Assignment

Task Description

Your task is to create a data connector that processes customer data from a CSV file, transforms it as needed, and reliably sends it to a third-party service called ShowAds. This service uses the data to display web banners (ads) to customers when they browse certain websites.

Input Data

The CSV file (data.csv), contains customer data and can include millions of unique records. Each row in the file represents one customer. An example customer entity contains the following fields:

```
{
  "Name": "John Doe",                                   // string
  "Age": 35,                                            // integer
  "Cookie": "26555324-53df-4eb1-8835-e6c0078bb2c0",     // string (UUID format)
  "Banner_id": 35                                       // integer (0-99)
}
```

Validation Requirements

Before sending data to ShowAds, make sure all customers match the following criteria:

- **Name Validation:** The Name field must contain only letters and spaces.
- **Age Filtering:** Customers under 18 must not be processed. The minimum and maximum age limits should be configurable at runtime without the need to re-deploy the container when they are changed.
- **Banner ID Validation:** Ensure Banner_id is between 0 and 99. Do not process records with invalid Banner_id.

Skip all invalid records, but log errors for debugging. After the data passes the validation, send it to the ShowAds API.

ShowAds API

The ShowAds API provides three endpoints described below. You can find it deployed on https://meiro-assignment-968918017632.europe-west1.run.app

POST /auth

Purpose: Return an access token which expires every 24hours. This token needs to be included in the Authorization header of each request sent to the API as Authorization: Bearer <token>.

Request body:

```
{
        "ProjectKey": string
}
```

Response:

```
{
        "AccessToken": string
}
```

For the sake of simplicity, the API returns a token for any Projectkey sent so you can make up your own. Mind the 24h token expiration though.

Response status codes:

- 200 OK if the operation is successful
- 400 Bad request in case of invalid payload (missing or empty ProjectKey)
- 500 Internal server error if there is an error on the server side. The API is not always reliable.
- 429 Too many Requests if the API is under heavy load

—------------------------------

POST /banners/show

Purpose: The web banner will be shown to the visitor (your customer) next time they visit a website.

Request body:

```
{
    “VisitorCookie”: string
    “BannerId”: integer
}
```

Response status codes:

- 200 OK if the operation is successful
- 401 Unauthorized if access token is invalid or expired
- 400 Bad request in case of invalid payload
- 500 Internal server error if there is an error on the server side. The API is not always reliable.
- 429 Too many Requests if the API is under heavy load

—------------------------------

POST /banners/show/bulk

Purpose: The web banner will be shown to all visitors (your customers) in the payload next time they visit a website. There is an internal limit of 1000 visitors per request.

Request body:

```
{
    “Data”: [
            {
                “VisitorCookie”: string
                “BannerId”: integer
            },
            {
                “VisitorCookie”: string
                “BannerId”: integer
            },
            …
        ]

}
```

Response status codes:

- 200 OK if the operation is successful
- 401 Unauthorized if access token is invalid or expired
- 400 Bad request in case of invalid payload
- 500 Internal server error if there is an error on the server side.

- 429 Too many Requests if the API is under heavy load

Your task is to create a Python or Go module which loads the customer data from the csv file, validates it and sends it to the ShowAds API to display web ads to these customers. Once the data is sent to the API and it responds with a 200 OK status code, you can consider the job done.

Additional requirements:

- Wrap the code inside a Docker container
- Make sure your code is properly covered by tests
- Your code will be deployed in production. Include proper logs so you see what’s happening. Standard logging to stdout or stderr is enough.
- Document the code for your colleagues.
- Create a github repository with your solution.