APS 🤖💳

AgentPayment Sandbox

Sivasubramanian Ramanathan
Product Owner | Fintech & Innovation
Ex-BIS Innovation Hub Singapore

🌏 **Seeking Opportunities in Singapore**
I am looking for roles in **Product Management, Fintech, Payments, RegTech**, and **Digital Assets**.

"I am a Product person. **I build.**"

I have worked across product delivery, user research, and cross-agency collaboration. I enjoy solving complex problems and bringing structure to early ideas.

I care deeply about building products that create real impact.

The Problem: Protocol Fragmentation 🧩

Four major companies have released agent payment protocols. Each solves part of the puzzle.

Protocol	Company	Approach	Use Case
AP2	Google	A2A + Mandates	Multi-agent orchestration
x402	Coinbase	HTTP 402	Micropayments, pay-per-call
ACP	Shopify	OpenAPI	E-commerce checkout
UCP	Stripe	Universal API	Card payments

"*The landscape is fragmenting. Developers building agent commerce must learn 4 different specs.*"

The Developer Pain Point 😤

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

Option A: Real Test Accounts 💸

- Sign up for Stripe Test Mode
- Configure Shopify sandbox
- Apply for API access to each
- **Days of setup per protocol**

Option B: Build Mock Servers 🔧

- Read each protocol spec
- Implement endpoints from scratch
- Maintain when specs change
- **Weeks of work**

"*Neither option is acceptable for rapid prototyping.*"

The Gap I Identified 🔍

There is no **"Postman for agent payments"** today.

What Developers Cannot Do:

- ❌ Test full payment flows without real money
- ❌ Validate protocol compliance automatically
- ❌ Reproduce edge cases (errors, timeouts, 402)
- ❌ Learn multiple protocols interactively
- ❌ Compare protocol approaches side-by-side

What Would Help:

- ✅ Mock servers for all 4 protocols
- ✅ Interactive playground UI
- ✅ Validation against official specs
- ✅ Zero setup, zero cost

Introducing APS 🚀

Postman + Chaos Monkey + Case Manager for Agent Payments

What It Provides

- ⚡ **4 Mock Servers**: UCP, ACP, x402, AP2
- 🔍 **Inspector**: Validate your implementation
- 🎮 **Playground**: Explore protocols step-by-step
- 🛡️ **Security Analyzer**: Check signatures

Who Benefits
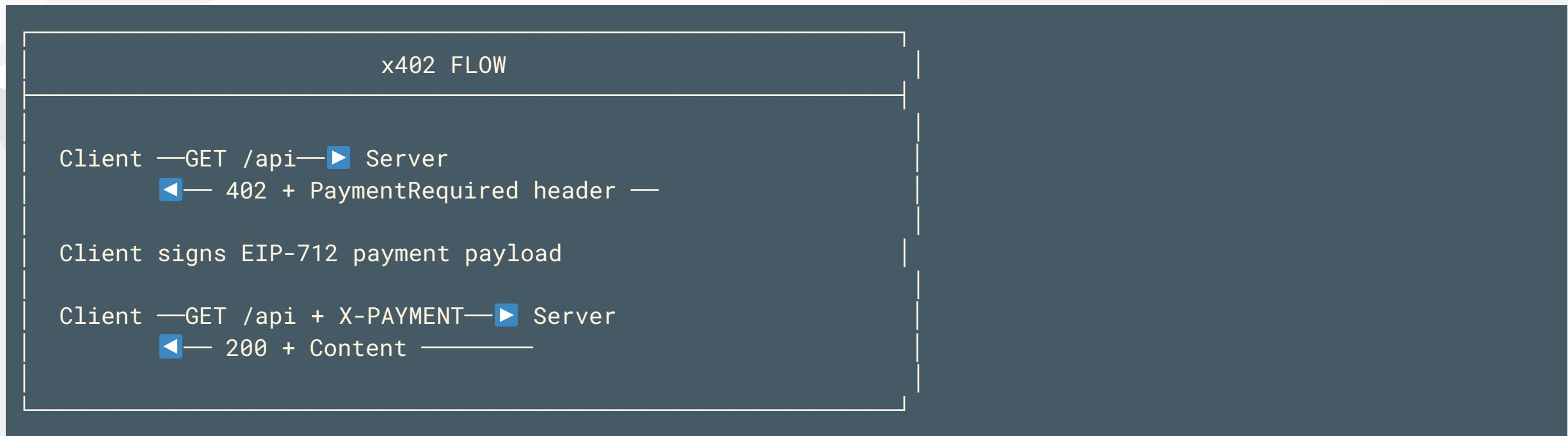
- Developers building AI shopping agents
- Teams integrating agent payments
- Engineers learning protocol specs
- Architects comparing approaches

"*One sandbox to test all four protocols.*"

The x402 Flow ⚡

Coinbase's HTTP 402 enables pay-per-request APIs.

```
┌─────────────────────────────────────────────┐
│                 x402 FLOW                     │
├─────────────────────────────────────────────┤
│                                               │
│ Client ──GET /api──▶ Server                   │
│        ◀── 402 + PaymentRequired header ──    │
│                                               │
│ Client signs EIP-712 payment payload          │
│                                               │
│ Client ──GET /api + X-PAYMENT──▶ Server       │
│        ◀── 200 + Content ───────────          │
│                                               │
└─────────────────────────────────────────────┘
```
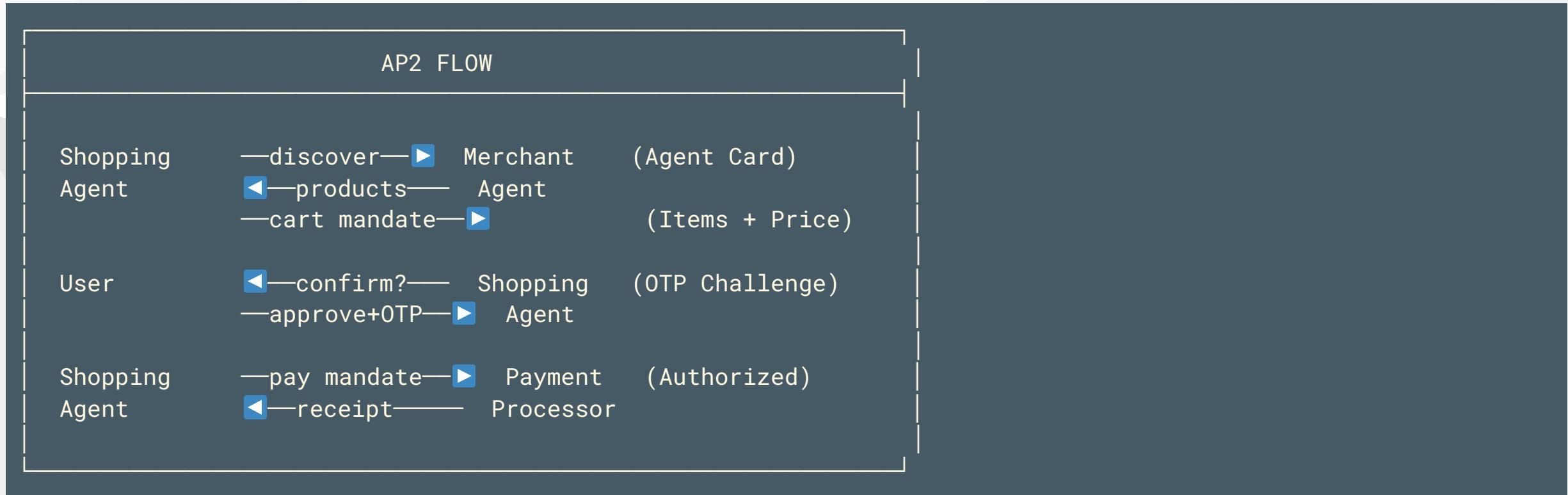
APS simulates the 402 response, payment header validation, and content delivery.

The AP2 Flow 🤖🤖🤖
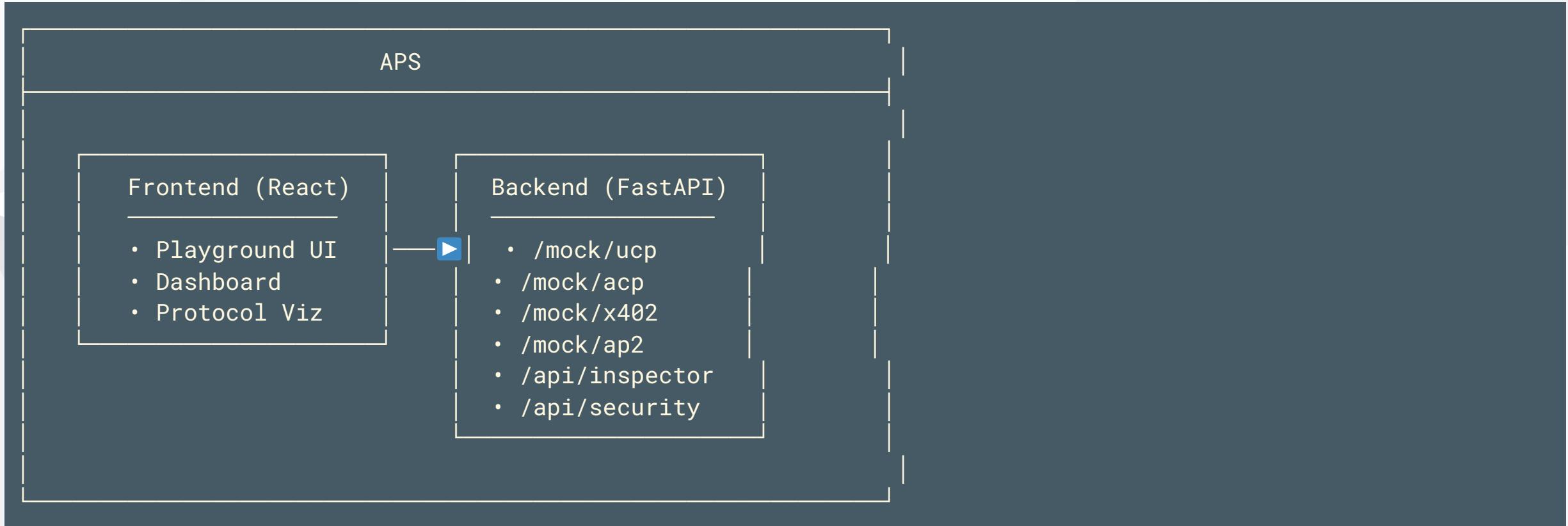
Google's AP2 enables agent-to-agent commerce with spending limits.

```
┌─────────────────────────────────────────────────┐
│                   AP2 FLOW                        │
├─────────────────────────────────────────────────┤
│                                                   │
│  Shopping      ──discover──▶  Merchant   (Agent Card) │
│  Agent         ◀──products──   Agent                  │
│                ──cart mandate──▶         (Items + Price) │
│                                                   │
│                                                   │
│  User          ◀──confirm?──   Shopping  (OTP Challenge) │
│                ──approve+OTP──▶  Agent                │
│                                                   │
│                                                   │
│  Shopping      ──pay mandate──▶  Payment  (Authorized) │
│  Agent         ◀──receipt──      Processor            │
│                                                   │
└─────────────────────────────────────────────────┘
```

APS simulates the full multi-agent flow including OTP verification.

Technical Architecture ⚙


```
┌─────────────────────────────────────────────────────────────┐
│ APS                                                         │
│ ┌──────────────────────────────────────────────────────┐  │
│ │                                                       │  │
│ │  ┌─────────────────────┐     ┌──────────────────────┐ │  │
│ │  │ Frontend (React)    │     │ Backend (FastAPI)    │ │  │
│ │  │ ─────────────────   │     │ ──────────────────   │ │  │
│ │  │                     │     │                      │ │  │
│ │  │ • Playground UI     │──▶│ •  /mock/ucp         │ │  │
│ │  │ • Dashboard         │     │ • /mock/acp          │ │  │
│ │  │ • Protocol Viz      │     │ • /mock/x402         │ │  │
│ │  │                     │     │ • /mock/ap2          │ │  │
│ │  │                     │     │ • /api/inspector     │ │  │
│ │  │                     │     │ • /api/security      │ │  │
│ │  └─────────────────────┘     └──────────────────────┘ │  │
│ │                                                       │  │
│ └──────────────────────────────────────────────────────┘  │
└─────────────────────────────────────────────────────────────┘
```

- **Frontend**: React + TypeScript + Vite + TailwindCSS
- **Backend**: Python + FastAPI + Pydantic

Demo Mode: Works on GitHub Pages 🌐

Live demo runs on static hosting. How?

The Challenge

GitHub Pages = no server.
API calls would fail.

What Users See

The Solution

```
⚠ Demo Mode

Running with mock data.
For live API calls, run:

cd backend
uvicorn app.main:app --port 8080
```

- Detect `github.io` hostname
- Return **realistic mock data**
- Show clear **Demo Mode** banner

"*Users understand the limitation and can run locally for full functionality.*"

Why This Matters 🌟

This project demonstrates my approach to building products:

1. **Problem-First Thinking**:
"Developers can't test agent payments" came before "let me build cool tech".

2. **Market Awareness**:
4 protocols from Google, Coinbase, Shopify, Stripe — the landscape is real.

3. **Full-Stack Execution**:
React frontend, Python backend, 8 doc files, 3 ADRs, GitHub Pages deploy.

4. **Developer Experience Focus**:
Making complex protocols **accessible and testable**.

About Me 👨‍💼

Sivasubramanian Ramanathan
Product Owner | Fintech, Payments & Digital Innovation
PMP | PSM II | PSPO II

I specialize in taking messy, real-world complexity and structuring it into reliable products.

Open for roles that sit between policy, technology, and stakeholder engagement.

Previous: **BIS Innovation Hub Singapore** - Cross-border payments, CBDCs, regulatory innovation.

Let's Connect 🤝

I am ready to bring this level of product thinking and technical depth to your team.

- 🌐 **Portfolio**: sivasub.com
- 💼 **LinkedIn**: linkedin.com/in/sivasub987
- 💻 **Code**: github.com/siva-sub/AgentPayment-Sandbox
- 📚 **Docs**: Full Documentation

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