Software Engineering Project

**Needs Statement**

**Purpose**

This document outlines the functional needs for the Code Clone Detection Tool. The purpose of this project is to help developers identify and eliminate code redundancy, enhancing code quality and maintainability in intermediate-sized software. The tool focuses on detecting redundant code within a program, making it valuable for refactoring and reducing technical debt. The primary objective is to tackle realistic software development challenges, including implementing effective algorithms for code comparison, creating user-friendly interfaces, conducting code quality assessments, and supporting refactoring efforts.

**Code Clone Detection Tool Functional Need Statement**

The Code Clone Detection Tool is a software application designed to assist programmers in detecting duplicate code (code clones) within a project. By identifying redundant code, the tool provides actionable insights for refactoring and optimizing the codebase, improving maintainability and reducing technical debt. Debt that can result in the accumulation of shortcuts or poor practices in software development that can lead to increased maintenance costs and more difficult future modifications. By addressing these issues early, the tool enables developers to maintain cleaner, more efficient code, enhancing long-term project stability and reducing the likelihood of costly refactoring efforts down the line.

1. Code Clone Detection Tool Functions
   1. Analyze an existing software project, written in Java or Python to identify code clones that are exact or near exact duplicates.
   2. Detect and categorize different types of code clones, including:
      1. Type 1 (exact clones): Identical code except for whitespace or comments.
      2. Type 2 (renamed clones): Identical code with variations in identifiers or variable names.
      3. Type 3 (modifications): Similar code with small additions or modifications.
   3. Display the locations of detected clones within the project, with details of their similarity percentage and classification.
   4. Generate a detailed report summarizing the detected clones and their potential impact on the project.
   5. Provide a graphical user interface (GUI) for navigation clone detection results that will allow the users to filter by the clone type or the percentage of similarity.
   6. Recommend refactoring strategies allowing users to remove or consolidate the redundant code clones.
2. Detection Specifications
   1. The tool will be able to parse Java and Python languages, analyzing files for clone detection.
   2. The tool will be able to ignore comments and blank lines during the analysis to ensure that only the logical code content is considered.
   3. Provide options for adjusting the sensitivity of clone detection, including thresholds for similarity percentage.
   4. The tool will handle large codebases efficiently, ensuring that detection results are available in a reasonable amount of time.
3. Clone Detection Calculations
   1. Code clone metrics are calculated based on the following attributes.
      1. Number of exact clones (Type 1)
      2. Number of renames clones (Type 2)
      3. Number of modified clones (Type 3)
      4. Percentage of total lines of code involved in cloning
   2. Detection Sensitivity can be adjusted by similarity thresholds:
      1. Detect clones with at least 70% similarity
      2. Detect clones with at least 90% similarity
      3. Detect exact clones at 100% similarity
   3. Report metrics will include:
      1. Number of clones found per file and project wide.
      2. Percentage of code duplication that includes the clones line versus the total lines of code.
      3. Locations of code clones with links to the source code lines.
4. Documentation Specifications
   1. The Code Clone Detection Tool design documentation must describe the tools architecture, including clone detection algorithms and the user interface.
   2. The user manual must describe the installation and operation of the tool, detailing how to load projects, run analysis, and interpret results.
   3. Software test results must demonstrate that the tool satisfies each of the functional requirements described in sections 1.1, 1.3, 2.1 through 3.3.