2012, May 10) is dense and powerful. Table 1 shows the five interfaces provided by the HTML5 Geolocation.

Interfaces of HTML5 Geolocation	
Interface	**Description**
`Geolocation`	The `Geolocation` is the main object in the API and is used to obtain the location information of devices and users.
`PositionOptions`	The `getCurrentPosition` and `watchPosition` methods accept `PositionOptions` objects as their third argument.
`Position`	Same as `Coordinates`
`Coordinates`	The coordinates of a geographical location
`PositionError`	The error information if the geographical information cannot be obtained

Table 1: HTML5 Geolocation Interfaces.

`Geolocation` is the most important interface of the HTML5 Geolocation. Table 2 shows the three methods, `clearWatch`, `getCurrentPosition`, and `watchPosition`, provided by the `Geolocation` interface.

Methods of `Geolocation` Interface	
Method	**Description**
`clearWatch`	Stop the watch process identified by the `watchID` argument.
`getCurrentPosition`	Get the current location in latitude and longitude coordinates.
`watchPosition`	Continue to monitor the position and invoke callback function when position changes.

Table 2: Methods of HTML5 `Geolocation` Interface.

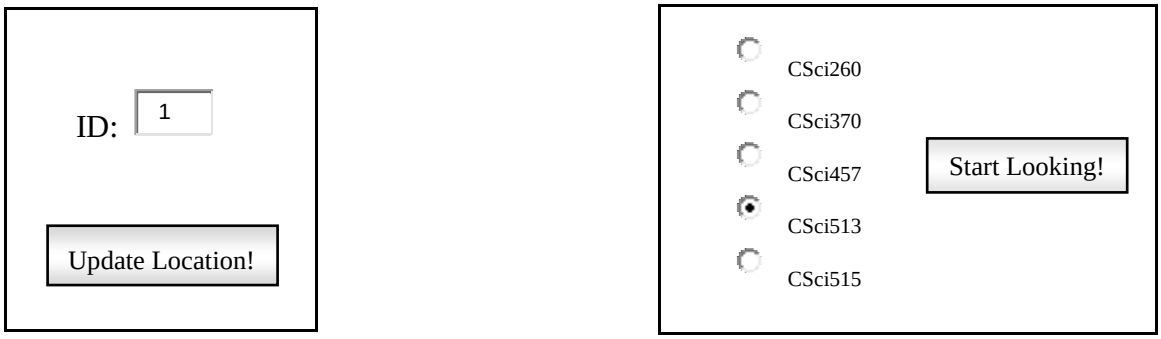
The `getCurrentPosition` or `watchPosition` method takes one, two or three arguments. When called, it must immediately return and then asynchronously attempt to obtain the current location of the device. If the attempt is successful, the callback function must be invoked with a new `Position` object, reflecting the current location of the device. Table 3 gives the attributes of the `Postion` interface.

`Position` Interface			
Attribute	**Value**	**Unit**	**Description**
`coords.latitude`	double	degrees	The latitude of postion
`coords.longitude`	double	degrees	The longitude of position
`coords.accuracy`	double or null	meters	The accuracy of position
`coords.altitude`	double or null	meters	The altitude above the mean sea level
`coords.altitudeAccuracy`	double or null	meters	The altitude accuracy of position
`coords.heading`	double or null	degrees clockwise	The heading from the true north
`coords.speed`	double or null	meters/second	The speed of the device
`Timestamp`	DOMTimeStamp	like the Date object	The date/time of the response

Table 3: Attributes of the `Position` Interface.

4.2 Updating Locations and Times

Figure 5 shows two of the system interfaces. Once the user clicks the button "Update Location!" in Figure 5.a, the application becomes active and the location information will be sent to the server whenever the location is changed. At the same time, the interface in Figure 5.b will appear and allow user to start looking for his/her classmates on the go.



(a) (b)

Figure 5: System Interfaces Including: (a) Sending Location and (b) Looking for Classmates.

The `getCurrentPosition` or `watchPosition` method is used to get the user's position. The difference between the two methods is the former only retrieves the position once and the latter keeps monitoring and sending positions until the method `clearWatch` is issued. Program 1 is a simple Geolocation example returning the latitude and longitude of the user's position by using HTML5 Geolocation and JavaScript (Mozilla Developer Network, n.d):

1. Call the JavaScript function `getLocation` by the event `onClick`.
2. Check if `window.navigator.geolocation` is supported.
3. If it is supported, run the `watchPosition` method.
4. If it is not supported, display a message to the user at the location found by `document.getElementById.innerHTML`.
5. If the `watchPosition` method is successful, it calls a callback function specified in its argument (`sendLocation` in this example) with a `position` object, which stores the returned location information.
6. The asynchronous callback function `sendLocation` retrieves the latitude and longitude from the attribute `coordinates` of the `position` object and calls the PHP program `updateLocation.php` to update the location information in the geographical database, GeoDB, at the server.

```
<!DOCTYPE html>
<html>
 <body>
      ID: <input type="text" size="4" id="id" value="1" />  
  <button onClick="getLocation( )">Update Location!</button>
  <p><div id="myDiv"></div></p>

  <script>
   var  x  = document.getElementById( "myDiv" );
   var  id = document.getElementById( "id" ).value;

   function  getLocation( ) {
    if ( navigator.geolocation )
     navigator.geolocation.watchPosition( sendLocation );
    else
     x.innerHTML = "Geolocation not supported!";
   }

   function  sendLocation( position ) {
    var  url = "updateLocation.php?id=" + id +
               "&lat="  + position.coords.latitude +
               "&long=" + position.coords.longitude;
    window.open( url, "_self" );           // Calling updateLocation.php
   }
  </script>
 </body>
</html>
```

Program 1: Watching for Current Location and Calling a Server Program to Update It.

The Program 2, a PHP (Hypertext Preprocessor Group, n.d.) program embedded with SQL commands, is called from Program 1. It is used to update the location and time information where the location along with the ID are sent by the GET method from the JavaScript in Program 1.

```
<?php

 $id       = $_GET[id];            // From URL
 $lat      = $_GET[lat];           // From URL
 $long     = $_GET[long];          // From URL
 $username = "username";
 $password = "passwd";
 $database = "database";
 $host     = "mysql.eng.university.edu";
 $conn     = mysql_connect( $host, $username, $password );

 if ( !$conn ) {
  die( 'Could not connect: ' . mysql_error( ) );
 }
 else {
  mysql_select_db( $database, $conn );
  $sql  = "UPDATE Users SET latitude=$lat, longitude=$long, time=now( )";
  $sql .= " WHERE PID=$id";
  if ( !mysql_query( $sql, $conn ) ) {
   echo "Error updating alocation: " . mysql_error( );
   die( "Unable to update a location" );
  }
  mysql_close( $conn );
  echo  "Update done!";
}

?>
```

Program 2: Updating User's Latitude, Longitude, and Time.

5 Location-Based Query Processing

The focus of this research is to show the technologies of HTML5 Geolocation and Google Maps APIs. Therefore, the proposed LBS is made simple on purpose. It is to find a classmate closest to the user. However, it is different from traditional LBS, such as finding a closest gas station, because the targets are moving constantly.

5.1 Marking the Client's Location on a Map

Many times the location-based services need to mark the users' locations on a map. Program 3 (W3School, n.d.) displays an interactive map with a marker and the options of zoom and drag. The map is sent back to the client browser from the server by using the PHP script, which generates the HTML page for the map by the command `echo`. Not only HTML5 Geolocation, this program also uses a couple of Google Maps APIs:

- The URL "`http://maps.google.com/maps/api/js?sensor=false`" stores the JavaScript file that loads all of the symbols and definitions for using the Google Maps APIs.
- The `google.maps.Map` constructor creates a new map inside a specified HTML element and the `google.maps.Marker` constructor creates a marker.

```
<?php

echo '<!DOCTYPE HTML>
<html>
 <body>
  <div id="mapCanvas"></div>

  <script src="http://maps.google.com/maps/api/js?sensor=false"></script>
  <script>
   var  mapCanvas = document.getElementById( "mapCanvas" );

   if ( navigator.geolocation )
    navigator.geolocation.getCurrentPosition( showPosition );
   else
    mapCanvas.innerHTML = "Geolocation not supported!";

   function  showPosition( position ) {
    latitude  = position.coords.latitude;
    longitude = position.coords.longitude;
    loc       = new google.maps.LatLng( latitude, longitude );
    mapCanvas.style.height = "250px";
    mapCanvas.style.width  = "500px";

    var  mapOptions = {
     center: loc,
     zoom: 12,
     mapTypeId: google.maps.MapTypeId.ROADMAP,
     mapTypeControl: false,
     navigationControlOptions: {
      style: google.maps.NavigationControlStyle.SMALL
     }
    };

    var  markMap = new google.maps.Map( document.getElementById (
     "mapCanvas" ), mapOptions );
    new google.maps.Marker ( {
     position: loc, map: markMap, title: "Hello!" } );
   }
  </script>
 </body>
</html>';

?>
```

Program 3: Marking the User's Location on a Map.

Figure 6 shows an exemplar map generated from Program 3. The map is centered at the location found by HTML5 Geolocation method `getCurrentPosition` and the location is marked by using the Google Maps API `google.maps.Marker`.

Figure 6: An Example of Program 3 Execution Results.

5.2 Drawing a Direction on a Map

Once the LBS server receives the query and location information from a user, it finds the best matches from the geographical database, GeoDB. For the proposed LBS, it finds the closest classmate with a valid location, whose time is within one hour, for example. The server then draws a direction from the user to the classmate found. Program 4 (Google, n.d.) is used by the server to send a map with a direction to the client by using the PHP command `echo`. It uses a couple of Google Maps APIs to draw a direction on a map:

- The class `google.maps.DirectionsService` computes directions between two or more places.
- The class `google.maps.DirectionsRenderer` renders directions retrieved in the form of a `DirectionsResult` object retrieved from the `DirectionsService`.
- The method `google.maps.DirectionsService.route` initiates a directions request to the `DirectionsService`. It requires passing a callback function which executes upon completion of the service request.

```
<?php

echo  '<!DOCTYPE HTML>
<html>
 <body onLoad="initialize( )">
  <div id="map_canvas" style="width:96%;height:400px"></div>

  <script src="https://maps.googleapis.com/maps/api/js?v=3.exp&sensor=false">
  </script>
  <script>
   var  directionsDisplay;
   var  directionsService = new google.maps.DirectionsService( );
   var  map;

   function  initialize( ) {
    directionsDisplay = new google.maps.DirectionsRenderer( );
    var  GF = new google.maps.LatLng( 47.9252569, -97.032855 );   // Changeable
```

```
    var  mapOptions = {
     zoom: 12,
     mapTypeId: google.maps.MapTypeId.ROADMAP,
     center: GF
    }
    map = new google.maps.Map(
     document.getElementById('map_canvas'), mapOptions );
    directionsDisplay.setMap( map );
    calcRoute( );
   }

   function calcRoute( ) {
    var  request = {
     origin: "the ralph, grand forks, nd",                          // Changeable
     destination: "streibel hall, grand forks, nd",                 // Changeable
     travelMode: google.maps.DirectionsTravelMode.WALKING
    };
    directionsService.route( request, function( response, status ) {
     if ( status == google.maps.DirectionsStatus.OK )
      directionsDisplay.setDirections( response );
    } );
   }
  </script>
 </body>
</html>';

?>
```

Program 4: Drawing a Direction on a Map.

Figure 7 shows an example of the Program 4 execution results. The origin and destination are "the ralph, grand forks, nd" and "streibel hall, grand forks, nd," respectively in the Program 4. Instead of text locations, the locations could use latitude and longitude, too. Another map can be easily generated by replacing the origin and destination.

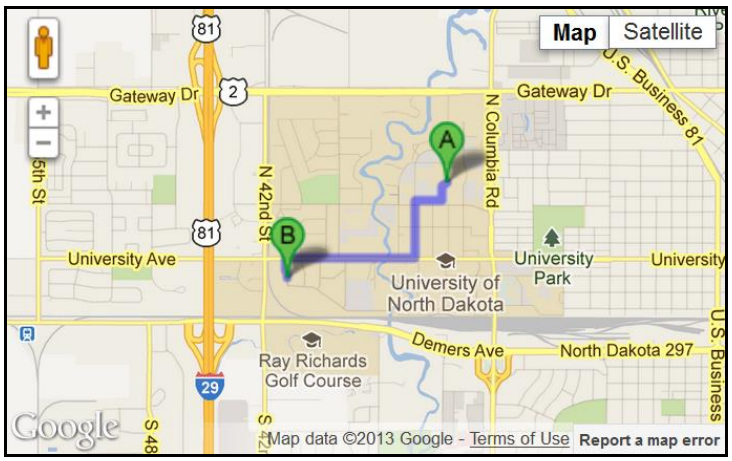


Figure 7: An Example of Program 4 Execution Results.

6 Summary

Nowadays there are more smartphones and tablet computers than PCs and servers. The omnipresence of smartphones and tablet computers makes location-based services like Foursquare (n.d.) extremely popular. Many mobile users depend on LBS such as navigation and recommendations to live their daily lives. Previously the LBS construction had to rely on complicated and difficult methods and procedures; e.g., the LBS developers had to be familiar with the details of various map utilities, like Google Maps and Bing Maps, and platforms, like Android and iOS. To make things worse, each map or platform has its own unique features and is not compatible with others. On the other hand, the platform of HTML5 Geolocation and Google Maps APIs is the browser, like Internet Explorer or Firefox. The developers' jobs become much easier because most developers are familiar with browser environment. Additionally, HTML5 Geolocation allows users to retrieve location information without needing the direct GPS support and Google Maps APIs can be easily applied without knowing the details. With them, LBS developers can quickly build an LBS application without needing to handle many details such as how to render maps and retrieve locations. This article proposes a simple LBS application and shows how it can be built by using HTML5 Geolocation and Google Maps APIs and hope more IT workers can join the LBS development after studying it.

The proposed LBS is made simple on purpose because this article focuses on the methods and technologies used instead of the LBS itself. Readers may apply the knowledge and technologies learned from this article to the following suggested applications:

- Find the interesting nearby places such as ethnic restaurants and movie theaters.
- Compare prices and perform product research.
- Advertisements and coupons are displayed based on users' locations.
- Use social networks to connect people within a short distance.

The following location-based research can also be considered:

- Detect any route anomalies. For example, an alert is generated if a pupil does not follow his/her regular route to school.
- Find travel recommendations based on route trajectories. For example, most people probably never heard the world's largest truck stop at Walcott, Iowa. With this feature, the drivers on the highway I-80 will be notified this interesting place when they are near Walcott.
- Indoor positioning and navigation are used to help users have a better visiting experience.
- Pre-fetch and cache map tiles based on drivers' current locations so maps can be displayed in time and power consumption is reduced.

Acknowledgments

This work is partially supported by the NSF (National Science Foundation) Grant #EPS-081442. The authors thank Dr. Martin Allen and reviewers for their valuable comments and great help.

References

Federal Communications Commission. (2012, May 25). *Location-Based Services Report*. Retrieved January 23, 2013, from http://hraunfoss.fcc.gov/edocs_public/attachmatch/DOC-314283A1.pdf

Foursquare. (n.d.). *About Foursquare*. Retrieved February 25, 2013, from http://foursquare.com/about/

Gartner. (2012, October 11). *Gartner Highlights Top Consumer Mobile Applications and Services for Digital Marketing Leaders*. Retrieved February 18, 2013, from http://www.gartner.com/newsroom/id/2194115

GeoNames. (n.d.). *About GeoNames*. Retrieved February 27, 2013, from http://www.geonames.org/about.html

Google. (n.d.). *Google Maps API*. Retrieved January 21, 2013, from https://developers.google.com/maps/

Kupper, A. (2005). *Location-Based Services: Fundamentals and Operation*. Wiley.

Mozilla Developer Network (MDN). (n.d.). *JavaScript*. Retrieved December 02, 2012, from https://developer.mozilla.org/en-US/docs/JavaScript

Oracle. (n.d.). *MySQL Document: MySQL Reference Manual*. Retrieved January 15, 2013, from http://dev.mysql.com/doc/

PHP Group. (n.d.). *PHP: Hypertext Preprocessor*. Retrieved December 18, 2012, from http://www.php.net/

Pyramid Research. (2011, May). *Location-Based Services: Market Forecast, 2011-2015*. Retrieved March 02, 2013, from http://www.pyramidresearch.com/store/Report-Location-Based-Services.htm

Steiniger, S., Neun, M., & Edwardes, A. (2006). Foundations of Location-Based Services. Retrieved December 13, 2012, from http://www.spatial.cs.umn.edu/Courses/Fall11/8715/papers/IM7_steiniger.pdf

W3Schools. (n.d.). *HTML5 Geolocation*. Retrieved January 25, 2013, from http://www.w3schools.com/html/html5_geolocation.asp

Wang, S., Min, J., & Yi, B. K. (2008, May 19-23). Location based services for mobiles: technologies and standards. In *Proceedings of the IEEE International Conference on Communication (ICC)*, Beijing, China.

World Wide Web Consortium (W3C). (2012, May 10). *Geolocation API Specification*. Retrieved February 6, 2013, from http://www.w3.org/TR/geolocation-API/