

User Stories & On-Chain Requirements

Core User Personas

Comic creators and readers, having one feature for collectors (that comic and rewards will have i. Common) for the initial phase, but the focus will be to target creators and readers adding the most value for them 1st, so they'll be the main users at the beginning.

Function Maps

Comic Creator

1. Account: connect wallet
2. Profile: profile picture, bio, links to social media, and a gallery for their created comics
3. Dashboard: number of readers, most popular story branches, and earnings
4. New comic project: comic's title, genre, cover art, and a short synopsis.
5. **Branching Narrative Builder:** Drag-and-drop interface to create decision nodes (choices) and link chapters/outcomes. Upload panels, Add Narrative Text, Create a "Branch/Choice", Link Choice to New Panel. This will live in the blockchain
6. **NFT configuration:** Set Rarity/Supply, set price, auto-generate NFTs per branch/chapter.
7. Cross-Story Linking: Search/add other creators' comics as "canonical branches" (e.g., "If reader chooses X, continue in Creator Y's Comic Z")
8. Royalty & Collaboration Setup: Set % splits for co-created branches (e.g., 70% to primary creator, 30% to linked comic's creator).
9. Publish comic

Reader

1. Account: connect wallet
2. Dashboard:
3. Browse Comics: A gallery or library to discover new comics, filterable by genre or what's trending.
4. View Comic Intro Page: See the cover art, synopsis, and creator details before starting
5. View My Journey/Collection: A personal page where a reader can see the unique paths they've taken and view the collectible endings they own.
6. **Make Choices:** When a choice point is reached, the options are presented as clear, clickable buttons directly integrated into the story. When a branch links to another comic, show preview + "Jump to This Story" button. This will live in the blockchain
7. **One-Click Mint/Purchase:** Upon success, the story continues, and the NFT representing that narrative path is sent to their wallet. Something like "Confirm Purchase to Own This Ending."
8. Path Tracker & Collectible Ledger: Visual map of taken paths + owned NFT branches
9. Free-to-Read / NFT-Gated Endings: Read 80% free, but rare endings require NFT purchase. (*This can be set by creator, the % free, mark that comic or branch?*)

Collector

1. Cross-Story Unlock Badges: Auto-generated NFT badges when collecting branches from ≥ 3 linked comics.

List of Potential On-Chain Requirements

User Story: "User creates a new branching narrative."

Potential On-Chain Requirements:

- This is an account to store the amount of chapters, price (for each chapter?)
- Initialize branch
- Store if is the genesis
- Store which is the previous and next branches
- Store which comic belongs
- Stores decision nodes/relationships

User Story: "User creates chapter."

Potential On-Chain Requirements:

- Initialize chapter
- Store which branch belongs
- Store if is the continuation from another comic branch, which comic and branch

User Story: "User links chapter to another comic."

Potential On-Chain Requirements:

- Store to what comic is linked
- To what chapter will start next
- Which branch the chapter is in and the next one
- Link function for chapters, account

User Story: "User links branch to another comic." (not sure if this can be merged with link chapter"

Potential On-Chain Requirements:

- Link function for branches, account.
- Store from and to branches
- Store to what comic is linked

User Story: "User make choice in branch."

Potential On-Chain Requirements:

- Choice is stored in state
- Choice function, account
- Store which branch is the previous and next

User Story: "User mint branch to read."

Potential On-Chain Requirements:

- Mint function
- If is linked, then split royalties

Process Appendix

Part A: Initial User & Function Mapping

1) Manual User Brainstorming

Manual Task: Based on your refined Value Proposition from Assignment 1, brainstorm a broad list of every potential user type who might interact with your project. Think beyond just the primary user. Consider different categories:

- **Direct Users:** Who will use the product day-to-day? (e.g., content creators, voters, investors)
- **Indirect Users/Beneficiaries:** Who benefits from the direct users' actions? (e.g., content consumers, project owners)
- **Administrators/Moderators:** Who will manage the system? (e.g., you as the developer, community moderators)
- **Stakeholders:** Who has a vested interest in the project's success but may not use it directly? (e.g., token holders, partners)

Output: A preliminary, comprehensive list of all potential user types.

1. Readers of the comics
2. Comic creators
3. Developers
4. Proof of content to follow platform rules
5. Comic Collectors
6. Investors
7. Reward collectors

2) AI-Assisted User Prioritization

- **AI Prompt:** "My project's value proposition is [Revolutionize branching narratives and NFT comics into a new model of "living comic" merging them together into a single, interconnected "comicverse" with deep on-chain mechanics. The focus is to ease the burden on creators so they can build deeper engagement while having a novel way to monetize their work in a more simple way; and readers/collectors unveil unique collectible journeys in a personalized story crafted by their choices, enabling cross-story collabs; where both have really easy to use UX and utilities. This leads to 3 pillars: verifiable digital scarcity and provenance for particular narratives, collab world-building

between creators and fans or other creators enhancing monetization, and “comicverse” having dynamic interconnected branching narrative experiences with blockchain ownership.]. Here is a brainstormed list of all potential user types: [1. Readers of the comics. 2. Comic creators. 3. Developers. 4. Proof of content to follow platform rules. 5. Comic Collectors. 6. Investors. 7. Reward collectors]. 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."

I agree with comic creator and collector, and reader,; but will also include the moderators like proof of content because they'll need tools that help to do the work in a smooth way. I don't think it is good to merge reward and comic collectors because the comic ones may want other targeted features than the other ones that I believe will just hold the comics for some kind of reward that they'll just claim and go.

Readers I prefer to maintain like that, and can have in mind the ones who love the branching narrative to be appealing for them, but I want to keep the general for starting.

For moderators I'm not sure to have them at start because will have to think about the rules

So, I think I'll go with

Comic creators and readers, having one feature for collectors (that comic and rewards will have i. Common) for the initial phase, but the focus will be to target creators and readers adding the most value for them 1st, so they'll be the main users at the beginning.

Gemini

Based on your powerful value proposition for a "living comicverse," the goal of a Proof-of-Concept (PoC) is to validate the most fundamental assumptions of your model with the smallest possible group of participants. You need to prove that the core loop of creation, consumption, and collection is viable and compelling.

From your list, here are the 3 most critical user types to focus on for an initial Proof-of-Concept:

1. **The Comic Creator**
2. **The Reader**
3. **The Comic Collector**

Here is the rationale for each, explaining why they are essential to proving your project's core value at this early stage.

1. The Comic Creator

Rationale: The Comic Creator is the engine of your entire ecosystem. Without compelling content, there is nothing for anyone else to do. For your PoC, you don't need hundreds of creators; you need one to three dedicated creators who can truly test your platform's core promises to them.

They are essential for proving:

- **Ease of Use:** Can a creator *actually* build a branching narrative using your tools without an immense technical burden? Proving that your UX is simpler than the alternatives is critical to the "ease the burden" part of your value proposition.
- **Monetization Viability:** Does your model for "a novel way to monetize" actually work? You need a creator to successfully mint narrative branches as NFTs and see a financial return. This directly tests the economic incentive that is supposed to attract future creators.
- **Engagement Tools:** Do the tools you provide for "collab world-building" actually lead to "deeper engagement"? The creator's direct feedback on whether fans are interacting in meaningful new ways is the only way to measure this.

In short, the Comic Creator validates the entire supply side of your "comicverse."

2. The Reader

Rationale: The Reader represents the purest form of product-market fit for the content itself. This user is motivated by story and experience, not by financial speculation. They are the bedrock of long-term, sustainable engagement.

They are essential for proving:

- **Narrative Appeal:** Is a "living comic" with branching narratives actually more engaging than a traditional linear comic? Readers' behavior—how many paths they explore, how much time they spend, and whether they complete the story—is the ultimate metric for whether the core experience is fun.
- **Personalized Journey Value:** Do users *feel* that the story is "personalized" and that their choices matter? Feedback from this group is crucial to validate that the branching structure creates a compelling "unique collectible journey" from a purely narrative perspective.
- **Usability (Consumption):** Can a non-technical user easily navigate the branching story? The Reader stress-tests the "really easy to use UX" for the consumption side of the platform. If they get confused or frustrated, the model fails.

The Reader validates the intrinsic entertainment value of your product, separate from any blockchain or financial element.

3. The Comic Collector

Rationale: The Comic Collector is the user who validates the entire economic model and the "deep on-chain mechanics" of your vision. While a Reader proves the story is good, the Collector proves it's valuable. There is overlap with the Reader, but their primary motivation is different.

They are essential for proving:

- **Scarcity and Ownership Value:** Will people actually pay for "verifiable digital scarcity and provenance for particular narratives"? The Collector is the one who puts their money on the line, proving that unique story branches have real-world value as assets. This directly tests your first pillar.
- **The "Comicverse" Concept:** Collectors, more than anyone, are incentivized by the idea of an interconnected "comicverse." The value of their asset (a unique story path from Comic A) could be enhanced by a "cross-story collab" with Comic B. They are the user type most likely to understand and financially support this long-term vision.

- **Blockchain Utility:** Does owning the NFT provide meaningful utility? The Collector is the user who will care most about token-gated access, rights, or the ability to participate in "collab world-building." They test whether your on-chain mechanics are just gimmicks or real value-adds.

The Comic Collector validates that your "living comic" is not just a novelty but a new, viable asset class that people are willing to invest in.

Deep Seek

Based on your value proposition and the need to prove core functionality with minimal resources, ****focus your initial Proof-of-Concept (PoC) exclusively on these 3 user types****:

1. ****Comic Creators****

Why critical?

- ****Core Pillar Validation****: Your #1 value prop is ***reducing creator burden*** while enabling monetization. If creators can't easily build branching narratives or mint chapters, the entire "comicverse" collapses.
- ****Simplicity Test****: PoC must prove your tools are simpler than coding in Twine or fighting Webtoon's algorithms.
- ****Monetization Hook****: They'll validate if NFT-based branch monetization beats traditional platforms (e.g., 90% royalties vs. Webtoon's 50% cut).

****PoC Goal****: ***Can a non-technical creator publish a branching comic chapter in <15 mins, with clear monetization paths?***

2. ****Experiential NFT Collectors**** (subset of "Comic Collectors" + "Reward Collectors")

Why critical?

- ****Scarcity/Provenance Proof****: They're the only group who will pay for ***verifiable narrative rarity*** (e.g., "I own the 1/10 NFT where Character X dies").
- ****Cross-Story Collab Demand****: Their feedback proves if interconnected "comicverse" assets feel valuable (e.g., "Does owning ***Creator A's*** rare ending unlock ***Creator B's*** story?").
- ****UX Stress Test****: They'll tolerate early blockchain friction ***if*** utility is unique—but will ruthlessly flag clunky flows.

****PoC Goal****: ***Will collectors pay 2x floor price for a branch-dependent outcome, and intuitively navigate cross-comic connections?***

3. ****Lore Masters**** (under "Readers" but more strategic)

Why critical?

- **Engagement Benchmark**: They're your "canary in the coal mine" for narrative depth. If branching choices feel trivial ("Choose sword or gun?"), they'll reject it.
 - **Organic Evangelists**: They map story universes for fun. Their buy-in = free stress-testing of "comicverse" interconnectivity.
 - **Bridge to Mainstream**: They recruit traditional comic fans if the experience feels revolutionary.
- PoC Goal**: *Do Lore Masters replay chapters to uncover 90% of branches, and create fan wikis without prompting?*

🚫 Why Exclude Others for PoC?

- **General Readers**: Too broad; feedback won't differentiate *why* your model beats Webtoon.
- **Developers**: Your VP emphasizes "no-code creator ease." Involving them now distracts from UX testing.
- **Investors**: Premature—they need traction with creators/collectors first.
- **Proof of Content**: A governance problem for Scale, not PoC.

⚙️ PoC Execution Blueprint

User Type	What to Test	Success Metric
Comic Creators	Onboarding flow, branch creation tool, 1-click NFT minting, royalty setup	≤15 min publish time; 0 support tickets
NFT Collectors	Branch purchase friction, cross-story utility clarity, resale mechanics	≥50% pay premium for rare branches; ≤3 clicks to access linked content
Lore Masters	Choice impact depth, replay value, "comicverse" coherence	≥4 replays/user; unsolicited fan theories/maps

****Tools to Fake Complexity****:

- Use visual branching tools like ****Twine**** for narratives + ****Metaplex**** for Solana NFT minting.
- Simulate cross-comic links with dummy QR codes (real blockchain integration comes later).

💡 Key Insight

Your PoC lives or dies by ****creator simplicity**** and ****collector perceived value****. If creators groan at your tool or collectors shrug at "owning" choices, pivot immediately. **The "comicverse" is a scaling problem—not a PoC requirement.** (*What does this mean?*)

3) Core Function Mapping

- **AI Task**: "For a project with this value proposition [Revolutionize branching narratives and NFT comics into a new model of "living comic" merging them together into a single, interconnected "comicverse" with deep on-chain mechanics. The focus is to ease the burden on creators so they can build deeper engagement while having a novel way to monetize their work in a more simple way; and readers/collectors unveil unique collectible journeys in a personalized story crafted by their choices, enabling cross-story collabs; where both have really easy to use UX and utilities. This leads to 3 pillars: verifiable digital scarcity and provenance for particular narratives, collab world-building between creators and fans or other creators enhancing monetization, and "comicverse" having dynamic interconnected branching narrative experiences with blockchain ownership] and focusing on these prioritized user types [Comic creators and readers, having one feature for collectors (that comic and rewards will have i. Common) for the initial phase, but the focus will be to target creators and readers adding the most value for them 1st, so they'll be the main users at the beginning.], help map out the key functions or interactions each user would need to perform."

Comic Creator

10. Account: connect wallet
11. Profile: profile picture, bio, links to social media, and a gallery for their created comics
12. Dashboard: number of readers, most popular story branches, and earnings
13. New comic project: comic's title, genre, cover art, and a short synopsis.
14. **Branching Narrative Builder**: Drag-and-drop interface to create decision nodes (choices) and link chapters/outcomes. Upload panels, Add Narrative Text, Create a "Branch/Choice", Link Choice to New Panel. This will live in the blockchain
15. **NFT configuration**: Set Rarity/Supply, set price, auto-generate NFTs per branch/chapter.
16. Cross-Story Linking: Search/add other creators' comics as "canonical branches" (e.g., "If reader chooses X, continue in Creator Y's Comic Z")
17. Royalty & Collaboration Setup: Set % splits for co-created branches (e.g., 70% to primary creator, 30% to linked comic's creator).
18. Publish comic

Reader

10. Account: connect wallet
11. Dashboard:
12. Browse Comics: A gallery or library to discover new comics, filterable by genre or what's trending.
13. View Comic Intro Page: See the cover art, synopsis, and creator details before starting
14. View My Journey/Collection: A personal page where a reader can see the unique paths they've taken and view the collectible endings they own.
15. **Make Choices**: When a choice point is reached, the options are presented as clear, clickable buttons directly integrated into the story. When a branch links to another comic, show preview + "Jump to This Story" button. This will live in the blockchain
16. **One-Click Mint/Purchase**: Upon success, the story continues, and the NFT representing that narrative path is sent to their wallet. Something like "Confirm Purchase to Own This Ending."
17. Path Tracker & Collectible Ledger: Visual map of taken paths + owned NFT branches
18. Free-to-Read / NFT-Gated Endings: Read 80% free, but rare endings require NFT purchase. (*This can be set by creator, the % free, mark that comic or branch?*)

Collector

2. Cross-Story Unlock Badges: Auto-generated NFT badges when collecting branches from ≥ 3 linked comics.

Gemini

Of course. Based on your value proposition and your strategic focus on **Comic Creators** and **Readers** for the initial phase, here is a map of the key functions and interactions each user would need to perform. This map is designed for an initial product, prioritizing core value over feature bloat.

Core Platform Functions (for the Initial Phase)

The focus is on simplicity and validating the core loop: **Creators build, and Readers experience.**

1. Key Functions for the Comic Creator

The overarching goal is to **"ease the burden"** and provide a **"simple way to monetize."** Every function should be designed with this principle in mind.

A. Onboarding & Profile Setup

- **Create Account**: Simple sign-up using a social login (Google, Twitter) or email, which then links to a crypto wallet (e.g., Phantom, Solflare) in a user-friendly way.
- **Build Creator Profile**: A simple public page with a profile picture, bio, links to social media, and a gallery for their created comics. This establishes their brand identity on the platform.

B. Comic Creation & Narrative Design (The "Living Comic" Editor)

- **Create New Comic Project:** A simple form to input the comic's title, genre, cover art, and a short synopsis.
- **Visual Narrative Editor:** This is your **most critical feature**. It should be a simple, node-based visual tool.
 - **Upload Panel/Page:** Allow creators to upload their art (JPEG, PNG, GIF) for a single scene or panel.
 - **Add Narrative Text:** Add dialogue, captions, and descriptions to a panel.
 - **Create a "Choice":** A button to create a choice point branching from a panel. For example, after Panel 5, they create two buttons: "Go into the cave" and "Climb the mountain."
 - **Link Choice to New Panel:** Easily drag a line from a "Choice" button to another uploaded panel, visually mapping out the story's branches.
- **Define NFT Properties (Simple Monetization):**
 - **Designate "Collectible" Panels/Paths:** The creator can toggle a switch on a specific panel or an entire story branch to designate it as a "collectible" (NFT).
 - **Set Rarity/Supply (Simple):** For a collectible path, they can set a simple supply number (e.g., "Only 100 readers can take this path and mint its ending"). This directly enables **verifiable digital scarcity**.
 - **Set Price:** Assign a price (e.g., in SOL or USDC) for minting a collectible ending or accessing a premium branch.

C. Publishing & Management

- **Publish Comic:** A one-click button to make the comic live and accessible to readers on the platform.
- **Creator Dashboard:** A simple dashboard showing key stats: **number of readers**, most **popular story branches**, and **earnings** from collected (minted) paths. This provides feedback and validates their efforts.

2. Key Functions for the Reader

The focus here is a **"really easy to use UX"** to "unveil unique collectible journeys." The technology should be nearly invisible.

A. Discovery & Onboarding

- **Browse Comics:** A gallery or library to discover new comics, filterable by genre or what's trending.
- **View Comic Intro Page:** See the cover art, synopsis, and creator details before starting.
- **Simple Login (Optional but Recommended):** Allow readers to create a simple profile to save their progress across multiple comics and manage their collection.

B. The Reading Experience

- **Start Reading:** A clear "Start" button to enter the comic.
- **Immersive Reader Interface:** A clean, full-screen view focused on the art and text of the current panel. Navigation should be intuitive (click/swipe to advance).
- **Make Choices:** When a choice point is reached, the options are presented as clear, clickable buttons directly integrated into the story.

- **Seamless Branching:** After a choice is made, the next panel loads instantly, making the experience feel fluid and responsive.

C. Interaction with On-Chain Elements

- **Encounter a "Collectible Path":** The interface should clearly but subtly indicate when a choice leads down a **premium** or collectible **path**. It might say, *"This choice leads to a rare ending. Mint it to own this part of the story."*
- **One-Click Mint/Purchase (The Collector Feature):**
 - This is the core feature for the **Collector** persona in the initial phase.
 - When a reader decides to purchase a collectible path, a simple pop-up connects to their wallet to approve the transaction. The language should be simple: "Confirm Purchase to Own This Ending."
 - Upon success, the story continues, and the NFT representing that narrative path is sent to their wallet.
- **View My Journey/Collection:** A personal page where a reader can see the unique paths they've taken and view the collectible endings they own. This reinforces the value of their "personalized story."

By focusing on this curated set of functions, you directly test your core hypotheses: Creators can easily build and monetize branching stories, and Readers find the experience engaging enough to read, explore, and even pay to own a unique piece of the narrative. This lays a solid foundation for the more complex "collab world-building" and "interconnected comicverse" features in the future.

Deep Seek

Based on your focus on **creators and readers** (with collectors as a secondary outcome), here's a functional map of core interactions aligned with your pillars:

🧠 **Comic Creator Functions**

(Validates: Monetization Simplicity + Collab World-Building)

1. **Branching Narrative Builder**

- **Function:** Drag-and-drop interface to create decision nodes (choices) and link chapters/outcomes.

- **Why:** Core to "easing burden" vs. coding in Twine/Unity. Must be intuitive.

2. **Cross-Story Linking**

- **Function:** Search/add other creators' comics as "canonical branches" (e.g., "If reader chooses X, continue in Creator Y's Comic Z").

- **Why:** Enables "comicverse" interconnectivity pillar.

3. **One-Click NFT Minting**

- *Function*: Auto-generate NFTs per branch/chapter with configurable scarcity (e.g., "Only 10 copies of the 'villain wins' ending").

- *Why*: Simplifies monetization ("novel way to monetize").

4. **Royalty & Collaboration Setup**

- *Function*: Set % splits for co-created branches (e.g., 70% to primary creator, 30% to linked comic's creator).

- *Why*: Incentivizes "collab world-building".

📖 **Reader Functions**

(Validates: Personalized Journeys + Comicverse UX)

1. **Choice-Driven Navigation**

- *Function*: Tap/click decisions that instantly load next branch (no wallet pop-ups mid-story).

- *Why*: "Personalized story crafted by choices" requires seamless immersion.

2. **Path Tracker & Collectible Ledger**

- *Function*: Visual map of taken paths + owned NFT branches (even if not a collector).

- *Why*: Shows "unique collectible journey" value without crypto complexity.

3. **Cross-Story Discovery**

- *Function*: When a branch links to another comic, show preview + "Jump to This Story" button.

- *Why*: Proves "interconnected comicverse" utility.

4. **Free-to-Read / NFT-Gated Endings**

- *Function*: Read 80% free, but rare endings require NFT purchase.

- *Why*: Balances accessibility with monetization.

⚠️ **Collector Utilities (Light-Touch)**

(Secondary Focus: Scarcity/Provenance)

1. **Ownership Dashboard**

- *Function*: View all owned branches, rarity stats (e.g., "1 of 10 copies"), and resell options.

- *Why*: Verifies "digital scarcity" without needing custom features.

2. **Cross-Story Unlock Badges**

- **Function**: Auto-generated NFT badges when collecting branches from ≥ 3 linked comics.
- **Why**: Incentivizes "comicverse" exploration.

🌱 **Prioritized PoC Workflow**

```mermaid

graph TD

A[Creator] -->|1. Build Branching Comic| B(Branching Tool)

A -->|2. Link to Other Comics| C(Cross-Story Search)

A -->|3. Mint NFTs per Branch| D(1-Click Mint)

E[Reader] -->|4. Make Choices| F(Choice Navigation)

E -->|5. Hit Cross-Story Link| G(Preview + Jump)

E -->|6. Unlock Rare Ending| H(NFT Purchase Flow)

I[Collector] -->|7. View Rarity| J(Ownership Dashboard)

...

### ### 🔑 **Why This Focus Wins**

1. **Creators First** → They supply content; simplify their pain (branching/minting) to attract supply.
2. **Readers Drive Demand** → If choices feel impactful and cross-links are exciting, collectors *emerge organically*.
3. **Avoid Collector Over-Engineering** → Your pillars (scarcity, collabs) are proven by creator/reader actions alone.

**Example**: A reader's choice in *Creator A's* comic unlocks a rare branch in *Creator B's* comic → Creator B earns royalties → Reader feels "comicverse" power → Collectors chase the rare branch NFT.

**Tools to Build This**:

- Branching: Modified **Twine** or **Ink** engine.
- UX: **Figma** prototype masking blockchain complexity.

- Minting: **\*\*Solana's Metaplex\*\*** with pre-built templates.

Focus here validates 80% of your value prop with 20% effort.

## 4) Deriving Core POC Requirements

- **AI Task:** Prompt the AI: "Based on these top 1-2 critical user interactions **[Make Choices]**: When a choice point is reached, the options are presented as clear, clickable buttons directly integrated into the story. When a branch links to another comic, show preview + "Jump to This Story" button. And **Branching Narrative Builder:** Drag-and-drop interface to create decision nodes (choices) and link chapters/outcomes. Upload panels, Add Narrative Text, Create a "Branch/Choice", Link Choice to New Panel. These two will live in the blockchain], what are the key technical requirements needed to build a proof-of-concept?"

### Tech requirements

1. Smart Contract Architecture
  - a. Branch storage contract (stores decision nodes/relationships)
  - b. NFT minting contract (with metadata standards for chapters/choices)
  - c. Cross-comic linking registry (for interconnected story verification)
  - d. On-Chain Data Structures.
2. Program Functions (Instructions):
  - a. Initialize ComicAccount.
  - b. Create a new PanelAccount and link it to the comic.
  - c. Create a BranchAccount and define the narrative links.
3. Decentralized Storage Solution
4. Creator Mode (Branching Narrative Builder)
5. Reader Mode (Choice Interface/Interaction System)
6. Interoperability System (Cross-Story Protocol)
7. Solana JSON-RPC Provider
8. Wallet Integration

### Gemini

Excellent. Focusing on these two core interactions is the perfect strategy for a Proof-of-Concept (PoC). The goal is to prove that the fundamental mechanics of on-chain narrative creation and consumption are feasible, engaging, and performant.

The phrase **"these two will live in the blockchain"** is the most critical technical constraint. It dictates an architecture where the comic's structure (the relationship between panels and choices) is defined and stored on-chain, not just the final ownership token.

Here are the key technical requirements to build this PoC, broken down by category.

---

#### 1. On-Chain Program (Smart Contracts)

This is the heart of your project. It will define the rules and data structures of the "comicverse." Given your background, the **Solana** blockchain is the logical choice due to its high speed and low transaction costs, which are essential for a fluid user experience.

- **Programming Language & Framework:**
  - **Rust** is the required language for Solana programs.
  - **Anchor Framework:** This is a non-negotiable requirement for a PoC. It dramatically simplifies development by abstracting away boilerplate code, handling security checks, and auto-generating a client-side library for your frontend to interact with the program.
- **Key On-Chain Data Structures:** You will need to define how to store the narrative graph on-chain.
  - **ComicAccount:** A central account for each comic. It will store metadata like the comic's title, the creator's public key, a link to the cover art, and a counter for the number of panels.
  - **PanelAccount:** An account for each individual panel or story node. It needs to store:
    - **comic\_id:** A reference back to the parent **ComicAccount**.
    - **panel\_id:** A unique identifier for the panel within the comic.
    - **content\_uri:** **A pointer (URI string)** to where the actual panel art and text are stored off-chain (e.g., on Arweave or IPFS).
  - **BranchAccount:** An account representing a choice point. It will store:
    - **source\_panel\_id:** The panel where the choice is presented.
    - **choices:** A list or array of structs, where each struct contains:
      - **choice\_text:** The text on the button (e.g., "Go into the cave").
      - **destination\_id:** The ID of the **PanelAccount** or **ComicAccount** this choice leads to. This allows for both intra-comic and cross-comic branching.
- **Key Program Functions (Instructions):**
  - **create\_comic(title, cover\_uri):** Initializes the **ComicAccount**.
  - **add\_panel(comic\_id, panel\_uri):** Creates a new **PanelAccount** and links it to the comic.
  - **create\_branch(source\_panel\_id, choices):** The most critical function. It creates a **BranchAccount** and defines the narrative links, effectively writing the story's structure to the blockchain.

## 2. Off-Chain Storage

It is technically and financially impossible to store high-resolution images and large amounts of text directly on Solana. The on-chain program only stores the *structure* and *proof*. The actual content must live elsewhere.

- **Decentralized Storage Solution:** You must use a service like **Arweave** or **IPFS**.
  - **Workflow:** When a creator uploads a panel in the builder, the frontend first uploads the asset to Arweave/IPFS. This service returns a permanent URI. That URI is what gets saved in the **PanelAccount** on-chain.
  - **Recommendation:** **Arweave** is often preferred for this use case because of its permanent, one-time-fee storage model, which aligns well with the concept of a permanent, on-chain comic.

### 3. Frontend Application (User Interface)

This is the visual layer that users interact with. It will be a single web application with two distinct modes.

- **Core Technology:** A modern JavaScript framework like **React** (recommended), Vue, or Svelte.
- **Creator Mode (Branching Narrative Builder):**
  - **Visual Diagramming Library:** You need a library to handle the drag-and-drop interface. **React Flow** is an excellent, purpose-built choice for building node-based editors.
  - **UI Components:** Forms for uploading panel images and adding narrative text.
  - **Blockchain Interaction:** When the creator hits "Publish," the frontend will:
    1. Upload all new panel assets to Arweave/IPFS.
    2. Construct a series of transactions to call your on-chain program (e.g., `add_panel`, `create_branch`).
    3. Prompt the user to approve these transactions with their wallet.
- **Reader Mode (Choice Interface):**
  - **State Management:** The UI will fetch the data for the current panel (e.g., the Arweave URI) from your on-chain program.
  - **Dynamic Rendering:** It will display the panel's content and then call the program again to see if a `BranchAccount` exists for that panel. If so, it will dynamically render the choice buttons.
  - **Navigation:** When a user clicks a choice, the frontend gets the destination ID and fetches the data for the next panel, creating the illusion of a seamless story.

### 4. Backend & Connectivity

While the logic is on-chain, you still need a bridge between your frontend and the blockchain network.

- **Solana JSON-RPC Provider:** Your application cannot talk to the blockchain directly. It needs a gateway.
  - **Requirement:** You must use an RPC provider like **Helius**, **Triton**, or **QuickNode**. Relying on the public RPC for a PoC is unreliable and will result in a poor user experience. Helius, in particular, offers enhanced APIs that can simplify fetching and parsing on-chain data.
- **Wallet Integration:**
  - **Requirement:** Use the **Solana Wallet-Adapter** libraries. This provides a standardized, plug-and-play UI for users to connect their Phantom, Solflare, or other wallets to your application.

By focusing on these specific technical components, you can effectively build a PoC that directly validates your two most critical user interactions while establishing a solid, scalable architectural foundation for the future.



## Deep Seek

Here are the key technical requirements for building a Proof-of-Concept (PoC) focused on your two critical user interactions:

### ### \*\*Core Technical Requirements\*\*

#### 1. \*\*Blockchain Infrastructure\*\*

- **Smart Contract Architecture**:
  - Branch storage contract (stores decision nodes/relationships)
  - NFT minting contract (with metadata standards for chapters/choices)
  - Cross-comic linking registry (for interconnected story verification)
- **Chain Selection**: Solana (low fees, high speed) using Metaplex NFT standard
- **On-Chain Data Structure**:

```
```rust
struct ComicChapter {
    chapter_id: u64,
    panels: Vec<IPFSCid>, // Image references
    choices: Vec<Choice> // Decision nodes
}

struct Choice {
    option_text: String,
    next_chapter: Pubkey, // Linked chapter address
    is_cross_comic: bool
}
```
```

#### 2. \*\*Choice Interaction System

- **Frontend SDK**:
  - React-based choice renderer with embedded wallet transactions
  - Choice selection triggers:
    1. Local state update (immediate UI response)
    2. On-chain choice recording (delayed confirmation)
- **Transaction Flow**:

```
```mermaid
sequenceDiagram
    Reader->>Frontend: Clicks choice
    Frontend->>Wallet: Request signature
    Wallet->>Blockchain: Submit choice_tx
    Blockchain->>Smart Contract: Record choice
    Smart Contract->>Frontend: Return next chapter ID
    Frontend->>IPFS: Fetch new panel
```
```

#### 3. \*\*Creator Branch Builder

- **Drag-and-Drop Engine**:
  - React Flow or BaklavaJS-based node editor

- Visual mapping of: `Panel → Choice → Linked Chapter`
- **\*\*On-Chain Publishing Flow\*\***:
  1. Upload assets to IPFS (panels/text)
  2. Construct chapter struct with choices
  3. Mint as NFT with metadata
  4. Register cross-comic links in registry contract

#### 4. **\*\*Interoperability System\*\***

- **\*\*Cross-Story Protocol\*\***:
  - Universal chapter addressing: `creator\_id/comic\_id/chapter\_id`
  - Link preview resolver:
 

```
```javascript
async function fetchPreview(chapterAddress) {
  const metadata = await blockchain.getMetadata(chapterAddress);
  return {
    title: metadata.name,
    cover: metadata.image,
    creator: metadata.creator
  };
}
...`
```

5. ****Hybrid Storage Solution****

Data Type	Storage	Rationale
Choice outcomes	Blockchain	Core value (provenance)
Panel images	IPFS/Arweave	Cost efficiency
Narrative text	Decentralized DB (Tableland)	Queryable text
UI state	Local/Session Storage	Instant UX

****Minimal Viable PoC Stack****

```
```mermaid
graph LR
 A[Creator Interface] -->|NextJS| B[Branch Builder]
 B -->|Metaplex| C[Solana Program]
 C -->|Chain Data| D[Blockchain]
 E[Reader Interface] -->|Wallet Adapter| F[Choice Engine]
 F --> D
```
```

G[IPFS] -->|CID Storage| H[Panels/Text]
I[Cross-Comic Registry] --> D

...

****Key Components**:**

1. ****Frontend**:**

- React + TypeScript
- Solana Wallet Adapter
- React Flow for node editor
- Tailwind CSS for UI

2. ****Backend/Blockchain**:**

- Solana Devnet
- Anchor Framework (Rust smart contracts)
- Metaplex NFT Standard
- IPFS via NFT.Storage

3. ****Tooling**:**

- Figma prototypes for UX validation
- Playwright for choice-flow testing
- Solana Explorer for chain verification

****Critical Path Implementation****

1. ****Week 1: Core Smart Contracts****

- Chapter NFT minting with choice metadata
- Choice recording contract (stores reader decisions)

2. ****Week 2: Creator MVP****

- Drag-and-drop editor (mock blockchain)
- IPFS upload integration
- "Publish Chapter" flow

3. ****Week 3: Reader MVP****

- Choice rendering engine
- Cross-comic preview resolver
- Wallet integration (phantom)

4. ****Week 4: Integration****

- Connect editor to actual minting
- Implement choice persistence
- Basic analytics dashboard

****Key Technical Risks & Mitigation****

1. ****Choice Latency Issue**:**

- ***Risk*:** Blockchain confirmation delays break immersion

- ***Solution***: Optimistic UI updates + fallback to session storage
2. ****Cross-Comic Link Verification****:
 - ***Risk***: Broken links when external comics change
 - ***Solution***: Immutable chapter addressing via NFT mint ID
 3. ****Creator Onboarding Friction****:
 - ***Risk***: Gas fees deter creation
 - ***Solution***: Sponsored minting via platform gas tank
 4. ****Metadata Bloat****:
 - ***Risk***: Choice data makes NFTs expensive
 - ***Solution***: Off-chain choice mapping with on-chain hashes

****Validation Metric****:

"Can a non-technical creator publish a branching chapter and have a reader complete one cross-linked choice path in under 3 minutes?"

Part B: Adversarial Analysis & Granularity Check

1) Critique & Refine User Stories / Requirements:

- **AI Prompt**: "Review my core user functions/stories **[Make Choices]**: When a choice point is reached, the options are presented as clear, clickable buttons directly integrated into the story. When a branch links to another comic, show preview + "Jump to This Story" button. And **[Branching Narrative Builder]**: Drag-and-drop interface to create decision nodes (choices) and link chapters/outcomes. Upload panels, Add Narrative Text, Create a "Branch/Choice", Link Choice to New Panel. These two will live in the blockchain] and requirements [
 - Smart Contract Architecture
 - * Branch storage contract (stores decision nodes/relationships)
 - * NFT minting contract (with metadata standards for chapters/choices)
 - * Cross-comic linking registry (for interconnected story verification)
 - * On-Chain Data Structures.
 - Program Functions (Instructions):
 - * Initialize ComicAccount.
 - * Create a new PanelAccount and link it to the comic.
 - * Create a BranchAccount and define the narrative links.
 - Decentralized Storage Solution
 - Creator Mode (Branching Narrative Builder)
 - Reader Mode (Choice Interface/Interaction System)
 - Interoperability System (Cross-Story Protocol)
 - Solana JSON-RPC Provider
 - Wallet Integration]. Considering my project's refined value proposition [Revolutionize branching narratives and NFT comics into a new model of "living comic" merging them together into a single, interconnected "comicverse" with deep on-chain mechanics. The

focus is to ease the burden on creators so they can build deeper engagement while having a novel way to monetize their work in a more simple way; and readers/collectors unveil unique collectible journeys in a personalized story crafted by their choices, enabling cross-story collabs; where both have really easy to use UX and utilities. This leads to 3 pillars: verifiable digital scarcity and provenance for particular narratives, collab world-building between creators and fans or other creators enhancing monetization, and “comicverse” having dynamic interconnected branching narrative experiences with blockchain ownership.], do these stories truly hit the mark? Are the requirements granular enough to map to specific technical components (e.g., database schemas, API endpoints, specific blockchain programs)? What's missing or unclear?"

1. Smart Contract Architecture
 - a. Branch storage contract (stores decision nodes/relationships)
 - b. NFT minting contract (with metadata standards for chapters/choices)
 - c. Cross-comic linking registry (for interconnected story verification)
 - d. On-Chain Data Structures.
 2. Program Functions (Instructions):
 - a. Initialize ComicAccount.
 - b. Create a new PanelAccount and link it to the comic.
 - c. Create a BranchAccount and define the narrative links.
 3. Decentralized Storage Solution
 4. Creator Mode (Branching Narrative Builder)
 - a. Upload all new panel assets to Arweave/IPFS.
 - b. Add panel
 - c. Create branch
 - d. Create chapter
 - e. Drafting Workflow like DraftChapter editable without on-chain fees, use localStorage
 - f. Add “comic discovery API” to discover/suggest relevant comics for cross-linking
 5. Reader Mode (Choice Interface/Interaction System)
 - a. Fetch the data for the current panel/branch/chapter
 - b. Dynamically render the choice buttons when choice exist
 - c. Make choice (click) fetch preview then go to minting the next batch of chapters
 - d. On-chain choice recording
 - e. “When is an NFT offered?” all branches are collectibles, so to start reading they mint the initial branch, and when they make a choice and want to continue reading they mint the next one. Or maybe it is better to have every chapter as NFT? or a collection of chapters that creates the branch?
 6. Interoperability System (Cross-Story Protocol)
 - a. Cross-Comic Protocol Rules: add some basic rules, i like the idea of “all links are public and permissionless initially” and then creators choose to keep it or not, or maybe have some option in the branch builder to allow it for a genre(s)?
 7. Solana JSON-RPC Provider
 8. Wallet Integration
- KPI's, I think they can be addressed later

Gemini

Of course. It's a wise move to refocus on the absolute core interactions for a Proof-of-Concept (PoC). Let's review this streamlined scope against your value proposition.

Considering it's Sunday evening in Caracas, this level of focus is perfect for defining a clear set of tasks for the upcoming week. This plan cuts through the noise and gets straight to the heart of your innovation.

Part 1: Do These Stories Truly Hit the Mark?

Yes, absolutely. These two user functions are the perfect choice for a PoC because they represent the fundamental, indivisible loop of your entire "comicverse":

1. **Branching Narrative Builder (Creation):** This directly serves your promise to **"ease the burden on creators."** The drag-and-drop interface is the tangible manifestation of that goal. It's the foundational tool that enables every other pillar—without it, there is no scarcity, no collaboration, and no "comicverse." It is the seed of the entire supply side.
2. **Make Choices (Consumption & Connection):** This function delivers the core value to the reader: the **"unique collectible journey"** and the **"personalized story."** The cross-comic "Jump to This Story" feature is a stroke of genius for a PoC because it's the simplest possible way to demonstrate the grand vision of an **"interconnected comicverse."**

Together, these two functions create a minimum viable ecosystem. They prove that the content can be created and that it can be experienced in the novel way you envision. They are the ideal foundation upon which every other feature (dashboards, detailed NFT stats, etc.) can be built later.

Part 2: Are the Requirements Granular Enough?

Your list of technical requirements is an excellent **architectural blueprint** for this PoC. It correctly identifies all the major systems needed to support these two core functions.

However, to move from this blueprint to a developer's task list, you need to add one more layer of detail. The current list is the "what"; the next step is defining the "how."

Here's an example of that next level of granularity for your key requirements:

- **From:** Branch storage contract
- **To (Technical Spec):**
 - **struct Panel:** creator: Pubkey, comic_id: u32, content_uri: String (link to Arweave), choices: Vec<Choice>.
 - **struct Choice:** text: String, destination_type: u8 (0 for Panel, 1 for Comic), destination_address: Pubkey.
 - **Function Signature:** fn create_panel_with_choices(ctx: Context<...>, comic_id: u32, content_uri: String, choices: Vec<Choice>) -> Result<()>
 - *This level of detail makes it clear how to implement the builder's output on-chain.*
- **From:** NFT minting contract
- **To (Technical Spec for a PoC):**
 - **Clarification:** Since your focused user stories don't include the *act of minting* yet, this contract can be simplified for the PoC.

- **Initial Goal:** Focus on creating an on-chain `CollectibleMetadata` account linked to a specific `Panel` or `Choice`.
- **struct `CollectibleMetadata`:** `panel_address: Pubkey, supply: u64, price: u64, minted_count: u64`.
- *This proves you can associate economic properties with narrative elements, deferring the actual `mint` transaction to a fast-follow feature.*
- **From:** `Creator Mode` (`Branching Narrative Builder`)
- **To (API Endpoint Definition):**
 - **POST `/api/v1/comic/publish`**
 - **Request Body (JSON):** A representation of the entire narrative graph created on the frontend (e.g., `{"panels": [...], "branches": [...]}`).
 - **Action:** This endpoint's backend logic would be responsible for uploading all assets to Arweave and then constructing and sending the batch of transactions to the on-chain program.

Your requirements are at the right level for strategic planning. The next step is this tactical, pre-code specification.

Part 3: What's Missing or Unclear?

Focusing on a PoC reveals the most critical, foundational questions that must be answered before writing the first line of code.

A. Critical Unclear Technical Decisions (The "How")

1. **Transaction Granularity & The "Draft" State:** This is the most important missing detail. When a creator is using the `Branching Narrative Builder`, does every single drag-and-drop action trigger a Solana transaction? That would be slow and costly.
 - **Question:** You need to define a **"Drafting Workflow."** The creator likely builds the entire comic locally (in the browser's state). When they are ready, they hit a single "Publish to Chain" button, which then executes the batch of transactions. How is this draft state saved? How does the UI communicate the cost of publishing?
2. **Data Fetching Strategy (Performance):** For the `Make Choices` function, a reader needs the next panel instantly.
 - **Question:** Does the frontend make a new RPC call to fetch the next panel *after* every choice is made? This could feel slow. Or, does **the frontend intelligently pre-fetch the data for all possible next choices as soon as a choice point is displayed**? This is a critical decision for user experience.
3. **Cross-Comic Protocol Rules:** The `Cross-comic linking registry` is a powerful idea, but its rules are undefined.
 - **Question:** Does Creator Y need to *approve* a link from Creator X? If not, what prevents someone from linking to the middle of another comic in a way that breaks its narrative or is otherwise abusive? You must define the rules of this protocol, even if they are simple for the PoC (e.g., all links are public and permissionless initially).

B. Missing PoC-Level Components & Content

1. **The "Genesis" Comics:** A **Reader** cannot make choices, and a **Creator** cannot use the cross-link feature, if the platform is an empty void.
 - **Missing Task:** Your PoC plan *must* include the creation of at least **two or three simple, interconnected "genesis" comics** that will be pre-loaded onto the platform. This is the content you will use to test, demo, and validate every feature.
2. **The Link Between Narrative and Value:** Your stories perfectly define the builder and the reader experience. Your requirements list an **NFT minting contract**.
 - **Missing Logic:** The specific **trigger** that connects the reader's action to the NFT contract is not yet defined in the user stories. When is an NFT offered? Is it upon choosing a path that the creator marked as "premium"? Is it at the end of a chapter? You need one simple user story to bridge this gap, for example: *"When a reader chooses a branch marked as 'collectible', they are prompted to mint the NFT of that choice to proceed."*

C. The Single Most Important Strategic Question for the PoC

- **Defining Success:** What does success look like? Before you build, you must define your Key Performance Indicators (KPIs) for this PoC.
 - **Examples:**
 - "A first-time creator can successfully build and publish a 10-panel branching comic in under 30 minutes."
 - "A reader, on average, explores at least 3 unique branches per session."
 - "The time between a reader clicking a choice and the next panel appearing is less than 1.5 seconds."

Defining these metrics now will ensure your technical decisions are aligned with a clear, measurable, and achievable goal.

Deep Seek

Based on your revised focus on the two core interactions and technical requirements, here's a focused critique addressing alignment with your value proposition and technical granularity:

 ****Strengths & Alignment****

1. ****Creator Burden Reduction****

- Drag-and-drop builder directly addresses ****"ease the burden on creators"**
- Visual linking of choices → panels enables ****"novel monetization in a simple way"**
- ***Value Prop Match*: 90%**

2. **Immersive Reader Experience**

- Integrated choice buttons + cross-comic jumps deliver **"personalized story crafted by choices"**
- ***Value Prop Match***: Strong on comicverse pillar

3. **Core Tech Foundation**

- Requirements cover essential blockchain components (storage, minting, linking)
- On-chain data structures enable **"verifiable digital scarcity"**

Critical Gaps & Solutions

1. Creator Burden Still Too High

Missing in Requirements:

- **Gasless Drafting System**:

```
```rust
```

```
// REQUIRED ADDITION
```

```
struct DraftChapter {
 panels: OffchainVec<IPFSCid>, // Stored in DB until minting
 choices: Vec<DraftChoice>, // Editable without on-chain fees
}
...
```

- **Auto-Linking Tools**: No way to discover/suggest relevant comics for cross-linking
- **Collaboration Workflows**: Missing joint editing/approval mechanisms

**Solution**:

- Add **local draft mode** with session storage
- Implement **comic discovery API**: `GET /comics?genre=sci-fi&status=active``

#### #### **2. Blockchain UX Friction Unresolved**

## **\*\*Unaddressed Issues\*\*:**

- Wallet pop-ups on every choice → breaks immersion
- Solana confirmation delays (400ms+) disrupt reading flow

## **\*\*Required Tech Additions\*\*:**

```mermaid

sequenceDiagram

Reader->> Frontend: Clicks choice

Frontend->> Local Cache: Immediately show next panel

Frontend->> Worker: Queue choice_tx

Worker->> Blockchain: Batch submit choices every 60s

...

**3. Vague Cross-Story Implementation**

****Granular Components Needed**:**

1. **Chapter Addressing Standard**

```rust

// REQUIRED SCHEMA

struct CrossLink {

source\_chapter: Pubkey,

target: ChapterIdentifier, // comic\_id/chapter\_index

version: u64 // Immutable reference

}

...

#### **2. \*\*State Passing Protocol\*\***

- Missing: How character attributes (health, items) transfer between comics (*What????*)

#### **3. \*\*Link Validation Endpoint\*\***

`GET /validate-link?source=X&target=Y` → Returns `{ valid: bool, fee: float }`

#### #### \*\*4. Incomplete On-Chain Data Structures\*\*

**Critical Additions**:

```
```rust
```

```
// CURRENT (INSUFFICIENT)
```

```
struct PanelAccount {
```

```
    content_cid: String
```

```
}
```

```
// REQUIRED UPGRADE
```

```
struct Chapter {
```

```
    panels: Vec<IPFSCid>,
```

```
    choices: Vec<Choice>,
```

```
    // NEW FIELDS
```

```
    state_passport: Vec<StateRule>, // e.g. "Requires: health>10"
```

```
    cross_links: Vec<CrossLink>
```

```
}
```

```
...
```

🛠️ **Granular Technical Mapping**

Blockchain Programs (Solana Anchor)

Function	Program	Instructions	
-----	-----	-----	
Choice Recording	<code>`branch_recorder.so`</code>	<code>`record_choice(chapter, choice_index, reader)`</code>	
Cross-Link Verification	<code>`comic_registry.so`</code>	<code>`verify_link(source, target) → bool`</code>	
NFT Minting	<code>`chapter_minter.so`</code>	<code>`mint_chapter(comic, chapter, rarity)`</code>	

****APIs & Services****

```mermaid

flowchart TD

A[Reader Frontend] --> B{Choice Made?}

B -->|Yes| C[Choice Engine]

C --> D[Local Cache]

C --> E[Tx Queue]

F[Creator Tool] --> G[Comic Draft API]

G --> H[IPFS Upload]

H --> I[Chapter Minter]

...

#### #### **\*\*Missing Endpoints\*\***

##### 1. **\*\*Choice Batching Service\*\***

`POST /queue-choice { session\_id, chapter, choice }`

##### 2. **\*\*Cross-Story Preview\*\***

`GET /preview?chapter\_id=XYZ → { title, cover\_img, creator }`

##### 3. **\*\*State Validation\*\***

`POST /check-access { reader\_wallet, chapter\_id } → { accessible: bool }`

#### ### 🚀 **\*\*Recommendations for PoC Viability\*\***

##### 1. **\*\*Optimistic UI for Choices\*\***

- Show next panel immediately → sync to blockchain later
- Use session storage for temporary journey tracking

##### 2. **\*\*Standardized Chapter Identifier\*\***

- Implement universal addressing: `comic\_owner/comic\_id/chapter:index`
- Store in `CrossLinkRegistry` account

### 3. **Creator Draft System**

- Allow full comic creation offline
- Single transaction to mint entire branch structure

### 4. **State Passport NFT**

```
```rust
// REQUIRED ADDITION

struct Passport {
    owner: Pubkey,
    health: u8,
    items: Vec<Pubkey> // Linked asset NFTs
}
```
```

- Grants cross-comic continuity

### 5. **Minimum Viable Cross-Story Flow**

- Start with **unidirectional links** ( $A \rightarrow B$ , not  $B \rightarrow A$ )
- Implement **immutable linking** (no updates after publish)

### ### 🗝️ **Critical Path Adjustments**

#### 1. **Replace "PanelAccount" with Chapter-Centric Model**

- Panels belong to chapters → reduce on-chain accounts

#### 2. **Simplify BranchAccount Structure**

```
```rust
struct Branch {
    from_chapter: Pubkey,
```

choices: Vec<Choice> // Embed directly

}

...

3. ****Delay Advanced Monetization****

- PoC should use fixed-price mints
- Dynamic pricing → Phase 2

****Test Metric****:

****"Can a reader complete a 5-choice path with 1 cross-comic jump without seeing a wallet popup after the initial sign?"****

This maintains your vision while solving the core UX/technical hurdles for a viable PoC.

Part C: Granularity & Clarity Refinement

1) Final Manual Review & Refinement

Action: Review every user story and function from Part B through the following lens. Your goal is to simplify and clarify, preparing for technical implementation.

- **De-Jargon:** Is the language free of technical jargon? Rewrite any story so that a non-technical stakeholder could understand it perfectly.
- **Granularity:** Does each story represent a single, concrete action?
- **Atomicity:** If any story or action involves more than one distinct step (e.g., "User signs up and creates a profile"), break it down into separate, smaller stories (e.g., Story 1: "User signs up for an account," Story 2: "User fills out their profile information").
- **Clarity of Action:** Is the action the user needs to take completely clear? Is the desired outcome obvious?
- **No Overlap:** Review your complete list of user stories. Do any two stories describe the same action? If so, merge them or eliminate the redundancy.

Part C Refinement Log

- "Solana JSON-RPC Provider" -> Solana blockchain provider to talk with the blockchain. But this is a functionality needed, not something an user has to do
- "NFT minting" -> NFT creation

- “Arweave/IPFS” -> decentralized storage

Part D: Defining Potential On-Chain Requirements

1) Brainstorming On-Chain Requirements for Each User Story

- **Action:** Take your final, refined list of user stories from Part C. For each individual story, create a simple bulleted list of potential on-chain requirements needed to make it happen. This is a brainstorming exercise to translate user actions into technical needs.

User Story: "User creates a new branching narrative."

Potential On-Chain Requirements:

- This is an account to store the amount of chapters, price (for each chapter?)
- Initialize branch
- Store if is the genesis
- Store which is the previous and next branches
- Store which comic belongs
- Stores decision nodes/relationships

User Story: "User creates chapter."

Potential On-Chain Requirements:

- Initialize chapter
- Store which branch belongs
- Store if is the continuation from another comic branch, which comic and branch

User Story: "User links chapter to another comic."

Potential On-Chain Requirements:

- Store to what comic is linked
- To what chapter will start next
- Which branch the chapter is in and the next one
- Link function for chapters, account

User Story: "User links branch to another comic." (not sure if this can be merged with link chapter"

Potential On-Chain Requirements:

- Link function for branches, account.
- Store from and to branches
- Store to what comic is linked

User Story: "User make choice in branch."

Potential On-Chain Requirements:

- Choice is stored in state
- Choice function, account
- Store which branch is the previous and next

User Story: "User mint branch to read."

Potential On-Chain Requirements:

- Mint function
- If is linked, then split royalties