Assignment 2: User Stories & On-Chain Requirements

Part A: Priority Users for POC

Selected Priority Users (4 Core Types):

1. **CS:GO Players** - Primary target market for skin-to-stablecoin trading
2. **Crypto-Native Gamers** - Early adopters with both gaming and crypto knowledge
3. **Professional Traders/Arbitrageurs** - Provide liquidity and validate trading mechanisms
4. **Platform Developers** - Essential for POC development and iteration

Selection Rationale: These four users represent the minimal viable ecosystem for POC validation - core users (CS:GO Players), technical early adopters (Crypto-Native Gamers), market makers (Professional Traders), and system builders (Platform Developers).

Core Functions by User Type

CS:GO Players:

- Connect Steam account and verify skin ownership
- List skins for sale with stablecoin pricing
- Browse and purchase skins using stablecoins
- Transfer skins between Steam and on-chain wallet

Crypto-Native Gamers:

- Connect multiple wallets and manage stablecoin conversions
- Access advanced trading features and automation
- Participate in governance decisions

Professional Traders/Arbitrageurs:

- Access real-time market data and price feeds
- Execute high-frequency trades with minimal slippage
- Set up automated trading strategies and bots

Platform Developers:

- Deploy and upgrade smart contracts

- Monitor system performance and security
- Manage escrow mechanisms and fee structures

Core POC Requirements

Critical User Interactions for POC:

1. **CS:GO Player lists and sells a skin for stablecoins**
2. **Crypto-Native Gamer purchases a skin using stablecoins**

Technical Requirements:

Blockchain/Smart Contract:

- Solana program for escrow and trade settlement
- NFT minting for on-chain skin representation
- USDC stablecoin payment processing
- Multi-signature wallet for fund custody

Integration:

- Steam API for skin verification
- Wallet connectivity (Phantom/Solflare)

Core Platform:

- User authentication and wallet linking
- Marketplace UI for listing/browsing
- Trading interface with escrow
- Transaction history system

Part B: Refined Requirements

Key Improvements After AI Critique:

Enhanced User Functions:

- Steam account connection broken into: authentication → verification → inventory sync → wallet linking
- Added onboarding flows for crypto newcomers
- Included error handling and transaction status updates
- Added dispute resolution mechanisms

Technical Requirements Refinement:

Solana Program Architecture:

- Marketplace program with listing, bidding, settlement instructions
- Escrow program with time-locks and dispute mechanisms
- NFT mint authority with metadata standards
- Fee collection and distribution system
- Multi-signature governance functions

Missing Components Added:

- Identity verification for compliance
- Gas fee management and optimization
- Real-time WebSocket connections
- Dispute resolution governance
- Regulatory compliance framework

Part C: Final Atomic User Stories

CS:GO Players:

1. User clicks connect Steam button
2. User completes Steam login
3. System reads user's skin inventory
4. User confirms which skins to link
5. User selects skin to sell
6. User sets price in digital coins
7. User confirms listing

Crypto-Native Gamers:

1. User browses available skins
2. User selects skin to buy
3. User approves payment from wallet
4. User receives skin confirmation
5. User connects wallet to platform
6. User switches between coin types

Professional Traders:

1. User views current skin prices
2. User sees price change alerts
3. User sets up buy/sell rules

Platform Developers:

1. Developer uploads new contract code
2. Developer activates contract update
3. Developer monitors system health

Part D: On-Chain Requirements by User Story

CS:GO Players:

"System reads user's skin inventory"

- Create user account on-chain storing Steam ID
- Store skin inventory hash for validation

"User confirms which skins to link"

- Mint NFTs for selected skins with metadata
- Link NFT to original Steam item ID
- Store owned NFT addresses in user account

"User selects skin to sell"

- Create marketplace listing account
- Store NFT mint address and seller wallet
- Store asking price in USDC units

"User confirms listing"

- Transfer NFT to escrow program address
- Set listing status to "active"
- Emit marketplace indexing event

Crypto-Native Gamers:

"User approves payment from wallet"

- Execute purchase instruction
- Transfer USDC to escrow account
- Create purchase record linking buyer to listing

"User receives skin confirmation"

- Complete trade settlement instruction

- Transfer NFT from escrow to buyer
- Transfer USDC to seller minus fees
- Create transaction history record

"User connects wallet to platform"

- Create/update user profile account
- Store wallet public key and preferences

Professional Traders:

"User views current skin prices"

- Oracle program for price data aggregation
- Price history account for historical data

"User sets up buy/sell rules"

- Create automated trading rule account
- Store trigger conditions and execution parameters

Platform Developers:

"Developer uploads new contract code"

- Program deployment using upgradeable loader
- Authority verification for upgrade permissions

"Developer monitors system health"

- System metrics account for volumes and fees
- Error logging for failed transactions

Process Appendix 2: User Stories & On-Chain Requirements

Part A

Task 1

Direct Users

- **CS:GO Players**: The primary users trading skins for stablecoins.
- **Gamers from Other Titles**: As cross-platform support grows, players from other games with tradable assets.
- **Crypto-Native Gamers**: Early adopters familiar with wallets, NFTs, and stablecoins.
- **NFT Collectors/Traders**: Users interested in collecting, flipping, or investing in gaming NFTs.
- **Content Creators/Streamers**: Influencers who showcase trades, review skins, or promote the platform.
- **Professional Traders/Arbitrageurs**: Users seeking profit from price differences across platforms.
- **Game Developers/Studios**: Integrating their games or assets into the marketplace.

Indirect Users / Beneficiaries

- **Viewers/Fans**: Audiences of streamers and content creators who are exposed to the platform.
- **Game Publishers**: Benefit from increased engagement and secondary market activity.
- **Guilds/Esports Teams**: Organize, sponsor, or facilitate trades for their members.
- **NFT Project Owners**: Projects whose assets are listed or traded on the marketplace.
- **Third-Party Analytics Providers**: Platforms offering data, price tracking, or market insights.

Administrators / Moderators

- **Platform Developers**: Responsible for building, maintaining, and upgrading the marketplace.
- **Community Moderators**: Oversee forums, Discord, and social channels to ensure healthy engagement.
- **Support Staff**: Handle user queries, disputes, and technical issues.
- **Smart Contract Auditors**: Ensure the security and integrity of escrow and trading contracts.
- **Compliance Officers**: Oversee KYC/AML processes and regulatory adherence.

Stakeholders

- **Token Holders**: If the platform issues a governance or utility token, these holders have a vested interest.
- **Investors/VCs**: Entities that have funded the project and seek its growth and profitability.
- **Partners/Integrators**: Wallet providers, stablecoin issuers, or other platforms integrated with the marketplace.
- **Advertisers/Sponsors**: Brands or projects seeking exposure to the gaming and crypto audience.
- **Solana Ecosystem Participants**: The broader Solana community benefits from increased network activity and adoption.
- **Regulators**: Interested in compliance and the legal operation of the marketplace.
- **Payment Processors/Stablecoin Providers**: Entities facilitating fiat on/off-ramps or stablecoin transactions.

Task 2: AI-Assisted User Prioritization

AI Analysis for POC Priority Users:

Based on a gaming skin-to-stablecoin marketplace on Solana, the 4 most critical user types for an initial POC are:

1. **CS:GO Players** - Core target market representing the primary value proposition of skin-to-stablecoin trading
2. **Crypto-Native Gamers** - Early adopters who understand both gaming assets and crypto, ideal for initial testing and feedback
3. **NFT Collectors/Traders** - Bridge users who bring trading expertise and can validate marketplace functionality
4. **Platform Developers** - Essential for building, maintaining, and iterating on the POC based on user feedback

Manual Decision & Rationale:
I agree with prioritizing CS:GO Players and Crypto-Native Gamers as they represent the core value proposition validation. I also agree with Platform Developers as essential for POC execution. However, I would consider Professional Traders/Arbitrageurs over NFT Collectors for POC as they provide liquidity and stress-test the trading mechanisms. The final selection focuses on users who can both validate core functionality and provide essential POC execution capabilities.

Task 3: Core Function Mapping

AI Analysis - Key Functions by User Type:

CS:GO Players:

- Connect Steam account to verify skin ownership
- List skins for sale with stablecoin pricing
- Browse and search available skins from other players
- Purchase skins using stablecoins from their wallet
- Transfer skins between Steam and on-chain wallet
- View transaction history and trade confirmations

Crypto-Native Gamers:

- Connect multiple wallets (Phantom, Solflare, etc.)
- Convert between different stablecoins (USDC, USDT)
- Set up automated trading rules/alerts
- Access advanced trading features (limit orders, bulk operations)
- Participate in governance/platform decisions
- Provide liquidity for trading pairs

Professional Traders/Arbitrageurs:

- Access real-time price feeds and market data
- Execute high-frequency trades with minimal slippage
- Set up arbitrage bots and automated strategies
- Access advanced charting and analytics tools
- Manage large-volume transactions with reduced fees
- Monitor cross-platform price differences

Platform Developers:

- Deploy and upgrade smart contracts
- Monitor system performance and transaction volumes
- Implement security measures and fraud detection
- Manage escrow mechanisms and dispute resolution
- Configure fee structures and reward systems
- Maintain integration with external APIs (Steam, price oracles)

Task 4: Deriving Core POC Requirements

Manual Task - Top 1-2 Critical User Interactions:

1. **CS:GO Player lists and sells a skin for stablecoins** - Core value proposition validation
2. **Crypto-Native Gamer purchases a skin using stablecoins** - Completes the trading loop

AI Analysis - Key Technical Requirements for POC:

Based on these two critical interactions, the key technical requirements are:

Blockchain/Smart Contract Requirements:

- Solana program to handle escrow and settlement of trades
- NFT minting capability to represent skins on-chain
- Stablecoin payment processing (USDC integration)
- Multi-signature wallet for secure fund custody
- Transaction fee management and revenue collection

Integration Requirements:

- Steam API integration for skin verification and ownership
- Wallet connectivity (Phantom/Solflare adapters)

Core Platform Requirements:

- User authentication and wallet linking system
- Marketplace UI for listing/browsing skins
- Trading interface with escrow visualization
- Transaction history and confirmation system
- Basic admin panel for monitoring and support

Part B

Task 1: Critique & Refine User Stories / Requirements

AI Critique Analysis:

Value Proposition Review: Gaming skin-to-stablecoin marketplace on Solana enabling seamless trading between traditional gaming assets and crypto.

User Functions/Stories Critique:

- **Missing Granularity:** Functions like "Connect Steam account" need breakdown into specific steps (authentication, verification, linking)
- **Technical Gaps:** Missing error handling, dispute resolution, and edge cases
- **User Experience Gaps:** No onboarding flow for crypto newcomers, missing transaction status updates
- **Security Concerns:** Insufficient detail on custody models and fraud prevention

Requirements Granularity Issues:

- **Blockchain Requirements:** Need specific Solana program architecture (accounts, instructions, state management)
- **Integration Requirements:** Missing API endpoint specifications, authentication flows, data schemas
- **Platform Requirements:** Need specific database design, caching strategies, real-time update mechanisms

Missing Critical Components:

- Identity verification system for compliance
- Dispute resolution mechanism and governance
- Gas fee management and optimization
- Cross-chain bridge considerations for future expansion
- Regulatory compliance framework (KYC/AML)

Refined Technical Requirements:

Solana Program Architecture:

- Marketplace program with listing, bidding, and settlement instructions
- Escrow program with time-locked releases and dispute mechanisms
- NFT mint authority management with metadata standards
- Fee collection and distribution program
- Multi-signature governance for admin functions

API Layer Specifications:

- Steam OAuth integration with skin inventory sync
- Real-time WebSocket connections for price updates
- REST API for user management and transaction history
- GraphQL endpoint for complex marketplace queries
- Webhook system for external integrations

Part C

Part C Refinement Log

Original User Stories → Refined Atomic Stories:

Before: "Connect Steam account to verify skin ownership"
After: Split into 4 atomic stories:

1. "User clicks connect Steam button"
2. "User completes Steam login"
3. "System reads user's skin inventory"
4. "User confirms which skins to link"
   Rationale: Split for atomicity - each represents single action

Before: "List skins for sale with stablecoin pricing"
After: Split into 3 atomic stories:

1. "User selects skin to sell"
2. "User sets price in stablecoins"
3. "User confirms listing"
   Rationale: De-jargoned "stablecoin pricing" and split into distinct steps

Before: "Purchase skins using stablecoins from their wallet"
After: Split into 4 atomic stories:

1. "User browses available skins"
2. "User selects skin to buy"
3. "User approves payment from wallet"
4. "User receives skin confirmation"
   Rationale: Split complex purchase flow into single actions

Before: "Access real-time price feeds and market data"
After: Refined to:

1. "User views current skin prices"
2. "User sees price change alerts"
   Rationale: Removed jargon "feeds/market data", made outcome clear

Before: "Deploy and upgrade smart contracts"
After: Split into 2 atomic stories:

1. "Developer uploads new contract code"

2. "Developer activates contract update"
   Rationale: Split deployment from upgrade, single actions

Final Atomic User Stories:

CS:GO Players:

1. User clicks connect Steam button
2. User completes Steam login
3. System reads user's skin inventory
4. User confirms which skins to link
5. User selects skin to sell
6. User sets price in stablecoins
7. User confirms listing

Crypto-Native Gamers:

1. User browses available skins
2. User selects skin to buy
3. User approves payment from wallet
4. User receives skin confirmation
5. User connects wallet to platform
6. User switches between coin types

Professional Traders:

1. User views current skin prices
2. User sees price change alerts
3. User sets up buy/sell rules

Platform Developers:

1. Developer uploads new contract code
2. Developer activates contract update
3. Developer monitors system health

Part D

Part D: On-Chain Requirements for Each User Story

CS:GO Players:

Story 1: "User clicks connect Steam button"

- No on-chain requirements (frontend UI action only)

Story 2: "User completes Steam login"

- No direct on-chain requirements (external Steam OAuth)

Story 3: "System reads user's skin inventory"

- Program instruction to create user account on-chain
- Account must store Steam ID for verification
- Account must store skin inventory hash for validation

Story 4: "User confirms which skins to link"

- Program instruction to mint NFTs for selected skins
- Each NFT account must store skin metadata (name, rarity, condition)
- NFT account must link to original Steam item ID
- User's main account must store list of owned NFT addresses

Story 5: "User selects skin to sell"

- Program instruction to create marketplace listing account
- Listing account must store NFT mint address
- Listing account must store seller's wallet address
- Listing account must store asking price in lamports (USDC)

Story 6: "User sets price in digital coins"

- Program instruction to update listing account price field
- Price validation to ensure positive value
- Price must be stored in smallest USDC units (micro-USDC)

Story 7: "User confirms listing"

- Program instruction to activate the listing
- Transfer NFT ownership to escrow program-derived address
- Set listing status to "active" in listing account
- Emit event log for marketplace indexing

Crypto-Native Gamers:

Story 1: "User browses available skins"

- No direct on-chain requirements (query existing listing accounts)
- Need indexing program to aggregate active listings

Story 2: "User selects skin to buy"

- No on-chain requirements (frontend selection)

Story 3: "User approves payment from wallet"

- Program instruction to execute purchase
- Transfer USDC from buyer to escrow account
- Verify buyer has sufficient USDC balance
- Create purchase record account linking buyer to listing

Story 4: "User receives skin confirmation"

- Program instruction to complete trade settlement
- Transfer NFT from escrow to buyer's wallet
- Transfer USDC from escrow to seller (minus fees)
- Update listing status to "sold"
- Create transaction history record account

Story 5: "User connects wallet to platform"

- Program instruction to create or update user profile account
- Account must store wallet public key
- Account must store user preferences and settings

Story 6: "User switches between coin types"

- Program instruction to swap tokens via integrated DEX
- Slippage protection mechanisms
- Update user's token balances in profile account

Professional Traders:

Story 1: "User views current skin prices"

- Oracle program to aggregate price data from multiple sources
- Price history account to store historical data
- Real-time price feed account updates

Story 2: "User sees price change alerts"

- Program instruction to create alert subscription account
- Account stores price thresholds and notification preferences
- Event emission when price thresholds are crossed

Story 3: "User sets up buy/sell rules"

- Program instruction to create automated trading rule account
- Account stores trigger conditions (price, time, volume)
- Account stores execution parameters (amount, limits)
- Automated execution program to monitor and execute rules

Platform Developers:

Story 1: "Developer uploads new contract code"

- Solana program deployment using upgradeable loader
- Program buffer account to store new bytecode
- Authority verification for upgrade permissions

Story 2: "Developer activates contract update"

- Program upgrade instruction to replace current program
- Migration logic to handle state transitions
- Version tracking in program data account

Story 3: "Developer monitors system health"

- System metrics account storing transaction volumes, fees collected
- Error logging account for debugging failed transactions
- Performance monitoring through program event emissions