Top 3 Users Per Category

Direct Users (Day-to-Day)

Challenge Creators:

1. Fitness enthusiasts creating gym attendance goals
2. Students creating study habit challenges
3. Creators/freelancers creating project completion goals

Verifiers:

1. Close friends designated as verifiers
2. Accountability partners (mutual verification relationships)
3. Gym buddies verifying each other

Indirect Users/Beneficiaries

Personal Beneficiaries:

1. Romantic partners benefiting from healthier/more productive significant others
2. Gym buddies who gain consistent workout partners
3. Study group members benefiting from more prepared participants

Ecosystem Beneficiaries:

1. Behavioral psychology researchers gaining data on commitment devices
2. Content creators making videos about the platform (viral marketing)
3. Corporate wellness programs seeking accountability tools

Administrators/Moderators

Platform Management:

1. Founder/developer (me) - technical development and strategic decisions
2. Content moderators reviewing flagged photos for inappropriate content
3. Community managers engaging users on Discord/Twitter

Community Governance (Future):

1. DAO members voting on platform parameters (fee %, bonus %, challenge limits)
2. Reputation committee handling appeals and edge cases
3. Community moderators elected/appointed by users

Stakeholders

Economic Stakeholders:

1. Early investors/angels funding development
2. Solana Foundation (interested in ecosystem growth)
3. Platform token holders (if tokenized in future)

Strategic Partners:

1. CrossFit gyms and fitness communities (tight-knit launch partners)
2. University student organizations (distribution channel)
3. Creator platforms (YouTube, Substack) integrating accountability

Regulatory/Compliance Stakeholders:

1. Legal advisors ensuring compliance with gambling laws
2. Privacy advocates ensuring photo data handling complies with laws
3. Financial regulators monitoring crypto-based betting

Edge Case Users

Adversarial Users:

1. Fraudsters trying to game the system with fake photos
2. Collusion rings (friends always approving each other)
3. Trolls creating inappropriate challenges

Peripheral Users:

1. Social media influencers showcasing their challenges
2. Journalists writing about the platform
3. Competitors analyzing the model

Critical User Types for Proof-of-Concept (Ranked by Priority)

1. Challenge Creators (Fitness Enthusiasts)

Rationale: This is the PRIMARY user proving the core value proposition. Need to validate that people will actually stake real money on personal goals when consequences are financial. Fitness enthusiasts are the beachhead because goals are visual (gym selfies), verifiable, and this demographic overlaps with crypto adoption. Without proving people will put "skin in the game," the entire concept fails. Success metric: 60% actually create and fund a challenge with real SOL.

2. Verifiers (Close Friends)

Rationale: The second half of the core mechanic. Need to prove friends will consistently verify evidence AND that the profit motive (earning from failures) sustains engagement beyond initial novelty. If verifiers ghost after week 1 or approve everything without looking, the accountability layer collapses. This validates whether economic incentives actually align verifier behavior. Success metric: 70% verifier participation rate across all challenges.

3. Content Moderators (Founder/Developer initially)

Rationale: Operational necessity to prevent immediate disaster. Even in small PoC (20 users), inappropriate photos, gaming attempts, or edge cases will emerge. Need active moderation to maintain trust and handle disputes. For PoC, founder can play this role, but proves the moderation workflow is manageable. Without this, one bad incident kills user trust. Success metric: <24hr response time on flagged content.

4. Repeat Challenge Creators

Rationale: Proves the platform creates behavior change, not just one-time novelty. If users complete one challenge but never return, the value proposition (habit formation through accountability) is unproven. Repeat usage validates that outcomes affect behavior and the experience was positive enough to try again. This is the clearest signal of product-market fit. Success metric: 40% of successful users create a second challenge.

Decision Making Criteria - Any user type not directly involved in the stake → verify → outcome loop is post-PoC. PoC validates the core mechanic works; everything else (partnerships, governance, additional verticals) comes after proving that foundation. Hence I focused only on users who touch the money flow (stake/slash/distribute) or who are required to keep the system functional during testing.

Core Function Mapping

1. Challenge Creators (Fitness Enthusiasts)

Onboarding:

- Connect wallet
- Receive small test SOL for first transaction

Challenge Creation:

- Define goal (e.g., "Gym 3x this week")
- Set stake amount (5-50 SOL)
- Set duration (1-4 weeks)
- Designate 2-5 verifiers from contacts
- Confirm and stake (2% fee deducted)

During Challenge:

- Upload photo evidence (IPFS via app)
- Track verification status
- View progress dashboard

Post-Challenge:

- Receive payout (success) or see slash distribution (failure)
- View updated reputation score
- Get prompted to create second challenge

---

2. Verifiers (Close Friends)

Onboarding:

- Receive notification that friend added them as verifier
- Connect wallet to receive potential earnings
- Review verification guidelines

During Challenge:

- Receive push notification when evidence submitted
- View photo evidence
- One-tap approve/reject
- Optional: add comment/feedback

Post-Challenge:

- Receive earnings from failed challenges
- View verification history/accuracy
- See reputation as verifier

---

3. Content Moderators (Founder/Developer)

Monitoring:

- View flagged content dashboard
- Review reports of inappropriate photos
- Monitor verification disputes

Actions:

- Approve/remove flagged evidence
- Manually resolve disputes when needed
- Ban users for ToS violations
- Adjust challenge parameters if gaming detected

Analytics:

- Track completion rates
- Monitor verifier participation
- Identify fraud patterns

---

4. Repeat Challenge Creators

Post-First-Challenge:

- View performance summary (completed/failed)
- Compare to previous attempts
- See verifier feedback

Second Challenge Creation:

- Quick-create from previous template
- Adjust parameters based on learnings
- Keep same verifiers or select new ones
- Stake again

Long-term:

- View challenge history
- Track improvement over time
- Build on-chain reputation

Key Technical Requirements for PoC

1. Blockchain/Smart Contract Layer

Solana Program (Anchor):

- Challenge account structure (PDA for each challenge)
- User account structure (reputation, history)
- Verification account structure (each verifier's decision)
- Escrow account (holds staked SOL)
- Platform fee treasury account

Core Instructions:

- `create_challenge` - initialize challenge, lock stake, deduct 2% fee
- `submit_evidence` - store IPFS hash on-chain
- `submit_verification` - verifier approves/rejects
- `finalize_challenge` - check consensus, distribute funds (stake + 1% bonus OR 5% slash to verifiers)
- `initialize_user` - create user account with reputation tracking

On-Chain Logic:

- Multi-sig verification (require X of Y verifiers)
- Time-based state transitions (challenge active → verifying → completed)
- Conditional distribution (success vs failure paths)
- Fee calculation and treasury management

---

2. Frontend/User Interface

Web Application (Next.js):

- Landing page explaining concept
- Wallet connection
- Challenge creation form
- Evidence upload interface
- Verifier dashboard (pending verifications)
- Challenge detail pages
- User profile/reputation page

Key Features:

- Photo upload to IPFS (via Pinata)
- Real-time challenge status updates
- Push notifications for verifiers (browser notifications)
- Mobile-responsive design

State Management:

- React Query for blockchain data fetching
- Wallet adapter state
- User session management

---

3. Storage Layer

Photo Evidence:

- IPFS integration (Pinata)
- Upload photos from device
- Store IPFS hash on-chain
- Retrieve images via gateway for verification

Metadata Storage:

- Challenge descriptions
- User profiles (username, bio)
- Evidence timestamps and GPS data (optional)

---

4. Infrastructure/DevOps

Blockchain:

- Solana devnet for PoC
- RPC endpoint (Helius or QuickNode)
- Transaction monitoring

Backend (Optional for PoC):

- Notification service for push alerts
- Cron jobs for checking expired challenges
- Analytics tracking (Mixpanel/PostHog)

Hosting:

- Vercel for frontend
- Simple backend on Railway if needed

---

5. Third-Party Integrations

Essential:

- **Wallet Abstraction:** Magic.link or Privy (gasless onboarding) or use sponsor account
- **IPFS Provider:** Pinata or NFT.storage (photo storage)
- **RPC Provider:** Helius or QuickNode (Solana access)

Nice-to-Have for PoC:

- Push notification service (OneSignal)
- Analytics (Mixpanel)
- Error tracking (Sentry)

---

6. Development Tools

Smart Contract:

- Anchor framework
- Rust toolchain
- Solana CLI
- Local validator for testing

Frontend:

- TypeScript
- Next.js 14
- Tailwind CSS
- @solana/web3.js
- @solana/wallet-adapter

Testing:

- Anchor tests for smart contracts
- Manual E2E testing with 3-5 users

Can Skip for PoC:

- Complex reputation algorithms (use simple counter)
- Automated notifications (manual for first users)
- Advanced fraud detection (handle manually)
- Mobile app (web responsive is enough)
- DAO governance
- Tokenomics beyond basic fee structure

Critical Gaps

Missing User Functions

Verifiers:

- No "accept/decline verifier role" flow
- No visibility into potential earnings before verifying
- No dispute mechanism when they disagree with creator

Creators:

- No cancel/abandon challenge flow
- No dispute unfair rejections
- No social sharing for viral growth

Edge Cases:

- What if creator uploads nothing?
- What if <50% verifiers vote?
- What if evidence uploaded after deadline?

Requirements Too Vague

Need Exact Specs:

❌ "Multi-sig verification" → ✅ "Require (verifier_count + 1) / 2 approvals"

❌ "Conditional distribution" → ✅ "Success: escrow - 2% fee + 1% bonus. Failure: 5% / num_verifiers"

❌ "Time-based transitions" → ✅ "Cron checks hourly, auto-fails if no evidence by deadline + 24h"

❌ "Gasless transactions" → ✅ "Using Helius/Octane? Who pays? Rate limits?"

❌ "Push notifications" → ✅ "OneSignal? Web Push API? Trigger timing?"

Biggest Misalignment

Value prop says: "Verifiers profit from failures creates engagement"

Reality: No features showing verifiers:

- Their earnings potential
- Total earned to date
- Incentive to recruit more creators

Missing: Verifier earnings dashboard

Cannot Map to Code

- Which sponsor service for gasless tx?
- Where does cron job run for auto-finalization?
- What if the IPFS gateway fails?
- Where is the notification queue stored?

Must Add

1. **Verifier earnings visibility** (core value prop)
2. **Dispute flow** (manual for PoC)
3. **Abandon challenge handling**
4. **Exact distribution formulas**
5. **Service specifications** (Helius tier? OneSignal? Pinata limits?)

Part B

User Stories & Technical Requirements Refinement

AI Critique

Critical Gaps:

- Verifiers missing earnings visibility - can't see potential earnings before participating, no dashboard showing total earned
- No dispute resolution mechanism for unfair verifications
- Edge cases unaddressed: abandoned challenges, low verifier participation, late evidence
- Technical specs too vague: "multi-sig verification" without exact threshold formula, "conditional distribution" without calculations, "gasless transactions" without service specification
- Value prop claims viral growth but no social sharing features listed

Misalignment with Value Prop: Claims "verifiers profit from failures creates engagement" but no features showing verifiers their economics, leaderboards, or recruitment incentives.

My Analysis

Valid concerns: Verifier earnings visibility is core to value prop but completely missing. Dispute mechanism needed for trust. Edge cases will occur in PoC with 20 users. Technical specs too vague for implementation.

Prioritization: Must have for PoC: verifier earnings dashboard, basic dispute flow, edge case handling. Can defer: advanced social features, DAO governance.

Refined User Stories - Added

Challenge Creators:

- Dispute verification: "As creator, I can dispute unfair rejection so moderator reviews"
- Cancel challenge: "As creator, I can cancel early with 20% penalty"

Verifiers:

- Accept/decline invitation: "As verifier, I can accept or decline role"
- Earnings visibility: "As verifier, I see potential earnings (if challenges fail, I earn X SOL)"
- Earnings dashboard: "As verifier, I view total earned, active challenges, conversion rate"

Edge Cases:

- No evidence by deadline + 24h = auto-fail
- <50% verifier participation = use existing votes only
- Late evidence = rejected automatically

Refined Technical Requirements

Smart Contract - Exact Specs:

Distribution formulas:

- Success: `escrow_balance + (stake_amount * 0.01)` from bonus_pool
- Failure: `(stake_amount * 0.05) / active_verifier_count`

Required approvals: `(verifier_count + 1) / 2`

State transitions:

- Active → Verifying: when all evidence submitted
- Verifying → Completed: when approvals >= threshold
- Active → Failed: when time > deadline + 24h with no evidence

Infrastructure - Service Specs:

Push notifications:

- Service: OneSignal
- Triggers: evidence submitted, challenge expiring, verification received

IPFS:

- Service: Pinata
- 3 fallback gateways

Auto-finalization:

- Clockwork checks every 6h
- Calls finalize_challenge for expired challenges

Rationale

Verifier earnings visibility: Without seeing potential earnings upfront, verifiers have no motivation beyond curiosity. Makes economics transparent (core to value prop).

Exact formulas: `(stake_amount * 0.05) / active_verifier_count` prevents implementation ambiguity and bugs.

Service specifications: Helius sponsored tx with rate limits enables cost estimation and debugging. Can't implement "gasless transactions" without knowing which service.

Edge case handling: PoC will hit these (forgotten verifications, abandoned challenges). Defined behavior prevents manual resolution overhead.

User Functions - What Needs Fixing

1. Challenge Creators

The Challenge Creation section is good - already atomic steps listed separately.

During Challenge:

- Upload photo evidence (IPFS via app) → **Upload photo as evidence**
- Track verification status ✓
- View progress dashboard ✓

Post-Challenge:

- Receive payout (success) or see slash distribution (failure) → Split into:
  - **Get stake + bonus back (if successful)**
  - **See stake distribution to verifiers (if failed)**
- View updated reputation score ✓
- Get prompted to create second challenge → **See prompt to create second challenge**

---

2. Verifiers

Onboarding:

- Receive notification that friend added them as verifier → **See notification that friend added them as verifier**
- Connect wallet to receive potential earnings ✓
- Review verification guidelines ✓

During Challenge:

- Receive push notification when evidence submitted → **See notification when friend submits evidence**
- View photo evidence ✓
- One-tap approve/reject → **Approve or reject evidence**
- Optional: add comment/feedback ✓

Post-Challenge:

- Receive earnings from failed challenges → **Get paid when friend fails challenge**
- View verification history/accuracy ✓
- See reputation as verifier ✓

---

3. Content Moderators

Actions:

- Approve/remove flagged evidence ✓
- Manually resolve disputes when needed ✓
- Ban users for ToS violations ✓
- Adjust challenge parameters if gaming detected → Split into:
  - **Spot suspicious patterns**
  - **Adjust parameters to prevent abuse**

Analytics:

- Track completion rates ✓
- Monitor verifier participation ✓
- Identify fraud patterns ✓

---

4. Repeat Challenge Creators

Second Challenge Creation:

- Quick-create from previous template ✓
- Adjust parameters based on learnings ✓
- Keep same verifiers or select new ones → **REMOVE** (redundant with "Adjust parameters")
- Stake again ✓

Summary of Changes

Removed implementation details: IPFS, push, one-tap **Fixed passive voice:** 5 instances changed to active **Split compound actions:** 2 stories split (payout/slash, detect/adjust) **Removed redundancy:** 1 story removed

Everything else stays as-is.

On-Chain Requirements for User Stories

1. Challenge Creators

Onboarding

User Story: Connect wallet

- No on-chain requirement (client-side wallet connection)

User Story: Receive small test SOL for first transaction

- Need faucet instruction or airdrop mechanism
- Check if user account exists, if new → send test SOL from treasury

---

Challenge Creation

User Story: Define goal

- Store goal text (String) in Challenge account
- Store required_proofs count (u8)

User Story: Set stake amount

- Store stake_amount (u64) in Challenge account
- Validate minimum stake (e.g., 0.1 SOL)

User Story: Set duration

- Store start_time (i64 - unix timestamp) in Challenge account
- Store end_time (i64) calculated from duration
- Store verification_deadline (i64) = end_time + 86400 seconds

User Story: Select verifiers

- Store verifiers array (Vec<Pubkey>, max 5) in Challenge account

- Store verifier_count (u8)
- Calculate required_approvals = (verifier_count + 1) / 2

User Story: Confirm and stake (2% fee deducted)

- Create Challenge PDA with seeds ["challenge", creator.key(), challenge_id.to_le_bytes()]
- Create Escrow PDA with seeds ["escrow", challenge.key()]
- Transfer stake_amount from creator to Escrow PDA
- Calculate platform_fee = stake_amount * 0.02
- Transfer platform_fee to Treasury PDA
- Store net escrow = stake_amount - platform_fee
- Initialize challenge status = Active
- Create or update User account with total_challenges++

During Challenge

User Story: Upload photo as evidence

- Store IPFS hash (String) in Evidence PDA
- Evidence PDA seeds: ["evidence", challenge.key(), evidence_index.to_le_bytes()]
- Store timestamp (i64)
- Increment evidence_count in Challenge account
- If evidence_count == required_proofs, change status to Verifying

User Story: Track verification status

- Read Challenge account: approval_count, rejection_count
- Read all Verification PDAs for this challenge
- No write operations (query only)

User Story: View progress dashboard

- Read Challenge account: evidence_count, required_proofs, end_time
- Calculate days_remaining from current_time and end_time
- No write operations (query only)

Post-Challenge

User Story: Get stake + bonus back (if successful)

- Check Challenge.status == Completed
- Check approval_count >= required_approvals
- Transfer escrow_balance from Escrow PDA to creator

- Transfer bonus (stake_amount * 0.01) from Bonus Pool PDA to creator
- Update User account: completed++, reputation_score += 10

User Story: See stake distribution to verifiers (if failed)

- Check Challenge.status == Failed
- Calculate per_verifier = (stake_amount * 0.05) / active_verifier_count
- Transfer from Escrow to each active verifier's account
- Update User account: failed++, reputation_score -= 5

User Story: View updated reputation score

- Read User account: completed, failed, reputation_score
- No write operations (query only)

User Story: See prompt to create second challenge

- Read User account: total_challenges count
- No on-chain requirement (UI logic)

---

2. Verifiers

Onboarding

User Story: See notification that friend added them as verifier

- No on-chain requirement (off-chain notification service)
- Read Challenge account to display details

User Story: Connect wallet to receive potential earnings

- No on-chain requirement (client-side wallet connection)

User Story: Review verification guidelines

- No on-chain requirement (static content)

During Challenge

User Story: See notification when friend submits evidence

- No on-chain requirement (off-chain notification triggered by Evidence PDA creation event)

User Story: View photo evidence

- Read Evidence PDA: ipfs_hash
- Retrieve photo from IPFS using hash
- No write operations

User Story: Approve or reject evidence

- Create Verification PDA with seeds ["verification", challenge.key(), verifier.key()]
- Store decision (enum: Approve/Reject) in Verification account
- Store timestamp (i64)
- Update Challenge account: approval_count++ or rejection_count++
- Check if threshold reached to change status to Completed/Failed

User Story: Optional: add comment/feedback

- Store comment (String) in Verification account
- No additional on-chain changes

Post-Challenge

User Story: Get paid when friend fails challenge

- Check Challenge.status == Failed
- Check this verifier submitted vote (Verification PDA exists)
- Transfer (stake_amount * 0.05) / active_verifier_count to verifier
- Update verifier's earnings history (could be separate account or off-chain)

User Story: View verification history/accuracy

- Query all Verification PDAs where verifier == this user
- Calculate approval rate from historical data
- No write operations (query only)

User Story: See reputation as verifier

- Read verifier-specific stats (could be separate VerifierStats account)
- Store total_verifications, accuracy_rate
- No mandatory on-chain storage (can calculate from Verification PDAs)

---

3. Content Moderators

Monitoring

User Story: View flagged content dashboard

- Query Evidence PDAs with flagged status
- Need flagged_status (bool) field in Evidence account
- No write operations (query only)

User Story: Review reports of inappropriate photos

- Read Evidence account: ipfs_hash, flag_reason
- Need flag_reason (String) field in Evidence account
- Read related Challenge and User accounts for context

User Story: Monitor verification disputes

- Query Challenge accounts with status == Disputed
- Read dispute details from Challenge or separate Dispute account

---

Actions

User Story: Approve/remove flagged evidence

- Update Evidence account: set approved/removed status
- If removed, update Challenge logic to handle missing evidence

User Story: Manually resolve disputes when needed

- Update Challenge.status from Disputed to Completed/Failed
- Override approval_count or rejection_count if needed
- Require moderator signature (check moderator_authority in program)

User Story: Ban users for ToS violations

- Update User account: banned (bool) = true
- Check banned status in all instructions, reject if banned

User Story: Spot suspicious patterns

- No on-chain requirement (analytics/off-chain monitoring)

User Story: Adjust parameters to prevent abuse

- Update program parameters (if using config account)
- Could store ConfigAccount with adjustable values: min_stake, max_stake, max_verifiers
- Require admin authority

Analytics

User Story: Track completion rates

- Query all Challenge accounts by status
- Calculate Completed / (Completed + Failed) ratio
- No write operations (analytics query)

User Story: Monitor verifier participation

- Query Verification PDAs grouped by verifier
- Calculate participation rate per verifier
- No write operations (analytics query)

User Story: Identify fraud patterns

- No on-chain requirement (off-chain pattern analysis)
- Could read evidence hashes and detect duplicates

---

4. Repeat Challenge Creators

Post-First-Challenge

User Story: View performance summary

- Read Challenge account: status, evidence_count, approval_count
- Read Verification PDAs for verifier feedback
- No write operations (query only)

User Story: Compare to previous attempts

- Query all Challenge PDAs where creator == this user
- Sort by created_at timestamp
- Calculate success rate trends
- No write operations (query only)

User Story: See verifier feedback

- Read Verification accounts: comment field
- No write operations (query only)

Second Challenge Creation

User Story: Quick-create from previous template

- Read previous Challenge account data
- Pre-fill new challenge with same values
- Create new Challenge PDA (separate from first)
- No special on-chain requirement (UI copies data)

User Story: Adjust parameters based on learnings

- Same as regular challenge creation
- Store modified values in new Challenge account

User Story: Stake again

- Same on-chain flow as first stake
- Create new Escrow PDA for new challenge
- Transfer new stake amount

Long-term

User Story: View challenge history

- Query all Challenge PDAs where creator == this user
- Sort by start_time descending
- No write operations (query only)

User Story: Track improvement over time

- Calculate success rate by month from historical Challenge PDAs
- No write operations (analytics query)

User Story: Build on-chain reputation

- User account stores: total_challenges, completed, failed, reputation_score
- Updated with each challenge completion
- Publicly readable by anyone

Summary of Key On-Chain Accounts Needed

1. **Challenge Account** - stores goal, stake, verifiers, status, timestamps, counts
2. **Escrow Account** - holds locked stake funds
3. **Evidence Account -** stores IPFS hash, timestamp, flagged status
4. **Verification Account -** stores verifier decision, comment, timestamp
5. **User Account -** stores stats, reputation, banned status
6. **Treasury Account** - collects platform fees
7. **Bonus Pool Account -** funds success bonuses
8. **Config Account** (optional) - stores adjustable platform parameters