

Lyft/Uber Comparison Application

Technical Design Document / Configuration Document

URL: <https://taxicompareapp.azurewebsites.net/>

Developers for this App:

Kar Ho Tan,

Mark Wang,

Leron George

December 9, 2018

**Revision History**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **Date Updated** | **Updated By** | **Version** | **Overview of Updates Made** |
| 1 | *11/06/2018* | *Leron George* | *Draft* | *Creation of document* |
| 2 | *12/2/2018* | Mark Wang | *Update* | *Additional technical info added* |
| 3 | *12/8/2018* | *Leron George* | *Additional Edits* | *Revisions to include further details* |
| 4 | *12/9/2018* | Kar Ho Tan | *Final Version* | *Review of complete document* |

***Reviewers / Developers***

|  |  |  |  |
| --- | --- | --- | --- |
| **Reviewer** | **Role** | **Reviewed On** | **Comments** |
| Kar Ho Tan | Developer/QA | 12/2/2018 |  |
| Mark Wang | Developer/QA | 12/2/2018 |  |
| Leron George | QA/Design and Documentation/Presentation | 12/7/2018 |  |

Contents

[Overview of Application’s purpose 4](#_Toc532057456)

[ASP.Net Architecture 4](#_Toc532057457)

[Architecture and topology for Azure 5](#_Toc532057458)

[How Azure Application Proxy Works 6](#_Toc532057459)

[Structural integrity and specifics of Application Architecture 7](#_Toc532057460)

[IDE: 7](#_Toc532057461)

[Application: 7](#_Toc532057462)

[Server: 7](#_Toc532057463)

[Summary of JavaScript, Class files, c# HTML, and APIs 8](#_Toc532057464)

[Database Tables 10](#_Toc532057465)

[Start Up Home Page 11](#_Toc532057466)

[Page – Requesting Login 11](#_Toc532057467)

[Page – Register/Create New Account for Login – 11](#_Toc532057468)

[Login Start up Page 12](#_Toc532057469)

[Fields 12](#_Toc532057470)

[Page – Taxi Price Compare Form – Displays the field locations and the Google API that allows navigations 13](#_Toc532057471)

[Fields 13](#_Toc532057472)

[Page – Search return Page – that maps the location path on the Map display from Points ***A*** to Point ***B*** 14](#_Toc532057473)

[Page – Order History – that maps the location path on the Map display from Points ***A*** to Point ***B*** 15](#_Toc532057474)

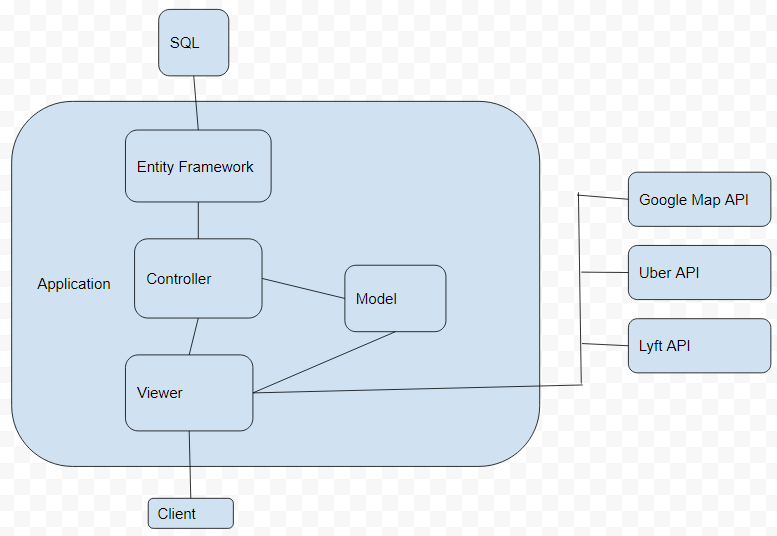
## Overview of Application’s purpose

**Application’s Url:** [**https://taxicompareapp.azurewebsites.net/**](https://taxicompareapp.azurewebsites.net/)

The purpose of this application is to provide a place that allows users to query various start and end location destinations in NYC that will compare Uber/Lyft prices per travel rates and store them in a repository database for historical reference and possibly reporting. The users will be able to track this information by creating account profiles via Azure cloud servers, search from and to locations using Google API Maps, review display price comparisons and select the desired price to save for historical review. The user will be able to view all order history for future comparisons and develop reports that display this detailed information.

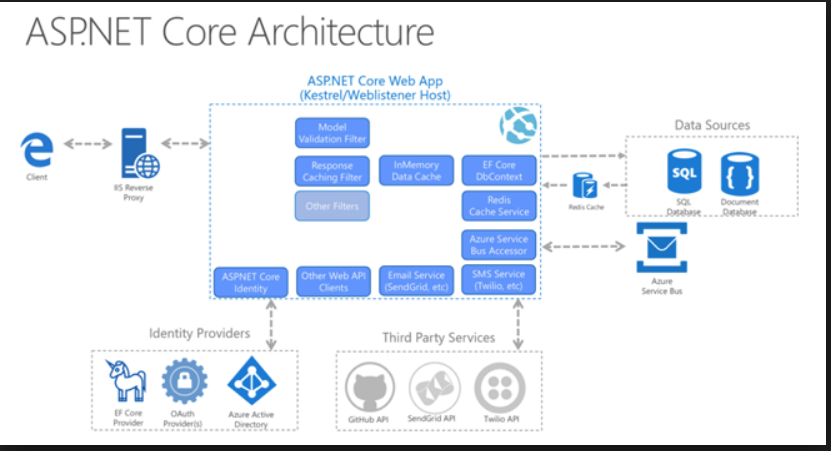
## ASP.Net Architecture

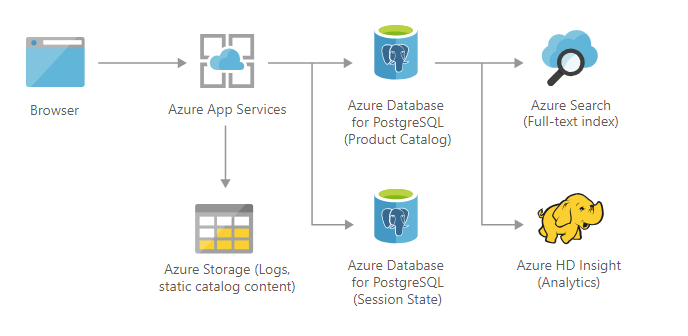
The Architecture Diagram for our application is shown below, we use SQL to maintain and validate all our data which our .net entity Framework consumes (Along with additional APIs, such as google Map, Uber and Lyft) and all is published to the cloud through azure to display to the client-side user interface.

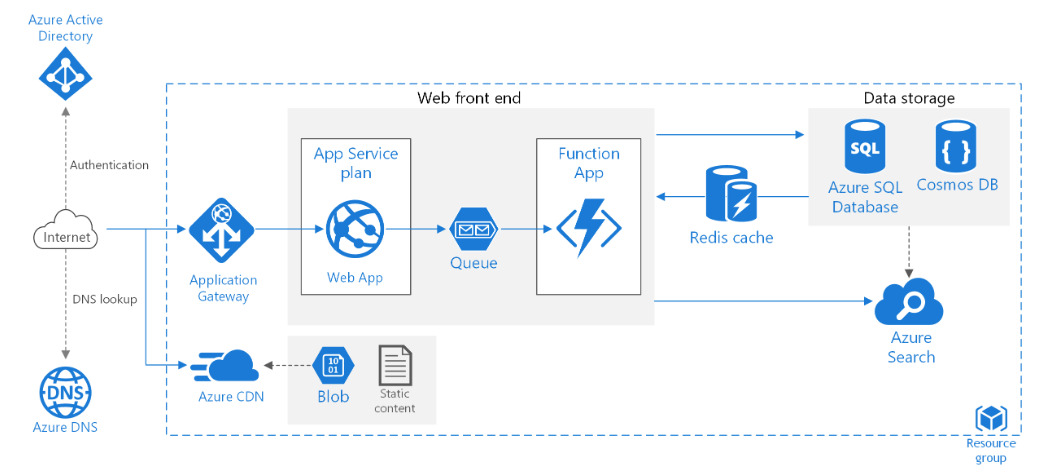


## Architecture and topology for Azure

Below are diagrams of the Architectural topology of the Azure cloud sever platform layered on our database connection and ASP.NET page used in this project along with many such examples for clarification.



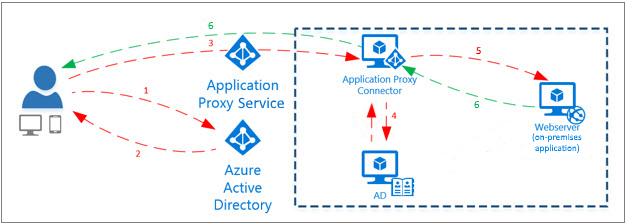




The Azure connection call between the web API and the database is described below:

1. Azure App proxy will redirect the call to On-Premise Web API (Web Service) through the Azure App Proxy Connector.
2. Web API will query the Database and get the results back to the User.

### How Azure Application Proxy Works



1. The user accesses the application through the Application Proxy service and is directed to the Azure AD sign-in page to authenticate.
2. After a successful sign-in, a token is generated and sent to the client device.
3. The client sends the token to the Application Proxy service, which retrieves the user principal name (UPN) and security principal name (SPN) from the token, then directs the request to the Application Proxy connector.
4. If you have configured single sign-on, the connector performs any additional authentication required on behalf of the user.
5. The connector sends the request to the on-premises application.
6. The response is sent through Application Proxy service and connector to the user.

## Structural integrity and specifics of Application Architecture

There are several components that go into the makeup and design of this application and each is described in more details more.

**Note:** API must be used via Https.

### IDE:

* Visual Studio 2017

### Application:

* **Mid-Tier Communicate with Database**: Entity Framework
* **Application Framework**: .net MVC 4.0
* **MVC** - Model, Viewer, Controller
* **Model**: Represent database entity model and page view state
  + Language: c#
* **Viewer**: Render UI page for client
  + Language: c# and HTML
* **Controller**: Contains logic how to
  + Language: c#
* Front End: HTML, JavaScript, jQuery, Bootstrap
* Backend: c#, SQL
* **Third Party Service**:
  + Google Map API,
  + Uber API,
  + Lyft API

### Server:

* Azure Web Service, IIS

### Summary of JavaScript, Class files, c# HTML, and APIs

*JavaScript* –used for various functionalities on the application page (e.g such as Map APIs, Uber and Lyft APIs)

* **Note**: The SearchAddress.js applies the Google Map (to display the Point locations from A to B given the coordinates and by Geo-Mapping) as well as the Uber and Lyft API to gather/retrieve records from the site in real time according to the search parameter address. The SearchAddress.js also parse this data and returns it to the search page allowing the user to navigate to the correct record and select it for historical tracking purpose and reporting. Some of the additional instrumental JavaScript are labeled below:
* bootstrap.js
* bootstrap.min.js
* jquery-3.3.1.intellisense.js
* jquery-3.3.1.js
* jquery-3.3.1.min.js
* jquery-3.3.1.slim.js
* jquery-3.3.1.slim.min.js
* jquery.validate-vsdoc.js
* jquery.validate.js
* jquery.validate.min.js
* jquery.validate.unobtrusive.js
* jquery.validate.unobtrusive.min.js
* modernizr-2.8.3.js
* SearchAddress.js

*Model Code* – used to define class entity of database model object and represent state of Page

* AccountViewModel.cs
* IdentityModel.cs
* ManageViewModel.cs
* SearchAddressViewModel.cs

*Controller Code* – Used to build the page through code blocks and encodes it to the server side. This was used for the registered and login authentication page, along with the order History page.

* AccountController.cs
* HomeController.cs
* ManageController.cs
* SearchAddressController.cs

*Viewer Code* – Used to render front end UI page

* \_ExternalLoginsListPartial.cshtml
* ConfirmEmail.cshtml
* ExternalLoginConfirmation.cshtml
* ExternalLoginFailure.cshtml
* ForgotPassword.cshtml
* ForgotPasswordConfirmation.cshtml
* Login.cshtml
* OrderHistory.cshtml
* Register.cshtml
* ResetPassword.cshtml
* ResetPasswordConfirmation.cshtml

*Class files* – Methods used to carry out the function calls throughout the application, such as registration and start up page. As well as displaying the Search results returned from the location search results.

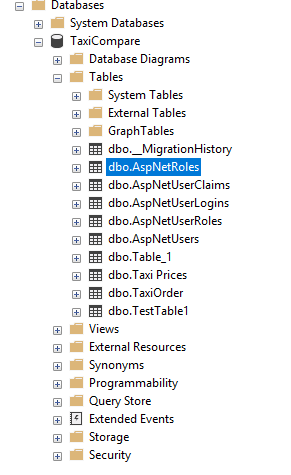
A List of variable Url for each of our desired API calls is listed below, we then passed these variables to a list array that we looped through and JSON parse as objects to display on the search results page.

* var googleFromURL = "https://maps.googleapis.com/maps/api/geocode/json?address=" + AddressFrom + "&key=AIzaSyDZ2b3ucOfoCQrm3eqU8qC-1EKP\_P2Sh2M";
* var googleToURL = "https://maps.googleapis.com/maps/api/geocode/json?address=" + AddressTo + "&key=AIzaSyDZ2b3ucOfoCQrm3eqU8qC-1EKP\_P2Sh2M";
* var googleFromURL = "https://maps.googleapis.com/maps/api/geocode/json?address=" + AddressFrom + "&key=AIzaSyDZ2b3ucOfoCQrm3eqU8qC-1EKP\_P2Sh2M";
* var googleToURL = "https://maps.googleapis.com/maps/api/geocode/json?address=" + AddressTo + "&key=AIzaSyDZ2b3ucOfoCQrm3eqU8qC-1EKP\_P2Sh2M";
* var url = "https://allorigins.me/get?url=" + encodeURIComponent("https://api.lyft.com/v1/cost?start\_lat=" + FromLat + "&start\_lng=" + FromLong + "&end\_lat=" + ToLat + "&end\_lng=" + ToLong + "") + "&callback=?";
* var Uberurl = "https://api.uber.com/v1/estimates/price?start\_latitude=" + FromLat + "&start\_longitude=" + FromLong + "&end\_latitude=" + ToLat + "&end\_longitude=" + ToLong + "&server\_token=tv52A1T0X3I6osqh7zX4b76O\_Usvmth6HQI5QWkp";

## Database Tables

Database Tables used: The Azure cloud connection stores all our user login information in the ASPNetUsers table, whenever a user attempts to login it will authenticate against this database table and retrieve a token to validate. If failed it will prompt the user to retry. Since the password, username and email address for each credential is stored on this local Database, therefore authentication is done through Azure couple with the local DB; new active users can create user profiles as well through the registration page.

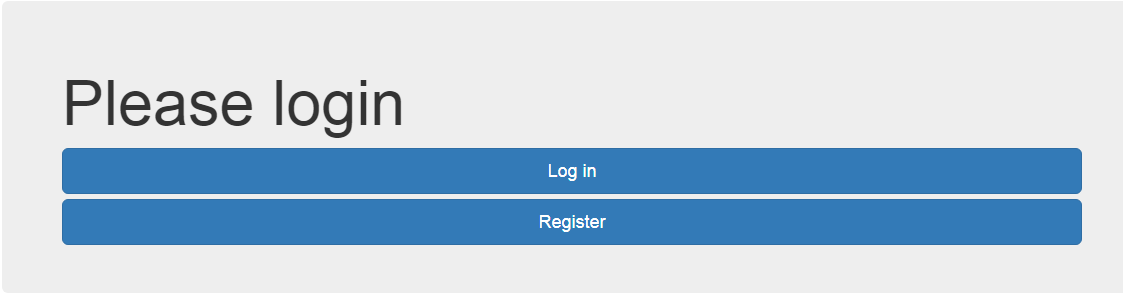
|  |  |
| --- | --- |
| Table Name | Description |
| Dbo.AspNetRoles | Contains all appropriate toles such as Admin |
| Dbo.AspNetUserLogins |  |
| Dbo.AspNetUsers | Contains all the user created accounts for the app |
| Dbo.Taxi Prices | Contains all the selected prices by any user |
| Dbo.TaxiOrder |  |



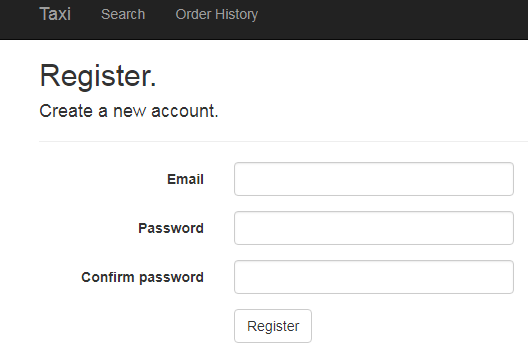
## Start Up Home Page

The Application landing page directs the user to create an account by registering or login with an existing user name and password.

### Page – Requesting Login

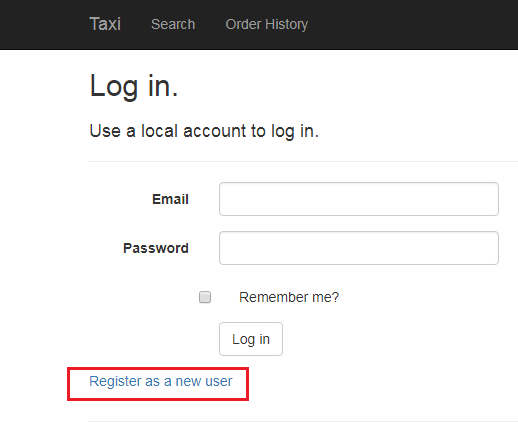


## Page – Register/Create New Account for Login –



## Login Start up Page

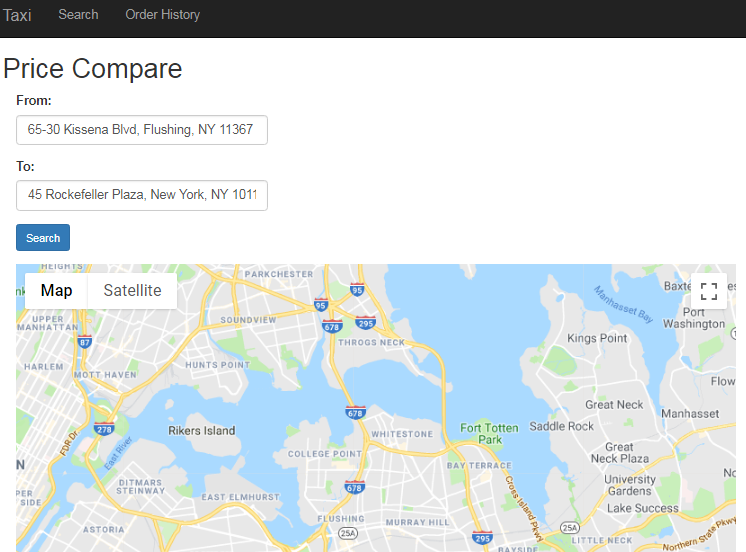
Login Page display published using Azure cloud authentication:



### Fields

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Display Name** | **Database Name** | **Data Type** | **Required** | **Properties** |
| Email | Email: | Single line of text  (Mandatory) | Yes | Max length: 25 |
| Password | Password: | Single line of text  (Mandatory) | Yes | Max length: 25 |

### Page – Taxi Price Compare Form – Displays the field locations and the Google API that allows navigations

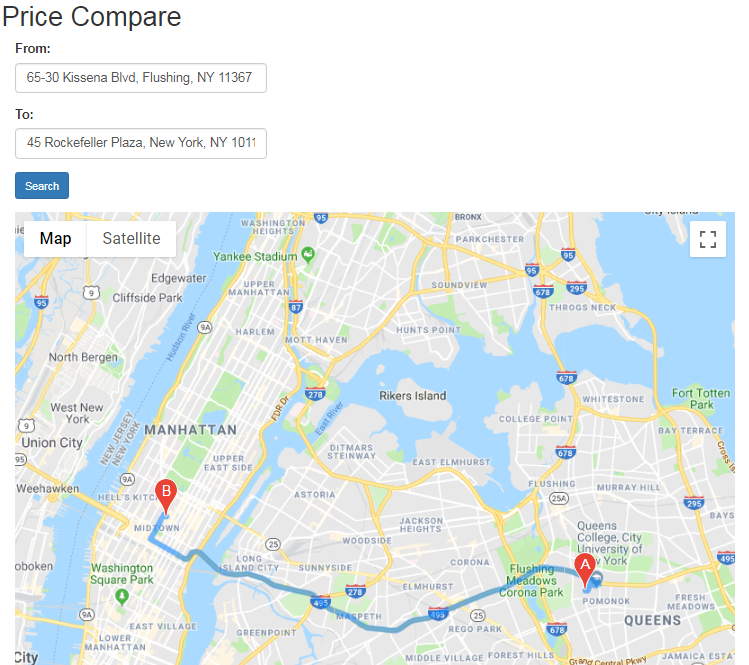


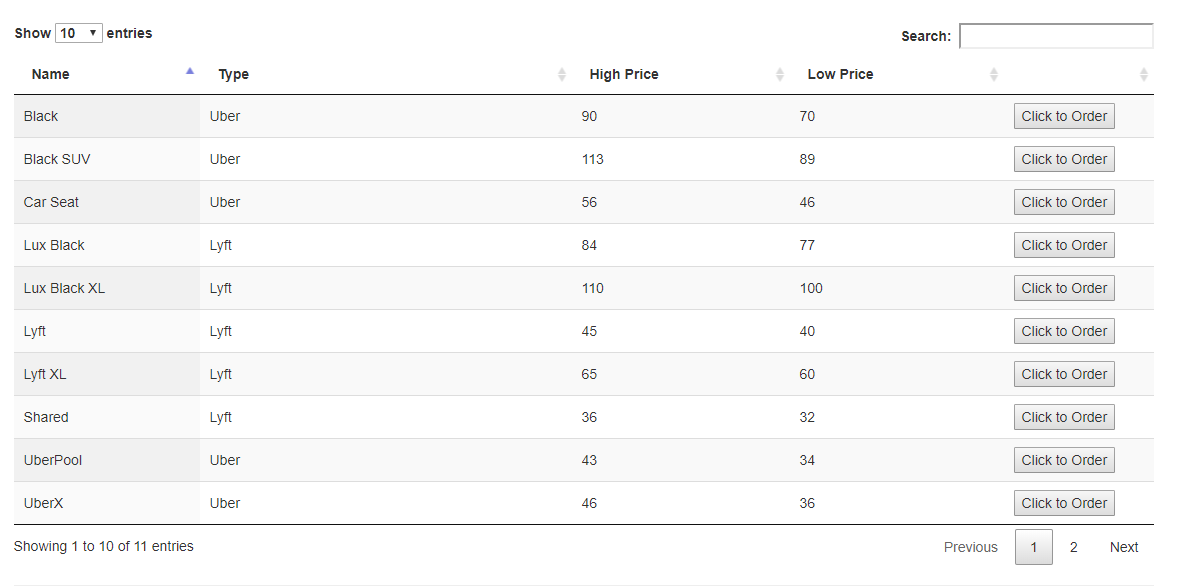
### Fields

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Display Name** | **Schema Name** | **Data Type** | **Required** | **Properties** |
| From | From: | Single line of text  (Mandatory) | Yes | Max length: 45 |
| To | To: | Single line of text  (Mandatory) | Yes | Max length: 45 |
| Search | Search | Single line of text  (Optional) | NO | Max length: Max |

### Page – Search return Page – that maps the location path on the Map display from Points ***A*** to Point ***B***

The User is able to manually write in the from and to address for travel or filter through the API Map for search results.





The Page displays the full results of the prices for the location between points A and B and displays the price for each available type of Taxi. The user can navigate through pages selecting the next marker at lower right hand of the page, the user can also use the search window that allows them to filter their search criteria.

### Page – Order History – that maps the location path on the Map display from Points ***A*** to Point ***B***

Once the user clicks on the ‘***Click to Order’*** button, the application will display a successful response and the user can navigate to the order history page to view all recent transactions under their profile. The User can use this information for reporting purposes.

