Objective

Create an API Server that reads a CSV file containing 100 person records, validates and stores them in memory, and provides two HTTP endpoints to retrieve this data.

Specifications

1. Create a CSV file called persons.csv that contains 100 records, each containing:
    a. Id (string, unique (UUID v4))
    b. First_Name (string)
    c. Last_Name (string)
    d. Email (string)
    e. Salary (float, 2 decimals)
2. Server Behaviour
    a. Start up and read persons.csv
    b. Validate each record (check required fields and types)
    c. Store valid records in memory as JSON
    d. Provide the following REST endpoints:
        i. GET /persons
            1. returns 10 records
        ii. GET /persons/<id>
            1. returns the record for a specific "id"
    e. Listen on port 8080

Implementation Outline

1. CSV Preparation
    a. Ensure persons.csv has the required headers:
        i. Id,First_Name,Last_Name,Email,Salary
    b. Add 100 rows of valid data
2. Server Initialization
    a. Choose a programming language (e.g., Python, Node.js, etc.)
    b. On application start, read the persons.csv file from the project directory
3. Data Validation
    a. Verify that each row has:
        i. A non-empty Id (unique)
        ii. Valid First_Name, Last_Name, and Email
        iii. A properly formatted numeric Salary

    b. If any row fails validation, decide whether to skip it or stop the startup process
4. In-Memory Storage
    a. Convert each valid row to JSON (or a dictionary/object)
    b. Store all records in a list or dictionary in memory (e.g., persons_list or persons_dict)
5. API Endpoints
    a. GET /persons
        i. Return the first 10 records in JSON format
        ii. Example response:

```
[
  {
    "Id": "6f46d731-a2cf-4ba5-84af-fc44738c35f2",
    "First_Name": "John",
    "Last_Name": "Doe",
    "Email": "john.doe@example.com",
    "Salary": 50000.00
  },
  ...
]
```

    b. GET /persons/<id>
        i. Look up the person by id
        ii. If found, return their record in JSON; otherwise, return a 404 error
6. Error Handling
    a. CSV Errors: Log or skip invalid lines
    b. 404: Return a “Person not found” message if an id isn’t recognized
    c. 500: Return an “Internal Server Error” message for unexpected issues
7. Testing
    a. Use a simple tool (like curl or Postman) to send requests to:
        i. GET http://localhost:8080/persons
        ii. GET http://localhost:8080/persons/<id>
    b. Verify correct records and status codes
8. Documentation
    a. Create simple README.md file