PART A : Main Document

Part A: User Stories & on-chain Requirements document

1. Core User Personas :

- **Solo RPG players :** They are the primary target and will experience the full battle flow, XP gain, and NFT rewards. Essential to test core game logic.
- **NFT Collectors(Gameplay focused):** The collectible progression (cNFTs) is a key proof of achievement and player motivation. Core to the value prop of on-chain verifiability.
- **Ecosystem Builders / Community :** While not end-users, they are essential for the launch success, they can help amplify and support the project early on(List includes communities like Turbin3 fam, Superteam, Magicblock, etc).
- **Testers :** Crucial for validating game balance and ensuring that early program interactions behave as expected

2. Final Function Map (by User Type) :

- **Solo RPG players :**
  - i. Start a new game
  - ii. Explore game maps and navigate towns.
  - iii. Battle trainers using turn-based combat.
  - iv. Receive NFT rewards upon victories.
- **NFT Collectors :**
  - i. Connect wallet and access game dashboard.
  - ii. View earned NFT badges in an interactive gallery.
  - iii. Share NFTs on social media.

- **Ecosystem Builders :**
    1. View cities themed after their protocol.
    2. Read lore or branding related to their protocol.
    3. Receive demo access or early showcase features.
- **Testers :**
    1. Access early test builds via devnet.
    2. Use debug options (like level skipping).
    3. Submit feedback using forms or in-game prompts.

3. Potential On-chain requirements(Mapped by user story) :

- **Solo RPG players :**
    1. Initialize on-chain player profile (PDA)
    2. Store wallet address and set default game state
- **Explore game maps:**
    1. Define accessible map areas on-chain
    2. Log player’s location(x and y coordinate on grid) and progression state
- **Battle trainers using turn-based combat:**
    1. Validate location and trainer availability
    2. Store trainer metadata (difficulty, type)
    3. Optionally log battle start and outcome

- **Receive NFT rewards after victories**
    1. **Mint cNFT on-chain with metadata (trainer ID, timestamp)**
    2. **Ensure uniqueness (1 NFT per trainer per player)**
    3. **Link NFT to player’s wallet**

- **Ecosystem Builders**
    1. View cities themed around their protocol
    2. Store protocol-to-city mapping on-chain
- **NFT Collectors :**
    1. Retrieve NFTs associated with player wallet
    2. View and share earned NFT badges
- **Testers :**
    1. Access early test builds
    2. Help with feature requests

PART B : Process Appendix

PART A: Initial user & Function mapping

1. **Brainstormed user types :**

a. Direct Users(Day-to-day players and participants)

- Solo RPG players : People who want to play the single player story mode and defeat the gym trainers.
- NFT Collectors(Gameplay focused): Players who collect NFTs as proof of achievement or for game progression.
- Competitive PvP players : users who want multiplayer features or tournaments

b. Indirect Users / Beneficiaries

- Retro gaming communities: People who engage with the nostalgia
- cNFT traders : Secondary market participants who buy/sell the collectible NFTs without paying much

c. Administrators / Moderators

- Community Moderators : Currently not planning for discord server/any kind of online group, but since it is an onchain game, there might be a need in future(for weekly quests and rewards).
- Testers : early testers who will give feedback or feature requests

d. Stakeholders (Non-players)

- Token Holders : If the game gets decent users, then there are chances of token launches(also token rewards can bring more users as well)
- Ecosystem Backers / Builders: People from solana community

- Partnership Projects / NFT Collab partners : Since each city/town in the game can be themed as a solana protocol(like SuperCity for Superteam), so we can try to collaborate with them
- Tech providers: MagicBlock can be a part of it cause there are several places where magicblock's ER can be used, so if the project succeeds, Magicblock can showcase it as well

2. Appendix Entry: "Step 2 – User Prioritization Process Log"

a. **Initial Brainstormed User Types:**
   i. **Direct Users :**
      1. Solo RPG players
      2. NFT Collectors
      3. Competitive PvP players

   ii. Indirect Users :
      1. Retro gaming communities
      2. cNFT Traders

   iii. Administrators / Moderators
      1. Community moderators
      2. Testers

   iv. Stakeholders :
      1. Token holders
      2. Ecosystem backers / Builders
      3. Partnership projects / NFT collab partners
      4. Tech providers (eg Magicblock)

b. AI Recommendation Summary :
   i. The AI prioritized: Solo RPG players, NFT collectors, Testers, Ecosystem Builders.
   ii. Optional mention: cNFT Traders for post-launch NFT market functionality.

c. Final Selection :

i. Solo RPG Players
ii. NFT Collectors
iii. Testers
iv. Ecosystem Builders / Community

Rationale:
These four user types allow us to prove the game's core value (fun, skill-based progression, and on-chain achievements), while enabling us to test technical reliability and gain support from the Solana ecosystem. We excluded PvP and token holders from the POC scope to avoid premature complexity.

3. Core Function Mapping by user type :

- **Prompt used :**
  i. I have a Solana-based on-chain RPG game. I want to identify the most critical user interactions for a proof-of-concept (POC). I've already mapped key user types and their high-value functions:
     1. Solo RPG Players: Want to explore the game, fight trainers, and earn NFTs.
     2. NFT Collectors: Interested in owning rare in-game NFTs tied to achievements.
     3. Testers: Want to test game mechanics and submit feedback.
     4. Ecosystem Builders: Want to assess whether the game leverages Solana meaningfully.

  ii. Help me derive the most critical user stories and the technical requirements that will support them. Focus on what I absolutely need to build first to test the concept.
- **Output :**
- **Solo RPG players :**
  i. Start game / save progress on-chain.
  ii. Navigate different maps

    iii. Enter and complete gym challenges(battles)
    iv. Receive NFT badges/items after winning battle
- **NFT Collectors :**
    i. View collection of owned NFTs
    ii. Showcase completed gyms NFTs

- **Ecosystem Builders :**
    i. Brand integrations
    ii. View city themed around their protocols.
- **Testers :**
    i. Play early builds
    ii. Access test-only features(level skipping, etc)

4. Derived POC requirements:

- A basic RPG engine: player movement, story progression, and trainer interactions.
- Turn-based battle system (can be minimal).
- Smart contract to mint NFTs upon a battle win.
- Web3 wallet connection to authenticate the user and receive NFTs.
- Feedback capture mechanism (can be as simple as a Google Form or JSON on-chain).
- Minimal UI for the above: basic UI for gameplay + NFT inventory or mint confirmation.

My Reasoning for Picking the Final 2 Critical Interactions:

I selected these two user stories for the following reasons:

1. **“Player defeats trainer and gets NFT”** is **the core loop** of our RPG that demonstrates:
   - Actual gameplay (exploration + battle).
   - A meaningful on-chain outcome (NFT).
   - Integration of Solana primitives (minting NFT, wallet connection).
   - What makes this an *on-chain game* vs. a regular web game.
2. **“Tester gives feedback after gameplay”** is essential because:
   - This ensures rapid iteration and useful feedback for early builds.
   - Helps validate whether the game loop is enjoyable or needs rework.
   - Increases stakeholder (community/dev) involvement early on.

These two interactions together prove both the **on-chain depth** and **user engagement potential** of the game.

❌ Discarded Ideas and Why:

1. **NFT Marketplace Integration**
   - Too early for a POC.
   - Doesn't validate game mechanics or core loop.
   - Can be added post-feedback stage.

2. **Multiplayer Battle System**

- High complexity, not essential for early validation.
- Solo play is enough to test fun and reward structure.

3. **On-chain Quest Authoring**
   - Adds cool extensibility, but not critical now.
   - Better as a Phase 2 feature once the game world has traction.

PART B: Adversarial Analysis & Granularity Check

1. Critique & Refine User Stories / Requirements:

- **Prompt used :**
  i. Review my core user functions/stories and requirements below. Considering my project's refined value proposition — a solo RPG game with on-chain progression, turn-based battles, and cNFT achievement rewards — do these stories truly hit the mark? Are the requirements granular enough to map to specific technical components (e.g., database schemas, API endpoints, or blockchain programs)? What's missing or unclear?" Core User Stories: Solo RPG player starts a new game and defeats a trainer to earn an achievement NFT. Tester plays early builds and submits feedback on quest flow. Technical Requirements: Basic playable RPG engine with turn-based battles Gym trainer encounter logic and win detection Smart contract to mint achievement NFTs on win Simple feedback collection system (e.g., form or on-chain feedback stub)
- **AI Feedback received :**
  i. **On User Stories:**
     1. ✅ The stories are strong for MVP focus. The player loop and tester loop are clear.

2. ⚠️ However, they don’t capture edge cases, such as losing battles, exiting mid-quest, or replaying.
3. ⚠️ There’s no story capturing the NFT collector’s experience, despite their importance.
4. ⚠️ Community users (like ecosystem builders) are not represented, yet are key in initial traction and integration feedback.

ii. On Technical Requirements:

1. ✅ Good start with the core mechanics.
2. ⚠️ Granularity is not sufficient to begin detailed system design.
    a. “Basic RPG engine” needs to be broken into components like battle state manager, player inventory system, story progression tracker, etc.
    b. No mention of data models (e.g., player state schema), on-chain/off-chain split, or expected interactions between frontend and smart contracts.
    c. NFT minting should clarify metadata format, signature verification (if any), and wallet connection assumptions.
3. Feedback system might be overlooked—what format is expected, and where is it stored?

iii. My Analysis :

1. AI feedback is valid and constructive. The critique helped me realize a few blind spots:

2. User Story Refinements

a. I missed out on modeling failure cases, which are key for engagement (e.g., retrying, XP on loss, partial progress).
b. I underestimated the NFT collector’s user story, especially their journey from mint to showcase.
c. While community/ecosystem users aren't core players, a story focused on game review/demo feedback is important for adoption.

3. Granularity Gaps in Requirements:

a. **I had umbrella phrases like “basic engine” or “mint NFT” — these need breakdowns to be mappable to components (off-chain battle logic, Solana program for minting, etc.)**
b. **I didn’t describe data flows or persistent state management — crucial for development.**
c. **NFT metadata and cNFT design must be scoped early to avoid rework.**
d. **Even the “feedback system” should have rough decisions made (on-chain vs Google Form vs Discord integration, etc.)**

PART C: Granularity & Clarity Refinement Log

1. Solo RPG Player Story
    a. Before: Player starts a new game and defeats a trainer to earn an achievement NFT.
    b. After: Player starts a new game from the homepage. Player explores the game map to find a trainer. Player battles the trainer in turn-based combat. Upon winning, the player receives an NFT reward. Rationale: Split into atomic steps for clarity and easier implementation.
2. Basic RPG Engine Requirement
    a. Before: Basic playable RPG engine with turn-based battles.

   b. After: Implement player movement across different maps. Create turn-based battle system. Track and update story progression per player. Enable game state saving via connected wallet. Rationale: Replaced broad term with specific systems for development planning.

3. Tester Feedback Flow
   a. Before: Tester plays early builds and submits feedback on quest flow.
   b. After: Tester accesses the test build via testnet or dev link. Tester plays with debugging tools enabled. Tester submits structured feedback via a form. Rationale: Clarified access, experience, and submission process.
4. NFT Collector Interaction
   a. Before: View collection of owned NFTs.
   b. After: User connects their wallet to the game. User sees a visual gallery of earned NFTs. Rationale: Clarified the steps required to access and view NFTs.
5. Ecosystem Builder Integration
   a. Before: View city themed around their protocols.
   b. After: User discovers in-game cities themed after Solana projects. User reads lore or info linking the location to the protocol. Rationale:

Made the integration more interactive and meaningful.

6. Feedback System
    a. Before: Feedback capture mechanism (Google Form or JSON on-chain).
    b. After: Include a feedback form accessible in-game. Store responses in Google Sheets or simple on-chain structure. Rationale: Separated form interface from storage method for clarity.
7. Battle Failure Case
    a. Before: (Not included)
    b. After: Player loses the battle and returns to a safe point. Player retries the trainer battle. Rationale: Added critical edge case missing from original plan.

8. NFT Collector Sharing Feature
    a. Before: (Not included)
    b. After: Player can share their earned NFTs to social media from inventory. Rationale: Supports community growth and user pride.

PART D : Granularity & Clarity Refinement Log

1. **Player starts a new game from the homepage.**
    a. Create an on-chain player profile (PDA or user state).
    b. Associate the player profile with their wallet address.
    c. Initialize starting values: location(grid based x and y coordinate), XP, badge count, etc.
2. **Player explores the game map to find a trainer.**
    a. Store available map areas and trainer data on-chain(Not sure).
    b. Validate player's current location before allowing interaction.
    c. Optionally log movement history or visited nodes using magicblock's ER.
3. **Player battles the trainer in turn-based combat.**
    a. Store trainer stats and battle logic seeds on-chain (or hashed if using off-chain engine).
    b. Verify a trainer is available at that location.
    c. Optionally log battle initiations (trainer ID, player ID).
4. **Upon winning, the player receives an NFT reward.**
    a. On-chain validation that player has defeated the trainer.
    b. Mint a cNFT using Metaplex or custom program.
    c. Include metadata: trainer name, badge level, player wallet.
    d. Ensure one-time mint per trainer (prevent abuse).

5. **User sees a visual gallery of earned NFTs :**
    a. NFTs must be minted on-chain and linked to the wallet.
    b. Query the wallet’s token accounts and filter by game-related NFT mint addresses.
6. **User discovers in-game cities themed after Solana projects.**
    a. Optional: Store metadata about city-protocol mapping on-chain (to decentralize branding logic).
    b. Track user visits or city unlocks on-chain for progression or achievements.
7. **Player loses the battle and returns to a safe point.**
    a. Update player’s location and state on-chain after loss.
    b. Optionally track number of losses for balancing or XP rewards.
8. **Store off-chain data tied to wallet with checkpointing.**
    a. Store key checkpoints (e.g., last battle won, last area reached) on-chain.
    b. Ensure state is syncable between on-chain and off-chain components.
    c. Optional: Add backup/recovery logic using player wallet as reference.