Group Members:

Muhammad Asad Ullah Turab - 221844
Muhammad Saad - 221816
Israr Ahmed - 221876
Mehboob Ali - 221824
Furqan Saeed - 221860

Software Re-engineering

Code Smell Detector - Project Report

1. Project Description

The Code Smell Detector is a Java-based static analysis software that takes Java source code as input and automatically identifies code smells and quality issues. Code smells are indicators of poor design or implementation choices that may lead to maintenance and scalability problems. The purpose of this software is to help developers analyze Java programs and receive structured feedback about potential improvements. The application provides results through a graphical interface, making it easy to understand and act upon detected issues.

2. Technologies Used

Programming Language: Java 25

Build Tool: Maven 4.0.0

User Interface: Java Swing

Architecture: Model-View-Controller (MVC)

File Handling: Java NIO

Pattern Matching: Java Regular Expressions

IDE: Visual Studio Code

Operating System Support: Cross-platform

3. Code Smells Detected

The software detects multiple common code smells including Long Method, Large Class, Too Many Parameters, God Object, Dead Code, Unused Variables, High Cyclomatic Complexity, Primitive Obsession, Long Parameter List, Missing Javadoc, Magic Numbers, Feature Envy, Lazy Class, and Speculative Generality. Each code smell is analyzed using predefined thresholds and categorized by severity.

4. Detection Approach

The analyzer reads Java source files line by line and applies pattern matching techniques using regular expressions. It calculates metrics such as method length, class size, parameter count, and conditional complexity. Detected values are compared against thresholds to determine whether a code smell exists. All findings are stored with detailed information including file name, line number, description, and severity.

5. How to Run the Software

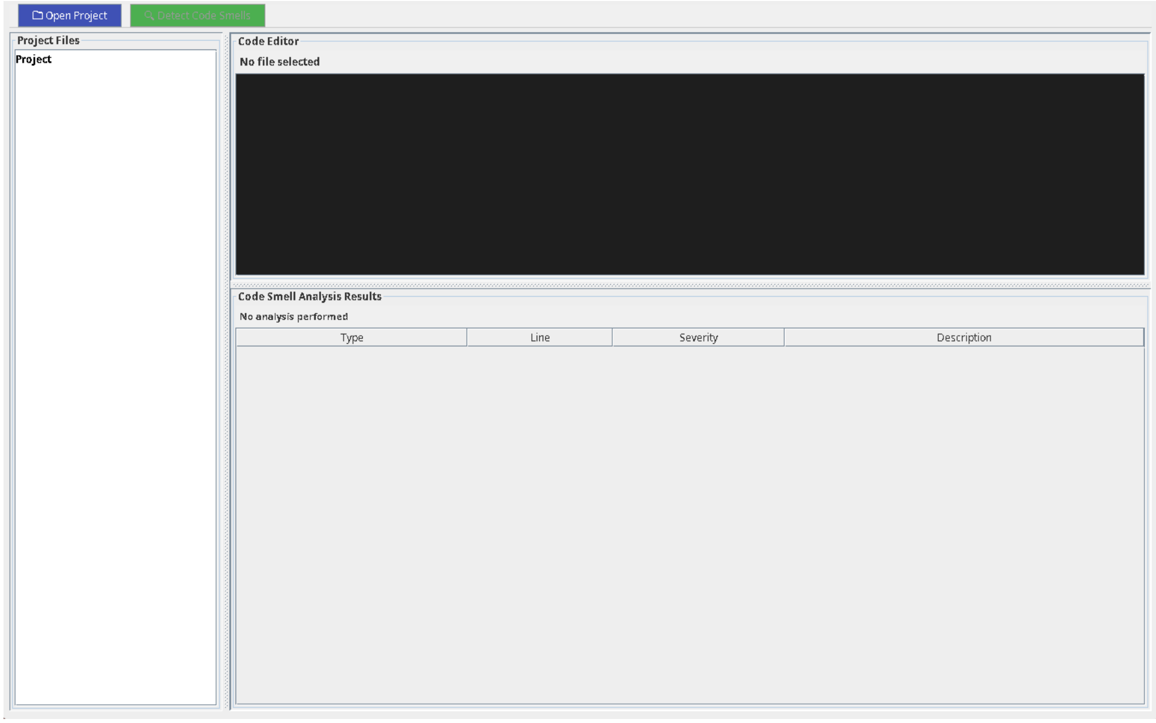
The project requires JDK 25 and Maven. After building the project using Maven, the application can be run either through an IDE or the terminal. Once launched, the user selects a Java file, and the software automatically analyzes it and displays the detected code smells.

6. Project Architecture

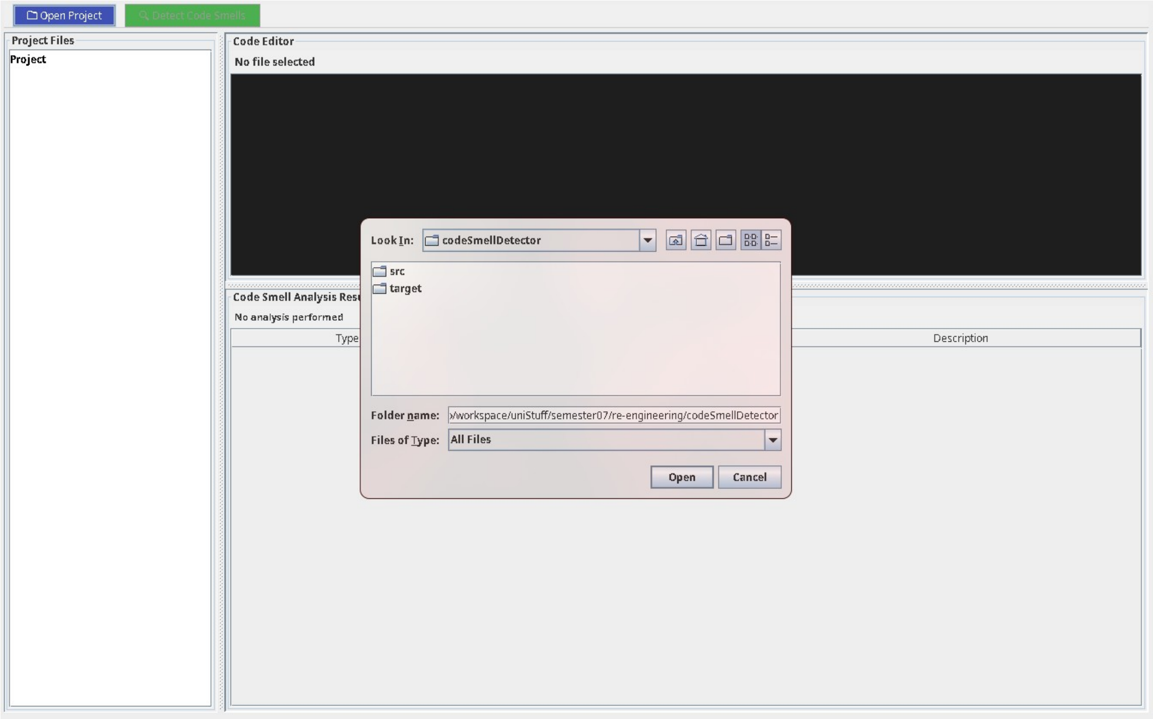
The project follows the MVC architecture. The Model includes data structures for code smells and analysis logic. The View consists of Swing-based user interface components. The Controller manages communication between the UI and analysis engine.

7. Features

The software provides automated static analysis, real-time feedback, severity-based categorization, and a user-friendly interface. It allows developers to quickly identify problematic areas in Java code.



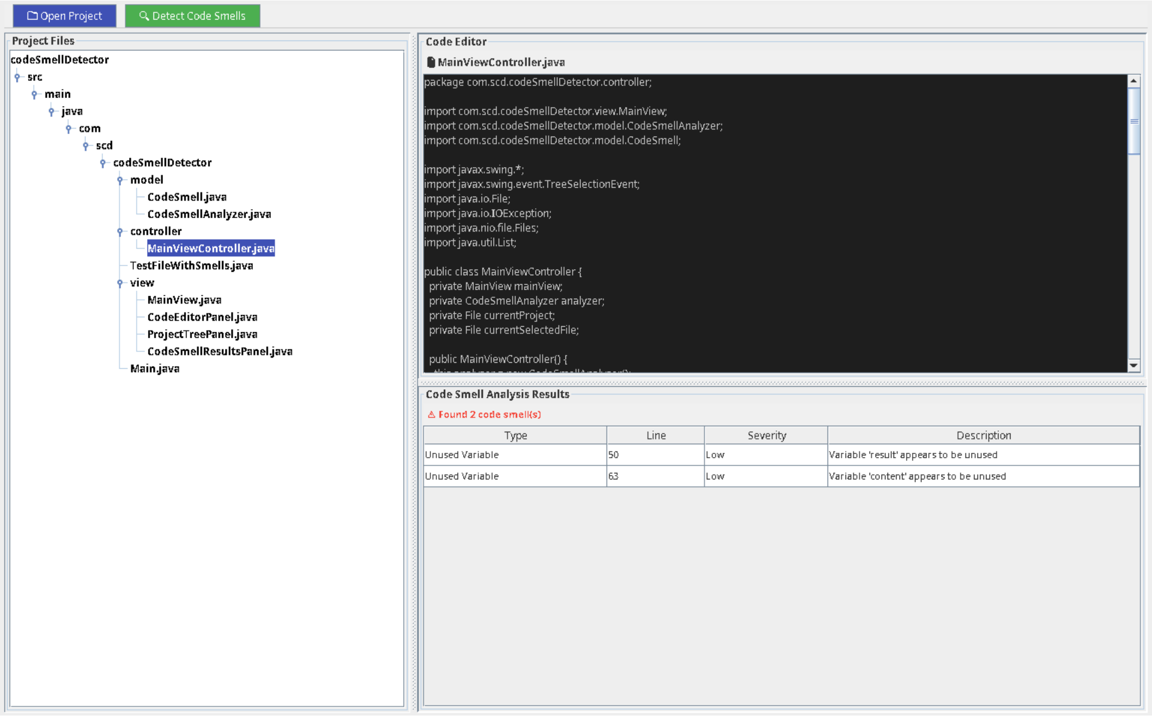
Type	Line	Severity	Description



Look In: codeSmellDetector

Folder name: /workspace/uniStuff/semester07/re-engineering/codeSmellDetector

Files of Type: All Files



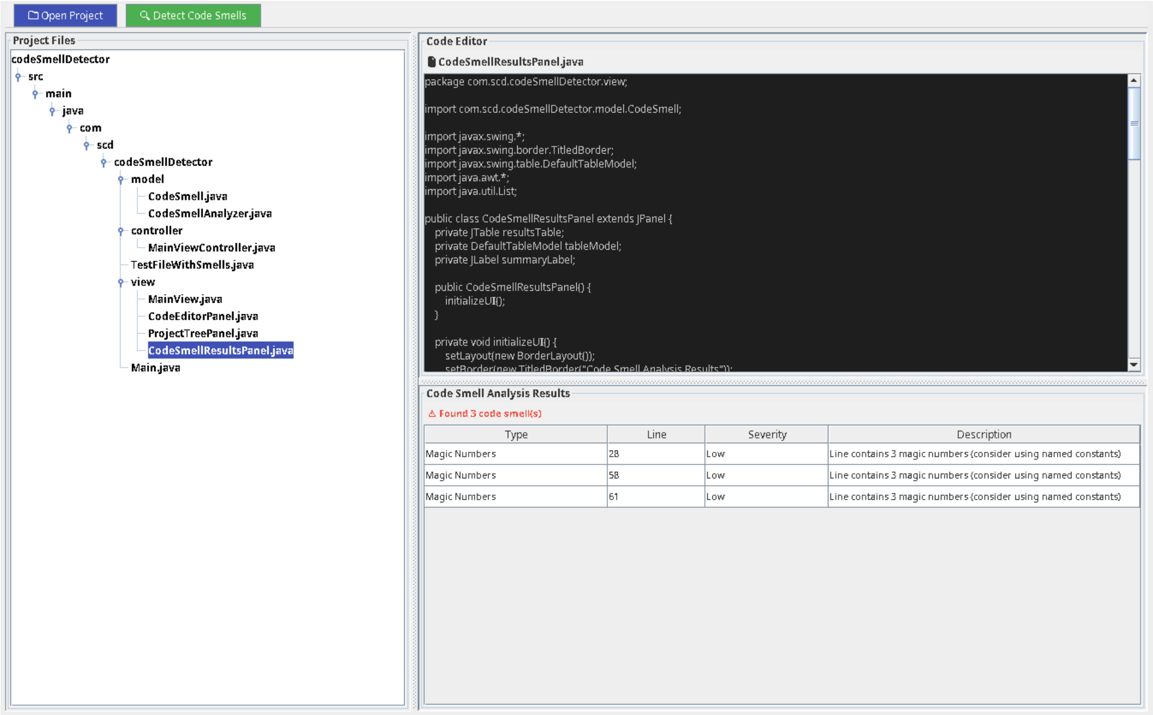
MainViewController.java

```
package com.scd.codeSmellDetector.controller;

import com.scd.codeSmellDetector.view.MainView;
import com.scd.codeSmellDetector.model.CodeSmellAnalyzer;
import com.scd.codeSmellDetector.model.CodeSmell;

import javax.swing.*;
import javax.swing.event.TreeSelectionEvent;
import java.io.File;
import java.io.IOException;
import java.nio.file.Files;
import java.util.List;

public class MainViewController {
  private MainView mainView;
  private CodeSmellAnalyzer analyzer;
  private File currentProject;
  private File currentSelectedFile;

  public MainViewController() {
    this.analyzer = new CodeSmellAnalyzer();
```

Code Smell Analysis Results

⚠ Found 2 code smell(s)

Type	Line	Severity	Description
Unused Variable	50	Low	Variable 'result' appears to be unused
Unused Variable	63	Low	Variable 'content' appears to be unused



CodeSmellResultsPanel.java

```
package com.scd.codeSmellDetector.view;

import com.scd.codeSmellDetector.model.CodeSmell;

import javax.swing.*;
import javax.swing.border.TitledBorder;
import javax.swing.table.DefaultTableModel;
import java.awt.*;
import java.util.List;

public class CodeSmellResultsPanel extends JPanel {
    private JTable resultsTable;
    private DefaultTableModel tableModel;
    private JLabel summaryLabel;

    public CodeSmellResultsPanel() {
        initializeUI();
    }

    private void initializeUI() {
        setLayout(new BorderLayout());
        setBorder(new TitledBorder("Code Smell Analysis Results"));
```

Code Smell Analysis Results

⚠ Found 3 code smell(s)

Type	Line	Severity	Description
Magic Numbers	28	Low	Line contains 3 magic numbers (consider using named constants)
Magic Numbers	58	Low	Line contains 3 magic numbers (consider using named constants)
Magic Numbers	61	Low	Line contains 3 magic numbers (consider using named constants)