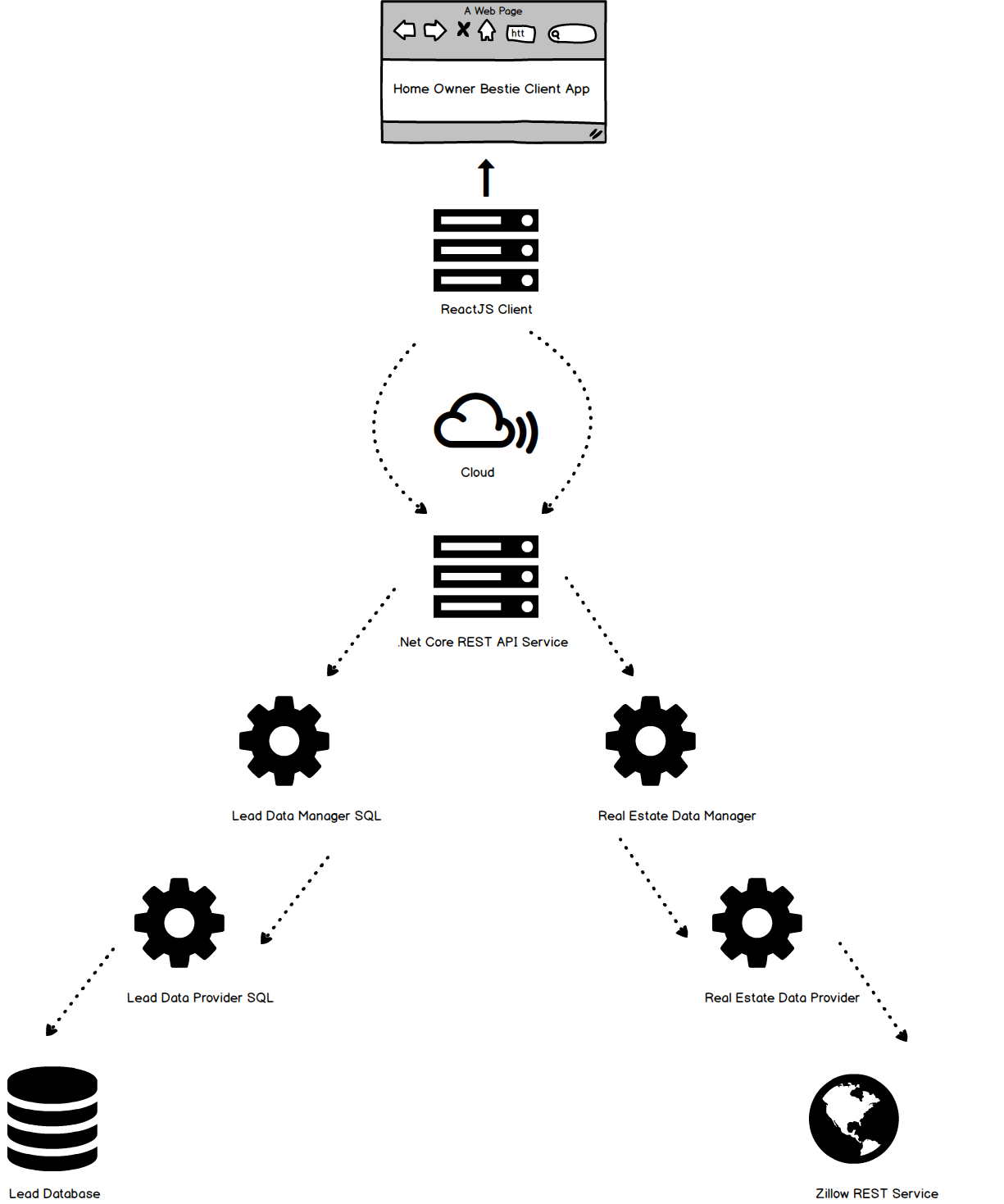
**Home Owner Bestie - Technical Spec**

**App Name:** Home Owner Bestie (HOB)

**Purpose of the App:** This is just a demo app to illustrate the React JS (front-end) and .Net Core (Back end)

**Demo Scenario:** the app shows how home owners can take help of app to get the ballpark on their home selling, renting values. The app will rely on the HOB back end REST service which wraps Zillow REST APIs to get real estate data and also to log the user activity to a database.

**Architecture Schematic**



**Server Architecture**

* I used .Net Core 2.2 and Entity Core for the design
* The service has several modules which are independent of each other. As you can see from the diagram above, it has 3 components
  + Lead Data – wrapper around Serves Zillow API service
    - Lead Data Manager – Take decision of how the request should be handled
    - Lead Data Provider – Just serves the data and nothing more
  + Real Estate Data – module to generate user activity and lead data
    - Real Estate Data Manager – Ditto as above
    - Real Estate Data Provider – Ditto as above
    - I used MSSQL (SQL Express) with Entity Framework core to work with data.
  + Service Layer – Main solution and the REST API layer where all the startup .Net Core things happen.
  + All these modules get their configuration from the appsettings.json file. With the dependency injections, the .Net core will pass on IConfiguration, DBContext to all the modules giving them a chance to store the configs such as data connection, api urls etc.
* The main module service which is exposed as REST used above modules to serve clients. With this architecture, its more organized and extensible. If there are new requirements for the app such as to get data from other real estate services such as Craiglist, Home Junction etc., then this can easily done as long as the modules implement all classes for the Real Estate interfaces.
* The abstraction is done to a level where the service will instantiate the Real Estate and Lead Data assemblies at the runtime in the ConfigureService method inside the StartUp class.

public void ConfigureServices(IServiceCollection services)

{

var CurrentPath = System.AppDomain.CurrentDomain.BaseDirectory;

var realEstateAssembly = AssemblyLoadContext.Default.LoadFromAssemblyPath(CurrentPath + Configuration["RealEstateData:Assembly"]);

var leadDataAssembly = AssemblyLoadContext.Default.LoadFromAssemblyPath(CurrentPath + Configuration["LeadData:Assembly"]);

services

.AddTransient(typeof(IRealEstateDataProvider), realEstateAssembly.GetType(Configuration["RealEstateData:DataProvider:Implementation"]))

.AddTransient(typeof(IRealEstateDataManager), realEstateAssembly.GetType(Configuration["RealEstateData:DataManager:Implementation"]))

.AddTransient(typeof(ILeadDataManager), leadDataAssembly.GetType(Configuration["LeadData:DataManager:Implementation"]))

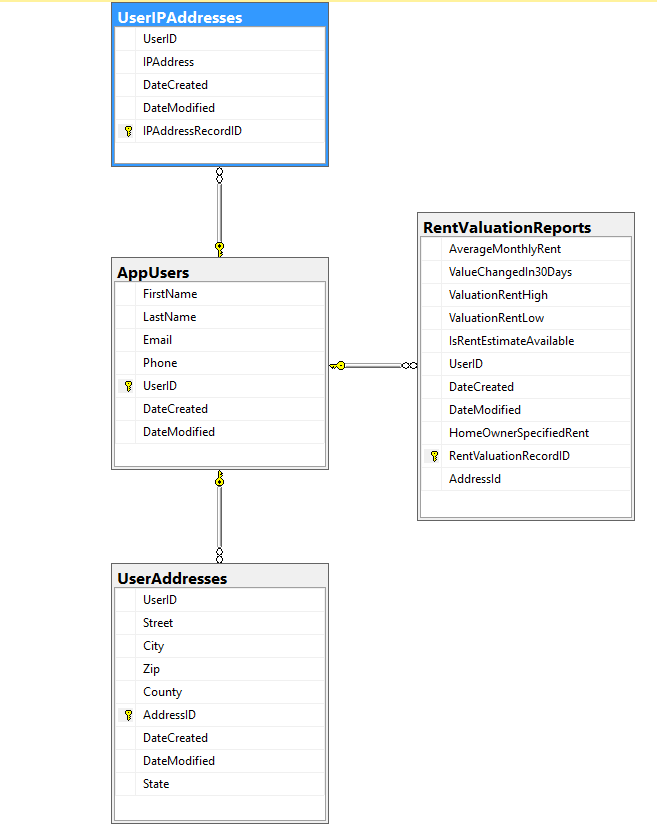
.AddTransient(typeof(ILeadDataProvider), leadDataAssembly.GetType(Configuration["LeadData:DataProvider:Implementation"]));

* Dependency injections interfaces
  + IRealEstateDataProvider
  + IRealEstateDataManager
  + ILeadDataManager
  + ILeadDataProvider
* We have two controllers in the Service
  + HOBUserController
    - POST - <https://homeownerbestieservice20190902104333.azurewebsites.net/appuser>
  + RentValuationController
    - POST - <https://homeownerbestieservice20190902104333.azurewebsites.net/rentvaluation/emailme>
    - POST - <https://homeownerbestieservice20190902104333.azurewebsites.net/rentvaluation>
* Hosting
  + Both the service and database is hosted on Microsoft Azure.
  + The current url is [https://homeownerbestieservice20190902104333.azurewebsites.net](https://homeownerbestieservice20190902104333.azurewebsites.net/)

**Client App Architecture**

* Developed using ReactJS and React Bootstrap
* The app has the following components
  + Get Started
  + Sign Up
  + Header
  + Navbar
  + Coming Soon
  + Protected Route – Custom Component to route to signup screen if the user is not signed up already.
  + Valuate Home Rent
* Data Service Layer
  + Components rely on the Data service for sending/fetching information to/from outside.
  + Data Service Layer internally has **Local Data Service** (LDS) and **Server Data Service (SDS)**. Both share the same parent class Generic Data Service. They are glued by chain of responsibility Adapter ‘AdapterHOBDataService’. Components only know this adapter. The adapter will just delegate the requests one by one in the chain and the concrete data services will know how to handle this. Ex: for the request SetUserDetails, LDS will set user cookies and SDS will make a request to server (REST Service) to actually signup the user into a persistent storage.
  + Check dataservice.js for more details.
* Framework features used
  + Promise
  + Async/Await – for waiting till the completion of the required operation.
* Third Party Libraries used
  + React Bootstrap – For responsive UI - <https://react-bootstrap.github.io/>
  + Axios – Made API requests easy - <https://www.npmjs.com/package/axios>
  + Yup – For form field data validation - <https://github.com/jquense/yup>
  + React Router v4 – For app links navigation - <https://reacttraining.com/react-router/>
  + Formik – For form validation - <https://jaredpalmer.com/formik/>
  + Toasted Notes – for in app alerts - <https://toasted-notes.netlify.com/>
  + Intl – For formatting numbers, dollar amounts - <https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Intl>
  + Google Geo Code & React Places Autocomplete – for auto suggest and searching addresses - <https://hibiken.github.io/react-places-autocomplete/>
  + Universal Cookies – Ease of use of working with cookies - <https://www.npmjs.com/package/universal-cookie>
  + Google Analytics – to track user activities
* Hosting
  + Hosted on Google Firebase hosting platform.
  + The current app url is  [https://home-owner-bestie-app-d881c.firebaseapp.com](https://home-owner-bestie-app-d881c.firebaseapp.com/)

**Database Schema**



**Limitations**

1. The app is currently tested on Google Chrome browser only. It may behave differently on other browsers.

**Scope for improvements**

1. App - Component – ValuationHomeRent – Too much going on here. May want to break into separate components and have them serve one purpose.