# 🧠 Overview

This Flask app performs static code analysis on a GitHub repository based on a provided **module name** and optionally compares it with the contents of an uploaded **PDF**. It uses multiple AI APIs (Hugging Face and OpenRouter) to generate summaries, improvement suggestions, and issue reports. It also supports a simple chatbot interface for interactive communication.

---

# 🔧 Core Functionalities

## 1. Cloning and Updating Repositories

**Function**: `clone_or_update_repo(repo_url, local_path)`

- Clones or pulls the latest updates from a given GitHub repo into a local directory.
- Ensures it doesn't mix up repositories (deletes and reclones if the remote URL doesn't match).

---

## 2. Finding Module-Related Code

**Function**: `find_module_files(local_path)`

- Walks through the repo directory and collects files with common code extensions: `.py`, `.js`, `.java`, `.cpp`, `.ts`.

**Function**: `extract_functions_and_comments(file_path, module_name)`

- Searches for functions and comments in code files that mention or relate to the given `module_name`.
- Supports basic recognition of Python, JavaScript, Java, C++, and TypeScript syntax.
- Captures:
    - Single-line and multi-line comments

- Function/class definitions containing the module name

---

## 3. Report Generation

**Function**: `generate_report(result_data, module_name)`

- Structures findings (functions + comments) into a human-readable text file.
- Uses emojis and formatting to enhance readability.
- Saves report to `report.txt` for download.

---

## 4. AI-Based Enhancements

### 📚 Summarization

**Function**: `generate_hf_summary(prompt_text)`

- Uses HuggingFace's `facebook/bart-large-cnn` to summarize the report content.

### 💡 Improvement Suggestions

**Function**: `generate_openrouter_suggestions(summary_text)`

- Asks OpenRouter (Mistral model) to act as a code reviewer and suggest improvements based on the summary.

### 💬 Chatbot Interaction

**Function**: `generate_openrouter_suggestions1(summary_text)`

- Responds to user queries or greetings using Mistral via OpenRouter.

### 📈 Semantic Similarity

**Function**: `adjusted_similarity_score(module_name, pdf_text, title_text, report_summary)`

- Compares the cleaned text of the PDF against the AI summary of the code report using semantic similarity scoring.
- Uses OpenRouter (Mistral) to score similarity between two chunks of text.

---

### 5. Security and Quality Audits

**Route**: `/generate_issues`

- Scans all the code files and sends them (up to 12,000 characters) to OpenRouter.
- Asks it to point out bugs, security risks, and bad practices.

---

## 🖼️ Web Interface

**Route**: `/`

- Main form for users to:
    - Enter GitHub repo URL
    - Specify module name
    - Upload a related PDF document
- Displays:
    - Code analysis report
    - AI summary
    - Suggestions for improvement
    - Semantic similarity score (if PDF provided)

**Route**: `/download_report`

- Allows downloading of the `report.txt` generated.

**Route**: `/chat`

- Accepts JSON POST requests with user messages and returns chatbot-style responses.

---

## 📁 Technologies Used

| Component | Tool / Library |
|---|---|
| Web Framework | Flask |
| Git Operations | GitPython (`git`) |
| PDF Parsing | PyPDF2 |
| NLP Model | SentenceTransformer |
| Summarization | HuggingFace API (`facebook/bart-large-cnn`) |
| AI Suggestions | OpenRouter (Mistral) |
| Env Management | `dotenv` |

---

## ✅ Strengths

- Modular and readable codebase.
- Uses AI to provide value-added insights, not just static analysis.
- Resilient handling of git errors and non-matching URLs.
- Friendly output format (emoji + bullet points).
- Truncates large inputs to stay within model limits.

- Interactive chatbot for follow-up.

---

## ⚠️ Potential Improvements

- **Better error handling/logging**: Currently, errors from API calls or file operations may not always be visible in the UI.
- **Security**: There is no input sanitization. Consider validating user input (especially `repo_url`).
- **UI/UX**: Depending on the `index.html`, user experience can be enhanced (e.g., progress indicators, highlighting matched code).
- **Caching**: If the same repo/module is analyzed again, caching results can speed up responses.
- **Testing**: Add unit tests for each utility function.

### Team Contributions

- **Raghuram**: Developed the Flask backend; implemented repository cloning and parsing; generated code suggestions and vulnerability insights.
- **Satyannarayana**: Analyzed and summarized client requirement documents.
- **Nayak**: Focused on code parsing and evaluating code optimality.
- **Raghuveer**: Created code summaries using Python's Abstract Syntax Tree (AST).
- **Sampath**: Designed the user interface and implemented bidirectional link visualization.
- **Aseem**: Conducted testing and validated traceability links.
- **Pranav**: Integrated chatbot functionality and managed project documentation.