**Introduction**

**Inside Chapter:**

**1.1 Geolocation**

**1.2 System Objectives**

**1.3 System Flow Diagram**

**1.4 User Requirements**

**1.5 System Requirements**

**1.1 Geolocation**

Geolocation refers to the identification of the geographic location of a user or computing device via a variety of data collection mechanisms. Typically, most geolocation services use network routing addresses or internal GPS devices to determine this location. Geolocation is a device-specific API. This means that browsers or devices must support geolocation in order to use it through web applications.

Geocoding - finding the geographical location of a given address - is one of the most popular features of the Google Maps API. Both the JavaScript Maps APIs and the Maps APIs for Python, Flash, … etc include classes that enable applications to perform geocoding, and there is also a RESTful web service that offers the option of making geocoding requests from server-side applications with output in both XML and JSON.

The Google Maps JavaScript API v3 introduced a new format for geocoding responses that offers a number of improvements over the format used in the v2 API:

* A flatter response format for address components that is easier to parse
* The ability to tag an address component with multiple types
* Both full names and abbreviations for countries and state
* Differentiation between rooftop and interpolated geocoder results
* Both the bounding box and recommended viewport for each result

The Geocoding Web Service is intended to enable precaching of geocoder results that you know your application will need in the future. For example, if your application displays property listings, you can geocode the address of each property, cache the results on your server, and serve these locations to your API application. This ensures that your application does not need to geocode the address of a property every time it is viewed by a user. However, we do ask that you regularly refresh your cache of geocoder results. 2,500 requests may be sent to the Geocoding Web Service per day from a single IP address. This is independent of any geocoding activity generated by applications using one of the client Maps APIs for geocoding. Maps API Premier quotas remain unchanged.

In conjunction with the launch of the new Geocoding Web Service we are also announcing the deprecation of the current service, now [retroactively named](http://en.wikipedia.org/wiki/Retronym) the "Geocoding V2 Web Service". Existing applications using the V2 Web Service need not worry though. Deprecation indicates that we no longer intend to pursue any further feature development, but we will continue to maintain and support the service in accordance with the deprecation policy set out in the Maps API Terms of Service.

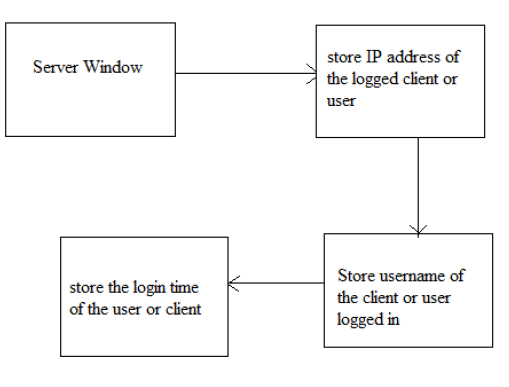
We hope that you find the new Geocoding Web Service easier to use and useful. As always we encourage you to check out the [Google Maps API Google Group](https://groups.google.com/group/Google-Maps-API) if you have any questions or comments relating to the APIs. We look forward to adding more great features to the Geocoding Web Service in future.

**1.2 System Objectives**

An online platform that enables localization and social tools between super user (companies) and traditional users like find location, get nearest location, and allow client to send private messages to another client or send multicast chatting messages.

Our system works on the web service, through which you can know the location of the user on the map and also find out the nearest and cheapest place containing a product if it is a drug or other products. It also allows all companies to register their products and communicate with customers.

It connects different client to each other and also client to main server. So, we have used the same concept here we are connecting two client or client and server with each other and providing the IP address we can talk with each other.

****

**1.3 System Flow Diagram**

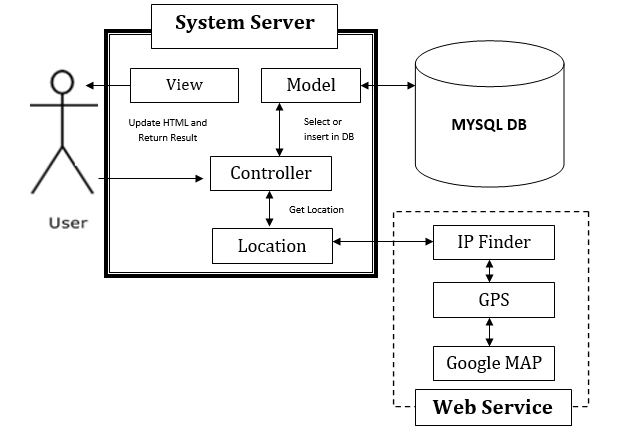


Figure 1.1 – System Flow Diagram

**System Component**:

There are three main components to any localization system:

* **Back-end Server** is based on a combination of system architecture and performance. We use MVC (Model, View and Controller) design pattern.
* **Storage space (Database)** is based on the number and capacity of the hard drives or databases built into the system. It is used to store user information such as location, personal information and product information.
* **Web Service** is an important part of the system design. We need for IP location finder API to get IP address and track routes of this IP to obtain location. Also, we need for google MAP API to create get exact location on a MAP and get the matrix distance API between all companies registered on the system that contain the user request and the actual user's location.

**1.4 User Requirements**

**1.4.1 Client Requirements**

* User should be able to search or view and use system without registration.
* User should be able to register, login and logout if he need to communicate with companies and rating them.
* Show Accurate Location of user on a MAP.
* User should be able to search for any item.
* User should be able to filter companies on a MAP.
* User should be able to get nearest location or cheapest price of specific product.
* User should be able to view company profile and rating it.
* User should be able to edit, update or delete he’s personal information.
* User should be able to Send and Receive direct messages from different companies

**1.4.2 Companies (super user) Requirements**

* Super user should be able to register and login into the system to use it.
* Show Accurate Location on a MAP.
* Super user should be able create profile and has full management of it.
* Super user should be able to track most searched item (product).
* Super user should be able to change location information, personal information and Logo.
* Super user should be able to Send and Receive direct messages from different clients at the same time.

**1.5 System Requirements**

* Platform: Windows 7,8 or 10
* Programming Language: PHP v5
* Supported Database: MySQL 4.2
* Supported Web Servers: Apache 2.2.x
* Supported Requestor for Web service: HTTPs
* Supported Web Service Format: JSON
* Supported Browsers:
* Mozilla Firefox
* Apple Safari 2.x
* Google Chrome