A Mid-Term Progress Report

on

PLAGIARISM DETECTION USING MACHINE LEARNING

Submitted in partial fulfilment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

COMPUTER SCIENCE AND ENGINEERING

SUBMITTED BY

SAHIL GULERIA (2104173)

HARNOOR SINGH (2203582)

KAUSHAL KUNDAN (2203586)

UNDER THE GUIDANCE OF

DR. HARDEEP SINGH KANG

(April-2025)

Department of Computer Science and Engineering

GURU NANAK DEV ENGINEERING COLLEGE,

LUDHIANA

INDEX

LIST OF FIGURES

LIST OF TABLES

Chapter 1

Introduction

The act of plagiarism is stealing another individual's intellectual property, which comprises their literary work, creative output or ideas. The act of claiming these as ones own without due acknowledgment and consent constitutes a direct violation of the principles of academic integrity, honesty and originality. Without doubt the practice has become increasingly prevalent across myriad settings be it essay writing, research reports or artistic works a reality indicating we overlook.

The origin of the word "Plagiarism" is from Latin, which means "Kidnapper" or "to kidnap". Plagiarism detection plays an essential role in academics because students, research scholars and teachers are intended to produce original research work.

A key reminder here though: while using machine learning based technologies to create content may seem like a grey area doing so without an assigned license for assessment leads it into the realm of academic dishonesty. Moreover, reusing one's own work without proper citation is still considered an act of plagiarism. In light of all this remember that exams have strict rules prohibiting instances of plagiarism which carry serious disciplinary consequences.

Learning and using the rules of good academic practice from the start of your university career is the greatest method to prevent plagiarism. Avoiding plagiarism involves using your academic skills to make your work as good as it can be, not just making sure your references are all accurate or changing enough terms so the examiner won't catch you paraphrasing.

In order to deter and fight plagiarism, educational institutions and professional communities frequently have severe regulations and sanctions in place. It is crucial to always correctly attribute and reference the sources utilized, whether through direct quotations, paraphrasing, or summarizing, as well as to adhere to the particular citation style requirements specified by the institution or publication, in order to prevent plagiarism.

The goal of this project is to develop a free plagiarism detector for use in research and teaching. These programmes use sophisticated algorithms and language analysis approaches to find patterns and similarities in texts, from straightforward word-for-word plagiarism to more complex plagiarism techniques like paraphrasing or mosaic writing.

Although they are useful tools, plagiarism checkers also have some restrictions. Firstly, they are not widely available, and the available ones are quite expensive, making it difficult for the common people to use them to their maximum use. Additionally, it may be necessary to use human judgement to distinguish between actual plagiarism and properly referenced excerpts because false positives and false negatives may occur.

To assure originality and uphold ethical standards in academic and professional work, using plagiarism detectors and developing strong research and writing practices are helpful ways.

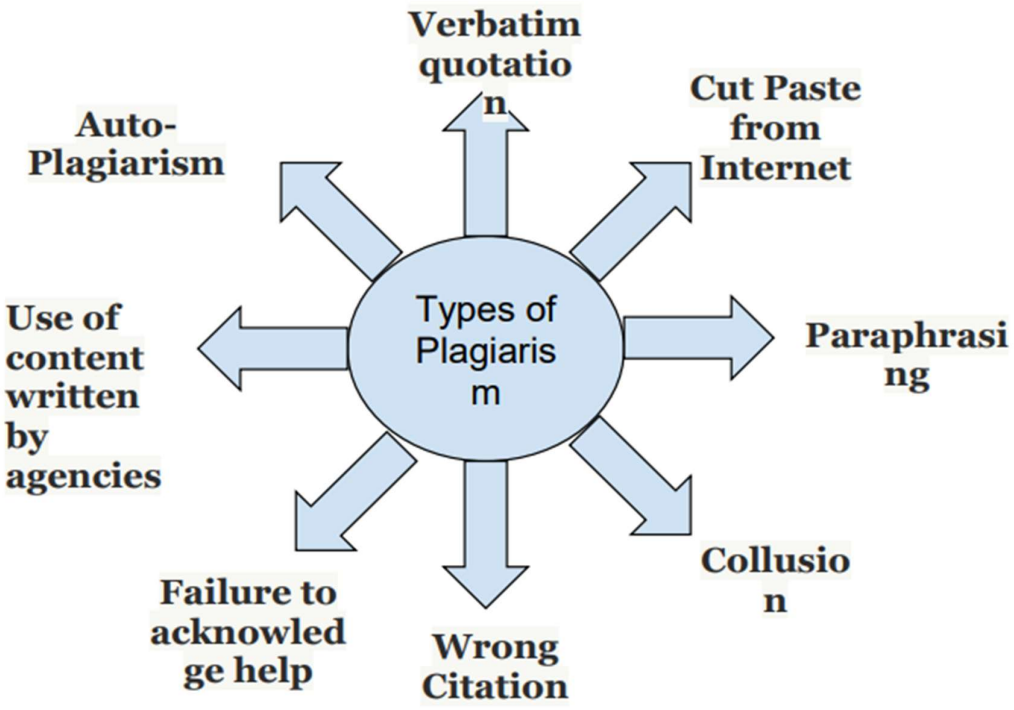

Figure 1.1 - Types of Plagiarisms

Below here, are listed 8 major types of Plagiarism practices that are commonly used by the students and researchers:

1. **Verbatim quotation:** Verbatim quotation: Statements must be identified as such with appropriate citations of the original sources and either quote marks or indentation. Always be upfront with the reader about which concepts and words you have taken directly from another source and which ones you have created yourself.

2. **Cut Paste from internet:** Any information collected from the Internet must be properly referenced and listed in the bibliography. It is vital to carefully assess anything

you read online because online content is less likely to have received the same comprehensive review as traditional sources.

3. **Paraphrasing:** Paraphrasing constitutes plagiarism if the author whose work you are using is not properly acknowledged. This entails altering a few words and their placement in addition to strictly following the argument's outline.

4. **Collusion:** Students may collaborate without permission, neglect to acknowledge assistance received, or fail to precisely follow the guidelines for group work projects. It is your responsibility to ensure that you are clear on how much collaboration is permitted and which parts of the work must be entirely unique.

5**. Wrong Citation:** According to the requirements of your field, it's essential to properly cite sources. Not only must you mention your sources (i.e., provide a bibliography), but you also need to identify the sources in a footnote or in-text citation. Additionally, you must not include in your bibliography any sources or things that you have not actually consulted.

6**. Failure to acknowledge help:** Every contribution made to the creation of your work, including ideas from other students, lab staff, and outside sources, must be acknowledged. It is crucial to include any further guidance that leads to material changes in the content or approach, even if it is unrelated to your tutoring or supervision or to simple modification.

7. **Use of content written by agencies:** You should never submit writing done for you by a professional company or use their services in the creation of your work, even with the author's approval. This study procedure must be carried out alone because it is essential to your intellectual growth and advancement.

8. **Auto-Plagiarism:** You may not submit work for assessment that you previously submitted (in full or in part), either for your present course or for another qualification of this or any other institution, unless specifically mentioned in the particular regulations for your course. Any earlier work of yours that is publishable or citable must be specifically cited. Similar works that have been submitted in a similar way will also be considered auto-plagiarism.

In conclusion, plagiarism checkers are essential for keeping ethical norms and preserving academic integrity. These methods use technology to help in the prevention and detection of

plagiarism, fostering an originality and intellectual honesty culture. To ensure that plagiarism checkers remain relevant in the changing environment of scholarly communication and information sharing, additional study and development in this area are crucial.

Objectives

1. To develop a web-based tool to detect plagiarism and provide report to the user.

The web-based plagiarism detection tool will analyze submitted content by comparing it against a vast sources to identify similarities. It will generate a detailed report highlighting, their sources, and provide a percentage of originality for the document. The tool aims to assist users in ensuring academic or content integrity.

2. To compare with available open-source plagiarism tools.

To compare with available open-source plagiarism detection tools, the project will evaluate the performance and accuracy, existing tools like Turnitin, Plagscan, and others. This comparison will help identify strengths and weaknesses in current solutions and inform the development of more efficient or user-friendly features for the new tool.

3. To study different algorithm in machine learning such as cosine similarity.

The project will study various machine learning algorithms, with a focus on methods like Cosine Similarity, which is commonly used for text comparison in plagiarism detection. By analyzing how Cosine Similarity measures the cosine of the angle between two vectors in a multidimensional space, the study will evaluate its effectiveness in identifying semantic similarity between documents.

Chapter 2

System Requirements

Hardware Required :

- Processor: 2.4 GHZ
- RAM: Minimum 4 GB
- Hard Drive: Maximum 1 TB
- System: Intel Core i3 or Ryzen 3

Software Required :

- Operating System: Windows 10,11
- Programming language: Python
- Programming language: Python
- Libraries: numpy, scikit, seaborn, matplotlib.
- IDE: Visual Studio Code.

Chapter 3

System Requirements Specification

A software requirements specification (SRS document) describes how a software system should be developed. Simply put, an SRS provides everyone involved with a roadmap for that project. It offers high-grade definitions for the functional and non-functional specifications of the software, and can also include use cases that illustrate how a user would interact with the system upon completion. An SRS should have enough information for developers to complete the software described. It not only lays out the description of the software under development but also the purpose it will serve: what the software is supposed to do and how it should perform.

3.1 Introduction

In the digital age, the ease of access to vast repositories of information has given rise to the pervasive issue of plagiarism, a practice that undermines the principles of academic and professional integrity. Plagiarism detection systems play a crucial role in upholding these principles by identifying instances of content misappropriation. Traditional methods, such as rule-based systems, have limitations in coping with the evolving nature of plagiarism. In response, this study introduces an innovative approach to plagiarism detection by harnessing the power of Machine Learning (ML) techniques. The ubiquity of digital content demands automated and adaptive systems capable of discerning intricate patterns and similarities within vast datasets. ML, with its ability to learn from data patterns and make predictions, offers a promising avenue for enhancing the accuracy and efficiency of plagiarism detection. This research explores the application of ML algorithms to the challenging task of distinguishing between original and plagiarized content, aiming to provide a more robust and scalable solution.

3.2 Purpose

In the era of Internet revolution, the abrupt act of plagiarism has been highest than ever. Hence the world is in dire need of up plagiarism detectors. Most of such systems in the market ask for personal details of the user. The aim of our group through this project is to provide the user,

especially teachers and educational institutions with a freely available, easy to use plagiarism detector.

We propose an idea to build a plagiarism detector using built in machine learning libraries. We focus on sci-kit library which contains various useful and efficient machine learning tools. Basic techniques like Vectorization and cosine similarity together could be used in building an efficient plagiarism detection system.

3.3 Overview

The purpose of this document is to identify unambiguously the user requirements and clearly define both functional and non-functional requirements of car rental system. In addition, this document is intended to cover technical goals as well as objectives of the proposed System.

3.4 Functional Requirement:

3.4.1. Scope of the work

This study encompasses the exploration of various ML algorithms, including but not limited to logistic regression, support vector machines, and neural networks, to identify the most effective approach for plagiarism detection. The research also considers different feature extraction methods, such as TF- IDF and word embeddings, to capture the semantic nuances of text.

3.4.2. Functional Requirements

- **User Authentication**
  - Users must be able to sign up and log in using Firebase Authentication.
  - OAuth-based authentication using Google Sign-In.
  - Password reset functionality.
- **Plagiarism Detection**
  - Users can submit **text or document files (TXT, PDF, DOCX)**.
  - The system will preprocess text (tokenization, stopword removal, stemming).

- The **cosine similarity** algorithm will compare the input text with existing datasets.
- Integration with **Google Search API** to compare input text against online sources.

➢ **Report Generation**

- A plagiarism percentage score will be displayed.
- The system will highlight **similar text portions** in the document.
- Users can download a **detailed plagiarism report (PDF format)**.

3.5 Non-Functional Requirements:

➢ **Performance Requirements**

- The system should process documents and generate reports in **under 1 minute**.
- Must support **simultaneous user requests** efficiently.

➢ **Security Requirements**

- Secure authentication via Firebase OAuth.
- **Data encryption** for user submissions.
- API keys and sensitive data should be **stored securely**.

➢ **Usability Requirements**

- Simple **dashboard for text submission and results display**.
- Support for both **desktop and mobile browsers**.

➢ **Scalability**

- The system should be **scalable** to handle a growing number of users.
- Cloud-based storage for text submissions and reports.

➢ **Look and Feel Requirement**

- Guidelines for the visual design of the web-application, focusing on the user experience principles.
- Ensuring a consistent look and feel across different parts of the web application.

Chapter 4

Software Design

4.1 Flowchart of the Major Project

Sci-kit-learn is a built-in library that is used for machine learning tools. It contains tools for machine learning and statistical modeling. This library has been used in the proposed system for feature extraction from the text. The Tf-idf vectorizer is used for word embedding, i.e., conversion of textual data into an array of numbers.

This converted form of textual data into the form of a vector is now utilized to detect the similarity between two text files. Cosine similarity computes the cosine of the angle between the two vector forms of text files. This computation results in a score that ranges from 0-1, hence providing us the information about the extent of similarity between the two input files.

The implementation approach involves four crucial steps that includes,

1) Input File

The file is supposed to be the input for the plagiarism detection system. It should be in text format (.txt extension).

2) Vectorization of text

Sci-kit built-in features make sure that the words obtained from the textual input get converted into a vector format.

3) Compute similarity

The resemblance of two text files is computed using the basic concept of Cosine Similarity. The similarity between two text files depicted in the form of vectors is computed using the dot product of both the vectors, i.e., $\cos\theta$ (θ being the angle between the two vectors).

4) Similarity Score

A similarity score is generated that signifies the amount of similarity detected between the two text files. The score is on a scale of 0-1(positive values of $\cos\theta$ ranges from 0 to 1).

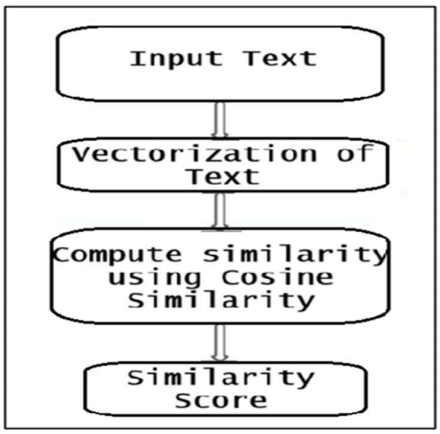

Figure 4.1 Flowchart of the Plagiarism Detection

4.2 Use Case Model/DFDS

4.2.1. Use Case Diagram

A Use Case Diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases.

The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted. Interaction among actors is not shown on the use case diagram. If this interaction is essential to a coherent description of the desired behavior, perhaps the system or use case boundaries should be re-examined. Alternatively, interaction among actors can be part of the assumptions used in the use case.

Use cases: A use case describes a sequence of actions that provide something of measurable value to an actor and is drawn as a horizontal ellipse.

Actors: An Actor models a type of role played by an entity that interacts with the subject (e.g., by exchanging signals and data), but which is external to the subject (i.e., in the sense that an instance of an actor is not a part of the instance of its corresponding subject). Actors may represent roles played by human users, external hardware, or other. subjects. Note that an actor does not necessarily represent a specific physical entity but merely a particular facet (i.e., "role") of some entity that is relevant to the specification of its associated use cases. Thus, a single physical instance may play the role of several different actors and, conversely, a given actor may

be played by multiple different instances. Association: An association specifies a semantic relationship that can occur between typed instances. It has at least two ends represented by properties, each of which is connected to the type of the end. More than one end of the association may have the same type.

System boundary boxes (optional): A rectangle is drawn around the use cases, called the system boundary box, to indicate the scope of system. Anything within the box represents functionality that is in scope and anything outside the box is not. Four relationships among use cases are used often in practice.

a. **Include:** In one form of interaction, a given use case may include another. "Include is a Directed Relationship between two use cases, implying that the behavior of the included use case is inserted into the behavior of the including use case. The first use case often depends on the outcome of the included use case. This is useful for extracting truly common behaviors from multiple use cases into a single description. The notation is a dashed arrow from the including to the included use case, with the label "«include»". There are no parameters or return values. To specify the location in a flow of events in which the base use case includes the behavior of another, you simply write include followed by the name of use case you want to include, as in the following flow for track order.
b. **Extend:** In another form of interaction, a given use case (the extension) may extend another. This relationship indicates that the behavior of the extension use case may be inserted in the extended use case under some conditions. The notation is a dashed arrow from the extension to the extended use case, with the label "«extend»". Modelers use the «extend» relationship to indicate use cases that are "optional" to the base use case.
c. **Generalization:** In the third form of relationship among use cases, a generalization/specialization relationship exists. A given use case may have common behaviors, requirements, constraints, and assumptions with a more general use case. In this case, describe them once, and deal with it in the same way, describing any differences in the specialized cases. The notation is a solid line ending in a hollow triangle drawn from the specialized to the more general use case (following the standard generalization notation.
d. **Associations:** Associations between actors and use cases are indicated in use case diagrams by solid lines. An association exists whenever an actor is involved with an interaction described by a use case. Associations are modeled as lines connecting use

cases and actors to one another, with an optional arrowhead on one end of the line. The arrowhead is often used to indicating the direction of the initial invocation of the relationship or to indicate the primary actor within the use case.

Identified Use Cases: The "user model view" encompasses a problem and solution from the preservative of those individuals whose problem the solution addresses. The view presents the goals and objectives of the problem owners and their requirements of the solution. This view is composed of "use case diagrams". These diagrams describe the functionality provided by a system to external integrators. These diagrams contain actors, use cases, and their relationships.

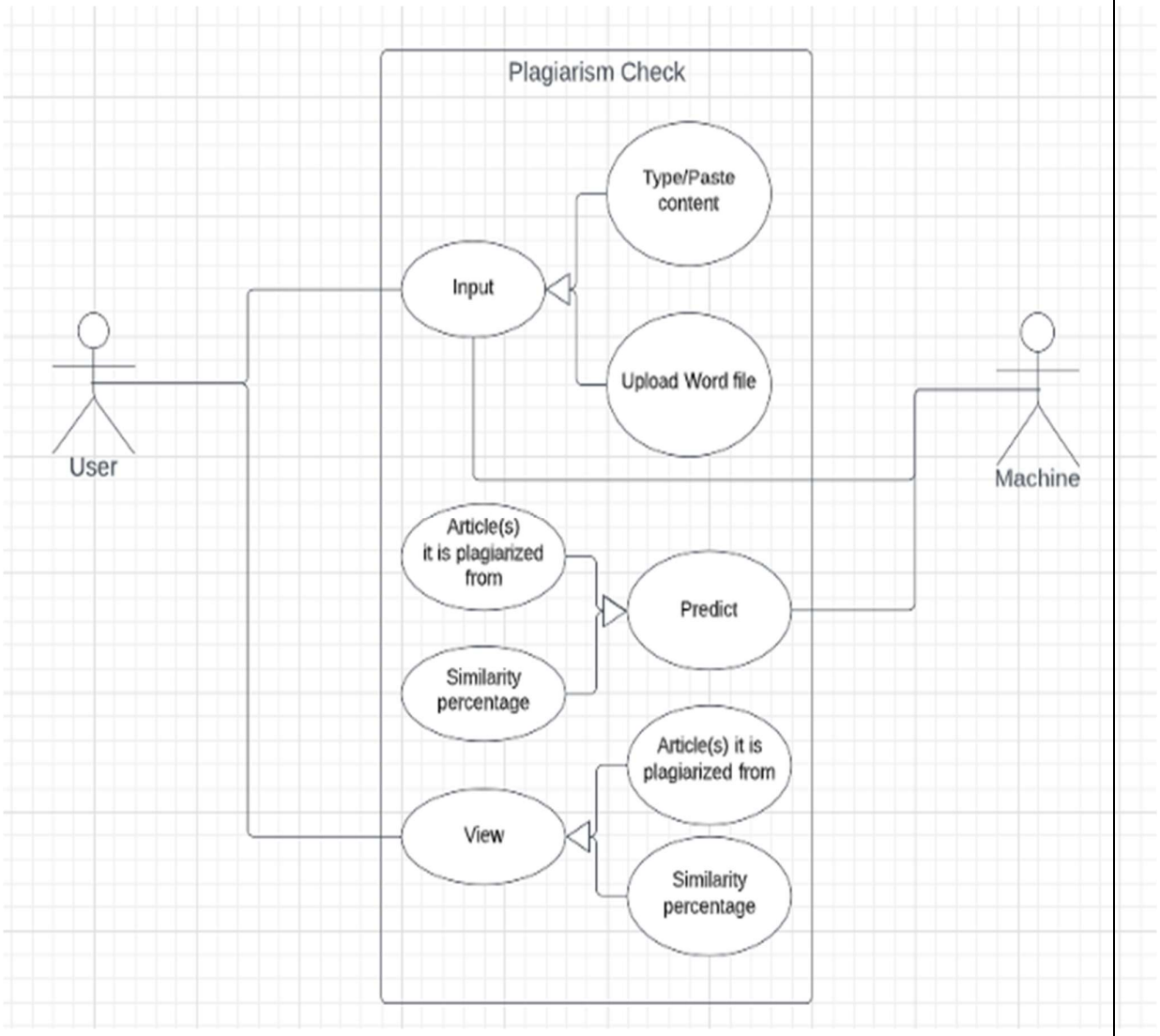

Figure: 4.2 Use case diagram

4.2.2 Class Diagram

A Class Diagram is a Structure Diagram whose primary purpose is to identify the potential classes in a system, their attributes, operations that each class has to perform, and how the classes in the system are connected to one another, i.e., identifying the relation between various classes.

Model, objects are entities that combine state (i.e., data), behavior (i.e., procedures, or methods) and identity (unique existence among all other objects). The structure and behavior of an object are defined by a class, which is a definition, or blueprint, of all objects of a specific type. An object must be explicitly created based on a class and an object thus created is considered to be an instance of that class. An object is similar to a structure, with the addition of method pointers, member access control, and an implicit data member which locates instances of the class (i.e. actual objects of that class) in the class hierarchy (essential for runtime inheritance features).

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, and the relationships between the classes.

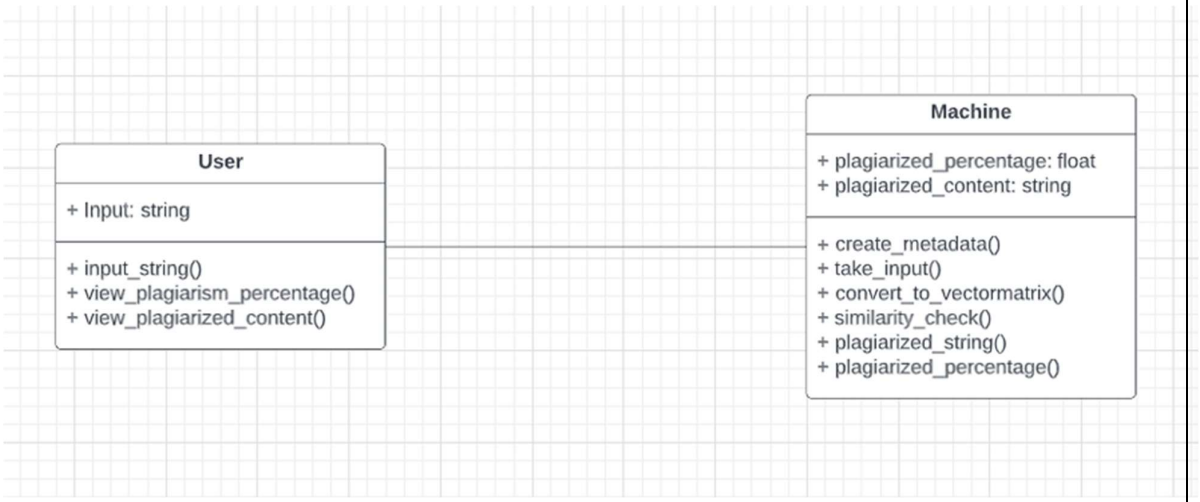

Figure:4.3 Class Diagram

The class diagram is the main building block in object-oriented modeling. It is used both for general conceptual modeling of the semantics of the application, and for detailed modeling translating the models into programming code. The classes in a class diagram represent both the main objects and or interactions in the application and the objects to be programmed. In the class diagram these classes are represented with boxes which contain the two parts:

- The upper part holds the name of the class.

- The middle part contains the attributes of the class.
- The lower part contains the operations of the class.

In our proposed system, we have two classes: User and Machine. The User has only one attribute, i.e., Input, and the operations that they can perform are to give input, view the plagiarized percentage and content after evaluation. Coming to Machine class, its attributes are plagiarized percentage and content. The operations that our machine can perform are creating metadata (Vector Search Index, Aka Vector Database), performing similarity checks, return plagiarized percentage and the plagiarized content.

4.2.3. Data Flow Diagram

It is a way of representing a flow of a data of a process or a system (usually an information system). The DFD also provides information about the outputs and inputs of each entity and the process itself. A data-flow diagram has no control flow, there are no decision rules and no loops. Specific operations based on the data can be represented by a flowchart.

Data flow diagram is the starting point of the design phase that functionally decomposes the requirements specification. A DFD consists of a series of bubbles joined by lines. The bubbles represent data transformation and the lines represent data flows in the system. A DFD describes what data flow rather than how they are processed, so it does not hardware, software and data structure.

There are several notations for displaying data-flow diagrams. The notation presented above was described in 1979 by Tom DeMarco as part of Structured Analysis.

For each data flow, at least one of the endpoints (source and/or destination) must exist in a process. The refined representation of a process can be done in another data-flow diagram, which subdivides this process into sub-processes.

A data-flow diagram (DFD) is a graphical representation of the "flow" of data through an information system. DFDs can also be used for the visualization of data processing (structured design). A data flow diagram (DFD) is a significant modeling technique for analyzing and constructing information processes. DFD literally means an illustration that explains the course or movement of information in a process. DFD illustrates this flow of information in a process based on the inputs and outputs. A DFD can be referred to as a Process Model.

The data-flow diagram is part of the structured-analysis modelling tools. When using UML, the activity diagram typically takes over the role of the data-flow diagram. A special form of data-flow plan is a site-oriented data-flow plan.

There are seven rules for construct a data flow diagram:

i. Arrows should not cross each other.
ii. Squares, circles and files must wear names.
iii. Decomposed data flows must be balanced.
iv. No two data flows, squares or circles can be the same names.
v. Draw all data flows around the outside of the diagram.
vi. Choose meaningful names for data flows, processes & data stores.
vii. Control information such as record units, password and validation requirements are not penitent to a data flow diagram.

Additionally, a DFD can be utilized to visualize data processing or a structured design. This basic DFD can be then disintegrated to a lower-level diagram demonstrating smaller steps exhibiting details of the system that is being modeled.

On a DFD, data items flow from an external data source or an internal data store to an internal data store or an external data sink, via an internal process. It is common practice to draw a context-level data flow diagram first, which shows the interaction between the system and external agents, which act as data sources and data sinks.

The DFD enables the software engineer to develop models of the information domain & functional domain at the same time. As the DFD is refined into greater levels of details, the analyst performs an implicit functional decomposition of the system.

The Data Flow Diagram has 4 components:

- **Process:** Input to output transformation in a system takes place because of process function. The symbols of a process are rectangular with rounded corners, oval, rectangle or a circle. The process is named a short sentence, in one word or a phrase to express its essence
- **Data flow:** Data flow describes the information transferring between different parts of the systems. The arrow symbol is the symbol of the data flow. A relatable name should be given to the flow to determine the information which is being moved. Data flow also represents Material along with information that is being moved.

- **Warehouse:** The data is stored in the warehouse for later use. Two horizontal lines represent the symbol of the store. The warehouse is simply not restricted to being a data file rather it can be anything like a folder with documents, an optical disc, a filing cabinet.
- **Terminator:** The Terminator is an external entity that stands outside of the system and communicates with the system. It can be, for example, organizations like banks, groups of people like customers or different departments of the same organization, which is not a part of the model system and is an external entity.
- **Context Diagram:** On the context diagram (also known as the Level 0 DFD'), the system's interactions with the outside world are modeled purely in terms of data flows across the system boundary. The context diagram shows the entire system as a single process, and gives no clues as to its internal organization.

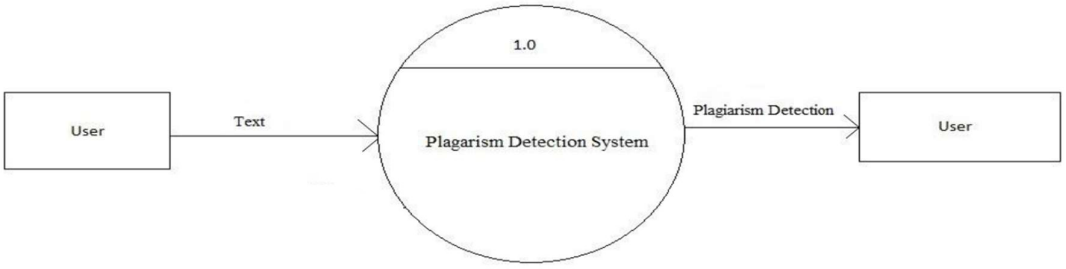

Figure 4.4: Context diagram of Plagiarism Detection System

4.2.4 Activity Diagram

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control. Activity diagrams are constructed from a limited repertoire of shapes, connected with arrows. The most important shape types:

- rounded rectangles represent activities;
- diamonds represent decisions;
- bars represent the start (split) or end (join) of concurrent activities;
- a black circle represents the start (initial state) of the workflow;

- an encircled black circle represents the end (final state).

Arrows run from the start towards the end and represent the order in which activities happen. However, the join and split symbols in activity diagrams only resolve this for simple cases; the meaning of the model is not clear when they are arbitrarily combined with the decisions or loops.

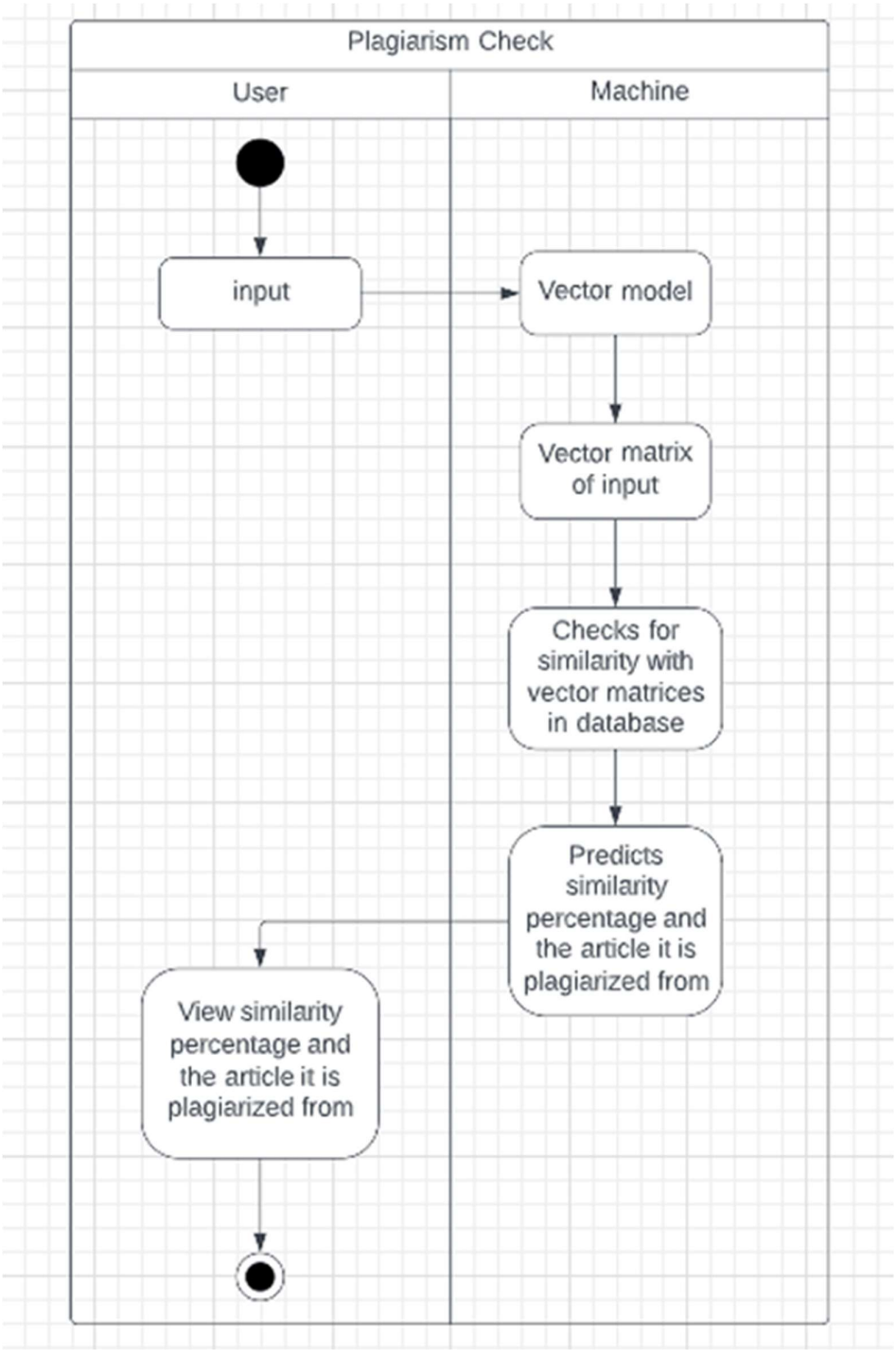

Figure 4.5: Activity Diagram

The website begins with text input from the user, which goes into the model which converts the input into a vectors. Based on this vector matrix, the model will query the database for similarity. Once the querying is done, the result is returned to the user for view.

4.2.5 Sequence Diagram

A Sequence Diagram is another Behavioral Diagram whose primary purpose is to describe how the different objects in the system interact. In most cases, a sequence diagram and an activity diagram are very similar to other.

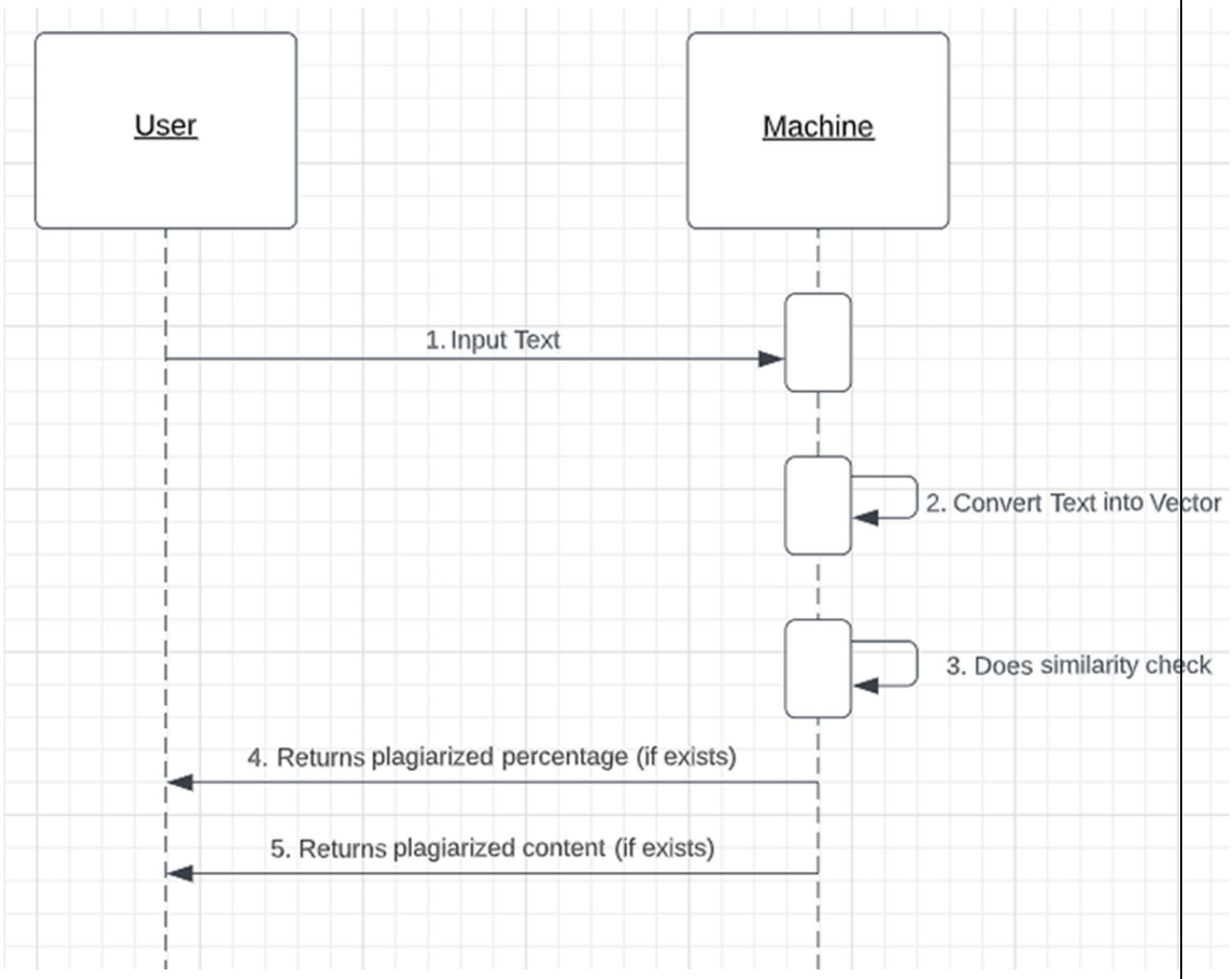

Figure 4.6: Sequence Diagram

Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams. A sequence diagram shows, as parallel vertical lines (lifelines), different processes or objects that live simultaneously, and, as horizontal arrows, the messages exchanged between them, in the order in which they occur. This allows the specification of simple runtime scenarios in a graphical

manner. If the lifeline is that of an object, it demonstrates a role. Note that leaving the instance name blank can represent anonymous and unnamed instances. In order to display interaction, messages are used. These are horizontal arrows with the message name written above them. Solid arrows with full heads are synchronous calls, solid arrows with stick heads are asynchronous calls and dashed arrows with stick heads are return messages.

Activation boxes, or method-call boxes, are opaque rectangles drawn on top of lifelines to represent that processes are being performed in response to the message (Execution Specifications in UML).

Objects calling methods on themselves use messages and add new activation boxes on top of any others to indicate a further level of processing. When an object is destroyed (removed from memory), an X is drawn on top of the lifeline, and the dashed line ceases to be drawn below it (this is not the case in the first example though). It should be the result of a message, either from the object itself, or another.

A message sent from outside the diagram can be represented by a message originating from a filled-in circle (found message in UML) or from a border of sequence diagram (gate in UML)

There are two objects in our proposed model: User and Machine. The website begins with text input from the user, which goes into the model which converts the input into a vector matrix. Based on this vector matrix, the model will query the database for similarity. Once the querying is done, the result is returned to the user for view.

4.2.6 Deployment Diagram:

Deployment diagrams are used to visualize the topology of the physical components of a system where the software components are deployed.

So deployment diagrams are used to describe the static deployment view of a system. Deployment diagrams consist of nodes and their relationships.

Purpose:

The name Deployment itself describes the purpose of the diagram. Deployment diagrams are used for describing the hardware components where software components are deployed. Component diagrams and deployment diagrams are closely related.

Component diagrams are used to describe the components and deployment diagrams shows how they are deployed in hardware.

UML is mainly designed to focus on software artifacts of a system. But these two diagrams are special diagrams used to focus on software components and hardware components.

So most of the UML diagrams are used to handle logical components but deployment diagrams are made to focus on hardware topology of a system. Deployment diagrams are used by the system engineers.

The purpose of deployment diagrams can be described as:

- Visualize hardware topology of a system.
- Describe the hardware components used to deploy software components.
- Describe runtime processing nodes.

How to draw Deployment Diagram?

Deployment diagram represents the deployment view of a system. It is related to the component diagram. Because the components are deployed using the deployment diagrams. A deployment diagram consists of nodes. Nodes are nothing but physical hardware's used to deploy the application. Deployment diagrams are useful for system engineers. An efficient deployment diagram is very important because it controls the following parameters:

- Performance
- Scalability
- Maintainability
- Portability

So before drawing a deployment diagram the following artifacts should be identified:

- Nodes
- Relationships among nodes

1. **Components**:
    a. **Web Server**: Represents the server hosting the web application.
    b. **Database Server**: Used to retrieve data using the google search api.
    c. **Machine Learning Algorithm**: The plagiarism detection system (e.g. consine similarity) deployed on the server.

d. **User's Browser**: Represents the client-side interface where users interact with the system.

2. **Connections**:
    a. The **Web Server** communicates with the **Database Server** to retrieve data.
    b. The **Web Server** interacts with the **Machine Learning Model** to make plagiarism detection.
    c. Users access the system through their **Browsers**.
3. **Deployment Nodes**:
    a. **Web Server Node**: Represents the physical or virtual server hosting the web application.
    b. **Database Server Node**: Represents the server where the database is hosted.
4. **Explanation**:
    a. The **Web Application** interacts with the **Machine learning algorithm** stored in the **Database** to retrieve information.
    b. The **Machine Learning Algorithm** resides within the **Web Application** and makes plagiarism detections based on input data.
    c. Users (represented by the **User** actor) access the system via their browsers.

So the following deployment diagram has been drawn considering all the points mentioned above:

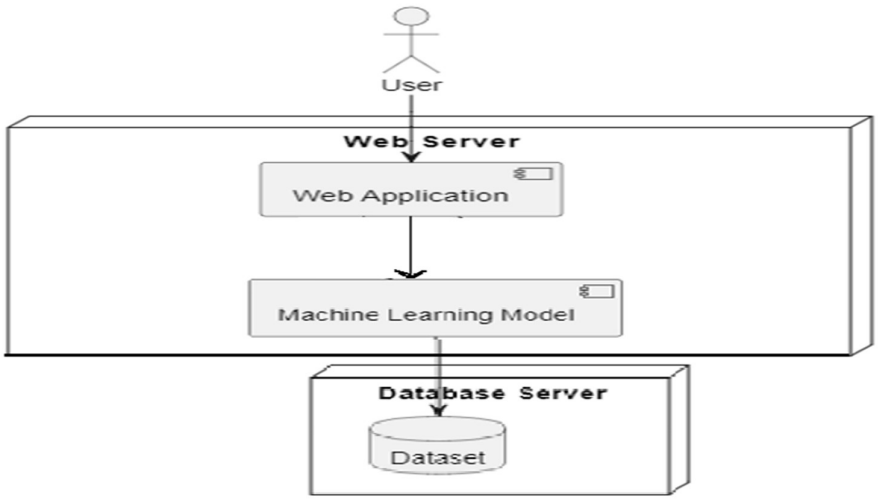

Figure:4.7 Deployment diagram

Chapter 5

Testing Module

5.1 Testing Modules

1. React.js Frontend Testing

- **Unit Testing:** React Testing Library to test individual components.
- **Integration Testing:** Test interactions between components and APIs (like Firebase OAuth).

2. Firebase OAuth Testing

- **Mock Authentication:** Use Firebase emulators or mock services to simulate login/logout scenarios.
- **Security Testing:** Validate token handling, session expiration, and access controls.

3. Python Backend Testing (app.py)

- **Unit Testing:** Use pytest for individual functions (e.g., vectorization, cosine similarity).
- **Integration Testing:** Test API endpoints using requests or Flask-Testing.

4. Cosine Similarity Algorithm Testing

- **Correctness:** Validate with known datasets to check if the similarity scores are accurate.
- **Performance:** Test with large text samples to measure response time and resource usage.

5. Google Search API Testing

- **Mock Responses:** Simulate API responses to test how your system handles different scenarios (e.g., no results, API errors).
- **Rate Limiting:** Check how your application handles API rate limits.

6. Plagiarism Report Generation

- **Data Accuracy:** Verify that the reports correctly reflect similarity scores and source matches.

- **Format Validation:** Ensure the report format is consistent and user-friendly.

7. Continuous Integration (CI) Pipeline

- Set up CI/CD pipelines (using GitHub Actions, GitLab CI, etc.) to automate tests on every code push.

5.2 Test Cases

5.2.1 Test Scenario:

Authenticate Admin by verifying login from the database.

Test Pre-Conditions:

1) Enter valid Account credentials to login

Test Steps:

1) Enter Username
2) Enter Password
3) Click Login button

Test Case	Test Data	Expected Results	Post Condition
Enter valid User Name and valid Password	<Valid User Name> <Valid Password>	Successful login	Home Page
Enter valid User Name and invalid Password	<Valid User Name> <Invalid Password>	A message "The email and password you entered don't match"	Stays on login page
Enter invalid User Name and invalid Password	<Invalid User Name> <valid password>	A message "The email and password you entered don't match"	Stays on login page
Enter invalid User Name and invalid Password	<Invalid User Name> <Invalid User Name>	A message "The email and password you entered don't match" is shown	Stays on login page

Table 3.1: Test Case for Login

5.4.2. Test Scenario:

Creating Account and authenticate User by verifying login from the database.

Test Pre-Conditions:

1) Enter valid details for creating account.

2) Enter valid Active Account credentials to login.

Test Steps:

1) Enter User Details

2) Enter Username and Password

3) Click Register/ Login button

Sr.no.	Data Input	Excepted Output	Actual Output	Pass/Fail
1.	All fields are empty	Error message: *indicate compulsory fields	Error message: *indicate compulsory fields	Pass
2.	Email	Error message: Invalid Email-address	Error message: Invalid Email-address	Pass
3.	Password and Confirm Password	Error message: Both password does not match	Error message: Both password does not match	Pass
4.	Login	The System give an error and denied from the login	Login should fail with an error 'Invalid Login'	Pass
5.	User	Login should be allowed and User Visitor side	Login successfully and user its User Page	Pass
6.	Validation Test Case	Pre-define format must be required in control	System give error to enter valid input	Pass
		Enter data in a compulsory field with required field validations.	Data must be field in a compulsory field otherwise error message is displayed	Pass

Table 3.2: Test case for Creating Account and Authenticate User

By executing these testing techniques and relevant test cases, the Plagiarism Detection System can be thoroughly evaluated for the accuracy, reliability, and user satisfaction and by conducting thorough testing using these techniques and test cases, we aim to develop a robust and user-friendly plagiarism detection system that includes reliable algorithm selection.

Chapter 6

Performance of the project Developed (so far)

1. Project Progress:

- The development of the Plagiarism detection system has progressed smoothly, with approximately half of the project completed.
- Core functionalities such as the user interface, input forms, report generation integration are implemented.

2. User Interface (UI) and User Experience (UX):

- The UI design has been developed with attention to usability and accessibility.
- Input forms are user-friendly, making it easy for users to enter texts for plagiarism detection.
- The UI layout is intuitive, guiding users through the detection process effectively.

3. Plagiarism detection Algorithm Integration:

- Initial integration of plagiarism detection algorithm, including cosine similarity and possibly other algorithms, is complete.
- The system is capable of generating reports on plagiarism detection based on user input.

4. Performance and Speed:

- Initial performance testing indicates satisfactory response times for generating reports.
- Further optimization may be required to ensure consistent performance as the user base grows.

5. Remaining Work and Challenges:

- Despite significant progress, there are still challenges and remaining tasks to be addressed in the second half of the project.
- Further refinement of plagiarism detection algorithm evaluation is needed to enhance accuracy.
- Comparison of system with some other open-source plagiarism detection tools and additional user customization options remains to be completed.

- Thorough testing, debugging, and user feedback integration are essential for ensuring a robust and reliable system.

Overall, the project has made substantial progress and is on track to deliver a functional and user-friendly Plagiarism Detection System. However, continued efforts in development, testing, and refinement are crucial to achieving the project's objectives effectively.

Chapter 7

Output Screens

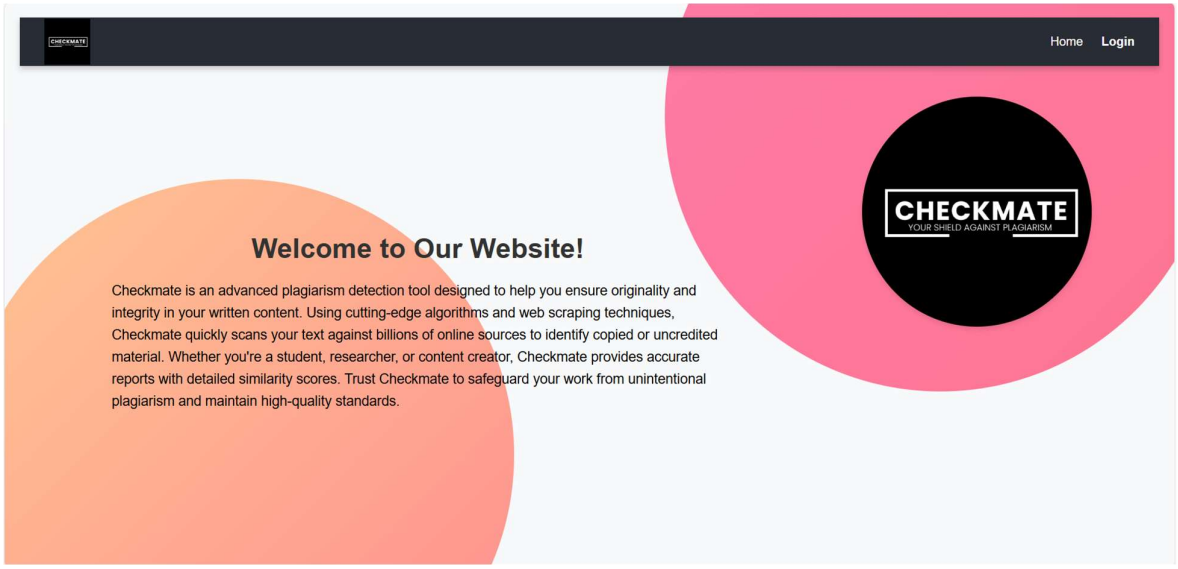

Figure 7.1: Home Page for Plagiarism Detection System

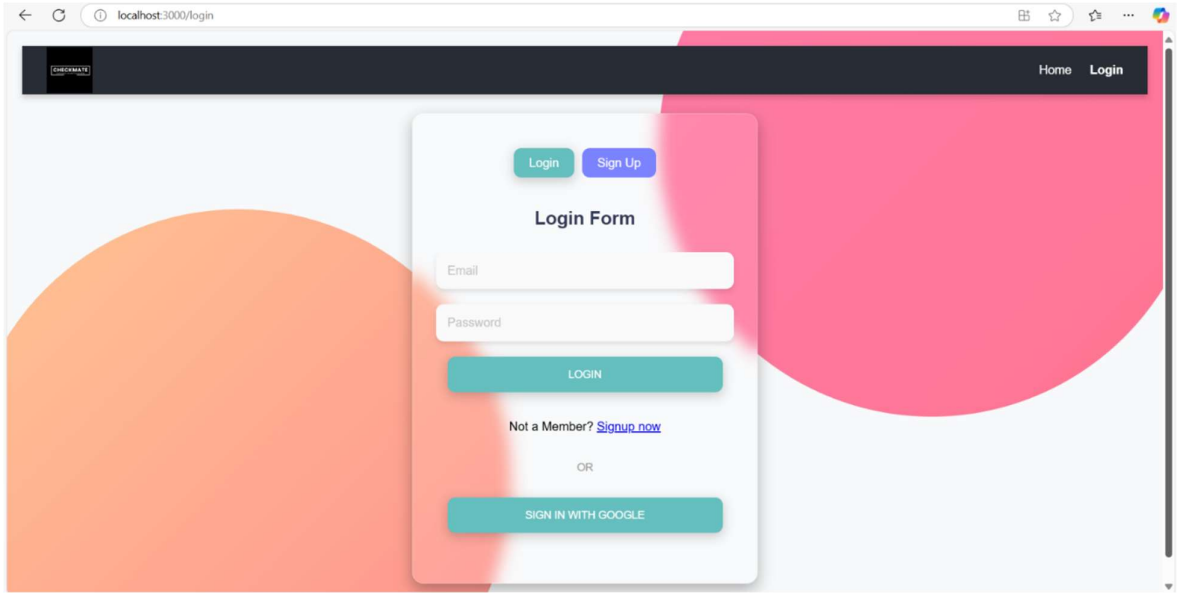

Figure 7.2: Login Page

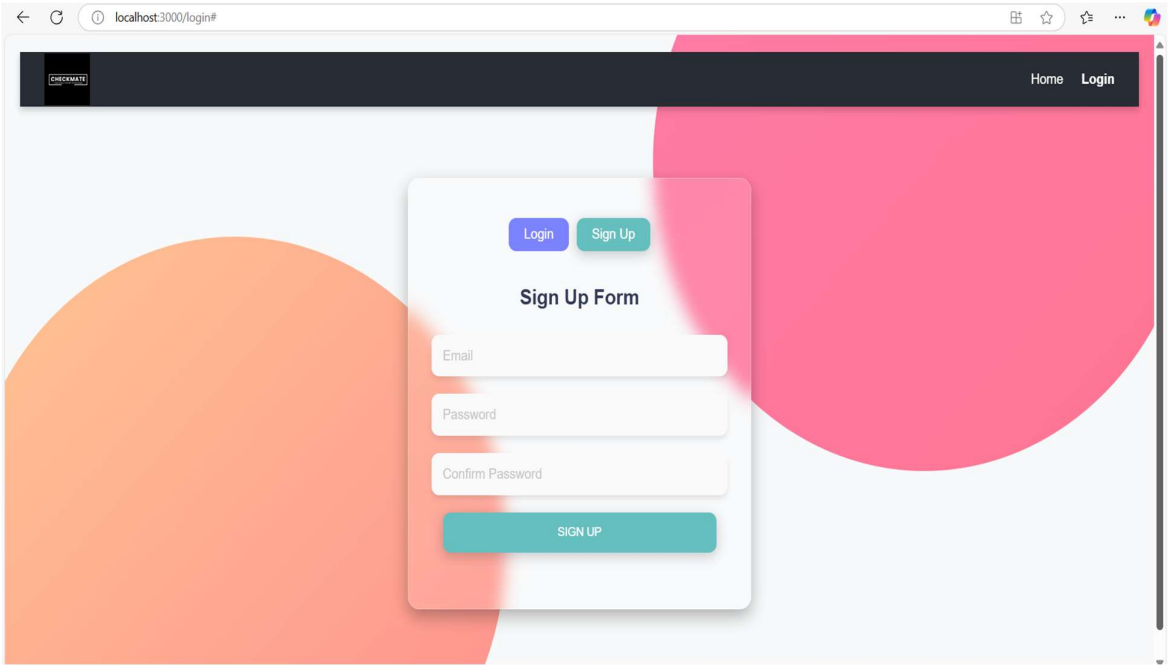

Figure 7.3: Sign Up Page

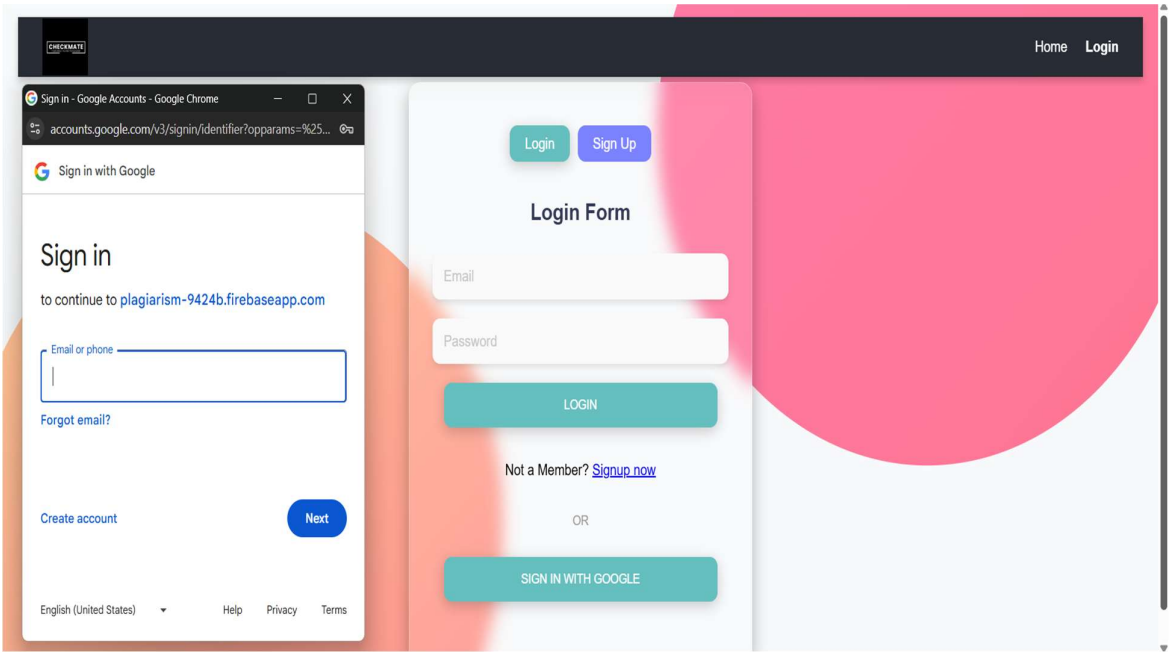

Figure 7.4: Login with Firebase Authentication

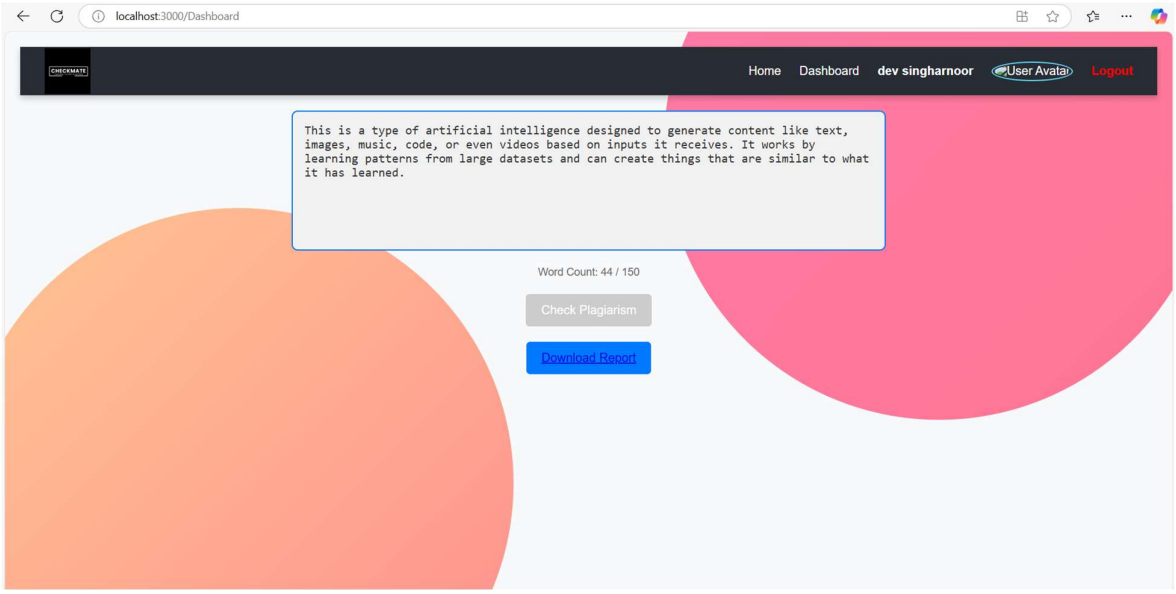

Figure 7.5: Dashboard Page for Plagiarism Detection System

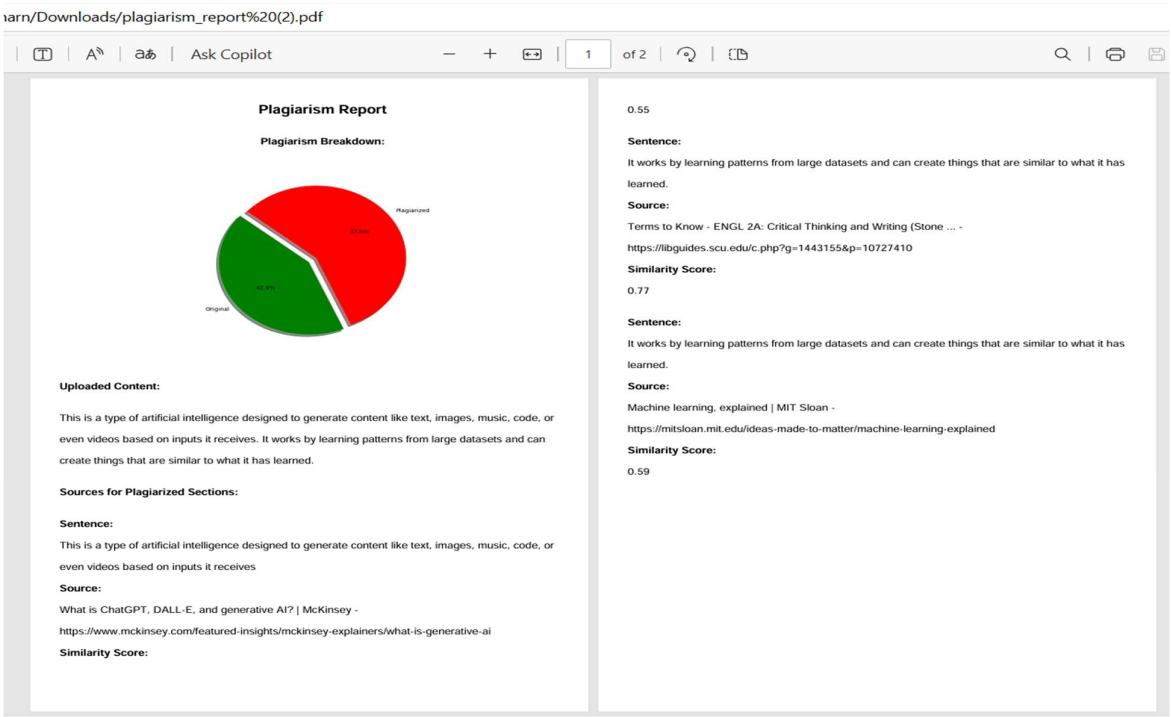

Figure 7.6: Generated Report in PDF form

References

[1] Y. P. Ma et al., "Plagiarism Detection Based on Lexical and Syntactic Features Using Convolutional Neural Network," in Proceedings of the IEEE International Conference on Computational Science and Computational Intelligence, 2019, pp. 698-703.

[2] S. V. Moravvej, S. J. Mousavirad, D. Oliva, G. Schaefer and Z. Sobhaninia, "An Improved DE Algorithm to Optimise the Learning Process of a BERT-based Plagiarism Detection Model," 2022 IEEE Congress on Evolutionary Computation (CEC), Padua, Italy, 2022, pp. 1-7

[3] El-Rashidy, Mohamed A., et al. "Reliable plagiarism detection system based on deep learning approaches." Neural Computing and Applications 34.21 (2022): 18837-18858.

[4]. Imam Much Ibnu Subroto and Ali Selamat, "Plagiarism Detection through Internet using Hybrid Artificial Neural Network and Support Vectors Machine," TELKOMNIKA, Vol.12, No.1, March 2014, pp. 209-218.

[5]. Upul Bandara and Gamini Wijayrathna ,"Detection of Source Code Plagiarism Using Machine Learning Approach," International Journal of Computer Theory and Engineering, Vol. 4, No. 5, October 2012, pp.674- 678

[6]. Chavan, Hiten, et al. "Plagiarism detector using machine learning." International Journal of Research in Engineering, Science and Management 4.4 (2021): 152-154.

[7] Siddharth Tata, Suguri Charan Kumar and Varampati Reddy Kumar, “Extrinsic Plagiarism Detection Using Fingerprinting”, International Journal of Computer Science and Technology (IJCST) Vol. 10, Issue 4, Oct - Dec 2019.