Part A: User Stories & On-Chain Requirements Document

Core User Personas

For the StreakSafe Proof-of-Concept (POC), we've identified the following core user personas, reflecting the project's unique dual-responsibility mechanism:

- **Active User:** This is the primary user who stakes SOL to commit to daily habits and submits photo proof of completion. Crucially, after submitting their own check-in, the same user is then assigned and is required to verify the check-ins of three other users in the same category. This persona embodies both the "Habit Committer" and "Peer Verifier" roles, making every participant integral to the decentralized verification process.
- **Admin:** This persona represents the privileged entity responsible for setting and updating the core configuration parameter**s** of the StreakSafe protocol. This includes defining rules like minimum/maximum check-in times, default stake amounts, reward rates, and designating the recipient of slashed funds.
- **Slash Recipient:** While not an actively interacting user in the traditional sense, this persona represents the designated wallet address or entity that receives a portion of the staked SOL when a user's check-in is rejected by the peer consensus. It's a critical component for demonstrating the "high-stakes" accountability of the protocol.

Function Maps

Here's a detailed mapping of the key functions and interactions for each prioritized user type:

Active User (Combines Habit Committer & Peer Verifier roles)

- **Connect their Solana wallet:** The fundamental first step to interact with the protocol.
- **Create and stake for a new habit in a specific category:** Users define their daily habit, select a relevant category (e.g., "Fitness," "Learning"), commit a specified amount of SOL as a stake, and initiate their habit streak. This action also makes them eligible for peer verification assignments.
- **Upload photo proof of habit completion to Arweave:** Users capture and upload an image to Arweave as visual evidence that they've completed their daily task.
- **Submit a daily check-in for their habit:** Users provide the Arweave transaction ID for their photo proof. Upon submission, their check-in is marked as "pending verification" by the system, and simultaneously, the user is assigned three random, pending check-ins from other users in the same category to verify.
- **View their own current streak progress and pending verification tasks:** Users can check their personal dashboard to see their active habits, current streak length, and a list of check-ins they need to verify for others.
- **View photo proof for an assigned check-in:** For each assigned verification task, the user can view the corresponding photo stored on Arweave.
- **Cast a "Verify" vote on an assigned check-in:** After reviewing the photo, the user can approve the check-in as valid.
- **Cast a "Reject" vote on an assigned check-in:** After reviewing the photo, the user can dispute the check-in as invalid.

- **Claim earned rewards for a verified streak:** If their own habit streak is successfully maintained and verified, users can claim their accumulated SOL rewards.
- **Claim earned incentives for successful verifications:** Users receive small SOL incentives for successfully casting votes that contribute to a verified check-in outcome.
- **Unstake their SOL from a completed habit:** Once a habit's duration is complete and successful, users can retrieve their initial staked SOL.

Admin

- **Set the minimum time between check-ins:** Defines the shortest permissible interval between a user's consecutive check-ins for the same habit (e.g., 23 hours).
- **Set the maximum gap between check-ins:** Defines the longest permissible interval between a user's check-ins to maintain a continuous streak (e.g., 25 hours).
- **Set the slash recipient wallet address:** Designates the public key where all slashed funds will be sent.
- **Set the default stake amount for new habits:** Establishes the standard SOL amount users must stake when creating a new habit.
- **Set the reward rate per successful streak day:** Determines how much SOL a user accrues in rewards for each day their streak is successfully maintained and verified.
- **Set the incentive rate per successful verification:** Determines the small SOL amount paid to an Active User for each correctly cast verification vote.

System-Driven Outcomes (Critical for protocol functionality)

- **System updates an Active User's eligibility for verification assignments:** When an Active User creates a habit or fulfills/fails verification duties, their status in the "verifier pool" is updated, influencing future assignment selections.
- **System determines a check-in's outcome based on peer consensus:** Once the validation period for a pending check-in ends, the system counts the "Verify" and "Reject" votes from the assigned three peers to determine if the check-in is approved or rejected (e.g., 2 out of 3 votes for consensus).
- **System updates streak, accrues rewards, and handles stake based on check-in outcome:** If a user's check-in is verified, their streak continues, and rewards accrue. If rejected, their streak breaks, and a portion of their staked SOL is slashed and sent to the Slash Recipient.
- **System penalizes Active Users for not completing assigned verifications:** If an Active User fails to cast a vote for an assigned check-in within the validation period, a small penalty (e.g., a temporary reduction in eligibility or reputation score) is applied.

Potential On-Chain Requirements

Active User (Combines Habit Committer & Peer Verifier roles)

- **Story AU1: User connects their Solana wallet.**
  - No direct on-chain requirement; handled by client-side wallet adapter integration for transaction signing.
- **Story AU2: User creates and stakes for a new habit in a specific category.**
  - Habit **PDA Account:**

        - ➢ Derived from creator_address (user's wallet) and a unique seed (e.g., habit_name + uuid).
        - ➢ Fields: creator_address (Pubkey), stake_amount (u64), habit_name (String), description (String), category (u8 enum), duration_days (u16), status (u8 enum: "active", "completed", "failed"), current_streak_length (u16), rewards_accrued (u64), last_check_in_timestamp (i64).
    - **Solana Program Function:** A public function to create the Habit PDA, transfer stake_amount SOL from the user's wallet to the Habit PDA (or a central Treasury PDA), and initialize its fields.
- **Story AU3: User uploads photo proof of habit completion to Arweave.**
    - No direct on-chain requirement for photo storage. The Arweave transaction ID (String) will be passed as an argument to the check-in function.
- **Story AU4: User submits a daily check-in for their habit.**
    - **CheckIn PDA Account:**
        - ➢ Derived from habit_id (parent Habit PDA) and a unique seed (e.g.,uuid).
        - ➢ Fields: habit_id (Pubkey), arweave_tx_id (String), timestamp_submitted (i64), status (u8 enum: "pending", "verified", "rejected"), validation_deadline (i64), assigned_verifiers ([Pubkey; 3]), votes_for (u8), votes_against (u8).
    - **Solana Program Function:** A public function to create the CheckIn PDA.
    - **Validation Logic:** Check min_check_in_time_seconds and max_check_in_gap_seconds against last_check_in_timestamp on the Habit PDA.
    - **Verifier Assignment Logic:** Select 3 random Pubkeys from ActiveUserRegistry entries that match the Habit's category and have sufficient verifier_reputation_score. Store these in assigned_verifiers. Requires an on-chain pseudo-random number generator (e.g., using Clock::current_slot().slot or a VRF oracle for robust production).
    - Update last_check_in_timestamp on the Habit PDA.
- **Story AU5: User views their own current streak progress and pending verification tasks.**
    - **Solana Program Functions:**
        - ➢ Function to fetch Habit PDAs by creator_address.
        - ➢ Function to fetch CheckIn PDAs where signer is in assigned_verifiers and status is "pending".
- **Story AU6: User views photo proof for an assigned check-in.**
    - **Solana Program Function:** Function to retrieve arweave_tx_id from the specific CheckIn PDA. (Client-side fetches from Arweave).
- **Story AU7: User casts a "Verify" vote on an assigned check-in.**
    - **Solana Program Function:** A public function to:
        - ➢ Atomically increment votes_for on the CheckIn PDA.
        - ➢ Validate signer is an assigned_verifier, has not already voted (voted_by check), and validation_deadline is not passed.
        - ➢ Add signer's Pubkey to voted_by array on the CheckIn PDA.
- **Story AU8: User casts a "Reject" vote on an assigned check-in.**
    - **Solana Program Function:** A public function identical to AU7, but atomically increments votes_against.
- **Story AU9: User claims earned rewards for a verified streak.**
    - **Solana Program Function:** A public function to:

        - ➢ Check Habit PDA status (e.g., "completed").
        - ➢ Transfer rewards_accrued SOL from the protocol's RewardsPool PDA to the user's wallet via Cross-Program Invocation (CPI).
        - ➢ Reset rewards_accrued on the Habit PDA to zero.
- **Story AU10: User claims earned incentives for successful verifications.**
    - **Solana Program Function:** A public function to:
        - ➢ Transfer incentives_accrued SOL from the RewardsPool PDA to the user's wallet via CPI.
        - ➢ Reset incentives_accrued on the ActiveUserRegistry PDA entry for that user to zero.
- **Story AU11: User unstakes their SOL from a completed habit.**
    - **Solana Program Function:** A public function to:
        - ➢ Check Habit PDA status ("completed", "failed").
        - ➢ Transfer stake_amount SOL from the Habit PDA (or Treasury PDA) back to the user's wallet via CPI.
        - ➢ Update Habit PDA status to "unstaked."

Admin

- **Config PDA Account:** A single PDA to store all global protocol parameters.
- **Solana Program Functions:** Public functions callable only by the designated admin_address (owner of the Config PDA) to update:
    - **Story A1:** min_check_in_time_seconds **(**u64**)**
    - **Story A2:** max_check_in_gap_seconds **(**u64**)**
    - **Story A3:** slash_recipient_address **(**Pubkey**)**
    - **Story A4:** default_stake_amount_lamports **(**u64**)**
    - **Story A5:** streak_reward_rate_per_day_lamports **(**u64**)**
    - **Story A6:** verifier_incentive_per_vote_lamports **(**u64**)**

System-Driven Outcomes (Critical for protocol functionality)

- **Story S1: System updates an Active User's eligibility for verification assignments.**
    - **Solana Program Logic:** Functions to update verifier_reputation_score, eligibility_status (e.g., "active", "suspended"), or pending_verification_count on the ActiveUserRegistry PDA entry of an Active User, triggered by their actions (habit creation, voting, or failing to vote).
- **Story S2: System determines a check-in's outcome based on peer consensus.**
    - **Solana Program Function:** A function (likely invoked by an off-chain keeper service, or implicitly by the final vote being cast) to:
        - ➢ Check if validation_deadline on CheckIn PDA has passed.
        - ➢ Compare votes_for and votes_against.
        - ➢ Update CheckIn PDA status to "verified" or "rejected".
        - ➢ If "verified", calculate and add verifier_incentive_per_vote_lamports to the incentives_accrued field of each of the 3 assigned_verifiers in their respective ActiveUserRegistry PDA entries.
- **Story S3: System updates streak, accrues rewards, and handles stake based on check-in outcome.**

    - **Solana Program Logic:** Within the determine_check_in_outcome function or a separate finalize_check_in function:
        - ➢ If CheckIn status is "verified": Increment current_streak_length on the Habit PDA. Calculate rewards using streak_reward_rate_per_day_lamports and add to rewards_accrued on the Habit PDA.
        - ➢ If CheckIn status is "rejected": Reset current_streak_length to 0 on the Habit PDA. Calculate slash_amount (e.g., a percentage of initial stake_amount). Transfer slash_amount SOL from the Habit PDA (or Treasury PDA) to the slash_recipient_address via CPI.
- **Story S4: System penalizes Active Users for not completing assigned verifications.**
    - **Solana Program Logic:** Within the determine_check_in_outcome function or a dedicated penalize_inactive_verifiers function:
        - ➢ For any assigned_verifier who did **not** vote (voted_by does not contain their Pubkey) before the validation_deadline, decrement their verifier_reputation_score or set a temporary inactivity_penalty_flag on their ActiveUserRegistry entry. This would impact their eligibility for future assignments.

Part B: Process Appendix

Initial Idea Overview

StreakSafe is a decentralized habit accountability protocol where users stake SOL to commit to daily goals. Task completion is verified through photo uploads stored on Arweave, and validated by a group of peer users in the same task category. If a check-in is verified, the user's streak continues and they become eligible for token rewards. If the check-in is not validated by the consensus, it does not count and their staked SOL may be partially slashed.

Part A: Initial User & Function Mapping Process

1) Manual User Brainstorming

Manual Task: Based on the refined Value Proposition from Assignment 1, I brainstormed a broad list of every potential user type who might interact with the project. This included Direct Users, Indirect Users/Beneficiaries, Administrators/Moderators, and Stakeholders.

Output:

- **Direct Users:** Habit Committers, Peer Verifiers, Admin
- **Indirect Users/Beneficiaries:** Habit Supporter, Community Members, Data Analysts, Developers
- **Administrators/Moderators:** Protocol Administrator, Community Moderators
- **Stakeholders:** Token Holders, Partners, Arweave Storage Providers, Solana Network Validators, Slash Recipient

2) AI-Assisted User Prioritization

AI Prompt:

"My project's value proposition is StreakSafe, a decentralized habit accountability protocol where users stake SOL to commit to daily goals. Task completion is proven through photo uploads stored on Arweave, which are then verified by a consensus of peer validators. If a check-in is verified, the user's streak continues and they become eligible for rewards. If the check-in is rejected, it does not count toward their streak and part of their staked amount may be slashed. Here is a brainstormed list of all potential user types: Habit Committers, Peer Verifiers, Admin, Habit Supporter, Community Members, Data Analysts, Developers, Protocol Administrator, Community Moderators, Token Holders, Partners, Arweave Storage Providers, Solana Network Validators, Slash Recipient. 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 (Initial):

1. Habit Committers
2. Peer Verifiers
3. Admin
4. Slash Recipient

Manual Action & Output (Initial Prioritization):

I largely agreed with the AI's initial recommendations for core POC users. The interaction between Habit Committers and Peer Verifiers is fundamental, and Admin is crucial for setup. Slash Recipient represents a key economic outcome.

Subsequent Clarification & Refinement:

After receiving the AI's output, I realized a critical detail from the project description: "any user who is checking in for the day has to verify check in of other 3 users." This means the Habit Committer and Peer Verifier roles are performed by the same underlying user.

Refined Prioritized User Types:

1. **Active User:** This single persona now combines the Habit Committer and Peer Verifier roles. This is critical because it highlights the core loop where participation in one function (checking in) directly enables and requires participation in another (verifying).
2. **Admin:** Remains crucial for initial setup and configurable parameters.
3. **Slash Recipient:** Represents the destination for slashed funds, validating the penalty mechanism.

This refinement reduces complexity for the POC by focusing on the core interaction pattern of a single active user type.

3) Core Function Mapping

AI Prompt (Initial - based on separate roles):

"For a project with this value proposition StreakSafe... and focusing on these prioritized user types Habit Committer, Peer Verifier, Admin, Slash Recipient, help map out the key functions or interactions each user would need to perform."

AI Output (Initial Functions for separate roles):

- **Habit Committer:** Stake SOL, Define Habit Goal, Upload Photo Proof, Initiate Check-in Validation, Claim Rewards, Unstake SOL.
- **Peer Verifier:** Discover Check-ins for Validation, View Photo Proof (from Arweave), Verify Check-in, Reject Check-in, Receive Validation Incentives.
- **Admin:** Set Minimum/Maximum Time Between Check-ins, Set Slash Recipient Address, Set Default Stake Amount, Set Reward Rate.
- **Slash Recipient:** Receive Slashed Funds.

Manual Refinement & AI Re-Prompt for Combined Role:

Recognizing the combined "Active User" role, I re-prompted the AI to consolidate functions and ensure the dual responsibility was clear.

AI Prompt (Revised - for combined "Active User"):

"For a project with this value proposition StreakSafe... and focusing on these prioritized user types Active User (Habit Committer + Peer Verifier), Admin, Slash Recipient, help map out the key functions or interactions each user would need to perform. Emphasize that the Active User takes on both roles."

AI Output (Revised Functions for combined "Active User"):

- **Active User:** Stake SOL for a Habit, Define Habit Goal, Upload Photo Proof & Submit Check-in (triggers peer assignment), Discover Assigned Check-ins for Verification, View Photo Proof (for others), Verify Assigned Check-in, Reject Assigned Check-in, Claim Rewards (for own streak), Receive Validation Incentives (for verifying), Unstake SOL (from own habit).
- **Admin:** Set Minimum/Maximum Time Between Check-ins, Set Slash Recipient Address, Set Default Stake Amount, Set Reward Rate (for both streaks and verifications).
- **Slash Recipient:** Receive Slashed Funds.

Analysis & Rationale: This revised mapping is much more accurate. It correctly integrates the verification duties into the "Active User" workflow, ensuring the core mechanism of mandatory peer verification is captured. The functions are logical and cover the essential interactions for the POC.

4) Deriving Core POC Requirements

Manual Task (Identifying Critical User Interactions):

From the revised function map, the most critical interactions for proving StreakSafe's core value are:

1. **Active User: Staking SOL, defining a habit, uploading proof, and submitting their check-in (which implicitly assigns them peer verification tasks).** This shows the commitment and initiation of the process.
2. **Active User: Viewing assigned proofs and casting votes (verify/reject) for others.** This demonstrates the decentralized, peer-driven verification.

AI Prompt (Initial - for separate roles):

"Based on these top 1-2 critical user interactions [described interactions], what are the key technical requirements needed to build a proof-of-concept?"

AI Output (Initial Technical Requirements):

- Smart Contract for Habit Management (deposit, create habit, record check-in, track streak, trigger validation)
- Integration with Arweave (upload, retrieve IDs)
- UI for Habit Committer (wallet connect, input fields, photo upload, submit check-in, streak display)
- Smart Contract for Validation (discover pending, cast vote, consensus logic, distribute rewards, slash stake)
- UI for Peer Verifier (list pending, view photo, verify/reject buttons)
- Smart Contract for Configuration (Admin functions)
- Basic Solana Smart Contract Development (program deployment, SOL transfers, data storage)
- Basic Security Considerations

Manual Refinement & AI Re-Prompt for Combined Role:

With the "Active User" concept solid, I refined the critical interactions and re-prompted the AI to get more specific technical requirements for this combined role.

AI Prompt (Revised - for combined "Active User"):

"Based on these top 1-2 critical user interactions: 1) Active User stakes SOL, defines a habit, uploads photo proof (to Arweave), and submits their check-in, simultaneously triggering the assignment of 3 peer verification tasks to them. 2) Active User views assigned photo proofs and casts votes (verify/reject) for the assigned check-ins. What are the key technical requirements needed to build a proof-of-concept?"

AI Output (Revised Technical Requirements for combined "Active User"):

- **Smart Contract for User/Habit/Check-in Management:** (Includes select 3 random...verifiers logic within check-in submission)
- **Smart Contract for Validation Logic:** (Includes Active User (acting as Peer Verifier) specifically)
- **Integration with Arweave**
- **User Interface (UI) for Active User:** (Includes sections for *both* own habits *and* assigned verifications)

- **General/Admin Technical Requirements for POC:** (Includes **Randomness for Peer Assignment** explicitly)

Analysis & Rationale: The revised technical requirements are much more granular and directly address the complexities introduced by the combined "Active User" role, especially the on-chain random assignment of verifiers. This sets a strong foundation for smart contract design.

Part B: Adversarial Analysis & Granularity Check Process

1) Critique & Refine User Stories / Requirements

AI Prompt:

"Review my core user functions/stories [Input from Part A, Step 3, AI Output: Active User, Admin, Slash Recipient functions] and requirements [Input from Part A, Step 4, AI Output: Technical Requirements]. Considering my project's refined value proposition StreakSafe... 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 (Critique):

- **User Functions/Stories:** Generally hit the mark, but suggested:
    - Explicitly mention Arweave TX ID storage on-chain.
    - Consider a "System determines outcome" story.
    - Missing user registration/onboarding.
    - Missing dispute resolution/malicious voting handling (possibly out of POC scope).
- **Technical Requirements:** Good high-level, but suggested more granularity for:
    - Specific Solana Program Accounts (PDAs), Instruction Data, State Management.
    - Explicitly mention the mechanism for selecting 3 peer verifiers (critical missing piece).
    - More specific reward/slashing calculation logic.
    - Error handling, Off-chain vs. On-chain distinction.

My Analysis:

The AI's critique was insightful and highly valuable. It confirmed that the user stories accurately reflected the core value but identified critical areas for increased granularity, particularly regarding the mechanism of selecting peer verifiers and the explicit on-chain data structures. The suggestion to include "System-Driven Outcomes" was particularly useful for clarity. The point about user registration/onboarding, while seemingly minor, is essential for a complete user flow even in a POC. The "same category" detail was identified as a subtle but important missing aspect in the technical requirements.

Rationale for Refined User Stories and Technical Requirements:

The refinement process directly addressed these critiques:

- **Focus on Core Loop:** Ensure every story directly contributes to proving the staking, photo-proof, peer-verification, and reward/slashing aspects.
- **Atomicity & Clarity:** Break down composite actions into single, clear steps (e.g., distinguishing between uploading proof and submitting a check-in).
- **On-chain/Off-chain Distinction:** Explicitly mention what data flows where (e.g., Arweave IDs on-chain).
- **Explicit Peer Selection:** Integrated the "user verifies 3 others" logic into both user stories and technical requirements, emphasizing the need for on-chain random assignment and category matching.
- **Basic Onboarding:** Added a minimal onboarding story (connecting wallet and creating a habit now marks user eligible for verification).
- **State Management Detail:** Began outlining specific PDAs and their fields (Habit PDA, CheckIn PDA, ActiveUserRegistry PDA, Config PDA) to prepare for smart contract design.
- **System-Driven Outcomes:** Added stories for system-level actions like determining consensus and applying penalties.
- **Verifier Accountability:** Included a basic concept of penalty for inactive verifiers (e.g., reputation score reduction).

Part C: Granularity & Clarity Refinement Process

1) Final Manual Review & Refinement

Action: I took the user stories and functions from the refined Part B and meticulously reviewed them against the criteria: **De-Jargon, Granularity, Atomicity, Clarity of Action, and No Overlap.** The goal was to ensure maximum clarity and readiness for technical implementation.

Part C Refinement Log:

- **Before:** User connects their Solana wallet.
  - **After: Story AU1: User connects their Solana wallet.** (No change, already atomic and clear)
- **Before:** Active User: Stake SOL for a Habit & Define Habit Goal.
  - **After: Story AU2: User creates and stakes for a new habit in a specific category.**
  - **Rationale:** Combined staking and habit definition for atomicity of initial setup. Added "in a specific category" to align with future peer verification logic.
- **Before:** Active User: Upload Photo Proof & Submit Check-in.
  - **After: Story AU3: User uploads photo proof of habit completion to Arweave.** AND **Story AU4: User submits a daily check-in for their habit.**
  - **Rationale:** Split for atomicity. Photo upload is an off-chain action that precedes the on-chain check-in submission. Story AU4 explicitly clarifies that *submitting their check-in* is the trigger for peer assignment for *their* check-in, and for *them* being assigned peer tasks.
- **Before:** Active User: Discover Assigned Check-ins for Verification.

    - **After: Story AU5: User views their own current streak progress and pending verification tasks.**
    - **Rationale:** Combined viewing their own status with seeing assigned tasks for a more realistic user flow on a dashboard.
- **Before:** Active User: View Photo Proof (from Arweave) for Others.
    - **After: Story AU6: User views photo proof for an assigned check-in.** (Minor rephrase for clarity)
- **Before:** Active User: Verify Assigned Check-in.
    - **After: Story AU7: User casts a "Verify" vote on an assigned check-in.**
    - **Rationale:** "Casts a vote" is more atomic and clearer than "verifies," which could imply the outcome.
- **Before:** Active User: Reject Assigned Check-in.
    - **After: Story AU8: User casts a "Reject" vote on an assigned check-in.** (Similar rationale to AU7)
- **Before:** Active User: Claim Rewards (for own streak).
    - **After: Story AU9: User claims earned rewards for a verified streak.** (Minor rephrase for clarity)
- **Before:** Active User: Receive Validation Incentives (for verifying others).
    - **After: Story AU10: User claims earned incentives for successful verifications.**
    - **Rationale:** Rephrased to emphasize active "claiming" and distinguish it from passive "receiving," making it an atomic user action.
- **Before:** Active User: Unstake SOL (from own habit).
    - **After: Story AU11: User unstakes their SOL from a completed habit.** (Minor rephrase for clarity)
- **Before:** Admin: Set Reward Rate.
    - **After: Story A5: Admin sets the reward rate per successful streak day.** AND **Story A6: Admin sets the incentive rate per successful verification.**
    - **Rationale:** Split for granularity. These are two distinct rates for different types of rewards, even if set by the same admin.
- **Before (Implicit):** System handles verifier activity/inactivity.
    - **After: Story S1: System updates an Active User's eligibility for verification assignments.** AND **Story S4: System penalizes Active Users for not completing assigned verifications.**
    - **Rationale:** Made explicit system behaviors that manage the verifier pool and introduce accountability for verifiers, as identified in the AI critique.
- **Before (Implicit):** System handles check-in outcome.
    - **After: Story S2: System determines a check-in's outcome based on peer consensus.** AND **Story S3: System updates streak, accrues rewards, and handles stake based on check-in outcome.**
    - **Rationale:** Split for clarity. One story focuses on the consensus determination, the other on the subsequent state changes and economic consequences.

Part D: Defining Potential On-Chain Requirements Process

Action: I took the final, refined list of user stories from Part C. For each individual story, I brainstormed and detailed a bulleted list of potential on-chain requirements, focusing on Solana Program Derived Addresses (PDAs), program functions, data fields, and critical logic (e.g., randomness, validation).

Process: This was an iterative mental exercise, translating each user action into its necessary blockchain components. I considered:

- What data needs to be stored persistently on-chain?
- How would that data be structured (e.g., in a PDA)?
- What functions would the smart contract need to expose?
- What inputs would those functions take?
- What validations or checks would be necessary (e.g., time, permissions)?
- How would SOL transfers occur (e.g., CPIs)?
- How would the core "assign 3 verifiers" logic work?

Refinements during this stage (captured in the final Part A output):

- **Explicit PDA Naming:** Clearly defined Habit PDA, CheckIn PDA, ActiveUserRegistry PDA, and Config PDA accounts.
- **Detailed PDA Fields:** Outlined specific fields for each PDA, including their data types (e.g., Pubkey, u64, String, i64 for timestamps).
- **Randomness Source:** Explicitly noted the need for on-chain randomness and suggested Clock::current_slot() as a POC option, acknowledging the need for VRF in production.
- **Category Matching:** Included logic to filter verifiers by category from the ActiveUserRegistry.
- **Verifier Accountability Mechanism:** Added basic requirements for tracking verifier_reputation_score and applying minor penalties for non-voting.
- **CPIs:** Explicitly mentioned Cross-Program Invocations for SOL transfers between accounts.