Source Code Plagiarism Detector

Background

Unfortunately, plagiarism is a major problem at many schools. There are different scenarios where two or more people produce different versions of the same code, one version is derived from another using copy-then-modify or developed collaboratively without permission. The idea is to develop an interactive tool that can detect such situations and visualize the result.

Specification

'Source Code Plagiarism Detector' is a web-based application which will allow the primary user (Faculty/TA) to compare and detect similarity between two programs written in the 'C' language. The application will go beyond a 'textual diff' and consider more sophisticated transformations such as renaming of variable/functions, extracting/organizing code into functions, moving code around etc. The interactive web application will allow the users to provide two inputs in the form of text boxes or uploaded files or directory structure. Upon completion, a report will be generated which explains how one version could be derived from another. An admin user will carry out maintenance tasks such as authorizing/creating new users, deleting existing users etc. A secondary user – 'Committee Member' will have overall birds-eye view of the system and he/she will be able to generate detailed periodical reports about overall, course specific violations etc.

Users/Primary actors

Following are the primary actors in this project:

1. **Faculty**: Professors etc. who will run the plagiarism software to find violations. They can also file a request to add/remove their TAs to/from the system.
2. **TAs**: Teaching assistants who will work with the professors and help them with the grading. They can use the application once they are granted the access upon Professor's request. Upon end of the tenure, the access will be revoked.
3. **Admin/Operations**: Handles administrative/maintenance tasks such new user approval, revoke access etc.
4. **Committee Members**: Authority on top of Faculty which handles addition/removal of Professors, their access etc. Can also generate periodical system wide reports etc.

Scope and Features

Web Application

- A simple yet compelling web-based interface for end-users.
- Authentication mechanism to impose access controls, restrict usage and provide user type specific features.
- Allows authenticated Faculty/TAs to compare programs and generate diff-based reports.
- Download reports in different formats like pdf, excel etc.
- Allows super users to file a site access enrollment requests i.e. a committee member can file a request for new professors, a professor can file a request for new TAs
- Allow super users to file a site access termination requests i.e. a committee member can file a request for an existing professor, a professor can file a request for TAs whose tenure is going to get over etc.
- Admin interface which will process new access grant/termination requests.
- Committee members can view/download system-wide/specific/granular/periodic reports about total number of violations instances etc.

What will not be covered

- Student access to the application
- Plagiarism detection against historical data (previous assignments)

Proposed/Planned Technology Stack

Feature	Technology/Tool
Implementation language (Backend)	Java, Spring Boot, REST/GraphQL
Implementation language (Frontend)	React, JS, HTML/CSS
Automated testing	JUnit
Version control	NEU GitHub
Issue tracking	NEU GitHub
Requirements/documentation	Confluence
Continuous integration	Jenkins
Messaging	Slack
Integration environment	AWS

Release Planning

Sprint 1 (Requirement)

1. Requirement gathering and analysis
2. Use case creation
3. Mock ups for UI
4. Project proposal documentation, decide on the programming language to target for the application
5. Brainstorming about the technologies to be used

Sprint 2 (Design)

1. UML diagrams creation
2. Create Java interfaces
3. Finalize technologies to be used
4. Read literature to understand the techniques, algorithms used in plagiarism detection, clone detection, creating abstract syntax trees (ASTs)
5. Role assignment

Sprint 3 (Implementation)

1. Creating wireframes from mock ups
2. Set up end-to-end project structure
3. Set up CICD with Jenkins on AWS instance

Sprint 4 (Coding/Testing)

1. Implement use case – Login
2. Implement use case – Request Registration/Enrollment
3. Implement use case – Approve Registration/Enrollment Request
4. Perform unit/integration/acceptance testing

Sprint 5 (Coding/Testing)

1. Implement use case – Compare source code
2. Implement use case – Download report
3. Perform unit/integration testing
4. Backlogs from Sprint 4

Sprint 6 (Coding/Testing)

1. Implement use case – Send email notifications
2. Implement use case – Update user
3. Implement use case – Delete user
4. Implement use case – Request to delete user
5. Perform unit/integration testing
6. Backlogs from Sprint 5

Sprint 7 (Miscellaneous)

1. Documentation and user manual
2. End-to-end testing
3. Code refactoring
4. Bug fixes
5. Backlogs from Sprint 6

Sprint 8 (Hand over)

1. Final presentation and report