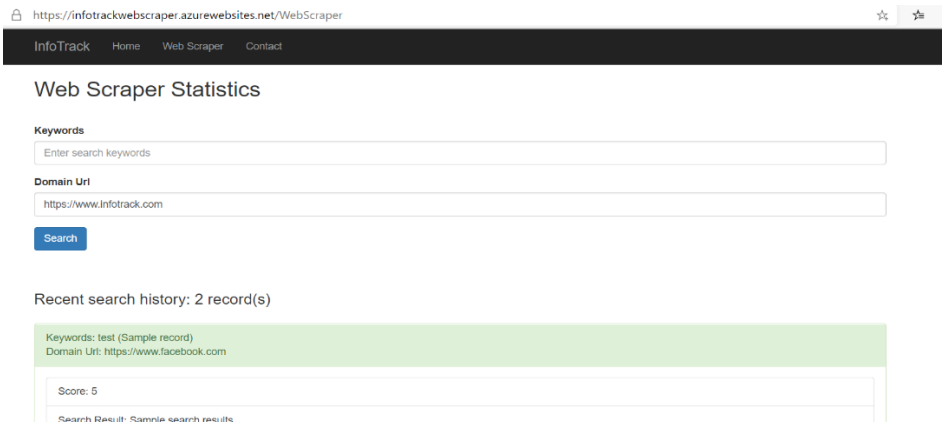
Project Documentation: InfoTrack Web Scrapper

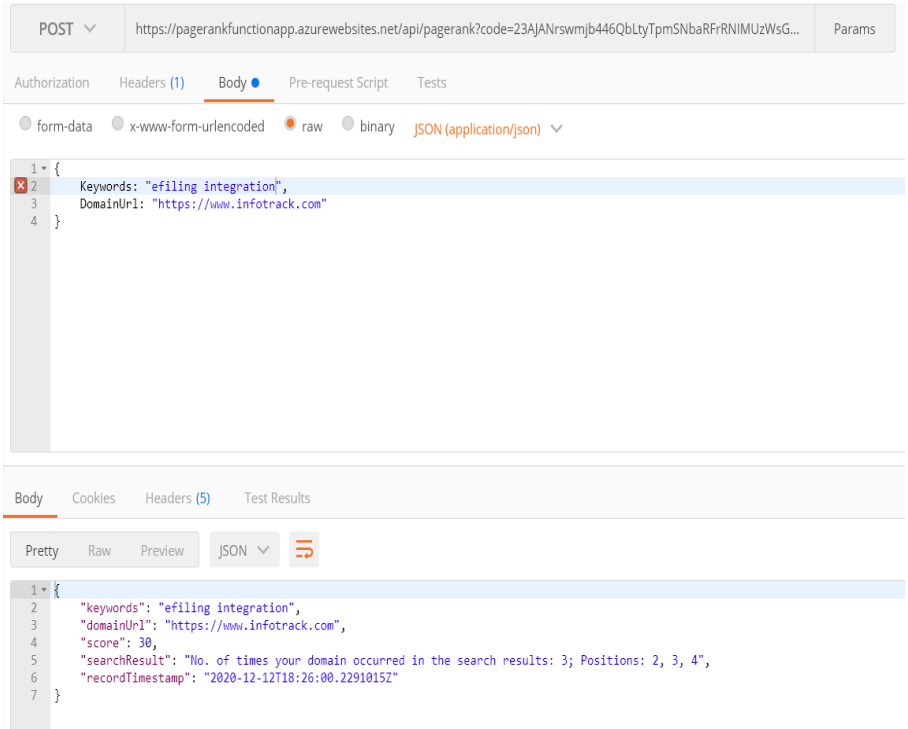
ASP.NET MVC Web Application: .Net Framework: 4.7.2 (Razor, Bootstrap, ASP.NET MVC 5.2)

https://infotrackwebscraper.azurewebsites.net/



Azure Function App - .Net Core 3.1

https://pagerankfunctionapp.azurewebsites.net/api/pagerank?code=23AJANrswmjb446QbLtyTpmSNbaRFrRNIMUzWsGzyf30qjwO37AiuQ==



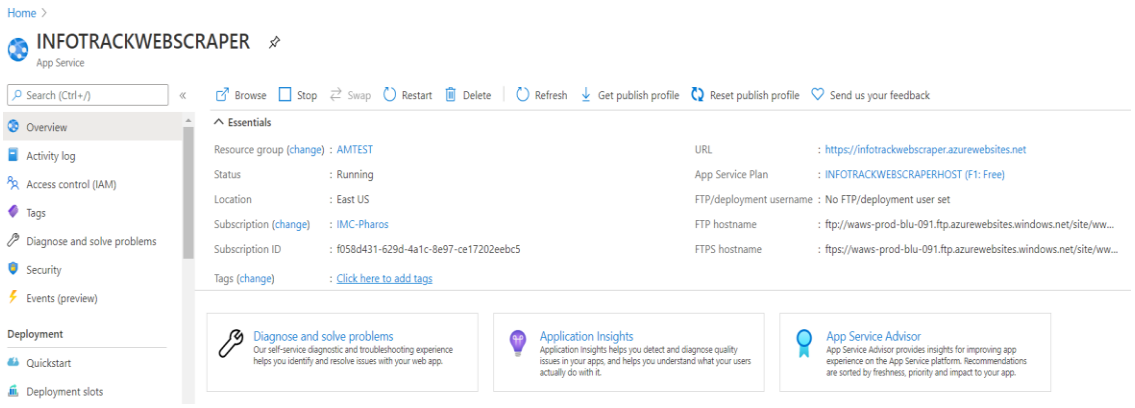
Key Technical features

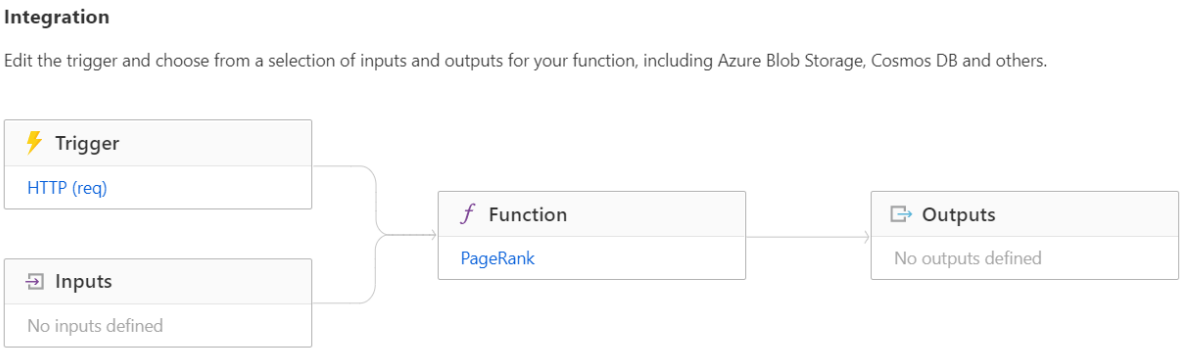
Microservice Architecture

1. Designed a serverless microservice architecture using Azure functions and deployed it on Azure Cloud Platform. The MVC app is deployed on a free app service plan, the initial load will be slow due to cold start, but it can be configured to avoid this if moved to paid version. The function app is deployed on a consumption plan and it can scale up based on the web traffic. The core logic to compute the search results is written in the function app so that the back-end logic can be substituted as we implement a more complex logic using Google or third party APIs without the need for the front end to change.

   Scope for improvements:

a. Create deployment slots for different development environments (dev, staging and production).
b. We can implement CI/CD pipeline for continuous delivery and testing.
c. Avoid cold starts by moving to a paid production environment.
d. Setup custom domains (DNS zones can be setup on 1 & 1 or Cloudflare) and secure the web channel using SSL certificates.
e. Secure the MVC app by using third party platforms like Cloudflare, this will avoid features like Bot Management and also help us with caching.
f. The function app provides a basic http end point security using function keys but can be more robust by adding the function behind an Azure API Management service + JWT token based restriction.

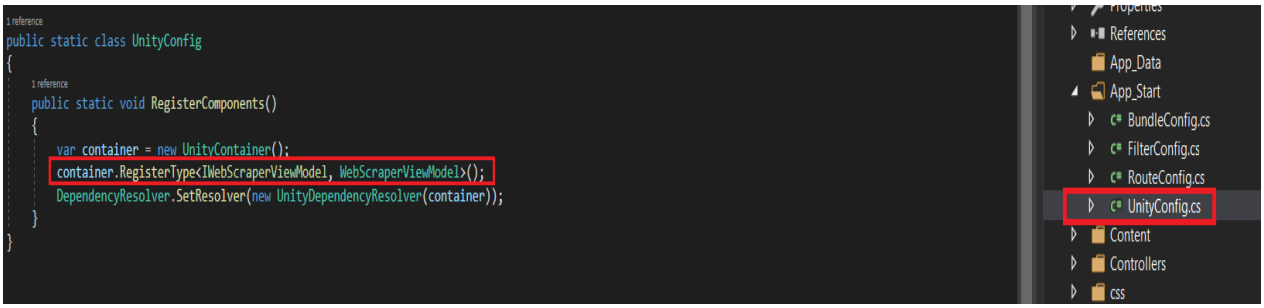




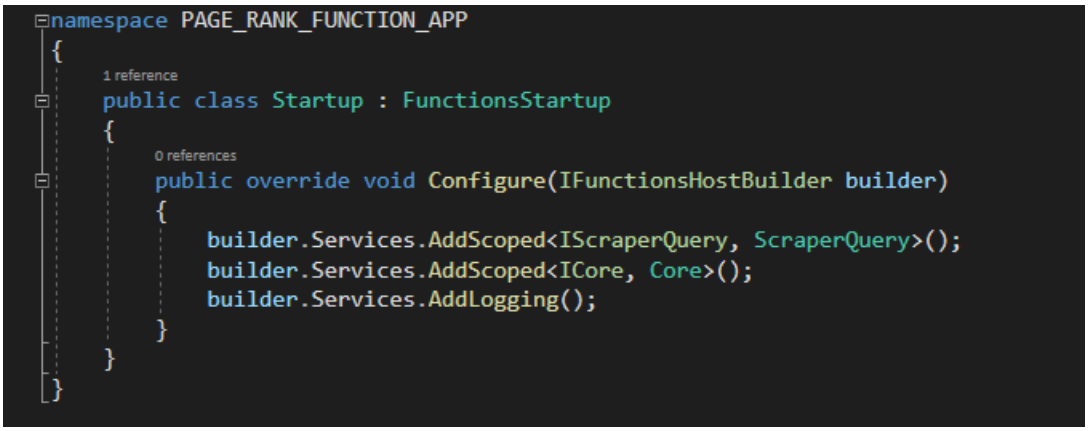
Dependency Injection

Implemented Unity container which is an open source IoC container for .NET applications supported by Microsoft.

MVC App:



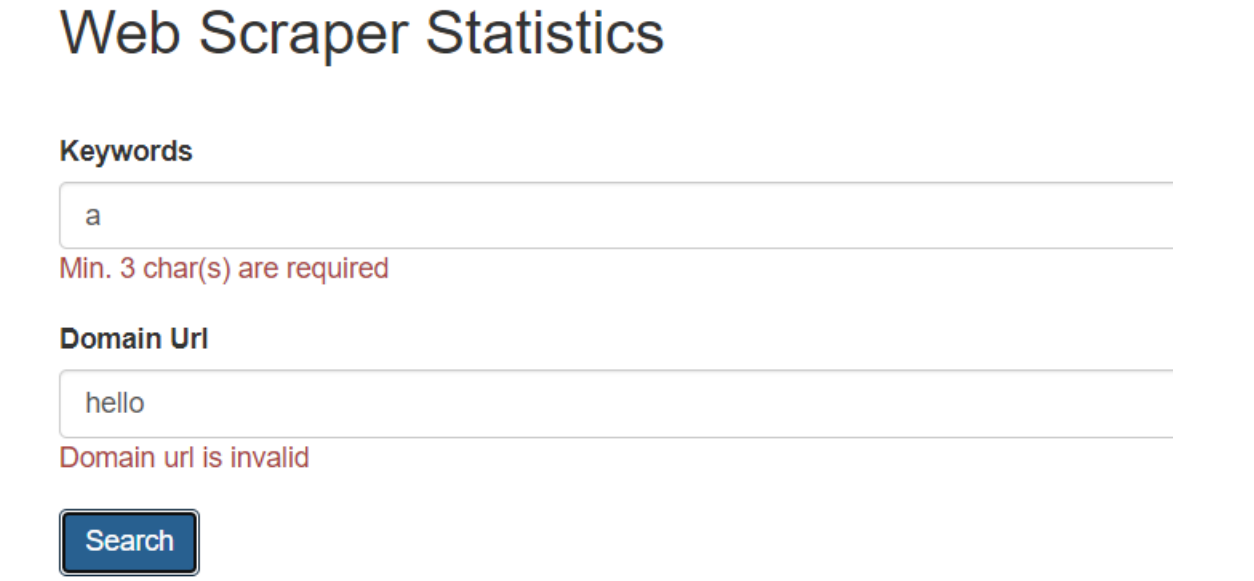
Function App: In-built IOC provided by .Net Core



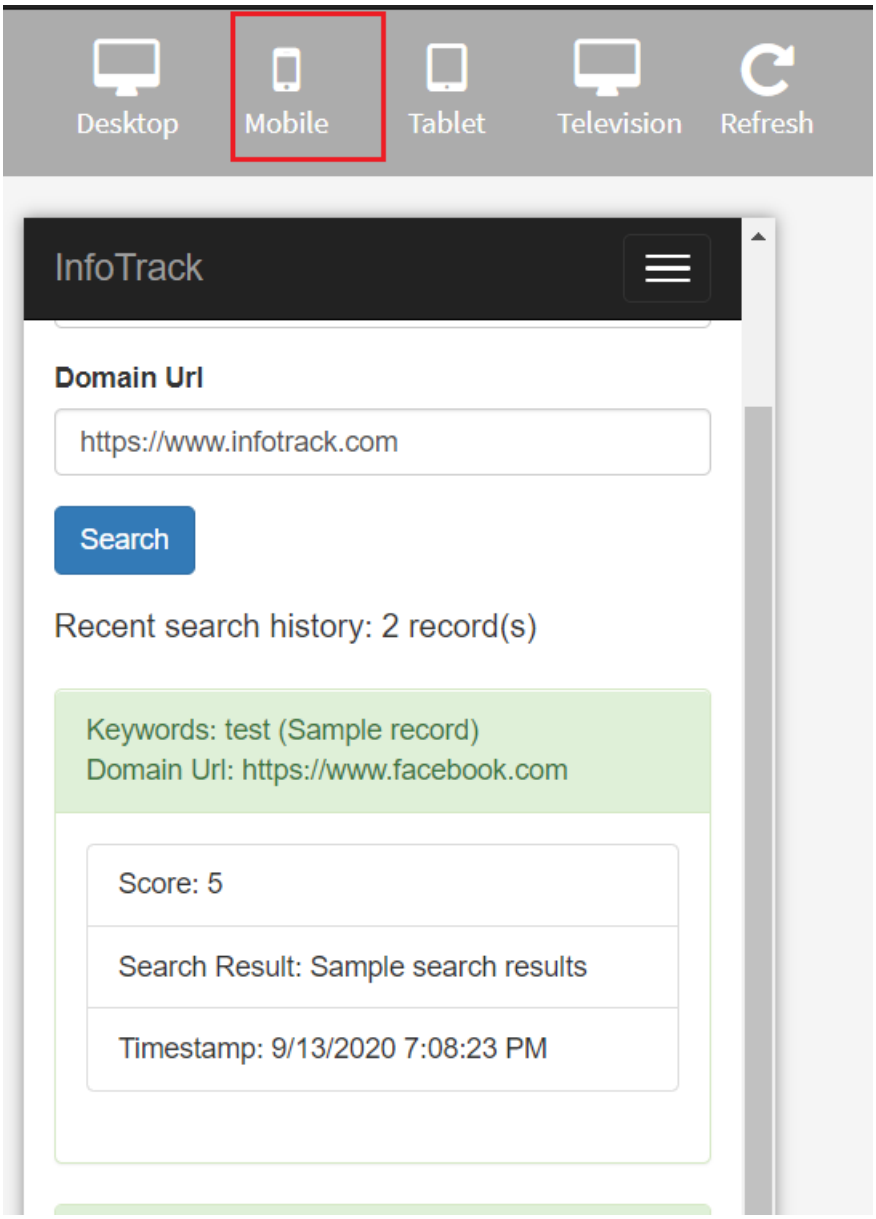
Client-Side Validation

Implemented model validation Using Data Annotations in ASP.NET MVC

```
namespace INFO_TRACK_WEB_SCRAPER.ViewModels
{
    /// <summary>
    /// The view model is an abstraction of the view 'Statistics' exposing public properties and commands.
    /// </summary>
    3 references
    public class WebScraperViewModel : IWebScraperViewModel
    {
        [Required(ErrorMessage = "Keywords are required")]
        [StringLength(maximumLength: 4000, MinimumLength = 3, ErrorMessage = "Min. 3 char(s) are required")]
        3 references
        public string Keywords { get; set; }

        [Required(ErrorMessage = "Domain url is required")]
        [Url(ErrorMessage = "Domain url is invalid")]
        [StringLength(maximumLength: 4000, MinimumLength = 5, ErrorMessage = "Min. (5) - Max. (4000) length")]
        3 references
        public string DomainUrl { get; set; }
```



Responsive web design using Bootstrap



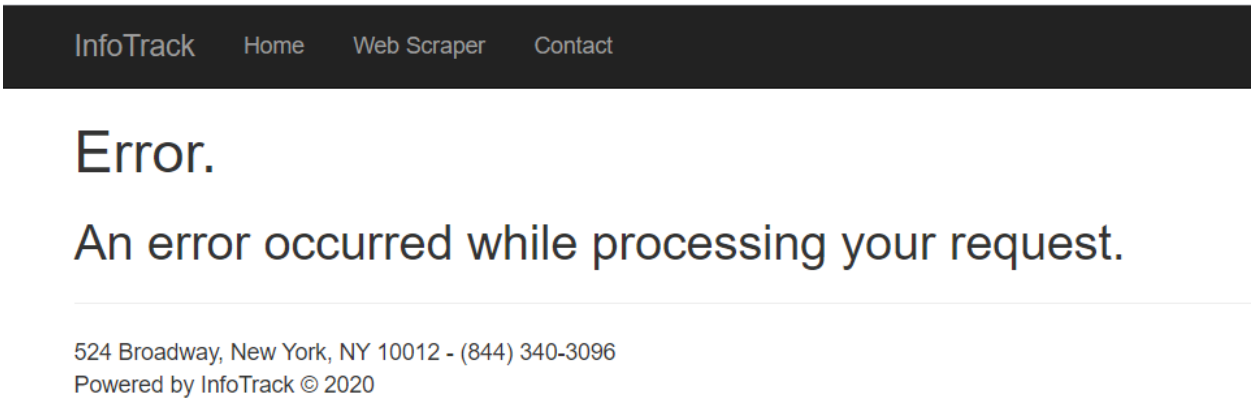
Error handling

MVC App: Implemented a generic base controller for handling errors and redirecting the user to a custom error page.

```
1 reference
public class InfoTrackBaseController : Controller
{
    // GET: InfoTrackBase
    1 reference
    public virtual ActionResult Index()
    {
        return View();
    }

    /// <summary>
    /// Called when an unhandled exception occurs in the action.
    /// </summary>
    /// <param name="objectExceptionContext">Information about the current request and action.</param>
    0 references
    protected override void OnException(ExceptionContext objectExceptionContext)
    {
        //Record exception in a persistent system (MongoDB).

        //Redirect to custom error page.
        objectExceptionContext.ExceptionHandled = true;
        objectExceptionContext.Result = new ViewResult
        {
            ViewName = "~/Views/Shared/Error.cshtml"
        };
    }
}
```

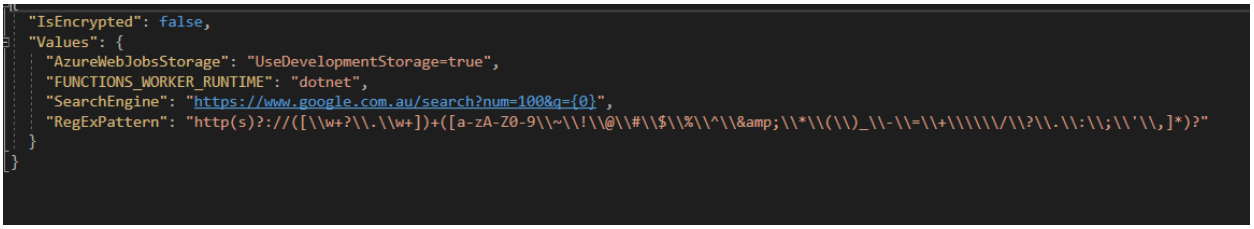
We can handle errors globally

```
public class MvcApplication : System.Web.HttpApplication
{
    0 references
    protected void Application_Start()
    {
        UnityConfig.RegisterComponents();
        AreaRegistration.RegisterAllAreas();
        FilterConfig.RegisterGlobalFilters(GlobalFilters.Filters);
        RouteConfig.RegisterRoutes(RouteTable.Routes);
        BundleConfig.RegisterBundles(BundleTable.Bundles);
    }

    /// <summary>
    /// Log exceptions globally.
    /// </summary>
    0 references
    protected void Application_Error(object sender, EventArgs e)
    {
        //Record exception in a persistent system (MongoDB).
        Exception objectException = Server.GetLastError();
    }
}
```



Core Logic

The function app uses https://www.google.com.au/search?num=100&q={0} as the primary search engine to lookup the keywords and the domain given domain url for matches based on the below regular expression pattern. I have also implemented a simple scoring model to award extra points if the given domain appears at the top of the list.



```
string stringSearchEngine = Environment.GetEnvironmentVariable("SearchEngine");
string stringKeywords = string.Format(stringSearchEngine, HttpUtility.UrlEncode(objectOfScraperQuery.Keywords));
Uri objectOfUri = new Uri(objectOfScraperQuery.DomainUrl);

int intOccurrenceCount = 0;
List<int> listOfMatchedPositions = new List<int>();

HttpWebRequest objectOfHttpWebRequest = (HttpWebRequest)WebRequest.Create(stringKeywords);
using (HttpWebResponse objectOfHttpWebResponse = (HttpWebResponse)objectOfHttpWebRequest.GetResponse())
{
    using (StreamReader objectOfStreamReader = new StreamReader(objectOfHttpWebResponse.GetResponseStream(), Encoding.ASCII))
    {
        string stringHtmlResponse = objectOfStreamReader.ReadToEnd();
        string stringRegEx = Environment.GetEnvironmentVariable("RegExPattern");

        MatchCollection objectOfMatchCollection = Regex.Matches(stringHtmlResponse, stringRegEx);
        for (int intLoopCounter = 0; intLoopCounter < objectOfMatchCollection.Count; intLoopCounter++)
        {
            try
            {
                string match = objectOfMatchCollection[intLoopCounter].Groups[0].Value;
                if (match.Contains(objectOfUri.Host))
                {
                    intOccurrenceCount++;
                    listOfMatchedPositions.Add(intLoopCounter + 1);
                }
            }
            catch (Exception)
            {

            }
        }
    }
}
objectOfScraperQuery.SearchResult = GetFormattedSearchResults(intOccurrenceCount, listOfMatchedPositions);
objectOfScraperQuery.Score = GetScoreCount(listOfMatchedPositions);
```

Property Validations Using Getter and Setters

```
private List<ScraperQuery> _ListOfScraperQueries;
4 references
public List<ScraperQuery> ListOfScraperQueries
{
    get
    {
        if (this._ListOfScraperQueries is null || _ListOfScraperQueries.Count == 0)
        {
            this._ListOfScraperQueries = GetDefaultScraperQueries();
        }
        return this._ListOfScraperQueries;
    }

    set
    {
        if (value != null)
        {
            foreach (ScraperQuery objectCurrentScraperQuery in value)
            {
                if (!string.IsNullOrEmpty(objectCurrentScraperQuery.Keywords) && !string.IsNullOrEmpty(objectCurrentScraperQuery.DomainUrl))
                {
                    this._ListOfScraperQueries.Add(objectCurrentScraperQuery);
                }
            }
        }

        if (this._ListOfScraperQueries is null || this._ListOfScraperQueries.Count == 0)
        {
            this._ListOfScraperQueries = GetDefaultScraperQueries();
        }
    }
}
```

```
private int GetScoreCount(List<int> listOfMatchedPositions)
{
    int intScoreCount = 0;
    try
    {
        foreach (int intPosition in listOfMatchedPositions)
        {
            switch (intPosition)
            {
                case int n when (n <= 10):
                    intScoreCount += 10;
                    break;

                case int n when (n > 10 && n <= 20):
                    intScoreCount += 8;
                    break;

                case int n when (n > 20 && n <= 30):
                    intScoreCount += 6;
                    break;

                case int n when (n > 30 && n <= 40):
                    intScoreCount += 4;
                    break;

                case int n when (n > 40 && n <= 50):
                    intScoreCount += 2;
                    break;

                case int n when (n > 50):
                    intScoreCount += 1;
                    break;
            }
        }
    }
    catch (Exception)
    {

    }
```

Sample Data: Maintains recent search history

Sorted by the most recent search in descending order.

Recent search history: 2 record(s)

Keywords: test (Sample record)
Domain Url: https://www.facebook.com

Score: 5

Search Result: Sample search results

Timestamp: 9/13/2020 7:58:29 PM

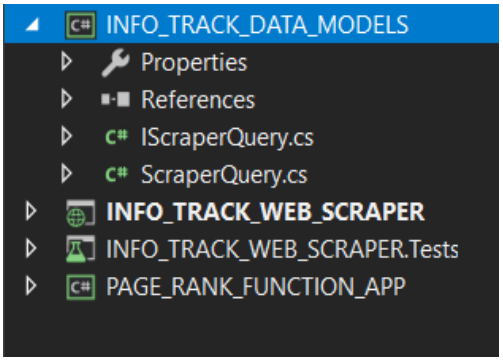
Keywords: efiling integration (Sample record)
Domain Url: https://www.infotrack.com

Score: 10

Search Result: Sample search results

Timestamp: 6/15/2020 7:58:29 PM

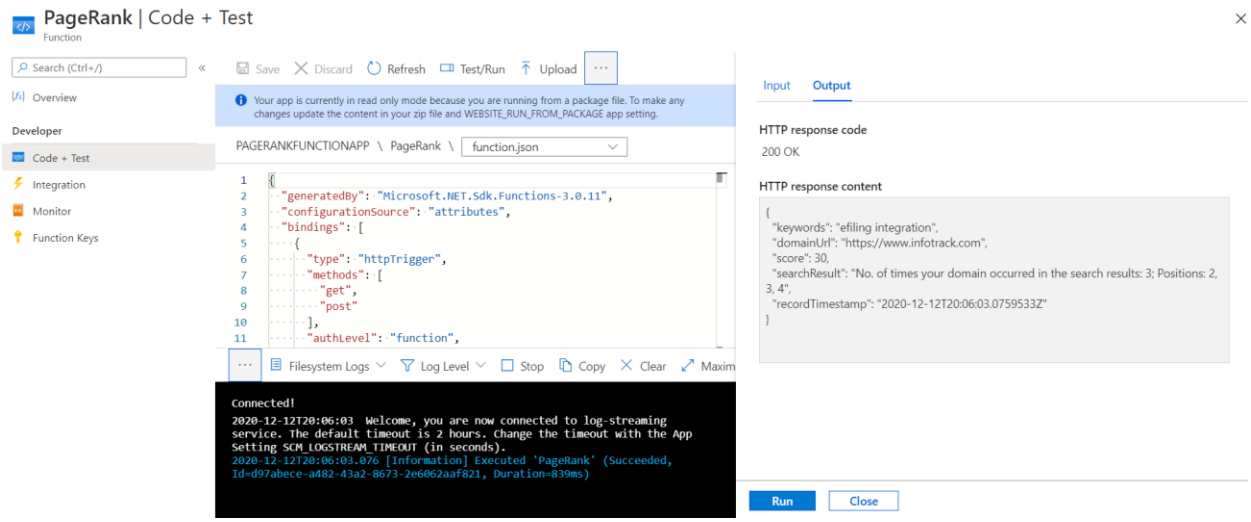
Separate reusable Data Models class library to share objects between the MVC and Function app.



Bundling JavaScript and CSS files to prevent multiple get requests to the server.

```
1 reference
public class BundleConfig
{
    // For more information on bundling, visit https://go.microsoft.com/fwlink/?LinkId=301862
    1 reference
    public static void RegisterBundles(BundleCollection bundles)
    {
        bundles.Add(new ScriptBundle("~/bundles/jquery").Include(
                    "~/Scripts/jquery-{version}.js"));

        bundles.Add(new ScriptBundle("~/bundles/jqueryval").Include(
                    "~/Scripts/jquery.validate*"));

        // Use the development version of Modernizr to develop with and learn from. Then, when you're
        // ready for production, use the build tool at https://modernizr.com to pick only the tests you need.
        bundles.Add(new ScriptBundle("~/bundles/modernizr").Include(
                    "~/Scripts/modernizr-*"));

        bundles.Add(new ScriptBundle("~/bundles/bootstrap").Include(
                  "~/Scripts/bootstrap.js"));

        bundles.Add(new StyleBundle("~/Content/css").Include(
                  "~/Content/bootstrap.css",
                  "~/Content/Site.css"));
    }
}
```

Verified Function app on the Azure cloud platform as well as Postman

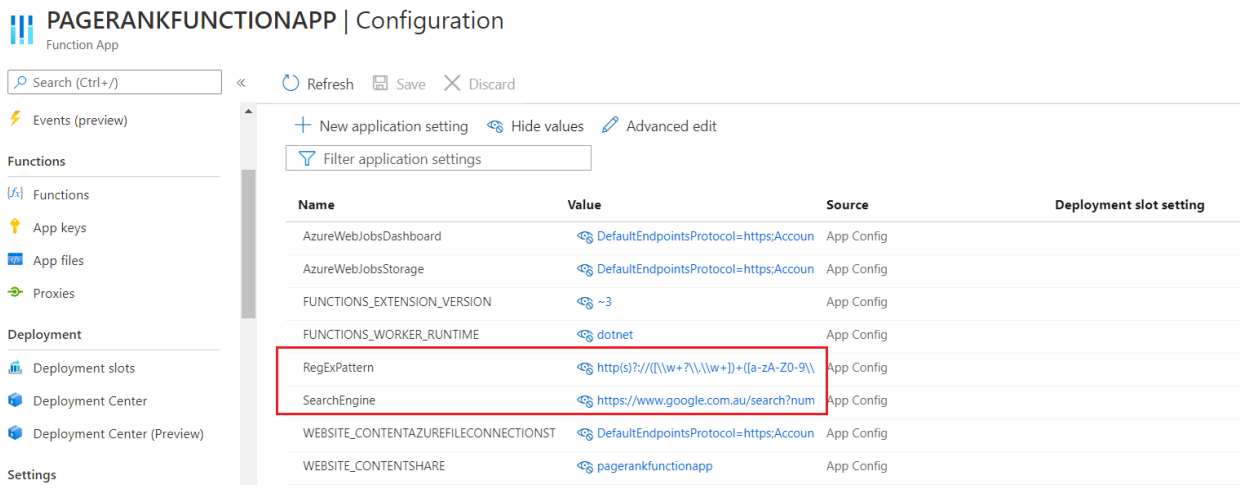


Easy to switch search engine provider and pattern matching by making it config based

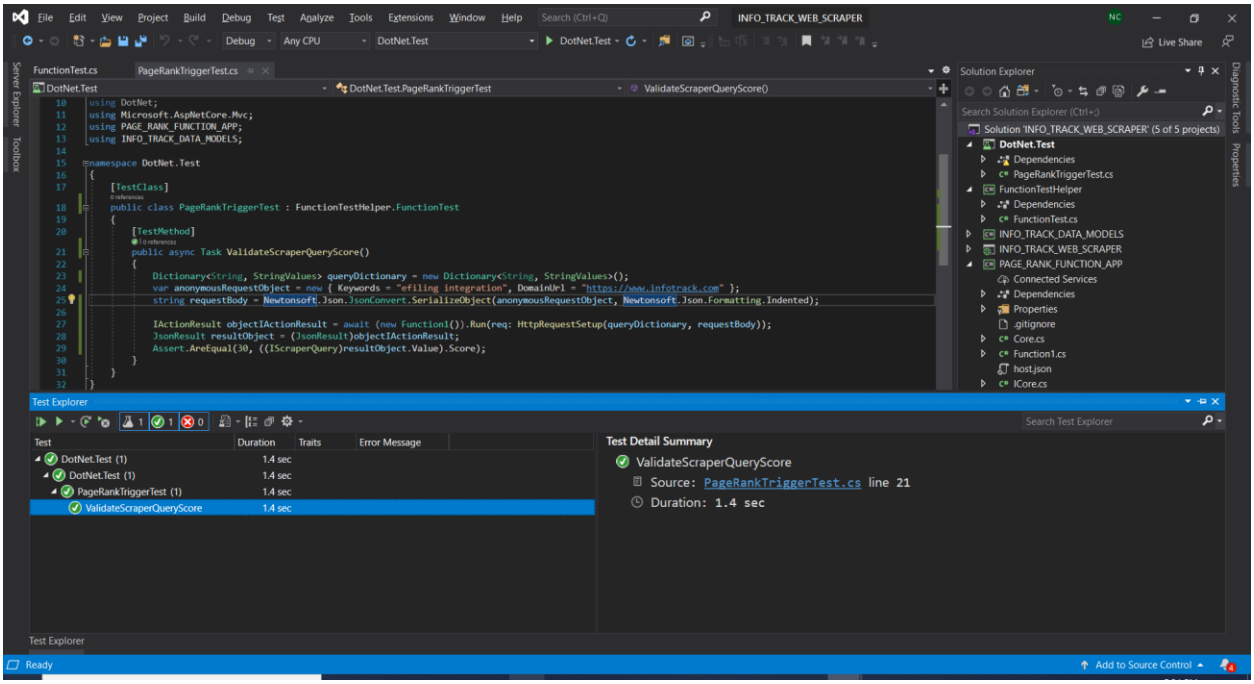
local.settings.json

Schema: https://json.schemastore.org/local.settings

```
{
  "IsEncrypted": false,
  "Values": {
    "AzureWebJobsStorage": "UseDevelopmentStorage=true",
    "FUNCTIONS_WORKER_RUNTIME": "dotnet",
    "SearchEngine": "https://www.google.com.au/search?num=100&q={0}",
    "RegExPattern": "http(s)?://([\\w+?\\.\\w+])+([a-zA-Z0-9\\~\\!\\@\\#\\$\\%\\^\\&amp;\\*\\(\\)_\\-\\=\\+\\\\\\|\?\\.\\:\\;\\'\\,]*)?"
  }
}
```



Unit Tests



Mocked http request to debug the Azure function