**Báo Cáo Đề Tài: Xây Dựng Công Cụ Đánh Giá Chất Lượng Mã Nguồn Dựa Trên Code Metrics**

**1. Mục Tiêu Của Đề Tài**

* Xây dựng một công cụ phân tích mã nguồn tự động để đánh giá chất lượng mã dựa trên các chỉ số:
  + **Lines of Code (LOC)**: Đo kích thước mã nguồn.
  + **Cyclomatic Complexity**: Đánh giá độ phức tạp của mã.
  + **Code Duplication**: Phát hiện mã trùng lặp.
  + **Cohesion & Coupling**: Đánh giá mức độ gắn kết và phụ thuộc giữa các thành phần.
* Hiển thị báo cáo chi tiết và đề xuất cải thiện chất lượng mã.
* So sánh kết quả với công cụ thương mại **SonarQube** để đánh giá hiệu quả.

**2. Các Bước Thực Hiện**

**Bước 1: Thu Thập Dữ Liệu và Phân Tích Mã Nguồn**

* **Công cụ thu thập metrics**:
  + Sử dụng thư viện Python như radon (LOC, Cyclomatic Complexity), flake8 (phát hiện lỗi), pylint (đánh giá tổng thể).
  + Phát hiện mã trùng lặp bằng thư viện jscpd hoặc pycopy.
  + Đánh giá **Cohesion & Coupling** bằng cohesion hoặc tự tính toán dựa trên cấu trúc mã.
* **Nguồn dữ liệu**:
  + Mã nguồn từ các dự án mã nguồn mở trên GitHub.
  + Dự án nội bộ hoặc bài tập môn học.

**Bước 2: Tiền Xử Lý Dữ Liệu**

* **Chuẩn hóa dữ liệu**:
  + Chuyển đổi các chỉ số thành dạng số (nếu cần).
  + Xử lý giá trị thiếu (nếu có).
* **Lưu trữ dữ liệu**:
  + Dùng pandas để lưu trữ và phân tích dữ liệu metrics.

**Bước 3: Xây Dựng Mô Hình Đánh Giá**

* **Phương pháp đánh giá**:
  + **Quy tắc dựa trên ngưỡng (Rule-based)**:
    - Ví dụ: Cyclomatic Complexity > 10 → Cảnh báo mã phức tạp.
    - Code Duplication > 5% → Đề xuất tối ưu.
  + **Machine Learning (tùy chọn mở rộng)**:
    - Phân loại chất lượng mã (tốt/trung bình/kém) dựa trên metrics.

**Bước 4: Hiển Thị Báo Cáo**

* **Công cụ trực quan hóa**:
  + Sử dụng matplotlib, seaborn hoặc plotly để vẽ biểu đồ.
  + Ví dụ:
    - Biểu đồ cột so sánh các chỉ số giữa các tệp.
    - Heatmap thể hiện mối tương quan giữa các metrics.
* **Đề xuất cải thiện**:
  + Tự động sinh báo cáo bằng Jinja2 hoặc ReportLab.

**Bước 5: So Sánh Với SonarQube**

* **Phương pháp so sánh**:
  + Chạy cùng một mã nguồn qua SonarQube và công cụ tự xây dựng.
  + Đánh giá sự khác biệt về kết quả metrics.
  + Phân tích ưu/nhược điểm của từng công cụ.

**3. Công Cụ Đề Xuất**

* **Ngôn ngữ lập trình**: Python.
* **Thư viện**:
  + Phân tích mã: radon, flake8, pylint, jscpd.
  + Xử lý dữ liệu: pandas, numpy.
  + Trực quan hóa: matplotlib, seaborn, plotly.
  + Báo cáo: Jinja2, ReportLab.
* **Môi trường phát triển**: Jupyter Notebook, VS Code.

**4. Báo Cáo Kết Quả**

* **Nội dung báo cáo**:
  + Mô tả phương pháp thu thập và phân tích metrics.
  + Kết quả đánh giá từ công cụ tự xây dựng.
  + So sánh với SonarQube.
  + Đề xuất cải thiện chất lượng mã.
* **Kết quả mong đợi**:
  + Một công cụ Python có thể quét mã nguồn và trả về báo cáo chất lượng.
  + Tài liệu đánh giá hiệu quả so với SonarQube.

**5. Hướng Mở Rộng**

* Tích hợp công cụ vào CI/CD pipeline (GitHub Actions, GitLab CI).
* Phát triển giao diện web bằng Flask/Django.
* Mở rộng phân tích sang ngôn ngữ khác (Java, C++).

**6. Kết Luận**

Đề tài giúp sinh viên hiểu rõ về **code metrics** và cách áp dụng Python để xây dựng công cụ đánh giá mã nguồn tự động. Kết quả có thể ứng dụng trong kiểm thử phần mềm hoặc cải thiện quy trình phát triển.