AgentPayment Sandbox

Test AI Agent Payments Without Real Money

Sivasubramanian Ramanathan
Product Owner | Fintech & Innovation

Open to roles in Product Management, Fintech, Payments, RegTech

The Problem I Solved

"I'm building an AI shopping agent. How do I test it?"

Without Sandbox	With APS
Real money or test accounts	**Free, instant, local**
Read specs, hope you got it right	**Automated Inspector tests**
Implement each protocol separately	**Unified testing for all 4**
Manual signature debugging	**Security Analyzer with scoring**

There was no "Postman for Agent Payments". So I built one.

The Protocol Landscape 📋

Google announced UCP with 20+ partners including Shopify, Stripe, Walmart, Target.

Protocol	Owner	Purpose
UCP	Google + Partners	Universal checkout standard
AP2	Google	Agent Payments (A2A extension)
ACP	OpenAI + Shopify	E-commerce checkout
x402	Coinbase	HTTP 402 micropayments

How A2A fits: A2A is Google's Agent-to-Agent messaging protocol. **AP2** is the payment extension built on top of A2A.

What I Actually Built 🔧

Backend (FastAPI)

- `ucp.py` — 475 lines
- `ap2.py` — 727 lines
- `x402.py` — 524 lines
- `acp.py` — 340 lines
- `inspector.py` — 491 lines
- `x402_schema.py` — 226 lines

Total: 2,700+ lines of Python

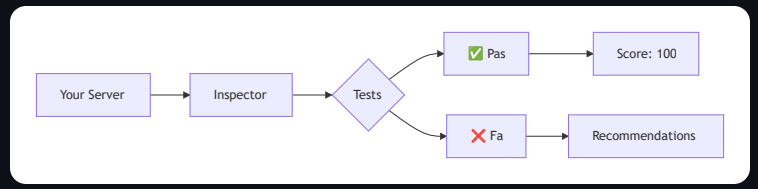
Frontend (React + TypeScript)

- Playground UI
- Dashboard with protocol cards
- Flow Runner (step-by-step)
- Security Analyzer panel

Demo Mode: Works on GitHub Pages with mock data

The Inspector: Compliance Testing 🔍

Point it at any server. It runs test suites and returns a score.



Test Suites

- **UCP**: 5 tests (discovery, checkout, idempotency)
- **ACP**: 5 tests (session states, line items)
- **x402**: 5 tests (402 response, CAIP-2 networks)
- **AP2**: 2 tests (agent card, message handler)

Output: Pass/Fail + Security Score + Recommendations

Schema Validators: Pydantic Power

Every request is validated against the official spec.

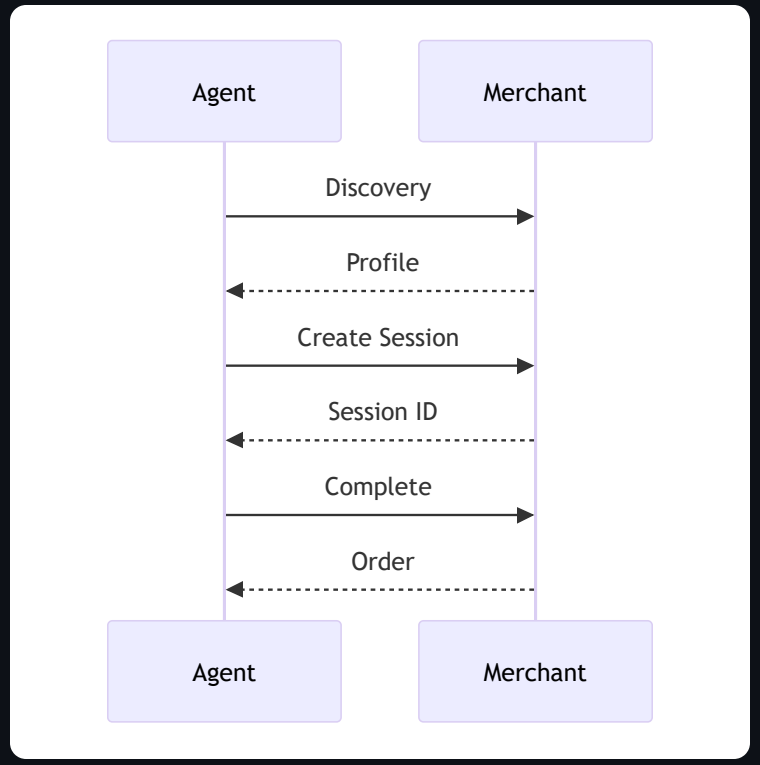
```python
# x402_schema.py - Validates EIP-3009 Authorization
class AuthorizationSchema(BaseModel):
    from_: str = Field(alias="from")  # EVM address
    to: str                           # Receiver address
    value: str                        # Amount (wei)
    nonce: str                        # 32-byte hex

    @field_validator("from_", "to")
    def validate_address(cls, v):
        if not v.startswith("0x") or len(v) != 42:
            raise ValueError("Must be valid EVM address")
        return v
```

Result: Agents learn the correct format before hitting production.

UCP Flow: Universal Commerce

Discovery → Session → Complete (Google + 20 Partners)



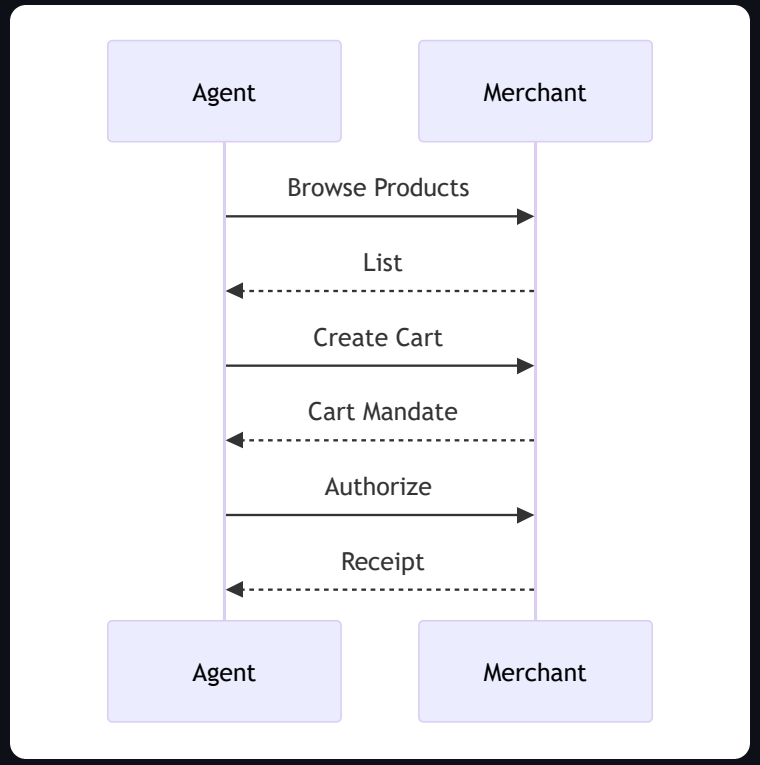
Endpoints Tested

1. `GET /.well-known/ucp`
2. `POST /checkout-sessions`
3. `PUT /checkout-sessions/{id}`
4. `POST /checkout-sessions/{id}/complete`
5. `Idempotency-Key` header

Code: `backend/app/mock/ucp.py`

AP2 Flow: Agent Mandates

A2A Messaging + Intent/Cart Mandates + OTP



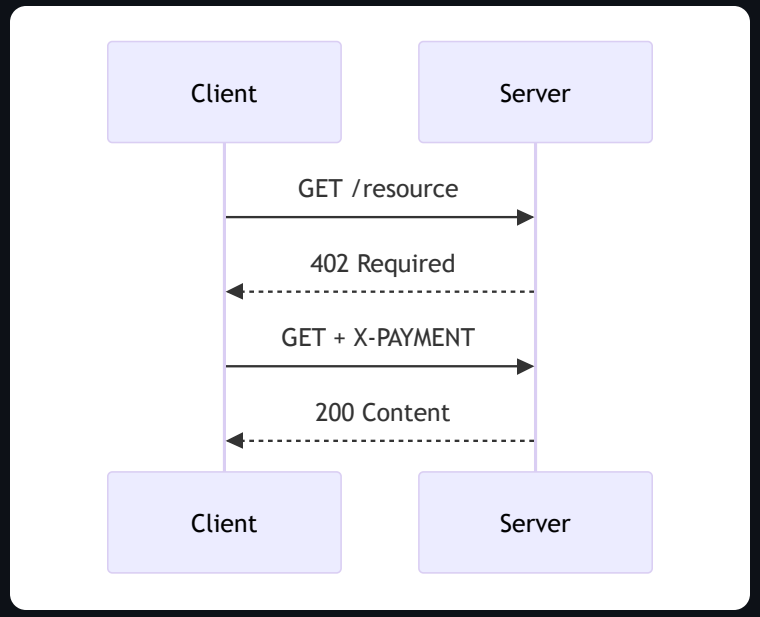
A2A Methods Implemented

- `ap2/createIntentMandate`
- `ap2/browseProducts`
- `ap2/createCart` → CartMandate
- `ap2/initiatePayment` → OTP
- `ap2/submitOtp` → Receipt

Code: `backend/app/mock/ap2.py`

x402 Flow: Micropayments ⚡

HTTP 402 + EIP-712 Signatures (Coinbase)



x402 v2 Features

- CAIP-2 network IDs (`eip155:84532`)
- PaymentRequired with `accepts` array
- EIP-3009 authorization
- Facilitator API: `/verify`, `/settle`

Code: `backend/app/mock/x402.py`

Why a Product Owner Built This 

"You wrote 2,700+ lines of code. Aren't you a PM?"

My Philosophy

1. **Build to Understand** — I prototype to learn the problem space
2. **Bridge Gaps** — Translate complex specs into testable artifacts
3. **De-risk Decisions** — Validate ideas before committing teams

What This Demonstrates

- I can read protocol specs (x402 v2, AP2, ACP, UCP)
- I can implement working software (FastAPI, React, Pydantic)
- I can document thoroughly (8 docs, 3 ADRs)

> **The best PMs accept complexity. They don't outsource understanding.**

GitHub Pages Demo

How I deployed a backend-heavy app to static hosting.

The Challenge

GitHub Pages = No Server.
API calls fail by default.

The Solution

```
const IS_DEMO = window.location
  .hostname.includes('github.io');

if (IS_DEMO) {
  return DEMO_DATA[endpoint];
}
```

Result

A frictionless live demo for recruiters.

Live Demo:
siva-sub.github.io/AgentPayment-Sandbox

GitHub:
github.com/siva-sub/AgentPayment-Sandbox

Let's Connect 🤝

Sivasubramanian Ramanathan

Product Owner | Fintech, Payments & Innovation

Open for roles in:

Product Management • Fintech • Payments • RegTech • Digital Assets

🌐 sivasub.com

💼 linkedin.com/in/sivasub987

💻 github.com/siva-sub

Thank You! 🙏

AgentPayment Sandbox — The Postman for Agent Payments

🔗 **Live Demo**: siva-sub.github.io/AgentPayment-Sandbox

📚 **Documentation**: /docs