Smart Apartment Data

Technical Assessment

Congratulations, you’ve made it to the second phase of our screening process.  We know your time is

valuable, so our objective is to advance this process forward as quickly and efficiently as possible.

Please complete this assignment within 1 week. If you need more time do let us.

know.

Background  
Search is a critical component of our Big Data infrastructure at Smart. ElasticSearch is an incredibly popular search technology and the skills you will learn in this assessment will likely be very valuable. Since we utilize ElasticSearch heavily, this will give you great exposure to one of the technologies you’ll be working with .

In this assessment you will be build a scalable search functionality using AWS ElasticSearch. You will need to setup a free trial account AWS and use their “free-tier” elastic instance. This will ensure you’re not charged for anything.

You will be provided basic JSON data files that need to be indexed (uploaded). Please be sure to watch the DEMO video included to help you understand the data you are working with. Your Search REST API needs to accept an input string and output the most relevant records that match.

Your Goal

Your goal is to demonstrate that you’re able to learn Elastic well, by reading documentation and implementing it correctly. We invite you to educate yourself on elastic search so you’re using good practices. And second, to create an enterprise solution. As such, be sure to break up your solution into separate projects, modules, etc. Though there will not be a lot of code you need to write, it is important that you structure the solution well.

We are not so much interested in “making it work”. We already know you’re a capable developer. Rather, we want to measure the quality of your design and quality of your code. For instance, we evaluate

“How is dependency injection being used?”

“Is there correct use of any design patterns?”  
“Is the code refactored, and readable”

Let’s get started…

Search Input

search phase: (required, string)   
market: (optional, string)

Limit: (int, default: 25, max how many results to be returned)

Feel free to include other optional input fields you deem necessary.

The assessment folder contains two files ‘properties.json’ and ‘mgmt.json’. properties.json represents apartment buildings. Mgmt.json represents the management companies that own the apartment buildings. The query string will need to be applied to only string fields in both files.

Schema for properties.json:

[  
 {  
 "property": {  
 "propertyID": 70034, //int  
 "name": "Sage at 1825 Place", //string  
 "formerName": "1825 Place 2", //string  
 "streetAddress": "15835 Foothill Farms Loop", //string  
 "city": "Pflugerville", //string  
 "market": "Austin", //string  
 "state": "TX", //string  
 "lat": 3.044956000000000e+001, //float  
 "lng": -9.765073000000000e+001 //float  
 }  
 }  
]

Schema for mgmt.json

[  
 {  
 "mgmt": {  
 "mgmtID": 27918, //int  
 "name": "Essex Property Trust AKA Essex Apartment Homes", //string  
 "market": "San Francisco", //string  
 "state": "CA" //string  
 }  
 }  
]

Note: in the United States addresses are divided into several portions.

Street Address (both the street number and name example: 100 main street)

City (example: “New York City”, “Los Angeles”)

State (2 characters code such as “NY” for New York and “CA” for California)

*Market*: this is an important field. It acts as the scope for the search. When ‘market’ is passed in, results should only be returned which match the markets (scope).

Note: multiple markets can be passed in such as *List<string> { ‘California’ , ‘South California’, ‘Seattle’}.* If no Market is specified, all markets should be searched and treated the same.

Output:  
Your REST API should be written in c# 5 and return the most relevant search results. ElasticSearch has many different types of fulltext search options. As such, you need to select the correct one. The user will enter a phrase that contains stop words like “and”, “or”, “into” “the”. A substring match is a dead-end because stop-words will be matched and hence results will not be returned correctly. For example: when the user types “stones and rocks” the search should return results that match “stone rocks apartments”. Notice: The user mistyped stones (extra s) , added stop word “and”.

What success looks like:

We would like to see your implementation of auto complete search using various built-in or custom analyzers, tokenizers, filters etc.Please do not copy/paste code from any website. Doing so will automatically disqualify your assessment. It is your code, and your design that we’re interested in ☺

Submission:

Please record a 5-10-minute video showing your implementation starting with the code to upload the data to a test application to query the cloud-search function.

You RET API should be developed in .NET core and upload your video as well as the code to Google Drive or any cloud-based drive and email this to us. If you’ve developed a program to upload the JSON files to the cloud be sure to include that as well. In short: upload 100% of your code into github for us to review.

Please email a link to github to [hr@smartapartmentdata.com](mailto:hr@smartapartmentdata.com) and [leo@smartapartmentdata.com](mailto:leo@smartapartmentdata.com)