

User Stories & On-Chain Requirements

YieldPay: Debt-Free Spending Protocol

Name: Love Gupta

Github : lovegupta112

Project: YieldPay - Solana-based yield-powered spending platform

Part A: User Stories & On-Chain Requirements Document

1. Core User Personas

Primary Persona: Active Crypto Staker

An individual who holds SOL tokens long-term and wants to make purchases without selling their assets. This user is comfortable with blockchain wallets and understands basic DeFi concepts. They represent the primary revenue source and core user of the platform.

Secondary Persona: Crypto-Native Merchant

A business owner or service provider who accepts cryptocurrency payments. They need reliable, upfront payment without dealing with credit risk or chargebacks. This user completes the payment loop and validates merchant-side value.

Supporting Persona: Platform Administrator

The developer who maintains the protocol, monitors system health, and can update parameters when necessary. While not part of the core user flow, this role is essential for platform operations.

2. Function Maps

Active Crypto Staker Functions

- Deposit SOL into the protocol
- View accumulated yield balance
- Make purchases using yield
- Withdraw yield separately from principal
- Withdraw entire principal to unstake

Crypto-Native Merchant Functions

- Register business profile with wallet address
- Receive SOL payments from customer yield
- View transaction history

Platform Administrator Functions

- Update protocol parameters like APY rate
 - Monitor total value locked in platform
-

3. User Stories

Story 1: Deposit SOL

As an active crypto staker, when I deposit 20 SOL into the protocol, I expect my principal balance to show 20 SOL and my yield balance to start at 0 SOL.

Acceptance Criteria:

- Principal balance displays 20 SOL
 - Yield balance displays 0 SOL
 - SOL transfers from my wallet to protocol vault
 - Deposit timestamp is recorded
-

Story 2: View Accumulated Yield

As an active crypto staker, when I check my dashboard after staking for 30 days, I expect to see my yield balance updated with earnings based on 7 percent annual rate.

Acceptance Criteria:

- Yield balance shows calculated earnings
 - Principal remains at original 20 SOL
 - Yield updates when page loads
 - Last update time is visible
-

Story 3: Purchase with Sufficient Yield

As an active crypto staker with 2 SOL yield, when I purchase an item costing 1.5 SOL, I expect the payment to process successfully.

Acceptance Criteria:

- My yield decreases from 2 SOL to 0.5 SOL
 - My principal stays at 20 SOL unchanged
 - Merchant receives 1.5 SOL in their wallet
 - Transaction appears in payment history
-

Story 4: Purchase Rejected for Low Yield

As an active crypto staker with only 0.5 SOL yield, when I try to purchase an item costing 1.5 SOL, I expect the transaction to fail with a clear error message.

Acceptance Criteria:

- Error message states insufficient yield balance
 - No SOL is transferred
 - My balances remain unchanged
 - Suggestion to wait for more yield appears
-

Story 5: Merchant Receives Payment

As a crypto-native merchant, when a customer completes a purchase, I expect to receive the SOL payment in my registered wallet immediately.

Acceptance Criteria:

- Payment arrives within 5 seconds
 - Amount matches purchase price exactly
 - My total earnings counter increases
 - Transaction is visible on blockchain
-

Story 6: Withdraw Yield Only

As an active crypto staker with 2 SOL yield, when I withdraw my yield, I expect to receive 2 SOL while my principal remains staked.

Acceptance Criteria:

- 2 SOL transfers to my wallet

- My yield balance resets to 0 SOL
 - My principal stays at 20 SOL
 - Yield continues accumulating on principal
-

Story 7: Withdraw Full Principal

As an active crypto staker, when I withdraw my principal, I expect to receive my full 20 SOL deposit plus any remaining yield.

Acceptance Criteria:

- Total amount transfers to my wallet
 - My staking account closes or resets to zero
 - No further yield accumulates
 - Transaction link provided for verification
-

4. Potential On-Chain Requirements

For Story 1: Deposit SOL

- Need vault account to securely store all user deposits
 - Need user staking account to track individual data
 - User account must store wallet address for ownership
 - User account must store principal amount deposited
 - User account must store yield accumulated so far
 - User account must store deposit timestamp
 - Need function to transfer SOL from user to vault
 - Need function to initialize user account
 - Must validate minimum deposit amount
-

For Story 2: View Accumulated Yield

- Need function to calculate time elapsed since last update
 - Calculation formula: $\text{principal} \times \text{rate} \times \text{time} \div \text{seconds per year}$
 - Interest rate stored as 700 basis points for 7 percent
 - Must update yield amount in user account
 - Must update last calculation timestamp
 - Function runs automatically before transactions
 - Must prevent overflow in calculations
-

For Story 3: Purchase with Sufficient Yield

- Need merchant account to store business information
 - Merchant account must have wallet address
 - Need payment function accepting user, merchant, and amount
 - Function must verify merchant is registered
 - Function must update user yield before checking balance
 - Function must check if user has enough yield
 - Subtract payment from user yield balance
 - Transfer SOL from vault to merchant wallet
 - Create payment record in history
 - Must set maximum payment limit for safety
-

For Story 4: Purchase Rejected for Low Yield

- Payment function must validate yield balance first
 - If insufficient, stop processing immediately
 - Return error with current yield and required amount
 - Transaction must cancel without state changes
 - Define clear error types
 - Frontend receives error details for display
-

For Story 5: Merchant Receives Payment

- Payment function must transfer SOL atomically
 - Reduce vault balance by payment amount
 - Increase merchant wallet balance by same amount
 - Transfer must be all-or-nothing operation
 - Emit event for frontend notification
 - Event includes merchant address, customer address, amount, timestamp
-

For Story 6: Withdraw Yield Only

- Need yield withdrawal function
- Function accepts user account and amount
- Verify amount does not exceed available yield
- Update yield balance before withdrawal
- Transfer SOL from vault to user wallet
- Subtract withdrawn amount from user yield
- Keep principal unchanged
- Log withdrawal event
- Allow multiple partial withdrawals

For Story 7: Withdraw Full Principal

- Need principal withdrawal function
 - Update yield one final time
 - Calculate total: principal plus remaining yield
 - Transfer full amount from vault to user wallet
 - Reset user account to zero or close account
 - Log principal withdrawal event
 - Return account rent if closing
-

Part B: Process Appendix

Part A: Initial User & Function Mapping

1) Manual User Brainstorming

Based on my refined value proposition from Assignment 1, I brainstormed potential user types across four categories:

Direct Users:

- Active crypto stakers with long-term holdings
- DeFi yield farmers
- Long-term crypto investors
- Crypto holders wanting to preserve principal
- Subscription service consumers

Indirect Users/Beneficiaries:

- Crypto-native merchants
- Online service providers
- Digital goods sellers
- Staking protocol providers

Administrators/Moderators:

- Platform developer and maintainer
- Smart contract auditors
- System monitors

Stakeholders:

- Future governance token holders

- Protocol partners
- Early investors

This preliminary list included 12 distinct user types.

2) AI-Assisted User Prioritization

AI Prompt Used: "My project's value proposition is: YieldPay allows users to make purchases without selling crypto while retaining investment exposure. Yield from staked tokens funds payments automatically, creating a debt-free spending model. Here is my brainstormed list of all potential user types: Active crypto stakers, DeFi yield farmers, long-term crypto investors, crypto holders wanting to preserve principal, subscription service consumers, crypto-native merchants, online service providers, digital goods sellers, staking protocol providers, platform developer, smart contract auditors, system monitors, future governance token holders, protocol partners, early investors. Based on the value proposition, which 2-5 of these user types are the most critical to focus on for an initial Proof-of-Concept? For each user you recommend, provide a brief rationale explaining why they are essential for proving the project's core value."

AI Response Summary:

The AI recommended focusing on 3 critical user types:

1. **Active Crypto Stakers:** They are the primary users who deposit funds and initiate purchases. Without them, the platform has no transaction volume or revenue. They validate the core value of spending without selling.
2. **Crypto-Native Merchants:** Essential to complete the payment flow. Users need a destination to spend their yield. Merchants validate that the payment model works for sellers and demonstrate instant settlement benefits.
3. **Platform Administrator:** Necessary for maintaining the protocol, updating parameters, and handling edge cases. While not part of core user flow, this role is needed to keep the system operational.

The AI noted that other users like governance token holders and protocol partners are important for scaling but not essential for proving the fundamental mechanism works.

My Analysis and Final Decision:

I agreed with the AI's recommendation to focus on Active Crypto Stakers and Crypto-Native Merchants as these two personas represent the complete transaction loop. A user deposits, earns yield, spends yield, and a merchant receives payment. This validates the core value proposition.

I also agreed with including Platform Administrator as a supporting persona since even a POC needs maintenance capabilities, though I deprioritized admin functions in the user stories to focus on the primary flow.

I disagreed with including DeFi-Curious Users as a separate persona for the POC. While they represent future growth, Active Crypto Stakers already validate the same user journey. Adding another user persona would not prove additional value for the POC and would dilute focus.

Final Prioritized List:

1. Active Crypto Staker (Critical)
2. Crypto-Native Merchant (Critical)
3. Platform Administrator (Supporting)

This decision focuses the POC on proving the essential payment loop while maintaining necessary operational controls.

3) Core Function Mapping

AI Prompt Used: "For a project with this value proposition: YieldPay allows users to make purchases without selling crypto while retaining investment exposure through automated yield payments, and focusing on these prioritized user types: Active Crypto Staker, Crypto-Native Merchant, Platform Administrator - help map out the key functions or interactions each user would need to perform."

AI Response:

The AI mapped these functions by user:

Active Crypto Staker:

- Deposit SOL to begin earning yield
- Monitor yield accumulation over time
- Execute purchases using accumulated yield
- Withdraw yield while keeping principal staked
- Withdraw full principal to unstake completely
- View transaction history

Crypto-Native Merchant:

- Register business with payment details
- Receive SOL payments from customers
- View payment history and earnings
- Withdraw accumulated earnings

Platform Administrator:

- Update yield percentage rate
- Pause protocol for emergencies
- View platform metrics like total value locked
- Adjust system parameters

My Refinement:

I reviewed the AI's suggestions and simplified the function list by removing less critical items for the POC. I removed "view transaction history" for users since it's not essential to prove core functionality. I also removed most admin functions except the ability to update parameters, as the POC should focus on user flow.

The refined function map is documented in Section 2 of Part A.

4) Deriving Core POC Requirements

Manual Analysis:

From the function map, I identified the top 2 most critical interaction paths:

1. User deposits SOL, yield accumulates automatically, user purchases item with yield
2. Merchant receives payment instantly in their wallet

These two paths prove the fundamental value: spending yield without touching principal, and merchants getting guaranteed payment.

AI Prompt Used: "Based on these top 2 critical user interactions: (1) user deposits SOL and yield accumulates automatically over time then user purchases item using only yield, and (2) merchant receives payment instantly in wallet - what are the key technical requirements needed to build a proof-of-concept?"

AI Response Summary:

The AI identified these requirements:

- Blockchain accounts to store user deposit and yield separately
- Secure vault to hold all user funds
- Time-based calculation for automatic yield growth
- Payment processing mechanism with balance validation
- Transfer function from vault to merchant wallet
- Merchant registration system to track verified sellers
- Account structure to prevent mixing principal with yield

Initial Technical Requirements List:

- User staking account with separate principal and yield fields
- Vault account for secure deposit storage
- Yield calculation based on time elapsed and fixed APY

- Payment function with sufficient balance check
 - SOL transfer from vault to merchant
 - Merchant account for registration
 - Atomic transaction processing
-

Part B: Adversarial Analysis & Granularity Check

1) Critique & Refine User Stories / Requirements

AI Prompt Used: "Review my core user functions and stories: user deposits SOL and sees principal and yield balances, user views accumulated yield over time, user purchases item with yield when balance is sufficient, user receives error when yield is insufficient, merchant receives payment, user withdraws yield only, user withdraws full principal. Also review my requirements: user staking account, vault account, yield calculation function, payment processing function, merchant account, transfer mechanisms. Considering my project's refined value proposition about debt-free spending through automated yield, do these stories truly hit the mark? Are the requirements granular enough to map to specific technical components like smart contract accounts and functions? What's missing or unclear?"

AI Critique Summary:

Stories Issues:

- Purchase story needs to specify exact balance changes
- Missing story about merchant registration
- Withdrawal stories should clarify what happens to yield accumulation after
- Stories lack specific timeframes or amounts for testing
- No story about viewing transaction on blockchain for transparency

Requirements Issues:

- Need more specificity on account field structure
- Unclear how yield calculation handles per-second precision
- Missing event emission for merchant notifications
- No mention of error handling types
- Transfer mechanism should specify atomic operation requirement
- Need to define basis points for interest rate storage

My Analysis and Refinements:

I agreed with the critique about needing more specificity. My initial stories were too vague for actual implementation testing.

Changes Made to Stories:

- Added specific amounts to acceptance criteria: 20 SOL principal, 2 SOL yield, 1.5 SOL purchase

- Added Story 5 specifically for merchant payment receipt
- Clarified in Story 6 that yield continues accumulating after withdrawal
- Added Story 7 to distinguish full unstaking from yield withdrawal
- Specified timeframes: "within 5 seconds" for payments

Changes Made to Requirements:

- Detailed account structure with specific fields: wallet address, principal, yield, timestamps
- Specified yield calculation using 700 basis points for 7 percent APY
- Added atomic transfer requirement for payment function
- Included event emission for merchant notifications
- Defined validation steps: check registration, update yield, check balance
- Added error type definitions for insufficient balance scenarios

Rationale: These refinements ensure stories are testable with clear pass/fail criteria. Requirements now provide enough detail for technical implementation without prescribing exact code. The granularity level allows a developer to understand what blockchain components are needed.

Part C: Granularity & Clarity Refinement

Part C Refinement Log

Change 1: Before: User can manage their staked funds After: Split into Story 6 (withdraw yield only) and Story 7 (withdraw full principal) Rationale: These are two distinct actions with different outcomes. Atomicity principle.

Change 2: Before: User makes purchase using protocol After: Story 3 (purchase with sufficient yield) and Story 4 (purchase rejected for insufficient yield) Rationale: Must define both success and failure paths. No overlap between stories.

Change 3: Before: System processes payment automatically After: Payment function verifies merchant, updates yield, checks balance, transfers SOL Rationale: Removed jargon "processes automatically" and specified exact sequence. Clarity of action.

Change 4: Before: User views their account balance After: User views accumulated yield balance with last update timestamp Rationale: Single, concrete action. Specified which balance since there are two types.

Change 5: Before: Merchant integrates with platform After: Merchant registers business profile with wallet address Rationale: Removed technical jargon "integrates" and specified exact action needed.

Change 6: Before: Platform validates user credentials After: System checks if user has at least 1.5 SOL yield Rationale: Removed jargon "validates credentials" and used plain language with specific amount.

Overlap Review:

- Story 3 and Story 4 are mutually exclusive scenarios, no overlap
 - Story 6 and Story 7 have different outcomes, no overlap
 - Story 1 and Story 2 are sequential actions, no overlap
 - All stories represent distinct user actions
-

Part D: Defining Potential On-Chain Requirements

Brainstorming Process

For each user story from Part C, I identified what must exist on the blockchain to enable that action. I focused on translating user-facing functionality into technical components.

Pattern followed:

- What account structure stores the data?
- What function executes the action?
- What validations must occur?
- What state changes happen?
- What information gets logged?

All on-chain requirements are documented in Section 4 of Part A. Each story has corresponding technical needs listed in simple, bulleted format that describes what is needed without specifying implementation details.

Summary

This assignment translated the YieldPay value proposition into actionable technical planning:

1. Brainstormed 12 potential user types across all categories
2. Used AI to prioritize down to 3 essential personas for POC
3. Mapped critical functions for each persona
4. Derived 7 core user stories with specific acceptance criteria
5. Identified on-chain requirements for each story

The iterative refinement through adversarial critique revealed gaps in specificity and missing error scenarios. Breaking broad stories into atomic pieces made them testable. The INVEST model principles ensured story quality.

Key insight: User stories must bridge non-technical stakeholder understanding with technical implementation clarity through specific examples and clear outcomes.