Assignment 2: User Stories & On-Chain Requirements

Project: ReputeSol

- Prapti Sharma

Part A: Initial User & Function Mapping

1) Manual User Brainstorming

Project Value Proposition (from Assignment 1 recap):
ReputeSol is an on-chain reputation framework that aggregates governance, contribution, and participation data across Solana to generate verifiable reputation scores for wallet addresses. It enables DAOs and dApps to make more trust-based, data-driven decisions and helps users showcase their credibility across the Solana ecosystem.

Brainstormed List of Potential User Types

Direct Users

- DAO contributors — members who perform tasks, vote, or contribute to proposals and earn reputation.
- Developers — integrate ReputeSol APIs or SDKs into their Solana projects.
- Governance participants — individuals voting in on-chain proposals or holding governance tokens.
- dApp builders — who embed reputation badges for users (like login reputation or access control).
- Wallet users — everyday users who wish to view or prove their on-chain activity and credibility.

Indirect Users / Beneficiaries

- DAOs — benefit from improved member credibility and reduced Sybil attacks.
- Protocol teams — gain a stronger understanding of user behavior across the ecosystem.

- Investors / Grant programs — can evaluate applicant reputation transparently.

Administrators / Moderators

- ReputeSol maintainers — manage contract deployments and updates.
- Community moderators — oversee disputes or incorrect score reporting.
- Oracle operators / indexers — fetch and verify governance and contribution data.

Stakeholders

- Solana ecosystem projects — benefit from plug-and-play identity verification.
- Partner protocols — using the reputation API for scoring users.
- Reputation data providers (like Civic, Gitcoin Passport) — potential integrators or collaborators.

---

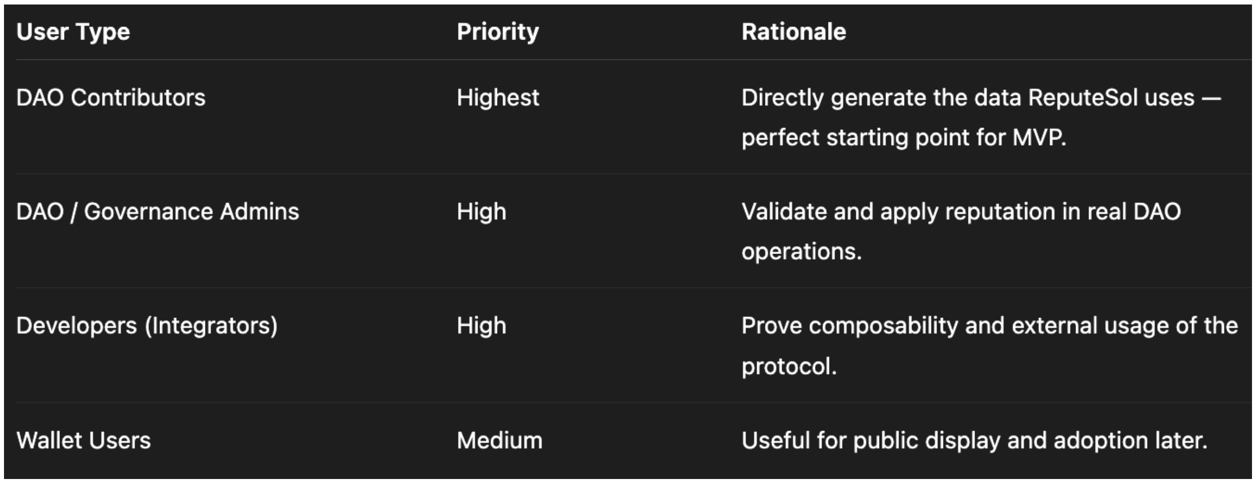
2) AI-Assisted User Prioritization

Project Value Proposition:
ReputeSol is an on-chain reputation framework for the Solana ecosystem. It aggregates wallet-level data (like governance participation, DAO activity, and contribution records) to create a verifiable reputation score. This helps DAOs, dApps, and users build trust and credibility without centralized verification.

AI Recommendations (Synthesized Output)
Based on the MVP and mentor's feedback ("start small with DAO contributor reputation or governance data"), the most critical 3–4 user types for your Proof of Concept (POC) are:

1. **DAO Contributors** – They are the main actors generating the reputation data. Tracking their proposal votes, discussions, and task completions is the first step in proving the framework's utility.
2. **DAO / Governance Admins** – They consume the reputation scores to reward or filter contributors.
3. **Developers (Integrators)** – Represent adoption. They integrate ReputeSol's API or contract logic into governance dashboards or DAO tools.
4. **Wallet Users (Secondary)** – Everyday users who want to view or share their reputation score.

Final Prioritized User List



User Type	Priority	Rationale
DAO Contributors	Highest	Directly generate the data ReputeSol uses — perfect starting point for MVP.
DAO / Governance Admins	High	Validate and apply reputation in real DAO operations.
Developers (Integrators)	High	Prove composability and external usage of the protocol.
Wallet Users	Medium	Useful for public display and adoption later.

Decision Rationale:
DAO contributors and governance admins are most critical, aligning with the mentor's suggestion to start from DAO-based reputation. Developers are essential for adoption, while wallet users provide long-term value for growth.

---

3) Core Function Mapping

Prioritized Users:
DAO Contributors
DAO / Governance Admins
Developers (Integrators)
Wallet Users (Secondary)

User Functions and Interactions

DAO Contributors

- Participate in DAO proposals (vote, create, or discuss proposals).
- Link their Solana wallet to ReputeSol.
- View their current "ReputeScore" derived from on-chain activity.
- Verify their DAO participation data via a simple dashboard.
- Delegate reputation or claim badges/NFTs reflecting milestones.

DAO / Governance Admins

- Query a contributor’s ReputeScore via ReputeSol’s dashboard or API.
- Set DAO-level parameters (e.g., weight of voting participation or proposal success).
- Use ReputeSol scores to filter contributors for bounties or rewards.
- Publish DAO-wide leaderboards or reward top contributors automatically.

Developers (Integrators)

- Access ReputeSol APIs or smart contract endpoints.
- Fetch or display user reputation data in their UI.
- Subscribe to updates when a user’s score changes.
- Contribute data sources or scoring modules (optional advanced phase).

Wallet Users (Secondary Persona)

- Connect wallet to the ReputeSol app to view their score.
- Share a public profile link or NFT badge as proof of credibility.
- Verify score authenticity via on-chain verification.

---

4) Deriving Core POC Requirements

Critical Interaction #1:
DAO Contributor earns and views a ReputeScore based on governance participation.

- Contributor connects wallet to ReputeSol.
- System indexes governance data (e.g., proposal votes, number of proposals created, participation rate).
- Smart contract calculates a ReputeScore based on weighted metrics.
- Contributor can view the score on the ReputeSol dashboard.

Why it matters: Core loop of reputation creation — turning raw on-chain data into a visible, verifiable score.

Critical Interaction #2:
DAO Admin queries contributor scores to identify top contributors or filter reward eligibility.

- DAO Admin accesses ReputeSol dashboard or API.
- Enters a contributor’s wallet address (or entire DAO list).
- ReputeSol contract returns verified score data from the chain.
- DAO Admin uses it to distribute rewards or assign weighted votes.

Why it matters: Core loop of reputation consumption — enabling DAOs to use scores meaningfully.

Technical Requirements for Interaction #1 (Contributor ReputeScore Creation)

- Store wallet addresses mapped to reputation score.
- Function to update scores based on governance data.
- Metadata (timestamps, DAO ID, participation count).
- Verification via on-chain function.
- Off-chain indexer for Realms DAO governance data.
- Weighted score formula and push updates via signed transactions.
- Frontend: Wallet connect + score dashboard.

Technical Requirements for Interaction #2 (DAO Admin Query)

- Read function to query scores by wallet.
- Aggregation function for ranking contributors.
- API endpoint for DAOs.
- Dashboard for querying and leaderboard generation.

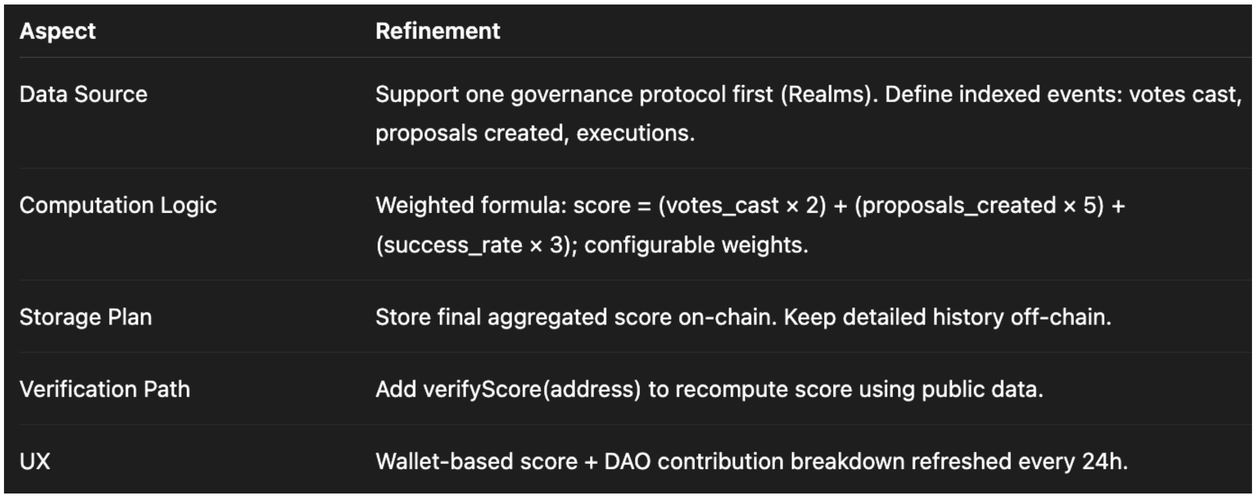
Part B: Adversarial Analysis & Granularity Check

Interaction #1 – Contributor earns and views a ReputeScore

Critique:

- How is data fetched? Realms data lives in multiple governance programs.
- Score updates could be costly.
- How can contributors verify that indexer data is fair?

Refinements:



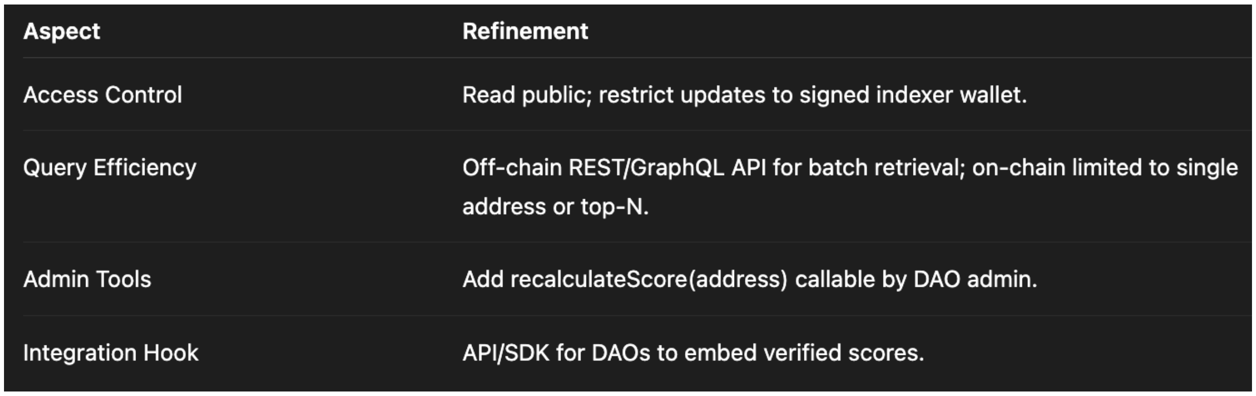
Aspect	Refinement
Data Source	Support one governance protocol first (Realms). Define indexed events: votes cast, proposals created, executions.
Computation Logic	Weighted formula: score = (votes_cast × 2) + (proposals_created × 5) + (success_rate × 3); configurable weights.
Storage Plan	Store final aggregated score on-chain. Keep detailed history off-chain.
Verification Path	Add verifyScore(address) to recompute score using public data.
UX	Wallet-based score + DAO contribution breakdown refreshed every 24h.

Interaction #2 – DAO Admin queries contributor scores

Critique:

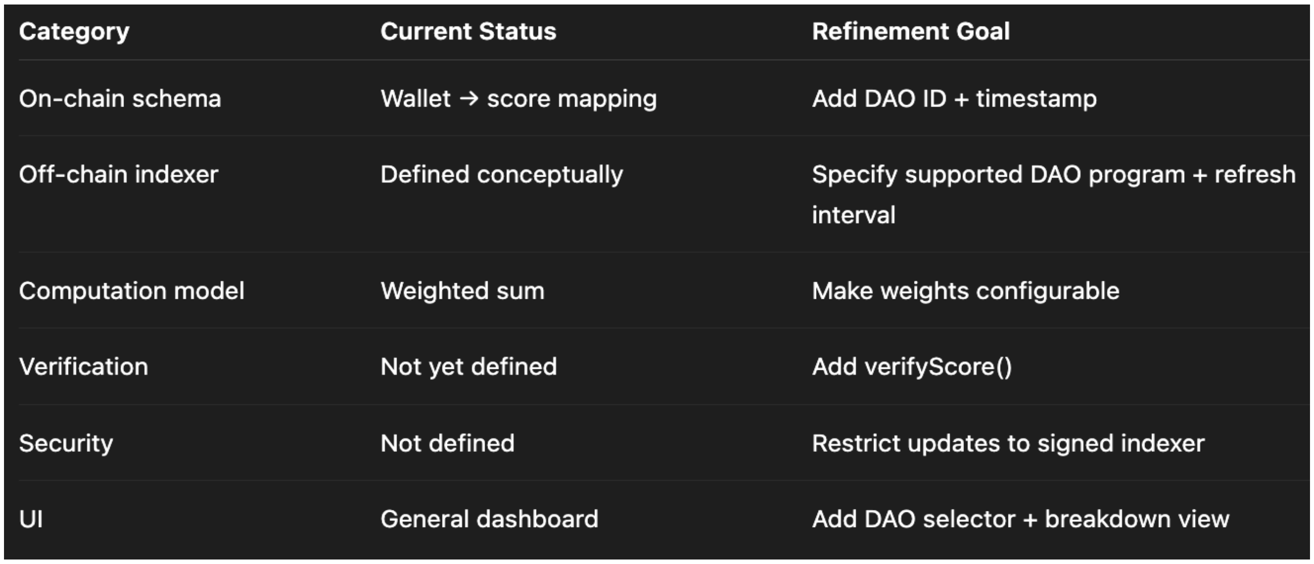
- Querying many contributors might be inefficient.
- Is there a permission model for data updates?
- Should admins trigger recalculation or dispute?

Refinements:



Aspect	Refinement
Access Control	Read public; restrict updates to signed indexer wallet.
Query Efficiency	Off-chain REST/GraphQL API for batch retrieval; on-chain limited to single address or top-N.
Admin Tools	Add recalculateScore(address) callable by DAO admin.
Integration Hook	API/SDK for DAOs to embed verified scores.

Granularity Check Summary



Category	Current Status	Refinement Goal
On-chain schema	Wallet → score mapping	Add DAO ID + timestamp
Off-chain indexer	Defined conceptually	Specify supported DAO program + refresh interval
Computation model	Weighted sum	Make weights configurable
Verification	Not yet defined	Add verifyScore()
Security	Not defined	Restrict updates to signed indexer
UI	General dashboard	Add DAO selector + breakdown view

Revised POC Definition (After Adversarial Review)

Goal:
Deliver an MVP that:

- Indexes governance data for one Solana DAO protocol (Realms).
- Calculates contributor reputation scores via a transparent weighted model.
- Stores scores on-chain for verifiable reputation.
- Allows contributors and DAOs to view and verify scores through a simple dashboard or API.
