NEXT-GEN SOFTWARE QUALITY

AI-Based Code Quality Analyzer

Deterministic Logic Fused with Generative Intelligence

Shaik Kudhan

Current Landscape

Analyzing the operational friction in high-velocity development

THE COST OF MANUAL DEBT

Review Bottlenecks: Manual cycles stall deployment velocity and impact developer mental bandwidth.

Invisible Debt: Logic complexity accumulates silently without a unified metrics dashboard.

Quality Drift: Fragmented coding standards across distributed teams lead to brittle architectures.

STRATEGIC OBJECTIVES

Structural Triage

Instant identification of "code smells" via deep AST parsing.

Metric Tracking

Quantifying health with Cyclomatic Complexity & MI scores.

AI Insights

Context-aware refactoring tips via Hybrid Relay LLMs.

Shift-Left Flow

Zero-trust commit gates via automated Git Hook integration.

THE ENGINEERING FOUNDATION

Infrastructure Layer	Technology Provider	Strategic Role
Logic Parsing	`ast` (Python Native)	Structural mapping and syntax deconstruction
Code Metrics	`radon`	Calculation of Complexity and Maintainability indices
Brain Layer	Gemini / Ollama	Hybrid cloud/local reasoning relay mechanism
Dashboard	Streamlit	Real-time interactive developer experience

Technical Architecture

A deep dive into Module 1 and the AI-driven feedback relay

AST LOGIC ANALYSIS

Module 1: Deep Inspection

We deconstruct Python scripts into non-linear logic maps using `ast.walk()`. This allows us to inspect imports, class hierarchies, and nested loops independently of formatting.

Decision Node Tracking: Our custom walker calculates logical depth by identifying conditional branches and try-except blocks.

ALGORITHMIC SCORING

100

Base Health

10

Complexity Ceiling

90+

Excellence Target

$$\text{HealthScore} = 100 - \sum \text{Complexity}_{\text{penalty}} + \sum \text{Debt}_{\text{smells}}$$

AUTOMATION IMPACT TREND

Aiming for 95% Automation Coverage in post-deployment maintenance

STRATEGIC ROADMAP

Phase 1.5

JS/TS AST Parsing Rules Expansion

Phase 2.0

Native CI/CD Runner Integration

Phase 2.5

Security Audit & OWASP Checkers

Phase 3.0

Self-Learning Style Guidelines

Standardizing Excellence.
One Commit at a Time.

https://github.com/Kudhan/code-analyzer.git

Q & A

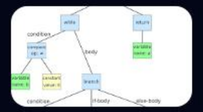
Thank you for your attention.

Bridging Analysis with AI Insights

IMAGE SOURCES

https://www.openprofessionalgroup.com/wp-content/uploads/2025/10/OPG_October2025_post.png

Source: www.openprofessionalgroup.com



https://upload.wikimedia.org/wikipedia/commons/thumb/c/c7/Abstract_syntax_tree_for_Euclidean_algorithm.svg/1280px-Abstract_syntax_tree_for_Euclidean_algorithm.svg.png

Source: en.wikipedia.org

https://static.vecteezy.com/system/resources/previews/002/082/174/non_2x/abstract-tech-background-floating-numbers-hud-background-matrix-particles-grid-virtual-reality-hardware-quantum-form-vector.jpg

Source: www.vecteezy.com