**5 closest Locations/Address case study**

**High-level functional requirements:**

1. Build a simple .NET application that allows a user to enter any real

Australian address.

1. Based on that address, the application finds the 5 closest locations (from the addresses in the provided TXT file given).
2. The output should be a list of addresses, sorted closest first.
3. The distance in kilometers should be visible for each result.

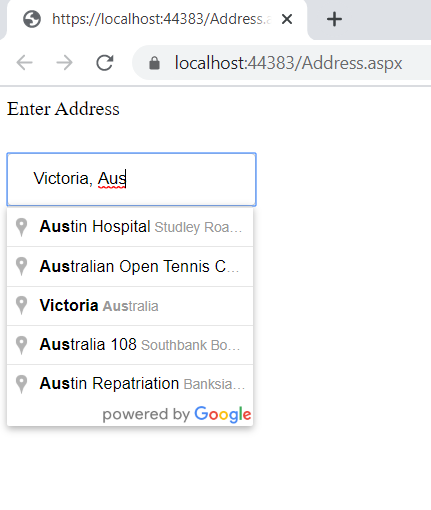
**Minimum technical requirements:**

You can use whatever implementation method and algorithm you like for your application, so long as:

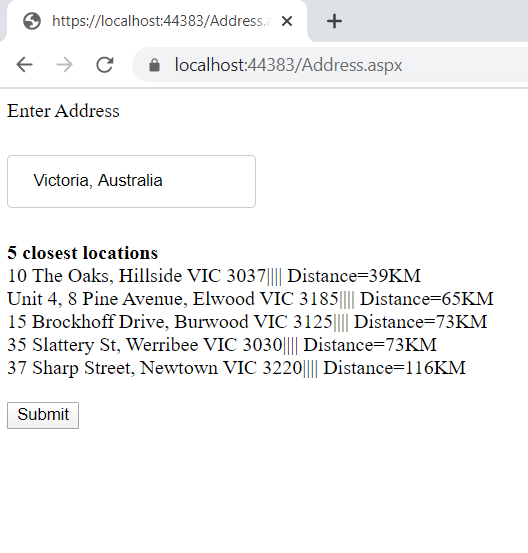
1. The underlying platform is .NET
2. The business logic and algorithm is correct (but a “flat surface” calculation is fine)
3. The “search” business logic (i.e. the ability to search for an address and find the 5 closest locations)
4. Is exposed via an API, written by you
5. The API has appropriate code coverage by unit tests and an appropriate structure
6. The output of the application contains the right information
7. The solution comes with a “readme” file containing any relevant instructions
8. The solution can be built and run by using the F5 command in Visual Studio

**How to use solution:**

1. You should have a Visual Studio 2019.
2. You should have a .NET framework 4.7.2.
3. You need to Sync a Git repo from here.
4. Once you are done with pulling the project from git you can directly use F5 for running it.
5. Once you get the Address.aspx page you need to follow the below steps.
   * 1. Enter any address which you want.
     2. As per requirement, you can add the Australian Address.



* + 1. Then click on Submit.
    2. You will get 5 closet locations from the entered location.
    3. This application will also show you the distance.



**Solution Structure:**

1. As per mentions, I use Apicontroller for business logic.
2. This API is internally created in the project.
3. API provides all the information.
4. The first API call is heavy, but once you make the call the subsequent call for this application will be smooth as I implemented cache which has expiry right now of 60 min.
5. I also have validation at the places where it is required .
6. I also have exception handling for risk-prone code in place.
7. I also have garbage management in place.
8. I also have a Unit test project in place which test api from backend with some test data.

For creating this solution I used Google Maps API so bit credit goes to Google Maps API services.

Ref:

<https://console.cloud.google.com/google/maps-apis/overview>

<https://developers.google.com/maps/documentation/javascript/directions>