

PART A :- Core User Personas

- 1) DeFi Power Users:- Closest to your core value prop; will immediately benefit from unlocking capital.
- 2) DAO Treasuries with Idle Capital :- Validates institutional utility and capital inflow. 3) Yield Farmers :- Validates cross-protocol utility and demand from high-activity users.
- 4) Solana-native Liquid Stakers :- Easiest to onboard due to native asset compatibility (mSOL, jitoSOL).
- 5) Leverage Farmers :- Important for stress-testing capital efficiency and demand-looping potential.

Function Maps

DeFi Power Users:-

- Deposit mSOL/jitoSOL
- Mint stablecoins
- Allocate to yield strategies
- Monitor collateral/health
- Repay & exit

DAO Treasuries:-

- Batch collateral deposits
- Treasury dashboard
- Whitelist strategies
- Mint stablecoins
- Monitor protocol-wide metrics

Stablecoin Liquidity Seekers:-

- Compare borrow rates

- Onboard staked assets
- Mint and use stablecoins
- Track yield & repayments
- Risk alerts

Potential On-Chain Requirements

User Story 1: "User creates a new token project."

Potential On-Chain Requirements:

- A smart contract function to create a new token project account.
- The account must store:
 - Creator's public key (authority).
 - Project metadata (name, symbol, image URI).
 - Project status (e.g., active, closed).
- The project should register a timestamp (created_at).
- Emit an event for off-chain indexing and tracking.

User Story 2: "User uploads and pins metadata (name, image, symbol) to IPFS."

Potential On-Chain Requirements:

- Store the resulting IPFS hash on-chain in the token project account.
- Ensure the hash is immutable after the project is published.

- Provide a function to validate that the metadata exists and is accessible.
- Optional: Add a metadata version number if future updates are allowed.

User Story 3: "User mints tokens to a recipient address."

Potential On-Chain Requirements:

- A mint function to mint a specified number of tokens.
- Only the token project owner can mint.
- Update the total supply counter on-chain.
- Emit a **Transfer** or **Mint** event.
- Ensure the recipient's account exists (or create associated token account if needed).

User Story 4: "User lists their token project on a decentralized registry."

Potential On-Chain Requirements:

- A registry program or account that stores all listed projects.
- Verify that only verified creators can list projects.
- Add an optional approval mechanism (e.g., community upvotes, DAO approval).
- Add filters/sorting fields like category, created_at, token count, etc.

User Story 5: "User contributes funds to a token project or campaign."

Potential On-Chain Requirements:

- A function to accept SOL or SPL token transfer.
- Funds must be routed to a campaign vault account.
- Store contribution records:

- Contributor address.
- Amount.
- Timestamp.
- Update the "raised amount" state variable.
- Prevent funding if the project status is inactive or closed.

PART B :-

Part C Refinement Log (Start your log like this)

Before (Original User Story) After (Refined User Story) Rationale

1 User signs up and links wallet signs up for an account	Story 1: User signs up for an account	Story 2: User links their crypto wallet Split into two stories for atomicity
	metadata	contract,” “dApp UI”)
	User clicks a button on the website to perform an action on the blockchain	
2 User interacts with dApp UI to call smart contract		Split for clarity and better mapping to UI/DB/API
	Admin views a list of submitted proposals	
3 Admin reviews user-submitted proposals and approves/disapproves them	Admin chooses to approve or reject each proposal	Clarified input steps and replaced “metadata” with simpler terms
	Creator enters project name, image, and description, then submits it	
4 Creator submits a project with	Removed jargon (“smart	
5 Buyer mints the token buy a token	Buyer clicks “Mint” to	Simplified and clarified action

Prompt:- My project's value proposition is **A capital-efficient stablecoin protocol** that allows users to deposit low-volatility assets (e.g., mSOL, jitoSOL) as collateral to mint a stablecoin, while still enabling them to deploy that collateral in whitelisted DeFi strategies (like staking, lending, or LPing) — unlocking stablecoin liquidity *without sacrificing yield potential*. Here is a brainstormed list of all potential user types: . 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.

Output:-

Great! Let's go step by step with your **AI-Assisted User Prioritization** task.

◆◆ Step 1: Your Project's Value Proposition

Let's first summarize your project's value prop in one clear sentence:

"**A capital-efficient stablecoin protocol** that allows users to deposit low-volatility assets (e.g., mSOL, jitoSOL) as collateral to mint a stablecoin, while still enabling them to deploy that collateral in whitelisted DeFi strategies (like staking, lending, or LPing) — unlocking stablecoin liquidity *without sacrificing yield potential*."

◆◆ Step 2: Full Brainstormed List of Potential User Types

Here's a representative list based on your earlier discussions and common market

segments: 1. DeFi Power Users (who stake mSOL/jitoSOL and want extra liquidity)

2. Yield Farmers

3. DAO Treasuries with idle capital

4. DeFi Protocols with underutilized treasuries

5. Leverage Farmers

6. Solana-native Liquid Stakers
7. Cross-chain Stablecoin Arbitrageurs
8. On-chain Hedge Funds
9. Retail Users looking for passive income
10. Builders who want composable stablecoins for integrations
11. DeFi aggregators looking for new primitives
12. Institutional Solana funds
13. NFT DAOs with idle SOL
14. LSD Aggregator Protocols
15. Whale Users wanting capital efficiency
16. New DeFi protocols looking for backing liquidity

💡💡 Step 3: AI-Assisted Prioritization (Based on Value

Prop) Here's a suggested prompt you could feed into an AI:

My project's value proposition is: A capital-efficient stablecoin protocol that allows users to deposit low-volatility assets (e.g., mSOL, jitoSOL) as collateral to mint a stablecoin, while still enabling them to deploy that collateral in whitelisted DeFi strategies (staking, lending, LPing). Here is a brainstormed list of all potential user types: [List above]. 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 (POC)? For each user you recommend, provide a brief rationale explaining why they are essential for proving the project's core value.

?? Step 4: AI Likely Response (Simulated Below)

Here's what the AI would likely recommend as the **top 2–5 critical user types for POC**:

✓ 1. DeFi Power Users (mSOL/jitoSOL holders wanting extra liquidity)

- **Rationale:** They are already engaging in staking but are limited by capital lock-up. Proving the protocol's utility for them showcases the ability to unlock capital efficiency without sacrificing yield.

✓ 2. DAO Treasuries with Idle Capital

- **Rationale:** DAOs have large balances of low-volatility assets and strong incentive to put idle capital to work without taking unnecessary risks. If they adopt your protocol, it validates institutional trust and real-world demand.

✓ 3. Yield Farmers

- **Rationale:** They are highly motivated by capital efficiency and want to maximize return across protocols. If they adopt your product, it shows PMF among hyperactive users who often set DeFi trends.

✓ 4. Leverage Farmers

- **Rationale:** These users seek recursive or leveraged positions and would be the first to loop stablecoin liquidity from their staked assets — a strong signal of product demand and protocol stickiness.

✓ 5. Solana-native Liquid Stakers

- **Rationale:** This group already holds assets like mSOL or jitoSOL and represents your easiest entry point, requiring minimal user education. Their adoption proves usability within the Solana ecosystem.

?? Step 5: Your Final Prioritized List + Rationale

After reviewing the AI's suggestions, your final list for your **initial Proof-of-Concept (POC)** might be:

Rank User Type Why You Chose Them

1 DeFi Power Users Closest to your core value prop; will immediately benefit from unlocking capital.

Validates institutional utility and capital inflow.

2 DAO Treasuries with Idle Capital




3 Yield Farmers Validates cross-protocol utility and demand from high-activity users.

Easiest to onboard due to native asset compatibility (mSOL, jitoSOL).

4 Solana-native Liquid Stakers

5 Leverage Farmers Important for stress-testing capital efficiency and demand-looping potential.

?? Summary of Manual Analysis

-  You **agreed** with AI's top picks but maybe dropped "Whale Users" or "NFT DAOs" for now due to niche or lower relevance in a POC.
-  You **focused on users with highest pain point** for capital inefficiency and idle yield, aligned to your protocol's core offering.
-  This prioritization helps you avoid building features for less urgent users (e.g. aggregators or cross-chain users) in early-stage.

Let me know if you want a Notion-ready version of this or want to use these segments in an investor or grant pitch.

Prompt:- For a project with this value proposition and focusing on these prioritized user types DeFi Power Users who stake SOL/mSOL/jitoSOL , help map out the key functions or interactions each user would need to perform

Output:- Great — let's proceed with Core Function Mapping (Step 3 of your AI founder sprint).

✓ Project Value Proposition (summary):

A capital-efficient stablecoin protocol where users deposit low-volatility assets like wSOL, mSOL, jitoSOL as collateral to mint stablecoins *without locking* their assets. These assets can be dynamically used in whitelisted DeFi strategies (lending, staking, LPing), allowing users to earn yield while accessing stablecoin liquidity. The protocol ensures solvency and composability with trusted DeFi protocols.

✓ Prioritized User Types (from AI-Assisted Prioritization):

1. DeFi Power Users who stake SOL/mSOL/jitoSOL
2. DAO Treasuries & Protocols with Idle Capital
3. Stablecoin Liquidity Seekers (Capital-Efficient Borrowers)

❖❖ AI Task Prompt:

For a project with this value proposition — [insert above] — and focusing on these

prioritized user types — [see above list] — help map out the key functions or interactions each user would need to perform.

❖❖ Output — Core Function Mapping:

1. DeFi Power Users (Staking SOL, mSOL, jitoSOL)

Goal: Access stablecoin liquidity while keeping their staked assets productive.

Key Functions:

- Connect wallet & dashboard view of staked assets
- Deposit mSOL, jitoSOL, etc., as collateral
- Mint stablecoins (e.g., USDX) against staked assets
- Choose whitelisted DeFi strategies for yield (auto/staking)
- Monitor collateral health, yield, and liquidation thresholds
- Repay stablecoin loan & withdraw collateral

2. DAO Treasuries & Protocols with Idle Capital

Goal: Utilize treasury assets productively while maintaining liquidity access.

Key Functions:

- Multi-sig wallet support for onboarding
- Batch deposit of SOL-based liquid-staking tokens
- Whitelist custom/internal strategies or choose from ecosystem options •

Mint stablecoins to support protocol operations or fund liquidity

- Monitor risk parameters across protocol/tokens
- Treasury dashboards for real-time tracking & strategy allocation

3. Stablecoin Liquidity Seekers (Capital-Efficient Borrowers)

Goal: Get access to USD-like liquidity without selling yield-generating tokens.

Key Functions:

- Search/compare borrowing rates vs. other stablecoin options •

On-ramp with existing staked tokens or swap SOL → mSOL

- Mint stablecoin and instantly use in DeFi (DEXs, payments, yield) •

Alerts on collateral ratio, yield status, repayment timelines

- Option to auto-compound yield or convert into more staked collateral

Summary Output Format (for your doc):

User Type Core Functions / Interactions

DeFi Power Users - Deposit mSOL/jitoSOL

- Mint stablecoins
- Allocate to yield strategies
- Monitor collateral/health
- Repay & exit

DAO Treasuries - Batch collateral deposits

- Treasury dashboard
- Whitelist strategies
- Mint stablecoins
- Monitor protocol-wide metrics

Stablecoin Liquidity Seekers - Compare borrow rates

- Onboard staked assets
- Mint and use stablecoins

- Track yield & repayments
- Risk alerts

Prompt:- Based on these top 1-2 critical user interactions Depositing liquid staking tokens (mSOL, jitoSOL, etc.) as collateral to mint a stablecoin. Routing deposited collateral into yield-generating strategies while maintaining solvency and liquidity tracking., what are the key technical requirements needed to build a proof-of-concept?

Output:- Great — you're now at **Step 4: Minimum Viable Feature Set of your AI Founder Sprint.**

Starting Point Recap

Top 1–2 Critical User Interactions (based on Step 3):

1. Depositing liquid staking tokens (mSOL, jitoSOL, etc.) as collateral to mint a stablecoin.
2. Routing deposited collateral into yield-generating strategies while maintaining solvency and liquidity tracking.

AI Task Prompt:

Based on these top 1-2 critical user interactions —

1. deposit collateral (mSOL, jitoSOL) to mint stablecoins
2. auto-deploy collateral to whitelisted yield strategies while maintaining solvency —
what are the key technical requirements needed to build a proof-of-concept?

⚙️ Output: Key Technical Requirements (for Proof of Concept)

💡 Smart Contract Layer (on Solana):

- **Collateral Vault Program:**
To accept deposits of whitelisted tokens (mSOL, jitoSOL, wSOL) as collateral.
- **Stablecoin Mint Program:**
Mints a synthetic stablecoin (e.g., USDX) based on collateral value and protocol LTV ratio.
- **Strategy Router/Registry Program:**
Allows deposited collateral to be programmatically routed to whitelisted strategies (e.g., staking, lending pools).
Optional in PoC: hardcode one or two strategies like mSOL staking or jitoSOL rewards.
- **Solvency Checker / Oracle Feed:**
Pulls real-time prices for collateral tokens (e.g., via Pyth or Switchboard) to calculate collateral ratio and trigger liquidation thresholds.
- **Liquidation Logic (optional for PoC):**
Allow third parties (keepers) to liquidate undercollateralized positions.

💡 Frontend (Web UI):

- **Wallet Connect (e.g., Phantom, Backpack)**
To authenticate and sign transactions.
- **Collateral Deposit UI**
Dropdown to select token → input amount → submit transaction.
- **Stablecoin Minting UI**
Show max borrowable USDX → input mint amount → confirm.
- **Dashboard / Position View**
Show deposited collateral, current LTV, accrued yield, stablecoin balance.

- **Basic Alert System (text only)**

Notify users when LTV is approaching risk thresholds (manual simulation or mock).

❖❖ Backend (Optional / for MVP polish)

- **Indexer / Off-chain State Tracker**

To track positions, yields, and update frontend efficiently.

- **Mock Oracle or Integration with Pyth**

To feed token prices to Solana programs in devnet/localnet.

✅ Summary Output Table

Component Description

Collateral Vault Program Accepts mSOL, jitoSOL; tracks user deposits

Stablecoin Mint Program Mints synthetic USDx based on LTV logic

Strategy Router (optional) Allocates collateral to yield strategies

Oracle Feed Integration Fetch price feeds to calculate collateral ratios **Wallet**

Connect (Frontend) Authenticates users via Phantom/Backpack **Minting +**

Deposit UI Enables collateral deposit and stablecoin minting **Dashboard View**

Shows LTV, yields, collateral, mint history

Would you like to proceed to Step 5: Prototype Build Plan?