写一个 Python 实现的最小支付系统样例（含 webhook + idempotency 处理）

下面是一个Python 最小支付系统样例，用 Flask 模拟一个典型支付流程，包括：

创建订单并发起支付

模拟支付网关回调（Webhook）

实现幂等处理（防止重复扣款）

🛠️ 环境要求
Python 3.8+

Flask

Redis（用于幂等性检查）

你可以用如下命令安装依赖：

bash
Copy
Edit
```
pip install flask redis
```
📦 目录结构
bash
Copy
Edit
```
payment_demo/
├── app.py # 主程序
├── redis_client.py # Redis 客户端工具
└── simulate_gateway.py # 模拟支付平台回调
```

1️⃣ redis_client.py
python
Copy
Edit
```
import redis
```

本地 Redis 默认配置

r = redis.Redis(host="localhost", port=6379, db=0, decode_responses=True)

def set_idempotency_key(key: str, value: str, ttl: int = 600):
if not r.exists(key):
r.set(key, value, ex=ttl)
return True
return False

def get_value(key: str):
return r.get(key)

2️⃣ app.py（主服务）
python
Copy
Edit
from flask import Flask, request, jsonify
from redis_client import set_idempotency_key, get_value
import uuid

app = Flask(**name**)

模拟数据库（内存）

ORDERS = {}
PAYMENTS = {}

@app.route("/create_order", methods=["POST"])
def create_order():
user_id = request.json["user_id"]
amount = request.json["amount"]
order_id = str(uuid.uuid4())

```
ORDERS[order_id] = {
    "user_id": user_id,
    "amount": amount,
```

```
    "status": "PENDING"
}

return jsonify({"order_id": order_id})
```

@app.route(“/initiate_payment”, methods=[“POST”])
def initiate_payment():
order_id = request.json[“order_id”]
payment_id = str(uuid.uuid4())
idempotency_key = f"payment:{payment_id}"

```
if not set_idempotency_key(idempotency_key, "initiated"):
    return jsonify({"error": "Duplicate payment"}), 409

PAYMENTS[payment_id] = {
    "order_id": order_id,
    "status": "PENDING"
}

# 假设支付链接为模拟跳转
return jsonify({
    "payment_id": payment_id,
    "payment_url":
f"http://localhost:5000/simulate_gateway_callback/{payment_id}"
})
```

@app.route(“/webhook/payment_callback”, methods=[“POST”])
def payment_callback():
payment_id = request.json[“payment_id”]
success = request.json[“success”]
idempotency_key = f"callback:{payment_id}"

```
if not set_idempotency_key(idempotency_key, "received"):
    return jsonify({"message": "Duplicate callback"}), 200

payment = PAYMENTS.get(payment_id)
if not payment:
    return jsonify({"error": "Invalid payment ID"}), 404

if success:
```

```
    payment["status"] = "PAID"
    order_id = payment["order_id"]
    ORDERS[order_id]["status"] = "PAID"
else:
    payment["status"] = "FAILED"

return jsonify({"message": "Callback processed"})
```

@app.route(“/orders”, methods=[“GET”])
def get_orders():
return jsonify(ORDERS)

if **name** == “**main**”:
app.run(debug=True)

3️⃣ simulate_gateway.py（模拟支付平台回调）
python
Copy
Edit
import requests
import sys

payment_id = sys.argv[1] if len(sys.argv) > 1 else None
if not payment_id:
print(“Usage: python simulate_gateway.py <payment_id>”)
exit()

res = requests.post(“http://localhost:5000/webhook/payment_callback”,
json={
“payment_id”: payment_id,
“success”: True
})

print(“Webhook sent:”, res.status_code, res.json())

🔁 流程演示（本地端口为 5000）
启动 Redis

启动 Flask 服务：

bash
Copy
Edit
python app.py
创建订单：

bash
Copy
Edit
curl -X POST http://localhost:5000/create_order -H “Content-Type: application/json” -d ‘{“user_id”: “u001”, “amount”: 100}’
发起支付：

bash
Copy
Edit
curl -X POST http://localhost:5000/initiate_payment -H “Content-Type: application/json” -d ‘{“order_id”: “<上一步返回的order_id>”}’
用返回的 payment_id 触发 webhook 模拟：

bash
Copy
Edit
python simulate_gateway.py <payment_id>
查看订单状态：

bash
Copy
Edit
curl http://localhost:5000/orders

✅ 幂等性保证点
场景 Key 功能
发起支付 payment:{payment_id} 避免重复提交支付
支付回调 callback:{payment_id} 避免重复处理回调（如多次回调）

🧠 下一步你可以扩展：
使用真实数据库（PostgreSQL, MySQL）

加入 MQ 消息队列实现异步状态流转

增加超时关闭、退款等状态

增加订单状态机控制

如你想，我也可以帮你接入 Stripe / 微信支付 真实网关接口版本。是否需要继续升级?