**一、System requirement analysis**

1、**Overview of system requirements**

The overall system requirements for this project include functional requirements, non-functional requirement requirements, performance requirements.The system is designed to help developers automatically detect bugs in code, analyze potential code quality issues, and provide suggestions for changes to improve software stability and security. The tool analyzes Python-based projects and provides an easy-to-understand visual interface.

In order to achieve this goal, we surveyed developers in the form of a questionnaire and wrote the following user stories and acceptance criteria:

1. As a developer, I want to upload a Python-based code, so that the system can analyze my code and generate a quality report.

**Acceptance criteria:**I select a Python project file in the interface.When I click the "Upload" button,the system should successfully upload the project and start analyzing its code quality.

(2)As a beginner, I want to see the CWE issue overview, code quality score, and detailed information after the project analysis is complete, so that I can quickly understand the security and quality of the project.

**Acceptance criteria:**When the analysis is complete,I would see the CWE issue overview and code quality score on a summary page.

(3)As a developer, I want to view the history page, so that I can track the trends of CWE issues and code quality scores through graphs and charts.

**Acceptance criteria:**When I access the history page, I will see the trend graphs and distribution charts for CWE issues and code quality scores.

1. As an experienced user, I want to upload a custom configuration file to override the relevant rules, so that I can categorize and detect CWE issues based on my project-specific needs.

**Acceptance criteria:**When I upload the file,the system will recognize CWE issues based on my configuration file.

1. As a developer, I want to export the analysis history data in CSV or PDF format, so that I can save and process the data offline.

**Acceptance criteria:**When I click the "Export" button and select CSV or PDF format,the system are supposed to download the history data file.

1. As a registered user, I want to log in to my account, so that I can access my personal data and use the system's features.

**Acceptance criteria:**When I enter my email/username and password and click "Login", I should be authenticated and granted access to my dashboard.

Based on the above user story, we used an object-oriented approach to analyze the overall functional requirements of the system, determined the relationship between the requirements, and drew the following use case diagram:

**2、Module division**

In order to realize the above user story and meet user needs, we divide the system into six modules as shown in the figure below.



**二、Module requirement analysis**

**1、User Authentication & Account Management Module**

This module mainly implements user login registration and user information management. The user story includes the following parts.

(1)As a new user, I want to register for an account, so that I can access the system's features.

(2)As a registered user, I want to log in to my account, so that I can access my personal data and use the system's features.

(3)As a logged-in user, I want to log out of my account, so that I can securely end my session.

(4)As a manager,I want to query and manage user information,so that I can manage all users.

The use case diagram of this module is as follows.



**2、File Analysis Module**

The file analysis module is the core component of the system, responsible for automatically processing Python project files uploaded by users and conducting in-depth code analysis. User stories mainly include:

1. As a system, I want to automatically analyze users’ Python project for CWE issues, so that I can let users identify potential security vulnerabilities.
2. As a system, I want to use Pylint or other static code analysis tools to identify code formatting issues (C), warnings (W), and errors (E), so that I can help developers improve their code quality.
3. As a system, I want to calculate a code quality score for users’ project, so that I can help user quickly understand the overall quality of the code.
4. As an advanced user, I want to upload a custom configuration file, so that I can override the default Pylint rules to meet my project-specific requirements.

The use cases diagram of this module are below.



**3、ChatGPT Interaction Module**

The role of the ChatGPT interaction module in this project is to provide an intelligent user interaction interface to help users communicate with the system in a natural language. Users can ask questions about code analysis, CWE issues, code quality scoring, etc. through the ChatGPT module, and the system will provide detailed explanations, suggestions, and functional guidance based on the user's input. This module simplifies the user's learning curve, allowing non-technical users to easily understand the analysis results and receive targeted support, thus improving the overall user experience.

As a user, I want the system to automatically send the English CWE issue descriptions and the corresponding code to ChatGPT, so that I can receive Chinese modification suggestions and a revised version of the code.



**4、Database Module**

The role of the database module in this project is to store and manage user-uploaded project data, analysis results, CWE issue records, code quality scores, user-defined configuration files, and historical analysis records. It ensures data persistence and security, supports quick retrieval of query and analysis results, and provides project version tracking and history viewing functions. The database module is the key foundation for the system to efficiently process and save large-scale data, supporting the normal operation and data interaction of the entire system.The use cases diagram of this module are below.

1. As a database,I want to store large amounts of users’ project submission records and keep the data safe ,so that I can help users save data.
2. As a user, I want to chech and modify my project submission records, so that I can track the number of CWE issues, code quality scores, and historical data for each submission.



**5、History Tracking & Visualization Module**

The visualization module intuitively presents analysis results through charts, trend charts, and distribution charts to help users understand complex code quality information and CWE problem detection results. It transforms abstract data into easy-to-understand graphics, improving users' analysis efficiency and decision-making capabilities.The use cases diagram of this module are below.

1. As a visualizing system,I want to generate a line chart or area chart of the CWE issue number ,so that I can show developers the CWE issue trend.
2. As a visualizing system,I want to generate a stacked bar chart or pie chart of CWE issue,so that I can let users understand CWE problem distribution.
3. As a visualizing system,I want to generate a line chart of changes in code quality scores,so that I can help developers understand score changes
4. As a visualizing system,I want to generate a heat map or scatter plot of CWE issues versus code quality scores,so that I can get the correlation between the two.



**6、Export & Report Generation Module**

Finally, this module can import and export users' historical records and project development processes in the form of reports or data tables, which is helpful to alleviate database pressure.The use cases diagram of this module are below.

1. As a user, I want to export my historical data as a CSV file, so that I can easily analyze and share my project submission records.
2. As a user, I want to generate a PDF report that includes visual charts and data summaries, so that I can present my analysis results in a professional format.



**三、Non-functional requirements analysis**

**1、security requirements**

In order to ensure that the software can resist various potential threats during operation and protect the interests of users, we need to implement the security of the software system during the design process.

1. Data security: In order to protect user privacy, we encrypt user information. At the same time, in order to prevent the impact of data loss, we use a database with backup function
2. System security: When the system crashes under high load, attack or abnormal situation, we adopt a caching strategy to cache the database and commonly used pages, and also use index optimization for the database.

**2、Data visualization**

Through visualization, users can quickly identify the changing trend of the number of bugs, thereby having a correct understanding of their programming habits. We visualize data in the following ways:

1. CWE Issue Trend Chart: Clearly shows the trend of CWE issues over time, helping to analyze the increase or decrease of security issues.
2. CWE Issue Distribution Chart: Visually displays the distribution of different types of CWE issues, helping to identify high-risk areas.
3. Code Quality Score Trend Chart: Shows the trend of code quality scores, making it easier to monitor improvements or declines in code quality.
4. CWE Issues vs. Code Quality Score Correlation Chart: Reveals the relationship between CWE issues and code quality, helping to identify how code quality impacts security.