**EMSI GEOGRAPHY LOOK UP**

**DESCRIPTION**

A web user interface that provides a user with a capability of geography look up with auto complete functionality.

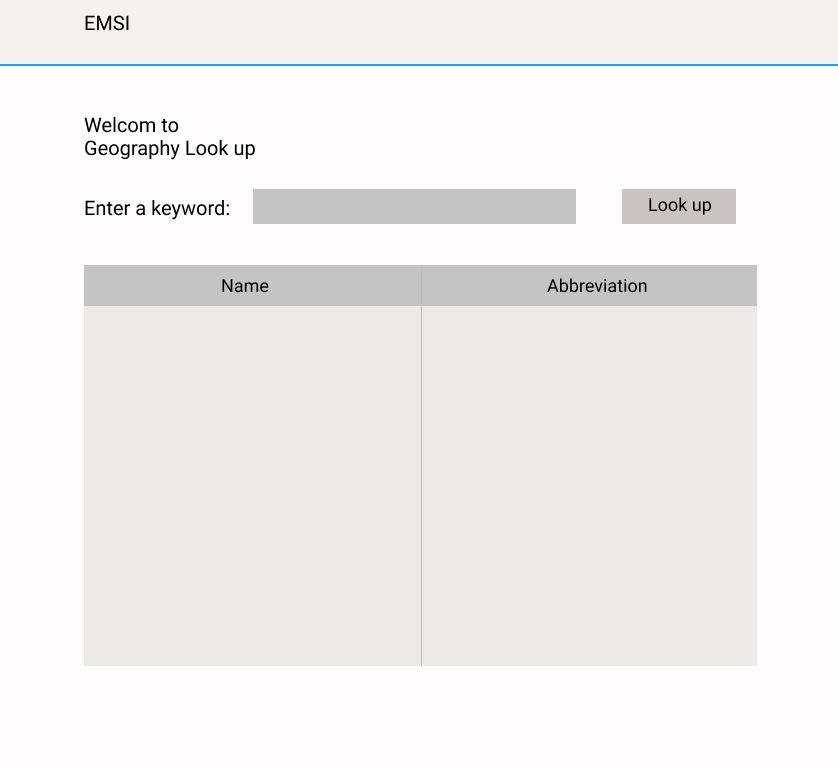
Example 1: User can enter a county and get all the underlying states and cities

Example 2: User can enter a state and get all the underlying cities.

This Web user interface interacts with the APIs to fetch details. The data is stored within an SQLite file.

**PROTOTYPE**

I have created the following wireframe to decide on the key features of the prospective product.



**PROJECT STRUCTURE**

The project is divided into three parts: WEB, API and TEST

WEB: This the user interface application, that deals with how the information is presented to the user. Website design will happen here using HTML, CSS, and Bootstrap. This application will interface with the API to fetch information using End points.

API: This is the application that servers APIs by providing End points to the users to consume. It will communicate with the database and fetch data.

Here is a link to the API documentation: <https://documenter.getpostman.com/view/6506382/TzeajRSf>

TEST: This is the application that will perform unit testing and integration testing for the API.

**SETUP**

**emsiproject.WEB**

1. The project is based on .NET 5. Please download the latest version from here [**DOWNLOAD**](https://docs.microsoft.com/en-us/dotnet/core/install/windows?tabs=net50)
2. The project UI is based on HTML, CSS and Bootstrap and it follow site.css for styling.
3. All the API calls can be found under APIClasses folder.
4. For the state management, the project uses Session, and the configuration can be found on startup.cs and the logic can be found under HomeController.
5. You may need to change the API configuration base address by going to appsettings.json depending on whatever base address you are using to host API.

NugetPackages Installed/Needed for this project:

1. RuntimeCompilation: Runtime Compilation support for Razor views and razor pages to see the changes after saving views, without the need to rebuild the application. Please find my blog [here](https://whilesohail.wordpress.com/2021/06/15/runtime-compilation-in-net-core/) to learn more.
2. Newtonsoft.json: This is a high performance json framework for .NET.

**emsiproject**

1. The project is based on .NET 5. Please download the latest version from here [**DOWNLOAD**](https://docs.microsoft.com/en-us/dotnet/core/install/windows?tabs=net50)
2. All the APIs are found under AreasController file.
3. Database Configurations can be found under DatabaseConfig.cs and startup.cs
4. Database connectivity details like *data source* can be found under appsettings.json.
5. This project communicates with sqlite3 database file which can be found under this project with name as “*areas.sqlite3*”

NugetPackages Installed/Needed for this project:

1. Microsoft.Data.Sqlite: Lightweight ADO.NET provider for SQLite.
2. For logging add this nuget packages: Serilog, Serilog.AspNetCore, Serilog.Exceptions, Serilog.Sinks.Console
3. Swashbuckle.AspNetCore for documenting APIs built on ASP.NET Core

Emsiproject.TEST

1. This project is a console-based application that is developed to write test cases for APIs.
2. Testing the APIs has been implemented in a Test-Driven Development environment.
3. This project contains various tests categorized into signature tests, operations tests, and connectivity tests.

NugetPackages Installed/Needed for this project:

1. MSTest.TestAdapter: Adapter to discover and execute MSTest Framework based tests.
2. MSTest.TestFramework: This is MSTest V2, the evolution of microsoft’s test framework.

**RUNNING AND DEBUGGING**

1. To start the application locally run both API and Web.

To do this, right click on the solution, click properties, expand Common Properties, Click Start up project, and select multiple startup projects. And select emsi project as start without debugging and emsiproject.web as start without debugging.

1. To run the tests, open test explorer window, run all the tests.
2. To debug follow step #1, except in the startup project window select both API and web project as “start”.

**PENDING WORK**

1. Display all the areas into categories as Counties, States and Cities.
2. Add self-signed TLS/SSL self-signed certificate.
3. Make application ready to run on Ubuntu (Docker).

**FUTURE IMPROVEMENTS**

**User interface side**

1. Caching mechanism can be implemented at the user interface to enhance user experience.
2. Top recent search can be displayed at the UI, for this we need to count the search for a particular query and store it into a separate table with key-value pair.
3. The Area list can be made to scroll, so as to fit the overflowing UI table.

**API Development**

1. Load balancer to distribute requests across various server instances, to make faster API serving.