User Stories & On-Chain Requirements

~ Ayush Soni

Part A: Initial User & Function Mapping

1. Manual User Brainstorming

SYPM Recap:
"Spend Your Portfolio" (SYPM) is a Solana-native app that lets users spend SPL tokens and NFTs directly at Solana merchants — via real-time token swap and checkout integration.

Potential User Types:

Direct Users:

- Everyday Solana users (retail buyers)
- DeFi-native users (with token holdings)
- NFT holders (who want to spend NFT-based value)
- Mobile-first users using Phantom/Solflare

Indirect Users / Beneficiaries:

- Solana ecosystem merchants
- DAO treasuries who disburse tokens
- Token project teams (more token utility = more adoption)

Admins / Moderators:

- Platform developers
- Backend maintainers
- KYB/Compliance team for merchant onboarding

Stakeholders:

- Solana DEX integrators (e.g. Jupiter)
- Protocol/Grant partners (e.g. Solana Foundation, MonkeDAO)
- Token protocols (e.g. Jito, Bonk) whose tokens get real-world utility

2. AI-Assisted User Prioritization

Prompt:

> "My project's value proposition is: 'SYPM enables Solana users to spend their token portfolio seamlessly with real-world merchants, converting SPL tokens at checkout through backend integrations.'
> Here's a brainstormed list of potential users: [list above]
> Based on this, which 2–5 user types are most critical for a Proof-of-Concept (POC)? Provide rationale."

AI Suggestions (summarized):

- Everyday Solana users — Core to proving the "spending" experience.
- Solana merchants — Without them, there's nothing to spend on.
- DeFi-native users — They have tokens but limited use cases.
- Platform developers — Required to build/test backend infra.

Final Chosen List:

- Solana retail users
- Solana merchants
- Platform developers

Rationale:
These 3 groups form the transaction triangle — spender, receiver, executor. Each is needed to demonstrate end-to-end flow.

3. Core Function Mapping

Retail Users:

- Connect wallet
- View eligible tokens in wallet (SPL or eligible NFTs)
- Select a product/service
- Choose token(s) or NFTs to pay

- Confirm swap quote
- Authorize transaction
- Receive confirmation

Supports payment via SPL tokens or eligible NFTs. NFTs will be evaluated based on oracle or floor price index.

Solana Merchants:

- Register as a merchant
- Set token acceptance preferences (SPLs or fallback USDC)
- Generate checkout request / QR
- View payment confirmation

Platform Developers:

- Integrate with token swap backend (e.g., Jupiter)
- Listen to wallet events
- Trigger swap + transfer
- Record payment metadata

4. Deriving Core POC Requirements

Top 2 Critical User Interactions for POC:

- User pays with any SPL token (or NFT) at checkout → receives confirmation
- Merchant receives swapped USDC (or preferred token) automatically

Initial Technical Requirements:

- Smart contract to accept user token input
- Route swap via Jupiter (off-chain or via CPI)
- Transfer USDC (or preferred token) to merchant
- Store transaction metadata (item, merchant ID, method)
- Emit confirmation logs

Users can choose equal splits, prioritize tokens, or use auto-mode based on least slippage.

Part B: Adversarial Analysis & Granularity Check

AI Critique Prompt:

> "Review my core user functions [above] and POC requirements. Do these stories truly align with the value prop? Are they granular enough to design technical architecture? What's unclear or missing?"

AI Critique Summary:

- Strong alignment with MVP goal
- Some interactions (e.g., “pay with SPL”) need to be broken down:
    - How do we know the token's swap route?
    - What if token has no liquidity?
- Lacks merchant authentication layer
- Tx metadata design is missing granularity

Refinements:

User Stories (After Critique):

Before:
→ User pays with SPL token → Merchant receives USDC

After (Granular):
→ User selects token → App checks swap path →
→ App confirms price → User confirms tx →
→ Token sent to swap handler →
→ USDC transferred to merchant →
→ Confirmation stored and sent

Technical Requirements (Refined):

- Token validation & swap quote fetch
- Swap handler via Jupiter backend
- On-chain escrow account to receive/disburse
- Transaction log recording (tx ID, timestamp, merchant ID)
- Merchant registry (whitelisted wallet PDAs)
- Fallback handling if swap fails or liquidity is unavailable

Part C: Granularity & Clarity Refinement

Finalized Atomic User Stories

Retail User:

- Connect wallet
- View SPL token and NFT balances
- Select product and token(s) or NFT
- App validates swap path & quote
- User confirms transaction
- Tokens locked → swap executed
- Confirmation sent

Merchant:

- Register via wallet
- Set accepted tokens / fallback USDC
- Generate QR code or payment session
- View incoming payments + confirmations

Part C Refinement Log (Examples):

Before Story	After Story	Rationale
"User pays with token"	Select token → Confirm quote → Authorize tx	Ensures clarity, atomicity
"Merchant receives payment"	Register merchant → View payment history	Clarifies full merchant flow
"Confirmation sent"	Platform stores tx metadata → Emits log to indexer/backend	Enables data tracking + auditability

Part D: On-Chain Requirements

User Story 1: User selects token & pays

- Verify token has valid swap route (via Jupiter API)
- Fetch quote & validate slippage threshold
- Lock token in escrow PDA
- On approval, execute swap (via CPI or off-chain router)
- Transfer swapped token (e.g., USDC) to merchant
- Emit event with tx metadata (user, token, amount, merchant ID)

If swap fails, transaction is aborted atomically and tokens remain escrowed for retry or refund.

User Story 2: Merchant registers & receives

- Create merchant registry (PDA per wallet)
- Store accepted tokens & fallback preference
- Generate session ID for checkout
- Map payment to merchant registry entry
- Route funds to merchant on success

User Story 3: Platform logs

- Log each payment on-chain or via indexer (e.g. Helius, Triton)
- Store tx metadata: timestamp, tx ID, merchant ID, token used
- Build off-chain dashboard to query logs and view merchant analytics