**Summary**:

I have created different folders to segregate the layers:

* **API** folder contains call to Google API
* **DataAccess** folder have database interaction classes
* **Models** folder holds project model classes
* **Services** folder keeps the services
* To apply dependency injection, I have created interfaces
* **Dependency injection** is injected in the startup.cs
* We can configure ocelot gateway to handle microservices call. I have added the nugget package and configuration file in project
* **Create Database** “GoogleTimeZoneDB” and execute the attached database script to generate a table or If you are using any other MSSQL database, please execute the attached script and update in project configuration

**Google Time Zone API Visual Studio Setup:**

* Setup the project in Visual Studio you will have to install following dependencies
  + Visual Studio 2019
  + SQL Server v14
  + Microsoft.EntityFrameworkCore.SqlServer
  + System.Net.Http.Json
  + Ocelot API Gateway
* Update ConnectionString as per your local settings
* Google APIKey:
  + Generate google API key from <https://developers.google.com/maps/documentation/timezone>
  + And update “APIKey” in appsettings.json

**API Configuration in IIS:**

* Publish the API in a folder
* In IIS create a new website and point it to the API publish folder
* Change connectionstring and API key as required

**Rest API Testing Assumption:**

* Set the defaulted sample URL in launchSettings.json so you can directly run the service
* Usually I used Ocelot API Gateway to support authentication, routing and caching
* Normally, we setup JWT token authentication. User needs to pass username, password, secretId and grant\_type etc in the header to get token. Token authentication expiry can be set 20 minutes or more.
* I assume that token has been acquired before making this end point call using above required parameters.
* I assume that request and response save in single database table because 1-1 relationship
* Normally we have validation on input parameters but I assume that we will response the exact exception. It can easily be modify in a service

**Testing in Postman:**

* Change the following port according to local configuration or IIS setting
* <http://localhost:2589/API/TimeZone/location=39.6034810,-119.6822510&timestamp=1331161200>
* I assume API is running locally at <http://localhost:8080>
* In the Postman create a Get call and paste <http://localhost:2589/API/TimeZone/> in the URL field
* Under Param tab create following key values
  + Key: location
  + Value: 40.469474,176.384219
  + Key: timestamp
  + Value: 1331161200
* I believe you have already acquired JWT token which is a Bearer type in Authorization tab
* Click on Send button