**Project Documentation: Receipt Processor**

This document outlines the steps involved in building and running a Dockerized receipt processing application using Flask.

**1. Application Overview:**

The Python application (`app.py`) uses Flask to create a REST API for processing receipts and calculating reward points based on predefined rules. It uses an in-memory dictionary to store receipt data and their corresponding points.

**2. API Endpoints:**

POST /receipts/process:

- Accepts a JSON payload representing a receipt.

- Calculates points based on the receipt data.

- Stores the receipt and its points in memory.

- Returns a JSON response with a unique ID for the processed receipt.

GET /receipts/<receipt\_id>/points:

- Retrieves the points associated with a given receipt ID.

- Returns a JSON response with the points.

**3. Point Calculation Rules (Implemented in `calculate\_points` function):**

1. One point for every alphanumeric character in the retailer name.

2. 50 points if the total is a round dollar amount with no cents.

3. 25 points if the total is a multiple of 0.25.

4. 5 points for every two items on the receipt.

5. If the trimmed length of the item description is a multiple of 3, round up (0.2 \* price) and add to points.

6. 6 points if the day in the purchase date is odd.

7. 10 points if the time of purchase is after 2:00pm and before 4:00pm.

**4. Dockerization:**

The application is Dockerized using the following `Dockerfile`:

```dockerfile

FROM python:3.9-slim-buster

WORKDIR /app

COPY requirements.txt .

RUN pip install -r requirements.txt

COPY . .

EXPOSE 5000

CMD ["python3", "app.py"]

From the directory containing your docker-compose.yml file, run:

**docker-compose up -d**

Stop the containers:

**docker-compose down**