Part A

Manual User Brainstorming

Direct Users:
- Salary Earners
- Pensioners
- Institution
- Families

Beneficiaries:
- Store Vendors
- Agricultural Partners/ bank partners

Administrators:
Dexzikon Team (managing smart contracts, liquidity and granting store permissions)

Stakeholders:
- Institutional investors
- Agricultural Cooperatives
- Development Partners

Prioritized Users for POC

1. Salary Earners: They're the most immediate target who understand monthly income flow and can lock savings with predictable APY. They validate the real-world utility of using yield for daily needs.

2. Store Vendors: Essential to test the actual redemption of APY in the real world. Without vendor participation, the loop isn't closed. They prove **food access** in the value prop.

3. Families*:* Families are the end consumers benefiting from early food and essentials. Their improved welfare showcases the platform's real-life impact on food security.

4. Dexzikon Admin Team: Needed to set up permissions, process transactions, manage wallet logic, handle vendor approvals. Critical to run the POC backend and smart contract lifecycle.

5. Agricultural Partners / Bank Partners*:* These institutions are critical to realizing the royalty model and supply chain. Their participation enables the reinvestment loop and validates the backend feasibility of the model.

Decision-Making Summary

- A**greed** with the AI's suggestion of salary earners and store vendors due to their immediate role in testing the APY-as-credit concept.
- **Added families** because impact visibility matters early. Demonstrating household benefit proves the social case.

- **Kept agricultural/bank partners** over institutions or investors, since they are the actual economic engine behind Dexzikon's royalty model.

User interactions

Salary Earners (Users/Savers)

Core Functions:

- Lock savings on-chain with a defined APY (via wallet).
- View projected APY and convert it to store credit.
- Redeem store credit to buy food or essentials from vendors.
- Track rewards/royalties earned from agriculture reinvestment.

Store Vendors

Core Functions:

- Accept APY-based store credit as payment.
- Set prices and manage inventory available for purchase through Dexzikon.
- Withdraw earnings into SOL or stablecoin after transactions.
- Get verified/onboarded by the Dexzikon admin.

Families

Core Functions:

- Access food and essentials early using the locked APY of the main saver (e.g. parent).
- Track deliveries or pickups through the vendor network.
- Monitor savings benefits (e.g. improved welfare, food security).
- Authorize trusted wallets (if multiple family members share access).

Dexzikon Admin Team

Core Functions:

- Approve savings deposits and vendor onboarding.
- Monitor smart contract activity and validate transactions.
- Manage wallet logic and user-role permissions.
- Handle dispute resolution and platform integrity.

Agricultural / Bank Partners

Core Functions:

- Receive platform investment to fund agriculture or lending.
- Share returns or royalties back into Dexzikon pools.
- Provide real yield data to support transparent APY tracking.
- Support audits or partnerships for scaling operations.

Top 2 Critical User Interactions

1. Salary Earners:

- Lock savings on-chain with a defined APY via wallet.
- Convert projected APY into store credit and redeem it with vendors.

2. Store Vendors:

- Accept store credit payments from users based on projected APY.
- Withdraw earnings into SOL or stablecoins.

Key Technical Requirements for the POC

1. On-Chain Savings & APY Logic

- Smart contract to lock SOL/SPL savings with time-based tracking.
- Fixed or simulated APY logic to generate store credit over time.
- Function to emit claimable store credit based on locked amount and time elapsed.

2. Store Credit System & Redemption

- Internal non-transferrable store credit ledger mapped to wallet.
- Function to redeem items using earned store credit.
- Vendor-side function to receive credits and log fulfilled orders.

3. Wallet & Role Management

- Solana wallet integration (e.g. Phantom).
- Basic role-based access for Users, Vendors, and Admin.
- Ability for Users to authorize shared family wallets (multi-access support).

4. Admin Dashboard (Manual for POC)

- Interface to approve vendors, simulate APY, and view system activity.
- Manual tools to override balances and resolve disputes.

5. Mock Backend + Yield Feed

- Mocked APY data source (simulating agricultural or lending returns).
- Lightweight backend for inventory, orders, and wallet mappings.

- Basic logging and analytics for testing flows and debugging

Part B

Refined User Stories & Core Functions

User/Saver (Salary Earner)

- Lock funds on-chain using Solana wallet and select desired lock duration.
- View projected APY and store credit equivalent before committing.
- Redeem store credit to purchase listed essentials from approved vendors.
- Track royalty rewards from agricultural returns via dashboard.
- Authorize trusted family wallets for credit access.

Store Vendor

- Accept Dexzikon-issued store credits as payment.
- List and manage essential goods (inventory, pricing, delivery options).
- View completed orders and withdraw earnings in SOL or stablecoins.
- Request platform verification and undergo onboarding via admin.

Family

- Access goods using the main user's unlocked store credit.
- Track delivery status and inventory from vendors.
- View savings impact (e.g. nutrition/food intake timeline, welfare benefits).
- Request access or be invited via main user.

Dexzikon Admin Team

- Approve vendors, savings intents, and family wallet sharing.
- Track smart contract logs, flag errors, and resolve disputes.
- Set APY rules (fixed or simulated) and adjust platform parameters.
- Oversee the backend logic for store credit issuance and role permissions.

Agricultural/Bank Partner

- Receive platform investments to deploy in real-world yield farming.
- Return royalties or gains back into Dexzikon smart contract pools.
- Provide APY data to be used for on-chain yield simulation.
- Participate in audits and performance reviews for transparency.

Refined Technical Requirements & Granular Mapping

1. **Savings Lock + APY Logic (Program)**
- `lockSavings(amount: u64, duration: u64) -> Tx`
- `simulateAPY(amount: u64, duration: u64) -> credit_estimate`
- `claimStoreCredit(wallet: Pubkey) -> Tx`
2. **Store Credit & Redemption System**
- Internal mapping: `wallet => creditBalance`
- `redeemItem(wallet: Pubkey, itemId: string, amount: u64)`
- `logRedemption(txId: string, itemId: string, vendor: Pubkey)`
3. **Wallet Roles & Permissions (Program)**
- Role-based logic: `Role::User`, `Role::Vendor`, `Role::Admin`
- Family sharing: `authorizeFamilyWallet(mainWallet: Pubkey, childWallet: Pubkey)`
4. **Admin Dashboard (Manual POC UI)**

   Components:

   - Vendor Approval Panel
   - Lock Simulation Control
   - Dispute Resolution Button
   - Logs Viewer

   Manual functions: `adjustCredit(wallet, amount)`, `forceWithdraw(vendor)`

5. **Backend + Mock Yield Feed**
- Endpoint: `/api/yield` - returns daily or weekly APY simulation data
- Endpoint: `/api/inventory?vendor=xyz` - fetches available items
- Endpoint: `/api/redemptions?wallet=abc` - fetch user order history

Part C

Dexzikon User Stories (Refined for Clarity & Granularity)

User Segment: Salary Earners (Savers)

1. **User connects their wallet.**

2. **User deposits savings into the Dexzikon smart contract.**
3. **User chooses an APY lock period for their savings.**
4. **User views the projected APY and how much store credit it translates to.**
5. **User uses their store credit to purchase food or essentials.**
6. **User views a list of their past redemptions or purchases.**
7. **User tracks the rewards or royalties generated from agriculture reinvestment.**

User Segment: Store Vendors

1. **Vendor applies for onboarding and verification.**
2. **Vendor gets verified by the Dexzikon admin.**
3. **Vendor sets the price of items they sell on Dexzikon.**
4. **Vendor lists their inventory for users to buy with store credit.**
5. **Vendor receives store credit payments from users.**
6. **Vendor withdraws their earnings in SOL or stablecoins.**

User Segment: Families

1. **Family member connects their wallet.**
2. **Family member gets authorized to access store credit by the main user.**
3. **Family member uses store credit to purchase food or essentials.**
4. **Family member tracks the delivery or pickup of items bought.**

5. **Family monitors welfare benefits from the saving system.**

User Segment: Dexzikon Admin Team

1. **Admin verifies and approves new savings deposits.**
2. **Admin reviews and verifies new vendor applications.**
3. **Admin monitors smart contract activity.**
4. **Admin manages wallet roles and permissions.**
5. **Admin handles user or vendor disputes.**

User Segment: Agricultural / Bank Partners

1. **Partner receives capital from Dexzikon savings pools.**
2. **Partner uses funds for agricultural investments or lending.**
3. **Partner reports real yield data back to the platform.**
4. **Partner shares profits as royalties to Dexzikon.**
5. **Partner provides audit reports for transparency.**

Part C Refinement Log

Before	After	Rationale
"User locks savings and gets APY"	Split into: "User deposits savings" + "User chooses APY lock period"	Split for atomicity

"Vendor gets onboarded and sets up shop"	Split into: "Vendor applies", "Vendor gets verified", "Vendor lists items"	Clear actions and roles
"Family accesses food and monitors benefits"	Split into 3 actions: access, buy, and monitor benefits	Increased granularity
"Admin handles platform integrity and roles"	Split into: monitor contracts, manage permissions, handle disputes	De-jargon and clarity
"Partner supports scaling and gives returns"	Split into 3 steps: receive funds, share returns, provide data	Clarified distinct responsibilities
Removed duplicate story: "User views projected APY and store credit"	Merged into one step: "User views projected APY and how much credit it gives"	Removed redundancy

Part D: Defining Potential On-Chain Requirements

---

1. User Registration

User Story:
A new user joins the platform and connects their wallet to create an on-chain identity.

Potential On-Chain Requirements:

- Generate a Program Derived Address (PDA) tied to the wallet address.
- Store wallet address, registration timestamp, and optional metadata.
- Initialize account state (e.g., registration complete, not yet locked).

---

2. Create and Lock Token Vault

User Story:
The user creates a vault to lock a certain amount of tokens for a fixed duration.

Potential On-Chain Requirements:

- Create a `Vault` account owned by the program and linked to the user PDA.
- Accept transfer of SPL tokens from the user's wallet to the vault account.
- Store:
    - Token mint address.
    - Lock amount.
    - Start and unlock timestamps.
- Set vault state as "locked."

---

3. Claim APY as Goods

User Story:
Once the lock period ends, the user can claim the yield in tokenized form (SPL/NFT goods).

Potential On-Chain Requirements:

- Check if current timestamp > unlock timestamp using `Clock` sysvar.
- Calculate accrued APY based on the vault's lock parameters.
- Mint or transfer the equivalent SPL/NFT goods token to the user.
- Update vault status to "claimed" or "closed."

---

4. Family Member Redeems on Behalf

User Story:
A pre-approved family wallet redeems the user's yield if they're unavailable.

Potential On-Chain Requirements:

- Link one or more guardian/family wallet addresses to the vault at creation.

- Implement access checks for authorized redeemers.
- Allow redemption only after unlock time.
- Transfer goods token to guardian wallet if eligible.

---

5. Admin Monitors and Manages Platform

User Story:
Platform admin oversees vaults, controls partners, and manages token flow.

Potential On-Chain Requirements:

- Admin PDA with elevated access permissions.
- Functions to:
    - Query total value locked (TVL).
    - Pause/unpause vault creation globally.
    - Add or remove farm partners from an allowlist.
- Update vault or partner metadata on-chain when needed.