Part A: User Stories & On-Chain Requirements

Core User Personas:

- ### Indie Movie Creators (Producers/Filmmakers):

Why essential:
They're the *supply side* of the marketplace, Indie creators are ideal for early adoption because they're usually underfunded, open to experimentation, and willing to use alternative financing channels. They'll test whether your tokenized crowdfunding mechanism can genuinely fund creative work and simplify distribution.

- ### Movie Fans (Community Backers):

Why essential:
They're the *emotional engine* of your entire model. If no fans are willing to back a film for sentimental, cultural, or speculative reasons, the premise collapses. Their participation validates the thesis that fandom can become ownership. Even a small but passionate fan group can provide critical early proof of user engagement and token demand.

- ### Platform Developer / Admin (You):

Why essential:
For a PoC, you *are* the infrastructure. You'll test deployment, security, token minting, crowdfunding logic, and UI/UX for creators and backers. Minimal admin involvement aligns with your vision of decentralization, but initially, your oversight ensures correct program execution, dispute handling, and fund routing.

Function Maps:

1. Indie Movie Creators (Producers/Filmmakers)

Goal: Launch a tokenized crowdfunding campaign for their film.
Key Functions / Interactions:

- **Create a Movie Pitch:** Upload synopsis, teaser/trailer (optional), budget breakdown, and funding goal.
- **Deploy Movie Token:** Mint a unique Solana-based token (via your dApp) representing participation or backing in their film.
- **Set Tokenomics:** Define total token supply, allocation (e.g. % for public sale, % for creators, % for platform fees), and crowdfunding duration.
- **Launch Crowdfunding Campaign:** Publish the token sale on the marketplace where fans can contribute USDC or SOL.
- **Monitor Funding Progress:** Real-time dashboard showing how much has been raised.
- **Withdraw Funds (post-raise):** Once the campaign ends, funds are released to the creator's wallet (possibly escrow-based milestone release in future).

Purpose in PoC: Demonstrates creators can easily mint tokens, raise funds, and interact with the decentralized platform without deep blockchain knowledge.

2. Movie Fans (Community Backers)

Goal: Discover projects they care about and back them with funds, receiving tokens in return.
Key Functions / Interactions:

- **Explore Movie Campaigns:** Browse ongoing film pitches, with metadata like trailer, creator, and genre.
- **Connect Wallet:** Use a Solana wallet (e.g., Phantom) to interact with the dApp.
- **Buy Movie Tokens:** Exchange USDC/SOL for movie-specific tokens during crowdfunding.
- **View Portfolio:** Track owned movie tokens and any associated perks (e.g. early screening access, digital collectibles, or future rewards).
- **Trade Tokens (Optional):** Post-PoC, tokens could be tradable on a DEX for liquidity testing.

Purpose in PoC: Proves that fans will engage emotionally *and financially* with tokenized movie ownership, validating the user experience and economic appeal.

3. Platform Developer / Admin (You)

Goal: Ensure smooth operation, transparency, and security of the platform.
Key Functions / Interactions:

- **Deploy Smart Contracts:** Write and deploy Solana programs for token creation, crowdfunding logic, and fund disbursement.
- **Platform UI Management:** Oversee campaign listings, troubleshoot errors, and handle disputes (if any).
- **Transaction Verification:** Ensure all token minting, purchases, and fund transfers execute correctly and securely.
- **Analytics and Testing:** Track metrics like total raised, number of contributors, and transaction success rate.
- **Security Oversight:** Implement basic verification to prevent fraudulent campaigns.

Purpose in PoC: Validates the technical reliability and decentralized workflow, ensuring your platform can handle real users with minimal oversight.

Potential On-Chain Requirements:

1. **TokenFactory Program:**
    - `create_movie_token()` – mints new SPL token and stores metadata URI
    - `set_tokenomics()` – defines initial distribution rules
2. **Crowdfunding Program:**
    - `init_campaign()` – links token mint to funding parameters
    - `contribute()` – transfers USDC/SOL from fan to campaign vault
    - `finalize_campaign()` – allows withdrawal if goal met
    - `refund()` – returns funds to contributors if failed

Your stories imply these exist but don't define *what each transaction does*. For mapping, you need these atomic actions clarified.

1) Launch Crowdfunding Campaign

Potential On-Chain Requirements:

- Create a new **Campaign account (PDA)** storing:
    - `creator_pubkey`
    - `movie_token_mint_address`
    - `funding_goal`
    - `deadline`
    - `total_raised` (initialized to 0)
- Enable a **campaign state machine** (e.g., `Draft`, `Active`, `Ended`).
- Store the IPFS hash of the movie pitch inside this account.
- Only the creator should have permission to activate their own campaign.

2) Mint Movie Token

Potential On-Chain Requirements:

- A smart contract function to **create and initialize a new SPL token mint**.
- Store metadata (name, symbol, total supply, mint authority) in an **on-chain program-derived account (PDA)**.
- Assign initial mint authority to the platform program to prevent re-minting.
- Initialize associated token accounts for the creator and the platform if needed.
- Emit an event (e.g., `MovieTokenCreated`) for off-chain indexing.

3) Buy Movie Tokens

Potential On-Chain Requirements:

- Function to **contribute USDC or SOL** to a specific campaign.
- Funds transferred into the **campaign's escrow PDA**.
- Update `total_raised` on the campaign account.
- Mint equivalent amount of **Movie Tokens** to the contributor's wallet (pro-rata).
- Fail if:
    - Campaign not `Active`.
    - Deadline passed.
    - Funding goal already met.
- Emit `ContributionReceived` event.

4) Withdraw Raised Funds

Potential On-Chain Requirements:

- Function to **release funds** from the campaign's escrow account to the creator's wallet.
- Must check:
    - Campaign reached funding goal.
    - Deadline has passed.
    - Caller is the campaign creator.
- Once executed, mark campaign state as `Completed`.
- Optional: Escrow can be split into milestones (future feature).
- Emit `FundsWithdrawn` event.

5) Trade Tokens (Post-PoC / Optional)

Potential On-Chain Requirements:

- Integrate with a **DEX (e.g., Raydium or Meteora)** to create a liquidity pool for movie tokens.
- Platform program might need to:
    - Approve token listing request.
    - Optionally seed initial liquidity from treasury or partner funds.
- Once live, trading is handled by existing DEX protocols.

Part B: Process Appendix:

Part A: Initial User & Function Mapping

1) Manual User Brainstorming

- **Direct Users:** Movie producers, Indie movie creators, Movie fans, Investors.
- **Indirect Users/Beneficiaries:** Movie industry, moviegoers, theatres.
- **Administrators/Moderators:** Project shouldn't take too many Admins. It will be an open platform not requiring much regulation. Only the developer
- **Stakeholders:** Everyone who uses the platform and invests in movies will be a stakeholder. Depending on the success the rest of the movie industry might be affected.

2) AI-Assisted User Prioritization

- **AI-Assisted Task:** Take the comprehensive list of potential user types you brainstormed in the previous step. You will now use AI to help you prioritize this list and focus on the most critical users for a Proof-of-Concept (POC).
- **AI Prompt:** Prompt the AI with: "My project's value proposition is: The core value proposition of this project is to give the general public the opportunity to support and gain ownership in movie productions, an opportunity that has so far been limited to a select few. Since movies evoke strong emotions and build passionate fandoms, it's natural for fans to want to contribute, even in small ways, to the creation of their favorite films. At the same time, creators benefit by gauging audience interest before committing significant resources, allowing them to make more informed production decisions. Here is a brainstormed list of all potential user types: **Direct Users:** Movie producers, Indie movie creators, Movie fans, Investors, **Indirect Users/Beneficiaries:** Movie industry, moviegoers, theatres, **Administrators/Moderators:** Project shouldn't take too many Admins. It will be an open platform not requiring much regulation. Only the developer, **Stakeholders:** Everyone who uses the platform and invests in movies will be a stakeholder. Depending on the success the rest of the movie industry might be affected. 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."

- AI Output:

1. Indie Movie Creators (Producers/Filmmakers)

Why essential:
They're the *supply side* of your marketplace — without films, there's nothing to fund or trade. Indie creators are ideal for early adoption because they're usually underfunded, open to experimentation, and willing to use alternative financing channels. They'll test whether your tokenized crowdfunding mechanism can genuinely fund creative work and simplify distribution.

---

2. Movie Fans (Community Backers)

Why essential:
They're the *emotional engine* of your entire model. If no fans are willing to back a film for sentimental, cultural, or speculative reasons, the premise collapses. Their participation validates the thesis that fandom can become ownership. Even a small but passionate fan group can provide critical early proof of user engagement and token demand.

---

3. Crypto Investors (Early Liquidity Providers)

Why essential:
They're the bridge between narrative and market mechanics. You'll need initial liquidity in the pools and a user base comfortable with wallets, tokens, and DEX trading. Crypto-native investors will test your technical flow — token issuance, LP setup, and trading — proving the marketplace component works end-to-end.

---

4. Platform Developer / Admin (You)

Why essential:
For a PoC, you *are* the infrastructure. You'll test deployment, security, token minting, crowdfunding logic, and UI/UX for creators and backers. Minimal admin involvement aligns with your vision of decentralization, but initially, your oversight ensures correct program execution, dispute handling, and fund routing.

That's enough.
You don't need moviegoers, theatres, or the broader industry yet — they only matter once you've shown that (1) creators can raise funds, (2) fans can contribute, and (3) the tokens behave like functional micro-equity.

If you were pitching this to a grant or incubator, you'd phrase the focus like this:

> "Our PoC targets three core user groups — indie filmmakers, movie fans, and crypto investors — to validate the end-to-end cycle of tokenized film funding and trading. These users together test the platform's creative, social, and economic assumptions before scaling to the wider film industry."

- **Manual Action & Output:** Analyze the AI's recommendations and rationale. Decide on your final, prioritized list of the top 2-5 key user types. Document your final list and a brief explanation of your decision-making process, noting where you agreed or disagreed with the AI's suggestions and why.
- Manual Action: I agree with AI about Indie Movie Creators, Movie Fans and the Platform Developer recommendations.

- Final List:
- **Indie Movie Creators (Producers/Filmmakers)**
- **Why essential:**
  They're the *supply side* of the marketplace, Indie creators are ideal for early adoption because they're usually underfunded, open to experimentation, and willing to use alternative financing channels. They'll test whether your tokenized crowdfunding mechanism can genuinely fund creative work and simplify distribution.
- **Movie Fans (Community Backers)**
- **Why essential:**
  They're the *emotional engine* of your entire model. If no fans are willing to back a film for sentimental, cultural, or speculative reasons, the premise collapses. Their participation validates the thesis that fandom can become ownership. Even a small but passionate fan group can provide critical early proof of user engagement and token demand.
- **Platform Developer / Admin (You)**
- **Why essential:**
  For a PoC, you *are* the infrastructure. You'll test deployment, security, token minting, crowdfunding logic, and UI/UX for creators and backers. Minimal admin involvement aligns with your vision of decentralization, but initially, your oversight ensures correct program execution, dispute handling, and fund routing.

3) Core Function Mapping

- **AI Task:** Now, using your final prioritized list of users, map out their functions. Prompt the AI: "For a project with this value proposition The core value proposition of this project is to give the general public the opportunity to support and gain ownership in movie productions, an opportunity that has so far been limited to a select few. Since movies evoke strong emotions and build passionate fandoms, it's natural for fans to want to contribute, even in small ways, to the creation of their favorite films. At the same time, creators benefit by gauging audience interest before committing significant resources, allowing them to make more informed production decisions and focusing on these prioritized user types:
- **Indie Movie Creators (Producers/Filmmakers)**
- **Why essential:**
  They're the *supply side* of the marketplace, Indie creators are ideal for early adoption because they're usually underfunded, open to experimentation, and willing to use alternative financing channels. They'll test whether your tokenized crowdfunding mechanism can genuinely fund creative work and simplify distribution.
- **Movie Fans (Community Backers)**
- **Why essential:**
  They're the *emotional engine* of your entire model. If no fans are willing to back a film for sentimental, cultural, or speculative reasons, the premise collapses. Their participation validates the thesis that fandom can become ownership. Even a small but passionate fan group can provide critical early proof of user engagement and token demand.
- **Platform Developer / Admin (You)**
- **Why essential:**
  For a PoC, you *are* the infrastructure. You'll test deployment, security, token minting, crowdfunding logic, and UI/UX for creators and backers. Minimal admin involvement aligns with your vision of decentralization, but initially, your oversight ensures correct program execution, dispute handling, and fund routing.

  help map out the key functions or interactions each user would need to perform."
- **Output:** A list of key functions and interactions mapped to your prioritized user types.
- AI Output:

1. Indie Movie Creators (Producers/Filmmakers)

Goal: Launch a tokenized crowdfunding campaign for their film.
Key Functions / Interactions:

- **Create a Movie Pitch:** Upload synopsis, teaser/trailer (optional), budget breakdown, and funding goal.

- **Deploy Movie Token:** Mint a unique Solana-based token (via your dApp) representing participation or backing in their film.

- **Set Tokenomics:** Define total token supply, allocation (e.g. % for public sale, % for creators, % for platform fees), and crowdfunding duration.

- **Launch Crowdfunding Campaign:** Publish the token sale on the marketplace where fans can contribute USDC or SOL.
- **Monitor Funding Progress:** Real-time dashboard showing how much has been raised.
- **Withdraw Funds (post-raise):** Once the campaign ends, funds are released to the creator's wallet (possibly escrow-based milestone release in future).

Purpose in PoC: Demonstrates creators can easily mint tokens, raise funds, and interact with the decentralized platform without deep blockchain knowledge.

---

2. Movie Fans (Community Backers)

Goal: Discover projects they care about and back them with funds, receiving tokens in return.
Key Functions / Interactions:

- **Explore Movie Campaigns:** Browse ongoing film pitches, with metadata like trailer, creator, and genre.
- **Connect Wallet:** Use a Solana wallet (e.g., Phantom) to interact with the dApp.
- **Buy Movie Tokens:** Exchange USDC/SOL for movie-specific tokens during crowdfunding.
- **View Portfolio:** Track owned movie tokens and any associated perks (e.g. early screening access, digital collectibles, or future rewards).
- **Trade Tokens (Optional):** Post-PoC, tokens could be tradable on a DEX for liquidity testing.

Purpose in PoC: Proves that fans will engage emotionally *and financially* with tokenized movie ownership, validating the user experience and economic appeal.

---

3. Platform Developer / Admin (You)

Goal: Ensure smooth operation, transparency, and security of the platform.
Key Functions / Interactions:

- **Deploy Smart Contracts:** Write and deploy Solana programs for token creation, crowdfunding logic, and fund disbursement.
- **Platform UI Management:** Oversee campaign listings, troubleshoot errors, and handle disputes (if any).
- **Transaction Verification:** Ensure all token minting, purchases, and fund transfers execute correctly and securely.
- **Analytics and Testing:** Track metrics like total raised, number of contributors, and transaction success rate.
- **Security Oversight:** Implement basic verification to prevent fraudulent campaigns.

Purpose in PoC: Validates the technical reliability and decentralized workflow, ensuring your platform can handle real users with minimal oversight.

Summary of Core Interactions (PoC Loop)

1. **Creator** → launches campaign, mints token.
2. **Fan** → buys tokens and becomes a supporter/holder.
3. **Admin** → oversees smart contract integrity and ensures proper fund flow.

This triangle — creator, backer, and protocol — forms your minimal self-sustaining ecosystem.

4) Deriving Core POC Requirements

- **Manual Task:** From the function map you just created, identify the top 1-2 most critical user stories or interaction paths essential for a POC.
- The Top 2 most critical user stories are Indie Movie creators and Movie Fans.
- **AI Task:** Prompt the AI: "Based on these top 1-2 critical user interactions:

1. Indie Movie Creators (Producers/Filmmakers)

Goal: Launch a tokenized crowdfunding campaign for their film.
Key Functions / Interactions:

- **Create a Movie Pitch:** Upload synopsis, teaser/trailer (optional), budget breakdown, and funding goal.

- **Deploy Movie Token:** Mint a unique Solana-based token (via your dApp) representing participation or backing in their film.

- **Set Tokenomics:** Define total token supply, allocation (e.g. % for public sale, % for creators, % for platform fees), and crowdfunding duration.

- **Launch Crowdfunding Campaign:** Publish the token sale on the marketplace where fans can contribute USDC or SOL.

- **Monitor Funding Progress:** Real-time dashboard showing how much has been raised.

- **Withdraw Funds (post-raise):** Once the campaign ends, funds are released to the creator’s wallet (possibly escrow-based milestone release in future).

Purpose in PoC: Demonstrates creators can easily mint tokens, raise funds, and interact with the decentralized platform without deep blockchain knowledge.

---

2. Movie Fans (Community Backers)

Goal: Discover projects they care about and back them with funds, receiving tokens in return.
Key Functions / Interactions:

- **Explore Movie Campaigns:** Browse ongoing film pitches, with metadata like trailer, creator, and genre.

- **Connect Wallet:** Use a Solana wallet (e.g., Phantom) to interact with the dApp.

- **Buy Movie Tokens:** Exchange USDC/SOL for movie-specific tokens during crowdfunding.

- **View Portfolio:** Track owned movie tokens and any associated perks (e.g. early screening access, digital collectibles, or future rewards).

- **Trade Tokens (Optional):** Post-PoC, tokens could be tradable on a DEX for liquidity testing.

Purpose in PoC: Proves that fans will engage emotionally *and financially* with tokenized movie ownership, validating the user experience and economic appeal.

what are the key technical requirements needed to build a proof-of-concept?"

- **Output:** An initial list of key technical requirements derived from the core user interactions.
- AI Output:

1. Smart Contract Layer (Solana Program)

Purpose: Enable trustless token creation, fund collection, and basic accounting for campaigns.

Requirements:

- **Token Minting Logic:**
  A Solana program that can create a new SPL token (movie token) per campaign with defined supply and metadata.

- **Crowdfunding Contract:**
  Handles USDC/SOL contributions, stores contributor details, and issues corresponding movie tokens automatically.

  - Set funding goals, campaign deadlines, and track total raised.

  - Optionally, escrow funds until the campaign ends.

- **Withdraw Mechanism:**
  Allows the creator to withdraw funds only after campaign success (or returns to backers if it fails, in later iterations).

- **Program Accounts Management:**
  Store campaign info (creator address, token mint, target amount, amount raised, deadline).

- **Upgradeable Logic (Optional for PoC):**
  Use Anchor framework for easier Solana program development and potential upgrades.

---

2. Frontend (dApp Interface)

Purpose: User-friendly gateway for creators and fans.

Requirements:

- **Campaign Creation UI:**
    - Form to input title, description, budget, trailer link, and funding goal.
    - Trigger on-chain transaction to deploy the movie token and campaign via the smart contract.
- **Campaign Listing Page:**
    - Displays all active campaigns with details and progress bars.
- **Wallet Integration:**
    - Phantom wallet (or Solflare) connect/disconnect for both creators and backers.
    - Use Solana libraries (e.g., `@solana/web3.js`, `@solana/wallet-adapter-react`).
- **Contribution Flow:**
    - Let fans buy movie tokens using USDC/SOL and confirm transactions on-chain.
- **Portfolio View:**
    - Display the user's movie token holdings and campaign participation history.

---

3. Backend (Optional / Off-chain Support)

Purpose: Enhance usability and metadata management without relying solely on the blockchain.

Requirements:

- **Metadata Server or IPFS Integration:**
  Store movie details (pitch, trailer, images) in IPFS or Arweave.
- **Indexing & Querying Layer:**
  Use services like **The Graph** (or Helium if Solana-compatible) to index campaigns, contributions, and ownership for fast queries.

- **Basic API Gateway (Node.js / Express):**
  Cache read operations for UI efficiency (campaign list, user portfolios, etc.).

---

4. Token Integration and Marketplace

Purpose: Simulate post-crowdfunding liquidity and demonstrate potential for token trading.

Requirements:

- **Token Standard:** SPL token with metadata (name, symbol, supply, URI).
- **DEX Integration (Optional for PoC):**
  Connect to a DEX like **Raydium** or **Jupiter** to enable mock liquidity pools or simulate swap behavior.
- **Liquidity Demo:**
  For now, seed minimal liquidity manually from your dev wallet to show trade potential.

---

5. Developer / Admin Tools

Purpose: Support platform operation, debugging, and campaign validation.

Requirements:

- **Admin Dashboard:**
  Simple view of all campaigns, funds raised, and token supply stats.
- **Error Handling + Event Logs:**
  UI feedback for failed transactions or network issues.
- **Test Suite:**
  Unit tests in Rust for Solana program logic (Anchor testing framework).

Tech Stack Summary

- **Blockchain:** Solana (Anchor + SPL tokens)
- **Frontend:** React + TypeScript + Solana Wallet Adapter
- **Backend:** Node.js (Express) + IPFS (for media)
- **Deployment:** Netlify/Vercel for UI, Devnet for on-chain testing
- **DEX Integration (Optional):** Raydium/Jupiter SDKs for token swaps

Part B: Adversarial Analysis & Granularity Check

1) Critique & Refine User Stories / Requirements:

- **AI Prompt:** "Review my core user functions/stories [Input from Part A, Step 3] and requirements

1. Indie Movie Creators (Producers/Filmmakers)

Goal: Launch a tokenized crowdfunding campaign for their film.
Key Functions / Interactions:

- **Create a Movie Pitch:** Upload synopsis, teaser/trailer (optional), budget breakdown, and funding goal.
- **Deploy Movie Token:** Mint a unique Solana-based token (via your dApp) representing participation or backing in their film.
- **Set Tokenomics:** Define total token supply, allocation (e.g. % for public sale, % for creators, % for platform fees), and crowdfunding duration.
- **Launch Crowdfunding Campaign:** Publish the token sale on the marketplace where fans can contribute USDC or SOL.
- **Monitor Funding Progress:** Real-time dashboard showing how much has been raised.
- **Withdraw Funds (post-raise):** Once the campaign ends, funds are released to the creator's wallet (possibly escrow-based milestone release in future).

Purpose in PoC: Demonstrates creators can easily mint tokens, raise funds, and interact with the decentralized platform without deep blockchain knowledge.

---

2. Movie Fans (Community Backers)

Goal: Discover projects they care about and back them with funds, receiving tokens in return.
Key Functions / Interactions:

- **Explore Movie Campaigns:** Browse ongoing film pitches, with metadata like trailer, creator, and genre.
- **Connect Wallet:** Use a Solana wallet (e.g., Phantom) to interact with the dApp.
- **Buy Movie Tokens:** Exchange USDC/SOL for movie-specific tokens during crowdfunding.

- **View Portfolio:** Track owned movie tokens and any associated perks (e.g. early screening access, digital collectibles, or future rewards).

- **Trade Tokens (Optional):** Post-PoC, tokens could be tradable on a DEX for liquidity testing.

Purpose in PoC: Proves that fans will engage emotionally *and financially* with tokenized movie ownership, validating the user experience and economic appeal.

- Considering my project's refined value proposition The core value proposition of this project is to give the general public the opportunity to support and gain ownership in movie productions, an opportunity that has so far been limited to a select few. Since movies evoke strong emotions and build passionate fandoms, it's natural for fans to want to contribute, even in small ways, to the creation of their favorite films. At the same time, creators benefit by gauging audience interest before committing significant resources, allowing them to make more informed production decisions, 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?"
- AI Output:

🧩 1. Fit with the Core Value Proposition

Your stories *do* hit the emotional and economic goals of the project:

- Creators get an accessible way to tokenize and fund their films.

- Fans get to own a piece of cultural creation.

This alignment is perfect — your platform's magic lies at that intersection. So in terms of *user intent*, yes, these stories "hit the mark."

But from a **system design perspective**, the stories currently describe *flows*, not *interactions with technical modules*. You'll need to break each story into concrete *state changes* and *data flows* across contracts, APIs, and front-end states.

🧠 2. Granularity Check — What's Missing for Implementation

Here's where you need to sharpen things up:

A. Data Models / Database Schemas (Off-chain Metadata Layer)

You mention "movie pitch," "funding goal," "tokenomics," etc., but you'll need clear data entities. For example:

```
MovieCampaign {

  id: string,

  creator_wallet: string,

  title: string,

  synopsis: string,

  trailer_url: string,

  funding_goal: number,

  amount_raised: number,

  token_mint_address: string,

  sale_start: timestamp,

  sale_end: timestamp,

  tokenomics: {

    total_supply: number,

    public_sale_pct: number,

    creator_allocation_pct: number,

    platform_fee_pct: number

  },

  metadata_uri: string (IPFS),

  status: enum ["active", "completed", "cancelled"]

}
```

You'd also need:

- `User` (wallet-linked) schema
- `Contribution` table (fan, campaign_id, amount, timestamp)
- Optional `Transaction` logs or history

Right now, those aren't explicitly defined in your stories — but they're essential for mapping technical components.

B. Blockchain Programs (On-chain Logic via Solana / Anchor)

You reference the "deploy token" and "launch crowdfunding campaign," but these need distinct program modules:

3. **TokenFactory Program:**
    - `create_movie_token()` – mints new SPL token and stores metadata URI
    - `set_tokenomics()` – defines initial distribution rules
4. **Crowdfunding Program:**
    - `init_campaign()` – links token mint to funding parameters
    - `contribute()` – transfers USDC/SOL from fan to campaign vault
    - `finalize_campaign()` – allows withdrawal if goal met
    - `refund()` – returns funds to contributors if failed

Your stories imply these exist but don't define *what each transaction does*. For mapping, you need these atomic actions clarified.

C. API / Frontend Endpoints

Each user story could translate into one or more API endpoints or contract calls.
For example:

Action	API/Contract Endpoint	Method	Output
Create Movie Pitch	`/api/campaigns/new`	POST	campaign_id
Deploy Token	`token_factory::create_movie_token()`	on-chain	token_mint_address
Launch Campaign	`/api/campaigns/launch`	POST	success/fail
Contribute to Campaign	`crowdfunding::contribute()`	on-chain	tx_signature
View Portfolio	`/api/user/:wallet/portfolio`	GET	list of owned tokens

Without these, a developer wouldn't know what to actually *build* or how to *interact* with the system.

D. Missing or Underdefined Parts

Here's what's currently unclear or not granular enough:

1. **Escrow / Fund Management Model:**
    - How are funds locked and released?
    - Is there a minimum goal check on-chain before funds unlock?
2. **Tokenomics Validation:**
    - How do you ensure percentages sum to 100%?
    - Is supply fixed or mintable later for revenue sharing?
3. **Marketplace / Token Visibility Layer:**

    - Will token listings be automatic after campaign launch?
    - Are fans able to sort/filter based on genre, funding progress, etc.?
4. **Wallet Interactions:**
    - How does the frontend handle transaction signing failures or wallet disconnections?
5. **Security & Verification:**
    - How to verify creators before letting them raise funds? (PoC may skip KYC, but it’s worth noting.)
6. **DEX Integration (Post-PoC Roadmap):**
    - What mechanism will make tokens tradable later? Manual listing, aggregator API, or direct liquidity pool?

E. Suggested Refinement Example

Instead of:

> “Deploy Movie Token: Mint a unique Solana-based token (via your dApp) representing participation or backing in their film.”

You could specify:

> “When a creator submits their pitch, the frontend calls `token_factory::create_movie_token()` passing metadata URI (stored on IPFS). The contract mints a new SPL token, returns its mint address, and stores it in the `MovieCampaign` record in the backend database.”

That’s now technically actionable — a dev could implement it directly.

⚙️ 3. Summary of Next Steps

To make your user stories technically complete, add:

- **Explicit state transitions** — what changes after each user action.
- **Mapped technical modules** — contract names, functions, APIs.

- **Data persistence structure** — what's stored on-chain vs off-chain.

- **UX states** — what feedback users get (pending tx, confirmed, failed).

- **Action:** Analyze the feedback. Refine your user stories for better alignment and ensure your technical requirements meet the necessary granularity for technical planning
- **Documentation:** Document the AI critique, your analysis, and the rationale for your refined user stories and technical requirements.
- My Analysis: I think the data storage will be performed using PDAs instead of off-chain databases. But the example schemas generated could be useful for creation of PDA state. The solana program instructions and handlers generated seem correct to me for the time being. The frontend and API creation is not the main focus of this PoC but if time permits I'll include it into the capstone. The questions raised in the granular missing section seem trivial and will be answered during building of the capstone.

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.

Documentation: In your appendix, create a "Part C Refinement Log." For each change you make, briefly note the "Before" story and the "After" story with a short rationale for the change (e.g., "Split into two stories for atomicity," "Removed jargon 'API call' and replaced with 'sends information'").

Story 1: Create a Movie Pitch

Plain Version:
A creator opens the platform, writes a short description of their movie, adds optional images or a teaser video, and sets how much money they need to raise.

Outcome:
A movie proposal appears in the “Draft” section of their dashboard.

Notes:

- Keep this step off-chain. Store metadata in the database or IPFS.

- One clear, single action — no jargon.

Story 2: Define Token Details

Plain Version:
The creator decides how many tokens will exist for their movie, how many will be sold to the public, how many they will keep, and how long the sale will last.

Notes:
This story isolates decision-making from blockchain interaction — atomic and clear.

Story 3: Mint Movie Token

Plain Version:
After confirming their details, the creator clicks “Deploy Token.” The system mints a new movie token on the Solana blockchain and links it to the project.

Outcome:
A blockchain transaction confirms the new token’s creation, and the mint address is shown on the creator’s dashboard.

Notes:
This is the first *on-chain* action, simple enough for a non-technical stakeholder to grasp.

Story 4: Launch Crowdfunding Campaign

Plain Version:
The creator publishes their project so that fans can start buying tokens with USDC or SOL.

Outcome:
The campaign moves from “Draft” to “Active” status on the platform marketplace.

Story 5: Monitor Progress

Plain Version:
While the campaign is live, the creator can view how much money has been raised and how many people have contributed.

Outcome:
Real-time updates appear on the dashboard (read-only).

Story 6: Withdraw Raised Funds

Plain Version:
When the campaign ends, if the goal is met, the creator can withdraw the collected funds to their connected wallet.
If not, the funds remain locked or are refunded to contributors.

Outcome:
Successful withdrawal triggers a “Completed” campaign state.

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: "Producer creates a new movie fundraiser."

Potential On-Chain Requirements:

- Need a way to create a new on-chain account to hold fundraiser data.
- This account must store the campaign Producer address for permissions.
- It needs to store a fundraising goal and a deadline.
- Must initialize a "total raised" amount to zero.

User Story: "Fan contributes funds to a fundraiser."

Potential On-Chain Requirements:

- Need a function to accept a SOL transfer from the Fan.
- The funds must be sent to an account controlled by the fundraiser.
- The "total raised" on the fundraiser account must be updated with the new amount.
- The function should fail if the fundraiser deadline has passed.
- ## Launch Crowdfunding Campaign

Potential On-Chain Requirements:

- Create a new **Campaign account (PDA)** storing:
  - `creator_pubkey`
  - `movie_token_mint_address`
  - `funding_goal`
  - `deadline`
  - `total_raised` (initialized to 0)
- Enable a **campaign state machine** (e.g., `Draft`, `Active`, `Ended`).
- Store the IPFS hash of the movie pitch inside this account.

- Only the creator should have permission to activate their own campaign.

- ## Mint Movie Token

Potential On-Chain Requirements:

- A smart contract function to **create and initialize a new SPL token mint**.
- Store metadata (name, symbol, total supply, mint authority) in an **on-chain program-derived account (PDA)**.
- Assign initial mint authority to the platform program to prevent re-minting.
- Initialize associated token accounts for the creator and the platform if needed.
- Emit an event (e.g., `MovieTokenCreated`) for off-chain indexing.

- ## Buy Movie Tokens

Potential On-Chain Requirements:

- Function to **contribute USDC or SOL** to a specific campaign.
- Funds transferred into the **campaign's escrow PDA**.
- Update `total_raised` on the campaign account.
- Mint equivalent amount of **Movie Tokens** to the contributor's wallet (pro-rata).
- Fail if:
  - Campaign not `Active`.
  - Deadline passed.
  - Funding goal already met.
- Emit `ContributionReceived` event.

- ## Withdraw Raised Funds

Potential On-Chain Requirements:

- Function to **release funds** from the campaign's escrow account to the creator's wallet.
- Must check:
  - Campaign reached funding goal.
  - Deadline has passed.
  - Caller is the campaign creator.
- Once executed, mark campaign state as `Completed`.
- Optional: Escrow can be split into milestones (future feature).
- Emit `FundsWithdrawn` event.

- ## Trade Tokens (Post-PoC / Optional)

Potential On-Chain Requirements:

- Integrate with a **DEX (e.g., Raydium or Meteora)** to create a liquidity pool for movie tokens.
- Platform program might need to:
  - Approve token listing request.
  - Optionally seed initial liquidity from treasury or partner funds.
- Once live, trading is handled by existing DEX protocols.