Project Overview

You are tasked with developing a RESTful API service using Python. The API will process a large dataset and return summary statistics. This project will test your ability to handle data efficiently, design a robust API, and document your work.

Dataset

- **Type**: CSV file
- **Content**: E-commerce transactions
- **Fields**: Transaction ID, User ID, Product ID, Timestamp, Transaction Amount
- **Size**: 1 million transactions

API Requirements

1. **Endpoints**:
    - `/upload`: Accepts a CSV file upload. Validates the file format and stores the data for processing.
    - `/summary/<user_id>`: Returns summary statistics of the uploaded data, including ***maximum***, ***minimum***, and ***mean*** `Transaction Amount`, for some user, in some date range.
2. **Technical Specifications**:
    - Use Python with FastAPI.
    - Implement data processing of the uploaded data.
    - Ensure the API is well-documented and follows RESTful principles.
    - Include error handling and input validation.
3. **Evaluation Criteria**
    - **Functionality**: The API should meet all the specified requirements and handle large datasets efficiently.
    - **Code** Quality: Code should be clean, well-organized, and follow best practices.
    - **Documentation**: Provide clear documentation on how to use the API, including setup instructions and API endpoint details.
    - **Testing**: Include unit tests for the API endpoints.
    - **Understanding**: Be prepared to explain your code and approach in a follow-up interview.
4. **Submission Guidelines**
    - Please send your code repository (e.g. GitHub link) along with a README file explaining your approach in a reply to the challenge email.
    - Ensure your submission is original. We will use plagiarism detection tools to compare submissions.
5. **Tips for Success**
    - Focus on writing clean, maintainable code.

- Ensure your API can handle concurrent requests and large data efficiently.
- Document your code and API endpoints thoroughly.
- Test your API to ensure it meets all requirements.

Code for generating test data

For convenience, you may use the code snippet below for generating sample data.

```python
import csv
import random
from pathlib import Path

from faker import Faker

TRANSACTIONS = 1_000_000
HEADERS = ["transaction_id", "user_id", "product_id", "timestamp",
"transaction_amount"]

# Initialize Faker
fake = Faker()

# Open the CSV file for writing
with Path("dummy_transactions.csv").open(mode="w", newline="") as file:
    writer = csv.DictWriter(file, fieldnames=HEADERS)
    writer.writeheader()

    # Generate dummy data
    for _ in range(TRANSACTIONS):
        writer.writerow(
            {
                "transaction_id": fake.uuid4(),
                "user_id": fake.random_int(min=1, max=1000),
                "product_id": fake.random_int(min=1, max=500),
                "timestamp": fake.date_time_between(start_date="-1y",
end_date="now"),
                "transaction_amount": round(random.uniform(5.0, 500.0), 2),
            }
        )
```