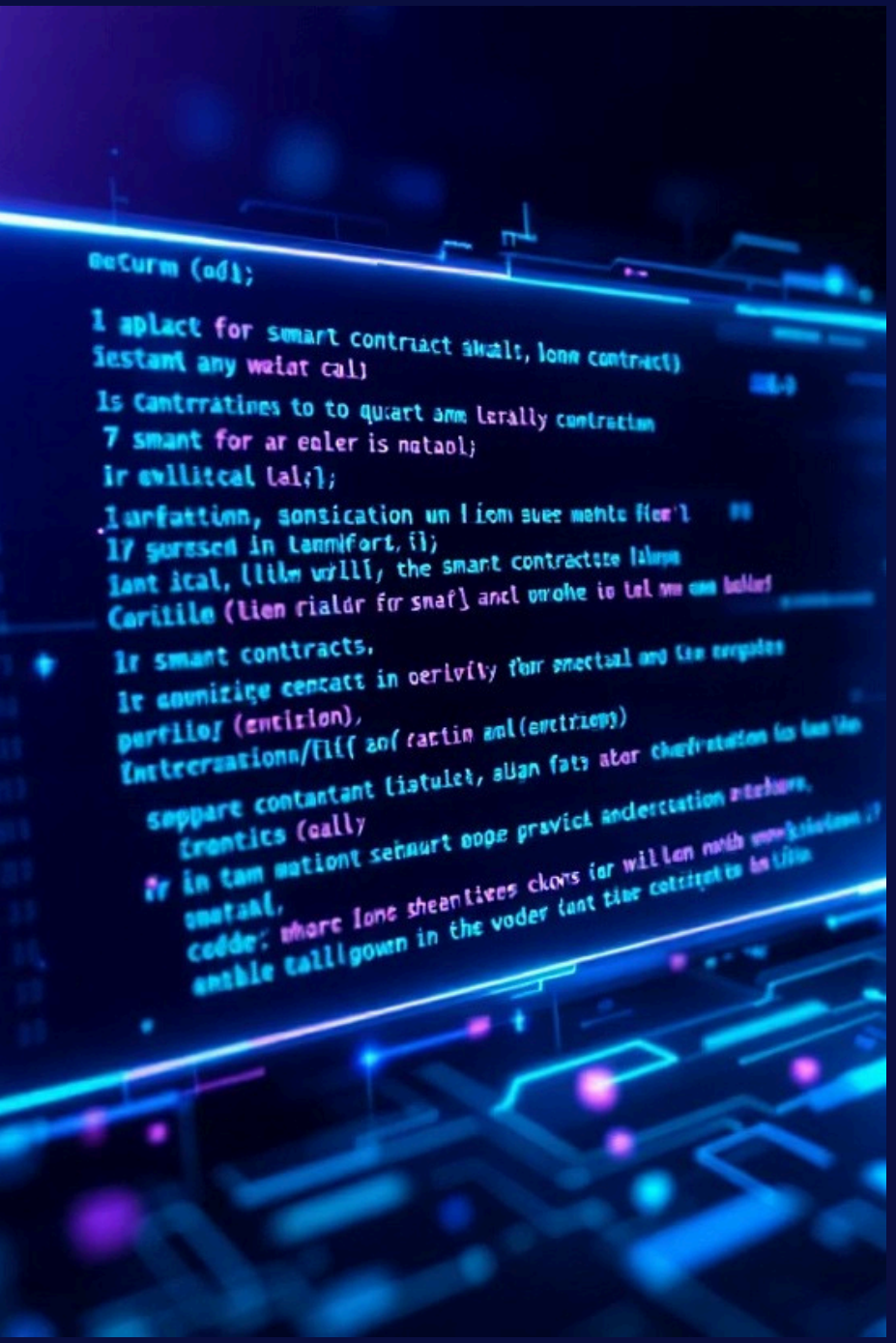
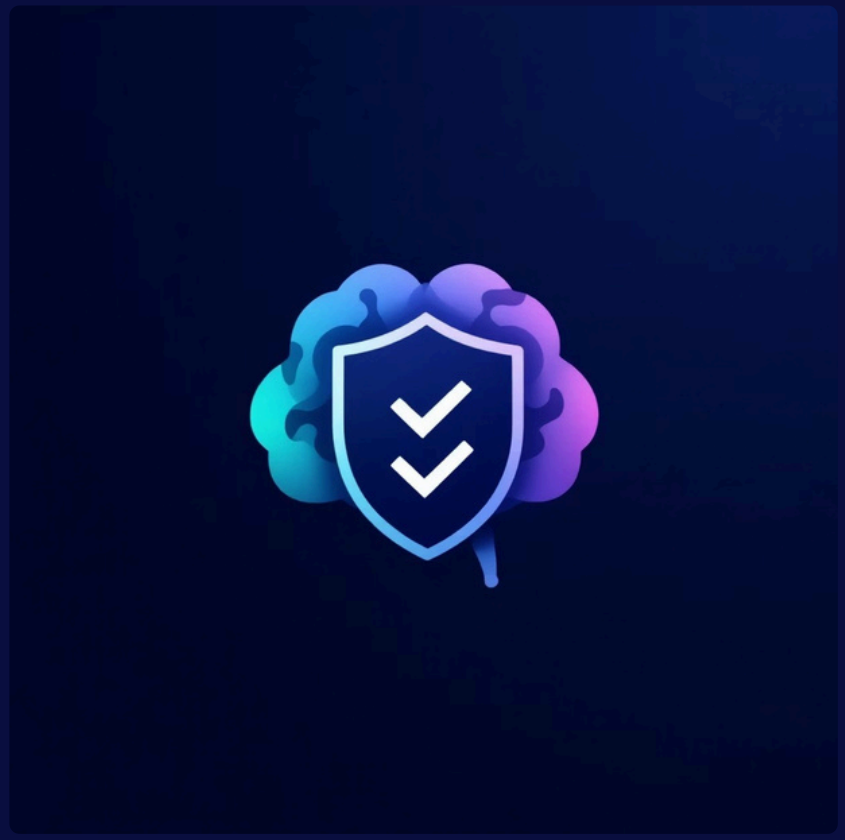


QUBIC SMARTGUARD + VOTING CONTRACT

An all-in-one solution for Qubic smart contracts

PITCH DECK

Qubic SmartGuard + Governance Voting CLI



Qubic SmartGuard: AI audit, validation, and documentation for Qubic contracts.

Governance Voting CLI: Advanced governance voting contract, production-ready.

Hackathon Team: SmartGuard 3 Qubic Track Lablab.ai Hackathon 2025

Problem Statement

Manual and error-prone audits

Theauditing ofsmartcontractsisa manual, time-consuming process prone to human errors, leading to vulnerabilities.

Lack of automated tools

Developers donot have accessto automated and accessible auditing tools to assist them in this critical process.

Complex deployment

Deployingproduction-qualitycontracts requires in-depth expertise of the command-line interface (CLI).

Security gaps

Existing solutionsoften lack enterprise-grade security and comprehensive documentation, essential for trust.

Our Innovative Solution

Smar tGuard

AnAI-powered platform for automated auditing, validation, and documentation of Qubic contracts.

Voting Contract and CLI

Anadvanced,production-readyvoting contract with comprehensive deployment tools for robust governance.

By combining SmartGuard and the Voting CLI, we offer an end-to-end platform for developing secure and professional Qubic smart contracts.

Key Features of SmartGuard

C++ Contract Analysis (QPI)

Uploadand analyzeyourC++contracts for immediate validation.

Automated AI Comments

Generateclearandconcisecodecomments for better readability.

Advanced Security Audit

Receivecomprehensiveauditreports to identify vulnerabilities.

Detailed Documentation

Createfunctionalspecifications,flow diagrams, and test plans.

Editable Simulation Scenarios

Testyourcontractsincustomizablesimulated environments.

Future RPC Integration

Prepareforreal-timeon-chain execution via RPC integration.

Technologies Powering Our Solution

Python & Streamlit

Backend logic & intuitive UI.

LangChain

AI agent orchestration.

Groq LLM

High-speedAI inference(Llama and Deepseek r1 models).

Mermaid.js

Visualizescontract flows.

C++ CLI

Directcontract interaction.

Git / GitHub

Versioncontrol& collaboration.

AI SmartGuard Workflow

Experienceastreamlinedworkflow forQubicsmartcontracts,withseamless AI assistant interaction via CLI.

Submit Contract

Securely upload your Qubic C++ smart contract files (QPI) via Streamlit .

AI Analysis

LangChain orchestrates Groq LLM to provide automated comments, semantic validation, and comprehensive security audits.

Docs & Simulation

Generate functional specifications with Mermaid diagrams, test plans, and interactive simulation scenarios for enhanced insights.

Reports & Future

Comprehensive audit reports and test plans are generated. Future RPC integration will enable direct on-chain execution.

Smart Contract Workflow CLI

C++ Contract Compilation

Transformthesourcecodeintoan executable bytecode for the Qubic blockchain.

Bytecode Validation

Ensurethecomplianceand security of the generated bytecode before deployment.

Testnet/Mainnet Deployment

Deployyourvalidatedcontractonthe Qubic test or main network.

Voting Function Calls

Interactdirectlywiththevoting contract functions via the command line.

Audit Analysis and Logging

Examinedetailedreportsandlogsfor comprehensive monitoring.

Example of CLI Workflow

Contract Compilation

```
qubic-cli -contractcompile
```

Converts your C++ code into executable Qubic bytecode.

Bytecode Validation

```
qubic-cli -contractvalidate
```

Verifies the integrity and compliance of the bytecode.

Network Deployment

```
qubic-cli -contractdeploy
```

Deploys your contract on the chosen Qubic network (testnet/mainnet).

Voting Operations

```
createProposal, registerVoter, castVote, getResults
```

Interact with the contract to manage the voting process.

Smart Contract Architecture

Key Data Structures

- **Proposals:**Details, status,results.
- **Votes:** Secure and timestamped records.
- **Voters:** Identities and reputation information.

Core Functions

- **initializeContract:**Prepares the contract for usage.
- **createProposal:** Submits a new idea for voting.
- **registerVoter:** Allows eligible users to register.
- **castVote:** Securely records a voter's choice.
- **getProposalResults:** Retrieves the final results of a proposal.
- **closeProposal:** Finalizes the voting process for a proposal.

System Integration

Oursolutionistheperfectunionoftwopowerfulcomponents, working in synergy for seamless Qubic development.

60%

Smar tGuard

Developed in Python with an intuitive Streamlit user interface. Focused on auditing, documentation, and simulation to ensure quality.

40%

Voting CLI

Written in C++ and directly integrated into the Qubic ecosystem. Handles compilation, deployment, and all voting operations on the chain.

Together, they form a complete platform for the development, deployment, and management of Qubic smart contracts.

Thank You!

We appreciate your time and interest in QubicSmartGuardandourVoting Contract solution. We are confident in its potential torevolutionizedecentralizedgovernance.