SCRIPT ANALYSIS

A PROJECT REPORT

Submitted in partial fulfillment of the requirements

for the award of the degree of

Bachelor of Computer Application

SUBMITTED BY

ADHIVEL V

2213141033052

Under the guidance of

Dr. G. KIRUTHIGA, M.C.A., M.Phil., Ph.D.,

ASSOCIATE PROFESSOR

BACHELOR OF COMPUTER APPLICATIONS

GURU NANAK COLLEGE

(AUTONOMOUS)

Affiliated to University of Madras | Accredited at ‘A++’ Grade by NAAC

Approved by AICTE | An ISO 9001 2015 Certified Institution

Guru Nanak Salai, Velachery, Chennai – 600 042.

MARCH - 2025

GURU NANAK COLLEGE

(AUTONOMOUS)

Affiliated to University of Madras
Accredited at ‘A++’ Grade by NAAC | An ISO 9001 2015 Certified Institution
Guru Nanak Salai, Velachery, Chennai – 600 042.

School of Information Technology

BACHELOR OF COMPUTER APPLICATIONS

BONAFIDE CERTIFICATE

This is to certify that, this is a bonafide record of work done by **ADHIVEL V 2213141033052** of for the Final Year Project during the Academic Year 2023-24.

PROJECT GUIDE **HEAD OF THE DEPARTMENT**

Submitted for the Project Viva Voce Examination held on ________________ at GURU NANAK COLLEGE (Autonomous), Guru Nanak Salai, Velachery, Chennai - 600 042.

Internal Examiner
Date:

External Examiner
Date:

DECLARATION

I **ADHIVEL V (Register No: 2213141033052**) studying III Year Bachelor of Computer Applications at Guru Nanak College (Autonomous), Chennai hereby declare that this the Report of my Project entitled**, SCRIPT ANALYSIS** is the record of the original work carried out by me under the Guidance and Supervision of **Dr. G. KIRUTHIGA** towards the partial fulfillment of the requirements of the award of the Degree of **BACHELOR OF COMPUTER APPLICATIONS.** I further declare that this has not been submitted anywhere for the award of Degree/Diploma or any other similar to this before.

PLACE: CHENNAI	**ADHIVEL V**
DATE:	**2213141033052**

ACKNOWLEDGEMENT

I would like to thank the **Principal Dr. T. K. AVVAI KOTHAI** and **Vice Principal Dr. P. V. KUMARAGURU** for providing the necessary resources and facilities for the completion of this project.

I extend my deepest thanks to**. Ms. R .CAROLINE KALAISELVI** whose guidance, support, and encouragement were invaluable throughout this endeavor. Her expertise and insights have been instrumental in shaping this project and enhancing its quality.

I owe my Guide **Dr. G. KIRUTHIGA** a debt of gratitude for his/her invaluable guidance, patience, and encouragement. His/Her mentorship has been a beacon of light, steering me through the complexities of this project and helping me realize my potential.

I also like to extend my thanks to the **Faculty Members of BCA,** for their valuable suggestion during the course of the study of my project.

Last but not least, I thank my **family and friends** for their unwavering encouragement and understanding during this journey.

ABSTRACT

The Script Analysis Web App is a comprehensive tool designed to analyze and evaluate programming code, offering deep insights into its structure, complexity, and overall quality. By categorizing its analysis into three key areas — overall code structure, code complexity, and code quality metrics — the app provides developers with a clear and intuitive way to assess and enhance their codebase. The overall code structure analysis focuses on evaluating total lines of code, code lines, and comment lines, giving a clear view of the code's organization and readability. The complexity analysis delves into cyclomatic complexity, maximum nesting depth, and the number of functions and classes, providing valuable insights into the logical flow and nested depth of the code. This tool is particularly useful for developers, code reviewers, project managers, and institutions, enabling users to make data-driven decisions to improve code quality and maintainability. Its user-friendly interface simplifies the process of uploading or pasting and analyzing code, delivering instant feedback through detailed reports that can be easily reviewed and shared. By supporting multiple programming languages, the app accommodates diverse coding environments, making it suitable for projects of various sizes and complexities. Ultimately, the Script Analysis Web App serves as a reliable companion for maintaining high coding standards, promoting best practices, and enhancing the overall software development process.

The Script Analysis Web App saves time and effort. Whether used in educational settings to teach best coding practices or in professional environments to enhance project quality, the app serves as a reliable companion throughout the software development lifecycle. It ultimately promotes cleaner, more efficient code, helping teams adhere to industry standards and build more maintainable, high-quality software solutions.

TABLE OF CONTENTS

INTRODUCTION

1.INTRODUCTION

The Script Analysis Web App is a powerful tool designed to help developers analyze and evaluate their code by providing key insights into its structure, complexity, and maintainability. Supporting multiple programming languages, it offers real-time analysis by generating detailed metrics such as line count, function definitions, class structures, and cyclomatic complexity. With features like syntax highlighting, multi-language support, and downloadable reports, this app simplifies the process of code review and helps developers improve code quality and maintainability.

1.1. OBJECTIVES

The primary objective of this web application is to provide a tool for analying script files, offering insights into code structure, complexity, and basic metrics. This application aims to:

- Help developers gain a better understanding of their code base.
- Identify key metrics such as the number of lines, classes, and functions.
- Measure cyclomatic complexity and maintainability of code.
- Offer additional features such as syntax highlighting, multi-language support, and real-time analysis.

1.2. PROJECT DESCRIPTION

The **Script Analysis Web App** is a powerful web-based tool designed to help developers analyze and evaluate their code by providing key insights into its structure, complexity, and maintainability. This tool supports multiple programming languages, including Python, Java, C++, JavaScript, Ruby, Go, Swift, PHP, and C#.

The application offers real-time analysis, allowing users to upload their script files and instantly receive detailed metrics on line count, function definitions, class structures, and cyclomatic complexity. With features like syntax highlighting, multi-language support, and downloadable reports, this tool helps developers optimize their code and improve maintainability.

Built using **HTML, CSS, and JavaScript** for the frontend and **Python (Flask)** for the backend, the app does not require a database. Instead, it processes script files dynamically, generating structured insights that can be viewed and saved by the user.

This tool is especially useful for software developers, educators, and students who want to analyze and refine their code with minimal effort.

1.3. KEY FEATURES & FUNCTIONALITY

- Real-Time Code Analysis: Instantly analyzes script files and provides structured insights without requiring a page refresh.
- Code Metrics: Displays essential statistics such as total lines of code, number of functions, classes, and methods.
- Complexity Analysis: Evaluates cyclomatic complexity and maintainability index, helping developers optimize their code.
- Syntax Highlighting: Enhances readability by adding color-coded syntax for different programming languages.
- Multi-Language Support: Supports various programming languages for flexible analysis.
- Error Detection: Identifies potential syntax and logical errors within the uploaded script.
- Downloadable Reports: Allows users to save the analysis results in formats like PDF or CSV for future reference.

1.4. MODULES IN THIS PROJECT

The **Script Analysis Web App** is structured into multiple modules, each responsible for handling a specific functionality of the application. These modules work together to analyze scripts, generate insights, and present results in an interactive interface.

1. Frontend Module (HTML, CSS, JavaScript)

- Provides an interactive and user-friendly interface.
- Allows users to upload script files for analysis.
- Displays real-time analysis results, including code metrics and complexity.
- Implements syntax highlighting for better readability.
- Supports downloadable reports for user convenience.

2. Backend Module (Python, Flask)

- Handles user requests and processes uploaded script files.
- Performs code analysis, extracting metrics such as line count, functions
- Computes complexity analysis, including cyclomatic complexity and maintainability index.
- Sends processed data to the frontend for visualization.

3. Code Analysis Engine

- Parses and analyzes script files dynamically.
- Identifies functions, classes, methods, and their relationships.
- Supports multiple programming languages for analysis.

4. Report Generation Module

- Generates downloadable reports summarizing analysis results.
- Formats reports into PDF for easy sharing and record-keeping.
- Highlights key findings, including complexity levels and maintainability scores.

SYSTEM CONFIGURATION

2. SYSTEM CONFIGURATION

2.1. HARDWARE CONFIGURATION

The following minimum hardware specifications are required to run the Script Analysis Web App efficiently:

Processor: Intel i3 or higher / AMD equivalent

RAM: 4GB minimum (8GB recommended for better performance)

Storage: At least 500MB of free disk space

Display: Minimum resolution of 1280x720 (recommended: Full HD 1920x1080)

Internet Connection: Required for accessing the web application when hosted online

2.2. SOFTWARE CONFIGURATION

The following software components are required for developing, running, and testing the application:

Operating System: Windows, Linux, or macOS

Programming Language: Python

Web Technologies: HTML, CSS, JavaScript

Backend Framework: Flask (Python)

Browser: Google Chrome, Mozilla Firefox, Microsoft Edge

Code Editor: VS Code, PyCharm, or any text editor of choice

Python Libraries & Dependencies:

1. Flask (for backend API handling)
2. Pylint (for code analysis and error detection)
3. Jinja2 (for rendering templates)

2.3. NETWORK CONFIGURATION

The Script Analysis Web App is a lightweight application designed to function in both offline and online environments. Below are the network configurations required for deployment and usage:

Local Deployment (Offline Mode)

- The application runs on localhost using Flask’s built-in development server.

- Default access is via http://127.0.0.1:5000/ or http://localhost:5000/.
- No internet connection is required unless additional external libraries or updates are needed.
- Internet required only for render template [code editor textbox.

INSTALLATION

3. INSTALLATION

3.1Prerequisites

Before installing the Script Analysis Web App, ensure your system meets the following requirements:

Hardware Requirements:

- **Processor:** Intel i3 or higher
- **RAM:** 4GB (8GB recommended)
- **Storage:** At least 500MB of free space

Software Requirements:

- **Operating System:** Windows, Linux, or macOS
- **Python:** Version 3.x installed
- **Web Browser:** Chrome, Firefox, or Edge

Dependencies:

The application requires the following Python packages:

- Flask (for backend)
- Pylint (for code analysis)
- Jinja2 (for HTML templating)

3.2. Installation Steps

Follow these steps to set up the Script Analysis Web App on your system.

Step 1: Clone or Download the Project Using Bash or Terminal(cmd)

Download the project from the repository or clone it using **Git**:

```
git clone https://github.com/rocke7-adhi/script-analysis.git

cd rocke7-adhi-script-analysis
```

Step 2: Install Python and Virtual Environment (Optional)

Ensure Python 3.x is installed:

```
python --version
```

For a **virtual environment** setup (recommended to avoid dependency conflicts):

```
python -m venv venv

source venv/bin/activate   # On Linux/macOS

venv\Scripts\activate      # On Windows
```

Step 3: Install Dependencies :

```
pip install -r requirements.txt
```

3.3. Running the Application

Step 4: Start the Flask Server (Run the **Flask application** using:)

```
python app.py
```

Step 5: Access the Web App

- Open your **web browser** and go to :

```
http://127.0.0.1:5000
```

- You should see the **Get Started** page, where you can upload and analyze your scripts.

2.4. Troubleshooting & Common Issues :

ISSUE	**SOLUTION**
Flask not found	Ensure Flask is installed: pip install flask
Port 5000 in use	Change the port: python app.py --port=5001
CSS or JS not loading	Clear browser cache or check file paths in static/
Syntax errors	Use python -m pylint <your_script.py> to check errors

SOFTWARE DESCRIPTION

4. SOFTWARE DESCRIPTION

The **Script Analysis Web App** is developed using a combination of **frontend** and **backend** technologies to provide an interactive and efficient tool for analyzing code scripts.

4.1. Frontend Description (HTML, CSS, JavaScript)

The frontend of the application is responsible for providing an intuitive and interactive user interface. It allows users to upload script files, view analysis results, and interact with various features.

1. **Technologies Used:**
   - **HTML (HyperText Markup Language):**
     - Provides the structure of the web pages.
     - Defines elements such as file upload, buttons, and result display sections.
   - **CSS (Cascading Style Sheets):**
     - Enhances the visual design of the interface.
     - Ensures responsive layout and consistent styling across devices.
   - **JavaScript (JS):**
     - Enables dynamic functionalities, such as real-time updates and interactive elements.
     - Handles event-driven actions like file uploads and API requests to the backend.
2. **Frontend Features:**
   - **User-Friendly Interface:** Designed with a clean and responsive layout.
   - **File Upload & Processing:** Allows users to upload script files for analysis.
   - **Real-Time Updates:** Displays analysis results dynamically without refreshing the page.
   - **Syntax Highlighting:** Improves code readability using color-coded syntax.

4.2. Backend Description (Python & Flask)

The backend of the application is responsible for handling script processing, performing code analysis, and managing communication between the frontend and the backend.

1. **Technologies Used:**

    - **Python:** The core programming language used for script processing and analysis.
    - **Flask:** A lightweight web framework that manages HTTP requests, processes scripts, and returns analysis results.

2. **Backend Features:**

    - **Handles User Requests:** Accepts script files from the frontend for analysis.
    - **Processes Code Analysis:** Extracts key metrics such as line count, function count, and complexity.
    - **Performs Complexity Analysis:** Measures cyclomatic complexity and code maintainability.
    - **Returns Results to Frontend:** Sends processed data back to the user interface for display.

FEASIBILITY STUDY

5. FEASIBILITY STUDY

The **Feasibility Study** evaluates the viability of the **Script Analysis Web App** by analyzing its technical, operational, economic, and scheduling feasibility. This chapter ensures that the project is practical, cost-effective, and achievable within the given constraints.

1. Technical Feasibility

Technical feasibility assesses whether the available technology, tools, and infrastructure can support the successful development and deployment of the web application.

Key Factors:

- **Technology Stack:** The application uses **HTML, CSS, JavaScript (Frontend)** and **Python Flask (Backend)**, which are widely supported and well-documented technologies.
- **Code Analysis Tools:** The project leverages tools like **Pylint** to analyze code structure and complexity.
- **Cross-Platform Compatibility:** The web application runs on all major operating systems, including Windows, Linux, and macOS.
- **Hosting & Deployment:** Can be deployed on cloud platforms such as **pythonanywhare**, ensuring scalability.

2. Financial Feasibility

Economic feasibility evaluates whether the benefits of the project outweigh the costs of development, deployment, and maintenance.

Cost Considerations:

- **Development Costs:** No licensing fees required for open-source technologies (Flask, Python, JavaScript).
- **Hosting & Infrastructure:** Minimal costs if hosted on free-tier cloud platforms like Heroku or GitHub Pages.
- **Maintenance Costs:** Low, as the application does not require a complex database or third-party integrations.

3. Operational Feasibility

Operational feasibility examines whether the application meets user needs and is easy to use.

Key Factors:

- **User-Friendliness:** Simple interface for uploading and analyzing scripts.
- **Automation:** Eliminates manual code analysis, improving efficiency.
- **Multi-Language Support:** Supports various programming languages, increasing usability.
- **Real-Time Feedback:** Users receive instant results without waiting.

4. Market Feasibility

Market feasibility assesses whether there is demand for the **Script Analysis Web App** and if it has a competitive advantage.

Target Audience:

- **Software Developers** – Need tools to analyze and optimize code.
- **QA Engineers** – Require automated code analysis for testing.
- **Students & Educators** – Can use the tool for learning programming concepts.

Market Demand:

- **Growing need for automated code review tools** due to the rising complexity of software development.
- **Competitor analysis:** Existing tools like **SonarQube** and **Codacy** focus on enterprise use, while this tool provides a **lightweight, free, and accessible alternative**.

5. Legal and Ethical Feasibility

Legal and ethical feasibility examines whether the project complies with relevant regulations and follows ethical coding practices.

Legal Considerations:

- **Compliance with Open-Source Licenses:** Ensures that any external libraries (e.g., Flask, Pylint) comply with **MIT, Apache, or GPL licenses**.
- **Data Privacy & Security:** Since no database is used, **no sensitive user data is stored**, minimizing legal risks.
- **Copyright Protection:** The application must respect **intellectual property laws**, ensuring users only analyze their own code.

Ethical Considerations:

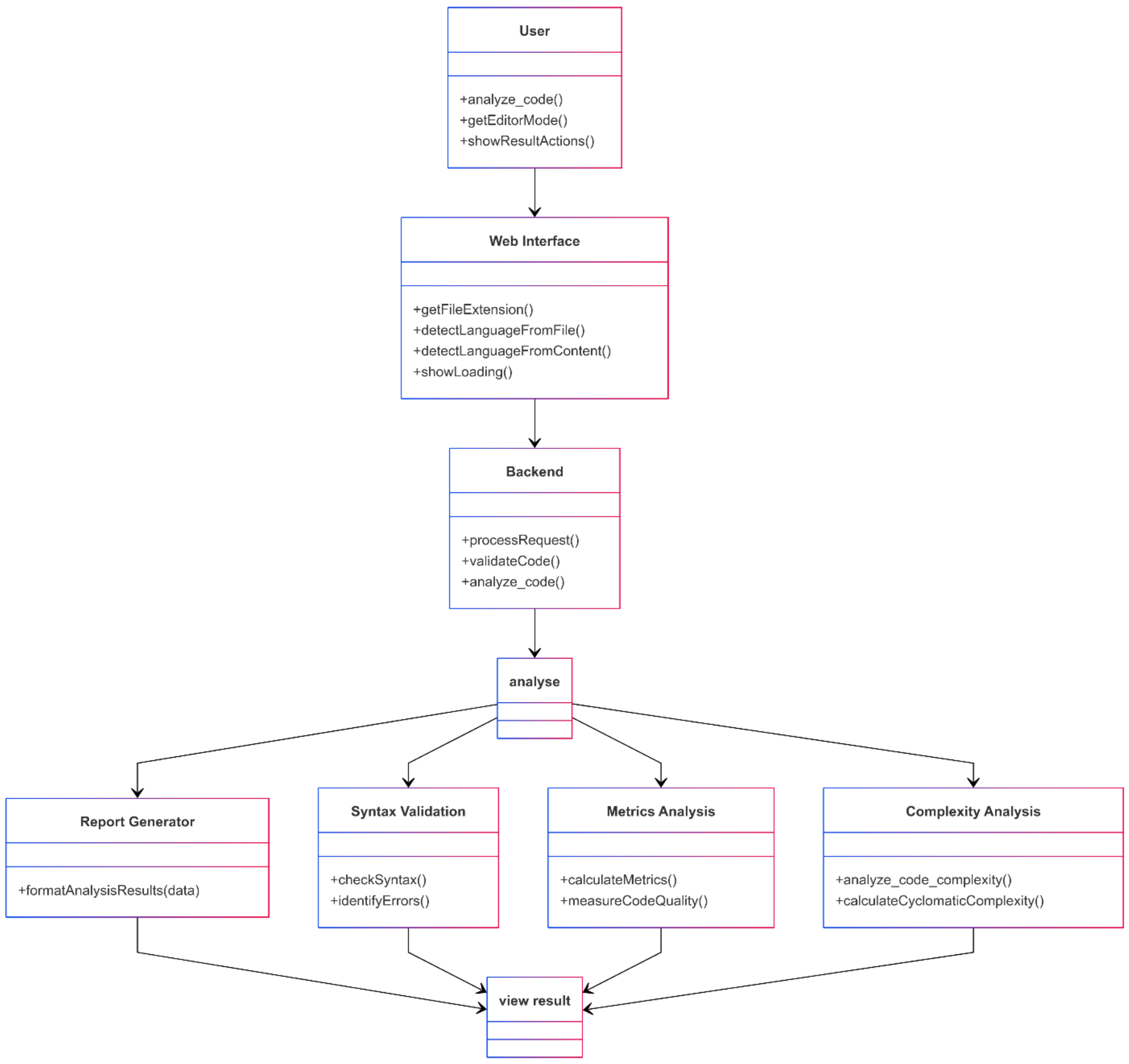
- **Fair Use of Code Analysis:** The tool should not be used to analyze proprietary code without permission.
- **Transparency:** Users should be informed about how the tool processes their scripts.

SYSTEM DESIGN

6. SYSTEM DESIGN

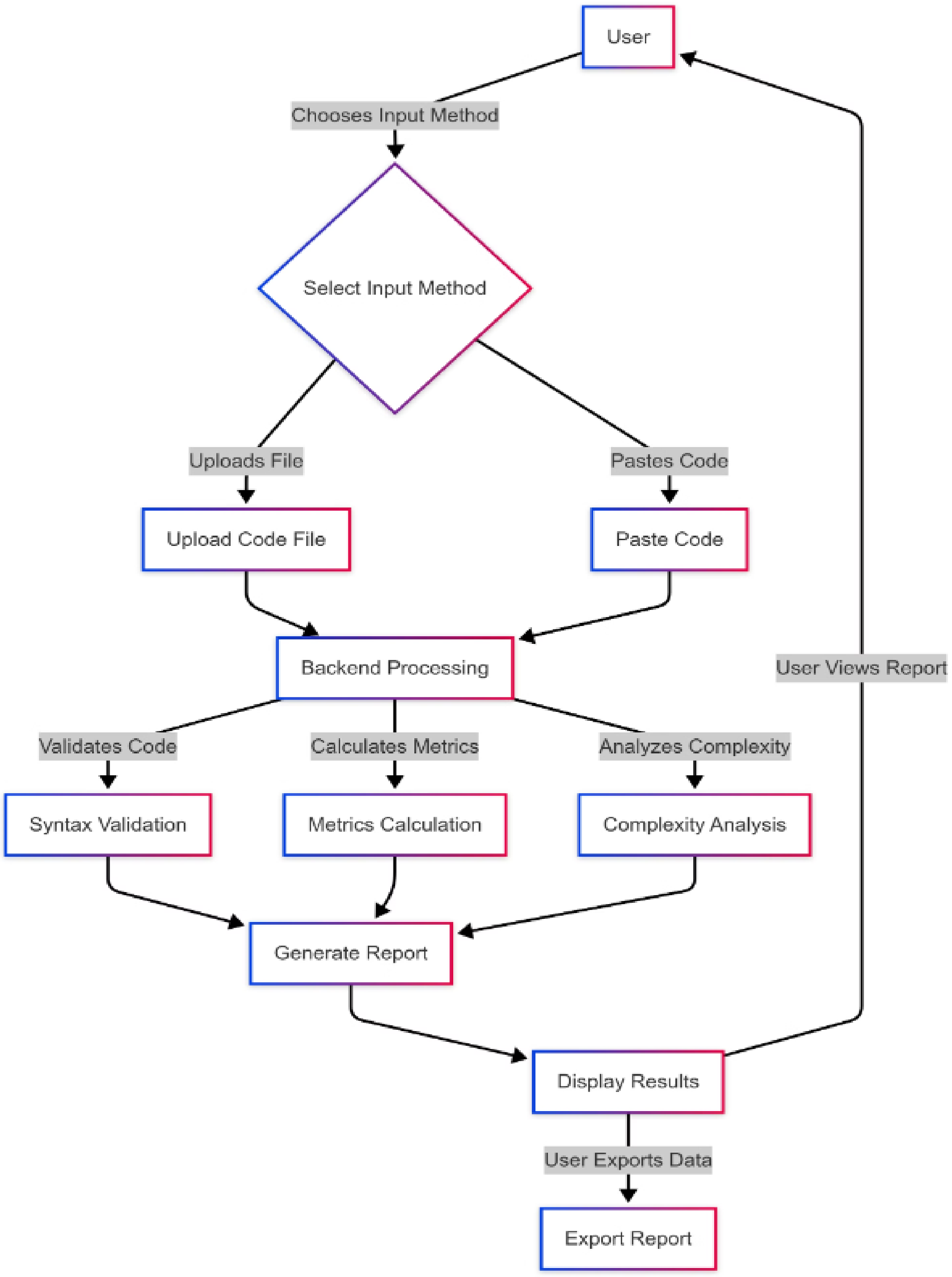
6.1. CLASS DIAGRAM

The class diagram illustrates the core components of the Script Analysis Web App and their interactions.It highlights key classes such as ScriptAnalyzer, CodeMetrics, ComplexityAnalyzer, ReportGenerator, FrontendController, and BackendController, showcasing their roles in handling code analysis, complexity measurement, and report generation.



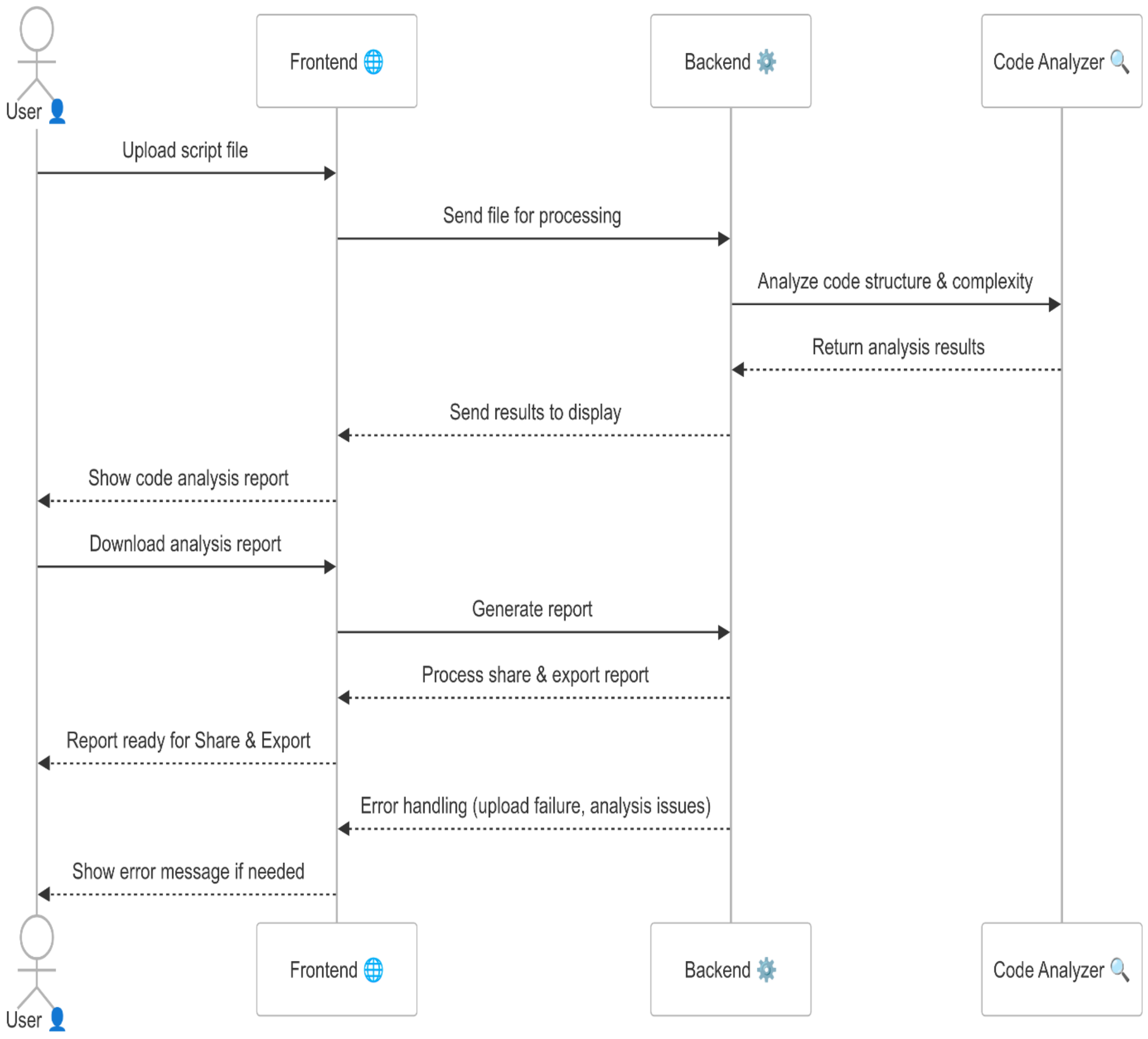
6.2. DATA FLOW DIAGRAM

The data flow diagram illustrates the movement of data through the Script Analysis Web App, depicting the process from file upload to result generation. It highlights key components such as the FlaskApp, CodeAnalyzer, ComplexityAnalyzer, and user interactions, ensuring a modular and efficient flow of data.



6.3. Sequence Diagram

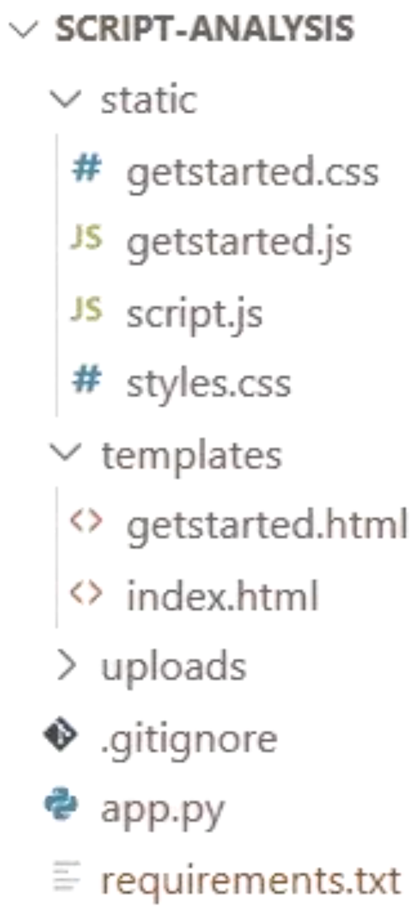
The sequence diagram shows the code analysis process in the Script Analysis Web App. The user uploads a code file, selects the language, and clicks "Analyze." The frontend sends the file and language to the backend via an HTTP request. The backend processes the code, extracts metrics like total lines, functions, classes, methods, objects, and cyclomatic complexity, then returns a JSON response. The frontend parses the JSON and displays the results, allowing the user to review, download, or share the analysis. This ensures smooth communication between components.



APPENDICES

7. APPENDICES

7.1. FOLDER STRUCTURE



```
SCRIPT-ANALYSIS
  static
    getstarted.css
    getstarted.js
    script.js
    styles.css
  templates
    getstarted.html
    index.html
  uploads
  .gitignore
  app.py
  requirements.txt
```

7.2. Coding

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Code Analysis Tool</title>
    <!-- Add CodeMirror CSS -->
    <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/codemirror/5.65.2/codemirror.min.css">
    <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/codemirror/5.65.2/theme/monokai.min.css">
    <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/codemirror/5.65.2/theme/dracula.min.css">
    <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/codemirror/5.65.2/theme/material.min.css">
    <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/codemirror/5.65.2/theme/nord.min.css">
    <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/codemirror/5.65.2/theme/github-dark.min.css">
    <link rel="stylesheet" href="{{ url_for('static', filename='styles.css') }}">
</head>
<body>
    <div class="loading-overlay">
        <div class="loading-spinner"></div>
        <div class="loading-text">Analyzing your code...</div>
        <div class="loading-progress">
            <div class="loading-progress-bar"></div>
        </div>
        <div class="loading-status">Please wait while we analyze your code</div>
    </div>
```

```
  <div class="container">
    <h1>Code Analysis Tool</h1>

    <form id="uploadForm">
      <div id="uploadSection">
        <div class="top-controls">
          <div class="language-select">
      <label for="language">Select Language:</label>
      <select id="language" name="language" required>
              <option value="auto">Auto Detect</option>
        <option value="python">Python</option>
        <option value="javascript">JavaScript</option>
        <option value="cpp">C++</option>
        <option value="java">Java</option>
        <option value="ruby">Ruby</option>
        <option value="go">Go</option>
        <option value="swift">Swift</option>
        <option value="php">PHP</option>
        <option value="csharp">C#</option>
      </select>
          </div>

          <div class="file-input-container">
            <div class="file-upload-area">
              <label for="file" class="file-upload-label">
                <svg width="16" height="16" viewBox="0 0 24 24" fill="none"
stroke="currentColor" stroke-width="2">
                  <path d="M21 15v4a2 2 0 0 1-2 2H5a2 2 0 0 1-2-2v-4"></path>
                  <polyline points="7 8 12 3 17 8"></polyline>
                  <line x1="12" y1="3" x2="12" y2="15"></line>
                </svg>
                <span>Choose a file or drag it here</span>
              </label>
              <input type="file" id="file" name="file"
accept=".txt,.java,.js,.py,.r,.c,.cpp,.cs,.jl,.go,.pl,.m,.kt,.php,.rs,.rb,.lua,.ts,.sas,.hx,.lisp,.f,.scala,.a
s,.asm,.clj,.coffee,.dart,.cbl,.groovy,.ex,.erl,.pas,.hs,.swift,.rkt,.ml,.scm,.cr,.elm,.fs,.tcl,.vb,.vala,.
adb,.m,.objc">
              <div class="file-info">
                <span class="file-name" id="file-name">No file chosen</span>
              </div>
            </div>
          </div>
          <div class="result-actions" style="display: none;">
            <div class="action-group">
              <button class="action-btn share-btn" title="Share Results">
                <svg width="16" height="16" viewBox="0 0 24 24" fill="none"
stroke="currentColor" stroke-width="2">
                  <circle cx="18" cy="5" r="3"></circle>
                  <circle cx="6" cy="12" r="3"></circle>
                  <circle cx="18" cy="19" r="3"></circle>
                  <line x1="8.59" y1="13.51" x2="15.42" y2="17.49"></line>
                  <line x1="15.41" y1="6.51" x2="8.59" y2="10.49"></line>
                </svg>
                Share
                <div class="dropdown-content">
                  <a href="#" id="emailShare">
                    <svg width="16" height="16" viewBox="0 0 24 24" fill="none"
stroke="currentColor" stroke-width="2">
                    </svg>
                    Email
                  </a>
                  <a href="#" id="whatsappShare">
                    <svg width="16" height="16" viewBox="0 0 24 24" fill="none"
stroke="currentColor" stroke-width="2">
```

```
                                        </svg>
                                        WhatsApp
                                    </a>
                                </div>
                            </button>
                            <button class="action-btn export-btn" title="Export Results">
                                <svg width="16" height="16" viewBox="0 0 24 24" fill="none"
stroke="currentColor" stroke-width="2">
                                </svg>
                                Export
                                <div class="dropdown-content">
                                    <a href="#" id="exportPDF">
                                        <svg width="16" height="16" viewBox="0 0 24 24" fill="none"
stroke="currentColor" stroke-width="2">
                                        </svg>
                                        Export as PDF
                                    </a>
                                    <a href="#" id="exportTXT">
                                        <svg width="16" height="16" viewBox="0 0 24 24" fill="none"
stroke="currentColor" stroke-width="2">
                                           <polyline points="14 2 14 8 20 8"></polyline>
                                            <line x1="16" y1="13" x2="8" y2="13"></line>
                                            <line x1="16" y1="17" x2="8" y2="17"></line>
                                        </svg>
                                        Export as Text
                                    </a>
                                    <a href="#" id="exportImage">
                                        <svg width="16" height="16" viewBox="0 0 24 24" fill="none"
stroke="currentColor" stroke-width="2">
                                            <rect x="3" y="3" width="18" height="18" rx="2" ry="2"></rect>
                                            <circle cx="8.5" cy="8.5" r="1.5"></circle>
                                            <polyline points="21 15 16 10 5 21"></polyline>
```

```
                              </svg>
                              Save as Image
                            </a>
                          </div>
                        </button>
                      </div>
                    </div>
                  </div>
                      <textarea id="codeEditor"></textarea>
                    </div>
                  </div>
                </div>
                <button type="submit" class="analyze-btn">Analyze Code</button>
              </form>
              <div id="results"></div>


    </div>
    <!-- Add CodeMirror JS -->
    <script
src="https://cdnjs.cloudflare.com/ajax/libs/codemirror/5.65.2/mode/clike/clike.min.js"></script
>
    <script
src="https://cdnjs.cloudflare.com/ajax/libs/codemirror/5.65.2/mode/ruby/ruby.min.js"></script
>
    <script
src="https://cdnjs.cloudflare.com/ajax/libs/codemirror/5.65.2/mode/go/go.min.js"></script>
    <script
src="https://cdnjs.cloudflare.com/ajax/libs/codemirror/5.65.2/mode/swift/swift.min.js"></script
>
    <script
src="https://cdnjs.cloudflare.com/ajax/libs/codemirror/5.65.2/mode/php/php.min.js"></script>
    <script src="{{ url_for('static', filename='script.js') }}"></script>
    <!-- Add these before your closing </body> tag -->
    <script src="https://cdnjs.cloudflare.com/ajax/libs/jspdf/2.5.1/jspdf.umd.min.js"></script>
    <script src="https://html2canvas.hertzen.com/dist/html2canvas.min.js"></script>
</body>
</html>
```

styles.css

```
body {
    font-family: Arial, sans-serif;
    margin: 0;
    padding: 20px;
    background-color: #f4f4f4;
    min-height: 100vh;
}

.container {
    max-width: 1200px;
    margin: 0 auto;
    padding: 20px;
    background-color: white;
    border-radius: 8px;
    box-shadow: 0 2px 4px rgba(0,0,0,0.1);
}

h1 {
    text-align: center;
    color: #333;
    margin-bottom: 30px;
}

h2 {
    color: #2c3e50;
    border-bottom: 2px solid #eee;
    padding-bottom: 10px;
    margin-top: 30px;
}

h3 {
    color: #34495e;
    margin: 10px 0;
}
```

```
form#uploadForm {
    margin: 20px 0;
    padding: 25px;
    border: 1px solid #ddd;
    border-radius: 8px;
    background-color: #f8f9fa;
    max-width: 1000px;
    margin-left: auto;
    margin-right: auto;
}

.form-group {
    margin-bottom: 20px;
}

label {
    display: block;
    margin: 15px 0 8px;
    color: #333;
    font-weight: bold;
}

select {
    width: 100%;
    max-width: 300px;
}
.upload-btn {
    display: inline-flex;
    align-items: center;
    gap: 6px;
    padding: 6px 10px;
    background-color: #4CAF50;
    color: white;
    border: none;
    border-radius: 4px;
    cursor: pointer;
    font-size: 13px;
    transition: background-color 0.3s;
    height: 28px;
    width: auto;
    min-width: 90px;
    flex-shrink: 0;
  }
.classes, .functions, .methods, .objects, .comments {
    padding: 20px;
    border-radius: 8px;
    margin: 15px 0;
    box-shadow: 0 1px 3px rgba(0,0,0,0.1);
  }

  .classes ul, .functions ul, .methods ul, .objects ul, .comments ul {
    list-style-type: none;
    padding-left: 0;
    max-height: 300px;
    overflow-y: auto;
    margin: 0;
  }

  .classes li, .functions li, .methods li, .objects li, .comments li {
    padding: 10px;
    margin: 5px 0;
    background-color: #fff;
    border-radius: 4px;
    font-family: monospace;
    font-size: 14px;
    border: 1px solid rgba(0,0,0,0.1);
  }
.classes { background-color: #e8f5e9; }
  .functions { background-color: #f3e5f5; }
  .methods { background-color: #e3f2fd; }
  .objects { background-color: #fff3e0; }
  .comments { background-color: #fce4ec; }

  .output {
    background-color: #f8f9fa;
    padding: 20px;
    border-radius: 8px;
    margin-top: 20px;
    overflow-x: auto;
  }
```

```
.error {
    color: #dc3545;
    padding: 15px;
    border: 1px solid #dc3545;
    border-radius: 4px;
    margin: 10px 0;
    background-color: #fff;
}

/* Responsive Design */
@media (max-width: 768px) {
    body {
        padding: 10px;
    }

    .container {
        padding: 15px;
    }

    .analysis-grid {
        grid-template-columns: 1fr;
    }

    h1 {
        font-size: 24px;
    }

    h2 {
        font-size: 20px;
    }
.action-btn:hover .dropdown-content,
.dropdown-content:hover {
    display: block;
    opacity: 1;
    transform: translateY(0);
}
.action-btn::after {
    content: '';
    position: absolute;
    height: 10px;
    left: 0;
    right: 0;
    bottom: -10px;
}

.dropdown-content a:hover {
    background-color: #f8f9fa;
    color: #000;
}

.dropdown-content a:first-child {
    border-radius: 4px 4px 0 0;
}

.dropdown-content svg {
    width: 16px;
    height: 16px;
    stroke: #666;
    transition: stroke 0.2s ease;
}

.dropdown-content a:hover svg {
    stroke: #333;
}

/* Mobile adjustments */
@media (max-width: 768px) {
    .action-group {
        flex-direction: column;
        width: 100%;
    }

    .action-btn {
        width: 100%;
        justify-content: center;
    }

    .dropdown-content {
        position: static;
        transform: none;
    }
    .action-btn::after {
        display: none;
    }
```

script.js

```
let editor;

document.addEventListener('DOMContentLoaded', () => {

    // Initialize CodeMirror
    editor = CodeMirror.fromTextArea(document.getElementById('codeEditor'), {
        mode: 'python',
        theme: 'monokai',
        lineNumbers: true,
        autoCloseBrackets: true,
        matchBrackets: true,
        indentUnit: 4,
        tabSize: 4,
        lineWrapping: true,
        height: 'auto'
    });

    // File upload elements
    const fileUploadArea = document.querySelector('.file-upload-area');
    const fileInfo = document.querySelector('.file-info');
    const fileName = document.getElementById('file-name');
    const fileInput = document.getElementById('file');
    const languageSelect = document.getElementById('language');

    // Handle file selection
    function handleFileSelect(file) {
        if (file) {
            fileName.textContent = file.name;
            fileInfo.classList.add('show');

            // Read and display file content in the editor
            const reader = new FileReader();
            reader.onload = function(e) {
                editor.setValue(e.target.result);
            };
            reader.readAsText(file);

            // Auto-detect language if needed
            if (languageSelect.value === 'auto') {
                const detectedLanguage = detectLanguageFromFile(file.name);
                if (detectedLanguage) {
                    languageSelect.value = detectedLanguage;
                    editor.setOption('mode', getEditorMode(detectedLanguage));
                }
            }
        } else {
            fileName.textContent = 'No file chosen';
            fileInfo.classList.remove('show');
            editor.setValue(''); // Clear editor if no file selected
        }
    }

    // File input change handler
    fileInput.addEventListener('change', (e) => {
        handleFileSelect(e.target.files[0]);
    });

    // Drag and drop handlers
    ['dragenter', 'dragover', 'dragleave', 'drop'].forEach(eventName => {
        fileUploadArea.addEventListener(eventName, preventDefaults, false);
    });

    function preventDefaults(e) {
        e.preventDefault();
        e.stopPropagation();
    }

    ['dragenter', 'dragover'].forEach(eventName => {
        fileUploadArea.addEventListener(eventName, () => fileUploadArea.classList.add('drag-
    over'), false);
    });

    ['dragleave', 'drop'].forEach(eventName => {
        fileUploadArea.addEventListener(eventName, () =>
fileUploadArea.classList.remove('drag-over'), false);
    });

    fileUploadArea.addEventListener('drop', (e) => {
        const dt = e.dataTransfer;
        const files = dt.files;
        fileInput.files = files;
        handleFileSelect(files[0]);
    });

    // Content-based language detection for pasted code
    editor.on('change', () => {
        const content = editor.getValue();
        if (content.length > 10 && languageSelect.value === 'auto') {
            const detectedLanguage = detectLanguageFromContent(content);
            if (detectedLanguage) {
                languageSelect.value = detectedLanguage;
                editor.setOption('mode', getEditorMode(detectedLanguage));
            }
        }
    });

    // Add loading state handling
    function showLoading() {

            const extension = getFileExtension(selectedLanguage);
            const blob = new Blob([codeContent], { type: 'text/plain' });
            const file = new File([blob], `code${extension}`, { type: 'text/plain' });

            formData.append('file', file);
            formData.append('language', selectedLanguage);

            const response = await fetch('/analyze', {
                method: 'POST',
                body: formData
            });

            const data = await response.json();

            document.querySelector('.loading-progress-bar').style.animation = 'progress 5s linear
forwards';
            await new Promise(resolve => setTimeout(resolve, 5000));

            if (data.error) {
                resultsDiv.innerHTML = `<div class="error fade-in">Error: ${data.error}</div>`;
                return;
            }


            resultsDiv.innerHTML = resultsHTML;
            resultsDiv.classList.add('fade-in');
            const showResultActions = () => {
                document.querySelector('.result-actions').style.display = 'block';
            };
            const actionButtons = document.querySelectorAll('.action-btn, .dropdown-content a');

            actionButtons.forEach(btn => {
                btn.addEventListener('click', (e) => {
                    e.preventDefault();
                    e.stopPropagation();
                    return false; // Prevent form submission
                });
            });

            // Share functionality
            document.getElementById('emailShare').addEventListener('click', (e) => {
                e.preventDefault();
                const subject = 'Code Analysis Results';
                const body = formatAnalysisResults(window.analysisData);
            {
                    // Try the mailto link first
                    window.location.href =
`mailto:?subject=${encodeURIComponent(subject)}&body=${encodeURIComponent(cleanBo
dy + truncationNote)}`;
                } catch (error) {
                    // Fallback: Copy to clipboard and show instructions
                    navigator.clipboard.writeText(body).then(() => {
                        alert('Analysis results copied to clipboard. Please paste into your email client.');
                    }).catch(() => {
                        // If clipboard fails, show content in a modal or alert
                        alert('Please copy the analysis results from the page and paste into your email
client.');
                    });
                }
            });

            document.getElementById('whatsappShare').addEventListener('click', (e) => {
                e.preventDefault();
                const text = formatAnalysisResults(data);
                window.open(`https://wa.me/?text=${encodeURIComponent(text)}`, '_blank');
            });

            // Export functionality
            document.getElementById('exportPDF').addEventListener('click', async (e) => {
                e.preventDefault();
                const { jsPDF } = window.jspdf;
                const doc = new jsPDF();
                doc.setFont('helvetica');
                doc.setFontSize(16):
                doc.text('Code Analysis Results', 20, 20);
                doc.setFontSize(12);
                const content = formatAnalysisResults(data);
                const lines = doc.splitTextToSize(content, 170);

                let y = 30;
                const pageHeight = doc.internal.pageSize.height;
                lines.forEach(line => {
                    if (y > pageHeight - 20) {
                        doc.addPage();
                        y = 20;
                    }
                    doc.text(line, 20, y);
                    y += 6;
                });

                doc.save('analysis-results.pdf');
            });
            document.getElementById('exportTXT').addEventListener('click', (e) => {
                e.preventDefault();
                const content = formatAnalysisResults(data);
                const blob = new Blob([content], { type: 'text/plain' });
                const url = window.URL.createObjectURL(blob);
                const a = document.createElement('a');
                a.href = url;
                a.download = 'analysis-results.txt';
                document.body.appendChild(a);
                a.click();
                window.URL.revokeObjectURL(url);
                document.body.removeChild(a);
            });

            document.getElementById('exportImage').addEventListener('click', (e) => {
                e.preventDefault();
                html2canvas(document.getElementById('results')).then(canvas => {
                    const link = document.createElement('a');
                    link.download = 'analysis-results.png';
                    link.href = canvas.toDataURL();
                    link.click();
                });
            });

            // Update the results display to show actions
            const originalResultsHTML = resultsDiv.innerHTML;
            resultsDiv.innerHTML = originalResultsHTML;
            showResultActions();
          } catch (error) {
            resultsDiv.innerHTML = `<div class="error fade-in">Error: ${error.message}</div>`;
          } finally {
            hideLoading();
            // Reset progress bar animation
            const progressBar = document.querySelector('.loading-progress-bar');
            progressBar.style.animation = 'none';
            progressBar.offsetHeight; // Trigger reflow
            progressBar.style.animation = '';
          }
        });

        // Format code button
        document.getElementById('formatCode').addEventListener('click', (e) => {
          e.preventDefault();
          e.stopPropagation();

          try {
            const mode = editor.getMode().name;
            let code = editor.getValue();

            if (!code.trim()) {
                    return; // Don't format empty code
            }

            // Basic formatting for different languages
            switch (mode) {
                    case 'javascript':
                        // Basic JS formatting
                        code = code.replace(/[{]/g, ' {\n    ')
                                 .replace(/[}]/g, '\n}\n')
                                 .replace(/;/g, ';\n')
                                 .replace(/\n\s*\n/g, '\n\n'); // Remove extra newlines
                        break;

                    case 'python':
                        // Basic Python formatting
                        code = code.replace(/:\s*/g, ':\n    ')
                                 .replace(/\n\s*\n/g, '\n\n')
                                 .replace(/([^:]);/g, '$1\n'); // Add newlines after statements
                        break;

                    default:
                        // Generic formatting for other languages
                        code = code.replace(/[{]/g, ' {\n    ')
                                 .replace(/[}]/g, '\n}\n')
                                 .replace(/;/g, ';\n')
                                 .replace(/\n\s*\n/g, '\n\n');
            }

            // Update editor content
            editor.setValue(code);

            // Auto indent all lines
            const totalLines = editor.lineCount();
            for (let i = 0; i < totalLines; i++) {
                    editor.indentLine(i);
            }

            // Refresh the editor to update the display
            editor.refresh();

        } catch (error) {
            console.error('Formatting failed:', error);
        }
});

document.getElementById('copyCode').addEventListener('click', () => {
    navigator.clipboard.writeText(editor.getValue())
        .then(() => {
            const btn = document.getElementById('copyCode');
            btn.innerHTML = '<svg>...</svg>Copied!';
            setTimeout(() => {
                btn.innerHTML = '<svg>...</svg>Copy';
            }, 2000);
        });
    });

    document.getElementById('clearCode').addEventListener('click', () => {
        if (confirm('Are you sure you want to clear the editor?')) {
            editor.setValue('');
        }
    });

    // Update keyboard shortcuts - remove save-related shortcuts
    editor.setOption('extraKeys', {
        'Ctrl-F': (cm) => {
            document.getElementById('formatCode').click();
        },
        'Ctrl-/': (cm) => {
            const selections = cm.getSelections();
            const mode = cm.getMode().name;
            const comment = mode === 'python' ? '#' : '//';

            const newSelections = selections.map(selection => {
                const lines = selection.split('\n');
                const commentedLines = lines.map(line => {
                    if (line.trimStart().startsWith(comment)) {
                        return line.replace(new RegExp(`^(\\s*)${comment}\\s?`), '$1');
                    }
                    return line.replace(/^(\s*)/, `$1${comment} `);
                });
                return commentedLines.join('\n');
            });

            cm.replaceSelections(newSelections);
        }
    });
    document.getElementById('themeSelect').addEventListener('change', (e) => {
        const theme = e.target.value;
        editor.setOption('theme', theme);
    });
    document.getElementById('fontSizeSelect').addEventListener('change', (e) => {
        const fontSize = e.target.value + 'px';
        document.querySelector('.CodeMirror').style.fontSize = fontSize;
    });
    window.addEventListener('load', () => {
        editor.setValue('');
        document.getElementById('language').value = 'auto';
        document.getElementById('themeSelect').value = 'monokai';
        document.getElementById('fontSizeSelect').value = '14';
        editor.setOption('theme', 'monokai');
        document.querySelector('.CodeMirror').style.fontSize = '14px';
    });

    // Indent code button
    document.getElementById('indentCode').addEventListener('click', (e) => {
        e.preventDefault(); // Prevent form submission
        const totalLines = editor.lineCount();
        editor.operation(() => {
            for (let i = 0; i < totalLines; i++) {
                editor.indentLine(i);
            }
        });
        editor.refresh(); // Refresh editor after indenting
    });

    // Add this to prevent form submission when clicking editor toolbar buttons
    document.querySelector('.editor-toolbar').addEventListener('click', (e) => {
        if (e.target.closest('button')) {
            e.preventDefault();
            e.stopPropagation();
        }
    });
    document.getElementById('language').addEventListener('change', (e) => {
        const selectedLanguage = e.target.value;
        if (selectedLanguage !== 'auto') {
            editor.setOption('mode', getEditorMode(selectedLanguage));
        }
    });
});
function getEditorMode(language) {
    const modeMap = {
        'python': {
            mode: 'python',
            mime: 'text/x-python',
            indentUnit: 4
        },
        'javascript': {
            mode: 'javascript',
            mime: 'text/javascript',
            indentUnit: 2
        },
        'cpp': {
            mode: 'clike',
            mime: 'text/x-c++src',
            indentUnit: 4
        },
        'java': {
            mode: 'clike',
            mime: 'text/x-java',
            indentUnit: 4
        },
        'ruby': {
            mode: 'ruby',
            mime: 'text/x-ruby',
            indentUnit: 2
        },
        'go': {
            mode: 'go',
            mime: 'text/x-go',
            indentUnit: 4
        },
        'swift': {
            mode: 'swift',
            mime: 'text/x-swift',
            indentUnit: 4
        },
        'php': {
            mode: 'php',
            mime: 'application/x-httpd-php',
            indentUnit: 4
        },
        'csharp': {
            mode: 'clike',
            mime: 'text/x-csharp',
            indentUnit: 4
        }
    };

    const config = modeMap[language] || { mode: 'text/plain', indentUnit: 4 };
    editor.setOption('mode', config.mime);
    editor.setOption('indentUnit', config.indentUnit);
    return config.mode;
}
function detectLanguageFromFile(filename) {
    const extensionMap = {
        '.py': 'python',
        '.js': 'javascript',
        '.cpp': 'cpp',
        '.hpp': 'cpp',
        '.h': 'cpp',
        '.java': 'java',
        '.rb': 'ruby',
        '.go': 'go',
        '.swift': 'swift',
        '.php': 'php',
        '.cs': 'csharp'
    };

    const ext = '.' + filename.split('.').pop().toLowerCase();
    return extensionMap[ext];
```

```
}

// Add this helper function to get file extension
function getFileExtension(language) {
    const extensionMap = {
        'python': '.py',
        'javascript': '.js',
        'cpp': '.cpp',
        'java': '.java',
        'ruby': '.rb',
        'go': '.go',
        'swift': '.swift',
        'php': '.php',
        'csharp': '.cs'
    };
    return extensionMap[language] || '.txt';
}

function detectLanguageFromContent(content) {
    const patterns = {
        python: {
            keywords: /\b(def|class|import|from|if|for|while|try|except|with|async|await)\b/,
            syntax: /:\s*$/m,
            imports: /^(?:from\s+\w+(?:\.\w+)*\s+import|\s*import\s+\w+)/m
        },
        javascript: {
            keywords: /\b(function|const|let|var|if|for|while|try|catch|class|import|export)\b/,
            syntax: /[{};]/,
            imports: /^(?:import\s+.*\s+from\s+[\'"].*[\'"]|require\s*\([\'"].*[\'"]\))/m
        },
        cpp: {
            keywords: /\b(class|struct|namespace|template|public|private|protected)\b/,
            syntax: /::|->|<>/,
            includes: /#include\s*[<"]/
        },
        java: {
            keywords: /\b(public|private|protected|class|interface|extends|implements|package)\b/,
            syntax: /;$/m,
            imports: /^import\s+[\w.]+(?:\s*\*)?;/m
        },
        ruby: {
            keywords: /\b(def|class|module|require|include|attr_accessor)\b/,
            syntax: /end$/m,
            requires: /^require\s+[\'"].*[\'"]/
        },
        go: {
            keywords: /\b(func|type|struct|interface|package|import|go|chan|defer)\b/,
            syntax: /:\=|<-/,
            imports: /^import\s+(?:\([^)]+\)|"[^"]+")/m
        },
        swift: {
            keywords: /\b(class|struct|enum|protocol|extension|guard|let|var)\b/,
            syntax: /->|@/,
            imports: /^import\s+\w+/m
        },
        php: {
            keywords: /\b(function|class|namespace|use|public|private|protected)\b/,
            syntax: /\$\w+|<?php/,
            imports: /^(?:require|include)(?:_once)?\s*\([\'"].*[\'"]\)/m
        },
        csharp: {
            keywords: /\b(class|namespace|using|public|private|protected|async|await)\b/,
            syntax: /;$/m,
            imports: /^using\s+[\w.]+;/m
        }
    };
    const scores = Object.entries(patterns).map(([lang, pattern]) => {
        let score = 0;
        const contentSample = content.slice(0, 1000); // Check first 1000 chars

        if (pattern.keywords.test(contentSample)) score += 2;
        if (pattern.syntax.test(contentSample)) score += 1;
        if (pattern.imports?.test(contentSample)) score += 3;

        return { language: lang, score };
    });
    const bestMatch = scores.reduce((max, curr) =>
        curr.score > max.score ? curr : max
    );

    return bestMatch.score > 2 ? bestMatch.language : null;
}
function formatAnalysisResults(data) {
    if (!data) return 'No analysis data available.';

    try {
        return `
Code Analysis Results
--------------------

BASIC METRICS
Total Lines: ${data.total_lines}
Empty Lines: ${data.empty_lines}
Comment Lines: ${data.comment_lines}
Import Count: ${data.import_count}

CODE STRUCTURE
Classes: ${data.code_structure.classes}
Functions: ${data.code_structure.functions}
Methods: ${data.code_structure.methods}
```

```
Objects: ${data.code_structure.objects}
Imports: ${data.code_structure.imports}

${data.classes.length > 0 ? `\nCLASSES FOUND\n${data.classes.map(cls => `-
${cls}`).join('\n')}\n` : ''}
${data.functions.length > 0 ? `\nFUNCTIONS FOUND\n${data.functions.map(func => `-
${func}`).join('\n')}\n` : ''}
${data.methods.length > 0 ? `\nMETHODS FOUND\n${data.methods.map(method => `-
${method}`).join('\n')}\n` : ''}
${data.objects.length > 0 ? `\nOBJECTS FOUND\n${data.objects.map(obj => `-
${obj}`).join('\n')}\n` : ''}
${data.comments.length > 0 ? `\nCOMMENTS FOUND\n${data.comments.map(comment =>
`- ${comment}`).join('\n')}\n` : ''}

ANALYSIS OUTPUT
${data.output || 'No issues found'}`;
  } catch (error) {
    console.error('Error formatting results:', error);
    return 'Error formatting analysis results. Please try exporting as PDF or TXT instead.';
  }
}
```

getstarted.html

```
 <!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Get Started - Code Analysis Tool</title>
  <link rel="stylesheet" href="{{ url_for('static', filename='getstarted.css') }}">
</head>
<body>
  <nav class="navbar">
    <div class="nav-container">
      <a href="/" class="logo">Code Analysis Tool</a>
      <div class="nav-links">
        <!-- <a href="/" class="nav-link">Home</a> -->
        <a href="/getstarted" class="nav-link active">Get Started</a>
      </div>
    </div>
  </nav>

  <div class="hero-section">
    <h1>Welcome to Code Analysis Tool</h1>
    <p>Analyze your code with powerful metrics and insights</p>
  </div>

  <div class="container">
    <div class="steps-section">
      <h2>How to Use the Tool</h2>
      <div class="step-card">
        <div class="step-number">1</div>
        <div class="step-content">
          <h3>Select Your Programming Language</h3>
          <p>Choose from multiple supported languages or use auto-detect:</p>
          <ul>
            <li>Python</li>
            <li>JavaScript</li>
            <li>Java</li>
            <li>C++</li>
            <li>And more...</li>
          </ul>
        </div>
      </div>

      <div class="step-card">
        <div class="step-number">2</div>
        <div class="step-content">
          <h3>Upload Your Code</h3>
          <p>Two ways to input your code:</p>
          <ul>
            <li>Upload a file directly</li>
            <li>Copy and paste your code into the editor</li>
          </ul>
          <div class="tip">Tip: You can drag and drop files directly into the upload
area</div>
        </div>
      </div>

      <div class="step-card">
        <div class="step-number">3</div>
        <div class="step-content">
          <h3>Analyze Your Code</h3>
          <p>Get comprehensive analysis including:</p>
          <ul>
            <li>Code Metrics</li>
            <li>Complexity Analysis</li>
            <li>Structure Overview</li>
          </ul>
        </div>
      </div>

      <div class="step-card">
        <div class="step-number">4</div>
        <div class="step-content">
          <h3>Review Results</h3>
          <p>Understand your code better with:</p>
          <ul>
            <li>Visual Metrics</li>
```

```
                <li>Detailed Reports</li>
                <li>Export Options</li>
            </ul>
        </div>
    </div>
</div>
<div class="features-grid">
    <div class="feature-card">
        <div class="feature-icon"></div>
        <h3>Basic Metrics</h3>
        <p>Analyze code structure, line counts, and basic statistics</p>
        <button class="learn-more" data-section="basic-metrics">Learn More</button>
    </div>
</div>
    <section id="code-structure" class="info-section">
        <h2>Code Structure Analysis</h2>
        <ul>
            <li>Class Definitions</li>
            <li>Function Declarations</li>
            <li>Method Implementations</li>
            <li>Object Instantiations</li>
        </ul>
        <div class="example-code">
            <pre><code>
    # Class Definition
    class Animal:
    # Method Implementation
        def __init__(self, name):
            self.name = name
    # Method Implementation
        def speak(self):
            return "Some sound"
    # Function Declaration
    def greet():
        print("Hello!")

    dog = Animal("Buddy")  # Object Instantiation
    print(dog.speak())  # Calling a method
    greet()  # Calling a function
            </code></pre>
        </div>
    </section>

    <section id="complexity" class="info-section">
        <h2>Complexity Analysis</h2>
        <ul>
            <li>Cyclomatic Complexity</li>
            <li>Maximum Nesting Depth</li>
            <li>Maintainability Index</li>
        </ul>
```

```
                <div class="example-code">
                    <pre><code>def complex_function(x):
    if x > 0:
        for i in range(x):
            if i % 2 == 0:
                print("Even")
            else:
                print("Odd")</code></pre>
                </div>
            </section>
        </div>

        <div class="cta-section">
s            <h2>Ready to Analyze Your Code?</h2>
            <p>Get started with our powerful code analysis tool</p>
            <a href="/" class="cta-button">Start Analyzing</a>
        </div>
    </div>

    <footer>
        <div class="footer-content">
            <p>&copy; 2024 Code Analysis Tool. All rights reserved.</p>
        </div>
    </footer>

    <script src="{{ url_for('static', filename='getstarted.js') }}"></script>
</body>
</html>
```

getstarted.css

```
:root {
    --primary-color: #2196F3;
    --secondary-color: #1976D2;
    --text-color: #333;
    --bg-color: #f4f4f4;
    --card-bg: #ffffff;
}

body {
    margin: 0;
    padding: 0;
    font-family: 'Arial', sans-serif;
    line-height: 1.6;
    color: var(--text-color);
    background-color: var(--bg-color);
}

.navbar {
    background-color: var(--card-bg);
    box-shadow: 0 2px 4px rgba(0,0,0,0.1);
    position: fixed;
    width: 100%;
    top: 0;
    z-index: 1000;
}


.step-content li {
    margin: 0.5rem 0;
    padding-left: 1.5rem;
    position: relative;
}

.step-content li::before {
    content: "→";
    position: absolute;
    left: 0;
    color: #2196F3;
}

.tip {
    background: #e3f2fd;
    padding: 1rem;
    border-radius: 4px;
    margin-top: 1rem;
    color: #1976D2;
}
body.dark-theme .step-card {

    background: #2d2d2d;
}

body.dark-theme .tip {
    background: #1a1a1a;
    color: #64B5F6;
}

body.dark-theme .step-content h3 {
    color: #64B5F6;
}

body.dark-theme .step-content li::before {
    color: #64B5F6;
}
```

getstarted.js

```
document.addEventListener('DOMContentLoaded', () => {
   const learnMoreButtons = document.querySelectorAll('.learn-more');

   const infoSections = document.querySelectorAll('.info-section');

   learnMoreButtons.forEach(button => {
      button.addEventListener('click', () => {

         const sectionId = button.getAttribute('data-section');
                  infoSections.forEach(section => {
            section.classList.remove('active');
         });

         // Show the selected section
         const sectionToShow = document.getElementById(sectionId);
         if (sectionToShow) {
            sectionToShow.classList.add('active');
            sectionToShow.scrollIntoView({
               behavior: 'smooth',
               block: 'start'
            });
         }
      });
   });

   // Add scroll reveal animation for feature cards
   const featureCards = document.querySelectorAll('.feature-card');

   const observerOptions = {
      threshold: 0.1,
      rootMargin: '0px 0px -50px 0px'
   };

   const observer = new IntersectionObserver((entries) => {
      entries.forEach(entry => {
         if (entry.isIntersecting) {
            entry.target.style.opacity = '1';
            entry.target.style.transform = 'translateY(0)';
         }
      });
   }, observerOptions);

   featureCards.forEach(card => {
      card.style.opacity = '0';
      card.style.transform = 'translateY(20px)';
      card.style.transition = 'all 0.5s ease';
      observer.observe(card);
   });
});
```

app.py

```
from flask import Flask, request, jsonify, render_template, session, redirect,
url_for
import os
import subprocess
import re
import ast
from radon.complexity import cc_visit
from radon.metrics import mi_visit
from radon.raw import analyze
import secrets

app = Flask(__name__)
app.secret_key = secrets.token_hex(16)  # Required for session management

# Function to analyze code based on file type
def analyze_code(file_path, language):
    try:
        # Read file contents first
        with open(file_path, "r", encoding="utf-8") as f:
            lines = f.readlines()
            content = ''.join(lines)
            total_lines = len(lines)

        # Common patterns for object-oriented features
        method_pattern = r'\s*def\s+\w+\s*\(.*\)'
        object_pattern = r'\w+\s*=\s*\w+\('

        # Set up language-specific patterns and try analysis
        if language == "python":
            comment_symbol = "#"
            class_pattern = r'class\s+\w+'
            function_pattern = r'def\s+\w+'
            method_pattern = r'\s+def\s+\w+\s*\(.*\)'
            object_pattern = r'\w+\s*=\s*\w+\('
            import_pattern =
r'^(?:from\s+\w+(?:\.\w+)*\s+import|\s*import\s+\w+(?:\s*,\s*\w+)*)'
            try:
                result = subprocess.run(["pylint", file_path], capture_output=True,
text=True)
                analysis_output = result.stdout or "No issues found"
            except FileNotFoundError:
                analysis_output = "Pylint not installed. Please install pylint for detailed
Python analysis."

        elif language == "javascript":
            comment_symbol = "//"
            class_pattern = r'class\s+\w+'
            function_pattern = r'function\s+\w+'
            method_pattern = r'(async\s+)?[\w.]+\s*\(.*\)\s*{'
            object_pattern = r'(const|let|var)\s+\w+\s*=\s*new\s+\w+'
            import_pattern =
r'^(?:import\s+.*\s+from\s+[\'"].*[\'"]|require\s*\([\'"].*[\'"]\))'
            try:
                result = subprocess.run(["jshint", file_path], capture_output=True,
text=True)
                analysis_output = result.stdout or "No issues found"
            except FileNotFoundError:
                analysis_output = "JSHint not installed. Please install jshint for detailed
JavaScript analysis."

        elif language == "cpp":
            comment_symbol = "//"
            class_pattern = r'class\s+\w+'
            function_pattern = r'\w+\s+\w+\(.*\)'
            import_pattern = r'#include\s*[<"].*[>"]'
            try:
                result = subprocess.run(["cppcheck", file_path], capture_output=True,
text=True)
                analysis_output = result.stdout or "No issues found"
            except FileNotFoundError:
                analysis_output = "Cppcheck not installed. Please install cppcheck for
detailed C++ analysis."

        elif language == "java":
            comment_symbol = "//"
            class_pattern = r'class\s+\w+'
            function_pattern = r'\w+\s+\w+\(.*\)'
            method_pattern = r'(public|private|protected)?\s+\w+\s+\w+\s*\(.*\)'
            object_pattern = r'\w+\s+\w+\s*=\s*new\s+\w+'
            import_pattern = r'import\s+[\w.]+(?:\s*\*)?;'
            try:
                result = subprocess.run(["javac", file_path], capture_output=True,
text=True)
                analysis_output = result.stdout or "No issues found"
            except FileNotFoundError:
                analysis_output = "Java compiler not found. Please install JDK for Java
analysis."
                result = subprocess.run(["ruby", "-wc", file_path],
capture_output=True, text=True)
                analysis_output = result.stdout or "No issues found"
            except FileNotFoundError:
                analysis_output = "Ruby not installed. Please install Ruby for detailed
analysis."
                result = subprocess.run(["go", "vet", file_path], capture_output=True,
text=True)
                analysis_output = result.stdout or "No issues found"
            except FileNotFoundError:
                analysis_output = "Go not installed. Please install Go for detailed
analysis."

        elif language == "swift":
            comment_symbol = "//"
            class_pattern = r'class\s+\w+'
            function_pattern = r'func\s+\w+'
            import_pattern = r'import\s+\w+'
            try:
                result = subprocess.run(["swiftc", file_path], capture_output=True,
text=True)
                analysis_output = result.stdout or "No issues found"
            except FileNotFoundError:
                analysis_output = "Swift compiler not found. Please install Swift for
detailed analysis."

        elif language == "php":
            comment_symbol = "//"
            class_pattern = r'class\s+\w+'
            function_pattern = r'function\s+\w+'
            import_pattern = r'(?:require|include)(?:_once)?\s*\([\'"].*[\'"]\)'
            try:
                result = subprocess.run(["php", "-l", file_path], capture_output=True,
text=True)
                analysis_output = result.stdout or "No issues found"
            except FileNotFoundError:
                analysis_output = "PHP not installed. Please install PHP for detailed
analysis."

        elif language == "csharp":
            comment_symbol = "//"
            class_pattern = r'class\s+\w+'
            function_pattern = r'\w+\s+\w+\(.*\)'
            import_pattern = r'using\s+[\w.]+;'
            try:
                result = subprocess.run(["csc", file_path], capture_output=True,
text=True)
                analysis_output = result.stdout or "No issues found"
            except FileNotFoundError:
                analysis_output = "C# compiler not found. Please install .NET SDK for
C# analysis."

        else:
            return {"error": "Language not supported yet"}

        # Enhanced Analysis
        empty_lines = sum(1 for line in lines if line.strip() == "")
        comment_lines = sum(1 for line in lines if
line.strip().startswith(comment_symbol))
        class_count = sum(1 for line in lines if re.search(class_pattern, line))
        function_count = sum(1 for line in lines if re.search(function_pattern, line))
        method_count = sum(1 for line in lines if re.search(method_pattern, line))
        object_count = sum(1 for line in lines if re.search(object_pattern, line))
        import_count = sum(1 for line in lines if re.search(import_pattern, line))

        # Find methods, classes, functions and comments
        methods = []
        objects = []
        classes = []
        functions = []
        comments = []
        imports = []

        for line in lines:
            # Extract method names
            method_match = re.search(method_pattern, line)
            if method_match:
                method_name = method_match.group().strip()
                methods.append(method_name)

            # Extract class names
            class_match = re.search(class_pattern, line)
            if class_match:
```

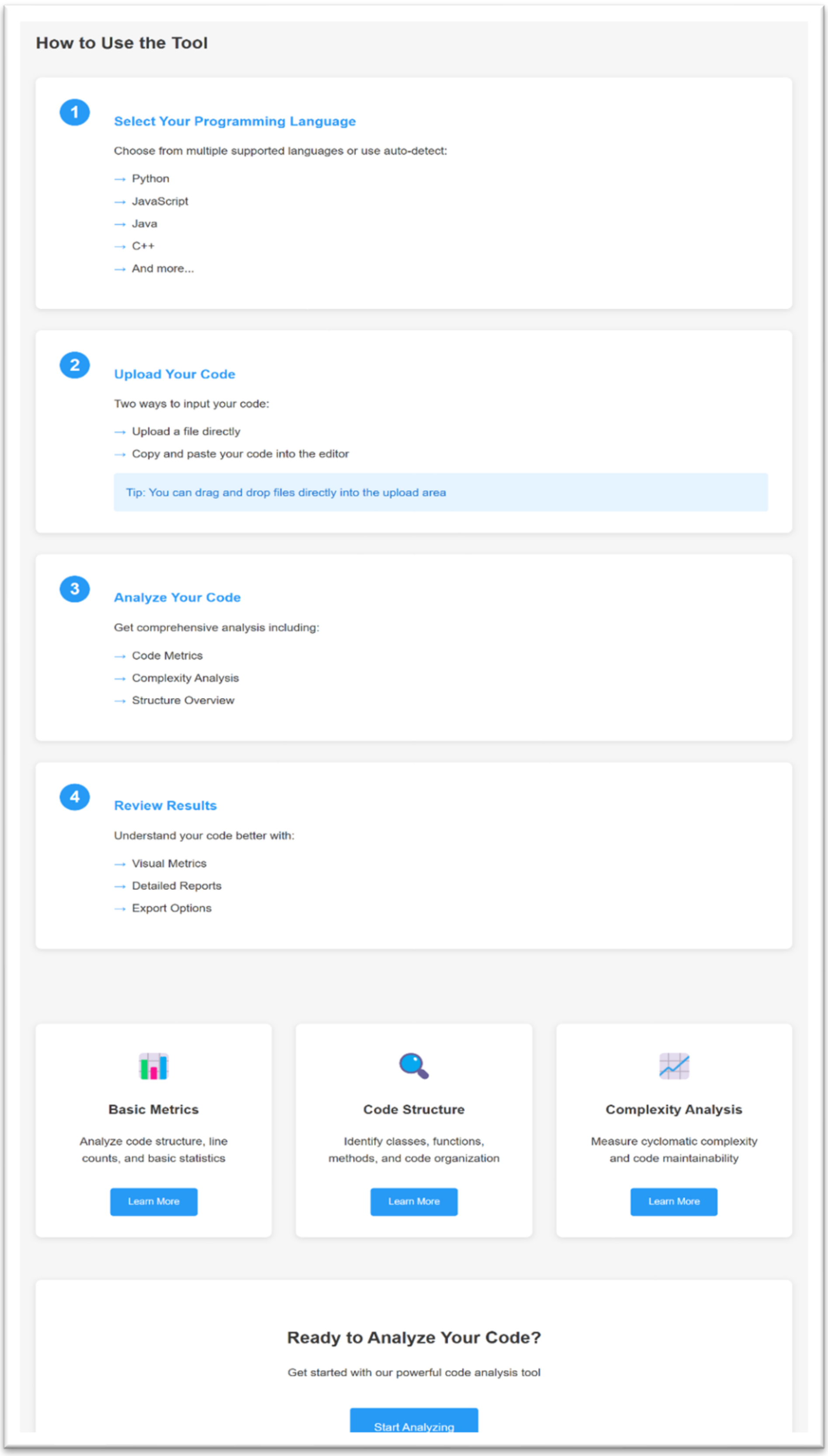
```
            class_name = class_match.group().strip()
            classes.append(class_name)

        # Extract function names
        function_match = re.search(function_pattern, line)
        if function_match:
            function_name = function_match.group().strip()
            functions.append(function_name)

        if line.strip().startswith(comment_symbol):
            comment_text = line.strip()
            comments.append(comment_text)

        object_match = re.search(object_pattern, line)
        if object_match:
            object_name = object_match.group().strip()
            objects.append(object_name)

          import_match = re.search(import_pattern, line)
        if import_match:
            import_statement = import_match.group().strip()
            imports.append(import_statement)

    complexity_metrics = analyze_code_complexity(content, language)
    result = {
        "output": analysis_output,
        "total_lines": total_lines,
        "empty_lines": empty_lines,
        "comment_lines": comment_lines,
        "class_count": class_count,
        "function_count": function_count,
        "method_count": method_count,
        "object_count": object_count,
        "import_count": import_count,
        "methods": methods,
        "objects": objects,
        "classes": classes,
        "functions": functions,
        "comments": comments,
        "imports": imports,
        "code_structure": {
            "classes": class_count,
            "functions": function_count,
                "methods": method_count,
                "objects": object_count,
                "imports": import_count
            },
            "complexity_analysis": {
                "cyclomatic_complexity":
complexity_metrics['cyclomatic_complexity'],
                "max_nesting_depth": complexity_metrics['max_nesting_depth'],
                "maintainability_index": complexity_metrics['maintainability_index'],
            }

        }

        return result
    except Exception as e:
        return {"error": str(e)}
def allowed_file(filename, language):
    extensions = {
        'python': ['.py'],
        'javascript': ['.js'],
        'cpp': ['.cpp', '.hpp', '.h'],
        'java': ['.java'],
        'ruby': ['.rb'],
        'go': ['.go'],
        'swift': ['.swift'],
        'php': ['.php'],
        'csharp': ['.cs']
    }
    return '.' in filename and \
           filename.rsplit('.', 1)[1].lower() in [ext[1:] for ext in
extensions.get(language, [])]

# Add this function for auto-detecting language
def detect_language(filename):
    extension_map = {
        '.py': 'python',
        '.js': 'javascript',
        '.cpp': 'cpp',
        '.hpp': 'cpp',
        '.h': 'cpp',
        '.java': 'java',
        '.rb': 'ruby',
        '.go': 'go',
        '.swift': 'swift',
        '.php': 'php',
        '.cs': 'csharp'
    }
    if '.' in filename:
        ext = '.' + filename.rsplit('.', 1)[1].lower()
        return extension_map.get(ext)
    return None
@app.route('/')
def index():
    """Route handler for the home page"""
    # Check if this is the user's first visit
    if not session.get('has_visited'):
        # First visit - mark as visited and redirect to getstarted
        session['has_visited'] = True
        return redirect(url_for('getstarted'))
    # Returning visitor - show the main page
    return render_template('index.html')

@app.route('/getstarted')
def getstarted():
    """Route handler for the get started page"""
    return render_template('getstarted.html')

@app.route('/reset-session')
def reset_session():
    """Helper route to reset the session (for testing)"""
    session.clear()
    return redirect(url_for('index'))

def analyze_code_complexity(content, language):
    """Analyze code complexity metrics"""
    try:
        # Basic complexity metrics
        complexity_metrics = {
            'cyclomatic_complexity': 0,
            'max_nesting_depth': 0,
            'maintainability_index': 0,

        }

        if language == "python":
            # Use radon for Python complexity analysis
            results = cc_visit(content)
            complexity_metrics['cyclomatic_complexity'] = sum(result.complexity for
result in results)
            complexity_metrics['maintainability_index'] = mi_visit(content,
multi=True)

            # Calculate max nesting depth
            tree = ast.parse(content)
            complexity_metrics['max_nesting_depth'] = get_max_nesting_depth(tree)

        else:
            # Basic complexity analysis for other languages
            lines = content.split('\n')
            current_depth = 0
            max_depth = 0
                 nesting_patterns = {
                'javascript': ['{', '}'],
                'java': ['{', '}'],
                'cpp': ['{', '}'],
                'csharp': ['{', '}'],
                'php': ['{', '}'],
                'ruby': ['do|{|begin|if|unless|case', 'end|},'],
                'swift': ['{', '}'],
                'go': ['{', '}'],
            }

            open_pattern, close_pattern = nesting_patterns.get(language, ['{', '}'])

            for line in lines:
                line = line.strip()
                if re.search(f'.*{open_pattern}.*', line):
                    current_depth += 1
                    max_depth = max(max_depth, current_depth)
                if re.search(f'.*{close_pattern}.*', line):
                    current_depth = max(0, current_depth - 1)

            complexity_metrics['max_nesting_depth'] = max_depth

            decision_patterns = [
                r'\bif\b', r'\bwhile\b', r'\bfor\b', r'\bforeach\b',
                r'\bcase\b', r'\bcatch\b', r'\b\|\|\b', r'\b&&\b'
            ]
            complexity = 1  # Base complexity
            for pattern in decision_patterns:
                complexity += len(re.findall(pattern, content))

            complexity_metrics['cyclomatic_complexity'] = complexity
                        operators = len(re.findall(r'[+\-
*/=<>!&|^~%]|\b(if|else|while|for|return)\b', content))
            operands = len(re.findall(r'\b[a-zA-Z_]\w*\b', content))
            unique_operators = len(set(re.findall(r'[+\-
*/=<>!&|^~%]|\b(if|else|while|for|return)\b', content)))
            unique_operands = len(set(re.findall(r'\b[a-zA-Z_]\w*\b', content)))

            if unique_operands != 0:
                difficulty = (unique_operators / 2) * (operands / unique_operands)
                complexity_metrics['difficulty_score'] = round(difficulty, 2)

        return complexity_metrics

def get_max_nesting_depth(node, current_depth=0):
    """Calculate maximum nesting depth for Python AST"""
    max_depth = current_depth
    for child in ast.iter_child_nodes(node):
        if isinstance(child, (ast.If, ast.For, ast.While, ast.Try, ast.With)):
            child_depth = get_max_nesting_depth(child, current_depth + 1)
            max_depth = max(max_depth, child_depth)
        else:
            child_depth = get_max_nesting_depth(child, current_depth)
            max_depth = max(max_depth, child_depth)
    return max_depth

if __name__ == "__main__":
    os.makedirs("uploads", exist_ok=True)
    app.run(debug=True)
```

requirements.txt

```
Flask==3.0.0
radon==6.0.1
pylint==3.0.3
Werkzeug==3.0.1
jinja2==3.1.2
```

7.3. SCREENSHOTS

Get-started page



How to Use the Tool

1 Select Your Programming Language

Choose from multiple supported languages or use auto-detect:

→ Python
→ JavaScript
→ Java
→ C++
→ And more...

2 Upload Your Code

Two ways to input your code:

→ Upload a file directly
→ Copy and paste your code into the editor

Tip: You can drag and drop files directly into the upload area

3 Analyze Your Code

Get comprehensive analysis including:

→ Code Metrics
→ Complexity Analysis
→ Structure Overview

4 Review Results

Understand your code better with:

→ Visual Metrics
→ Detailed Reports
→ Export Options

Basic Metrics

Analyze code structure, line counts, and basic statistics

Code Structure

Identify classes, functions, methods, and code organization

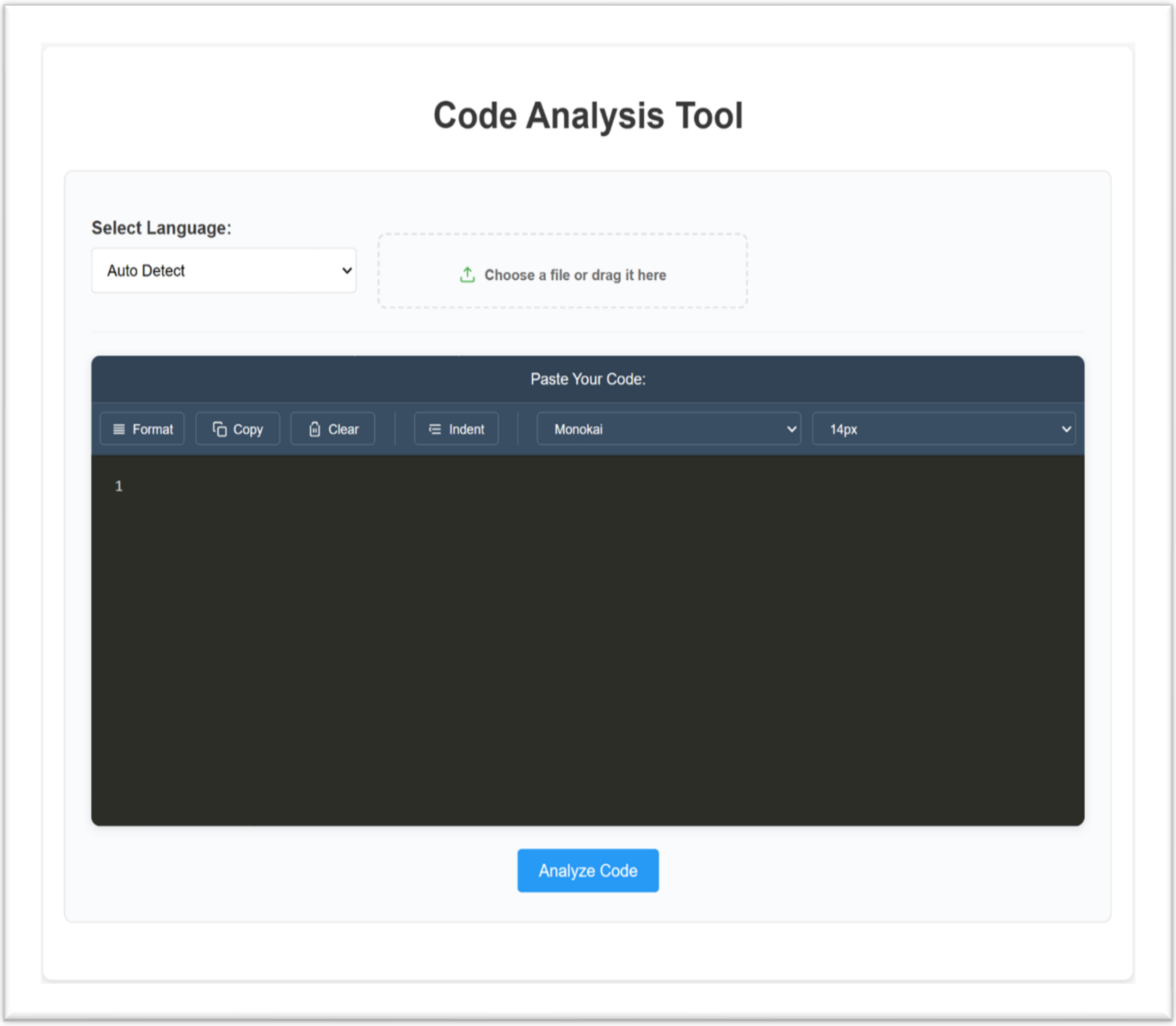
Complexity Analysis

Measure cyclomatic complexity and code maintainability

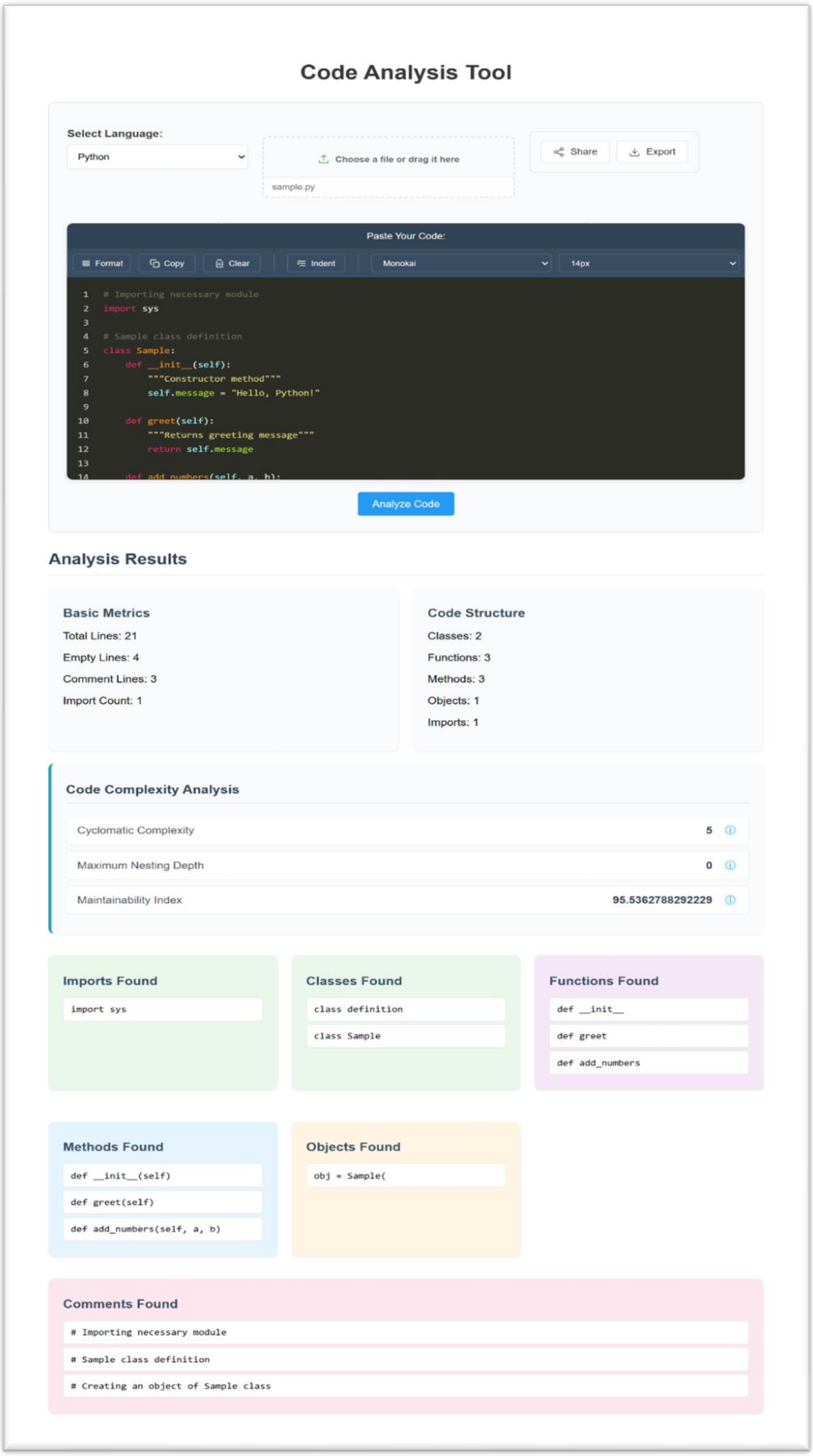
Ready to Analyze Your Code?

Get started with our powerful code analysis tool

Index page


Code Analysis Tool
Select Language:
Auto Detect
Choose a file or drag it here
Paste Your Code:
Format
Copy
Clear
Indent
Monokai
14px
Analyze Code

Analyse result



```
# Importing necessary module
import sys

# Sample class definition
class Sample:
    def __init__(self):
        """Constructor method"""
        self.message = "Hello, Python!"

    def greet(self):
        """Returns greeting message"""
        return self.message

    def add_numbers(self, a, b):
```

Analysis Results

Basic Metrics

Total Lines: 21

Empty Lines: 4

Comment Lines: 3

Import Count: 1

Code Structure

Classes: 2

Functions: 3

Methods: 3

Objects: 1

Imports: 1

Code Complexity Analysis

Cyclomatic Complexity	5
Maximum Nesting Depth	0
Maintainability Index	95.5362788292229

Imports Found

`import sys`

Classes Found

`class definition`

`class Sample`

Functions Found

`def __init__`

`def greet`

`def add_numbers`

Methods Found

`def __init__(self)`

`def greet(self)`

`def add_numbers(self, a, b)`

Objects Found

`obj = Sample(`

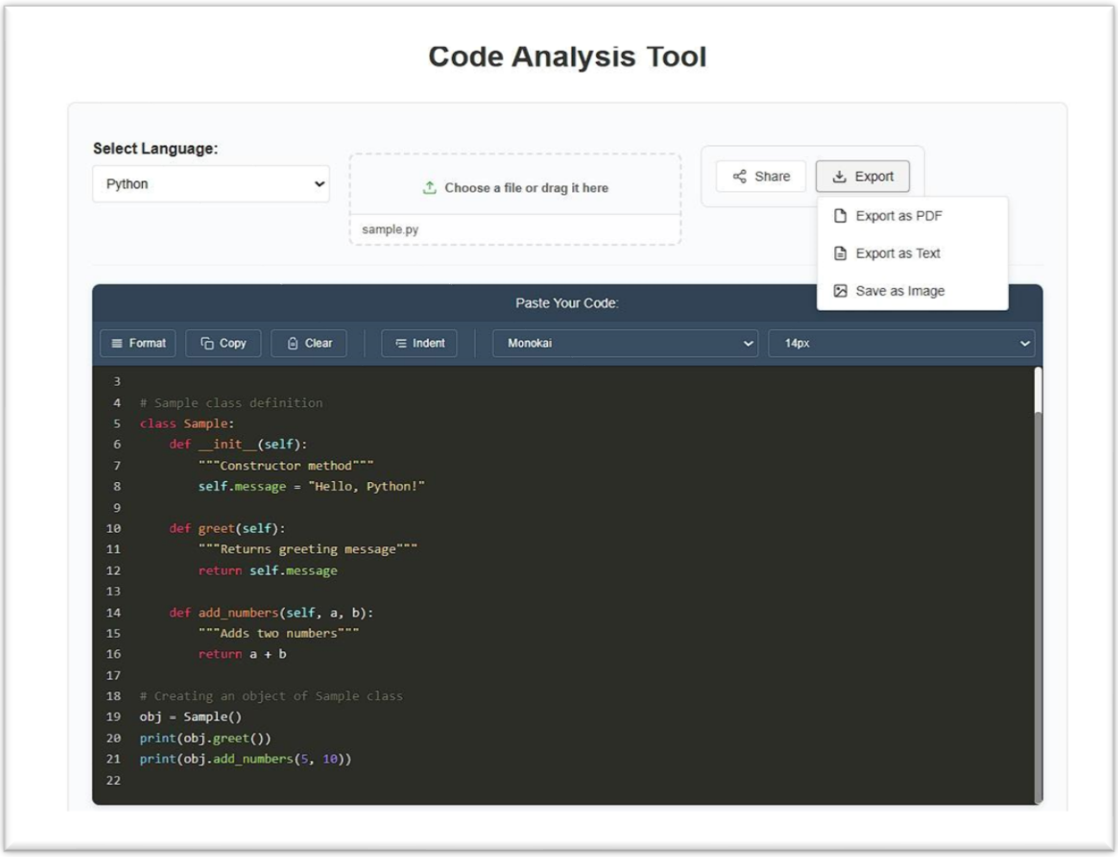
Comments Found

`# Importing necessary module`

`# Sample class definition`

`# Creating an object of Sample class`

Export: download result as pdf, txt, img format



Select Language: Python

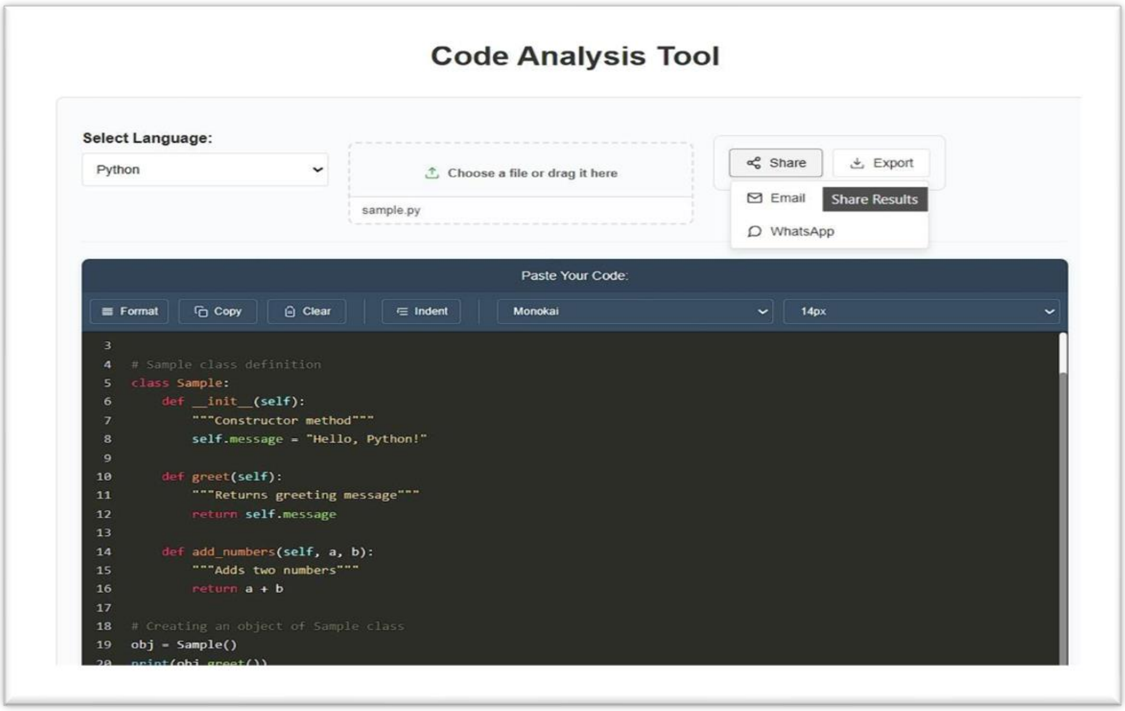
sample.py

Monokai

14px

```python

# Sample class definition
class Sample:
    def __init__(self):
        """Constructor method"""
        self.message = "Hello, Python!"

    def greet(self):
        """Returns greeting message"""
        return self.message

    def add_numbers(self, a, b):
        """Adds two numbers"""
        return a + b

# Creating an object of Sample class
obj = Sample()
print(obj.greet())
print(obj.add_numbers(5, 10))

```

Share: share via email or whatsapp



Select Language: Python

sample.py

Monokai

14px

```python

# Sample class definition
class Sample:
    def __init__(self):
        """Constructor method"""
        self.message = "Hello, Python!"

    def greet(self):
        """Returns greeting message"""
        return self.message

    def add_numbers(self, a, b):
        """Adds two numbers"""
        return a + b

# Creating an object of Sample class
obj = Sample()
print(obj.greet())
```

TESTING

8. TESTING

Testing is an essential phase in the development of the **Script Analysis Web App**, ensuring that the system functions correctly, delivers accurate analysis, and performs efficiently. The testing process covers unit testing, integration testing, functional testing and performance testing.

8.1 Testing Approach

Unit Testing

- Tested individual components such as **code metrics analysis, complexity calculation,** using Python
- Verified that the *radon* and *Pylint* packages provided correct outputs for different script inputs.

Integration Testing

- Ensured seamless communication between the **frontend (HTML, CSS, JavaScript)** and **backend (Flask, Jinja2, Werkzeug)**.
- Validated that script files uploaded via the web interface were correctly processed and return results.

Functional Testing

- Tested core functionalities, including:
  - Script upload and validation.
  - Complexity and maintainability analysis.
  - Code structure breakdown (functions, classes, method detection & etc).
  - Syntax highlighting and report generation.
- Verified system behavior with valid and invalid inputs (e.g., empty files, unsupported file formats).

Performance Testing

- Evaluated system responsiveness by testing with **small (1 lines), medium (500 lines), and large (2000+ lines) scripts**.
- Measured processing time and ensured performance optimizations were in place for handling larger scripts efficiently

8.2 Issues Identified & Fixes Implemented

Issue	Identified Problem	Solution Implemented
Slow processing of large scripts	Execution delays when analyzing large code files	Optimized Radon and Pylint calls, improved Flask request handling
Application crash on invalid file uploads	Unexpected termination when unsupported formats were uploaded	Added exception handling for invalid file types
Incorrect code metric calculations	Some scripts returned incorrect function and class counts	Improved parsing logic for better multi-language support
Unresponsive UI for large scripts	Page appeared frozen while processing large files	Implemented a loading indicator for better user experience

8.3 How Complexity Analyse is Calculated

1. Cyclomatic Complexity (CC):

Cyclomatic Complexity (CC) measures the number of independent paths in a program. It helps determine how complex a function is, indicating how many different ways the code can execute.

The formula for Cyclomatic Complexity (CC) is:

$$CC = E - N + 2P$$

Where:

E = Number of edges (transitions in the control flow graph)

N = Number of nodes (decision points + start/end)

P = Number of connected components (usually 1 for a single function)

Step-by-Step Breakdown of complex_function(x) :

```
def complex_function(x):
    if x > 0:# Decision Point 1
        for i in range(x): # Loop (Implicit decision
        point) - 2
            if i % 2 == 0: # Decision Point 3
                print("Even")
            else: # Decision Point 4
                print("Odd")
```

Decision Points Identified:

1. if x > 0 (1st decision point)
2. for i in range(x) (Loop adds complexity, considered as an implicit branch)
3. if i % 2 == 0 (2nd decision point)
4. else branch (3rd decision point)

Control Flow Graph (CFG)

A Control Flow Graph (CFG) represents the program’s flow using nodes and edges:

```
                  (Start)
                     |
   [if x > 0] ----> No ----> (End)
                     |
                    Yes
                     |
 [for loop] ----> Exit Loop -> (End)
                     |
               [if i % 2 == 0]
                   /    \
              [Even]    [Odd]
                   \    /
             (Loop back to for)
```

1. if x > 0 (First decision point)
2. for i in range(x) (Loop introduces a decision)
3. if i % 2 == 0 (Another decision point)
4. else branch (Implicitly part of the if, but counted as an additional path)

Applying the Cyclomatic Complexity Formula

1. Nodes (N):

    Start

    if x > 0

    for i in range(x)

    if i % 2 == 0

    else

    print("Even")

    print("Odd")

    End

  Total Nodes = 8

2. Edges (E):

    Start → if x > 0

    if x > 0 → No → End

    if x > 0 → Yes → Loop

    Loop → if i % 2 == 0

    if i % 2 == 0 → Print "Even"

    if i % 2 == 0 → No → Print "Odd"

    Print → Loop again (or exit)

  Total Edges = 10

3. Connected Components (P):

  Since we have only one function, P = 1.

Now, applying the formula:

$$CC = E - N + 2P$$

$$CC = 10 - 8 + (2 \times 1)$$

$$CC = 4$$

Final Answer:

- else If your tool counts loops as decision points, then CC = 4.
- If your tool ignores loop conditions in CC calculation, then CC = 3

2. Nesting Depth

Nesting Depth (MND) measures the deepest level of nested structures (such as loops and conditionals) within a function. It helps assess code complexity by determining how many layers of indentation are needed to follow the logic.

Step-by-Step Analysis of Nesting Depth :

```
def complex_function(x):                    # Level 0
        if x > 0:                           # Level 1
                for i in range(x):          # Level 2
                        if i % 2 == 0: # Level 3
                                print("Even")
                        else:               # Level 3 (same level as `if i % 2 == 0`)
                                print("Odd")
```

How Nesting Depth is Counted :

- Level 0 → def complex_function(x): The function itself.
- Level 1 → if x > 0: First conditional statement.
- Level 2 → for i in range(x): Loop inside the if statement.
- Level 3 → if i % 2 == 0: Condition inside the loop.
- Level 3 → else: At the same level as if i % 2 == 0, so it does not increase depth further

Why Not 4?

- Some might mistakenly count else as an additional level, but it is at the same level as if (i % 2 == 0), so it does not contribute to depth.
- No additional nested condition inside if (i % 2 == 0) or else , so depth does not increase further

CONCLUSION

9. Conclusion

The **Script Analysis Web App** successfully provides a comprehensive tool for analyzing source code across multiple programming languages. By offering features such as code metrics, complexity analysis, structure breakdown, syntax highlighting, and real-time error detection, the application helps developers better understand their code and improve maintainability. multi-language support, covering popular programming languages like Python, Java, C++, JavaScript, Ruby, Go, Swift, PHP, and C#. This versatility ensures that developers from diverse backgrounds can benefit from the tool, regardless of their preferred coding language. The modular design of the application enhances its scalability and maintainability. The separation of frontend and backend components allows for seamless updates and improvements,

The integration of **Flask, Radon, Pylint, and Jinja2** enables accurate and efficient analysis while maintaining a lightweight, user-friendly interface.

This project contributes to better software quality, improved coding standards, and enhanced debugging processes, making it a valuable tool for developers, educators, and software engineers.significantly reduces the time required for debugging, making the development process more efficient.

Through the use of cyclomatic complexity measurement, the app provides insights into code maintainability, helping developers identify areas for optimization. The syntax highlighting feature improves readability, and the option to download analysis reports offers a convenient way to document findings.

FUTURE ENHANCEMENTS

10. FUTURE ENHANCEMENTS

To further improve the **Script Analysis Web App**, the following enhancements are planned:

Expanded Language Support

- Extending support for additional programming languages such as **Kotlin, TypeScript, Rust, and R**.
- Enhancing parsing capabilities for complex scripts with multiple interdependent files.

Advanced Code Analysis

- Implementing **AI-based code quality suggestions** to improve maintainability.
- Adding **security vulnerability detection** to identify potential security risks.
- Providing **code duplication detection** to help maintain cleaner codebases.

User Experience Improvements

- Introducing **dark mode and customizable themes** for better readability.
- Allowing **real-time collaborative code analysis**, where multiple users can review a script together.
- Adding an **export feature for reports** in different formats (DOCX, JSON, Excel).

REFERENCES

11. REFERENCES

This chapter provides references used in the development, research, and implementation of the **Script Analysis Web App**. The sources include journals, books, and web-based resources that contributed to understanding programming analysis, complexity measurement, and software architecture.

11.1 Journal References

- Tahir, A., & MacDonell, S. G. (2021). A Systematic Mapping Study on Dynamic Metrics and Software Quality. *Journal of Software: Evolution and Process, 33(2),* e2345.
- Jin, S., Zhang, M., Guo, Y., He, Y., Li, Z., Chen, B., Zhu, B., & Xia, Y. (2023). Software Code Quality Measurement: Implications from Metric Distributions. *Journal of Software Maintenance and Evolution: Research and Practice*.
- Yao, J., & Shepperd, M. (2021). The Impact of Using Biased Performance Metrics on Software Defect Prediction Research. *Empirical Software Engineering, 26(5),* 1–27.
- Kaur, G., & Kaur, P. (2022). A Comprehensive Study on Code Quality Metrics. *International Journal of Advanced Computer Science and Applications, 13(4)*.
- McCabe, T. J. (1976). *A Complexity Measure*. IEEE Transactions on Software Engineering, SE-2(4), 308-320.
- Chidamber, S. R., & Kemerer, C. F. (1994). *A Metrics Suite for Object-Oriented Design*. IEEE Transactions on Software Engineering, 20(6), 476-493.
- Halstead, M. H. (1977). *Elements of Software Science*. Elsevier.

11.2 Book References

- Sommerville, I. (2015). *Software Engineering (10th Edition)*. Pearson Education.
- Pressman, R. S. (2019). *Software Engineering: A Practitioner's Approach (9th Edition)*. McGraw-Hill.
- Martin, R. C. (2009). *Clean Code: A Handbook of Agile Software Craftsmanship*. Prentice Hall.

11.3 Web References

- Flask Framework Documentation: https://flask.palletsprojects.com/
- Radon Package Documentation: https://radon.readthedocs.io/
- Pylint Documentation: https://pylint.pycqa.org/
- Jinja2 Documentation: https://jinja.palletsprojects.com/
- Cyclomatic Complexity: https://www.geeksforgeeks.org/cyclomatic-complexity/