PROJECT PHASE A

OBJECTIVE

Source code plagiarism is a severe problem in academia. In academia, programming assignments are used to evaluate students in few courses that involves programming. Therefore, checking programming assignments for plagiarism is essential. If a course consists of a large number of students, it is not possible to check each assignment by instructors. Hence, we are presenting a solution for detecting plagiarism in Python code.

We chose our solution to support detection of plagiarism in Python as it has a solid claim to being the fastest-growing major programming language. The numbers don't lie so here are few statistics which shows the rapid usage of Python.

1. There is 80% increase in the number of Python users from 2012-2017 (June).
2. We can see the growth in traffic of Python across all programming languages/platform here https://goo.gl/VjJfQv

Python can be advantageous in terms of identifying plagiarism as programmatically Python is/has.

1. Readable: Python's syntax is very clear, so it is easy to understand program code. syntax of Python is almost identical to the simplified "pseudo-code" used by many programmers to prototype and describe their solution to other programmers.
2. Easy Syntax: Python's syntax is easy to learn.

WORKING

We intend to build a web application that supports course-wise submission of assignments for students, which will then be tested for plagiarism. Instructors have a portal where they can run plagiarism tests on either the homework files submitted by the students or individual files uploaded by the instructor. Once the test finished running, it displays list of plagiarized files and additional statistics for that file. We have included an additional feature where a click on "Compare" (as shown in mockup) opens a display highlighting the code that is similar in the files. On the backend, we intend to use two algorithms to detect plagiarism of source code. The first one being Naïve Bayes Classifier as this can be handy to identify patterns. Since we do not have any dataset and no training is involved, we plan on using k-Nearest Neighbor algorithm, also known as memory-based learning which is an unsupervised machine learning algorithm.

TECHNOLOGY STACK

The technology stack for the software is shown below.

Frontend	HTML, CSS, Javascript
Backend	Java
Libraries	Weka (machine learning library)
Database	MongoDB

PLAN

We have planned our project for 7 weeks from now and a brief plan is as shown below.

Week and task highlight	Plan and execution
1. Setting up development environment and wireframe of our application.	1. Installation of all IDE such as Webstorm, Eclipse and other related plugins and libraries that are required. 2. Setting up MongoDB. 3. Hosting our project on Heroku so setting Heroku or any hosting environment.
2. Detailed architecture	1. We intend to create UML diagrams which will have all representations and relations between each of the Java class. 2. Start to create interfaces.
3. Development: Basic start	1. Start building the website as decided in the mockup. Getting basic functionalities in place. 2. Start implementing Naïve Bayes classifier.
4. Development: Extend the algorithm	1. Depending on the results obtained by Naïve Baye's classifier, we will further improve it by adding more features or implement a new algorithm. 2. Start working on algorithm for comparing and highlighting similar code.
5. Development and Testing	1. We finish all the development work by this week. 2. We also start testing all the functionalities. 3. Seek feedback from instructors.
6. Final tests and finishing	1. Finish all the pending testing. 2. Work on the feedbacks received.
7. Presentations	Final Presentation

USE CASES

Below are the use cases identified. We have defined three users - Students, Instructors and Admin.

1)

Use Case	User Login
Primary Actor	Instructor and Student
Goal in Context	To access the web portal
Preconditions	User needs to have credentials for logging in
Trigger	User wants to upload assignments or check for plagiarism.
Scenario	1. User wants to use the application 2. User visits the login page 3. User enters username and password 4. Clicks on login button
Exceptions	Login fails as user isn't added to the system • User should request for access Username and password is incorrect • Display that credentials are wrong

2)

Use Case	Role assignment and Adding new user
Primary Actor	Admin
Goal in Context	To add and authorize a user
Preconditions	Admin must be logged in
Trigger	A user isn't registered on the website and needs access.
Scenario	1. Instructor isn't registered on the plagiarism detector website. 2. Instructor needs to check for plagiarism for online submissions 3. Instructor requests admin for login information 4. Admin creates a new user for instructor and assigns the role as "Instructor"
Exceptions	Username already exists • Choose a new username

	Password too weak • A stronger password should be used

3)

Use Case	Running the plagiarism test
Primary Actor	Instructor
Goal in Context	To detect plagiarism between the homework files submitted by students and generating a report of the same.
Preconditions	Instructor should be logged in and students should have uploaded the assignment files.
Trigger	Instructor wants to check for plagiarism between files
Scenario	1. User logs into the app 2. User wants to use the plagiarism app 3. They upload files using the upload feature 4. After uploading, the user clicks on the check button which starts the plagiarism detector 5. After plagiarism detection completes, it will generate a report
Exceptions	User is not able to run the files • Display message saying “files not upload” • Retry uploading

4)

Use Case	View plagiarism report
Primary Actor	Instructor
Goal in Context	To view in-depth report about plagiarism in the homework submissions
Preconditions	Instructor should be logged in and should have run the plagiarism test
Trigger	Instructor has run the plagiarism test and wants to view the in-depth report of plagiarism in the homework submissions.
Scenario	1. Instructor logs into the website 2. Instructor uploads the documents to be checked for plagiarism 3. Instructor receives the list of files where plagiarism is detected

	4. Instructor wants to view the in-depth report of plagiarism for all the plagiarized files.
Exceptions	Disrupted internet connection • Display the message “Resolve any internet connectivity errors and re-run the plagiarism test.”

5)

Use Case	Download Report
Primary Actor	Instructor
Goal in Context	To store the generated plagiarism results
Preconditions	The detection algorithm has finished running
Trigger	Instructor wants to download the report
Scenario	1. User is waiting for the results 2. Once the report is generated, user wants to download the report for later use 3. They click on download button to store the reports
Exceptions	Download fails • Display message “Check internet connectivity”

6)

Use Case	Upload Files
Primary Actor	Student
Goal in Context	To upload documents for submission
Preconditions	User should be logged in
Trigger	User has logged in and needs to submit homework
Scenario	1. Student wants to submit homework 2. Student logs into the website 3. Student selects course and assignment number 4. Student clicks on “Upload” and selects files to be uploaded
Exceptions	Only one file is uploaded

	• Alert user to upload 2 or more files Files are not of same type • Discard uploaded files and request the user to upload files of the same type

7)

Use Case	View snapshot of plagiarized content
Primary Actor	Instructor
Goal in Context	To view and recheck the content (code) that was plagiarized between 2 documents
Preconditions	User should be logged in and must have run the plagiarism test
Trigger	1. Instructor has run the plagiarism test. 2. Instructor wants to see the same/plagiarized content of 2 documents
Scenario	1. Instructor has successfully run the plagiarism test 2. Instructor receives a list of plagiarized files 3. Instructor selects 2 files that have been detected as plagiarized 4. Instructor clicks on “Compare Snapshot” option to view the 2 files side by side with plagiarized content highlighted to compare them manually.
Exceptions	Less than 2 files selected • Alert user to select exactly 2 files to view snapshot of plagiarized content.

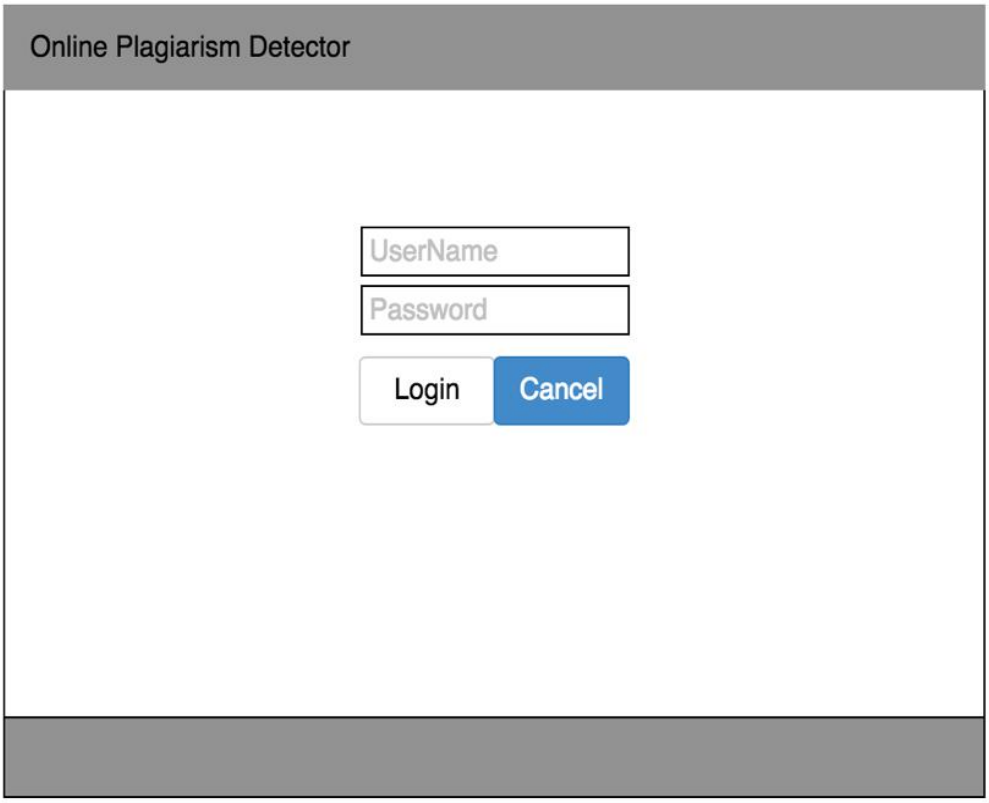
8)

Use Case	Submitting assignments
Primary Actor	Student
Goal in Context	Submission of respective course assignment
Preconditions	There is a course assignment in Python language for submission
Trigger	Student wants to submit the python code
Scenario	1. Student logs in with his student account 2. Student selects course and assignment to submit 3. Student clicks on Upload button to uploads files for the assignment 4. Clicks on submit
Exceptions	Can't upload the files • Display message "Upload failed. Please try again"

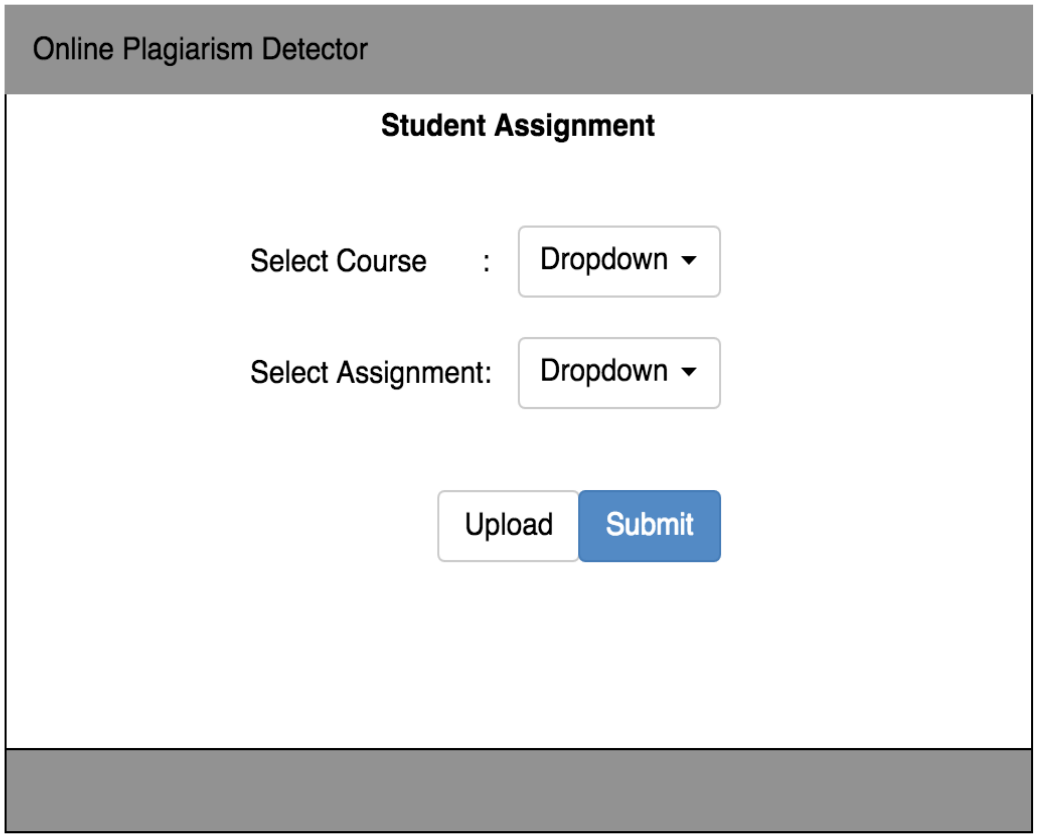
UI- MOCKUP

Designed using *Moqups* which is an online user interface mockup tool

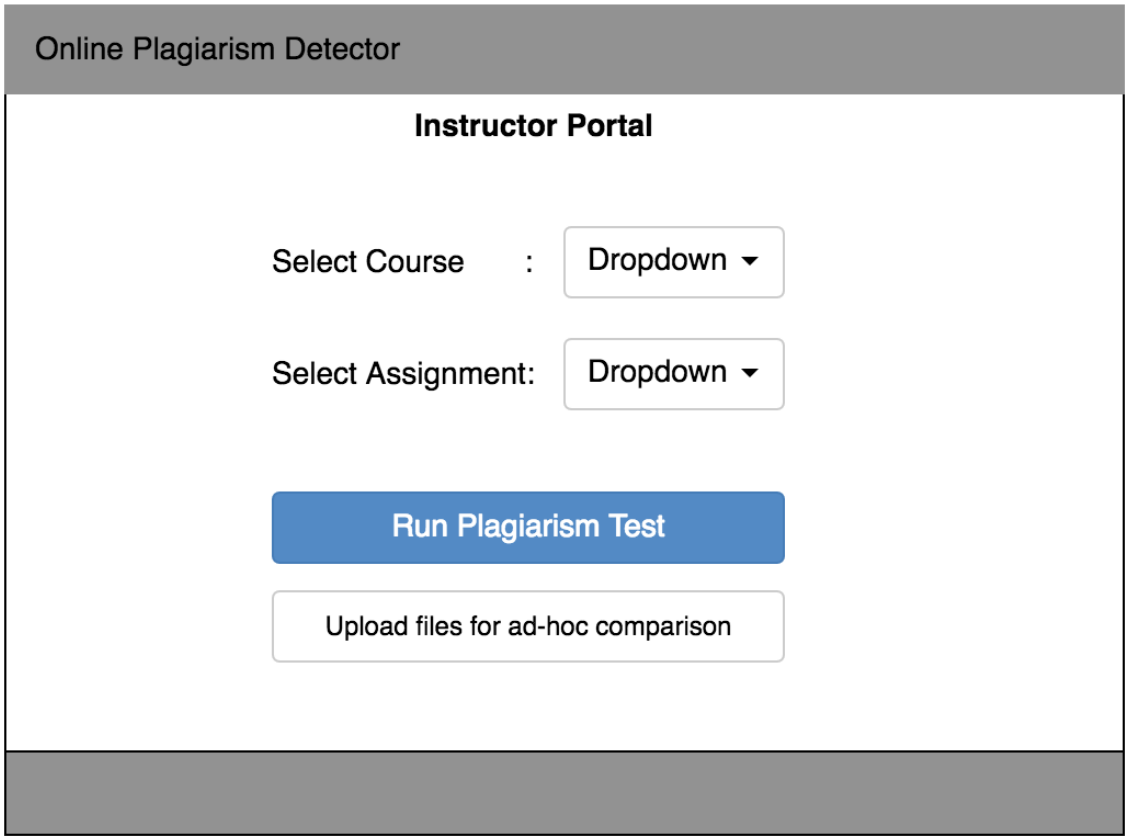
1) Login module.



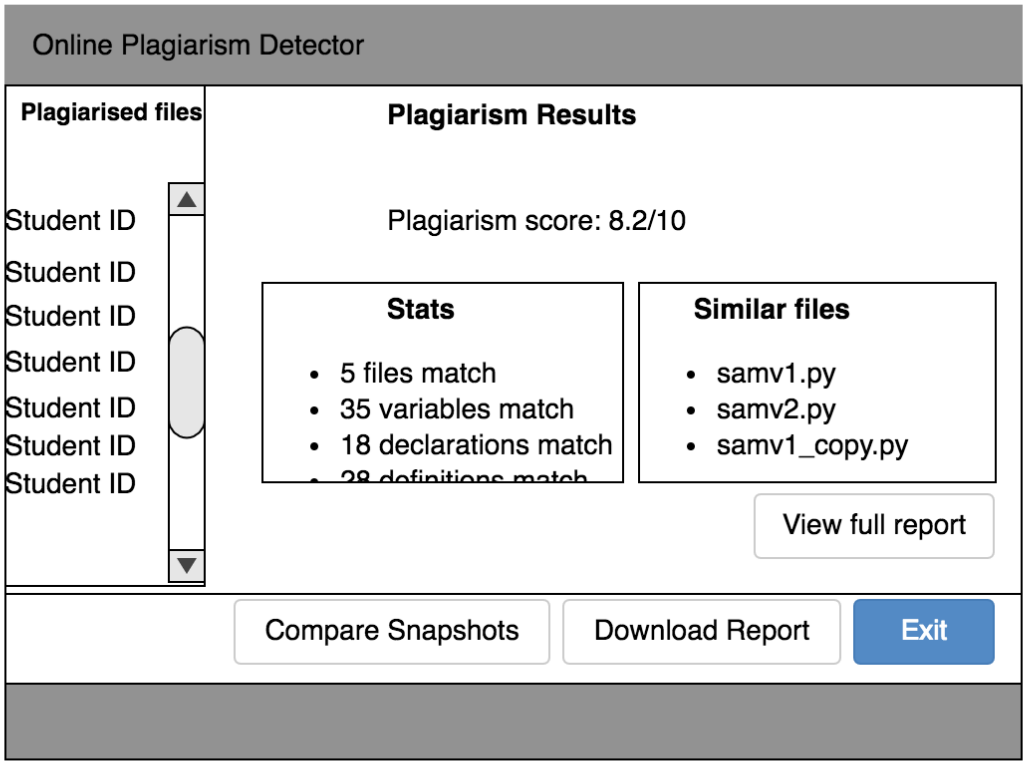
2) Student Portal



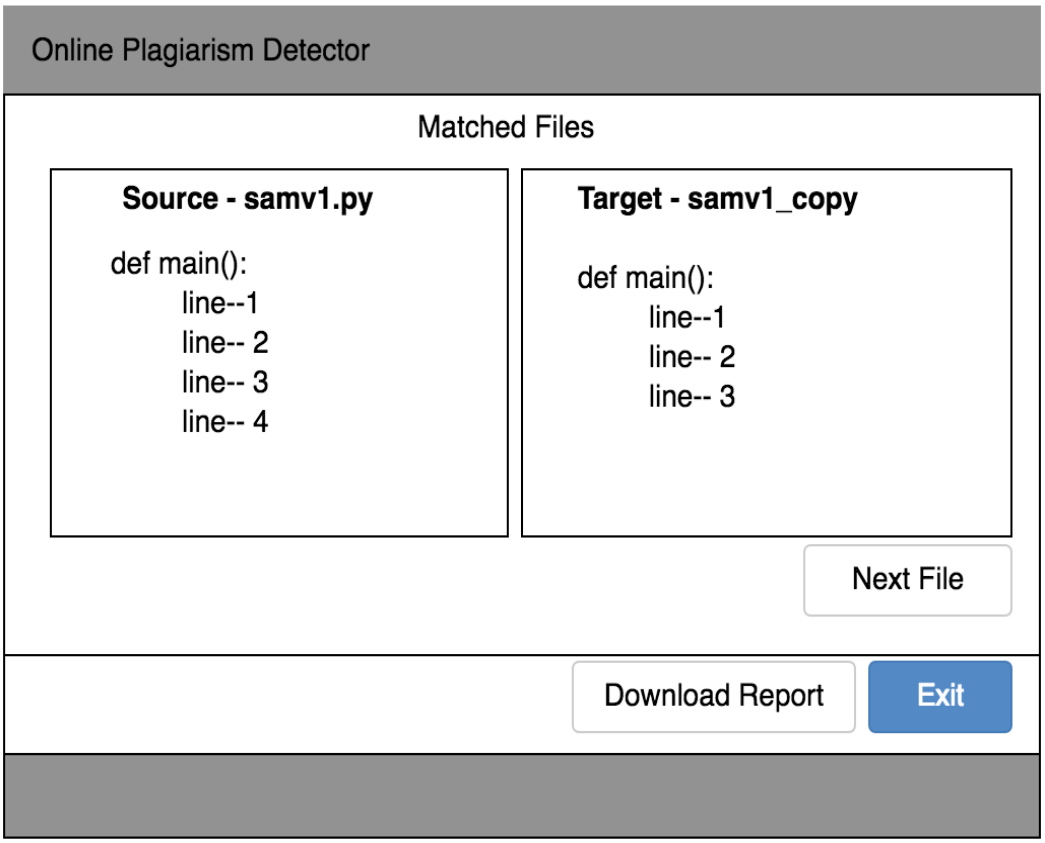
3) Instructor Portal



4) Plagiarism Results



5) Comparing the files and identifying similar Python code.



REFERENCES:

1. https://en.wikiversity.org/wiki/Python_Concepts/Why_learn_Python
2. http://ijmlc.org/papers/50-A243.pdf