ClearScore - Backend Technical Test

To get an idea of your technical experience we'd like you to develop a simple micro-service that collates financial products from a small selection of partners. You're free to use any JVM based language, along with any tools, frameworks and libraries you feel are appropriate.

We'll be looking for the following when reviewing:

- Documented code with a coherent project structure
- Products being returned in a timely manner
- Upstream APIs returning errors or taking too long to respond
- Unit tests (we're huge fans of TDD!)

Once completed please include the following in a single ZIP file:

- All code, including tests
- A bash script `start.sh` that will start up your service locally, it should support the following environment variables:
  - HTTP_PORT: The port to expose your service on
  - CSCARDS_ENDPOINT: The base url for CSCards
  - SCOREDCARDS_ENDPOINT: The base url for ScoredCards
- A short README outlining how you've designed your service and how you'd intend to deploy it

The Challenge

Your micro-service should expose a single endpoint that consumes some information about the users financial situation and return credit cards recommended for them, sorted based on their eligibility and the cards APR (annual percentage rate).

A swagger definition (Microservice-swagger.json) which your API should conform to has been included in the zip file. The data you'll be returning comes from 2 partner APIs, described in the sections below.

Each partner returns an eligibility rating (i.e. how likely it is the user will be approved) and an APR for each card (watch out for the scales in the sections below), these should be used along with the formula below to sort the cards returned from your API, with the highest scoring cards being ranked higher. The score should be returned in the response for each card as `card-score`.

$$sortingScore = eligibility * ((1/apr)^2)$$

Here's an example request and response from the service:

```
POST /creditcards
{
  "firstname": "John",
  "lastname": "Smith",
  "dob": "1991/04/18",
  "credit-score": 500,
  "employment-status": "FULL_TIME",
  "salary": 28000
}
Response:
[
  {
    "provider": "ScoredCards"
    "name": "ScoredCard Builder",
    "apply-url": "http://www.example.com/apply",
    "apr": 19.4,
    "features": [
      "Supports ApplyPay",
      "Interest free purchases for 1 month"
    ],
    "card-score": 0.212
  },
  {
    "provider": "CSCards",
    "name": "SuperSaver Card",
    "apply-url": "http://www.example.com/apply",
    "apr": 21.4,
    "features": [ ]
    "card-score": 0.137
  },
  {
    "provider": "CSCards",
    "name": "SuperSpender Card",
    "apply-url": "http://www.example.com/apply",
    "apr": 19.2,
    "features": [
      "Interest free purchases for 6 months"
    ]
    "card-score": 0.135
  }
]
```

Partner 1 - CSCards

The first partner provides a JSON API to get eligible cards. They're only interested in a users full name, date of birth and credit score to make their decisions. The response is a list of cards, including an `eligibility` rating from 0.0 to 10.0. See below for an example request and response, and see CSCards-swagger.json for their API definition.

API base url: https://y4xvbk1ki5.execute-api.us-west-2.amazonaws.com/CS

```
POST /v1/cards
{
  "fullName": "John Smith",
  "dateOfBirth": "1990/01/23",
  "creditScore": 500
}
Response:
[
  {
    "cardName": "SuperSaver Card",
    "url": "http://www.example.com/apply",
    "apr": 21.4,
    "eligibility": 6.3
  },
  {
    "cardName": "SuperSpender Card",
    "url": "http://www.example.com/apply",
    "apr": 19.2,
    "eligibility": 5.0,
    "features": [
      "Interest free purchases for 6 months"
    ]
  }
]
```

Partner 2 - ScoredCards
Our other partner uses all the users information to make their scoring decisions, but they separate out their features into two sections (attributes and introductory offers), these need to be combined together. The eligibility rating is provided in the `approval-rating` field and is on a scale from 0.0 to 1.0. The `attributes` and `introductory-offers` fields will always be present, but can be returned as empty lists. An example request and response is below, and see ScoredCards-swagger.json for the swagger definition.

API base url: https://m33dnjs979.execute-api.us-west-2.amazonaws.com/CS

```
POST /v2/creditcards
{
  "first-name": "John",
  "last-name": "Smith",
  "date-of-birth": "1991/04/18",
  "score": 341,
  "employment-status": "PART_TIME",
  "salary": 18500
}
Response:
[
  {
    "card": "ScoredCard Builder",
    "apply-url": "http://www.example.com/apply",
    "annual-percentage-rate": 19.4,
    "approval-rating": 0.8,
    "attributes": [
      "Supports ApplePay"
    ]
    "introductory-offers": [
      "Interest free purchases for 1 month"
    ]
  }
]
```