Fora Coding Challenge

Overview

Fora Financial is a FinTech company that offers loans to small businesses. We've created a take-home “coding challenge” to both see how you creatively solve problems as well as act as a conversation point during a potential future interview phase. Feel free to solve the problem as you see fit so long as you use C# and .NET. Also, we want to be as respectful of your time as we can be, so please don't spend more than 3-4 hours on this. Please submit your code to our recruiter in a .zip file. Thank you!

Step 1:

Using the SEC's EDGAR API, import and persist data for the CIKs (Central Index Key) listed at the bottom of this document.

API Endpoint: `https://data.sec.gov/api/xbrl/companyfacts/CIK##########.json` where ########## is the entity's 10-digit Central Index Key (CIK), including leading zeros.

To save you some time in troubleshooting, consider specifying the User-Agent header in requests as “PostmanRuntime/7.34.0” and the Accept header in requests as “*/*”. Otherwise, you will receive authorization errors 😄

Here's an example response from the API (yes, it's ugly 😂):

```
{
    "cik": 1543151,
    "entityName": "UBER TECHNOLOGIES, INC.",
    "facts": {
        ...
        "us-gaap": {
            ...
            "NetIncomeLoss": {
                ...
                "units": {
                    "USD": [
                        {
                            "start": "2017-01-01",
                            "end": "2017-12-31",
                            "val": -4033000000,
                            "accn": "0001543151-20-000010",
                            "fy": 2019,
                            "fp": "FY",
                            "form": "10-K",
                            "filed": "2020-03-02",
                            "frame": "CY2017"
                        },
                        ...
                    ]
                }
            },
            ...
        }
    }
}
```

You will be interested in “cik”, “entityName”, and the NetIncomeLoss > units > USD array – specifically the “val”, “form”, and “frame” fields. We’ve also included a response C# contract/class (EdgarCompanyInfo.cs) at the bottom of this document to save you time. Pay special attention to the comments on the fields within InfoFactUsGaapIncomeLossUnitsUsd because they tell you exactly what data we are interested in collecting.

Step 2 below will help you understand exactly why we want to collect this data.

Step 2:

We’d like to offer an HTTP-based API endpoint for retrieving a list of companies as well as the amount of funding they are eligible to receive. The request should optionally allow the user to supply a parameter that can be used to return only companies where their name starts with the specified letter. The response payload for the endpoint should be in the following format (this is important since we will use the output to test for correct output!):

```
[
  {
    "id": 1,
    "name": "Uber, Inc.",
    "standardFundableAmount": 123.45,
    "specialFundableAmount": 234.56
  },
  ...
]
```

How to calculate Standard Fundable Amount:

- Company must have income data for all years between (and including) 2018 and 2022. If they did not, their Standard Fundable Amount is $0.
- Company must have had positive income in both 2021 and 2022. If they did not, their Standard Fundable Amount is $0.
- Using highest income between 2018 and 2022:
    - If income is greater than or equal to $10B, standard fundable amount is 12.33% of income.
    - If income is less than $10B, standard fundable amount is 21.51% of income.

How to calculate the Special Fundable Amount:

- Initially, the Special Fundable Amount is the same as Standard Fundable Amount.
- If the company name starts with a vowel, add 15% to the standard funding amount.
- If the company's 2022 income was less than their 2021 income, subtract 25% from their standard funding amount.

CIKs:

```
18926,892553,1510524,1858912,1828248,1819493,60086,1853630,1761312,1851182,
1034665,927628,1125259,1547660,1393311,1757143,1958217,312070,310522,186184
1,1037868,1696355,1166834,915912,1085277,831259,882291,1521036,1824502,1015
647,884624,1501103,1397183,1552797,1894630,823277,21175,1439124,52827,17307
73,1867287,1685428,1007587,92103,1641751,6845,1231457,947263,895421,1988979
,1848898,844790,1541309,1858007,1729944,726958,1691221,730272,1308106,88414
4,1108134,1849058,1435617,1857518,64803,1912498,1447380,1232384,1141788,154
9922,914475,1498382,1400897,314808,1323885,1526520,1550695,1634293,1756708,
1540159,1076691,1980088,1532346,923796,1849635,1872292,1227857,1046311,1710
350,1476150,1844642,1967078,14272,933267,1157557,1560293,217410,1798562,103
8074,1843370
```

EdgarCompanyInfo.cs

```
using System.Text.Json.Serialization;

public class EdgarCompanyInfo
{
    public int Cik { get; set; }
    public string EntityName { get; set; }
    public InfoFact Facts { get; set; }
    public class InfoFact
    {
        [JsonPropertyName("us-gaap")]
        public InfoFactUsGaap UsGaap { get; set; }
    }

    public class InfoFactUsGaap
    {
        public InfoFactUsGaapNetIncomeLoss NetIncomeLoss { get; set; }
    }

    public class InfoFactUsGaapNetIncomeLoss
    {
        public InfoFactUsGaapIncomeLossUnits Units { get; set; }
    }

    public class InfoFactUsGaapIncomeLossUnits
    {
        public InfoFactUsGaapIncomeLossUnitsUsd[] Usd { get; set; }
    }

    public class InfoFactUsGaapIncomeLossUnitsUsd
    {
        /// <summary>
        /// Possibilities include 10-Q, 10-K,8-K, 20-F, 40-F, 6-K, and
their variants. YOU ARE INTERESTED ONLY IN 10-K DATA!
        /// </summary>
        public string Form { get; set; }

        /// <summary>
        /// For yearly information, the format is CY followed by the year
number. For example: CY2021. YOU ARE INTERESTED ONLY IN YEARLY INFORMATION
WHICH FOLLOWS THIS FORMAT!
        /// </summary>
        public string Frame { get; set; }

        /// <summary>
        /// The income/loss amount.
        /// </summary>
        public decimal Val { get; set; }
    }
}
```