Parametrized String Matching Implementation for Software Plagiarism Check

Team
amit.tomar
(MT2013008)

siddhesh.dosi
(MT2013150)

srinivas.r.vaidya
(MT2013152)

@iiitb.org

07 - Feb - 2014

1 Introduction

This project aims at developing a Parametrized String Matching Implementation for Software Plagiarism Check, that given a collection of files which contain code in some programming language, will show a set of possible duplications of parts of the code among these. Comparing pieces of software will require discounting comments (optional and language dependent), extra/blank lines and spaces, variable renaming etc. The theory of parametrized string matching will be used to implementat this project. System will have an easy-to-use UI for selecting files/folders and shall report the plagiarism related information (matches found) in the UI in a nice manner.

2 Functional Requirements

Requirement.No.	1.1
Input	File option
Output	User prompted to select files from multiple folder.
Processing	Populate the folder structure of file system.

Requirement.No.	1.2
Input	Browsing and selection of files and next button
Output	User is promted to enter code snippet to be ignored while processing
Processing	File path validation

Requirement.No.	1.3
Input	Check plagiarism
Output	Plagiarism related log is generated.
Processing	Generate parameterized suffix tree to check amount of plagiarism.

Requirement.No.	2.1
Input	Folder option
Output	User is prompted to select Folder.
Processing	Search for all the files in the selected folder and populate a list.

Requirement.No.	2.2
Input	Selection/Deselection of files from the populates list and next button
Output	User is promted to enter code snippet to be ignored while processing
Processing	File path validation

Requirement.No.	2.3
Input	Check plagiarism
Output	Plagiarism related log is generated.
Processing	Generate parameterized suffix tree to check amount of plagiarism.

3 Non - Functional Requirements

External interface requirements :

1. Shall be portable, across various hardware and software platforms.
2. Shall be scalable and reliable.
3. Shall be easy to use.

Performance requirements :

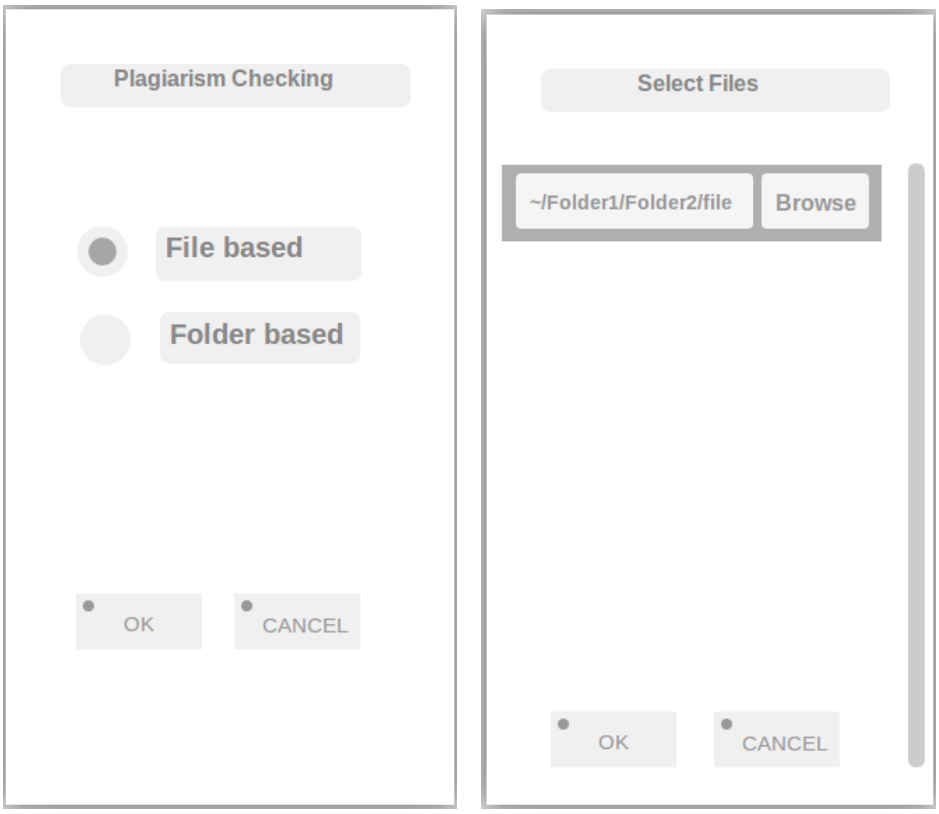
1. Shall have a good response time.

4 Goals of implementation

Plagiarism is a serious issue in computer science courses involving assessment of programming assignments [1]. Being electronic in nature, it is very easy to copy code and it is difficult to differentiate between the original and copied work. Thus, there is a need for a tool to detect plagiarism automatically, assisting professor to check for any kind of copying done by students.

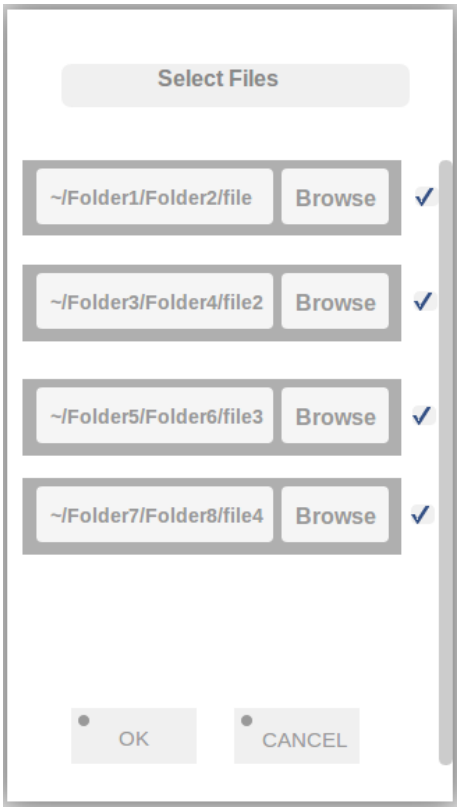
5 UI Flow

Following screen shots show the various UI screens :

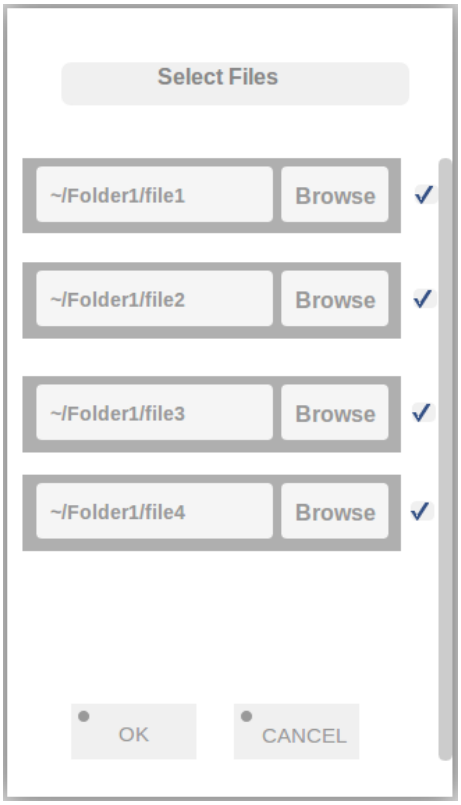


(a) Starting screen

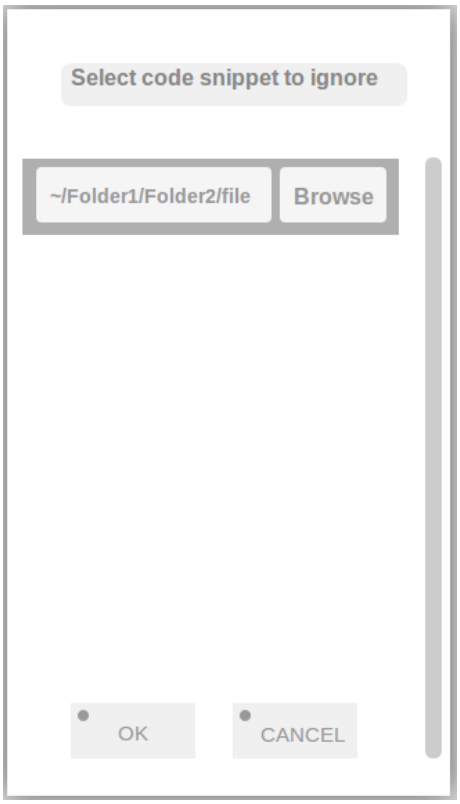
(b) Screen to select file



(c) Screen to select/deselect file



(d) Screen to display all the files in the selected folder



(e) Screen to select code snippet

(f) Screen to display plagiarism analysis report

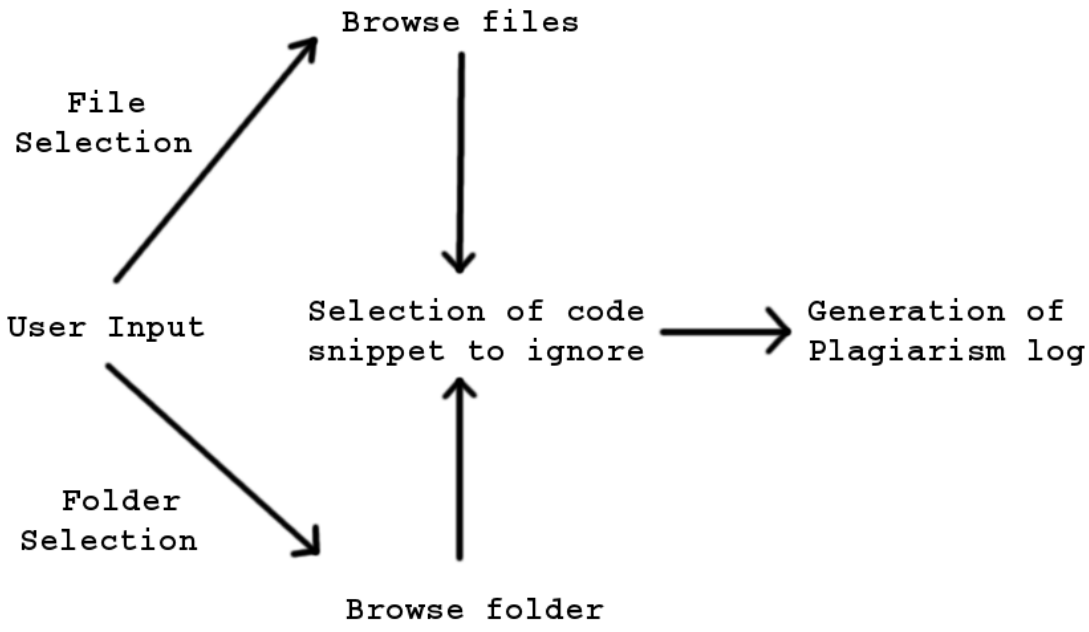


Figure 1: Decision Tree

6 Hardware and Software requirements

S.No.	Software	Version	Purpose
1	Ubuntu Linux	13.04	Operating system.
2	GitHub	1.8.3.2	Version Control
3	Spyder	2.2.1	IDE for Python
4	GIMP	2.6	Image editing for documentation
5	Gummi	0.6.5	LaTeX editing for documentation

References

[1] "An Anti-Plagiarism Editor for Software Development Courses", *Proceeding ACE '05 Proceedings of the 7th Australasian conference on Computing education - Volume 42 Pages 83-90* , 2005.