APS 🤖💳

AgentPayment Sandbox

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

🌏 **Seeking Opportunities in Singapore**
Product Management • Fintech • Payments • RegTech

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

The best Product Owners don't just write specs—they prototype. I built APS to deeply understand the protocols I might one day govern.

This is how I learn: by building.

The Paradigm Shift 🌍

Checkout forms were designed for humans. Agents don't have fingers.

1. The Old World (Human Commerce) 👤

- Human → Browser → Click "Buy" → CAPTCHA → Enter Card → Done
- **Designed for**: Eyeballs and fingers
- **Result**: CAPTCHAs actively block automation

2. The New World (Agentic Commerce) 🚀

- Agent → API Discovery → Structured Checkout → Crypto Auth → Done
- **Designed for**: Autonomous software agents
- **Result**: Mandates and signatures replace passwords

> **The question is no longer IF agents will transact, but HOW.**

The Problem I Actually Solved 🧩

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

Challenge	Without Sandbox	With APS
Testing payments	Real money or test accounts	Free, instant, local
Protocol compliance	Read specs, hope you got it right	Automated Inspector
Multiple protocols	Implement each one separately	Unified testing for all 4
Edge cases	Hard to reproduce	Controllable mock responses

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

The Protocol Landscape 📋

Four major protocols have emerged. APS tests all of them.

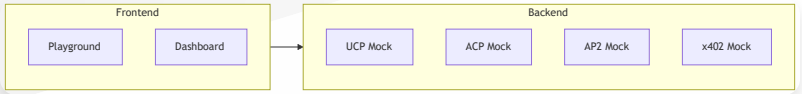
Protocol	Maintainer	What It Does
UCP	**Google + Partners**	Universal checkout (Shopify, Walmart, Target)
AP2	**Google**	Agent-to-Agent mandates with A2A messaging
ACP	**OpenAI/Shopify**	OpenAPI-based e-commerce checkout
x402	**Coinbase**	HTTP 402 "Payment Required" for micropayments

How they relate:

- **UCP** = The universal standard (broadest adoption)
- **AP2** = High-trust purchases (mandates, OTP challenges)
- **ACP** = E-commerce focus (fulfillment, variants)
- **x402** = Metered access (pay-per-request APIs)

Introducing APS 🚀

Unified Sandbox for Agentic Commerce



What I Built

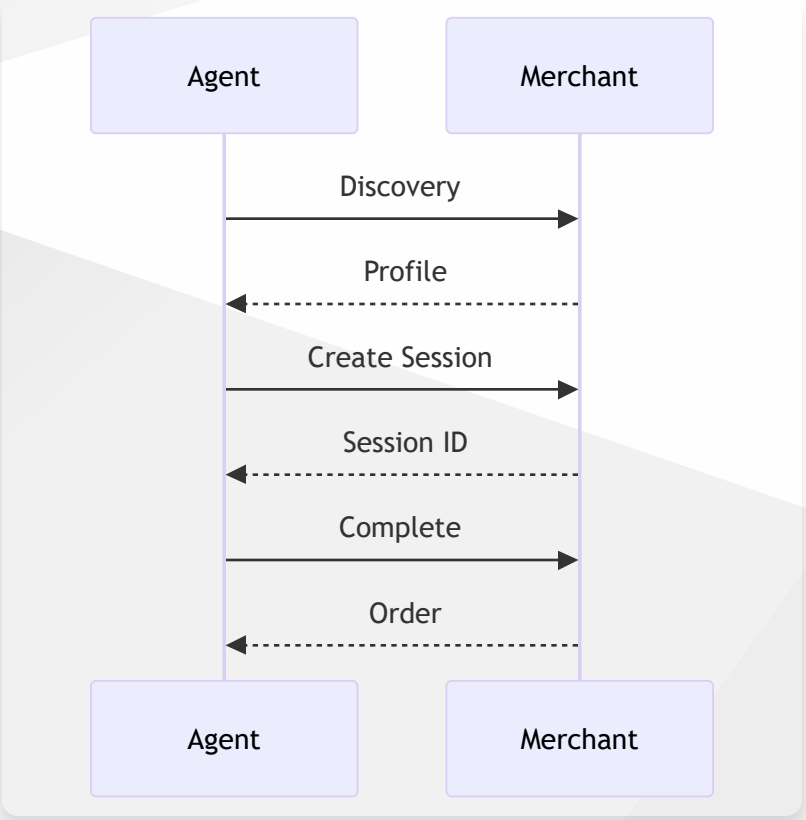
- ⚡ **4 Mock Servers**: UCP, AP2, ACP, x402
- 🔍 **Inspector**: Validates requests against specs
- 🛡️ **Security Analyzer**: Signature verification
- 🎮 **Playground UI**: Interactive protocol explorer

Tech Stack

- Frontend: React + TypeScript + Vite
- Backend: FastAPI + Pydantic (2,000+ lines)

UCP Flow: Universal Commerce 💳

Discovery → Session → Complete (Google + Shopify + Walmart)



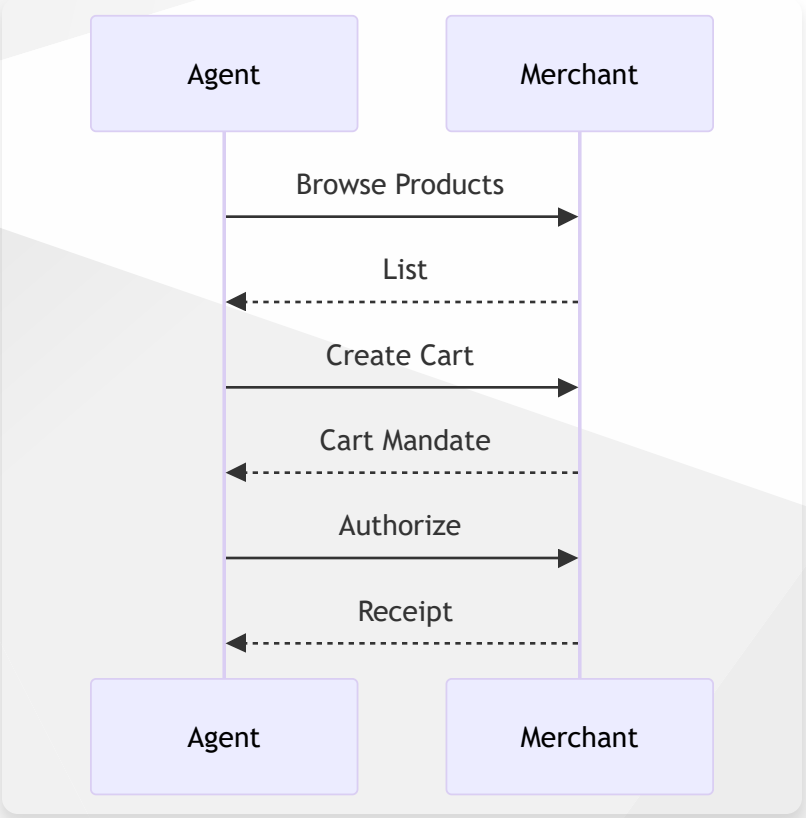
APS Tests

1. **Discovery**: `/.well-known/ucp`
2. **Create Session**: POST `/checkout-sessions`
3. **Update Session**: PUT `/checkout-sessions/{id}`
4. **Complete**: POST `/complete`
5. **Idempotency**: `Idempotency-Key` header

Code: `ucp.py` (475 lines)

AP2 Flow: Agent Mandates 🤖

A2A Messaging + Intent/Cart Mandates + OTP (Google)



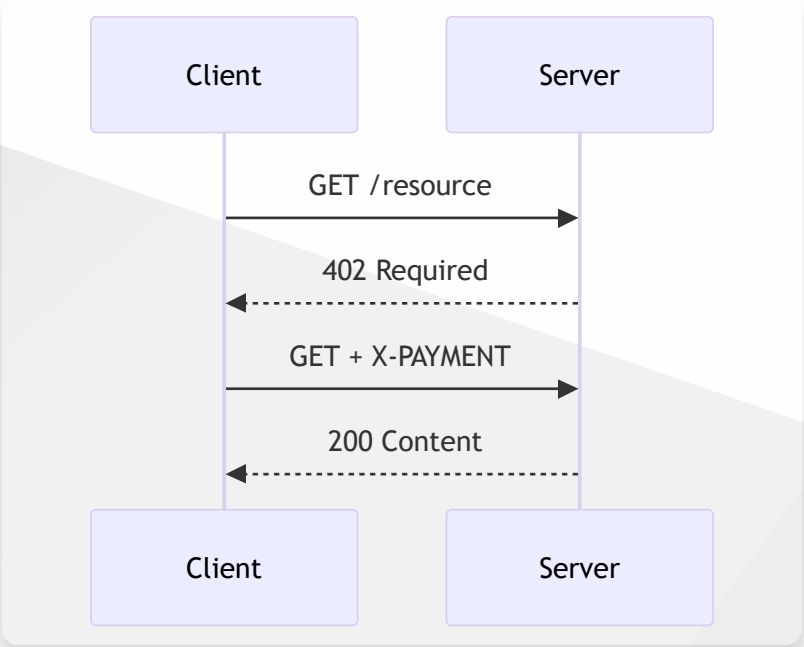
APS Tests

1. **Agent Card**: `/.well-known/a2a`
2. **Message Handler**: POST `/message` (JSON-RPC)
3. **Intent Mandate**: `ap2/createIntentMandate`
4. **Cart Mandate**: `ap2/createCart`
5. **OTP Flow**: `ap2/initiatePayment` → `submitOtp`

Code: `ap2.py` (727 lines)

x402 Flow: Micropayments ⚡

HTTP 402 + EIP-712 Signatures (Coinbase)



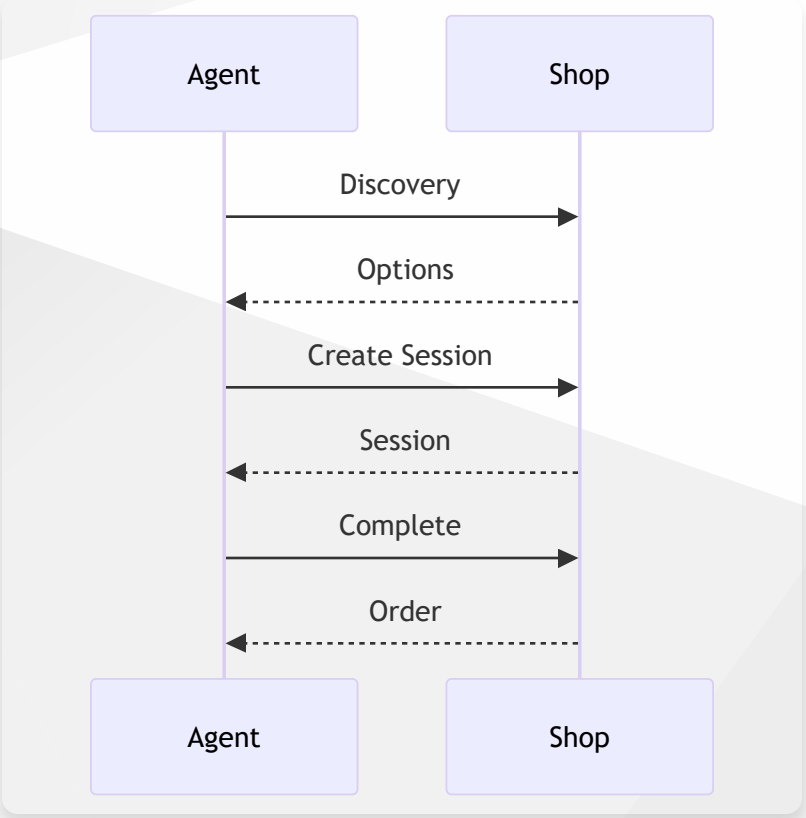
APS Tests

1. **Request Resource**: GET `/resource/{id}`
2. **402 Response**: PaymentRequired header
3. **Sign Payment**: EIP-712 + EIP-3009
4. **Retry**: GET + `X-PAYMENT` header
5. **Facilitator**: `/verify`, `/settle`

Code: `x402.py` (524 lines)

ACP Flow: E-commerce

OpenAPI Checkout + Fulfillment (Shopify/OpenAI)



APS Tests

1. **Discovery**: `/.well-known/checkout`
2. **Create Session**: POST `/checkout_sessions`
3. **Add Fulfillment**: Address + options
4. **Complete**: POST `/complete`
5. **API Version**: `API-Version: 2026-01-16`

Code: `acp.py` (340 lines)

Demo Mode: GitHub Pages 🌐

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

The Challenge

GitHub Pages = No Server.
API calls usually fail.

The Solution

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

```
// frontend/src/services/api.ts
const IS_DEMO = window.location.hostname
  .includes('github.io');

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

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

Result

A frictionless live demo for recruiters.

Why a Product Owner Built This 👨‍💼

"You wrote 2,000+ lines of code. Aren't you a PM?"

My Philosophy

1. **Build to Understand**: I prototype to deeply 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 (AP2, x402, ACP, UCP)
- I can implement working software (FastAPI, React)
- I can document thoroughly (8 docs, 3 ADRs)

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

Let's Connect 🤝

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

- 🌐 **Portfolio**: sivasub.com
- 💼 **LinkedIn**: linkedin.com/in/sivasub987
- 💻 **Code**: GitHub/APS
- 📚 **Docs**: Documentation

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