GLEAC Levenshtein Distance Prototype

Anand Shah - [anand\_shah0604@outlook.com](mailto:anand_shah0604@outlook.com)

Date Created: 25-NOV-2020

Last Revision: 1.0

**Table of Contents**

[1 Problem statement, data and technical specification 3](#_Toc57208461)

[1.1 Introduction 3](#_Toc57208462)

[1.2 Prototype/POC 3](#_Toc57208463)

[1.3 Data Validation Specifications 3](#_Toc57208464)

[1.4 Technical specifications: 3](#_Toc57208465)

[2 Solution Design 3](#_Toc57208466)

[2.1 Introduction 3](#_Toc57208467)

[2.2 Platform 4](#_Toc57208468)

[2.2.1 Development 4](#_Toc57208469)

[2.2.2 Components 5](#_Toc57208470)

[2.2.3 Solution Structure 6](#_Toc57208471)

[2.3 Screen 8](#_Toc57208472)

[2.3.1 Introduction 9](#_Toc57208473)

[2.3.2 Views 9](#_Toc57208474)

[2.3.2.1 Setup token and access the application 9](#_Toc57208475)

[2.3.2.2 Criteria 9](#_Toc57208476)

[2.3.2.3 Result 10](#_Toc57208477)

[3 Test (Rest API) 13](#_Toc57208478)

[3.1 Test Using Postman 13](#_Toc57208479)

[3.1.1 Token Generation 13](#_Toc57208480)

[3.1.2 Levenshtein distance Calculation using generated token 14](#_Toc57208481)

[3.2 Test using Fiddler 14](#_Toc57208482)

[3.2.1 Token Generation 15](#_Toc57208483)

[3.2.2 Levenshtein distance Calculation using generated token 15](#_Toc57208484)

[4 Revision Log 16](#_Toc57208485)

# Problem statement, data and technical specification

## Introduction

The Levenshtein distance is a number that tells you how different two strings are. The higher the number, the more different the two strings are.

For example, the Levenshtein distance between “kitten” and “sitting” is 3 since, at a minimum, 3 edits are required to change one into the other.

* kitten ? sitten (substitution of “s” for “k”)
* sitten ? sittin (substitution of “i” for “e”)
* sittin ? sitting (insertion of “g” at the end).

## Prototype/POC

To create SPA and web APIs for finding the Levenshtein distance (LD) between two strings.

## Data Validation Specifications

All validation checks have to be implemented on both front-end and API side. Identify, list and implement all possible necessary checks.

## Technical specifications:

* API needs to be REST-based Web API using JSON for data transfer.
* Calls have to be made over SSL. Or at least they need to be redirected to SSL.
* We should be able to access the API with POSTMAN/Fiddler. But kindly protect the API with authorization token of your choice.
* The web interface should allow the user to enter two strings and on submitting the same,

the calculated distance should be stored in local/session storage.

# Solution Design

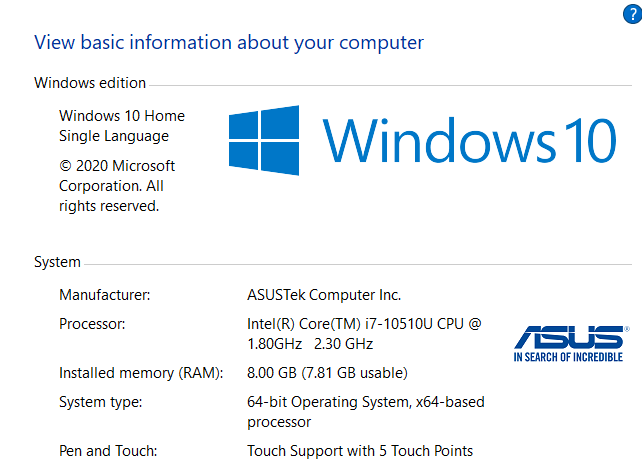
## Introduction

The Levenshtein distance web application has been structured into the following components considering separation of concerns.

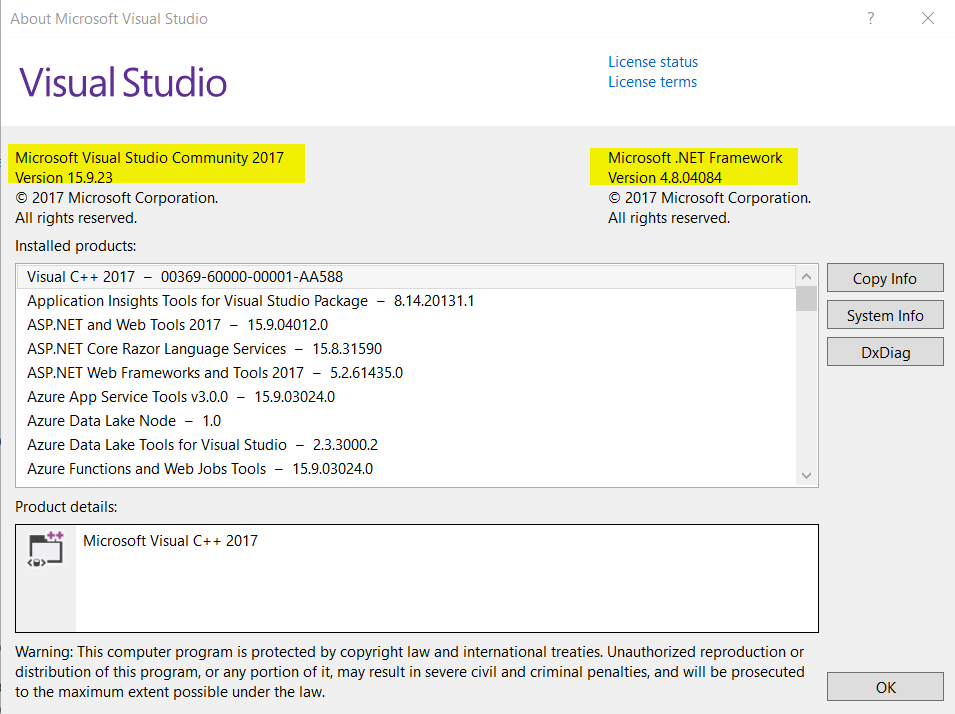
## Platform

### Development

* OS: Windows 10

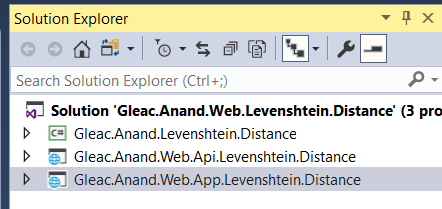


* IDE: (Integrated Development Environment): Visual Studio 2017 Community Edition



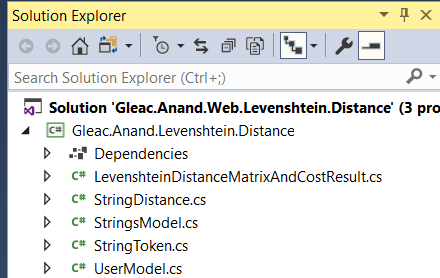
* Microsoft.AspNetCore.Razor.Design(2.1.2)
* SDKs:
  + Microsoft.NETCore.App(2.1)
  + Microsoft.AspNetCore.App(2.1.1)
* Development WEB Server: Kestrel

### Components



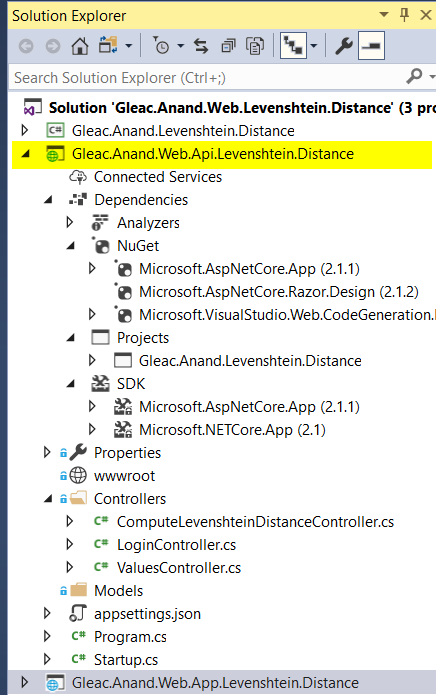
* General Library (**Gleac.Anand.Levenshtein.Distance**):
  + It is a class library that contains domain classes and algorithm(s) required for calculating the distance between two strings following Levenshtein algorithm.
* Restful API/WEB API (**Gleac.Anand.Web.Api.Levenshtein.Distance**):
  + It is a ASP .NET Core API project which has been used to develop Restful API for the following operations.
    - Generating the JWT token.
    - Returning the results of Levenshtein algorithm utilizing the generated token. The token is used to access the WEB/Restful APIs that are tagged as authorized.
* WEB APP **Gleac.Anand.Web.App.Levenshtein.Distance)**:
  + It is a ASP .NET Core MVC project which has been used to develop SPA and its different views. It uses ASP .NET core libraries, Bootstrap framework and Jquery framework.

### Solution Structure



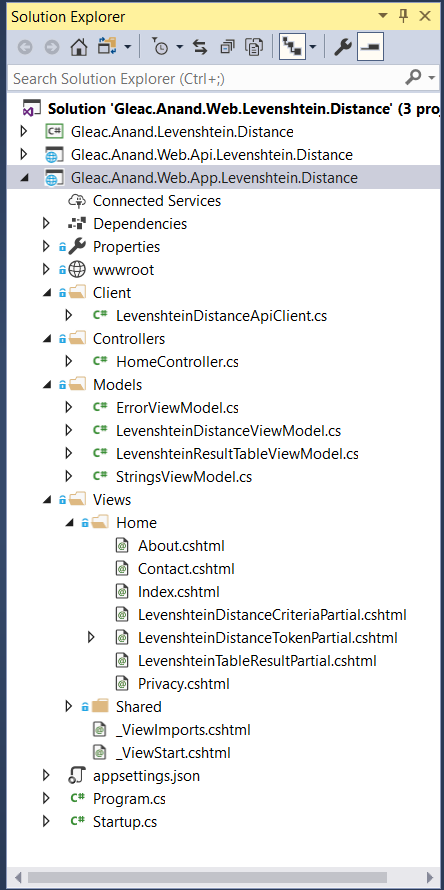
General Library (**Gleac.Anand.Levenshtein.Distance**) – expanded view

* StringDistance.cs:
  + The class calculates the Levenshtein distance and gives the following result:
    - Cost
  + The compute array that helps to generate the cost matrix table.
* StringsModel.cs:
  + The class contains the first/reference string and second/hypotheses strings. It is an input to the class “StringDistance.cs” that computes the result using Levenshtein algorithm.
* StringToken.cs:
  + The class acts as the repository for the generated JWT token. JWT token is exchanged from WEB API to WEB APP and vice versa via StringToken.cs.
* UserModel.cs:
  + The class acts as the repository for user details like username, password etc. It is the repository that stores the details and is used to exchange from WEB API to WEB APP and vice versa.



Restful API/WEB API (**Gleac.Anand.Web.Api.Levenshtein.Distance**) – expanded view

* Compute.LevenshteinDistanceConroller.cs:
  + The WEB API controller/class calculates the Levenshtein distance and gives the following result:
    - Cost
  + The compute array that helps to generate the cost matrix table.
* LoginController.cs:
  + The class/controller computes the result using General Library (**Gleac.Anand.Levenshtein.Distance**)
* ValuesController.cs:
  + The class/controller is used for testing purpose and not utilized for Levenshtein.Distance calculation.



WEB APP (**Gleac.Anand.Web.App.Levenshtein.Distance) –** expanded view

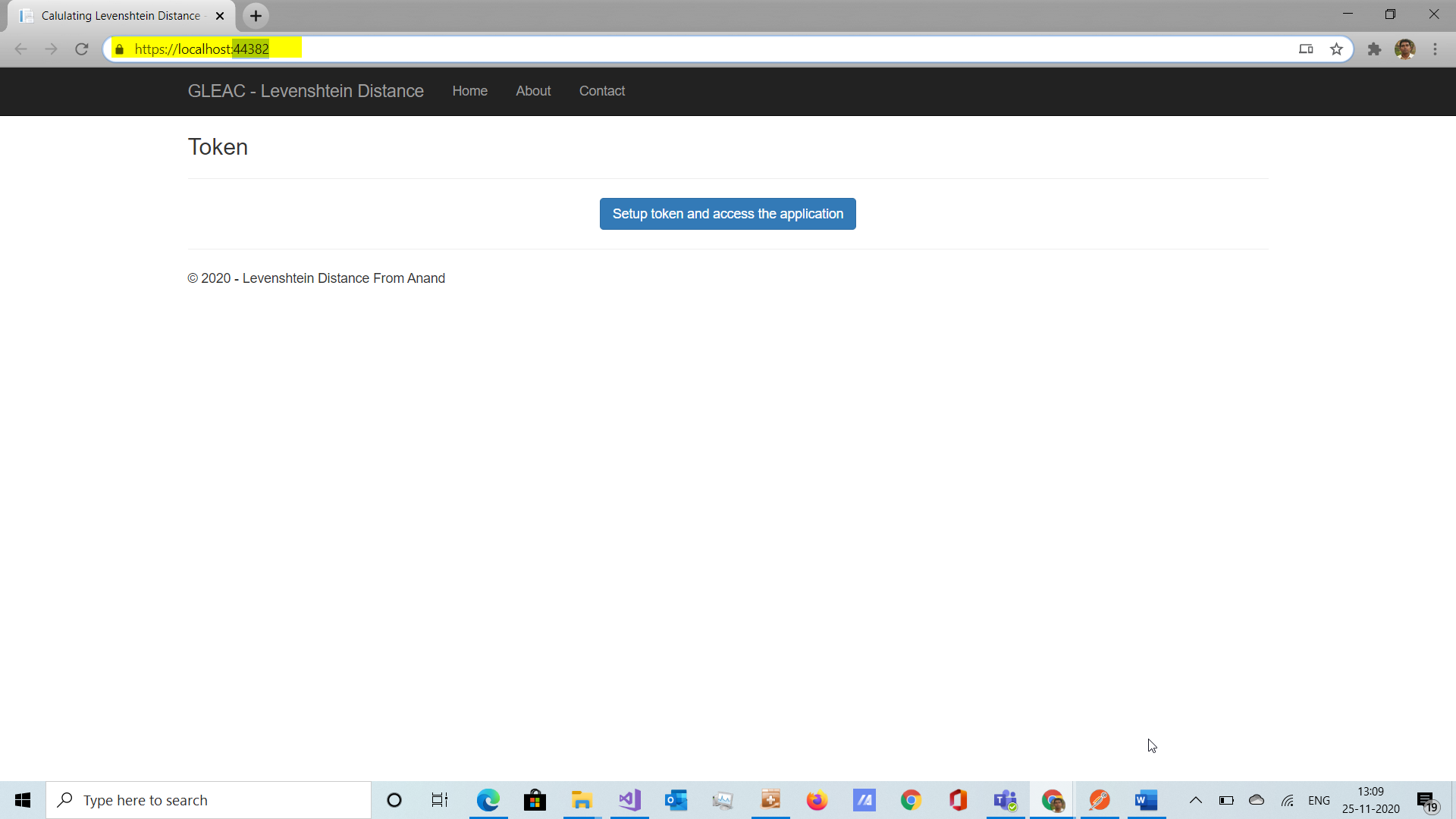
The views are explained in the screen section.

## Screen

### Introduction

### Views

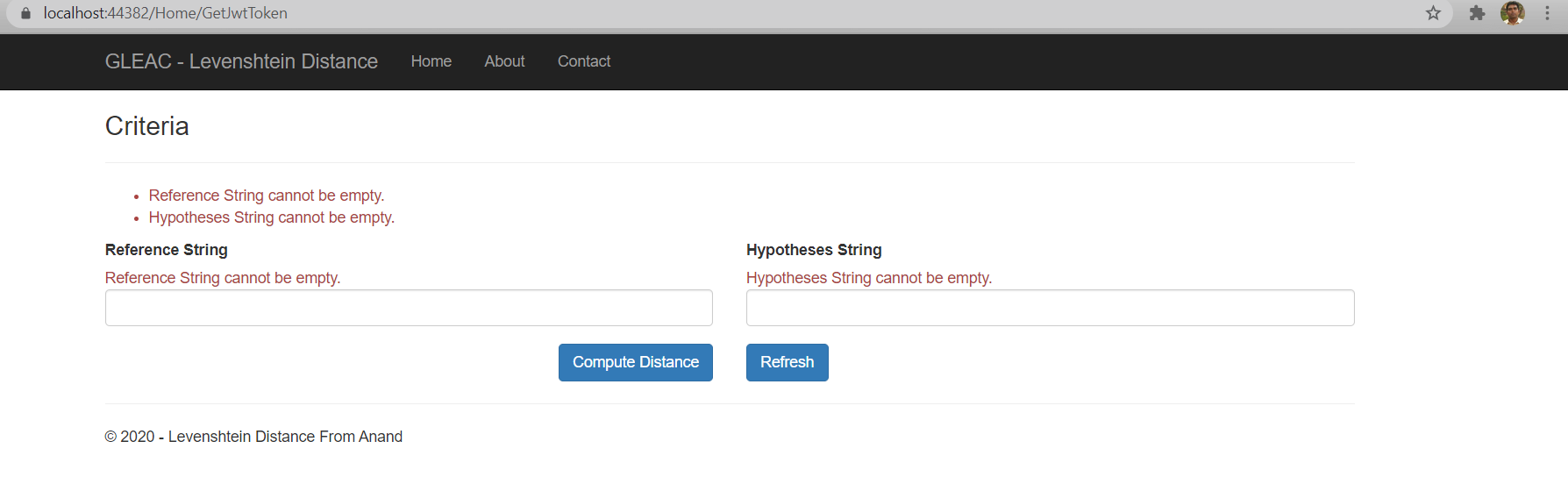
#### Setup token and access the application

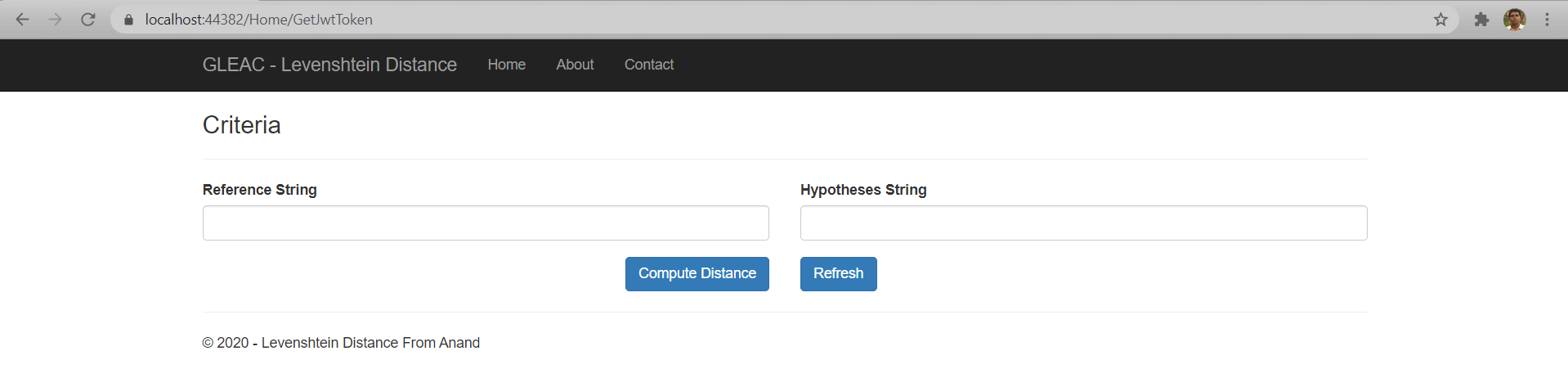


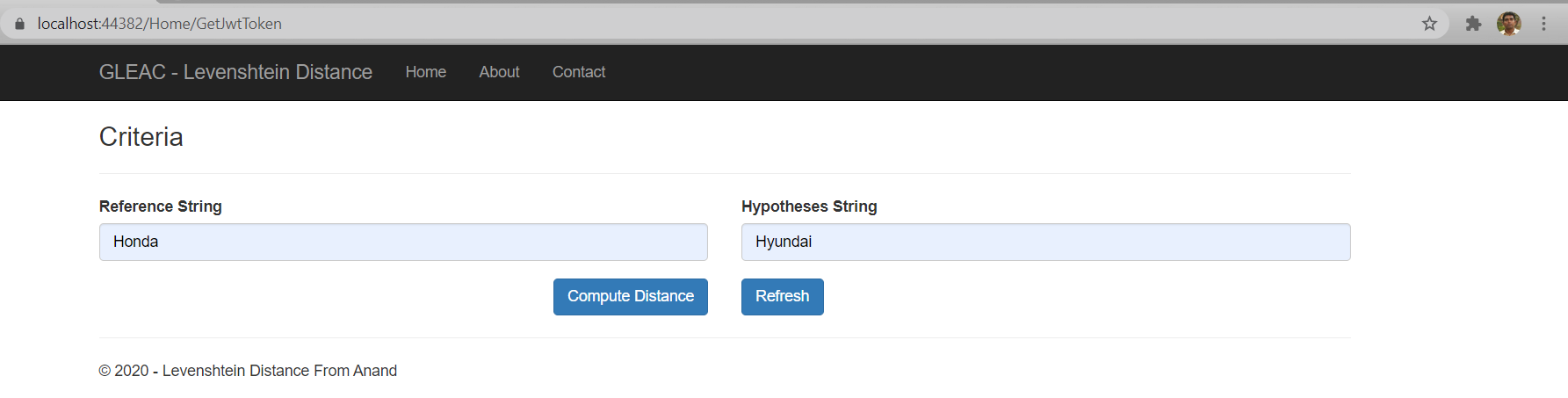
LevenshteinDistanceTokenPartial.cshtml

The view is used to setup the access token and show the criteria view.

#### Criteria



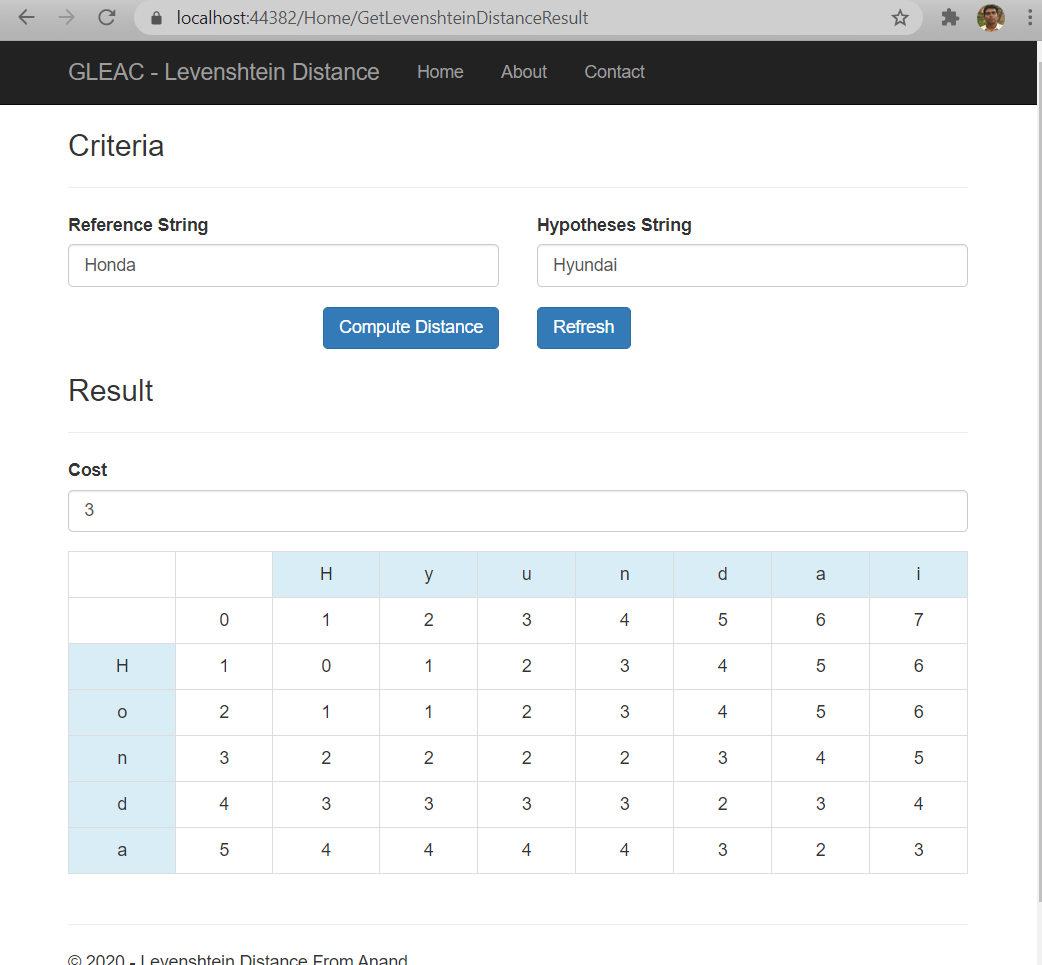




LevenshteinDistanceCriteriaPartial.cshtml

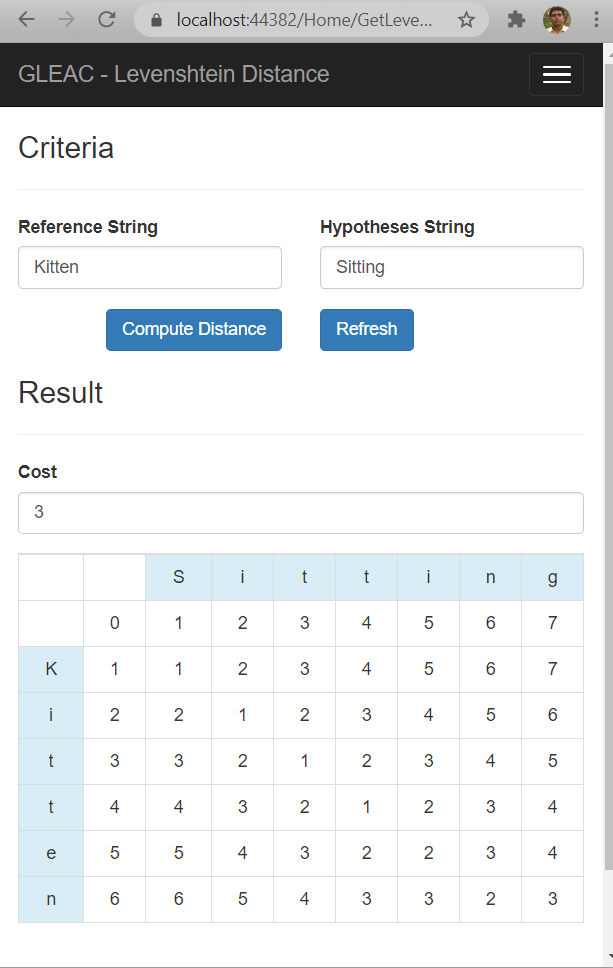
The view takes two strings as the input and shows the result view.

#### Result



LevenshteinTableResultPartial.cshtml

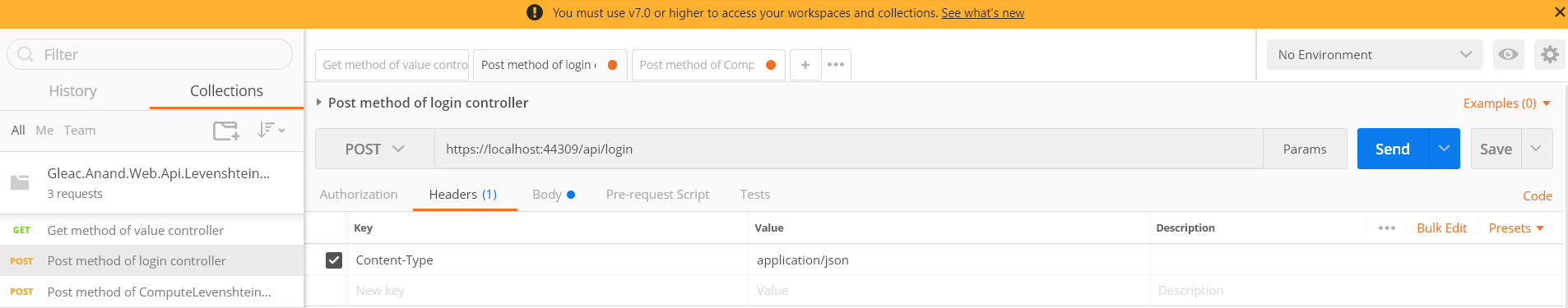
The result view takes the string as input and shows the cost and matrix.



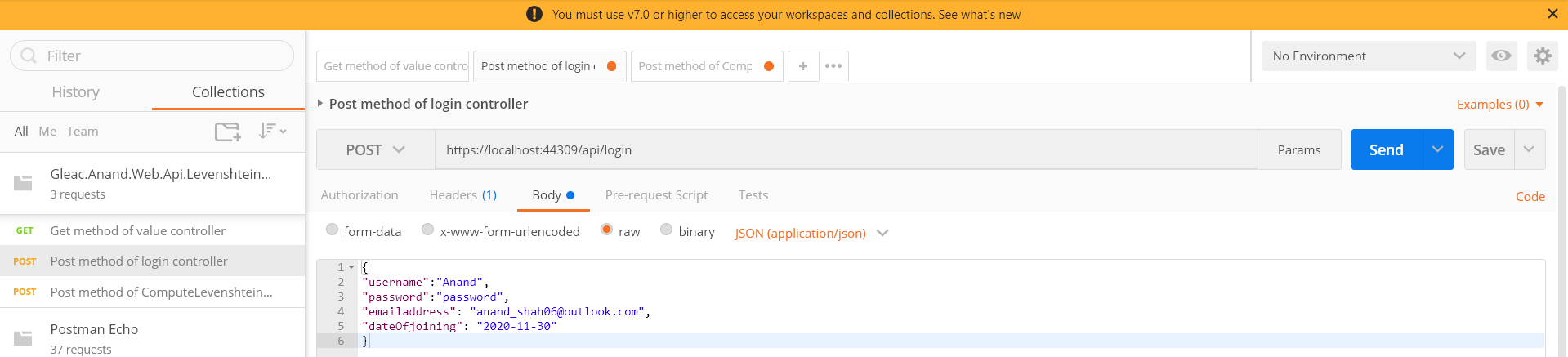
# Test (Rest API)

## Test Using Postman

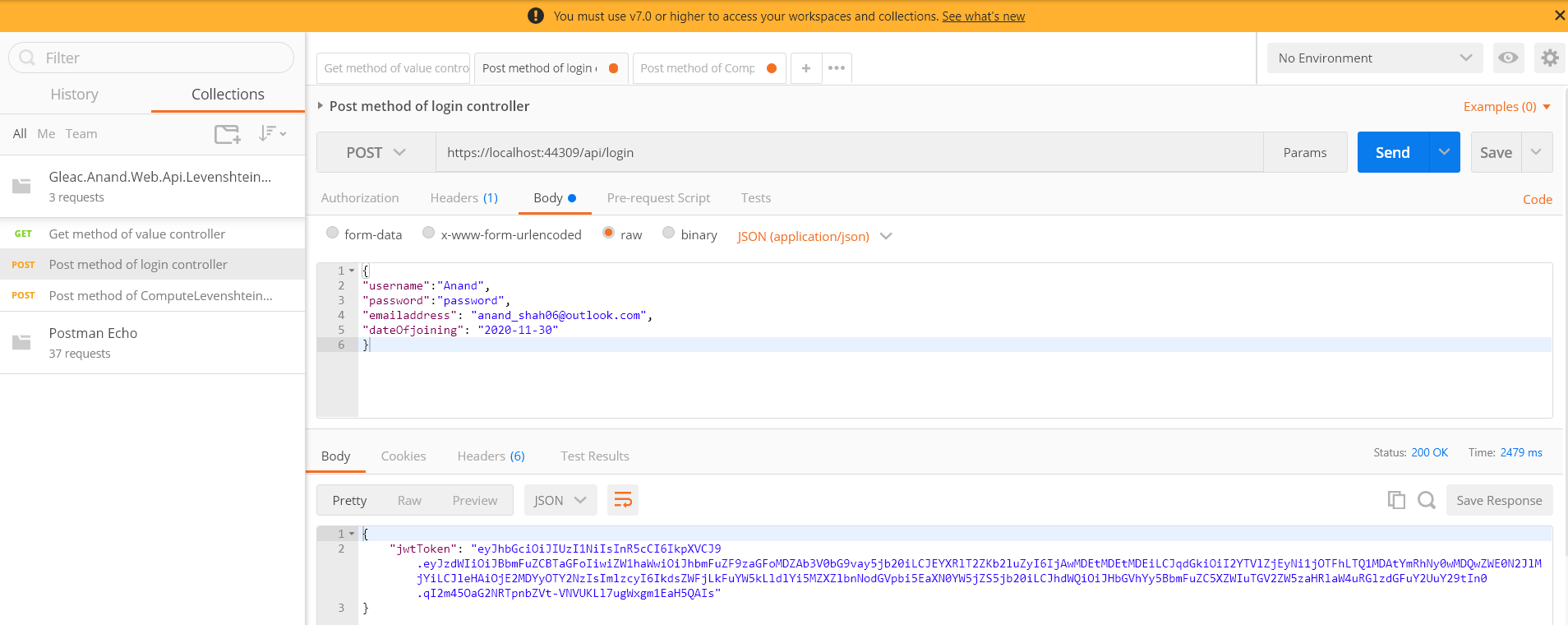
### Token Generation



Setting header for the login Rest/Web API.

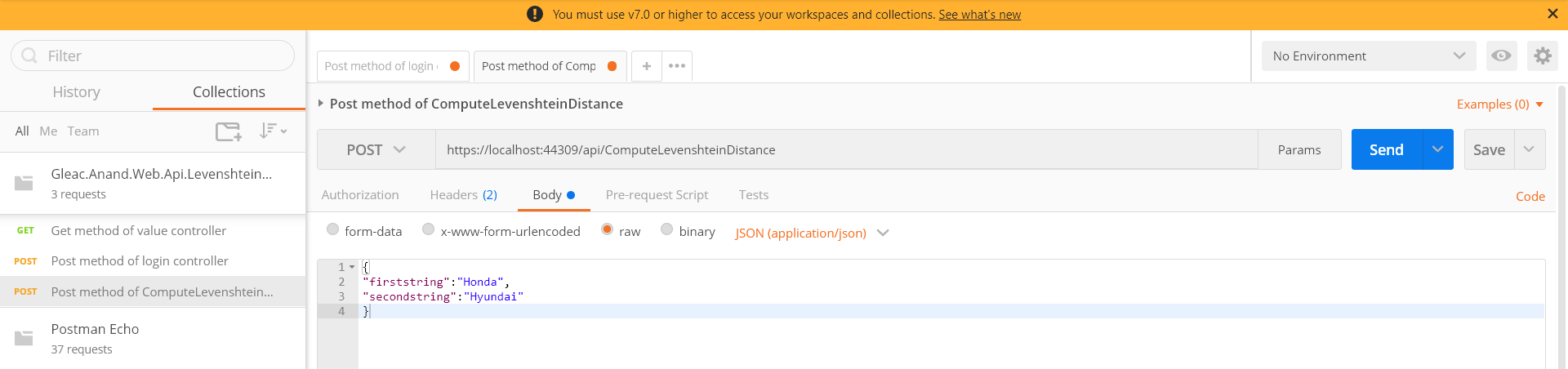


Setting the inputs as shown to generate JWT token. Primarily username is used. Others are for info.

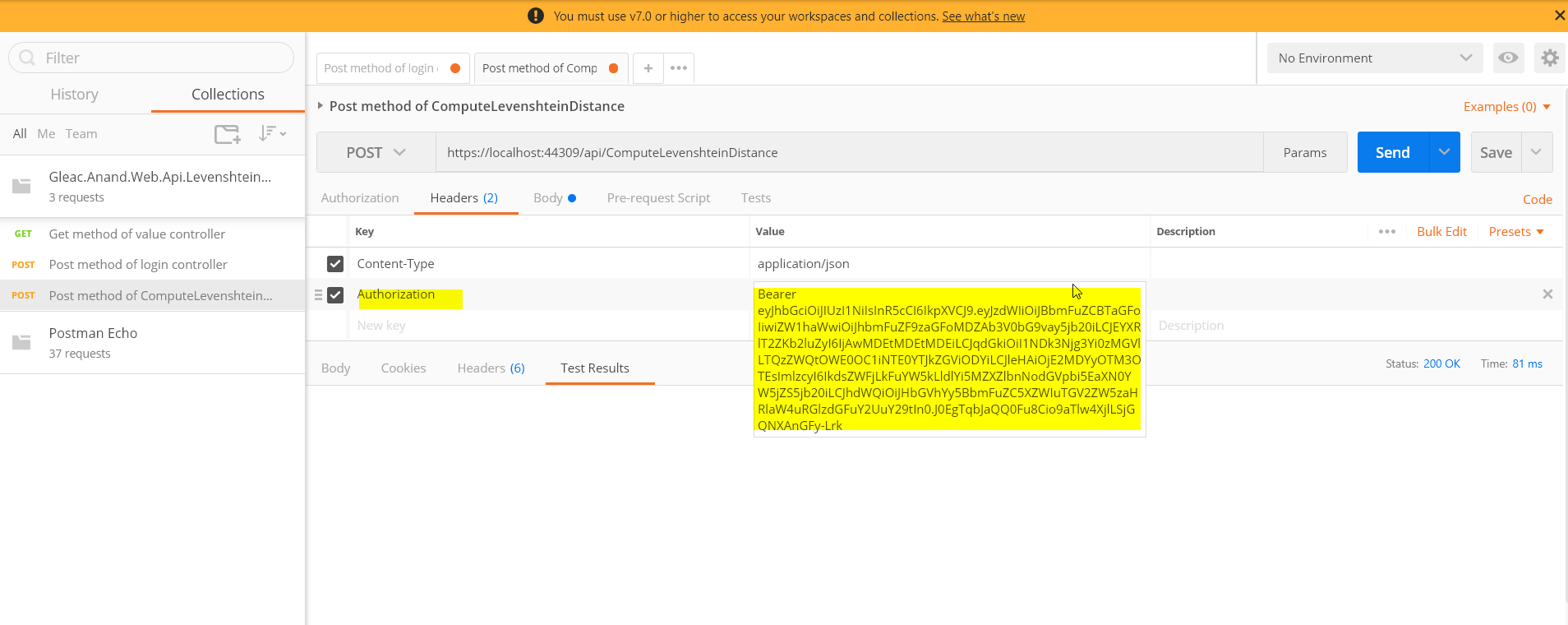


The output showing the generated token.

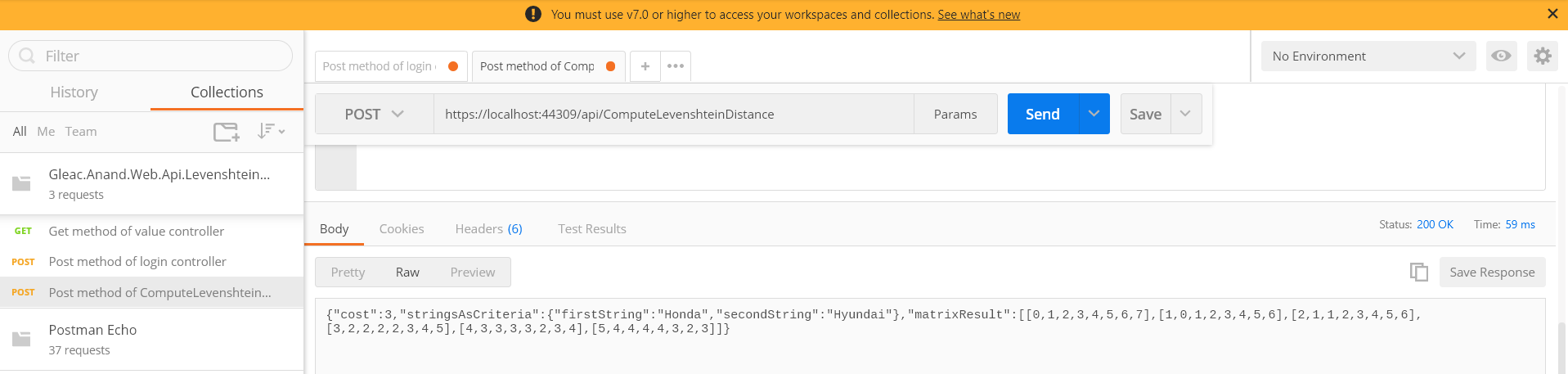
### Levenshtein distance Calculation using generated token



The Rest/Web API takes inputs as first string and second string.



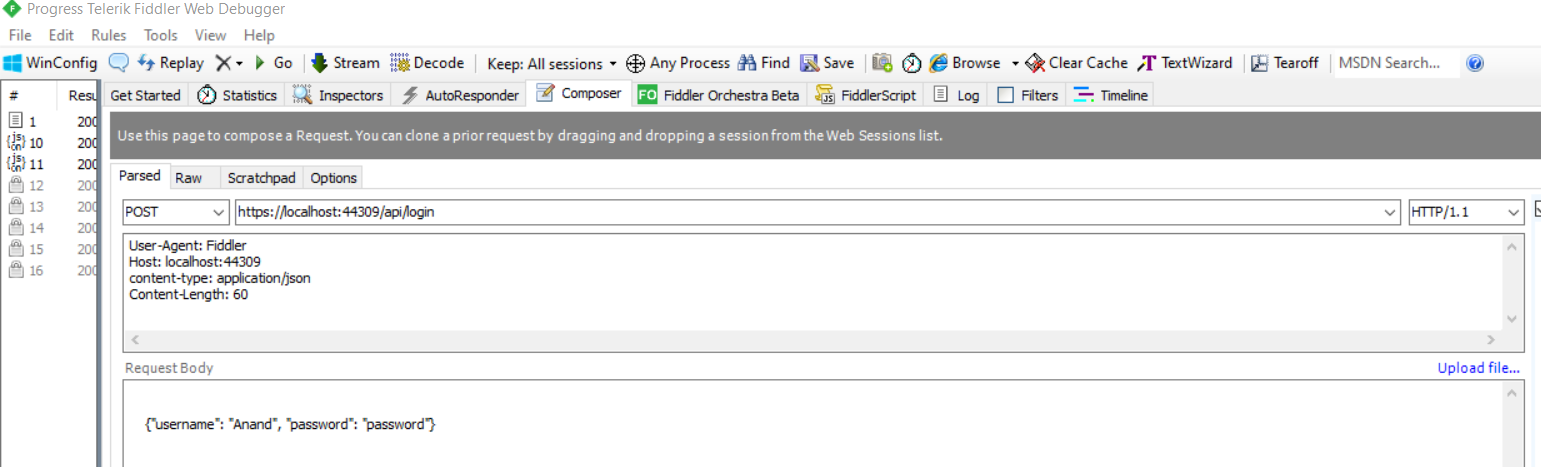
Since the WEB API is tagged as authorized, authorization header is set with JWT token.



The output showing the generated result.

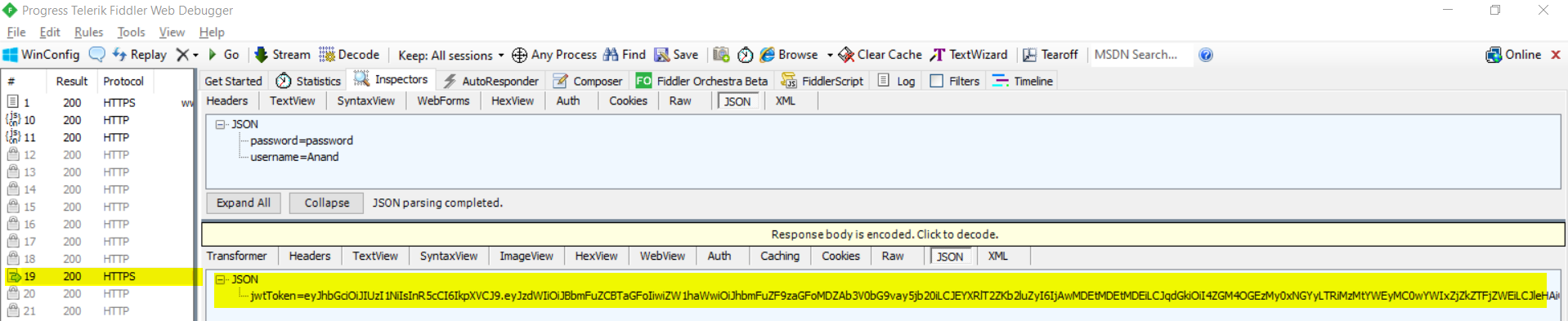
## Test using Fiddler

### Token Generation



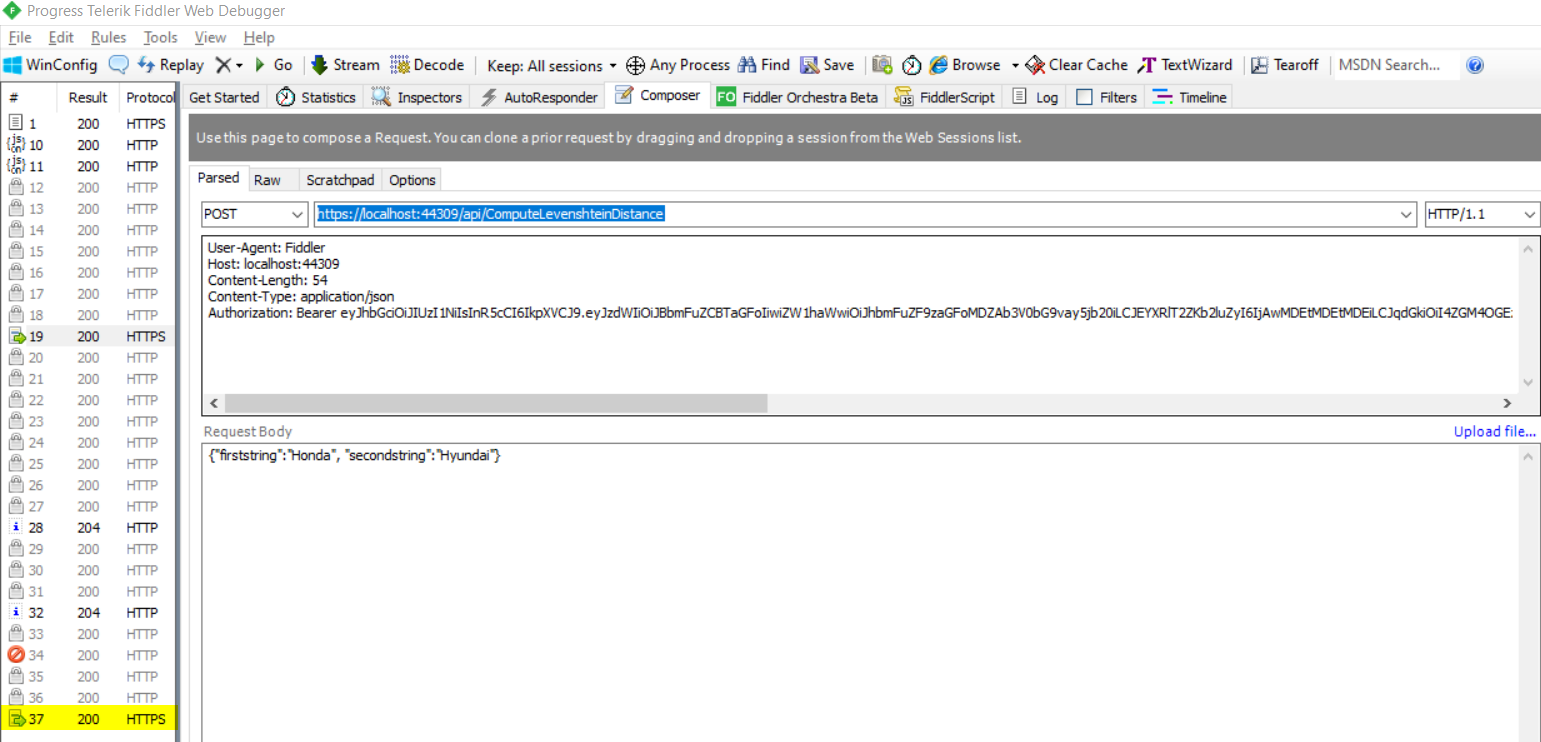
Setting header for the login Rest/Web API.

Setting the inputs as shown to generate JWT token. Primarily username is used. Others are for info.



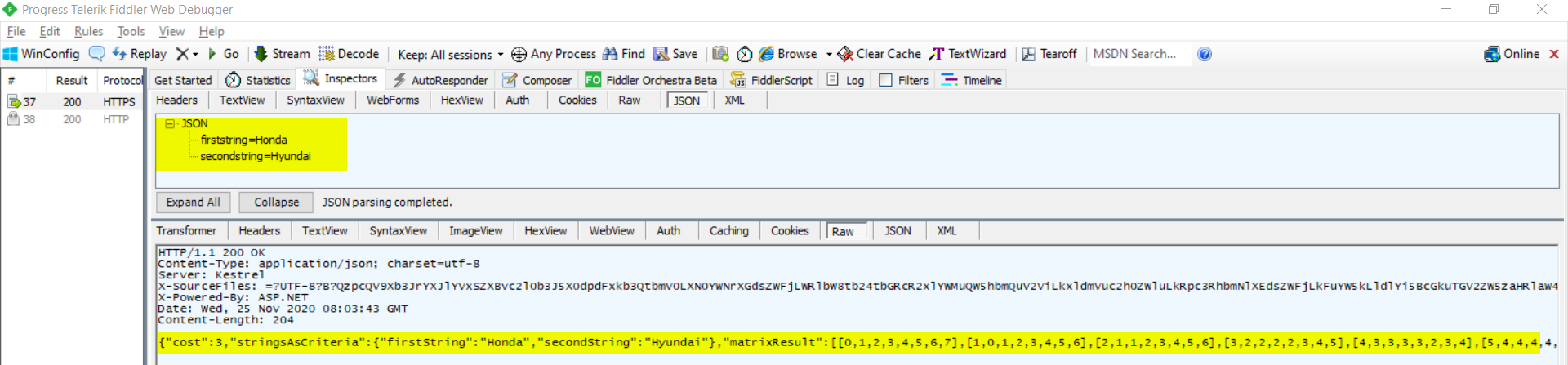
The output showing the generated token.

### Levenshtein distance Calculation using generated token



The Rest/Web API takes inputs as first string and second string.

Since the WEB API is tagged as authorized, authorization header is set with JWT token.



# Revision Log

|  |  |  |  |
| --- | --- | --- | --- |
| **Revision Log** | | | |
| Revision log is updated by the user who updates the document.  Initial revision no. should begin with 1 and then increment by 1 (i.e., 1, 2) | | | |
| **Rev#** | **Revised By** | **Date** | **Revision Details** |
| 1 | Anand Shah | 25-NOV-2020 | Document created and updated with content. |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |