Comprehensive Code Analysis Workflow Report

# Overview

This report outlines the architecture, workflow, and error-handling strategies of the Code Analysis System. The platform integrates Flask, Hugging Face Models, Google Gemini API, and static/dynamic analysis tools to provide multi-layered evaluation of Python and Java source code.

# Phase 1: User Interface & File Upload

* • User accesses http://127.0.0.1:5000.
* • Flask serves a dark-themed index.html.
* • User selects a .py or .java file.
* • JavaScript validates extension and size (<10MB).
* • Form submits file to /analyze.

# Phase 2: Backend Processing Pipeline

* • Flask receives uploaded file.
* • Validates file type (.py / .java).
* • Saves file to uploads/.
* • Reads file content.
* • Detects programming language.

# Phase 3: AI Model Analysis (Parallel Processing)

* • Hugging Face Models (CodeT5) → Code review, Unit tests, Documentation (fallback).
* • Hugging Face Models (CodeBERT) → Bug detection & classification.
* • Gemini API → Bug detection score, time complexity, quality metrics, structured docs, corrected code.

# Phase 4: Static Analysis & Compilation Checks

* • Python: AST parsing, unittest + coverage, Bandit security scan, Radon complexity.
* • Java: Compilation with javac, error reporting, class/method/package structure.

# Phase 5: Scoring & Quality Assessment

* • Weighted scoring system combining review, test, documentation, and UX scores.
* • Quality metrics include maintainability, readability, best practices.

# Phase 6: Report Generation

* • JSON Report → core metrics, AI outputs, static analysis, metadata.
* • PDF Report → styled with ReportLab, organized sections with results.

# Phase 7: Frontend Display & User Interaction

* • Results displayed in browser with scores, progress bars, bug reports, corrected code.
* • Download JSON and PDF report options available.

# Phase 8: Error Handling & Fallbacks

* • Gemini unavailable → fallback to Hugging Face.
* • Hugging Face failure → fallback to static analysis.
* • Compilation errors → high bug score + corrected code attempt.
* • User-friendly error handling in frontend and backend.

# Workflow Diagram

The diagram below illustrates the complete workflow. For the final version, replace this placeholder with a professionally designed flowchart.

[Workflow Diagram Placeholder - Flowchart with Phases 1 to 8]

# Key Features Summary

✅ Dual AI Integration: Hugging Face + Gemini

✅ Multi-language Support: Python + Java

✅ Comprehensive Analysis: Static + Dynamic + AI

✅ Scoring System: Weighted quality assessment

✅ Report Generation: JSON + PDF formats

✅ Resilient Design: Multiple fallback mechanisms

✅ User-Friendly Interface: Dark theme + responsive frontend

# Conclusion

This workflow ensures robust, multi-layered, and resilient code analysis, providing developers with actionable feedback, improved readability, and reliable bug detection. The system merges AI-driven insights with traditional static analysis, producing a highly informative and user-friendly evaluation process.