Lab 11 – Web services and API

Note: This lab is optional. No marks are allocated to the lab tasks.

Task 1

Use a browser to visit the URL http://ws.cdyne.com/ip2geo/ip2geo.asmx
Click on the ResolveIP link, and enter the `ipAddress` (e.g., 119.224.142.40) and `licenseKey` 0. An XML fragment similar to the following should be returned with the geo information of the input IP address. This is an example of a Web service designed to be consumed by a web application. While this works fine from a browser, why cannot we write a Javascript that directly invokes a Web service?

```
<IPInformation xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="htt
p://www.w3.org/2001/XMLSchema-instance"xmlns="http://ws.cdyne.com/">
<Country>New Zealand</Country>
<Organization/>
<Latitude>-41</Latitude>
<Longitude>174</Longitude>
<AreaCode>0</AreaCode>
<TimeZone/>
<HasDaylightSavings>false</HasDaylightSavings>
<Certainty>90</Certainty>
<RegionName/>
<CountryCode>NZ</CountryCode>
</IPInformation>
```

Task 2

Get the following files from Lecture 11 example code:

webservice.htm
webservice.js
applicationproxy.php

These files create a “proxy” that means that the JavaScript and the invocation of the Web service have the “same source”.

Note: For security reasons it does not work when you are using AUT Intranet as you need to use AUT firewall proxy server (see applicationproxy.php line 12). But the code works when you use a public (e.g., Auckland City Free Wifi) or a private (e.g., at home) internet connection.

a) Modify the PHP file so that it gets the XML response listed in Task 1, load it into a DOM object, then converts the XML DOM object into JSON string (using json_encode() function) and sends it back to the client.
b) At the client side, use JavaScript to display the received information using a table.

Task 3

Get “Restful Web Service - Node+MongoDB” code from Lecture 11 and make the code work on your own device.