

**A Cloud based software to check
Plagiarism**

A PROJECT REPORT

**Submitted as a Jth Component for the course
MCA**

by

Ruchi Mantri

20MCA0132

Dev Sharma

20MCA0129

Arka Seal

20MCA0126

**Under the Guidance of
Prof. Chellatamilan T**



VIT[®]
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

School of Information Technology & Engineering

June, 2021

DECLARATION BY THE CANDIDATE

I hereby declare that the project report entitled "**A Cloud based software to check Plagiarism**" by me to VIT University, Vellore in partial fulfilment of the requirement for the award of the degree of **MCA** is a record of Jth component of project work carried out by me under the guidance of **Prof. Chellatamilan T**. I further declare that the work reported in this project has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.

Place: Vellore

Signature of the Candidate

Date: 6-July-2021

Dev Sharma (20MCA0129)

TABLE OF CONTENTS

1. Introduction

2. Literature Survey and Review

3.1 Literature Summary

3. System Design

4.1 Frame work

4. System Implementation

5.1 Code and/or Architecture Development

5.2 Test Results

5. Results and Discussion

6.1 Output/Results

6.2 Discussion

6. Conclusion

7.1 Conclusion

7. References

1.

INTRODUCTION

1.1 Introductory Remarks

- **Cloud is more of a platform than a technology, a platform which provides the user to implement their ideologies based on infrastructural design, platform design or software design. We intend to use the Software as a Service privileges of Cloud Computing to detect the plagiarism (or similarity index) at a smaller level.**
- **We aim to develop a dynamic website based on the said model to take input from the user in form of text(s) and output the similarity index of that text.**

2.

Literature Survey and Review

[1] At present, accessing and publishing new knowledge are much easier than they were in the past. This has given rise to a “copy-and-paste” culture, which is a violation of intellectual property. A plagiarism checker can help to identify such incident. Unfortunately, existing plagiarism checkers still do not offer services that handle many users at once like what Google’s do. We conjecture that there is a bottleneck that they still need to overcome before making their adoption more widespread. This paper proposes a framework to improve the throughput of existing plagiarism checkers to tackle the bottleneck in question. Our framework leverages stream processing via Apache Storm.

[2] The performance of a plagiarism checker can be prohibitively expensive if the size of the document database to be checked against is large. To improve the checker’s performance, we propose a method that organizes the document database into categories based on a label assigned to the document. The label is derived from simple heuristics applied over the first few pages of the document. For the original document to be checked for possible plagiarism, a set of probability values is assigned based on the likelihood of its belonging to specific categories.

[3] Plagiarism is a serious problem, especially in academia and education. Detecting it is a challenging task, particularly in natural language texts. Many plagiarism detection tools have been developed for diverse natural languages, mainly English. Language-independent tools exist as well, but are considered as too restrictive as they usually do not consider specific language features. In this paper, we introduce APlag, a new plagiarism detection tool for Arabic texts, based on a logical representation of a document as paragraphs, sentences, and words, and new heuristics for text comparison. We describe its main attributes and present the results of some experiments conducted on a dummy test set.

[4] Plagiarism is a serious problem identified amongst research community.

Plagiarism has been around for centuries, but with the use of Internet and easy access to material in electronic format has made it easier to plagiarize materials of others. On the other hand plagiarism detection is now as easy as plagiarizing a document. There are a number of anti-plagiarism software available either freely or commercially, which can be used for plagiarism detection. But the commercial software's are too expensive. So it is not affordable for an individual to purchase those software's. This paper highlights the plagiarism detection software's which are freely available online, that can be downloaded free of cost.

[5] One method to prevent plagiarism is to rely on plagiarism checker application. However, prevention of plagiarism to date in Indonesia still relies on the plagiarism checker held by institutions/organizations outside Indonesia for example Turnitin, Duplin Checker, Copy leaks, Paper Rater, Grammarly, etc. Plagiarism checker applications currently use methods that are applied for English. Development of Plagiarism checker application for Bahasa requires support from multiple disciplines.

[6] This research aims to determine the effectiveness of the implementation of integrated Project Based Learning (PJBL), to integrate Telegram Messenger (TM) and Plagiarism Checker (PC) on student learning outcomes. The sample consisted of 20 students as an experimental class and 20 students as a control class. The experimental class was given a learning treatment using the integrated PJBL model and PC while the control class used conventional models (lectures, discussions, presentations and demonstrations). Data collection techniques used observation, documentation, questionnaires, and tests.

[7] In so many worldly applications, there exists a need to perform spell checking. Spell checking of many words at once is the invention of this work. It is a requirement in activities like: book spell checking before publication, plagiarism detection in a thesis against its references. We briefly list the eleven applications that this project attends with success.

[8] Plagiarism represents a serious and growing problem in science, with only a fraction of such publications detected and retracted. Any initiative to deal efficiently with the problem of plagiarism would require a joint effort of academic publishers and editors. The most effective measure would be to establish a common plagiarism detection system, adopted by all peer-reviewed journals and major publishers, with automatic uploading and cross-checking of each newly submitted manuscript with both published material and all further and ongoing submissions.

[9] In this paper a plagiarism detection framework is proposed based on coding style. Furthermore, the typical style based approach is improved to better detect plagiarism in programming codes. The plagiarism detection is performed in two phases: in the first phase the main features representing a coding style are extracted. In the second phase the extracted features are used in three different modules to detect the plagiarized codes and to determine the giver and takers of the codes. The extracted features for each code developer are kept in a history log, i.e. a user profile as his/her style of coding, and would be used to determine the change in coding style.

[10] A big part of lifelong learning is the move from residential lectures to distance education. Distance education falls under the multi-modal policy of the teaching institution and thereby a change in student contact. The lecturer facilitating the distance education course is also faced with a problem where the quality and originality of submitted assignments need to be checked. This has always been a difficult task, as going through practical assignments and looking for similarities is a tedious job. Software checkers are available, but as yet, have not been integrated into popular online e-learning systems.

3.

System design

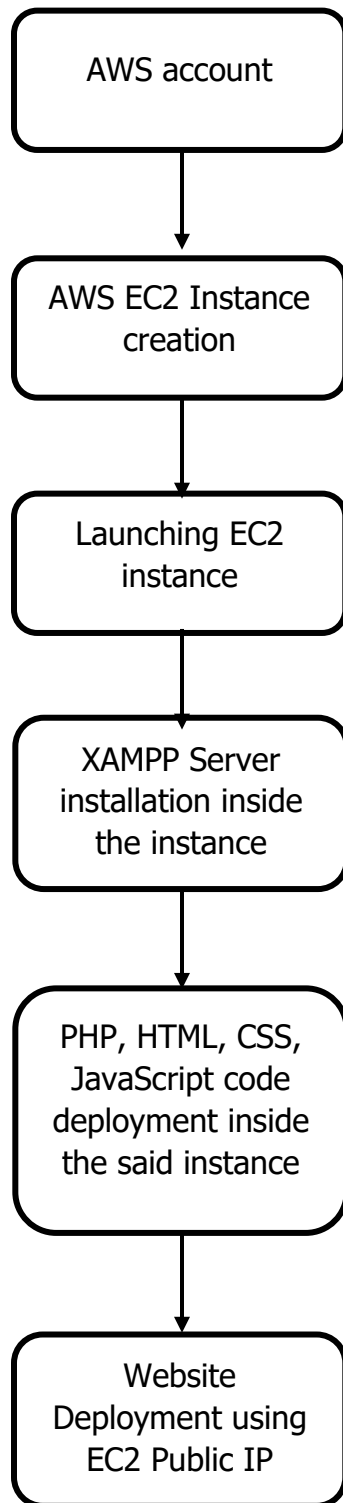
3.1 Frame work

The System Design include a code development for our desired Plagiarism web software and its deployment on a Cloud platform.

Code development requirements include:

- AWS Console access for required AWS elements.
- AWS EC2 instance developments with required Security Rules and Roles.
- VSCode for PHP, HTML, CSS and JavaScript coding.
- XAMPP Server for local hosting of web server at localhost.
- Public IP for deployment.

Flow Chart:



4.

System Implementation

4.1 Code and/or Architecture Development

Code for the developed Plagiarism Similarity Index checker is coded below in VSCode and later it is deployed on AWS cloud platform component EC2.

Cloud environment of AWS was used for mainly these factors:

- Easy public deployment of the web application using EC2 instance.
- S3 bucket usage for increased size of documents storing which may require while comparing such sensitive documents of related inter-domains. (This feature is not included now as we use only 3 documents for comparing as a test working model which was sufficient in those 30GB of EC2 windows virtual computer of free tier)
- AWS Lambda functions and other requisites for API development for developing our implementation as a service on request; this is also not developed at present. It can be act as a future prospect with further study on these basis.

Implementing code:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
  <style type="text/css">

body{
  font-size:20px;
  font-weight:bold;
```

```

font-family: 'Times New Roman', Times, serif;
background-image:url("pp.jpg");
background-repeat:no-repeat;
background-position:center;
background-size:cover;
}
h3{
font-size:40px;
display: inline;
color:red;
text-decoration: underline;
font-style:normal;
font-family: 'Times New Roman', Times, serif;
}
p{
font-size:15px;
display:inline;
font-weight:bolder;
position: relative;
left:80px;
top:55px;
font-size:25px;
color:blue;
}

input[type=file]{
background-color:rgb(241, 241, 143);
border:none;
color:white;
padding:16px 20px;
margin:4px 2px ;
cursor:pointer;
color:black;
}

```

```

input[type=submit]{
    background-color:green;
    border:none;
    color:white;
    padding:16px 20px;
    margin:4px 2px ;
    cursor:pointer;
}

.mane{
    background-color: rgb(0,0,0,0.5);
    width:800px;
    margin: auto;
    color: white;
}

a:hover{
    background-color:yellow;
}

</style>

</head>
<div class="mane">
<center>
<h3 style="">Plagiarism Checker</h3>
</center>
<div>
<body>
    <form action="?" method="post" enctype="multipart/form-data">
        <p> Select File:</p>
        <center> <div class="mane">
            <a><input type="file" name="file"/></a>
            <input type="submit" value="upload " name="upload"/>
        </form>

<?php

```

```

if(isset($_POST['upload'])){

    $file_name=$_FILES['file']['name'];
    $file_type=$_FILES['file']['type'];
    $file_size=$_FILES['file']['size'];
    $file_temp_loc=$_FILES['file']['tmp_name'];

    $file_store="uploads/".$file_name;
    $imageFileType = strtolower(pathinfo($file_name,PATHINFO_EXTENSION));
    $uploadok=1;
    echo"<br>";

    if ( $_FILES["file"]["size"] > 500000) {
        echo "Sorry, your file is too large.<br>";
        $uploadOk = 0;
    }

    if($imageFileType != "txt" ) {
        echo "Sorry, only txt files are allowed.<br>";
        $uploadOk = 0;
    }
    if($uploadok==1){
        move_uploaded_file($file_temp_loc,$file_store);
    }
    else{
        echo "not uploaded";
    }
}
?>

<br>
<br>

<?php

```

```

$files=glob("*.txt");
$arr=$files;

if(isset($_POST['upload']))
{
    $cc=0;
    $input=file_get_contents($file_name);
    $in=explode(" ", $input);
    $strng=Array();
    $kk=0;
    for($j=0;$j<count($arr);$j++)
    {
        $contents=file_get_contents($arr[$j]);
        $cc=0;
        echo "found matches in document ".$arr[$j].": "<br> "<br>";
        for($i=0;$i+3<=count($in);$i++)
        {
            $wr=array_slice($in,$i,3);
            $s=implode(" ", $wr);
            $pattern=preg_quote($s, '/');
            $pattern="/$pattern/";
            if(preg_match_all($pattern, $contents, $matches))
            {
                $ss=implode("\n", $matches[0]);
                $cc+=str_word_count($ss);

                echo '<span style="color:red;">'. $ss. " "</span>';

                if(count($in)-$i>=2)
                    $i+=2;

                $jj=$i;
                while(true)
                {
                    $jj+=1;

```

```

        $ex_wr=array_slice($in,$jj,1);
        $ex_s=implode(" ",$ex_wr);
        $ex_pattern=preg_quote($ex_s, '/');
        $ex_pattern="/$ex_pattern/";
        if(preg_match($ex_pattern,$contents,$ex_matches))
        {
            $ex_ss=$ex_matches[0];
            $cc+=str_word_count($ex_ss);
            echo '<span style="color:red;">'.$ex_ss." ".'</span>';
        }
        if($cc>=count($in)||empty($ex_matches))
            break;
    }
    $i=$jj-1;
}
else
{
    if($i+3==count($in))
        echo " ".$in[$i]." ".$in[$i+1]." ".$in[$i+2];
    elseif($i+3>count($in))
        echo $i.$in[$i+1];
    else
        echo $in[$i]."\n";
}
}
}
$prcnt=$cc/count($in)*100;
array_push($strng,$prcnt);
echo "<br>". "total matches: " .($prcnt)."%". "<br>". "<br>";
}
echo"<br>";
echo"<br>";
rsort($strng);
$avg=0;
for($k=0;$k<count($strng);$k++)

```

```

{
    $avg+=$strng[$k];
}
$avg=$avg/count($strng);

echo'<span style="color:green;font-
size:24px;">'. "Maximum Similarity Index: ".$strng[0]."%". '</span>'. "<br>";
echo'<span style="color:green;font-
size:30px;">'. "Cumulative Similarity Index: ".$avg."%". '</span>'. "<br>";
}
?>
</center>
</div></body></html>

```

Setting up AWS EC2 instance as an Infrastructure as a Service

The screenshot shows the AWS Management Console interface for the Asia Pacific (Mumbai) Region. The left sidebar contains navigation links for EC2 Dashboard, Events, Tags, Limits, Instances, Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity Reservations, Images, AMIs, Elastic Block Store, Volumes, Snapshots, and Lifecycle Manager.

The main content area is divided into several sections:


- Resources:** A table showing the number of resources used in the region:

Resource	Count
Instances (running)	3
Elastic IPs	0
Key pairs	1
Placement groups	0
Snapshots	0
Dedicated Hosts	0
Instances	4
Load balancers	0
Security groups	5
Volumes	4
- Account attributes:** A section showing account details such as Supported platforms, VPC, Default VPC, Settings, EBS encryption, Zones, Default credit specification, and Console experiments.
- Launch instance:** A section with a 'Launch instance' button and a note: 'To get started, launch an Amazon EC2 instance, which is a virtual server in the cloud. Note: Your instances will launch in the Asia Pacific (Mumbai) Region.'
- Service health:** A section showing the status of the service, indicating it is 'operating normally' in the Asia Pacific (Mumbai) Region.
- Zones:** A table showing the available zones in the region:

Zone name	Zone ID
Asia Pacific (Mumbai) Zone A	ap-south-1a
Asia Pacific (Mumbai) Zone B	ap-south-1b
Asia Pacific (Mumbai) Zone C	ap-south-1c
- Explore AWS:** A section with promotional content, including 'Get Up to 40% Better Price Performance' and '10 Things You Can Do Today to Reduce AWS Costs'.

Selecting required Roles and Adding security protocols as per required

Step 1: Choose an Amazon Machine Image (AMI)


**Ubuntu Server 20.04 LTS (HVM), SSD Volume Type** - ami-0c1a7f89451184c8b (64-bit x86) / ami-0d18acc6e813fd2e0 (64-bit Arm)

Free tier eligible

Root device type: ebs Virtualization type: hvm ENA Enabled: Yes

Select

64-bit (x86)
64-bit (Arm)


**Microsoft Windows Server 2019 Base** - ami-034a4d85b5ef5e779

Windows
Free tier eligible

Microsoft Windows 2019 Datacenter edition [English]
Root device type: ebs Virtualization type: hvm ENA Enabled: Yes

Select


64-bit (x86)

**Are you launching a database Instance? Try Amazon RDS.**

Amazon RDS

Amazon Relational Database Service (RDS) makes it easy to set up, operate, and scale your database on AWS by automating time-consuming database management tasks. With RDS, you can easily deploy **Amazon Aurora**, **MariaDB**, **MySQL**, **Oracle**, **PostgreSQL**, and **SQL Server** databases on AWS. **Aurora** is a MySQL- and PostgreSQL-compatible, enterprise-class database at 1/10th the cost of commercial databases. [Learn more about RDS](#)

Launch a database using RDS


**Microsoft Windows Server 2019 Base with Containers** - ami-02411353e28c5e518

Windows
Free tier eligible

Microsoft Windows 2019 Datacenter edition with Containers [English]
Root device type: ebs Virtualization type: hvm ENA Enabled: Yes

Select

64-bit (x86)

**Microsoft Windows Server 2019 with SQL Server 2017 Standard** - ami-0075b383709465bea

Windows

Microsoft Windows 2019 Datacenter edition, Microsoft SQL Server 2017 Standard [English]

Select

64-bit (x86)

Cancel and Exit

ap-south-1.console.aws.amazon.com/ec2/v2/home?region=ap-south-1#LaunchInstanceWizard

Services Search for services, features, marketplace products, and docs [Alt+S]

1. Choose AMI 2. Choose Instance Type 3. Configure Instance 4. Add Storage 5. Add Tags 6. Configure Security Group 7. Review

Step 6: Configure Security Group

A security group is a set of firewall rules that control the traffic for your instance. On this page, you can add rules to allow specific traffic to reach your instance. For example, if you want to set up a web server and allow Internet traffic to reach your instance, add rules that allow unrestricted access to the HTTP and HTTPS ports. You can create a new security group or select from an existing one below. [Learn more](#) about Amazon EC2 security groups.


Assign a security group: ☒ Create a new security group
☐ Select an existing security group

Security group name:

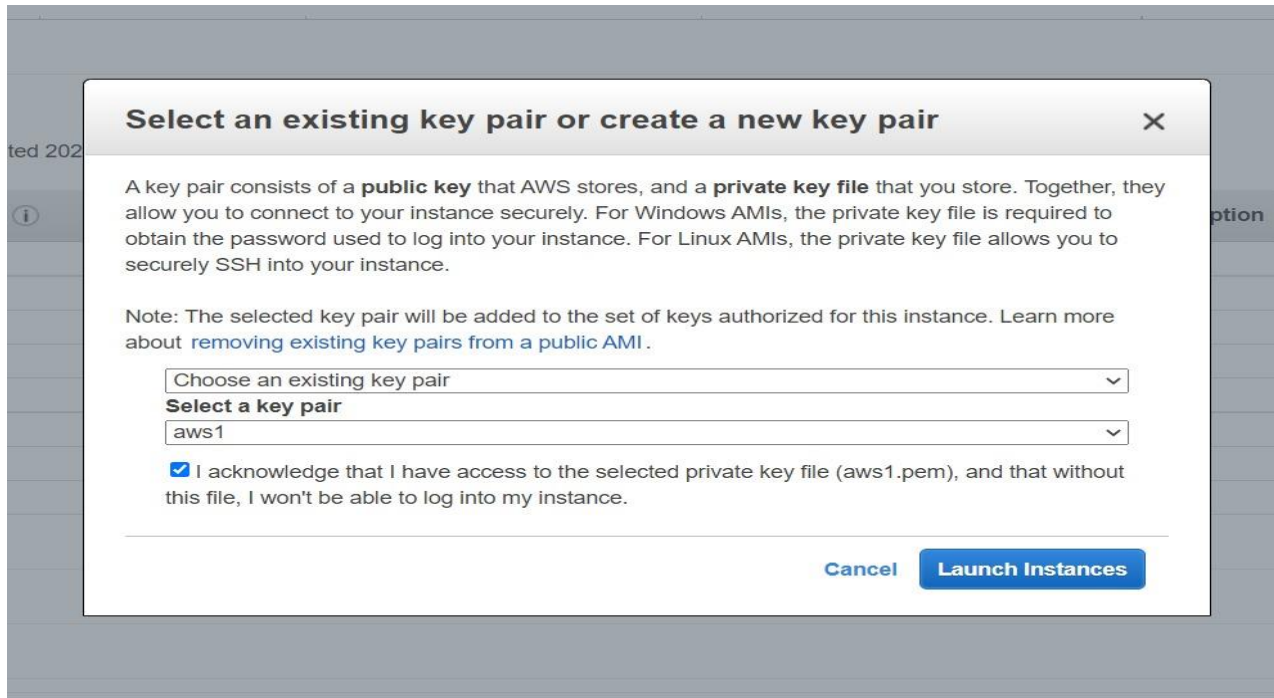
Description:

Type	Protocol	Port Range	Source	Description
RDP	TCP	3389	Anywhere 0.0.0.0/0 :::0	e.g. SSH for Admin Desktop
HTTP	TCP	80	Anywhere 0.0.0.0/0 :::0	e.g. SSH for Admin Desktop
HTTPS	TCP	443	Anywhere 0.0.0.0/0 :::0	e.g. SSH for Admin Desktop
SSH	TCP	22	Anywhere 0.0.0.0/0 :::0	e.g. SSH for Admin Desktop

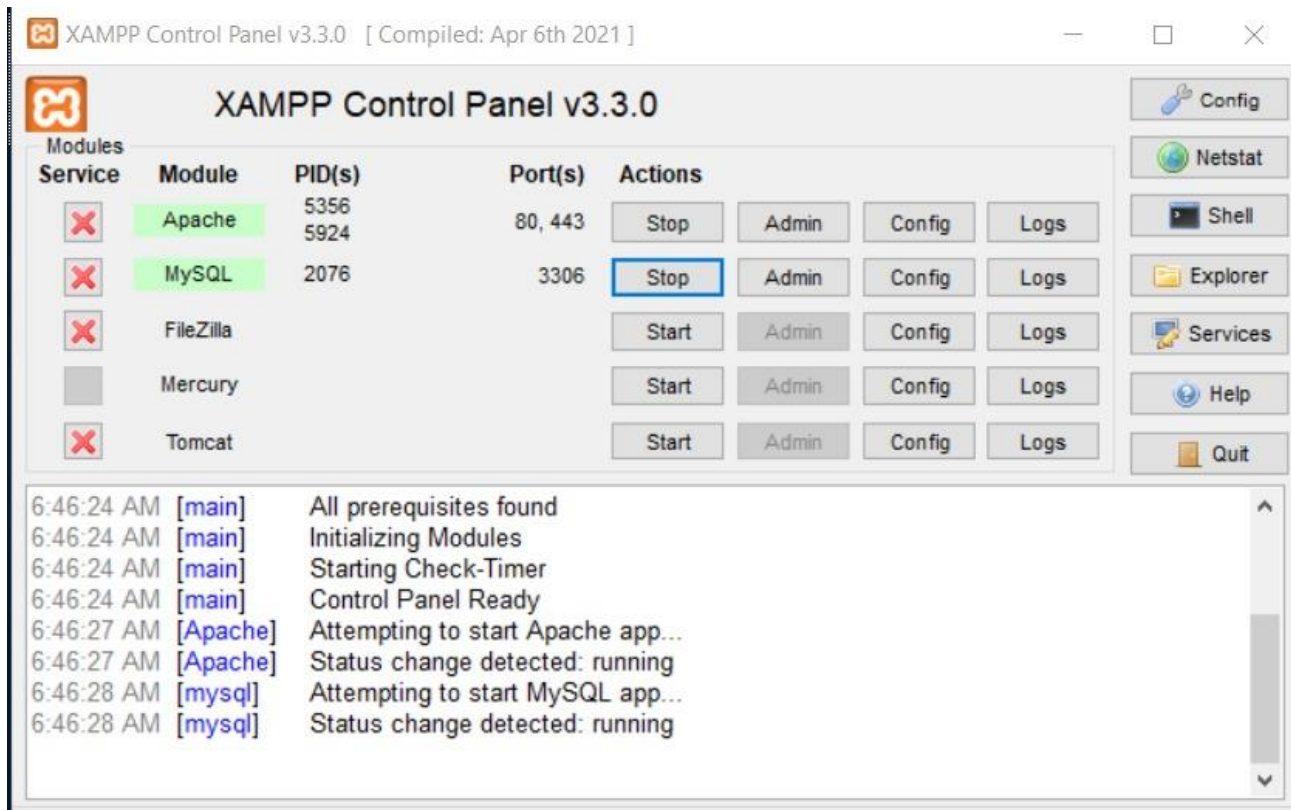
Add Rule

 **Warning**
Rules with source of 0.0.0.0/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only.

Cancel Previous Review and Launch

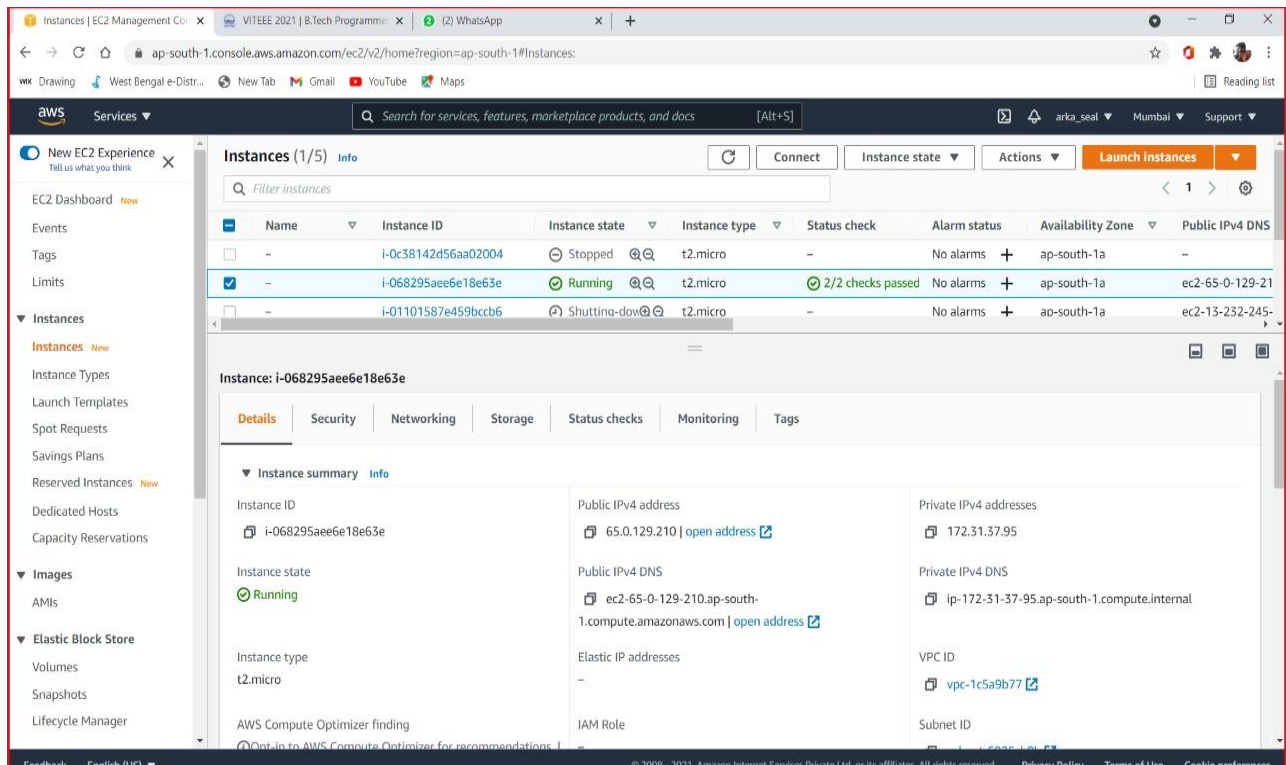


Setting up XAMPP Server inside the Virtual EC2 Instance Window

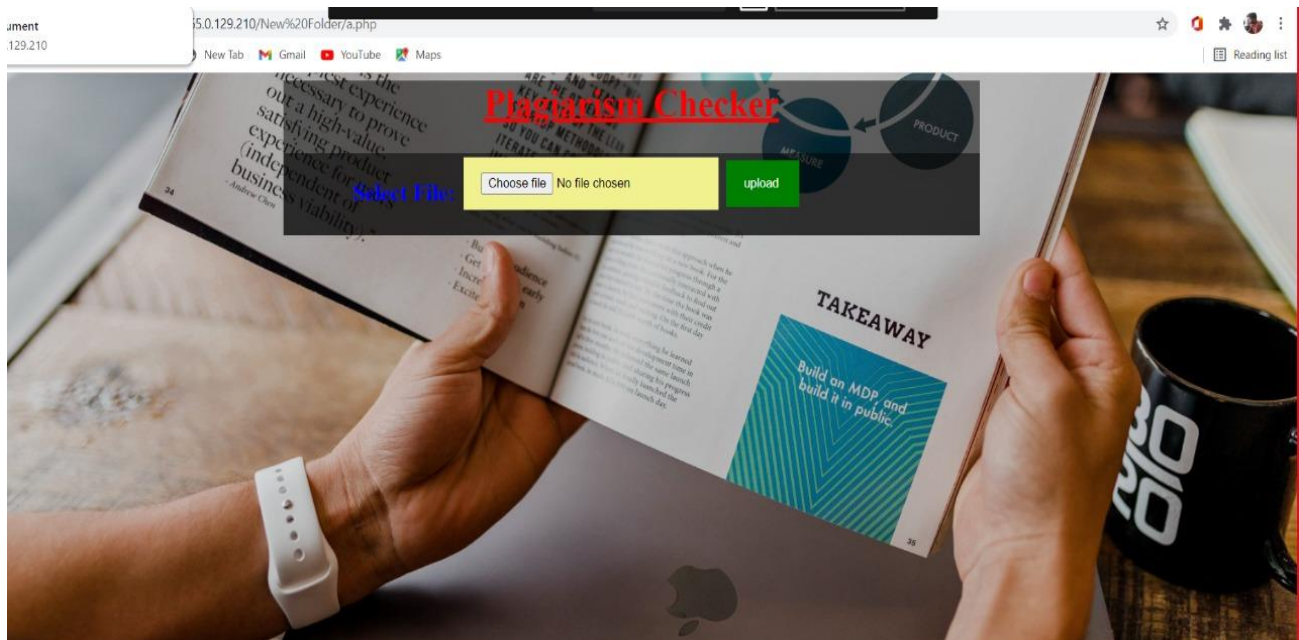


4.2 Test Results

Running instance of EC2 with required Roles and Securities having Public IP and Private IP



Deployment of our Plagiarism checking Website



Document x +

Not secure | 65.0.129.210/New%20Folder/a.php?

wx Drawing West Bengal e-Distr... New Tab Gmail YouTube Maps Reading list

Plagiarism Checker

Select File: Choose file No file chosen upload

found matches in document 1.txt:

My name is Ruchi Mantri. My Roll Number is
total matches: 70%

found matches in document 2.txt:

My name is Ruchi Mantri. My Roll Number is
total matches: 70%

found matches in document 3.txt:

My name is Ruchi Mantri. My Roll Number is 20MCA0132.
total matches: 100%

found matches in document b.txt.txt:

My name is Ruchi Mantri. My Roll Number is 20MCA0132.
total matches: 0%

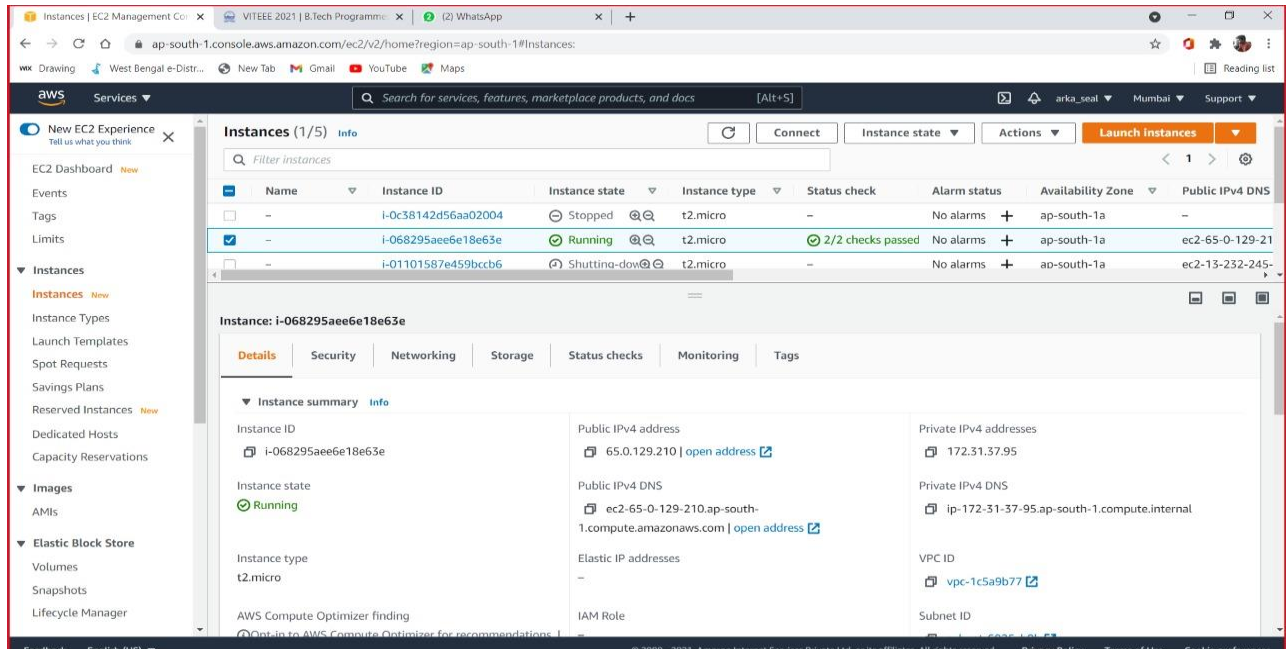
Maximum Similarity Index: 100%
Cumulative Similarity Index: 60%

5.

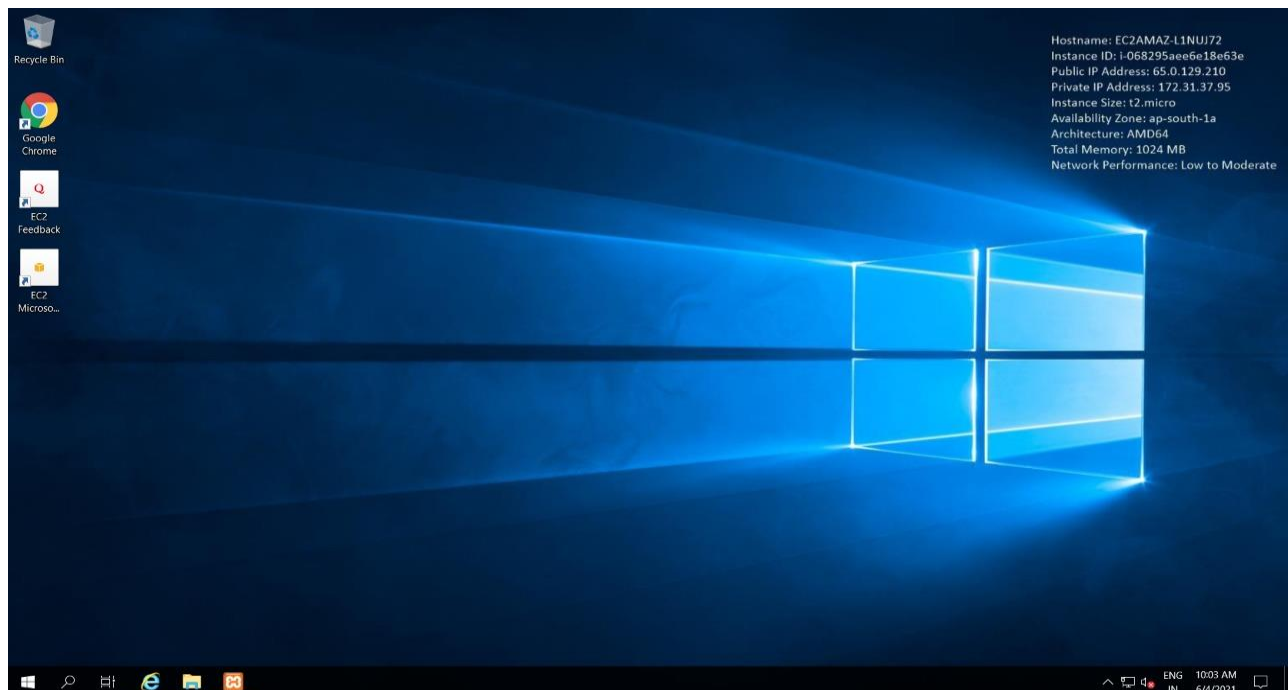
Results and Discussion

5.1 Output/Results

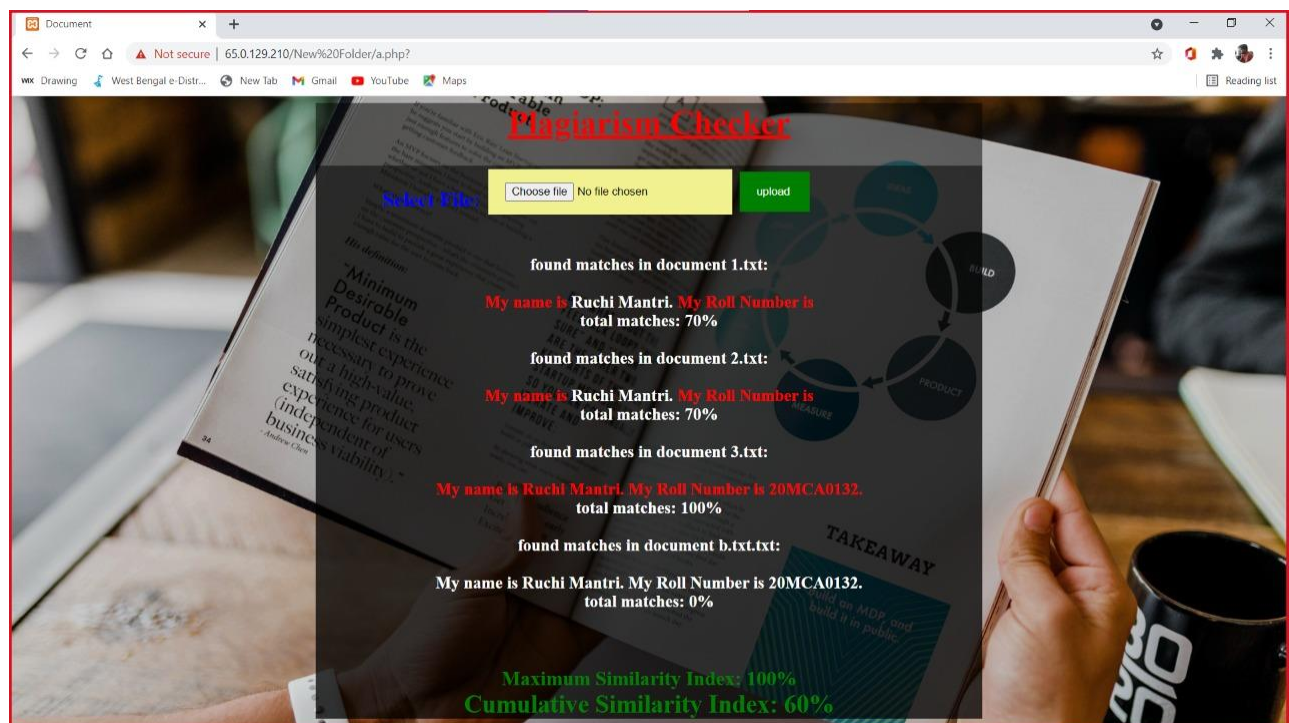
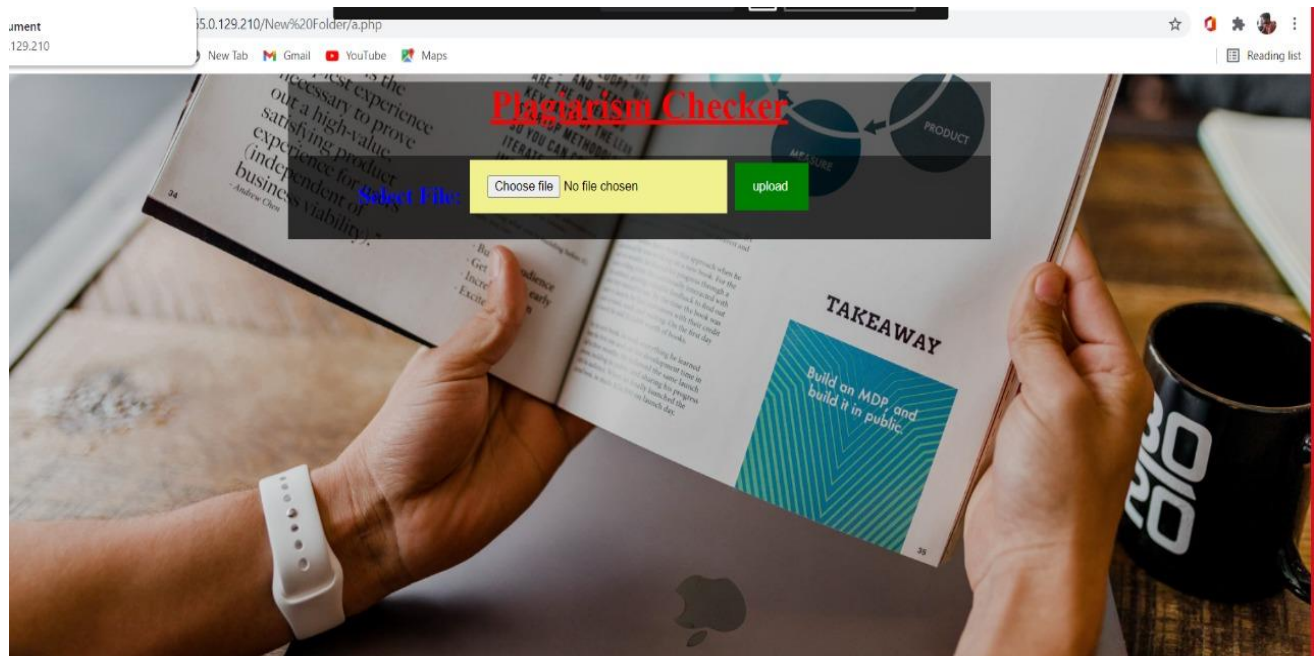
AWS EC2 running Instance



AWS EC2 Virtual Windows Computer



Publicly hosted web application for Plagiarism checking



5.2 Discussion

Our implementation is about developing a website for Similarity index checking using Cloud Infrastructure. Similarity index is developed using our own Algorithm which checks for similarity if consecutively 3 or more words are matched with the contained documents.

On comparing with other Plagiarisms, ours' provide a more specific and detailed oriented easy way to check the similarities as per words and percentage with each respective documents with which similarities are found, with a Similarity Index report at the bottom. For the time being it is worked out with only 4 sample papers but using a real-time google API document retriever it can be actualized as a more precise working model for Plagiarism report checker. And as per required we can modify the code for 7-8 consecutive words comparing.

6.

Conclusion

7.1 Conclusion

As a conclusion it can be said that our Plagiarism viewer is better compared to representation of others as it displays all the text similarities separately without any hustle with its respective matched documents. But as a future scope it can be well mentioned that our implementation can be optimized to use Cloud resources to its widest extent by utilizing features like AWS Lambda, S3 Bucket, etc. and also the implementation as a service deserves mention.

7.

REFERENCES

Journal/Articles

1. Thanayut Seetongchuen, Paruj (2019) 'Improving Plagiarism Checker Throughput with Apache Storm', 16 International Conference on Electrical Engineering Electronics, Computer (ECTI-CON 2019)
2. Chanchana Sornsoontorn, Sunisa Rimcharoen, and Nutthanon Leelathakul, Asanee Kawtrakul and Paruj Ratanaworabhan (2017) 'Using Document Classification to Improve the Performance of a Plagiarism Checker:', 21st International Computer Science and Engineering Conference ICSEC(2017)
3. Mohamed El Bachir Menai, Manar Bagais (2011) 'APlag: A Plagiarism Checker for Arabic Texts', The 6th International Conference on Computer Science & Education August 3-5, 2011. SuperStar Virgo, Singapore (ICCSE 2011)
4. Richa Tripathi, Puneet Tiwari, K. Nithyanandam (2015) 'Avoiding Plagiarism in Research through Free Online Plagiarism Tools', 4th International Symposium on Emerging Trends and Technologies in Libraries and Information Services(2015)
5. Wayan Gede Suka Parwita (2019) 'String Matching Based Plagiarism Detection for Document in Bahasa Indonesia', 5th International Conference on new media studies Bali, Indonesia (2019)
6. H Pratama and I Prastyaningrum (2019) 'Effectiveness of the use of Integrated Project Based Learning model, Telegram messenger, and plagiarism checker on learning outcomes', IOP Conf. Series: Journal of Physics: Conf. Series 1171 (2019)

Conference Papers

7. Leena Alhussaini (2012) 'Applications of ZIPPER, the Holistic Spell Checker Software', 3rd International Conference on System Science, Engineering Design

and Manufacturing Informatization (2013)

8. Ivan Jaric (2015) 'High time for a common plagiarism detection system', 1 September 2015 / Published online: 23 September (2015)
9. S. Arabyarmohamady, H. Moradi, M. Asadpour (2012) 'A Coding Style-based Plagiarism Detection', International Conference on Interactive Mobile and Computer Aided Learning (IMCL) (2012)
10. Emil Marais, Ursula Minnaa, David Argles (2006) 'Plagiarism in e-learning systems: Identifying and solving the problem for practical assignments', Proceedings of the Sixth International Conference on Advanced Learning Technologies (ICALT'06)