

Synkio Safety — Grant Application Draft + Full PRD

SECTION 1 — GRANT APPLICATION DRAFT (DD.xyz / Webacy)

Project Name

Synkio Safety

Project Summary

Synkio Safety is a real-time, AI-powered risk intelligence layer embedded inside Synkio—the chat-first P2P payments and social-commerce platform. It scans wallets, tokens, URLs, contracts, transactions, and approvals to protect users from scams, drainer links, malicious addresses, and high-risk interactions.

Synkio Safety uses **DD.xyz APIs + BlockRader + Synkio's custom AI engine** to deliver the simplest safety experience for everyday users who buy, sell, and pay online.

What Problem Are You Solving?

Most crypto-based payments are still unsafe for ordinary users. People get drained, scammed, or tricked through: - Fraudulent wallet addresses - Drainer links - High-risk tokens - Suspicious contracts - Malicious approvals

Synkio solves payment simplicity; Synkio Safety solves payment trust.

How We Will Integrate DD.xyz APIs

We will integrate the following endpoints directly into the Synkio Safety engine: - **Threat Risk API** → Wallet screening before every payment. - **Approval Risk API** → Auto-scan a user's approvals and flag malicious contracts. - **Transaction Risk API** → Pre-transaction warnings. - **Exposure Risk API** → Wallet Health Score. - **Contract Risk API** → Token & contract safety badges. - **URL Risk API** → Chat link scanning inside Synkio.

Detailed Integration Examples

1. Payment Flow Safety Check

Before a user pays a seller, Synkio Safety runs: - Wallet threat analysis - Token risk analysis - Contract risk score

2. Chat Link Protection

Any link shared in Synkio chat is auto-scanned with URL Risk API.

3. Wallet Health Dashboard

Users can see: - Safety score (Low / Medium / High / Critical) - Risky approvals - Exposure risk

Technical Implementation

- Backend: Node.js (NestJS) safety microservice
- Infra: BlockRader + DD.xyz + OpenAI/Anthropic model for reasoning
- On-chain: Wallet/token scanners + Thirdweb for revokes
- UI: Synkio React front-end with Safety module

Why Our Team Can Execute

The Synkio team has shipped: - Wallet integrations - Smart contract modules - Transaction flows - AI-powered chat features

The safety module is fundamentally data ingestion + scoring + UI, all within our capabilities.

Timeline (Within 3 Months)

Month 1: API integration + risk engine core **Month 2:** UI, scoring, chat link safety, transaction warnings
Month 3: Wallet health dashboard + final optimization

How This Grant Helps

The DD.xyz credits allow us to: - Query hundreds of thousands of wallet addresses - Run continuous risk evaluation - Deliver real due-diligence intelligence inside a consumer app

Differentiation

Unlike traditional tools built for traders or developers, Synkio Safety targets: - Everyday buyers - Sellers - P2P users - Social commerce participants

We make due diligence **invisible**, automatic, and instant.

Expected Impact

- Safer crypto commerce globally
- Reduced scam exposure for users
- More trust for web3 payments

Ask

We are requesting **\$10,000 DD.xyz API credits** to power Synkio Safety's intelligence engine.

SECTION 2 — FULL PRD: Synkio Safety

1. Product Overview

Synkio Safety is the intelligence and protection layer for all Synkio payments, chats, and user interactions. It automatically scans wallets, transactions, URLs, tokens, contracts, and approvals to prevent scams, drainer attacks, and malicious on-chain behavior.

The user sees simple results; the engine handles all the complexity.

2. Key Goals

1. Make Synkio the safest P2P payment platform.
 2. Reduce user mistakes, bad addresses, and fraud attempts.
 3. Provide real-time, on-chain due diligence without complexity.
-

3. Core Features

3.1 Wallet Risk Scanning

- Run Threat Risk API checks before every payment.
- Score: Safe / Medium / Risky / Dangerous.
- Show reason tags: "Linked to malicious contract", etc.

3.2 URL Safety (Chat Protection)

- Auto-scan all links in chat.
- Warning overlay for malicious URLs.
- Block opening dangerous links.

3.3 Token & Contract Analysis

- Use Contract Risk + Token risk signals.
- Show warnings when users attempt to pay with risky assets.

3.4 Approval Risk Dashboard

- Read all approvals for user wallet.
- Highlight:
 - Unlimited allowances
 - Unknown contracts
 - Past drainers
- Allow revokes via Thirdweb.

3.5 Transaction Risk Scoring

- Pre-transaction checks using Transaction Risk API.
- Real-time warnings for:
 - High-risk receivers
 - Code red contracts

3.6 Wallet Health Score

A unified score based on: - Exposure risk - Threat risk - Approval risks - Contract/token exposure

3.7 Vendor Trust Badges

For Synkio sellers: - Green = verified safe - Yellow = caution - Red = avoid

4. Target Users

- Buyers
 - Sellers
 - P2P traders
 - Social commerce users
 - Web3 beginners who need protection
 - Synkio creators and vendors
-

5. User Stories

Buyer

As a buyer, I want to know if the seller's wallet is safe so I don't send money to a scammer.

Seller

As a seller, I want Synkio to warn buyers if a fake wallet imitates mine.

Everyday User

As a user, I want Synkio to warn me before I click a malicious link.

Crypto User

As a crypto user, I want to see my risky approvals and revoke them.

6. System Architecture

6.1 Components

- Safety Microservice (NestJS)
- DD.xyz API
- BlockRader signals
- Synkio AI reasoning engine
- Thirdweb (for revokes)

6.2 Flow: Payment

1. User selects recipient.
 2. Safety engine checks:
 3. Threat Risk API
 4. Contract Risk API
 5. UI displays:
 6. Green (safe) or Red (dangerous)
 7. User proceeds or cancels.
-

7. API Integrations

DD.xyz APIs:

- Threat Risk
- Approval Risk
- Exposure Risk
- Transaction Risk
- Contract Risk
- URL Risk

Wrapped Through Synkio Safety Engine

We normalize everything into:

```
{
  riskLevel: "low|medium|high|critical",
  reasons: [...],
  suggestions: ...,
  metadata: ...
}
```

8. UX/UI Requirements

8.1 Wallet Safety Banner

- Display safety status above each payment.
- Colors: green, yellow, red.

8.2 Chat Warnings

- Inline URL badge.
- Tap → safety details.

8.3 Dashboard

- Health score a- Approvals list a- Risk history

8.4 Vendor Badges

- Small colored tag next to username.
-

9. KPIs

- % of flagged unsafe addresses
 - Reduction in user scam complaints
 - Engagement with safety dashboard
 - Successful revokes
-

10. Roadmap

Phase 1: Wallet + URL + Token scanning

Phase 2: Approvals dashboard + revokes

Phase 3: Smart contract deep scanning

Phase 4: Developer APIs

Phase 5: Synkio Safety standalone app/website

11. Risks + Mitigations

- **API downtime** → local fallback cache
 - **Slow queries** → async, cached results
 - **User confusion** → simple UX copy
-

12. Success Criteria

Synkio Safety succeeds if: - Users stop losing funds to scammers. - Synkio becomes known as the safest chat-commerce platform. - Safety becomes a competitive advantage.

End of document.

Solana Grant Application Draft + Full PRD for Synkio Safety

GRANT APPLICATION — Solana Foundation Grant Program

Project Name: Synkio Safety

Organization: Synkio Technologies

Project Category: Payments, Consumer Safety, Risk Scoring, Wallet Intelligence, On-chain Commerce

Chain: Solana

Stage: MVP → Pilot → Production

1. Executive Summary

Synkio Safety is a real-time, Solana-powered security layer designed for everyday buyers and sellers who transact through Synkio — a chat-based payments platform. It protects users from common Web3 risks: sending to the wrong wallet, interacting with malicious contracts, approving bad tokens, or paying fraudulent vendors.

Our goal: **make on-chain payments feel as safe and intuitive as mobile money.**

Solana provides the ideal foundation due to its low fees, high throughput, and thriving consumer ecosystem.

The grant will accelerate development of: - The Safety Engine (risk scoring, contract analysis, behavioral checks) - A Solana-native wallet reputation layer - Real-time transaction simulation - Vendor verification with on-chain proofs - Plug-and-play APIs for devs building consumer commerce flows

2. Problem Statement

Most real users don't transact on-chain because: - They're scared of sending funds to the wrong wallet. - They can't verify vendors. - They don't understand approvals. - They can't check if a contract is risky. - They don't want to lose money.

Solana solves cost and speed — Synkio Safety solves trust.

There's a massive gap: **simple, user-facing protection for real people transacting day-to-day.**

3. Proposed Solution: Synkio Safety

A security layer that runs before every payment.

Core Features

1. **Wallet Risk Score**

- 2. New wallet? Old wallet? High-risk patterns?
- 3. Flag scam-associated wallets using heuristics + open databases.

4. **Contract Safety Checks**

- 5. Simulate transactions before broadcasting.
- 6. Detect unsafe contract methods.
- 7. Check for freeze, mint, upgrade authority patterns.

8. **Vendor Verification Layer**

- 9. Vendor reputation history
- 10. Authentication proofs
- 11. Trusted network badges

12. **URL + Off-chain Safety**

- 13. Basic phishing detection
- 14. Redirect checks

15. **Human-Friendly Prompts**

- 16. "This wallet has no history. Are you sure?"
- 17. "This contract can move your tokens."

18. "Vendor not verified."

19. **Synkio Safety API** for other Solana developers

20. Risk scoring

21. Wallet intelligence

22. Transaction checks

This creates a safety net for mainstream users entering Web3 commerce.

4. Why Solana?

Solana is the only chain that can make real-time, AI-assisted safety checks feel instant because: - Near-zero transaction costs - High throughput for simulation and indexing - Fast confirmation times - Expanding consumer and payments ecosystem

Synkio Safety becomes a consumer-grade security stack for Solana's next 100M users.

5. Technical Architecture

Components

- **Core Safety Engine** (Rust / TypeScript)
- **On-chain risk indexer** (Helius / RPC providers)
- **Transaction simulation module** (Solana Runtime Simulation)
- **Vendor verification registry** (Solana program)
- **API gateway** (NestJS)
- **Front-end components** (React / Svelte optional)

Data Flow

1. User attempts to pay or approve.
 2. Synkio Safety intercepts.
 3. Runs risk checks (wallet, contract, token, vendor).
 4. Produces a real-time human-readable security summary.
 5. User proceeds → transaction broadcast.
-

6. Milestones (Grant Deliverables)

M1 — Core Safety Engine (6–8 weeks)

- Wallet risk scoring
- Basic contract analysis

- Token authority checks
- MVP API endpoints

M2 — Vendor Verification System (6 weeks)

- On-chain vendor registry program
- Reputation badges
- Verified vendor onboarding

M3 — Transaction Simulation Layer (8 weeks)

- Solana runtime simulation integration
- Real-time warnings
- Approval safety

M4 — Developer APIs + Docs (4 weeks)

- Public endpoints
- SDK (TS + Rust client)
- Sandbox + test suite

M5 — Synkio Integration Launch (3–4 weeks)

- Final UI components
- Mobile + web integration
- User testing + telemetry

7. Grant Budget

Total Request: \$65,000

Breakdown: - \$20k Engineering (Safety Engine) - \$15k Vendor Registry Program - \$10k Transaction Simulation Infrastructure - \$8k Security + audits - \$7k Frontend + UX - \$5k Documentation + DevRel

We will co-fund additional infra and staffing.

8. Team

- **Gar Manji (Lead Engineer):** Blockchain, backend, NestJS, Thirdweb, wallet infra.
- **Rust Team (4 devs):** EVM, Rust, Solana program dev.
- **Design + Product:** UX for consumer Web3.

Team already shipping Synkio — a chat-based payments platform.

9. Impact for Solana Ecosystem

Synkio Safety fills a critical gap: - Protects newcomers - Increases transaction confidence - Reduces user losses - Boosts adoption for consumer apps - Gives devs a plug-and-play safety layer

This is infrastructure Solana *needs* to onboard the next wave.

PRODUCT REQUIREMENTS DOCUMENT (PRD)

1. Product Overview

Synkio Safety is a pre-transaction risk engine integrated into Synkio. It analyzes wallets, contracts, tokens, and vendors to prevent user mistakes and scams.

Target users: - Buyers and sellers using Synkio - New crypto users - Small merchants - P2P traders - Solana developers who need a safety layer

2. Goals

Primary Goals

- Prevent user losses
- Build trust in on-chain payments
- Provide clear, understandable risk signals

Secondary Goals

- Provide APIs to third-party apps
 - Build a Solana-native safety infrastructure layer
-

3. Key Features (Detailed Requirements)

3.1 Wallet Risk Scoring

- Age of wallet
- Transaction history
- Interaction graph
- Known scam associations
- Behavioral anomalies
- Real-time flags

3.2 Contract Safety Analysis

- Detect dangerous methods
- Identify upgradeable programs
- Detect authority risks
- Simulate interacts and flag unexpected behavior
- Show warnings in plain language

3.3 Token Safety Checks

- Freeze authority
- Mint authority
- Supply anomalies
- Honeypot patterns
- Fake token impersonation detection

3.4 Vendor Verification System

- Vendor registry on Solana
- Badges (Verified, Trusted, First-time vendor)
- Vendor rating aggregation
- Transaction volume history

3.5 Safety Prompts

- Must be simple and human-readable
- Red/yellow/green status
- One-line explanations

3.6 Developer Safety API

- WalletScore API
- ContractCheck API
- TransactionSim API
- TokenCheck API

4. Non-Functional Requirements

- <250ms response time
 - High uptime API
 - Clear documentation
 - GDPR-compliant data handling
 - Resilient caching + rate-limiting
-

5. System Architecture

- Solana Program (Vendor Registry)
 - Solana Indexing Layer
 - Node-based API Gateway
 - Rust-based analyzers
 - Redis cache layer
 - Postgres / Neon for metadata
-

6. User Flows

6.1 Payment Flow with Safety Check

1. User enters amount
2. System fetches wallet + vendor info
3. Safety Engine runs checks
4. Warnings displayed
5. User confirms
6. Transaction broadcast

6.2 Developer Flow

1. Dev calls Safety API
 2. Receives structured risk report
 3. Integrates into their UX
-

7. Success Metrics

- 95% user understanding of safety messages
 - Reduction of failed/scam transactions
 - API adoption by external devs
 - Vendor registry growth
 - Monthly active protected users
-

8. Risks & Mitigations

- **False positives:** Use adjustable thresholds.
 - **High load:** Add caching + horizontal scaling.
 - **Security:** Perform audits.
-

9. Timeline

~24-30 weeks total (see milestone breakdown above).

10. Appendices

- UI examples
 - Safety categories
 - Threat models
 - API schemas
-

The draft is ready for refinement. Let me know if you want: - A more formal academic tone - A more aggressive startup tone - Visual diagrams - API schema tables - A polished version for submission