**AUTOMATED REFACTORING TOOL**

Scoping Document

Luke

Gilbert



Table Of Contents

[PROJECT DEFINITION 1](#_Toc198725497)

[Description 1](#_Toc198725498)

[Minimum Viable Product 1](#_Toc198725499)

[Functional Requirements 1](#_Toc198725500)

[Non-functional Requirements 1](#_Toc198725501)

[Non Essentials (Nice to Haves) 1](#_Toc198725502)

[PROJECT PROGRAMME 2](#_Toc198725503)

[WORKFLOWS 3](#_Toc198725504)

[TECHNOLOGIES 4](#_Toc198725505)

[Frontend 4](#_Toc198725506)

[Backend 4](#_Toc198725507)

[Database 4](#_Toc198725508)

[Security 4](#_Toc198725509)

[Testing 4](#_Toc198725510)

# PROJECT DEFINITION

## Project Description

Create a VSCode extension which is used to automate the refactoring process of Python Code. The extension should use a rule-based approach to detection and refactoring of code smells. This will be achieved through using the Python’s in-built ast module to represent code in an abstract manner to allow for syntactic pattern matching to be implemented. The extension should include the following components:

* Automatic Detection of Code Smells
* Refactoring Suggestions
* Code Quality Metrics
* Code Change Previews

The tool should identify the following code smells:

* Non-idiomatic Python Code
* Long Method
* Large Class
* Duplicate Code
* Dead Code
* Feature Envy
* Inappropriate Intimacy
* Conditional Complexity
* Lazy Element
* Magic Numbers
* Fallacious Comments
* Type Embedded in Name
* Binary Operator in Name

## Minimum Viable Product

### Functional Requirements

The project should consider the following functional requirements:

* Works with open-source Python code
* User interface
* Code change preview and metrics
* Preservation of code functionality pre and post refactor

### Non-functional Requirements

* Clear explanations of code refactoring
* User friendly interface

## Non-Essentials (Nice to Haves)

N/A

# PROJECT PROGRAMME

# WORKFLOWS

# TECHNOLOGIES

## Frontend

## Backend

## Database

## Security

## Testing