CS 5500 Managing Software Development

Phase A

Project Statement:

Academic integrity is one of the most important university policies, expected to be followed and respected by all students. Universities all around the world have some kind of restrictions on work collaboration when it comes to submitting course work. When a student copies their work from other students or from the web without citation or permission, it is considered as Plagiarism. Plagiarism is against the policies of academy honesty and is not acceptable.

To detect plagiarism in submitted work, it takes some kind of systematic mechanism. For handwritten work, it would take manual reading and comparison at the discretion of the grader. But even for electronic submissions, It takes extra work from professors or TAs to determine whether a student is plagiarising. To be more specific to the Computer Science domain, teaching staff have to not only check Students' written documents, but also their programs for plagiarism. It may take more time than just checking plain text.

To solve this problem, our project aims to develop a web-based code plagiarism detector. The product will help teaching staff to detect plagiarism in coding by highlighting pieces of code that are determined to be copied. Comparison will be done against the work submitted by other students. We plan to compare whole files, and also individual pieces of code that have a high chance of similarity. We plan on comparing the whole file to each other by comparing their Syntactic Trees and functions by building a call stack for that function and comparing it with the call stacks of functions in the other program. To make the checks more accurate, we will be dropping keywords and identifiers that are specific to the language. Comments will also be compared but keywords won't be ignored here. Based on the results of these checks, we will give a similarity score to the documents. Documents that have a similarity score above a certain threshold will be considered to be plagiarized. If the algorithm works well for single file to file comparison, we plan to extend the design to complete projects that would allow an entire project to be compared with another one for plagiarism.

Our product will choose Java as the target language since it is one of the most common programming languages in the object-oriented paradigm and many universities include in as a part of their curriculum for teaching basic object oriented programming principles, or even for building advanced applications. Our project will be implemented using Java on the backend for running our detection algorithm and user interface will be built using HTML, CSS and JavaScript.

Use Cases:

Use case #1:

Use Case:	Uploading files for plagiarism detection
Primary Actor:	Teaching staff
Goal in Context:	1. Upload selected files from user's local file system 2. Show name and delete button of uploaded files
Preconditions:	1. User opened the start page of plagiarism detector
Trigger:	User clicks on the "Upload Files" button
Scenario:	1. User clicks on the "Upload Files button" 2. Local file explorer window pops up 3. User selects the files for plagiarism detection 4. User clicks on 'Open' button from file explorer window 5. File uploads to the system 6. File name and delete button shows under the "Plagiarism Detection" button
Exceptions:	1. Uploaded files are *not* java files: re-upload java file 2. System freezes for a long time and cannot show results: user should check if their network is established and refresh pages. If it still does not work, user should find contact information in help page and contact system administrators.
Priority:	Essential, must be implemented
When Available:	First increment
Channel to Actor:	Via Interactive webpage
Secondary Actor:	System Support Team
Channels to Secondary Actors:	Email
Open Issues:	Should there be a way to let user enter URL (e.g. Github repository URL) as input, and let system fetch the projects/files.

Use Case #2:

Use Case:	View overall code similarities summary
Primary Actor:	Teaching staff
Goal in Context:	After running plagiarism detecting algorithms on the uploaded two projects, show the similarity summaries.
Preconditions:	1. User has uploaded the files that are being compared 2. User has requested for a plagiarism check on the code files by clicking the 'Detect Plagiarism'
Trigger:	User clicks the 'Detect Plagiarism' button to run the plagiarism system and wait for system response.
Scenario:	1. User clicks on the 'Detect Plagiarism' button. The system shows loading page, waits for the back-end algorithm to return and show results on web page 2. System runs plagiarism detecting algorithms 3. System return the result to front-end webpage 4. User views the overall rank of the code similarities
Exceptions:	1. Uploaded files are less than two: follow instructions on the prompt and upload at least two files 2. System freezes for a long time and cannot show results: user should check if their network is established and refresh pages. If it still does not work, user should find contact information in help page and contact system administrators.
Priority:	Essential, must be implemented
When Available:	First increment
Channel to Actor:	Interactive Webpage
Secondary Actor:	System Support Team
Channels to Secondary Actors:	Email
Open Issues:	Should we accept the threshold for considering file a case of plagiarism as a user input?

Use case #3:

Use Case:	View single code comparison in detail

Primary Actor:	Teaching staff
Goal in Context:	1. User views code similarities in two side by side panes comparing two code files 2. Code that is thought to be similar is highlighted.
Preconditions:	1. User has uploaded the files that are being compared 2. User has clicked on a particular similarity case
Trigger:	User clicks on one of the code pairs that are detected to be similar
Scenario:	1. User clicks on the result shown in the summary 2. User views sections of the code that are detected to be similar 3. Code blocks that are considered to be similar are highlighted
Exceptions:	System freezes for a long time and cannot show results: user should check if their network is established and refresh pages. If it still does not work, user should find contact information in help page and contact system administrators.
Priority:	Essential, must be implemented
When Available:	First Increment
Channel to Actor:	Interactive Webpage
Secondary Actor:	System Support Team
Channels to Secondary Actors:	Email
Open Issues:	

Use Case #4:

Use Case:	Exploratory Code Comparison
Primary Actor:	System
Goal in Context:	1. Comparison of two code files line by line 2. Finding blocks of code with obvious similarity

Preconditions:	1. User has uploaded the files that are being compared 2. User has requested for a plagiarism check on the code files
Trigger:	User clicks ‘Check for Plagiarism’ button
Scenario:	1. System reads code line by line and does a similarity comparison 2. White spaces and blank lines are ignored, keywords are ignored 3. Every same word encountered increases the similarity score
Exceptions:	System freezes for a long time and cannot show results: user should check if their network is established and refresh pages. If it still does not work, user should find contact information in help page and contact system administrators.
Priority:	Essential, must be implemented
When Available:	First Increment
Channel to Actor:	Java classes and methods
Secondary Actor:	None
Channels to Secondary Actors:	None
Open Issues:	

Use Case #5:

Use Case:	Comparing similar methods
Primary Actor:	System
Goal in Context:	1. Building a call stack for each function in the file 2. Finding other functions with a similar/same call stack
Preconditions:	1. User has uploaded the files that are being compared 2. User has requested for a plagiarism check on the code files

	3. Exploratory analysis does not give a high similarity score
Trigger:	User clicks ‘Check for Plagiarism’ button
Scenario:	1. Each function is parsed to check for function calls, looping structures, return statements 2. A stack is built based on the order of calls 3. Comparison is done amongst similar stacks
Exceptions:	System freezes for a long time and cannot show results: user should check if their network is established and refresh pages. If it still does not work, user should find contact information in help page and contact system administrators.
Priority:	Essential, must be implemented
When Available:	Third Increment
Channel to Actor:	Java classes and functions
Secondary Actor:	None
Channels to Secondary Actors:	None
Open Issues:	

Use case #6:

Use Case:	Download similarity report
Primary Actor:	Teaching staff
Goal in Context:	1. User downloads the report that summarises the overall rank of the code similarities in plain text and diagrams.
Preconditions:	1. User has entered URL list or uploaded files as input 2. User has clicked the ‘Detect Plagiarism’ button
Trigger:	User clicks the Download Report button to download that document and wait for system response.
Scenario:	1. User clicks on the ‘Download Report’ button 2. The system shows loading page, wait for the back-end algorithm to return and show results on web page.

	3. The specified report is saved on user's computer.
Exceptions:	System freezes for a long time and cannot show results: user should check if their network is established and refresh pages. If it still does not work, user should find contact information in help page and contact system administrators.
Priority:	Essential, must be implemented
When Available:	First Increment
Channel to Actor:	Interactive Webpage
Secondary Actor:	System support team
Channels to Secondary Actors:	Email
Open Issues:	

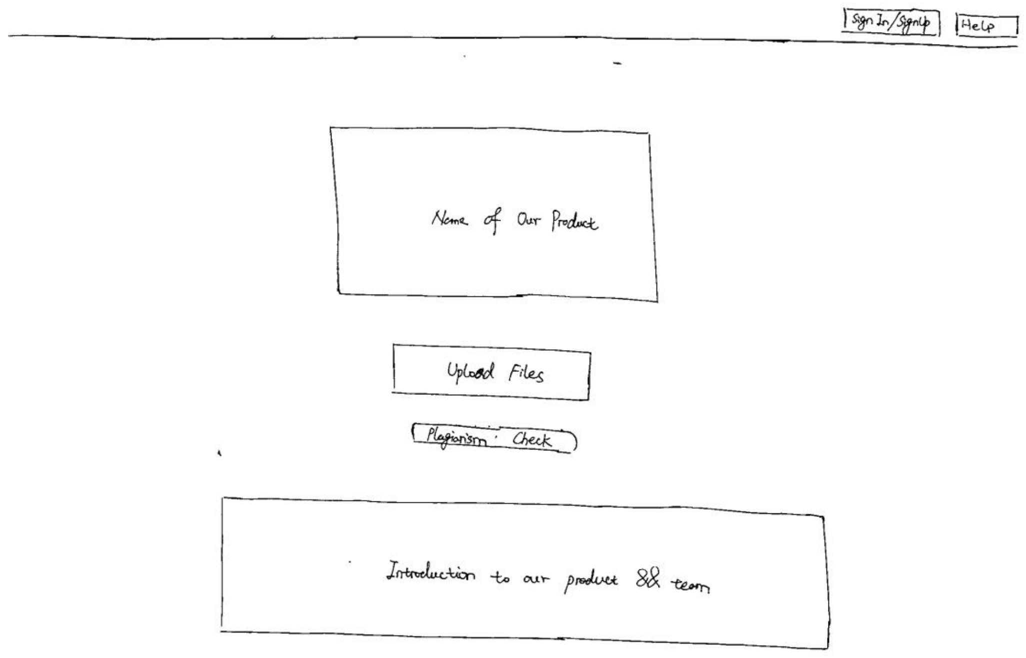
Use Case #7:

Use Case:	Comment comparison
Primary Actor:	System
Goal in Context:	1. Finding all comments in two files 2. Checking similarity between comment lines
Preconditions:	1. User has uploaded the files that are being compared 2. User has requested for a plagiarism check on the code files
Trigger:	User clicks 'Check for Plagiarism' button
Scenario:	1. Each comment is compared with every comment in the second file 2. Multiple whitespaces are ignored and replaced with a single space, keywords are not ignored 3. Any code present along with the comment in the same line is stripped off 4. Resulting strings are compared word by word for similarity, or by using an n-gram model
Exceptions:	System freezes for a long time and cannot show results: user should check if their network is established and refresh pages. If it

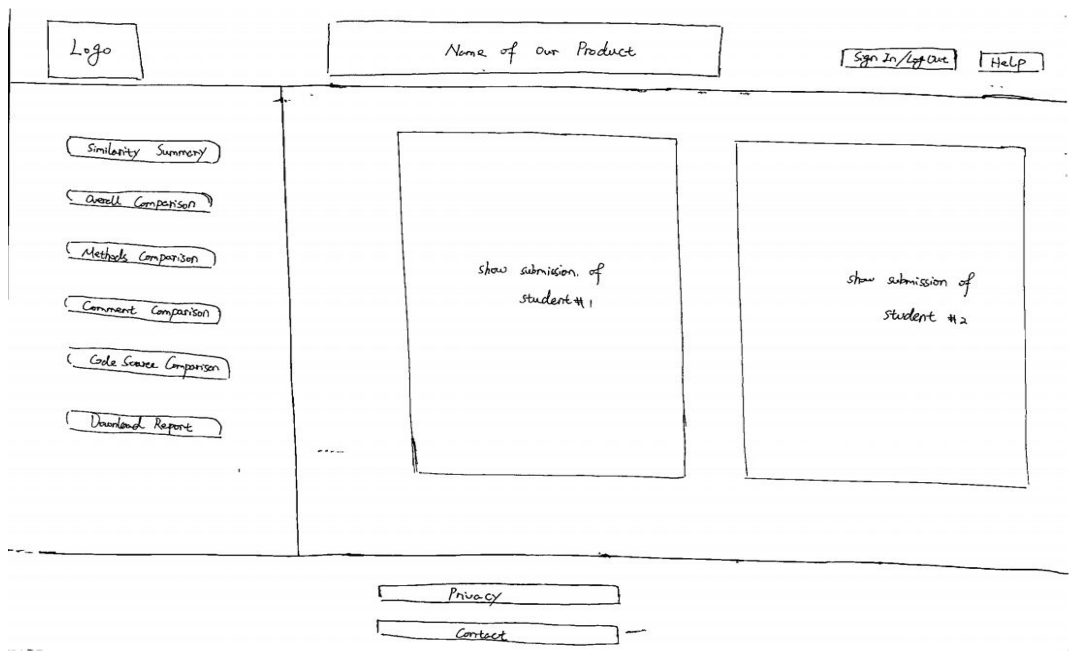
	still does not work, user should find contact information in help page and contact system administrators.
Priority:	Suggested, may be implemented
When Available:	Fourth Increment
Channel to Actor:	Java classes and functions
Secondary Actor:	None
Channels to Secondary Actors:	None
Open Issues:	

Mock-Up of User Interface:

Start Page:



Main Page:


Logo
Name of our Product
Sign In/Log Out
Help
Similarity Summary
Overall Comparison
Methods Comparison
Comment Comparison
Code Source Comparison
Download Report
show submission of student #1
show submission of student #2
Privacy
Contact