Receipt Processor

Hardik Sandeep Fulfagar
fulfagarhardik@gmail.com

Table of Contents

Introduction

The **Receipt Processor** is a web service built to process receipts, calculate points based on specific rules, and provide detailed breakdowns of how these points are earned. The service is implemented in **Go** and operates in-memory, meaning no data persists after the application stops.

API Endpoints

1. Process Receipts

- **Path**: `/receipts/process`
- **Method**: POST
- **Description**: Processes a receipt JSON payload, validates it, calculates points, and stores the receipt in memory.

  Request Payload Example:

```
{
  "retailer": "Target",
  "purchaseDate": "2022-01-01",
  "purchaseTime": "13:01",
    "items": [
      {
        "shortDescription": "Mountain Dew 12PK",
        "price": "6.49"
      },{
        "shortDescription": "Emils Cheese Pizza",
        "price": "12.25"
      },{
        "shortDescription": "Knorr Creamy Soup",
        "price": "1.26"
      },{
        "shortDescription": "Doritos Nacho Cheese",
        "price": "3.35"
      },{
        "shortDescription": "   Klarbrunn 12-PK 12 FL OZ  ",
        "price": "12.00"
      }
    ],
    "total": "35.35"
}
```

  Response Example::

```
{ "id": "cb445f45-21e3-48b6-acd9-3150c9ed429c" }
```

2. Get Points

- **Path**: `/receipts/{id}/points`
- **Method**: GET
- **Description**: Retrieves the total points awarded for a receipt.

  Response Example:
  ```
  { "points": 28 }
  ```

3. Get Breakdown *(Additional feature)*

- **Path**: `/receipts/{id}/breakdown`
- **Method**: GET
- **Description**: Provides a detailed breakdown of how the points were calculated for the receipt.

  Response Example:
  ```
  {
    "breakdown": [
      "6 points - retailer name (Target) has 6 alphanumeric characters",
      "10 points - 5 items (2 pairs @ 5 points each)",
      "3 points - \"Emils Cheese Pizza\" is 18 characters (a multiple of
  3), item price 12.25 * 0.2 = 2.45 which is rounded to: 3 points",
      "3 points - \"Klarbrunn 12-PK 12 FL OZ\" is 24 characters (a multiple
  of 3), item price 12.00 * 0.2 = 2.40 which is rounded to: 3 points",
      "6 points - purchase day is odd"
    ],
    "points": 28
  }
  ```

How to Run

Prerequisites

1. Clone the git repository:

   ```
   git clone https://github.com/hsf6/receipt-processor-challenge.git

   cd receipt-processor-challenge
   ```

2. Install Go.
3. Ensure `payload.json` is present in the project directory.

Steps

1. **Run the Server**:
   ```
   go run main.go
   ```
   (The server will start on `http://localhost:8080`.)

2. **Run the Client**:
   ```
   go run client.go
   ```
   (The client will read the `payload.json`, process it, and fetch the breakdown.)

Payload Example

Sample: `payload.json`

```
{

  "retailer": "Target",
  "purchaseDate": "2022-01-01",
  "purchaseTime": "13:01",
  "items": [
    {"shortDescription": "Mountain Dew 12PK", "price": "6.49"},
    {"shortDescription": "Emils Cheese Pizza", "price": "12.25"}
  ],
  "total": "18.74"
}
```

Additional features and implementaiton details

1. **Detailed Breakdown (Additional Endpoint)**
   - The /breakdown endpoint provides a clear, human-readable explanation of how the points were earned.
2. **Logging**
   - Logs all requests, errors, and internal processes for traceability and debugging.
3. **Validations**
   - Each field is validated against real-world constraints, such as valid date and time formats, numeric total and price values, and allowed characters in retailer names.
4. **In-Memory Storage**
   - Receipts are stored in a thread-safe map with a mutex to ensure concurrency safety.