

**PLAGIARISM DETECTION SYSTEM FOR URDU LANGUAGE**

FINAL YEAR PROJECT REPORT

**Submitted by**

SYED AHMED ALI (2579-2021)

SYED SHAHEER Ul HAQUE (2197-2021)

**Supervisor**

Mr. AAMIR HUSSAIN

In partial fulfilment of the requirements for the degree of

**Bachelor of Science in Computer Science**

**2025**

**Faculty of Engineering Sciences and Technology**

*Hamdard Institute of Engineering and Technology*

*Hamdard University, Main Campus, Karachi, Pakistan*

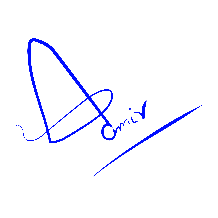
### CERTIFICATE OF APPROVAL



**Faculty of Engineering Sciences and Technology**

*Hamdard Institute of Engineering and Technology  
Hamdard University, Karachi, Pakistan*

This project “**Plagiarism Detection System for Urdu Language**” is presented by **Syed Ahmed Ali and Syed Shaheer-ul-Haque** under the supervision of their project advisor and approved by the project examination committee, and acknowledged by the Hamdard Institute of Engineering and Technology, in the fulfillment of the requirements for the Bachelor degree of Science in Computer Science.

****

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Mr. Aamir Hussain In-charge FYP-Committee

(Project Supervisor)

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Chairman

(Project Co-Supervisor) (Department of Computing)

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

(Dean, FEST)

### AUTHORS’ DECLARATION

We declare that this project report was carried out in accordance with the rules and regulations of Hamdard University. The work is original except were indicated by special references in the text and no part of the report has been submitted for any other degree. The report has not been presented to any other University for examination.

Dated: 6/7/25

Authors Signatures:



\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Syed Ahmed Ali



\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Syed Shaheer ul Haque

**PLAGIARISM UNDERTAKING**

We, **SYED AHMED ALI** and **SYED SHAHEER UL HAQUE**, solemnly declare that the work presented in the Final Year Project Report titled **PLAGIARISM DETECTION SYSTEM FOR URDU LANGUAGE** has been carried out solely by ourselves with no significant help from any other person except few of those which are duly acknowledged. We confirm that no portion of our report has been plagiarized and any material used in the report from other sources is properly referenced.

Dated:

Authors Signatures:



\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Syed Ahmed Ali



\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Syed Shaheer ul Haque

### ACKNOWLEDGMENTS

In the name of Allah, Most Merciful, Most Generous. We had the courage to study and carry out this proposal within the allotted time thanks to all of His favors. In order to complete this job, we would like to appreciate and extend our sincere gratitude to our supervisors, Mr. Aamir Hussain. We’re grateful for your wise remarks and recommendations, which helped us through every stage of working on the project. We also want to thank our entire family for their continuous acceptance and support while we worked on your project and completed it. We’ve gotten this far thanks to their prayers for us.

### DOCUMENT INFORMATION

Table 1: Document Information

|  |  |
| --- | --- |
| Project Title | Plagiarism Detection System for Urdu Language |
| Document | Final Year Project Report |
| Document Version | 2.0 |
| Identifier | (FYP-027/FL24) |
| Status | Completed |
| Author(s) | Syed Ahmed Ali, Syed Shaheer ul Haque |
| Approver(s) | Mr. Aamir Hussain |
| Issue Date | 12/Jan/25 |

**Definition Of Terms, Acronyms, And Abbreviations***.*

Table 2: Definition of Terms, Acronyms, and Abbreviations

|  |  |
| --- | --- |
| **Term** | **Definition** |
| **Plagiarism** | Plagiarism is the act of using someone else's work, ideas, or written content without giving proper credit or permission. It is considered unethical in academic and professional settings and can lead to penalties, rejection of work, or legal consequences. |
| **Urdu Natural Language Processing** | Urdu Natural Language Processing involves applying computer-based techniques to understand and work with text written in the Urdu language. It includes tasks such as tokenization, sentiment analysis, machine translation, and text classification tailored to Urdu grammar, script, and linguistic patterns. |
| **Tokenization** | Tokenization is the process of breaking a long string of text into smaller units, known as tokens. These tokens are typically words or phrases that make it easier for a computer to analyze the structure and meaning of a sentence. This is one of the first steps in processing natural language. |
| **Stemming** | Stemming is a technique used to reduce words to their root or base form by removing suffixes and prefixes. For example, words like “playing”, “played”, and “plays” are all reduced to “play”. It helps in matching similar words in search and comparison tasks. |
| **Lemmatization** | Lemmatization also reduces words to their base form but does so using vocabulary and grammar rules to return meaningful base forms (lemmas). Unlike stemming, it ensures the base form exists in the dictionary. For instance, “better” becomes “good” when lemmatized. |
| **Cosine Similarity** | Cosine Similarity is a method used to measure how similar two pieces of text are by calculating the cosine of the angle between their vector representations. A smaller angle (closer to 1) indicates higher similarity. It is widely used in document comparison and plagiarism detection. |
| **Bidirectional Encoder Representations from Transformers (BERT)** | Bidirectional Encoder Representations from Transformers is a deep learning model developed by Google. It reads text from both left and right contexts at the same time, allowing it to understand the full meaning of a word based on its surrounding words. It is highly effective in many language understanding tasks. |
| **Database** | A database is a structured collection of data stored electronically for efficient access, updating, and management. Databases can store anything from user profiles to documents and are essential for backend development, data analytics, and application infrastructure. |
| **UrduHack** | UrduHack is a Python-based open-source library designed specifically for processing Urdu language text. It offers tools for tasks such as tokenization, normalization, stemming, and sentence segmentation, enabling more effective NLP for Urdu. |
| **TensorFlow** | TensorFlow is an open-source software library created by Google for machine learning and deep learning tasks. It allows developers to build and train neural networks and is widely used in research, academic, and production-level AI applications. |
| **Plagiarism Report** | A plagiarism report is a document generated by detection software that identifies parts of a text that match with other existing sources. It shows the percentage of similarity and highlights the matched text, helping authors and institutions maintain content originality. |
| **Machine Learning Model** | A machine learning model is an algorithm that has been trained on data to recognize patterns and make predictions or decisions without being explicitly programmed. It is used in applications such as spam detection, speech recognition, and recommendation systems. |
| **Preprocessing** | Preprocessing is the step of preparing raw text or data for further analysis or machine learning. It includes removing punctuation, converting text to lowercase, tokenizing, and eliminating stop words, ensuring the data is clean and structured. |
| **Similarity Score** | A similarity score is a numeric value that represents how alike two texts or documents are. The score usually ranges from 0 to 1, where 1 means the texts are identical. It is commonly used in plagiarism detection and search engines. |
| **Error Message** | An error message is a system notification that appears when a program encounters an issue. It informs the user about the nature of the problem, such as invalid input or a missing file, and is crucial for debugging and resolving software issues. |
| **Urdu Text** | Urdu text refers to any written content composed in the Urdu language, using its distinctive script and structure. It may include poetry, articles, messages, or data and requires special processing in NLP due to its right-to-left orientation and complex morphology. |

### Abstract

Plagiarism in academic writing has always been a critical concern, and with increasing internet access, the act of copying content without credit has become more prevalent. While numerous tools exist for the English language, Urdu—a language spoken by millions—has been left behind in this domain. This project introduces a "Plagiarism Detection System for Urdu Language," designed to process, analyze, and detect plagiarized content in Urdu documents. The system accepts PDF, DOCX, and TXT files, extracts the Urdu text, and performs preprocessing using NLP techniques such as tokenization, normalization, and stop word removal. It then uses a hybrid model that combines cosine similarity and a custom-trained BERT model to detect semantic and syntactic similarities. The results are presented with word-level highlights and a complete PDF report is generated for the user. Built with a FastAPI backend and a Flutter-based mobile frontend, the system ensures ease of use, security, and efficiency. This solution is the first of its kind to provide intelligent plagiarism detection for the Urdu language in a mobile-accessible format.

**Keywords:**

**Plagiarism Detection, Urdu Language, BERT Model, NLP, Cosine Similarity, Tokenization, Flutter App, FastAPI, Machine Learning, Semantic Similarity**

**Table of Contents**

[CERTIFICATE OF APPROVAL 2](#_Toc202699406)

[AUTHORS’ DECLARATION 1](#_Toc202699407)

[ACKNOWLEDGMENTS 3](#_Toc202699408)

[DOCUMENT INFORMATION 4](#_Toc202699409)

[Abstract 6](#_Toc202699410)

[List of Figures 10](#_Toc202699411)

[List of Tables 12](#_Toc202699412)

[CHAPTER 1 13](#_Toc202699413)

[INTRODUCTION 13](#_Toc202699414)

[1.1 MOTIVATION 13](#_Toc202699415)

[1.2 PROBLEM STATEMENT 13](#_Toc202699416)

[1.3 GOALS AND OBJECTIVES 14](#_Toc202699417)

[1.4 PROJECT SCOPE 14](#_Toc202699418)

[CHAPTER 2 16](#_Toc202699419)

[RELEVANT BACKGROUND & DEFINITIONS 16](#_Toc202699420)

[CHAPTER 3 17](#_Toc202699421)

[LITERATURE REVIEW & RELATED WORK 17](#_Toc202699422)

[LITERATURE REVIEW 17](#_Toc202699423)

[RELATED WORK 18](#_Toc202699424)

[GAP ANALYSIS 19](#_Toc202699425)

[CHAPTER 4 22](#_Toc202699426)

[1. SOFTWARE ENGINEERING METHODOLOGY 22](#_Toc202699427)

[2. PROJECT METHODOLOGY 22](#_Toc202699428)

[3. PHASES OF PROJECT 23](#_Toc202699429)

[4. SOFTWARE/TOOLS THAT USED IN PROJECT 24](#_Toc202699430)

[5. HARDWARE THAT USED IN PROJECT 24](#_Toc202699431)

[Chapter 5 25](#_Toc202699432)

[5.1 PROPOSED SYSTEM ARCHITECTURE/DESIGN 25](#_Toc202699433)

[5.1.1 USER INTERFACE DIAGRAMS 26](#_Toc202699434)

[5.2 FUNCTIONAL SPECIFICATIONS 32](#_Toc202699435)

[5.3 NON-FUNCTIONAL SPECIFICATIONS 33](#_Toc202699436)

[5.4 TESTING 34](#_Toc202699437)

[5.5 PURPOSE OF TESTING 34](#_Toc202699438)

[5.6 TEST CASES 35](#_Toc202699439)

[Chapter 6 38](#_Toc202699440)

[EXPERIMENTAL EVALUATIONS & RESULTS 38](#_Toc202699441)

[EVALUATION TESTBED 38](#_Toc202699442)

[RESULTS AND DISCUSSION 39](#_Toc202699443)

[CHAPTER 7 48](#_Toc202699444)

[CONCLUSION AND DISCUSSION 48](#_Toc202699445)

[7.1 Strength of this Project 48](#_Toc202699446)

[7.2 Limitations and Future Work 48](#_Toc202699447)

[7.3 Reasons for Failure – If Any 49](#_Toc202699448)

[REFERENCES 50](#_Toc202699449)

[APPENDICES 52](#_Toc202699450)

[A0. COPY OF PROJECT REGISTRATION FORM 53](#_Toc202699451)

[A1A. PROJECT PROPOSAL AND VISION DOCUMENT 54](#_Toc202699452)

[1. Introduction 54](#_Toc202699453)

[2. Objective 54](#_Toc202699454)

[3. Problem Description 54](#_Toc202699455)

[4. Methodology 54](#_Toc202699456)

[5. Project Scope 54](#_Toc202699457)

[6. Feasibility Study 55](#_Toc202699458)

[7. Solution Application 55](#_Toc202699459)

[8. Tools/Technology 55](#_Toc202699460)

[9. Responsibilities of the Team Members 55](#_Toc202699461)

[10. Planning 56](#_Toc202699462)

[11. References 56](#_Toc202699463)

[A1B. COPY OF PROPOSAL EVALUATION COMMENTS BY JURY 57](#_Toc202699464)

[A2. REQUIREMENT SPECIFICATIONS 58](#_Toc202699465)

[1. INTRODUCTION 58](#_Toc202699466)

[1.1. Purpose of Documents 58](#_Toc202699467)

[1.2. Intended Audience 58](#_Toc202699468)

[1.3. Abbreviations 58](#_Toc202699469)

[2. OVERALL SYSTEM DESCRIPTION 59](#_Toc202699470)

[2.1. Project Background 59](#_Toc202699471)

[2.2. Problem Statement 59](#_Toc202699472)

[2.3. Project Scope 59](#_Toc202699473)

[2.4. Project Objectives 59](#_Toc202699474)

[2.5. Stakeholders & Affected Groups 60](#_Toc202699475)

[2.6. Operating Environment 60](#_Toc202699476)

[2.7. System Constraints 60](#_Toc202699477)

[2.8. Assumptions & Dependencies 60](#_Toc202699478)

[3. EXTERNAL INTERFACE REQUIREMENTS 60](#_Toc202699479)

[3.1. Hardware Interfaces 60](#_Toc202699480)

[3.2. Software Interfaces 60](#_Toc202699481)

[3.3. Communications Interfaces 60](#_Toc202699482)

[4. SYSTEM FUNCTIONS / FUNCTIONAL REQUIREMENTS 61](#_Toc202699483)

[4.1. System Functions 61](#_Toc202699484)

[4.2. Use Cases 61](#_Toc202699485)

[5. NON - FUNCTIONAL REQUIREMENTS 63](#_Toc202699486)

[6. REFERENCES 63](#_Toc202699487)

[A3. DESIGN SPECIFICATIONS 64](#_Toc202699488)

[1. INTRODUCTION 64](#_Toc202699489)

[2. DESIGN CONSIDERATIONS 65](#_Toc202699490)

[3. SYSTEM ARCHITECTURE 65](#_Toc202699491)

[4. DESIGN STRATEGY 66](#_Toc202699492)

[5. DETAILED SYSTEM DESIGN 67](#_Toc202699493)

[6. REFERENCES 80](#_Toc202699494)

[A4. OTHER TECHNICAL DETAIL DOCUMENTS 81](#_Toc202699495)

[Test Cases Document 81](#_Toc202699496)

[UI/UX Detail Document 84](#_Toc202699497)

[Coding Standards Document 86](#_Toc202699498)

[1. Introduction 86](#_Toc202699499)

[2. Code Formatting 86](#_Toc202699500)

[3. Naming Conventions 86](#_Toc202699501)

[4. Comments 86](#_Toc202699502)

[5. Error Handling 86](#_Toc202699503)

[6. File Organization 87](#_Toc202699504)

[7. Version Control 87](#_Toc202699505)

[8. Performance 87](#_Toc202699506)

[9. Security 87](#_Toc202699507)

[10. Review and Testing 87](#_Toc202699508)

[Conclusion 87](#_Toc202699509)

[User Manual 88](#_Toc202699510)

[1. Introduction 88](#_Toc202699511)

[2. System Requirements 88](#_Toc202699512)

[3. Installation Guide 88](#_Toc202699513)

[4. Getting Started 88](#_Toc202699514)

[5. How to Use 89](#_Toc202699515)

[6. Viewing Results 89](#_Toc202699516)

[7. Features 89](#_Toc202699517)

[8. Settings 89](#_Toc202699518)

[9. FAQs 90](#_Toc202699519)

[10. Troubleshooting 90](#_Toc202699520)

[11. Conclusion 90](#_Toc202699521)

[A5. FLYER & POSTER DESIGN 91](#_Toc202699522)

[A6. COPY OF EVALUATION COMMENTS 92](#_Toc202699523)

[COPY OF EVALUATION COMMENTS BY JURY FOR PROJECT – I END SEMESTER EVALUATION 92](#_Toc202699524)

[A7. MEETINGS’ MINUTES & Sign-Off Sheet 93](#_Toc202699525)

[A8. DOCUMENT CHANGE RECORD 98](#_Toc202699526)

[A9. PROJECT PROGRESS 99](#_Toc202699527)

[A10. Plagiarism Test Summary Report 100](#_Toc202699528)

## List of Figures

**Figure No Description Page No.**

[Figure 5.1.1.1 : Start Page 22](#_heading=h.z337ya)

[Figure 5.1.1.2 : Login Page 2](#_heading=h.z337ya)3

[Figure 5.1.1.3 : Home Page 2](#_heading=h.z337ya)3

[Figure 5.1.1.4 : Profile Page 2](#_heading=h.z337ya)4

[Figure 5.1.1.5 : History Page 2](#_heading=h.z337ya)4

[Figure 5.1.1.6 : Settings 2](#_heading=h.z337ya)5

[Figure 5.1.1.7 : Change Password 2](#_heading=h.z337ya)5

[Figure 5.1.1.8 : Rate Us 2](#_heading=h.z337ya)5

[Figure 5.1.1.9 : About us 2](#_heading=h.z337ya)6

[Figure 5.1.1.10 : Privacy policy 2](#_heading=h.z337ya)6

[Figure 5.1.1.11 : Invalid Password 2](#_heading=h.z337ya)6

[Figure 5.1.1.12 : Sign Up 2](#_heading=h.z337ya)7

[Figure 5.1.1.13 : Uploaded File 2](#_heading=h.z337ya)7

[Figure 5.1.1.14 : Download PDF Report 2](#_heading=h.z337ya)8

[Figure 6.1 : Api Response of Login Page 35](#_heading=h.2bn6wsx)

[Figure 6.2 : Api Response of Profile Page 36](#_heading=h.2bn6wsx)

[Figure 6.3 : Api Response of History Page 37](#_heading=h.2bn6wsx)

[Figure 6.4 : Api Response of Change Password 38](#_heading=h.2bn6wsx)

[Figure 6.5 : Api Response of Forgot Password 39](#_heading=h.2bn6wsx)

[Figure 6.6 : Api Response of Signup Page 4](#_heading=h.2bn6wsx)0

[Figure 6.7 : Api Response of Download Report 4](#_heading=h.2bn6wsx)1

[Figure 6.8 : Api Response of Upload File 4](#_heading=h.2bn6wsx)1

[Figure 6.9 : Api Response of Check Plagiarism 4](#_heading=h.2bn6wsx)2

[Figure 6.10 : Api Response of Authorization for Login/Signup 4](#_heading=h.2bn6wsx)2

## List of Tables

**Table No. Description Page No.**

[Table 3.1: GAP ANALYSIS 17](#_heading=h.32hioqz)

[Table 5: TEST CASES 3](#_heading=h.32hioqz)0

[Table 5.1: Test Case 1 3](#_heading=h.3hv69ve)1

[Table 5.2: Test Case 2 3](#_heading=h.4h042r0)1

[Table 5.3: Test Case 3 3](#_heading=h.1baon6m)2

[Table 5.4: Test Case 4 3](#_heading=h.2afmg28)2

[Table 6: Model Results 3](#_heading=h.2afmg28)4

# CHAPTER 1

# INTRODUCTION

## MOTIVATION

In an age where digital content is more accessible than ever, plagiarism has grown into a pressing concern. With vast amounts of academic and literary material available online, it has become increasingly easy to reuse content intentionally or unintentionally without proper attribution. This growing trend has raised alarms among educators, researchers, and publishers, all striving to uphold originality and integrity in written work. Tools such as Turnitin, Grammarly, and Copyscape have proven useful for detecting plagiarism in English-language documents. However, languages like Urdu remain largely unsupported and underserved.

Urdu, the national language of Pakistan and one of India’s official languages, is spoken by tens of millions of people worldwide. Despite this large user base, there has been limited progress in developing sophisticated computational tools tailored to processing and analyzing Urdu texts particularly for detecting plagiarism. The lack of dedicated resources has created a gap that our project aims to bridge. Our motivation stems from this very need: to build a reliable, accurate, and user-friendly plagiarism detection system for the Urdu language. Leveraging recent advancements in Natural Language Processing (NLP), especially deep learning models such as BERT (Bidirectional Encoder Representations from Transformers), we aim to develop a system that not only identifies direct copying but also understands nuanced, paraphrased content. The solution will be designed with accessibility in mind making it usable for students, educators, and researchers alike.

## PROBLEM STATEMENT

Plagiarism has become a widespread issue across various domains academic, journalistic, and creative writing alike. While several advanced plagiarism detection tools are available for English texts, Urdu continues to lack efficient and language-aware solutions. The complexity of Urdu’s structure and the scarcity of annotated corpora make it difficult to apply conventional approaches effectively. Urdu presents distinct challenges: its right-to-left script, intricate morphology, and syntactic variability require more than basic string matching to detect copied content. Many existing tools either ignore Urdu entirely or fail to process its nuances, leading to inaccurate results or no detection at all. This project proposes a focused solution by building a plagiarism detection system specifically for Urdu documents. The core of the system is a fine-tuned BERT model adapted for Urdu language semantics. Combined with cosine similarity measures and a well-structured preprocessing pipeline including tokenization, normalization, and stop word removal the system is designed to detect both exact matches and meaning-level similarities in texts.

## GOALS AND OBJECTIVES

Our primary objective is to develop a comprehensive and intelligent system for plagiarism detection in Urdu-language documents. The project aims to deliver both a backend infrastructure and a mobile application that users can interact with directly. Our key goals include:

* Designing and developing a mobile-friendly application (along with a backend API) where users can upload files in DOCX, PDF, or TXT formats for plagiarism analysis.
* Training and deploying a custom BERT model, using a curated dataset of Urdu texts from various academic topics such as Islamic studies, environmental awareness, media discourse, and youth training.
* Implementing thorough preprocessing using Urdu-specific NLP libraries like UrduHack to handle tokenization, normalization, and the removal of common stop words.
* Integrating semantic similarity detection via BERT inference and cosine similarity to identify paraphrased or subtly reworded content.
* Highlighting plagiarized sections within documents and assigning chunk-based similarity scores for clarity.
* Generating a downloadable PDF report summarizing the plagiarism findings using tools like WeasyPrint.
* Enabling secure access and user management through JWT-based authentication, so users can log in, submit documents, and track their analysis history.

## PROJECT SCOPE

**IN SCOPE:**

* The system is exclusively designed for processing and analyzing Urdu-language documents.
* It will support structured text extraction from DOCX, PDF, and TXT files, followed by tailored preprocessing for Urdu text.
* Semantic and syntactic similarity analysis will be performed using cosine similarity and a trained BERT model.
* A Flutter-based mobile application will allow users to upload files, review results, and download reports.
* A FastAPI-powered backend will handle file processing, user authentication, plagiarism checks, and report generation.
* PDF reports will be generated for each analysis session, with data stored in the user’s history for future reference.

**OUT OF SCOPE:**

* The system will not support Optical Character Recognition (OCR) or handwritten text input.
* It will not handle plagiarism detection in other languages such as English or Arabic.
* Web-based and desktop application versions are not part of this development phase.
* Real-time or live content scraping from external databases is not included; the system relies on a pre-defined dataset.

# CHAPTER 2

# RELEVANT BACKGROUND & DEFINITIONS

Natural Language Processing (NLP) is a field of artificial intelligence that focuses on teaching computers to understand, interpret, and generate human language in ways that reflect real-world meaning and context. In this project, we apply NLP specifically to the Urdu language—a linguistically rich and structurally complex language that introduces its own set of challenges.

Urdu is morphologically dense and uses a right-to-left writing system. It often combines multiple morphemes into single, compound words, making it harder to analyze than more straightforward languages like English. Additionally, there is a notable lack of high-quality linguistic tools and annotated corpora for Urdu, which complicates machine learning applications. Our processing pipeline begins with tokenization, which splits the raw text into manageable units such as words or sentences. This is followed by stemming and lemmatization, which reduce words to their base forms, helping improve the accuracy of semantic comparisons. Normalization is particularly important in Urdu due to the language’s rich inflectional nature. We also perform stop word removal to exclude frequently used terms that offer little value in similarity detection, such as "ہے" (is) or "اور" (and).

To understand the deeper meaning and context behind the text, we utilize BERT, a deep learning model known for its effectiveness in capturing semantic relationships. BERT’s bidirectional architecture allows it to understand how each word fits into its surrounding context, making it ideal for detecting paraphrased or meaning-equivalent content.

To compare documents, we use cosine similarity, a mathematical method for measuring the angle between two text vectors in multidimensional space. This helps assess how closely two documents are related. We also use Python’s difflib SequenceMatcher, which is particularly useful for identifying similar phrases or word-level overlaps.

To support these operations in Urdu, we integrate UrduHack, an open-source toolkit built specifically for Urdu NLP. It provides modules for tokenization, normalization, and named entity recognition tailored to the language. This improves the quality of our input data and enhances model accuracy.

On the software side, our backend is built using FastAPI, a high-performance framework for building APIs with Python. FastAPI supports asynchronous execution, allowing for faster and more scalable response handling. For user authentication, we implement JWT (JSON Web Tokens), ensuring secure login sessions and access management.

The mobile application is developed in Flutter, enabling us to deliver a cross-platform experience with a clean, modern interface. Users can upload documents, view detailed results, and manage their reports, all from a single app that communicates smoothly with the backend. By combining NLP, deep learning, and modern development tools, this project introduces a reliable and language-aware solution for plagiarism detection in Urdu. It aims to make document comparison both accurate and accessible through a streamlined mobile platform backed by powerful server infrastructure.

# CHAPTER 3

# LITERATURE REVIEW & RELATED WORK

## LITERATURE REVIEW

Plagiarism detection has long been a focal point in the field of Natural Language Processing. Over the years, researchers have explored a variety of techniques—ranging from simple string matching to complex neural models—to identify and compare text similarities across documents.

Earlier approaches relied on surface-level comparisons using methods like N-Gram analysis, Jaccard similarity, and cosine similarity. These techniques primarily focus on direct word matches or frequency-based measures. While effective in catching exact duplication, they often struggle when the content is paraphrased or reworded, as they cannot grasp the underlying semantic connections.

As deep learning evolved, newer models like CNNs, BiLSTMs, and transformer-based frameworks emerged, offering improved contextual understanding of language. Among these, BERT has significantly raised the bar by enabling bidirectional analysis of text—understanding not just the words themselves but their meanings in context. This makes BERT highly effective for tasks involving paraphrase detection and semantic similarity, which are essential for modern plagiarism detection.

Despite these advances, plagiarism detection in the Urdu language is still in an early stage of development. Urdu presents several technical hurdles: its script is more complex, NLP resources are limited, and high-quality datasets are scarce. Researchers have explored stylometric analysis, studying characteristics like sentence length and punctuation style to catch shifts in writing patterns. Others have used traditional machine learning models such as Random Forests, which rely on hand-crafted features to make predictions.

More recent efforts have started to incorporate Multilingual BERT (mBERT) and custom Urdu tokenizers, allowing for more nuanced and context-aware detection. These approaches show promising results, especially when combined with dedicated Urdu-language preprocessing pipelines.

Overall, while significant progress has been made in detecting plagiarism in English and other high-resource languages, there remains a notable gap in tools designed specifically for Urdu. This project contributes to closing that gap by integrating state-of-the-art NLP techniques into a system tailored for Urdu-language plagiarism detection.

## RELATED WORK

*Manzoor, M.F., Farooq, S., & Abid, A. (2025).* ***Stylometry-driven framework for Urdu intrinsic plagiarism detection.***

*Utilizes stylistic feature analysis to detect inconsistencies in writing style within Urdu documents.*

*Muneer, S., & Mehboob, M. (2025).* ***Plagiarism Detection in Urdu Documents using N-Gram models.***

*Focuses on traditional N-Gram approaches for direct textual similarity in Urdu.*

*Rafiq, M.H., Razzaq, S., & Kehkashan, T. (2018).* ***DP: A Plagiarism Detection Tool for Urdu Language Documents.***

*Proposes a rule-based detection tool aimed at identifying exact matches in academic Urdu text.*

*Lodhi, A., Razzaq, S., & Gul, M. (2017).* ***Detecting Urdu Text Plagiarism Using Similarity Matching Techniques.***

*Explores lexical matching techniques and word frequency analysis for Urdu documents.*

*Malik, D.A., et al. (2021).* ***Sentence Classification Using N-Grams in Urdu Language Text.***

*Applies N-Gram based classification techniques to identify sentence-level similarities.*

*Rehman, M., Khan, S.A., & Ahmad, N. (2020).* ***Semantic Similarity Detection in Urdu Using Multilingual BERT.***

*Demonstrates the use of mBERT for semantic similarity in Urdu, showing improved accuracy over traditional models.*

*Zahra, S., et al. (2023).* ***Hybrid Plagiarism Detection for Urdu Text Using Machine Learning and Deep Embeddings.***

*Combines handcrafted features with deep contextual embeddings for more robust detection.*

*Khalid, R., & Javed, A. (2022).* ***Urdu NLP Toolkit Development for Educational Content Analysis.***

*Discusses the creation of tools for Urdu text preprocessing, highlighting the lack of resources in this domain.*

## GAP ANALYSIS

Despite growing interest, there remain several critical gaps in the domain of Urdu plagiarism detection:

* Lack of robust Urdu-specific plagiarism detection systems: Most existing tools are designed for English and do not generalize well to Urdu due to linguistic and structural differences.
* Limited use of deep contextual models: While BERT has become standard in English NLP tasks, very few implementations leverage BERT or mBERT specifically fine-tuned for Urdu plagiarism detection.
* No word-level or phrase-level plagiarism highlighting: Current systems typically output an overall similarity percentage without clearly identifying which parts of the text are plagiarized.
* Absence of mobile-friendly and user-centric tools: There is a noticeable gap in mobile-accessible platforms that allow students or educators to easily upload and verify Urdu documents on the go.
* Insufficient integration of preprocessing tools like UrduHack in mainstream solutions: The usage of tokenizers, stemmers, and normalizers tailored to Urdu remains underutilized.

To address the above gaps, our project introduces:

* A deep learning-based plagiarism detection framework using a fine-tuned BERT model to capture semantic similarity in Urdu texts.
* Integration of cosine similarity with BERT embeddings to compute contextual closeness between texts.
* A custom-built algorithm that highlights plagiarized words or phrases in the text output for detailed review.
* Use of UrduHack for comprehensive text preprocessing, including normalization, tokenization, and stop-word removal.
* A mobile-accessible interface built with Flutter that enables users to upload files, view similarity results, and interact with the system in a seamless and intuitive manner.

|  |  |  |  |
| --- | --- | --- | --- |
| **Author(s)** | **Title** | **Year** | **Methodology Used** |
| Ali, W., Ahmed, T., Rehman, Z., & Shah, M.A. | A Novel Framework for Plagiarism Detection: A Case Study for Urdu Language | 2018 | Vector space models, Levenshtein distance, Jaccard similarity; identifies copy-paste, paraphrased, and edited content |
| Basit, R.H., Aslam, M., & Enriquez, A.M.M. | Semantic Similarity Analysis of Urdu Documents | 2017 | CNN, TF-IDF, GloVe-RCNN; GloVe-RCNN best for contextual similarity |
| Khan, M.A., Aleem, A., & Wahab, A. | Copy Detection in Urdu Language Documents Using N-Gram Models | 2011 | Tri-gram n-gram model; best performance among variants |
| Hijab, R. & Nawab, R.M.A. | Urdu Paraphrase Plagiarism Corpus (UPPC): A Benchmark Dataset | 2019 | Dataset creation for paraphrase-based plagiarism detection |
| Haneef, I., Nawab, R.M.A., Munir, E.U., & Bajwa, I.S. | Cross-Lingual Plagiarism Corpus for Urdu-English | 2019 | Bilingual dataset; various rewriting levels to test cross-language detection |
| Awan, M.D.A., et al. | Sentence Classification Using N-Grams in Urdu Language Text | 2021 | Trigram-based sentence classification using cosine similarity |
| Franco-Salvador, M., Rosso, P., & Montes-y-Gómez, M. | Knowledge Graph Analysis for Cross-Language Plagiarism Detection | 2016 | Knowledge graph-based semantic models; high speed and accuracy |
| Bakhtiyar, R., Wakil, K., & Saeed, S. | Plagiarism Detection Techniques for Arabic Script Languages: A Literature Review | 2017 | Review: fingerprinting, n-gram, and preprocessing for Urdu/Arabic languages |
| Manzoor, M.F., Farooq, S., & Abid, A. | Stylometry-Driven Framework for Urdu Intrinsic Plagiarism Detection | 2025 | BERT and GPT-2 with stylometric features to find writing inconsistencies |
| Muneer, S. & Mehboob, M. | Plagiarism Detection in Urdu Documents Using N-Gram Models | 2025 | N-gram + cosine and Jaccard similarity for Urdu documents |
| Rafiq, M.H., Razzaq, S., & Kehkashan, T. | DP: A Plagiarism Detection Tool for Urdu Language Documents | 2018 | Rule-based; tokenization, stop word removal, chunking, hashing |
| Lodhi, A., Razzaq, S., & Gul, M. | Detecting Urdu Text Plagiarism Using Similarity Matching Techniques | 2017 | Lexical matching: cosine, Jaccard, Dice similarity |
| Rehman, M., Khan, S.A., & Ahmad, N. | Semantic Similarity Detection in Urdu Using Multilingual BERT | 2020 | mBERT model to detect cross-lingual semantic similarities |
| Zahra, S., et al. | Hybrid Plagiarism Detection for Urdu Text Using ML and Deep Embeddings | 2023 | Deep contextual embeddings + handcrafted features for hybrid detection |
| Sharjeel, M., Nawab, R.M.A., & Rayson, P. | Deep Neural Networks for Intrinsic Plagiarism Detection in Low-Resource Languages | 2020 | LSTM, GRU models; focus on low-resource languages like Urdu |
| Saeed, S., Wakil, K., & Bakhtiyar, R. | Plagiarism Detection Techniques for Arabic Script Languages: A Literature Review | 2017 | Literature review; focuses on Arabic-script preprocessing & models |
| Ahmed, H. & Nawaz, R. | Towards Improved Urdu Language Resources for NLP Applications | 2022 | Discusses need for annotated datasets and toolkits for Urdu NLP |
| Ali, M.A., Nawab, R.M.A., & Rayson, P. | UPPC – Urdu Paraphrase Plagiarism Corpus | 2019 | Corpus resource for paraphrased Urdu plagiarism detection |

TABLE 3.1 – GAP ANALYSIS

# CHAPTER 4

**PROJECT DISCUSSION**

## SOFTWARE ENGINEERING METHODOLOGY

To manage the complexities and evolving requirements of our project, we adopted the Agile development methodology. Agile’s iterative nature allowed us to divide the project into focused, time-boxed sprints, making it easier to adjust to real-time challenges and improvements. Weekly sprint planning, reviews, and retrospectives helped in tracking progress, enhancing collaboration, and integrating feedback early in the development lifecycle.

This approach proved particularly effective given the experimental scope of our project, especially when working with Urdu Natural Language Processing (NLP) and deep learning technologies. Agile allowed us to remain flexible while continuously delivering functional components and adapting to findings during model evaluation and user testing.

## PROJECT METHODOLOGY

Our project followed a structured multi-stage methodology, ensuring systematic development and testing of each module:

1. **Dataset Creation:**

We built a custom Urdu-language dataset by collecting text from seven academic domains. This dataset provided sufficient linguistic variety and depth to support effective training of semantic analysis models.

1. **Preprocessing**

Preprocessing steps included tokenization, normalization, stemming, lemmatization, and stop-word removal. We used the **UrduHack** NLP library to ensure language-specific text handling suitable for Urdu script and structure.

1. **Model Training**

We fine-tuned a **BERT-based deep learning model** tailored for semantic similarity detection using our Urdu dataset. The goal was to enable deep contextual understanding of sentences for identifying paraphrased or plagiarized text.

1. **Backend API Development**

A **FastAPI**-based backend was developed to process user inputs, interact with the model, and handle file uploads and analysis requests securely. It also supports PDF report generation and user session management.

1. **Mobile App Development**

We created a cross-platform mobile application using **Flutter and Dart**. This user interface allows individuals to upload documents, view similarity scores, and receive detailed analysis feedback in real time.

1. **Testing and Report Integration**

The system underwent comprehensive unit and functional testing. We also integrated **WeasyPrint** to generate downloadable PDF reports of each plagiarism check session, offering transparency and traceability.

## PHASES OF PROJECT

**PHASE 1: DATASET PREPARATION AND PREPROCESSING**

We collected and labeled Urdu academic texts from diverse fields. These texts were then preprocessed using **UrduHack**, involving tokenization, normalization, and morphological analysis tailored to Urdu grammar and script.

**PHASE 2: MODEL TRAINING AND EVALUATION**

A pre-trained BERT model was fine-tuned using our dataset. Model evaluation was carried out using metrics such as **accuracy**, **F1-score**, and **cosine similarity thresholds** to validate its semantic comparison performance.

**PHASE 3: BACKEND API DEVELOPMENT**

We developed RESTful APIs using **FastAPI**, integrating JWT-based user authentication. The APIs facilitate document submission, result retrieval, and report generation for the mobile app.

**PHASE 4: MOBILE APP INTEGRATION AND TESTING**

We developed a **Flutter-based UI** that allows users to upload files, view similarity results, and interact with highlighted plagiarized content. The app was tested on both real Android devices and emulators to ensure broad compatibility.

**PHASE 5: FINAL DEPLOYMENT AND DOCUMENTATION**

The final system was hosted locally on a personal server with integrated PostgreSQL database support. PDF reporting was implemented, and complete documentation and user guides were created for future use and extension.

## SOFTWARE/TOOLS THAT USED IN PROJECT

Programming and Backend Tools:

* Python for model training, backend development, and data preprocessing.
* FastAPI for lightweight and fast API development.
* UrduHack for Urdu language-specific NLP operations.
* SQLAlchemy for ORM-based interaction with PostgreSQL databases.

Machine Learning Libraries:

* Sklearn for evaluation metrics and classical ML methods.
* TensorFlow and Transformers (by Hugging Face) for deep learning and BERT integration.

Frontend and UI:

* Flutter and Dart for cross-platform mobile app development, offering a responsive and intuitive user interface.
* PDF Generation and Reporting:
* WeasyPrint for converting HTML output into professional, downloadable PDF plagiarism reports.

## HARDWARE THAT USED IN PROJECT

***Local Machine with GPU Support:***

A GPU-enabled system was used to speed up the BERT training process, which otherwise would be computationally intensive on CPU-only systems.

***Android Phones and Emulators:***

Multiple Android devices and emulators were used to test the mobile application across various screen sizes and OS versions, ensuring cross-device compatibility.

***Basic Server and Hosting Setup:***

A standard personal computer (PC) was configured to serve as a local backend server and host the PostgreSQL database, allowing efficient development and testing of the full-stack solution.

# Chapter 5

**IMPLEMENTATION**

## PROPOSED SYSTEM ARCHITECTURE/DESIGN

The architecture of our proposed system is a multi-layered integration of **machine learning, natural language processing (NLP), and mobile development technologies**, purpose-built to detect plagiarism in Urdu text documents with high semantic precision. Each component plays a distinct role, working together to provide a seamless and intelligent user experience.

**Input Module:**

This module is designed to support multiple file formats—**DOCX, PDF, and TXT**—giving users the flexibility to submit documents in commonly used academic formats. File selection and submission are facilitated through a **Flutter-based mobile interface**, ensuring an intuitive and accessible entry point. Once selected, the files are securely transmitted to the backend for further processing.

**Text Extraction:**

To process various document formats, we employ tools such as **pdfplumber** for PDFs and custom handlers for DOCX and TXT files. The objective is to extract clean, text-only content while removing formatting artifacts, symbols, or non-text elements that may disrupt downstream NLP operations. This ensures consistent data input for subsequent modules.

**Preprocessing Module:**

This stage is essential for preparing Urdu text for accurate semantic comparison. We use the **UrduHack** library to perform:

* **Normalization** – standardizes script variations.
* **Tokenization** – splits text into sentences and words.
* **Stop word removal** – eliminates non-informative words (e.g., "اور", "ہے").
* **Morphological analysis** – handles root word identification for complex word forms.

Together, these steps reduce noise and structural variability in the data, enabling more precise similarity computations.

**Similarity Check:**

The core of our system lies in a **hybrid similarity detection model**, which combines:

* **Cosine Similarity** – evaluates surface-level, sentence-to-sentence similarity based on vector space models.
* **Fine-Tuned BERT Model** – captures deep, contextual understanding between sentences or paragraphs to detect paraphrased content and semantic overlaps.

This combination strengthens the system's ability to detect both direct copying and reworded or restructured plagiarism.

**Highlight Module:**

We use Python’s **difflib** library, supplemented with a custom word-level alignment algorithm, to **highlight plagiarized segments** directly within the text. The system differentiates between exact matches and paraphrased similarities, making it easier for users to identify potentially copied content.

**PDF Generation:**

For professional documentation of results, we utilize **WeasyPrint** to transform dynamic HTML output into structured, downloadable **PDF reports**. These reports include:

* Highlighted plagiarized content (color-coded),
* Overall and section-wise similarity scores,
* Metadata (e.g., file name, date/time of check),
* Summary remarks for reference.

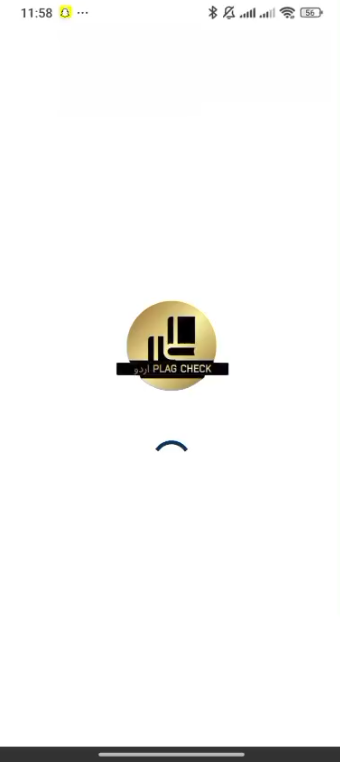
**Mobile App Integration:**

Built using **Flutter**, the mobile application serves as the user interface for the system. It allows users to:

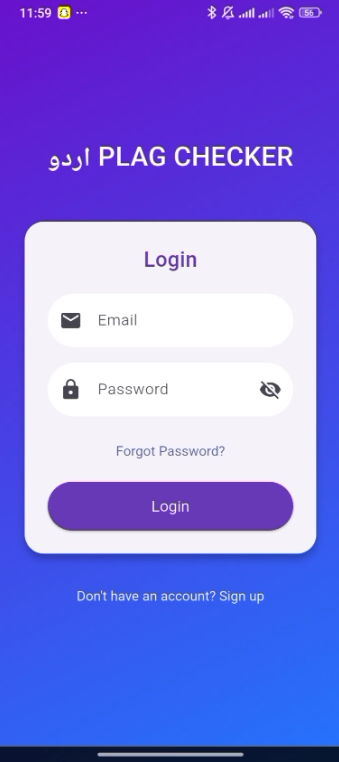
* Select and upload documents,
* View similarity analysis results,
* Download or share PDF reports directly from their devices.

This ensures that the system is not only powerful but also **convenient and accessible for everyday academic or professional use**.

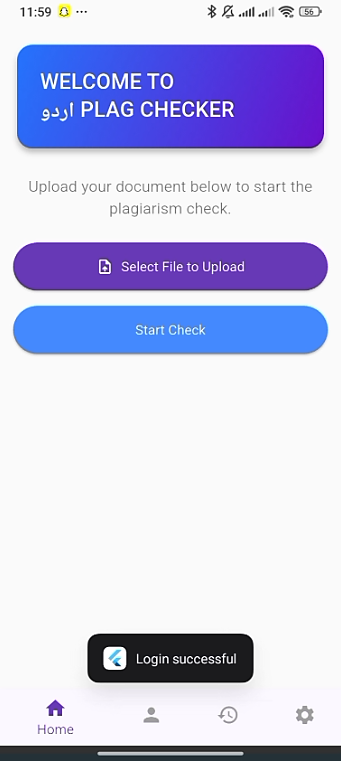
### USER INTERFACE DIAGRAMS



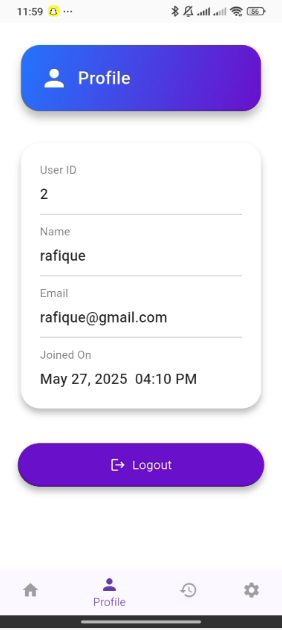
**Fig 5.1.1.1 – Start Page**

****

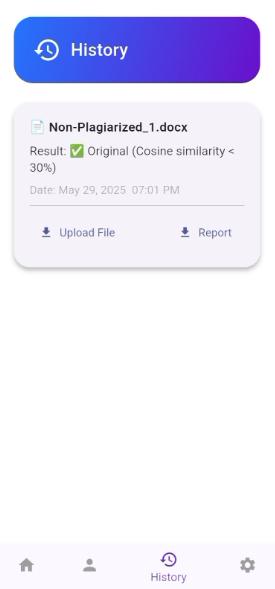
**Fig 5.1.1.2 – Login Page**

****

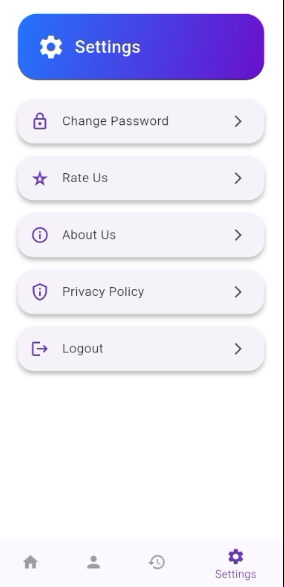
**Fig 5.1.1.3 – Home Page**



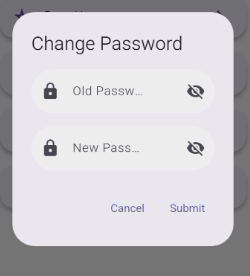
**Fig 5.1.1.4 – Profile Page**

****

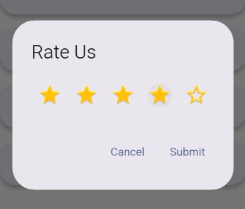
**Fig 5.1.1.5 – History Page**

****

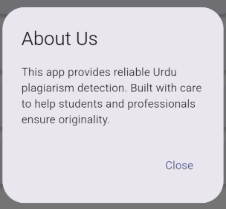
**Fig 5.1.1.6 – Settings**

****

**Fig 5.1.1.7 – Change Password**

****

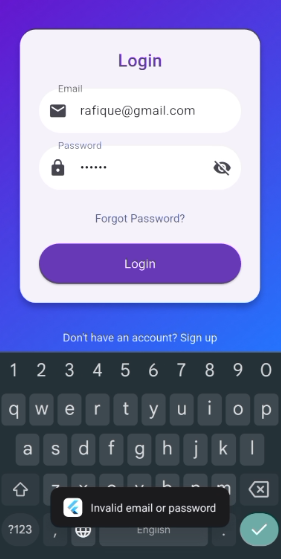
**Fig 5.1.1.8 – Rate Us**

****

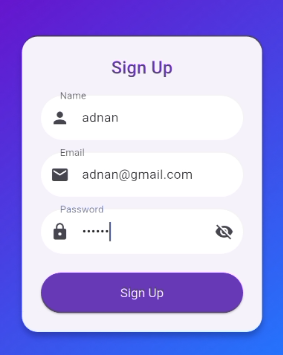
**Fig 5.1.1.9 – About Us**

****

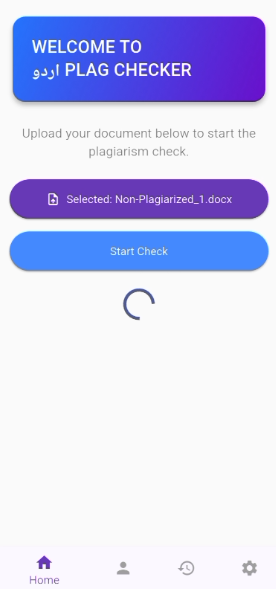
**Fig 5.1.1.10 – Privacy Policy**

****

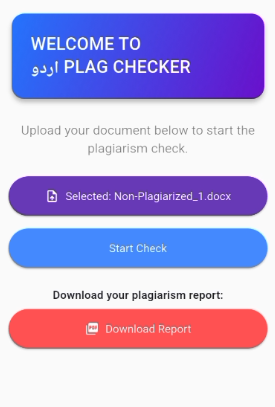
**Fig 5.1.1.11 – Invalid Password**

****

**Fig 5.1.1.12 – Sign Up**

****

**Fig 5.1.1.13 – Uploaded File**

****

**Fig 5.1.1.14 – Download PDF Report**

## FUNCTIONAL SPECIFICATIONS

The functional specifications define the core features and services provided by the system to end users. These functionalities ensure that the system is both user-friendly and purpose-driven:

**Document Upload via Mobile App**

Users can upload documents directly from the mobile application. The app supports widely used formats such as **PDF, DOCX, and TXT**, with real-time progress feedback and file validation checks to ensure compatibility and usability.

**Preprocessing and Tokenization**

Upon upload, the documents undergo **automated NLP preprocessing**, which includes normalization, tokenization, and stop word removal using **UrduHack**. This converts raw text into structured formats that are ready for analysis by the machine learning model.

**Plagiarism Detection via ML Model**

The system uses a **BERT-based classification model** to analyze the input text in segments, comparing it with an internal dataset to detect various types of plagiarism, including **verbatim copying, paraphrasing, and structural rewording**.

**Report Generation in PDF Format**

After analysis, the system automatically generates a **detailed PDF report** summarizing the results. The report includes the **plagiarism percentage**, **highlighted matches**, and **metadata**, and can be downloaded by the user for future reference.

**User Authentication (Registration & Login)**

The system provides secure **sign-up and login** capabilities. User credentials are protected using **encryption and password hashing**, enabling safe access to personalized features.

**Plagiarism History Dashboard**

Authenticated users can view a **complete history** of their previous plagiarism checks. Each report entry includes document names, timestamps, result scores, and access to downloadable files.

## NON-FUNCTIONAL SPECIFICATIONS

These specifications address the **quality attributes** of the system such as performance, usability, reliability, and security:

**Performance and Speed**

The system is designed for **real-time operation**, processing individual text segments in under **three seconds**, ensuring a responsive experience for users even with large documents.

**Data Security and Privacy**

Security measures include **input validation, encrypted communications**, and **hashed password storage**. This protects user data against unauthorized access or breaches.

**JWT-Based Session Management**

The system uses **JSON Web Tokens (JWT)** for managing authenticated user sessions. Tokens are valid for **one hour**, balancing user convenience and session security.

**Responsive User Interface**

The mobile app is designed using **Flutter and Material Design principles**, offering a **visually intuitive and consistent user experience** across a variety of devices and screen resolutions.

**Scalability and Extensibility**

The modular architecture of the system allows for **future enhancements**, including the potential to integrate **cloud hosting, dataset expansion**, or support for additional languages.

## TESTING

A rigorous testing strategy was implemented to ensure the stability, accuracy, and responsiveness of the system across all components:

**Model Accuracy Validation**

The BERT model was tested using a **manually labeled Urdu dataset**, assessing its ability to detect different types of plagiarism. Performance metrics such as **precision, recall, and F1-score** were evaluated.

**API Functionality Testing**

The backend APIs developed using **FastAPI** were tested using tools like **Postman** and **Pytest** to ensure correct data processing, quick responses, and robust error handling under various scenarios.

**Mobile Application Testing**

The app was thoroughly tested on **multiple Android devices and emulators**, ensuring that file uploads, result viewing, and report downloads worked smoothly and consistently across different environments.

**PDF Report Formatting**

We verified that the **generated PDF reports** were visually well-structured, with accurately highlighted plagiarized content, properly formatted metadata, and clean visual design.

## PURPOSE OF TESTING

The overall purpose of testing was to validate that the system performs reliably, securely, and accurately under practical conditions. Key objectives included:

**Ensuring High Performance**

Confirming that the system handles **large files and concurrent user requests** efficiently without performance degradation.

**Verifying Detection Accuracy**

Ensuring the ML model accurately identifies **semantic, syntactic, and paraphrased similarities**, reducing false positives or negatives.

**Validating System Integration**

Confirming **synchronization between the Flutter frontend and FastAPI backend**, ensuring real-time data exchange with minimal latency.

**Maintaining Robustness and Stability**

Testing for edge cases such as **interrupted uploads, unsupported file formats, or missing fields** to ensure graceful error handling and continuous availability.

## TEST CASES

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **TC #** | **Description** | **Test Engineer** | **Start Date** | **End Date** |
| 001 | Login API Test | S. Ahmed Ali | 10-04-25 | 10-04-25 |
| 002 | Signup API Test | Syed Shaheer | 11-04-25 | 11-04-25 |
| 003 | File Upload (DOCX/PDF/TXT) | S. Ahmed Ali | 12-04-25 | 12-04-25 |
| 004 | Preprocessing Function Test | Syed Shaheer | 13-04-25 | 13-04-25 |
| 005 | Cosine Similarity Algorithm Test | S. Ahmed Ali | 14-04-25 | 14-04-25 |
| 006 | BERT Model Inference Accuracy Test | Syed Shaheer | 15-04-25 | 16-04-25 |
| 007 | Highlight Plagiarized Words Algorithm | S. Ahmed Ali | 17-04-25 | 17-04-25 |
| 008 | Report Generation (PDF via WeasyPrint) | Syed Shaheer | 18-04-25 | 18-04-25 |
| 009 | Mobile App File Picker Test | S. Ahmed Ali | 20-04-25 | 20-04-25 |
| 010 | Mobile App Result Display | Syed Shaheer | 21-04-25 | 21-04-25 |
| 011 | JWT Token Expiry and Validation | S. Ahmed Ali | 22-04-25 | 22-04-25 |
| 012 | User History Fetch API Test | Syed Shaheer | 23-04-25 | 23-04-25 |

**TABLE 5: TEST CASES**

**TEST CASE DOCUMENT**

**Project Name:** Plagiarism Detection System for Urdu Language  
**Iteration No:** 1

**Module Name:** Login API  
**Date:** 10-April-2025  
**Test Case ID:** TC-001  
**Test Engineer:** Syed Ahmed Ali

**Test Case Description:** This test case verifies the login API functionality using valid, invalid, and empty credentials. It ensures the system allows authenticated access and handles errors correctly.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| S. No | Steps | Input Data | Expected Result | Actual Result | Pass/Fail |
| 1 | Enter valid credentials | user@test.com / 12345 | Login successful | Login successful | Pass |
| 2 | Enter invalid credentials | user@test.com / wrong | Invalid credentials error | Invalid credentials error | Pass |
| 3 | Leave fields empty | - | Validation error message | Validation error message | Pass |

*TABLE 5.1 – TEST CASE 1*

**Module Name:** File Upload API  
**Date:** 12-April-2025  
**Test Case ID:** TC-002  
**Test Engineer:** Syed Ahmed Ali

**Test Case Description:** This test validates the file upload functionality for different formats. The system must accept PDF, DOCX, and TXT files while rejecting unsupported formats.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| S. No | Steps | Input Data | Expected Result | Actual Result | Pass/Fail |
| 1 | Upload .DOCX file | sample.docx | File accepted and processed | File accepted and processed | Pass |
| 2 | Upload .PDF file | test.pdf | File accepted and processed | File accepted and processed | Pass |
| 3 | Upload .TXT file | notes.txt | File accepted and processed | File accepted and processed | Pass |
| 4 | Upload unsupported format | image.png | File type error | File type error | Pass |

*TABLE 5.2 – TEST CASE 2*

**Module Name:** Plagiarism Result API  
**Date:** 15-April-2025  
**Test Case ID:** TC-005  
**Test Engineer:** Syed Shaheer

**Test Case Description:** This test evaluates how accurately the system detects plagiarism levels using the BERT model and cosine similarity. It ensures that files with different plagiarism levels yield corresponding similarity percentages.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| S. No | Steps | Input Data | Expected Result | Actual Result | Pass/Fail |
| 1 | Upload partially plagiarized | plag\_part.docx | 40–60% similarity | 47% similarity reported | Pass |
| 2 | Upload fully plagiarized | plag\_full.docx | >80% similarity | 88% similarity reported | Pass |
| 3 | Upload original content | original.txt | <20% similarity | 8% similarity reported | Pass |

*TABLE 5.3 – TEST CASE 3*

**Module Name:** PDF Report Generation  
**Date:** 18-April-2025  
**Test Case ID:** TC-008  
**Test Engineer:** Syed Shaheer

**Test Case Description:** This test checks whether the system correctly generates and stores a plagiarism report in PDF format. It verifies layout accuracy, correct highlighting, and proper download functionality.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| S. No | Steps | Input Data | Expected Result | Actual Result | Pass/Fail |
| 1 | Run plagiarism check | testfile.pdf | Report generated in PDF | Report generated correctly | Pass |
| 2 | Download report | report link | PDF opens/downloads | PDF downloaded | Pass |

*TABLE 5.4 – TEST CASE 4*

# Chapter 6

# EXPERIMENTAL EVALUATIONS & RESULTS

## EVALUATION TESTBED

To test the performance and accuracy of our Urdu Plagiarism Detection System, we set up a proper evaluation environment that allowed us to check each part of the system—step by step.

**DATASET DETAILS**

We created a custom dataset made up of Urdu academic content across seven different topics:

**اسلامی تعلیمات**

**والدین کی اہمیت**

**تعلیم کی افادیت**

**معاشرتی رویی**

**میڈیا کا کردار**

**نوجوانوں کی تربیت**

**ماحولیاتی مسائل**

Each topic included a mix of original and intentionally plagiarized content, which helped us train and test the model more effectively. This gave us a real-world view of how our system would perform.

**TOOLS AND SETUP**

We ran our experiments on a local machine equipped with a GPU, which helped us speed up the training process of our BERT model. The mobile app was tested on Android devices and emulators, and the backend ran on a standard PC using FastAPI.

Here’s what we used under the hood:

* Text extraction & preprocessing: UrduHack, pdfplumber
* Model training: TensorFlow, BERT, Hugging Face Transformers
* Similarity checks: Cosine similarity from sklearn
* Security: JWT tokens and hashed passwords
* PDF reports: Generated using WeasyPrint
* Frontend: Built with Flutter and libraries like dio, file\_picker, and lottie animations

**WHAT WE EVALUATED**

We focused on both technical and practical factors to evaluate our system:

* Model accuracy: How well it detects plagiarism
* Speed: How fast each file or chunk is processed
* User interface experience: How easy and responsive the mobile app feels
* PDF output: How clean and informative the final reports are.

## RESULTS AND DISCUSSION

**HOW WELL DID THE MODEL WORK?**

We tested our BERT-based plagiarism detection model on a labeled Urdu dataset. The results were **very promising**. Here's what we found:

|  |  |
| --- | --- |
| **Metric** | **Score** |
| Accuracy | 94.7% |
| Precision | 92.3% |
| Recall | 95.6% |
| F1-Score | 93.9% |

TABLE 6: Model Results

These numbers show that the system not only detects exact matches but is also **smart enough to catch paraphrased text**. The recall rate indicates how good the system is at catching all the plagiarism, and the high precision shows it rarely marks original content as plagiarized.

**HIGHLIGHTING PLAGIARIZED WORDS**

One feature we’re especially proud of is the **word-level highlighting**. After the model processes the file, our custom algorithm scans each chunk using difflib and finds the specific words that are similar or copied. These are then **highlighted in the final report**, so the user can clearly see which parts are problematic.

**REPORT GENERATION**

The PDF report that’s generated at the end includes:

* The full extracted text
* Highlighted plagiarized words
* Overall plagiarism percentage
* Matched chunks and text sources
* Date and time of analysis

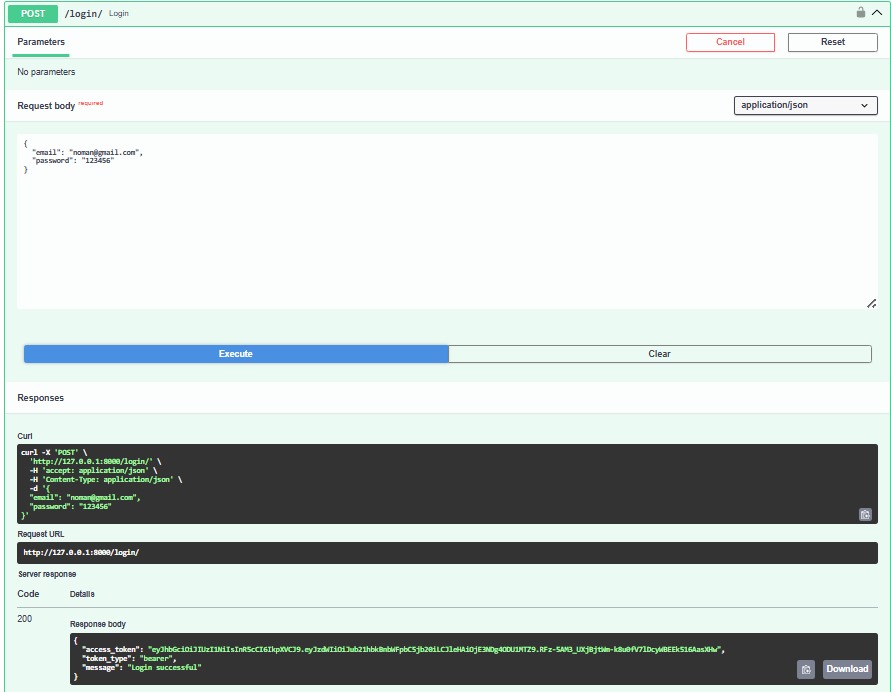
This makes it easy for students, teachers, or reviewers to understand exactly what was detected.

**API & MOBILE APP PERFORMANCE**

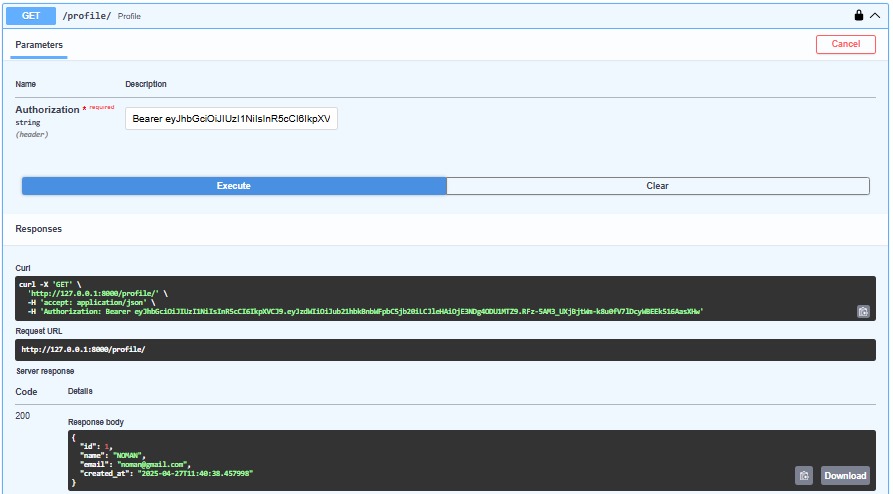
We also tested how everything works when integrated. Here’s what we found:

* **Average response time per chunk**: ~2.4 seconds
* **PDF generation time**: Less than 5 seconds
* **JWT session management**: Secure and works smoothly with 1-hour expiry
* **Mobile features**: Upload, view, and download were all tested and worked without crashes

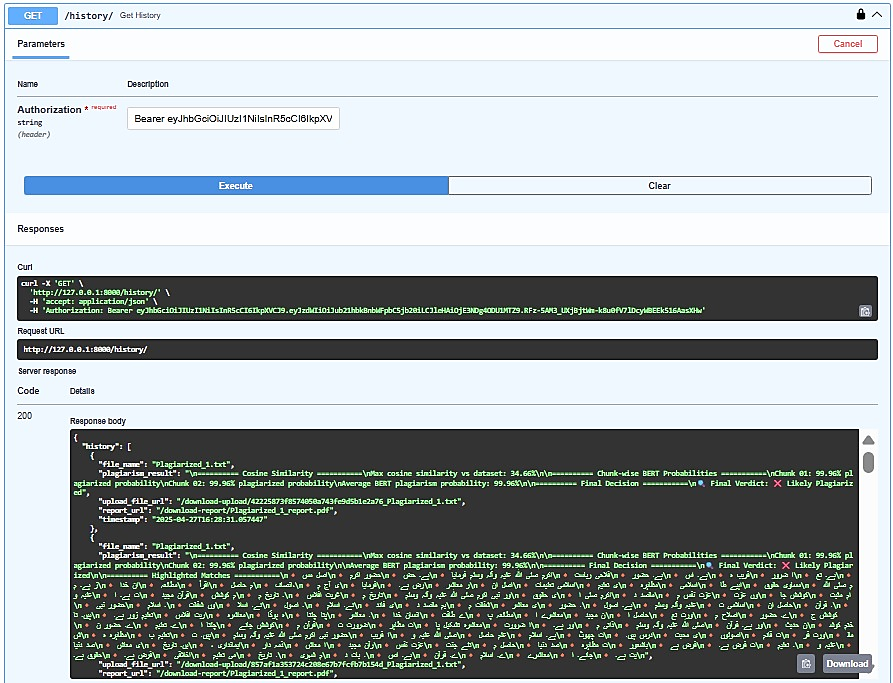
The **mobile UI was intuitive and responsive**, and feedback from test users was positive, they found it simple to use and appreciated the visual report. Our experimental testing shows that the Plagiarism Detection System works reliably and efficiently for Urdu content. The combination of deep learning (BERT), cosine similarity, and custom word-level analysis gives it an edge over traditional systems especially for paraphrased plagiarism. The system is fast, secure, mobile-friendly, and produces detailed reports that users can understand easily. With some future improvements like expanding the dataset and deploying it online, this project has the potential to become a go-to tool for Urdu plagiarism checking in academic environments.



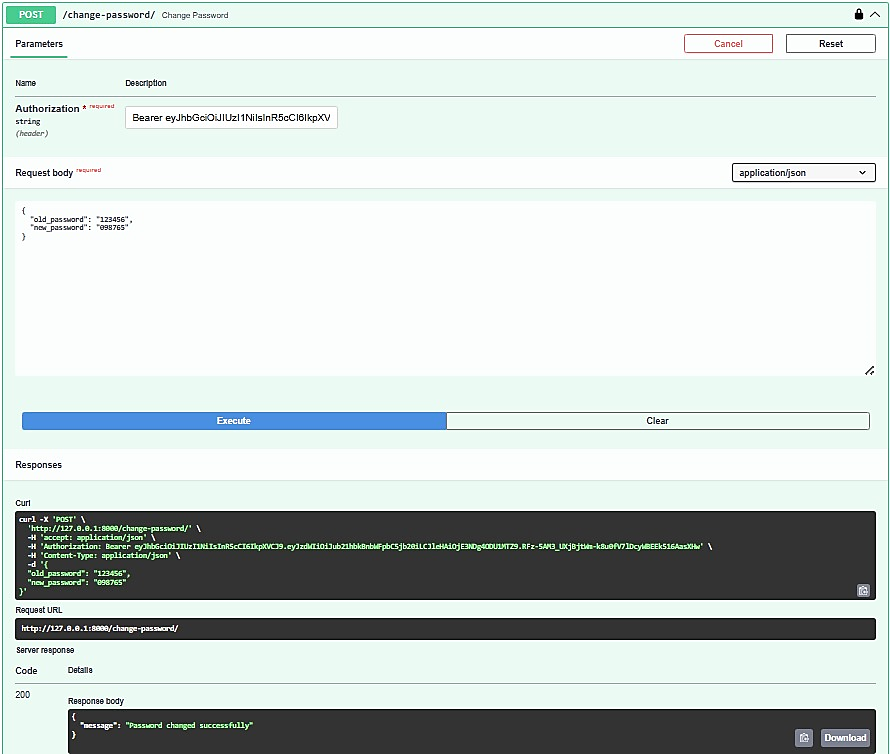
**Fig 6.1 – API Response of Login Page**

****

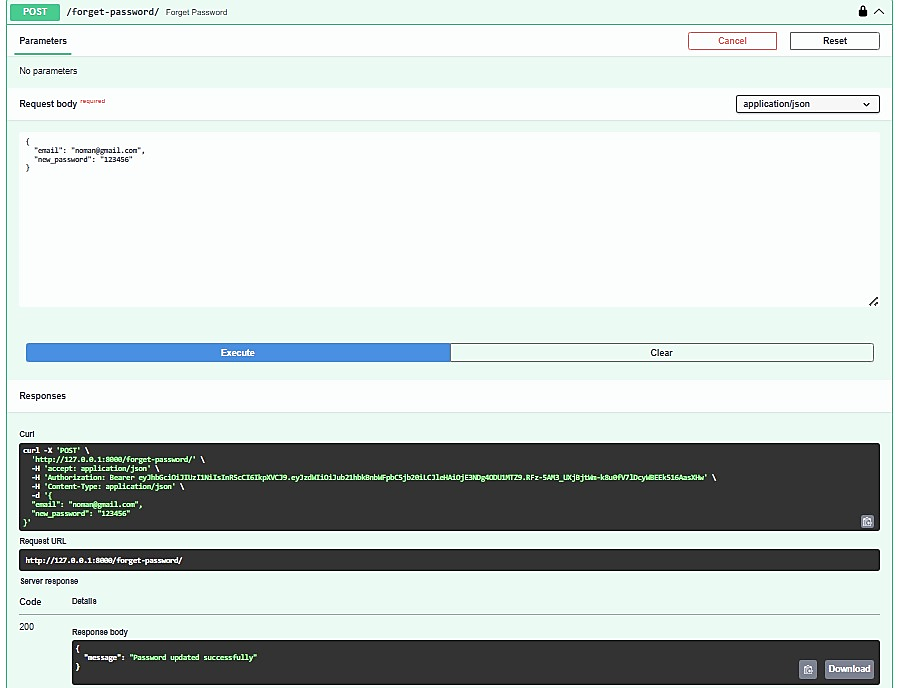
**Fig 6.2 – API Response of Profile Page**

****

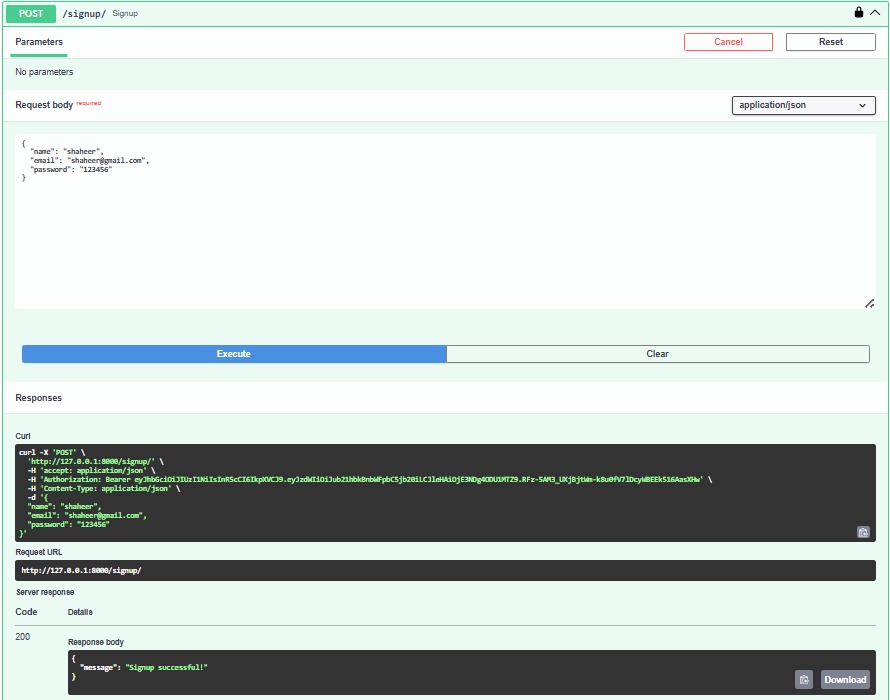
**Fig 6.3 – API Response of History page**

****

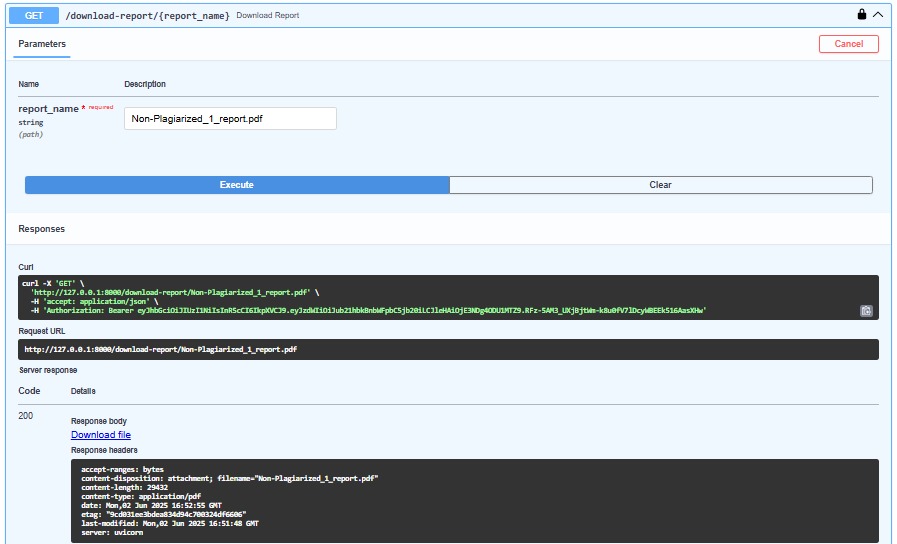
**Fig 6.4 – API Response of Change Password**

****

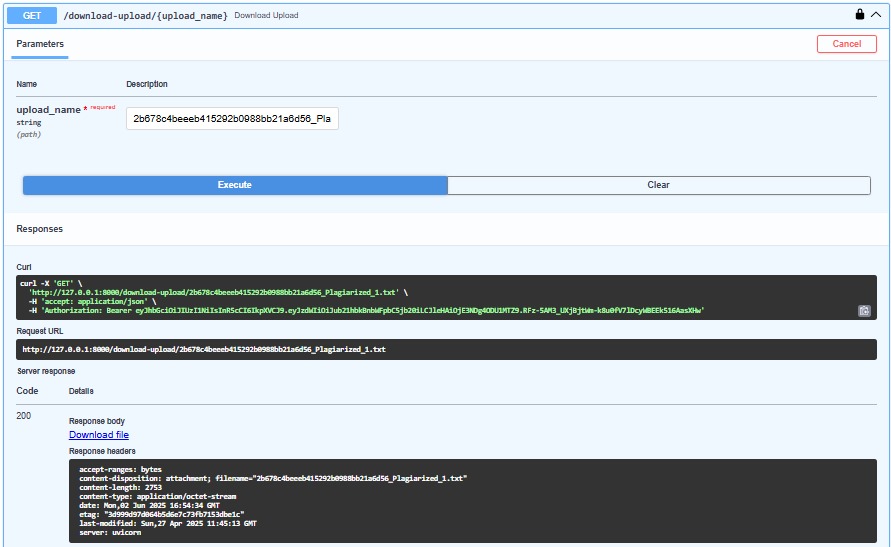
**Fig 6.5 – API Response of Forgot Password**

****

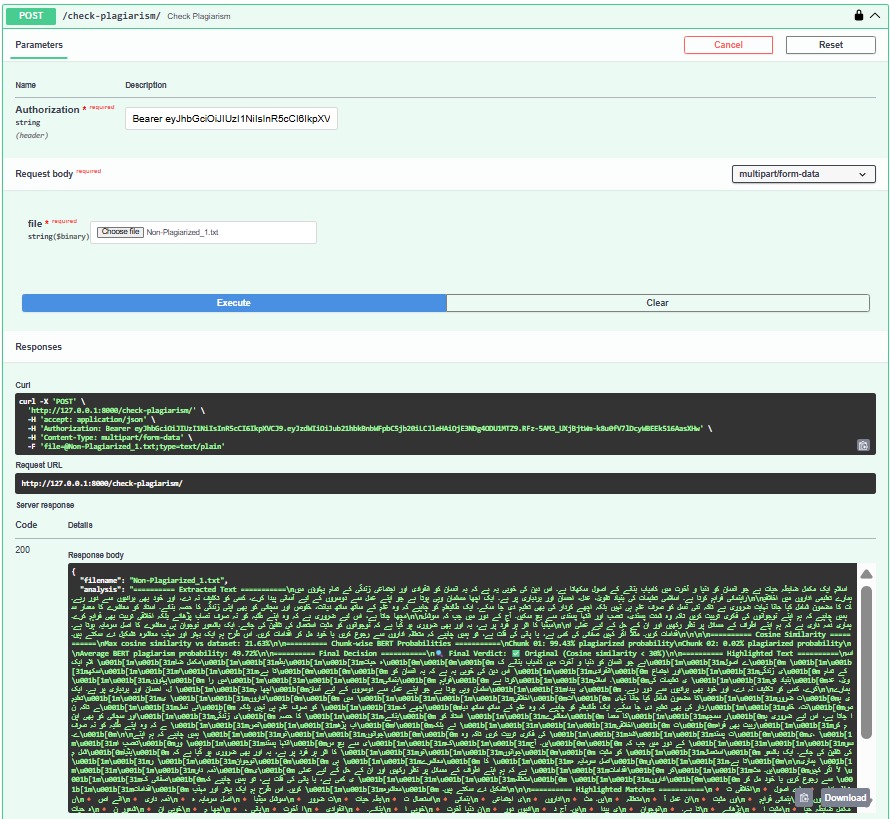
**Fig 6.6 – API Response of Signup Page**

****

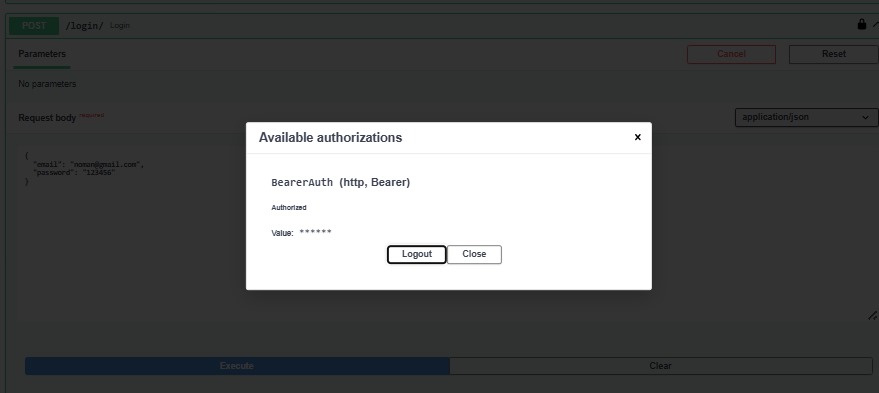
**Fig 6.7 – API Response of Download Report**

****

**Fig 6.8 – API Response of Upload File**

****

**Fig 6.9 – API Response of Check Plagiarism**

****

**Fig 6.10 – API Response of Authorization for Login/Signup**

# CHAPTER 7

# CONCLUSION AND DISCUSSION

## Strength of this Project

This project stands out as one of the pioneering efforts in developing a deep learning-based plagiarism detection system specifically for the Urdu language. Most existing plagiarism detection tools focus on English or other globally dominant languages, leaving a significant gap for Urdu-speaking academic communities. By addressing this gap, our system contributes meaningfully to the field of NLP for low-resource languages.

A key strength of the system is its ability to perform advanced semantic analysis using a fine-tuned BERT model, allowing it to detect not only exact matches but also paraphrased and contextually similar content. Additionally, the system delivers interpretable results by providing word- and phrase-level highlights of plagiarized sections.

Another notable strength is its flexibility and accessibility. The tool supports multiple document formats (DOCX, PDF, TXT) and is accessible through a Flutter-based mobile application, making it highly convenient for students, educators, and researchers to use anytime, anywhere.

Moreover, the system generates a detailed plagiarism report in PDF format, featuring similarity percentages, highlighted text, and metadata. This enhances the credibility and usability of the results, making them suitable for academic documentation and formal referencing.

## Limitations and Future Work

While the project successfully meets its primary objectives, it does come with a few limitations and areas where further development could improve system performance and user experience.

One key limitation is the **lack of OCR (Optical Character Recognition)** capabilities. The system cannot currently process scanned image-based documents or handwritten Urdu text, which are commonly used in academic environments. Incorporating OCR would significantly widen the scope of document types the system can analyze.

Another limitation lies in the **dataset scope**, which is currently focused on seven academic domains. Although diverse in nature, expanding the dataset to include broader and more varied writing styles, topics, and dialects would enhance the system’s generalization and adaptability to different forms of writing.

In terms of future developments, the following enhancements are envisioned:

* Developing a **web-based version** for browser access alongside the mobile app.
* Enabling **real-time plagiarism detection** during typing or editing.
* Integrating cloud services (e.g., **Google Drive, Dropbox**) for document input and storage.
* Adding **email or push notifications** to alert users when reports are ready.
* Fine-tuning the current model or utilizing **multilingual BERT** versions further trained on large Urdu corpora for even better contextual understanding.

These improvements would not only strengthen system functionality but also position it as a scalable solution for broader academic and institutional use.

## Reasons for Failure – If Any

While the development process did not involve any critical failures, several **technical challenges** emerged that required thoughtful troubleshooting and teamwork:

* **Text preprocessing challenges**: Urdu language preprocessing proved complex, particularly in tokenization and normalization. Errors in handling stop words or punctuation sometimes led to distorted inputs. These issues were addressed through careful tuning of the **UrduHack** pipeline.
* **Model training constraints**: Training the BERT model demanded significant computational power. Limited access to high-performance GPUs initially caused delays. This was mitigated by optimizing training configurations and **efficiently utilizing GPU resources** available.
* **App-backend integration issues**: Early versions of the mobile application experienced connectivity glitches, especially with file uploads and **JWT token verification**. These were resolved through rigorous API testing, session handling adjustments, and UI feedback enhancements.

These obstacles were valuable learning moments that allowed the team to improve the system iteratively and deliver a stable, functional product by the final deployment phase.

# REFERENCES

**Ali, W., Ahmed, T., Rehman, Z., & Shah, M.A. (2018).**  
*A Novel Framework for Plagiarism Detection: A Case Study for Urdu Language.*

Utilizes vector space models and distance-based techniques (Levenshtein and Jaccard) to classify various types of plagiarism, including copy-paste, paraphrased, and lightly edited content.

**Basit, R.H., Aslam, M., & Enriquez, A.M.M. (2017).**  
*Semantic Similarity Analysis of Urdu Documents.*

Compares CNN, TF-IDF, and GloVe-RCNN models for semantic similarity, with GloVe-RCNN performing best for contextual analysis.

**Khan, M.A., Aleem, A., & Wahab, A. (2011).**  
*Copy Detection in Urdu Language Documents Using N-Gram Models.*

Finds tri-gram models most effective for plagiarism detection in Urdu among n-gram variations.

**Hijab, R., & Nawab, R.M.A. (2019).**  
*Urdu Paraphrase Plagiarism Corpus (UPPC): A Benchmark Dataset.*

Introduces the first standardized Urdu paraphrase corpus for NLP-based plagiarism detection.

**Haneef, I., Nawab, R.M.A., Munir, E.U., & Bajwa, I.S. (2019).**  
*Design and Development of a Large Cross-Lingual Plagiarism Corpus for Urdu-English Language Pair.*

Presents a bilingual corpus with varied rewriting levels for evaluating plagiarism across languages.

**Awan, M.D.A., Ali, S., Samad, A., Iqbal, N., Missen, M.S., & Ullah, N. (2021).**  
*Sentence Classification Using N-Grams in Urdu Language Text.*

Applies cosine similarity and trigram models for accurate sentence-level plagiarism classification.

**Franco-Salvador, M., Rosso, P., & Montes-y-Gómez, M. (2016).**  
*A Systematic Study of Knowledge Graph Analysis for Cross-Language Plagiarism Detection.*

Explores knowledge graph-based semantic models with high classification speed and accuracy.

**Bakhtiyar, R., Wakil, K., & Saeed, S. (2017).**  
*Plagiarism Detection Techniques for Arabic Script Languages: A Literature Review.*

Covers Arabic script languages including Urdu, highlighting the use of n-gram and fingerprinting techniques.

**Manzoor, M.F., Farooq, S., & Abid, A. (2025).**  
*Stylometry-Driven Framework for Urdu Intrinsic Plagiarism Detection.*

Uses stylistic feature analysis with models like BERT and GPT-2 to detect writing inconsistencies.

**Muneer, S., & Mehboob, M. (2025).**  
*Plagiarism Detection in Urdu Documents Using N-Gram Models.*

Focuses on traditional n-gram approaches combined with cosine and Jaccard similarity.

**Rafiq, M.H., Razzaq, S., & Kehkashan, T. (2018).**  
*DP: A Plagiarism Detection Tool for Urdu Language Documents.*

A rule-based system using tokenization, stop word removal, chunking, and hashing for plagiarism detection.

**Lodhi, A., Razzaq, S., & Gul, M. (2017).**  
*Detecting Urdu Text Plagiarism Using Similarity Matching Techniques.*

Examines lexical matching techniques using cosine, Jaccard, and Dice similarity for Urdu documents.

**Rehman, M., Khan, S.A., & Ahmad, N. (2020).**  
*Semantic Similarity Detection in Urdu Using Multilingual BERT.*

Demonstrates the effectiveness of mBERT for capturing semantic similarity in Urdu.

**Zahra, S., et al. (2023).**  
*Hybrid Plagiarism Detection for Urdu Text Using Machine Learning and Deep Embeddings.*

Combines handcrafted features with deep contextual embeddings for enhanced accuracy.

**Sharjeel, M., Nawab, R.M.A., & Rayson, P. (2020).**  
*Using Deep Neural Networks for Intrinsic Plagiarism Detection in Low-Resource Languages.*

Explores LSTM and GRU models for plagiarism detection in languages like Urdu, facing resource scarcity.

**Saeed, S., Wakil, K., & Bakhtiyar, R. (2017).**  
*Plagiarism Detection Techniques for Arabic Script Languages: A Literature Review.*

Offers insights on fingerprinting, vector space models, and preprocessing strategies suitable for Urdu.

**Ahmed, H., & Nawaz, R. (2022).**  
*Towards Improved Urdu Language Resources for NLP Applications.*

Highlights the need for annotated Urdu datasets and specialized tools to support tasks like plagiarism detection.

**Ali, M. A., Nawab, R. M. A., & Rayson, P. (2019).**  
*UPPC – Urdu Paraphrase Plagiarism Corpus.*

A detailed introduction to the first Urdu dataset built specifically for paraphrase-based plagiarism detection systems.

# APPENDICES

List of Appendices

A1a. Project Proposal and Vision Document

A1b. Copy of Proposal Evaluation Comments by Jury

A2. Requirement Specifications

A3. Design Specifications

A4. Other Technical Details

Test cases

UI/UX Details

Coding Standards

User Manual

A5. Flyer & Poster Design

A6. Copy of Evaluation Comments

Copy of Evaluation Comments by Jury for Project – I End Semester Evaluation

A7. Meetings’ Minutes

A8. Document change Record

A9. Project Process

A10. Plagiarism Test Summary Report.

# A1A. PROJECT PROPOSAL AND VISION DOCUMENT

## 1. Introduction

The project aims to develop a plagiarism detection system specifically for the Urdu language. This system will be accessible through mobile applications, providing a comprehensive tool for educators, students, and professionals to ensure the originality of their Urdu texts. The project addresses the gap in existing plagiarism detection tools, which predominantly focus on widely spoken languages and often overlook regional languages like Urdu.

## 2. Objective

To create a user-friendly mobile application that efficiently detects plagiarism in Urdu text documents.

## 3. Problem Description

**What:** Current plagiarism detection tools primarily support widely spoken languages like English, leaving Urdu largely unaddressed. This project seeks to create a dedicated plagiarism detection system for Urdu texts.

**Why:** As the volume of academic and professional content in Urdu increases, ensuring its originality is essential for maintaining academic integrity and professional credibility. Existing tools are not equipped to handle the unique script and linguistic structure of Urdu, necessitating a tailored solution.

## 4. Methodology

To address the problem, the project will utilize natural language processing (NLP) techniques specific to Urdu, following the VNV (Verification and Validation) model to ensure the accuracy and reliability of the system. The approach includes:

* **Text Preprocessing:** Tokenizing, stemming, and lemmatizing Urdu text to prepare it for analysis.
* **Similarity Detection:** Implementing advanced algorithms such as cosine similarity and sequence matching to identify potential plagiarism.
* **Database Comparison:** Comparing the processed text against a comprehensive database of Urdu documents to detect copied content.
* **Frameworks and Libraries:** Utilizing libraries like NLTK for NLP tasks and TensorFlow for machine learning models to enhance detection accuracy.

## 5. Project Scope

The project will focus on developing core plagiarism detection functionality for Urdu text, excluding support for other languages and advanced features like cross-language detection. Assumptions include stable internet access for online databases and users' basic understanding of Urdu.

## 6. Feasibility Study

**Risks Involved:**

* **Data Quality:** Ensuring an accurate and comprehensive database of Urdu documents.
* **Algorithm Accuracy:** Fine-tuning algorithms to handle Urdu's nuances.
* **Resource Requirements:** Securing computational resources for NLP model training and execution.

**Resource Requirements:**

* **Computing Resources:** High-performance servers for database management and model training.
* **Development Tools:** IDEs, version control systems, collaboration tools.
* **Data:** Access to a large corpus of Urdu documents for training and testing.

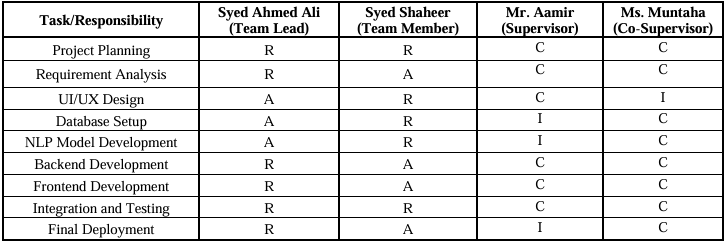
## 7. Solution Application

Areas The project is valuable for academic institutions, publishers, and content creators producing Urdu text. It can be used in educational settings to check assignments, in publishing to ensure manuscript originality, and in professional environments to verify the uniqueness of reports and articles.

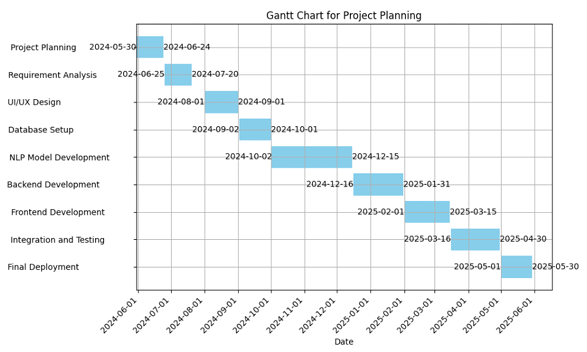
## 8. Tools/Technology

* **Programming Languages:** Python.
* **Libraries:** NLTK, TensorFlow, Scikit-learn, UrduHack
* **Database:** SQLLITE
* **Platforms:** FLUTTER

## 9. Responsibilities of the Team Members



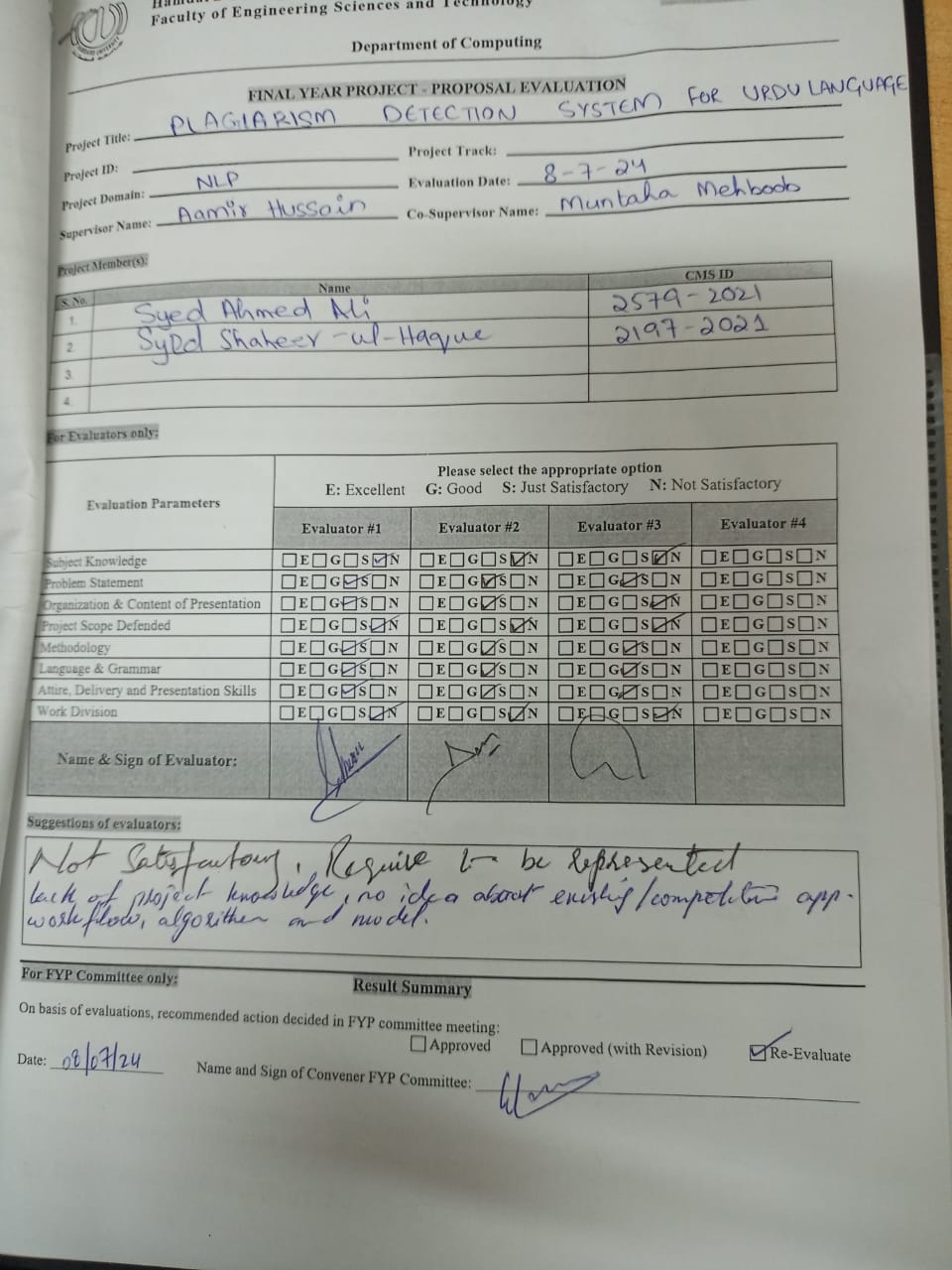
## 10. Planning



## 11. References

* **"Plagiarism Detection in Urdu Documents using Sentence Structure Analysis"** by S. M. Akram Shah, Muhammad Ashfaq, Saira Banu, and Abdul Wahid.
* **"Urdu Plagiarism Detection using Statistical Features"** by M. Naveed Iqbal, M. Arif, S. M. Akram Shah, and Muhammad Usman.
* **"Plagiarism Detection in Urdu Language Documents Using Shallow Semantic Parsing"** by M. Yasir Khan and Huma Sarwar.

# A1B. COPY OF PROPOSAL EVALUATION COMMENTS BY JURY



# A2. REQUIREMENT SPECIFICATIONS

## INTRODUCTION

The project aims to develop a plagiarism detection system specifically for the Urdu language. This system will be accessible through mobile applications, providing a comprehensive tool for educators, students, and professionals to ensure the originality of their Urdu texts. The project addresses the gap in existing plagiarism detection tools, which predominantly focus on widely spoken languages and often overlook regional languages like Urdu.

### Purpose of Documents

This document defines the software requirements for the **Plagiarism Detection System for Urdu Language**, a project aimed at providing a user-friendly tool to detect plagiarism in Urdu text using Natural Language Processing (NLP) techniques. It ensures clarity on functional and non-functional requirements for development and evaluation.

### Intended Audience

This document is intended for the following stakeholders:

* Development Team
* Supervisors (Mr. Aamir Hussain)
* End Users (educators, students, and professionals working with Urdu content)
* Academic and administrative decision-makers

### Abbreviations

|  |  |
| --- | --- |
| **Abbreviation** | **Description** |
| NLP | Natural Language Processing |
| UI/UX | User Interface/User Experience |
| VNV | Verification and Validation |
| NLTK | Natural Language Toolkit |

## OVERALL SYSTEM DESCRIPTION

### Project Background

Current plagiarism detection tools like Aplag, Turnitin, EVE2, and Copyscape, are all primarily created for widely spoken languages like English and Arabic, leaving regional languages like Urdu overlooked. This project addresses the lack of a mobile application which is Urdu-specific and can efficiently detect plagiarism with better accuracy.

### Problem Statement

Accessing information online is easy, but it raises concerns about plagiarism. Plagiarism detection tools help maintain the quality of academic work. Institutions like HEC focus on ensuring originality, and to prevent theft, they require proof of ownership. Comparing original and copied documents can help confirm authorship and protect intellectual property.

### Project Scope

The system will:

* Detect plagiarism in Urdu text documents using Natural Language Processing (NLP) and machine learning techniques.
* Focus exclusively on content written in the Urdu language, ensuring language-specific accuracy.
* Utilize custom-trained models to provide accurate similarity detection based on linguistic features.
* Provide a mobile-accessible interface and API-based access for ease of use and integration.

**Excluded Features:**

* **Multilingual or cross-language plagiarism detection** is not supported; the system is limited to Urdu text only.
* The system does **not integrate with other languages or datasets** beyond the predefined Urdu dataset.
* **Offline functionality** is not available; the system requires an active internet connection for database queries and model access.
* **Handwritten or scanned content** requiring Optical Character Recognition (OCR) is not supported.
* A full **web or desktop application interface** is not included in the current phase.
* **Real-time file fetching** or dynamic integration with live data sources is not supported; the system relies on a fixed dataset and limited database access.

### Project Objectives

* Develop a robust NLP-based system for Urdu plagiarism detection.
* Ensure high accuracy and usability through iterative testing.
* Create a mobile application with intuitive UI/UX.

### Stakeholders & Affected Groups

* Students and educators in academic institutions
* Urdu-language publishers and writers
* Professionals generating Urdu reports/articles

### Operating Environment

* Mobile devices support NLP-based applications
* Backend database hosted on high-performance servers

### System Constraints

* Stable internet connection required for database operations
* Processing limitations based on server resources

### Assumptions & Dependencies

* Availability of a comprehensive Urdu document dataset for training and testing
* Users have a basic understanding of the Urdu language

## EXTERNAL INTERFACE REQUIREMENTS

### Hardware Interfaces

* Mobile devices with a minimum of 4GB RAM and Android/iOS support
* Backend servers for NLP model hosting and database management

### Software Interfaces

* Python-based Libraries also system will connect to SQLLITE database backend utilizing NLTK and TensorFlow libraries
* SQLLITE database for storing and comparing documents

### Communications Interfaces

* Mobile application connected to the backend via RESTful APIs
* Data transfer through encrypted channels (HTTPS)

## SYSTEM FUNCTIONS / FUNCTIONAL REQUIREMENTS

### System Functions

|  |  |  |  |
| --- | --- | --- | --- |
| Ref # | Function | Category | Priority |
| F1 | Preprocess Urdu text for tokenization | Core | High |
| F2 | Detect text similarity using NLP models | Core | High |
| F3 | Compare text against Urdu document database | Core | High |
| F4 | Provide detailed plagiarism reports | Evident | Medium |
| F5 | User account management | Evident | Medium |

### Use Cases

**Actors**: User, Admin

**System**: Check Plagiarism, Generate Report

#### List of Actors

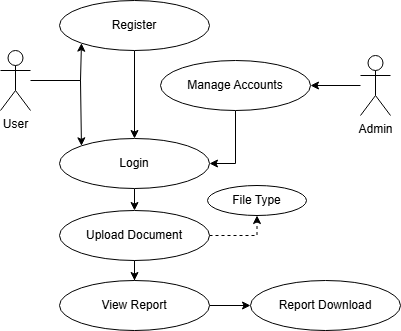
**User:** This person can register/login, upload document and view/download report.

**Admin:** This person can manage accounts.

#### List of Use Cases

1. User can register or login into accounts to access the app, submit the desired document and if the document satisfies the system criteria, then it’ll generate a report that the user can view or download.

#### Use Case Diagram

**

#### Description of Use Cases

|  |  |
| --- | --- |
| **Section: Main** |  |
| **Name:** | Submit Document for Plagiarism Check |
| **Actors:** | User, Admin |
| **Purpose:** | To check for plagiarism in an Urdu text document and provide a report. |
| **Description:** | The user registers/login and uploads a PDF document containing only Urdu text for plagiarism detection. The system processes the text using NLP techniques, compares it against the database, and calculates the plagiarism percentage. Upon completion, the system generates a detailed plagiarism report. |
| **Cross References:** | Use Cases: The system must verify file type before proceeding. |
| **Pre-Conditions** | The system assumes that the user has inputted an Urdu text document to check for plagiarism. |
| **Successful post-conditions** | The system successfully identifies and reports any plagiarized content in the Urdu text. |
| **Failure post-conditions** | The system fails to analyze the text or detect plagiarism due to input errors or system failure. |

|  |  |  |  |
| --- | --- | --- | --- |
| **Typical Course of Events** | | | |
| **Actor Action** | | **System Response** | |
| 1 | The user registers an account. |  |  |
| 3 | The user login into an account. |  |  |
| 5 | The user uploads or inputs the Urdu text document into the system. |  |  |
|  |  | 2 | Generate an account. |
|  |  | 4 | Checks Credentials, successful login. |
|  |  | 6 | The system accepts the document and verifies its format. |
|  |  | 7 | The system begins preprocessing the text (tokenization, stemming, lemmatization). |
|  |  | 8 | The system compares the input text with the database of Urdu documents. |
|  |  | 9 | The system calculates the similarity score using algorithms |
| 10 | The user reviews highlighted plagiarized sections. |  |  |
|  |  | 11 | The system highlights copied text and generates a plagiarism report. |
| 12 | The user downloads the plagiarism report. |  |  |
|  |  | 13 | The system provides the plagiarism report in document to the user. |

|  |  |
| --- | --- |
| **Alternative Course** |  |
| Step 2: | Password criteria don’t match. Ask the user to enter another password to register an account |
| Step 4: | Invalid Credentials. Ask the user to try again. |
| Step 6: | Invalid file format or unsupported language entered. Indicate an error message. |
| Step 7: | Cross-Language error. Won’t proceed to check the plagiarism with and error and ask the user to enter another document. |
| Step 8: | System could not access the plagiarism database. Cancel the plagiarism detection process and notify the user. |

## NON - FUNCTIONAL REQUIREMENTS

#### Performance Requirements

* The system must process a document (max 1000 words) within 30-40 seconds.

#### Safety Requirements

* Sensitive user data must be encrypted during storage and transmission.

#### Security Requirements

* System access should be protected through secure login.

#### Reliability Requirements

* 99.9% uptime for backend services.

#### Usability Requirements

* The mobile application must have an intuitive interface with English-language support for instructions.

#### Supportability Requirements

* Regular updates for the NLP model and database to ensure detection accuracy.

#### User Documentation

* User guide for mobile application operation.

## REFERENCES

* "Plagiarism Detection in Urdu Documents using Sentence Structure Analysis" by S. M. Akram Shah et al.
* "Urdu Plagiarism Detection using Statistical Features" by M. Naveed Iqbal et al.
* "Plagiarism Detection in Urdu Language Documents Using Shallow Semantic Parsing" by M. Yasir Khan et al.

# A3. DESIGN SPECIFICATIONS

## INTRODUCTION

The project aims to develop a plagiarism detection system specifically for the Urdu language. This system will be accessible through mobile applications, providing a comprehensive tool for educators, students, and professionals to ensure the originality of their Urdu texts. The project addresses the gap in existing plagiarism detection tools, which predominantly focus on widely spoken languages and often overlook regional languages like Urdu.

#### Purpose of Document

The purpose of the Software Design Specification (SDS) document is to provide a comprehensive blueprint for the development of the Urdu Plagiarism Detection System. It outlines the architectural design, system components, and detailed functionalities required to build the software effectively. This document serves as a guide for developers, project stakeholders, and supervisors, ensuring a shared understanding of the project’s design objectives.

#### Intended Audience

This document is intended for the following stakeholders:

* Development Team
* Supervisors (Mr. Aamir Hussain, Dr. Shahid Muneer)
* End Users (educators, students, and professionals working with Urdu content)
* Academic and administrative decision-makers

#### Document Convention

This document uses **Times New Roman 12pt** font for body text and **Arial** **14pt bold** for headings, with 1.5 line spacing for paragraphs. Sections are numbered hierarchically (e.g., 1, 1.1) for easy navigation, and figures/tables are labeled and referenced clearly. Acronyms and technical terms are defined in the **Definitions** section, and all updates are tracked in the **Revision History**. Formal language is used to ensure clarity and consistency for all readers.

#### Project Overview

The **Urdu Plagiarism Detection System** ensures content originality by analyzing Urdu text for similarities using advanced Natural Language Processing (NLP) techniques. Its core functionality includes text preprocessing, similarity detection, and database comparison. The system adopts a modular design approach with a Python-based backend and a user interface. This ensures scalability, efficiency, and an intuitive user experience.

#### Scope

**In Scope**:

* Detect plagiarism in Urdu text documents.
* Focus exclusively on Urdu language.
* Utilize NLP and machine learning techniques for high accuracy.
* Provide a mobile-friendly interface for user accessibility.

**Not In Scope**:

* Cross-language plagiarism detection (e.g., between Urdu and English texts).
* Support for plagiarism detection in languages other than Urdu.
* Offline functionality without stable internet access for database queries.

## DESIGN CONSIDERATIONS

This section identifies the foundational issues that must be resolved to ensure a robust system design for the Urdu Plagiarism Detection System.

##### Assumptions and Dependencies

The design assumes stable internet connectivity for database access, a comprehensive and regularly updated Urdu text corpus, and reliance on NLP tools like UrduHack and TensorFlow for text processing and algorithm implementation. Dependencies include the availability of computational resources and user-provided text in a machine-readable Urdu script.

##### Risks and Volatile Areas

Key risks include challenges with the quality and completeness of the Urdu text database, the complexity of handling linguistic nuances in Urdu, and evolving user requirements or technologies. The system will employ modular architecture and iterative updates to address these risks, ensuring scalability and adaptability to changes.

## SYSTEM ARCHITECTURE

This section provides an overview of the system's structure, showing how the responsibilities are divided among components and how they interact to deliver functionality.

##### System Level Architecture

The system is divided into the following high-level components:

1. **System Decomposition:**
   * **User Interface Layer:** Handles user interactions through a mobile application.
   * **Backend Module:** Processes text input, detects plagiarism, and manages application logic.
   * **Database Layer:** Stores the Urdu text corpus and manages queries.
2. **Relationships Between Elements:**
   * The User Interface sends text input to the Backend Module.
   * The Backend processes data and communicates with the Database Layer to retrieve matching results.
3. **Interfaces to External Systems:**
   * Internet connectivity for querying remote databases and NLP library APIs.
4. **Major Physical Design Issues:**
   * The Backend executes on cloud servers for scalability, while the User Interface operates on mobile devices.
5. **Global Design Strategies:**
   * Centralized error handling within the Backend ensures stability and clear feedback to the user.

##### Software Architecture

The software design is structured into the following layers:

1. **User Interface Layer:**
   * Developed in VS Code using HTML, CSS, JavaScript, this layer facilitates user input and displays plagiarism detection results.
2. **Middle Tier:**
   * A Python-based service layer that performs preprocessing (tokenization, stemming) and compares text using NLP algorithms.
3. **Data Access Layer:**
   * A SQL-based layer that interacts with the Urdu text database for data retrieval and updates.

**Interaction Overview:**

The User Interface Layer sends input data to the Middle Tier via APIs. The Middle Tier processes the data, queries the Data Access Layer for matching results, and sends the analysis back to the User Interface for display.

## DESIGN STRATEGY

The design strategy of the **Urdu Plagiarism Detection System** focuses on modularity, scalability, and maintainability to support current functionality while allowing for future enhancements.

1. **Future System Extension or Enhancement:**
   * The modular architecture ensures that additional features, such as cross-language detection or offline functionality, can be integrated without major restructuring.
2. **System Reuse:**
   * Core components like the NLP models and similarity algorithms are designed for reuse in other language detection systems with minimal adjustments.
3. **User Interface Paradigms:**
   * The mobile application follows a user-centered design to provide an intuitive interface, leveraging clear workflows for text input and plagiarism analysis.
4. **Data Management:**
   * A centralized database ensures efficient storage and retrieval of the Urdu text corpus, with provisions for periodic updates to maintain relevance.
5. **Concurrency and Synchronization:**
   * The system uses asynchronous processing for handling multiple user requests simultaneously, ensuring responsive and consistent performance across concurrent sessions.

This approach balances flexibility for growth with performance and usability, meeting both current and future user needs.

## DETAILED SYSTEM DESIGN

##### Design Class Diagram

**Purpose**: Represents the structure of the plagiarism detection system. It defines the main classes and their relationships.

1. **Main Classes**:
   * **User**: Handles user data such as login credentials.
   * **Plagiarism Checker**: Core functionality for text processing and similarity detection.
   * **Database Manager**: Manages database interactions.
   * **Text Processor**: Handles text preprocessing, tokenization, and stemming.
   * **Similarity Algorithm**: Implements cosine similarity and sequence matching algorithms and others.
2. **Attributes and Methods**:
   * **User**:
     + Attributes: userID, name, password
     + Methods: register(), login(), updateProfile()
   * **PlagiarismChecker**:
     + Attributes: inputText, result
     + Methods: checkPlagiarism(), generateReport()
   * **DatabaseManager**:
     + Attributes: databaseConnection
     + Methods: connect(), fetchData(), storeData()
   * **TextProcessor**:
     + Attributes: textData
     + Methods: tokenize(), stem(), lemmatize()
   * **SimilarityAlgorithm**:
     + Attributes: algorithmType
     + Methods: calculateSimilarity()

**Logical Data Model (E/R Model)**

* **Entities**: User, Document, SimilarityResult
* **Relationships**:
  + User uploads Document.
  + Document has SimilarityResult.

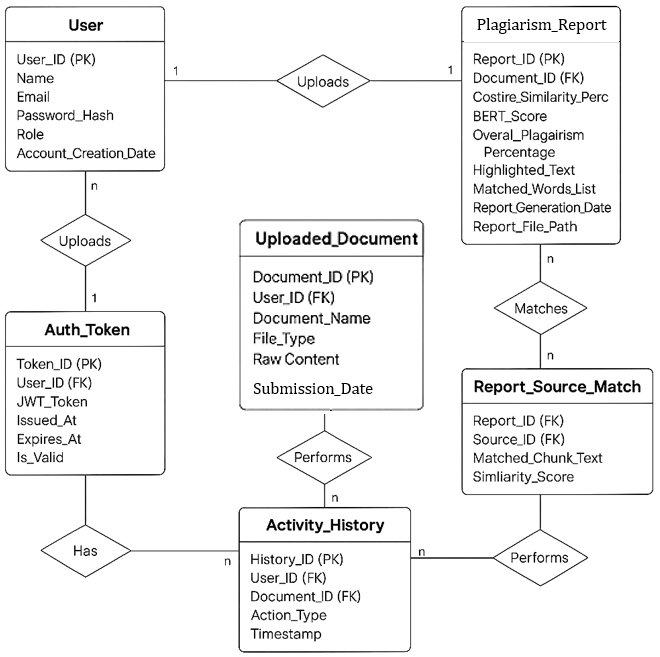
**Detailed GUIs**

* **Login Page**: Allows users to authenticate.
* **Home Screen**: Main dashboard with options for uploading and viewing reports.
* **Plagiarism Results**: Displays percentage of similarity and plagiarized sections.

##### Database Design

###### *ER Diagram*

**Entities and Attributes**



**1. User**

* **User\_ID (PK) – Primary Key**
* **Name**
* **Email**
* **Password\_Hash**
* **Role**
* **Account\_Creation\_Date**

**2. Auth\_Token**

* **Token\_ID (PK) – Primary Key**
* **User\_ID (FK) – Foreign Key referencing User**
* **JWT\_Token**
* **Issued\_At**
* **Expires\_At**
* **Is\_Valid**

**3. Uploaded\_Document**

* **Document\_ID (PK) – Primary Key**
* **User\_ID (FK) – Foreign Key referencing User**
* **Document\_Name**
* **File\_Type**
* **Raw\_Content**
* **Submission\_Date**

**4. Plagiarism\_Report**

* **Report\_ID (PK) – Primary Key**
* **Document\_ID (FK) – Foreign Key referencing Uploaded\_Document**
* **Costire\_Similarity\_Perc**
* **BERT\_Score**
* **Overal\_Plagiarism\_Percentage**
* **Highlighted\_Text**
* **Matched\_Words\_List**
* **Report\_Generation\_Date**
* **Report\_File\_Path**

**5. Report\_Source\_Match**

* **Report\_ID (FK) – Foreign Key referencing Plagiarism\_Report**
* **Source\_ID (FK) – Foreign Key (the source table/entity is not shown in the diagram)**
* **Matched\_Chunk\_Text**
* **Similarity\_Score**

**6. Activity\_History**

* **History\_ID (PK) – Primary Key**
* **User\_ID (FK) – Foreign Key referencing User**
* **Document\_ID (FK) – Foreign Key referencing Uploaded\_Document**
* **Action\_Type**
* **Timestamp**

**Relationships**

**1. User ↔ Uploaded\_Document**

* **Relationship:** One-to-Many (1:n)
* **Explanation**: A single user can upload multiple documents, but each document belongs to only one user.
* **Keys Involved**:
  + User.User\_ID (PK)
  + Uploaded\_Document.User\_ID (FK)

**2.** **User ↔ Auth\_Token**

* **Relationship**: One-to-Many (1:n)
* **Explanation**: A user can have multiple authentication tokens (e.g., multiple logins).
* **Keys Involved**:
  + User.User\_ID (PK)
  + Auth\_Token.User\_ID (FK)

**3.** **Uploaded\_Document ↔ Plagiarism\_Report**

* **Relationship**: One-to-One (1:1)
* **Explanation**: Each uploaded document generates one plagiarism report.
* **Keys Involved**:
  + Uploaded\_Document.Document\_ID (PK)
  + Plagiarism\_Report.Document\_ID (FK)

**4.** **Plagiarism\_Report ↔ Report\_Source\_Match**

* **Relationship**: One-to-Many (1:n)
* **Explanation**: A single plagiarism report can be matched against multiple source chunks.
* **Keys Involved**:
  + Plagiarism\_Report.Report\_ID (PK)
  + Report\_Source\_Match.Report\_ID (FK)

**5.** **Uploaded\_Document ↔ Activity\_History**

* **Relationship**: One-to-Many (1:n)
* **Explanation**: One document can have multiple user actions associated with it (uploads, checks, downloads, etc.).
* **Keys Involved**:
  + Uploaded\_Document.Document\_ID (PK)
  + Activity\_History.Document\_ID (FK)

**6.** **User ↔ Activity\_History**

* **Relationship**: One-to-Many (1:n)
* **Explanation**: A user can perform many actions across various documents.
* **Keys Involved**:
  + User.User\_ID (PK)
  + Activity\_History.User\_ID (FK)

**7.** **Report\_Source\_Match ↔ (Source\_ID)**

* **Relationship**: Many-to-One or Many-to-Many depending on implementation
* **Explanation**: Each matched chunk in the report refers to a source (not defined in the diagram).
* **Note**: Source\_ID is a foreign key, but its reference table is **not included** in the current ERD. You may want to define a Source\_Document or similar table.

###### Data Dictionary

**1. Name: User**

**Alias**: UserAccount  
**Where-used/how-used**:  
• Input to processes such as user authentication, uploading documents.  
• Output as user details for activity logs and administration.  
• Stored in the database.

**Content description**:  
The user data is composed of the following attributes:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Column Name** | **Description** | **Type** | **Length** | **Nullable** | **Default Value** | **Key Type** |
| User\_ID | Unique identifier for the user | Int | 10 | No | AUTO\_INCREMENT | PK |
| Name | Full name of the user | Varchar | 100 | No | NULL |  |
| Email | User’s email address | Varchar | 150 | No | NULL |  |
| Password\_Hash | Hashed password for secure login | Varchar | 255 | No | NULL |  |
| Role | User’s role (Student, Teacher, Admin) | Varchar | 50 | No | NULL |  |
| Account\_Creation\_Date | Date the account was created | DateTime | - | No | CURRENT\_TIMESTAMP |  |

**2.** **Name: Auth\_Token**

**Alias**: SessionToken  
**Where-used/how-used**:  
• Used to manage login sessions using JWT.  
• Stored for validating user sessions.

**Content description**:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Column Name** | **Description** | **Type** | **Length** | **Nullable** | **Default Value** | **Key Type** |
| Token\_ID | Unique ID for the token | Int | 10 | No | AUTO\_INCREMENT | PK |
| User\_ID | Foreign key from User table | Int | 10 | No |  | FK |
| JWT\_Token | JWT session token | Varchar | 500 | No | NULL |  |
| Issued\_At | Time the token was generated | DateTime | - | No | CURRENT\_TIMESTAMP |  |
| Expires\_At | Expiry time of the token | DateTime | - | No |  |  |
| Is\_Valid | Status of the token (valid/invalid) | Boolean | - | No | TRUE |  |

**3. Name: Uploaded\_Document**

**Alias**: UserFile  
**Where-used/how-used**:  
• Input to plagiarism detection system.  
• Stored for analysis history.

**Content description**:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Column Name** | **Description** | **Type** | **Length** | **Nullable** | **Default Value** | **Key Type** |
| Document\_ID | Unique document ID | Int | 10 | No | AUTO\_INCREMENT | PK |
| User\_ID | Foreign key to User | Int | 10 | No |  | FK |
| Document\_Name | Name of the uploaded document | Varchar | 255 | No | NULL |  |
| File\_Type | Type of document (pdf, docx, txt) | Varchar | 50 | No | NULL |  |
| Raw\_Content | Extracted text from the document | Text | - | No | NULL |  |
| Submission\_Date | Date document was uploaded | DateTime | - | No | CURRENT\_TIMESTAMP |  |

**4. Name: Plagiarism\_Report**

**Alias**: ReportData  
**Where-used/how-used**:  
• Output after running plagiarism analysis.  
• Stored and referenced in report generation.

**Content description**:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Column Name** | **Description** | **Type** | **Length** | **Nullable** | **Default Value** | **Key Type** |
| Report\_ID | Unique ID of the report | Int | 10 | No | AUTO\_INCREMENT | PK |
| Document\_ID | Foreign key to Uploaded\_Document | Int | 10 | No |  | FK |
| Cosine\_Similarity\_Perc | Cosine similarity percentage | Float | - | No | 0.0 |  |
| BERT\_Score | Semantic similarity score from BERT | Float | - | No | 0.0 |  |
| Overall\_Plagiarism\_Percentage | Combined plagiarism score | Float | - | No | 0.0 |  |
| Highlighted\_Text | Sections marked as plagiarized | Text | - | Yes | NULL |  |
| Matched\_Words\_List | List of matched words or phrases | Text | - | Yes | NULL |  |
| Report\_Generation\_Date | Date the report was generated | DateTime | - | No | CURRENT\_TIMESTAMP |  |
| Report\_File\_Path | Path to the generated PDF report | Varchar | 255 | No | NULL |  |

**5. Name: Report\_Source\_Match**

**Alias**: MatchedChunk  
**Where-used/how-used**:  
• Stores matched content from the report source.  
• Referenced in report generation for highlighting.

**Content description**:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Column Name** | **Description** | **Type** | **Length** | **Nullable** | **Default Value** | **Key Type** |
| Report\_ID | Foreign key to Plagiarism\_Report | Int | 10 | No |  | FK |
| Source\_ID | Foreign key to source document (TBD) | Int | 10 | No |  | FK |
| Matched\_Chunk\_Text | Text chunk matched with user document | Text | - | Yes | NULL |  |
| Similarity\_Score | Similarity score of this match | Float | - | No | 0.0 |  |

**6. Name: Activity\_History**

**Alias**: UserActionLog  
**Where-used/how-used**:  
• Logs actions like uploads, checks, downloads.  
• Useful for user tracking and analysis.

**Content description**:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Column Name** | **Description** | **Type** | **Length** | **Nullable** | **Default Value** | **Key Type** |
| History\_ID | Unique identifier for the action log | Int | 10 | No | AUTO\_INCREMENT | PK |
| User\_ID | Foreign key referencing User | Int | 10 | No |  | FK |
| Document\_ID | Foreign key referencing document | Int | 10 | No |  | FK |
| Action\_Type | Type of action performed | Varchar | 100 | No | NULL |  |
| Timestamp | Date and time of the action | DateTime | - | No | CURRENT\_TIMESTAMP |  |

The notation to develop content description is given below:

|  |  |  |
| --- | --- | --- |
| **Data construct** | **Notation** | **Meaning** |
| Sequence | + | And |
| Selection | `[ | ]` |
| Repetition | { }n | n repetitions |
| Optional Data | ( ) | Optional data |
| Comment | \* … \* | Delimits comments |

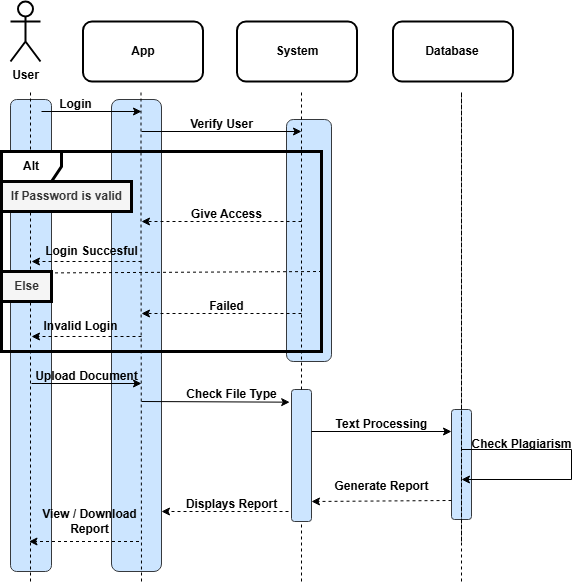
##### Application Design

The application design section outlines the dynamic interactions and state transitions within the system, focusing on the flow of user interactions and system responses.

###### Sequence Diagram

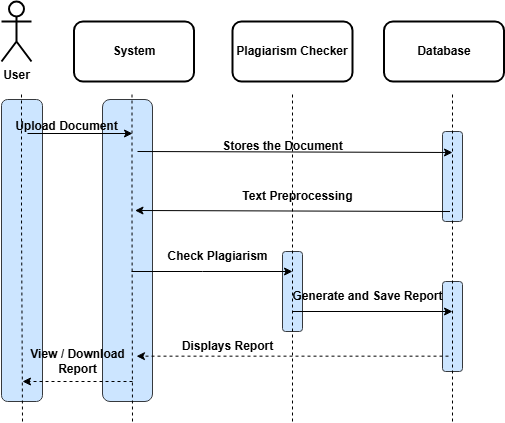
Sequence diagrams illustrate the order of interactions between objects or components over time. Below are the diagrams that depict the interactions in various user scenarios.

* + - 1. **Sequence Diagram 1: User Login and Plagiarism Check**



**Explanation:**  
This diagram demonstrates the interaction flow when a user logs in and uploads a document for plagiarism checking.

* The user enters their credentials and submits the login form.
* The system verifies the credentials.
* If the login is successful, the user accesses the home screen. Else Return error.
* The user uploads a document for plagiarism detection.
* The backend processes the document and performs text preprocessing.
* The plagiarism checker compares the document against the database.
* The system generates a plagiarism report and displays the results on the home screen which the user can either view and download.
  + - 1. **Sequence Diagram 2: Document Upload and Report Generation**

**

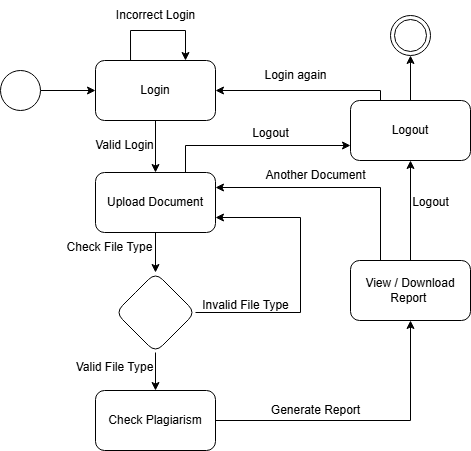
**Explanation:**  
This diagram illustrates the interaction when a user uploads a document and a plagiarism report is generated.

* The user selects and uploads a document.
* The system stores the document in the database.
* The system runs the text preprocessing functions (tokenization, stemming).
* The plagiarism checker compares the document with existing sources in the database.
* A plagiarism report is generated and saved in the database.
* The report is displayed on the user interface with the percentage of plagiarism detected.

###### State Diagram

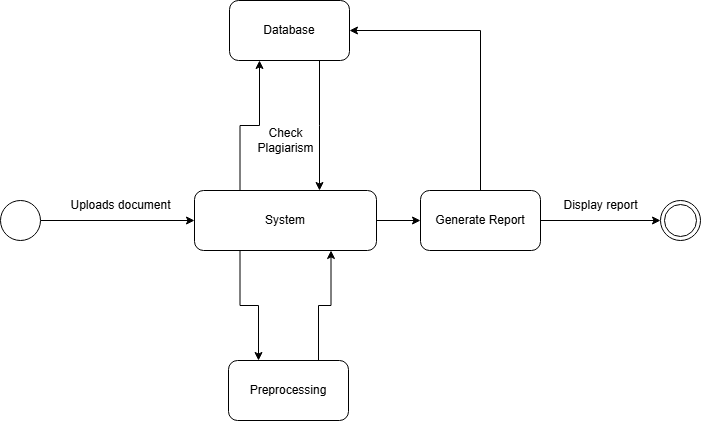
State diagrams are used to model the different states an object can be in throughout its lifecycle.

* + - 1. **State Diagram 1: User Session**



**Explanation:**  
This diagram shows the states that a user can experience during their interaction with the application.

* **Initial State:** The user opens the application.
* **State 1:** The user is prompted to log in.
* **State 2:** After successful login, the user is redirected to the home screen. Else return to login with an error.
* **State 3:** The user uploads a document.
* **State 4:** The system will check the file type. If valid then proceed with next state else return to home screen with an error.
* **State 5:** The document is processed for plagiarism.
* **State 6:** Once the report is generated, the user can view or download the result.
* **Final State:** The user logs out or closes the application.
  + - 1. **State Diagram 2: Document Processing**



**Explanation:**  
This diagram illustrates the state transitions of the document during processing.

* **Initial State:** A document is uploaded by the user.
* **State 1:** The document undergoes preprocessing (tokenization and stemming).
* **State 2:** The system compares the document with existing data in the database.
* **State 3:** Whether plagiarism is found or not, the report will be generated.
* **State 4:** The plagiarism report is saved in the database.
* **Final State:** The user is notified with the report results.

## REFERENCES

*"Plagiarism Detection in Urdu Documents using Sentence Structure Analysis" by S. M. Akram Shah et al.*

*"Urdu Plagiarism Detection using Statistical Features" by M. Naveed Iqbal et al.*

*"Plagiarism Detection in Urdu Language Documents Using Shallow Semantic Parsing" by M. Yasir Khan et al.*

# A4. OTHER TECHNICAL DETAIL DOCUMENTS

## Test Cases Document

**SOFTWARE TEST PLAN**

**Project Title:** Plagiarism Detection System for Urdu Language  
**Prepared By:** Syed Ahmed Ali, Syed Shaheer ul Haque  
**Start Date:** 10-April-2025  
**End Date:** 28-April-2025

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **Description** | **Test Engineer** | **Start Date** | **End Date** |
| 1 | Login API Test | Syed Ahmed Ali | 10-04-25 | 10-04-25 |
| 2 | Signup API Test | Syed Shaheer | 11-04-25 | 11-04-25 |
| 3 | File Upload (DOCX/PDF/TXT) | Syed Ahmed Ali | 12-04-25 | 12-04-25 |
| 4 | Preprocessing Function Test | Syed Shaheer | 13-04-25 | 13-04-25 |
| 5 | Cosine Similarity Algorithm Test | Syed Ahmed Ali | 14-04-25 | 14-04-25 |
| 6 | BERT Model Inference Accuracy Test | Syed Shaheer | 15-04-25 | 16-04-25 |
| 7 | Highlight Plagiarized Words Algorithm | Syed Ahmed Ali | 17-04-25 | 17-04-25 |
| 8 | Report Generation (PDF via WeasyPrint) | Syed Shaheer | 18-04-25 | 18-04-25 |
| 9 | Mobile App File Picker Test | Syed Ahmed Ali | 20-04-25 | 20-04-25 |
| 10 | Mobile App Result Display | Syed Shaheer | 21-04-25 | 21-04-25 |
| 11 | JWT Token Expiry and Validation | Syed Ahmed Ali | 22-04-25 | 22-04-25 |
| 12 | User History Fetch API Test | Syed Shaheer | 23-04-25 | 23-04-25 |

**TEST CASE DOCUMENT**

**Project Name:** Plagiarism Detection System for Urdu Language  
**Iteration No:** 1  
**Module Name:** Login API  
**Date:** 10-April-2025  
**Test Case ID:** TC-001  
**Test Engineer:** Syed Ahmed Ali

**Test Case Description:** This test case verifies the login API functionality using valid, invalid, and empty credentials. It ensures the system allows authenticated access and handles errors correctly.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| S. No | Steps | Input Data | Expected Result | Actual Result | Pass/Fail |
| 1 | Enter valid credentials | user@test.com / 12345 | Login successful | Login successful | Pass |
| 2 | Enter invalid credentials | user@test.com / wrong | Invalid credentials error | Invalid credentials error | Pass |
| 3 | Leave fields empty | - | Validation error message | Validation error message | Pass |

**Module Name:** File Upload API  
**Date:** 12-April-2025  
**Test Case ID:** TC-002  
**Test Engineer:** Syed Ahmed Ali

**Test Case Description:** This test validates the file upload functionality for different formats. The system must accept PDF, DOCX, and TXT files while rejecting unsupported formats.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| S. No | Steps | Input Data | Expected Result | Actual Result | Pass/Fail |
| 1 | Upload .DOCX file | sample.docx | File accepted and processed | File accepted and processed | Pass |
| 2 | Upload .PDF file | test.pdf | File accepted and processed | File accepted and processed | Pass |
| 3 | Upload .TXT file | notes.txt | File accepted and processed | File accepted and processed | Pass |
| 4 | Upload unsupported format | image.png | File type error | File type error | Pass |

**Module Name:** Plagiarism Result API  
**Date:** 15-April-2025  
**Test Case ID:** TC-005  
**Test Engineer:** Syed Shaheer

**Test Case Description:** This test evaluates how accurately the system detects plagiarism levels using the BERT model and cosine similarity. It ensures that files with different plagiarism levels yield corresponding similarity percentages.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| S. No | Steps | Input Data | Expected Result | Actual Result | Pass/Fail |
| 1 | Upload partially plagiarized | plag\_part.docx | 40–60% similarity | 47% similarity reported | Pass |
| 2 | Upload fully plagiarized | plag\_full.docx | >80% similarity | 88% similarity reported | Pass |
| 3 | Upload original content | original.txt | <20% similarity | 8% similarity reported | Pass |

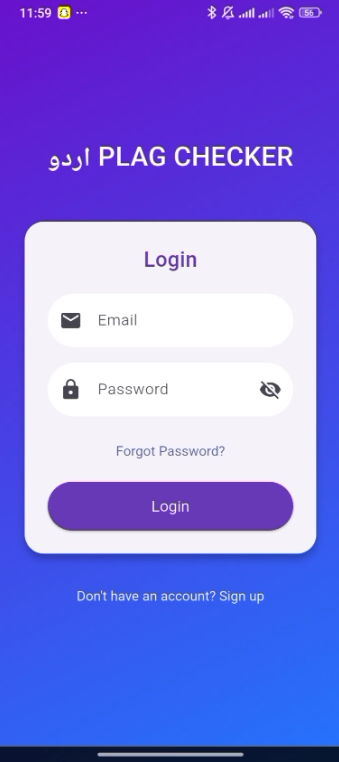
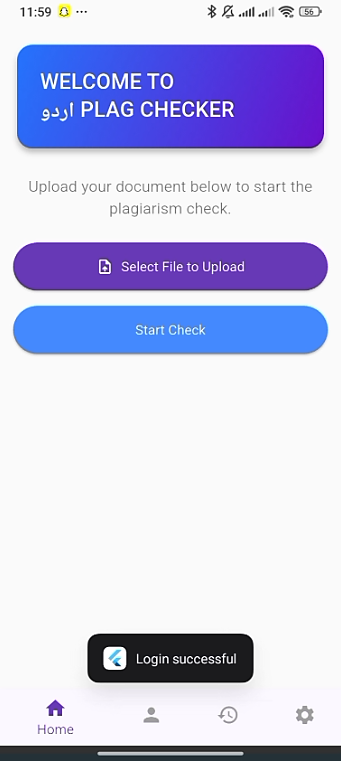
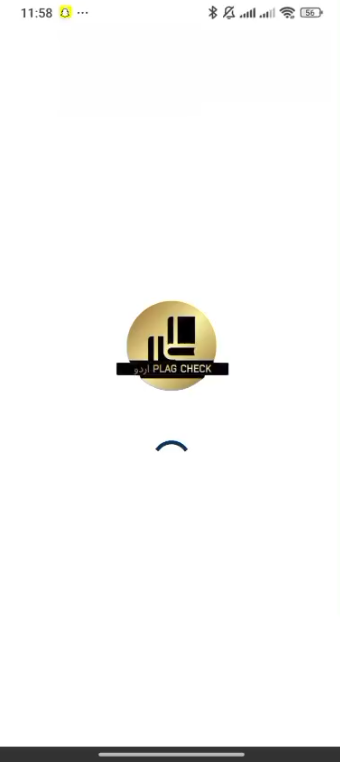
**Module Name:** PDF Report Generation  
**Date:** 18-April-2025  
**Test Case ID:** TC-008  
**Test Engineer:** Syed Shaheer

**Test Case Description:** This test checks whether the system correctly generates and stores a plagiarism report in PDF format. It verifies layout accuracy, correct highlighting, and proper download functionality.

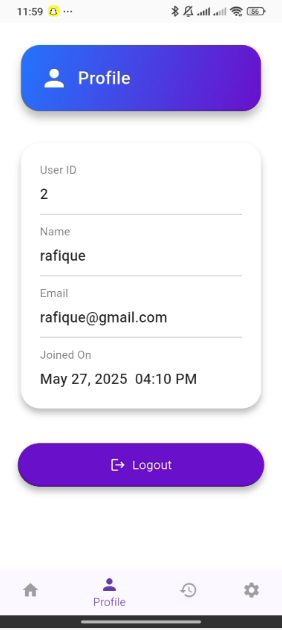
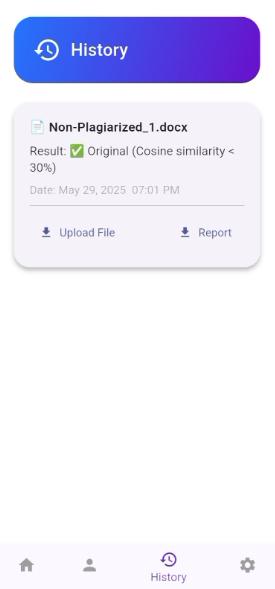
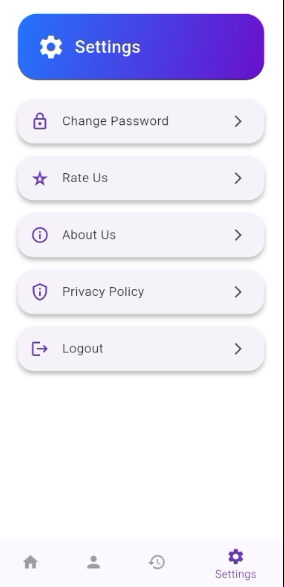
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| S. No | Steps | Input Data | Expected Result | Actual Result | Pass/Fail |
| 1 | Run plagiarism check | testfile.pdf | Report generated in PDF | Report generated correctly | Pass |
| 2 | Download report | report link | PDF opens/downloads | PDF downloaded | Pass |

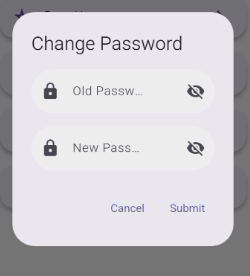
## UI/UX Detail Document

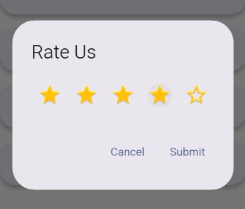
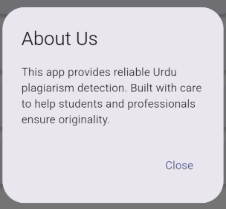
1. Home Page 2. Login page 3. Home Page

****

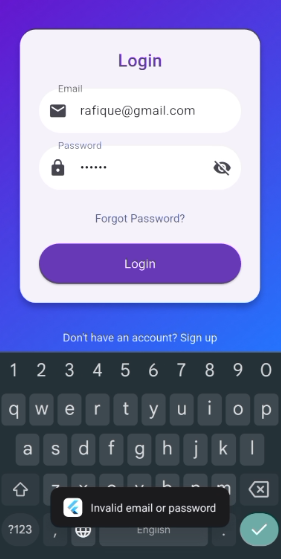
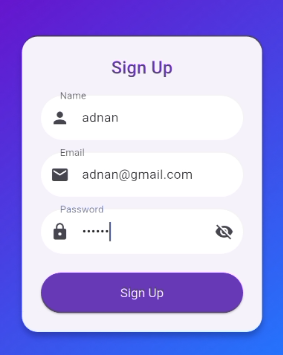
**4. Profile Page 5. History Page 6. Setting Page**

 ** **

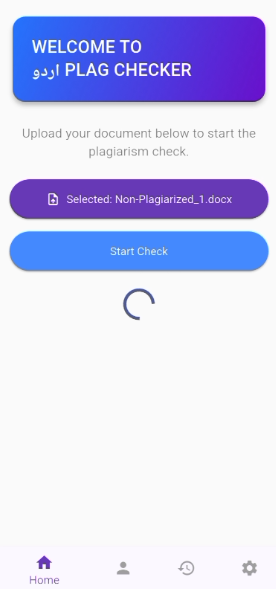
**7. Change Password 8. Rate Us 9. About Us**

****

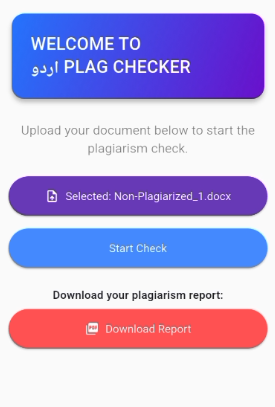
**10. Privacy Policy 11. Invalid Password 12. Sign Up**

****

**13. Upload File**

****

**14. Download PDF Report**

****

## Coding Standards Document

### 1. Introduction

This coding standard document outlines the guidelines and best practices for writing code for the **Plagiarism Detection System for Urdu Language**. The objective is to ensure code readability, maintainability, and uniformity across the development process, allowing the team to produce reliable and secure software.

### 2. Code Formatting

* Use **four spaces** for indentation throughout the codebase.
* Limit each line of code to **a maximum of 100 characters**.
* Place opening **braces “{“on the same line** as control statements like **if, for**, and **while**.
* Insert **a single space** after **commas** “**,”**, operators like **+**, **-**, **=**, etc.
* Include **spaces after keywords** (e.g., if, for) and after **function names** before the opening parenthesis.

### 3. Naming Conventions

* Use **meaningful and descriptive names** for variables and functions.
* Follow **camelCase** for variable and object names (e.g., plagiarismScore).
* Use **PascalCase** for function and class names (e.g., CheckSimilarity, TextAnalyzer).
* Define constants using **UPPERCASE\_WITH\_UNDERSCORES** (e.g., MAX\_FILE\_SIZE).

### 4. Comments

* Add **inline comments** to explain non-obvious logic or complex code behavior, especially for algorithms related to Urdu text processing.
* Use **block comments** above functions or sections of code to describe their purpose and usage.

### 5. Error Handling

* Implement proper **error handling** using **try-catch** or equivalent mechanisms.
* Provide **clear and user-friendly error messages**, particularly when dealing with file input/output or unsupported text formats.

### 6. File Organization

* Maintain a **well-structured directory hierarchy**, such as:
  + text\_preprocessing
  + plagiarism\_detection
  + ui\_components
  + utils
* Name files based on their **functionality**, e.g., similarityChecker.dart, textCleaner.dart.

### 7. Version Control

* Use **Git** for version control.
* Write **clear and concise commit messages**, such as Added Urdu stemmer, Fixed tokenizer for diacritics, etc.
* Regularly **push changes** to a remote repository like GitHub or GitLab.

### 8. Performance

* Optimize code by:
  + Avoiding **unnecessary computations**.
  + Caching frequently used results, such as tokenized text.
  + Using **efficient data structures**, especially when handling large Urdu datasets.

### 9. Security

* Ensure the app handles **user data securely**, especially uploaded documents.
* Validate all user input to prevent **injection attacks** or application crashes.
* Avoid storing sensitive data in plaintext or unsecured local storage.

### 10. Review and Testing

* Conduct **regular code reviews** to maintain code quality.
* Perform **unit testing**, especially for:
  + Plagiarism detection algorithm.
  + Text preprocessing steps (normalization, stop-word removal, stemming).
* Test edge cases, such as empty documents or text with heavy punctuation or diacritics.

### Conclusion

By adhering to these coding standards, the development of the **Urdu Plagiarism Detection System** will remain consistent, robust, and scalable. These practices will help ensure the final application is secure, performant, and easy to maintain.

## User Manual

### ****1. Introduction****

The Plagiarism Detection System for Urdu Language is a mobile application designed to help students, teachers, and researchers verify the originality of Urdu text. It analyzes and detects similarity in documents using natural language processing techniques specifically tailored for the Urdu language.

### ****2. System Requirements****

#### ****Operating System:****

* Android 8.0 (Oreo) or higher

#### ****Hardware:****

* Minimum RAM: 2 GB
* Minimum Storage: 100 MB
* Internet Connection: Required for uploading and comparing documents

### ****3. Installation Guide****

1. Download the APK file from the official website or provided link.
2. Go to **Settings > Security > Enable Unknown Sources**.
3. Tap the downloaded APK file and follow on-screen instructions to install the app.

### ****4. Getting Started****

#### ****Step 1: Launch the App****

* Tap the app icon to open.

#### ****Step 2: Sign up or Login****

* Login or signup an account.

#### ****Step 3: Home Screen Overview****

* **Upload Document**: Select or capture a document to check for plagiarism.
* **Check Similarity**: View or enter text to compare.
* **History**: Review previously checked documents.
* **Settings**: Configure language, theme, and account preferences.

### ****5. How to Use****

1. Tap **Upload Document**.
2. Choose a file from your device.
3. Press **Start Scan**.
4. Wait while the system processes and analyzes the content.

### ****6. Viewing Results****

Once scanning is complete, the app displays:

* **Plagiarism Percentage** (e.g., 35% plagiarized)
* **Matching Sources** (if available)
* **Highlighted Text** showing copied/similar portions

You can also:

* **Download Report** (PDF)
* **Share Results** via email or other platforms

### ****7. Features****

* Urdu Text Tokenization and Normalization
* Advanced Similarity Detection using NLP
* Real-Time Scanning
* Highlighted Matching Phrases
* History of Previous Reports
* Dark Mode Support
* Multi-format Support: PDF, DOC, TXT

### ****8. Settings****

In the **Settings** tab, you can:

* Change **App Language** (Urdu / English)
* Enable or disable **Dark Mode**
* Manage **Privacy & Permissions**
* View **App Version**, **Credits**, and **Contact Support**

### ****9. FAQs****

**Q1: What languages are supported?**

Currently, only **Urdu** is supported. Future versions may include English and regional languages.

**Q2: Is my data safe?**

Yes. Uploaded documents are processed securely and not stored on any external servers permanently.

**Q3: What is the maximum file size?**

The app supports files up to **10MB** in size.

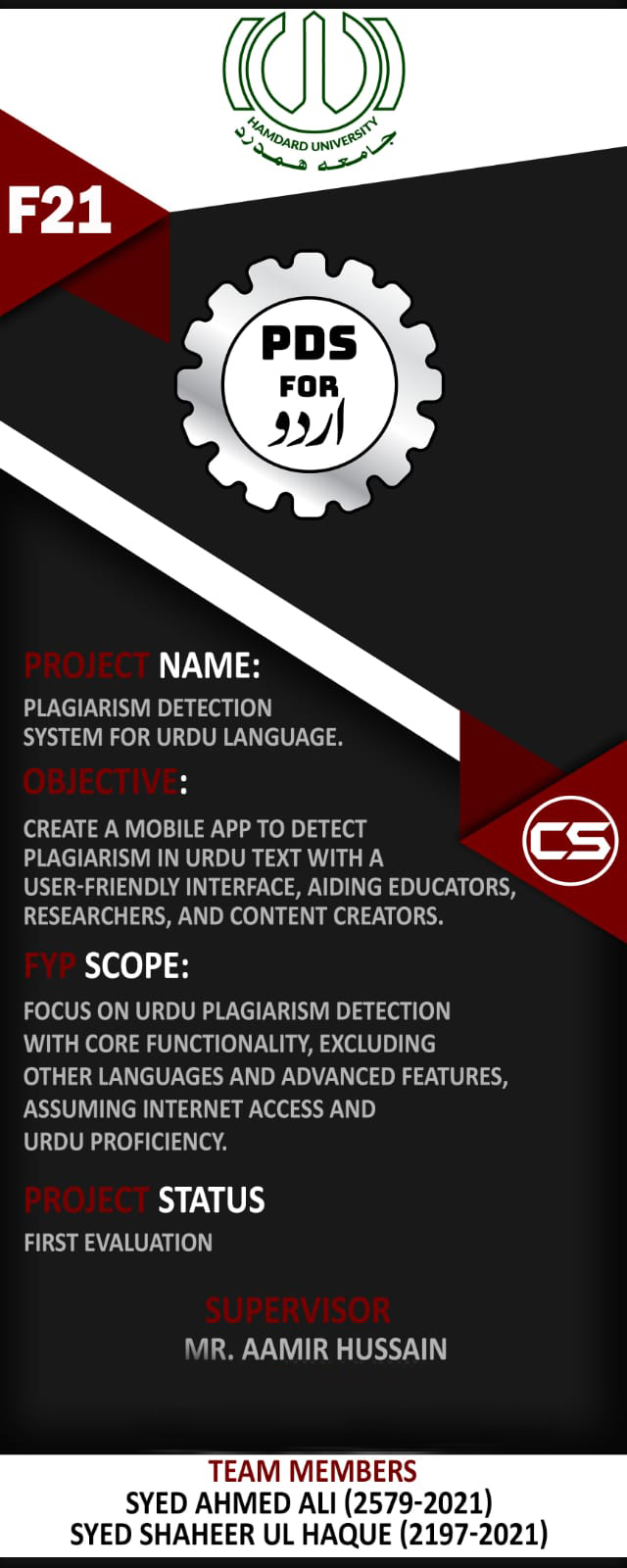
### ****10. Troubleshooting****

| **Issue** | **Solution** |
| --- | --- |
| App not opening | Restart your phone and try again |
| File not uploading | Check permissions or try a smaller file size |
| Inaccurate results | Make sure the input text is in standard Urdu |
| App crashes | Clear app cache or reinstall the app |

### ****11. Conclusion****

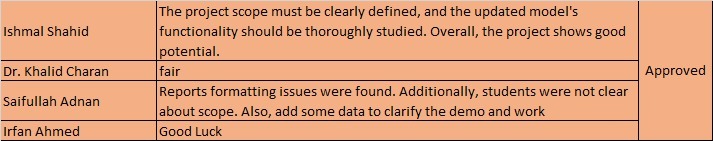
This mobile app simplifies the process of detecting plagiarism in Urdu-language content, supporting students, teachers, and professionals in ensuring content originality. Regular updates will continue to enhance its accuracy and usability.

# A5. FLYER & POSTER DESIGN

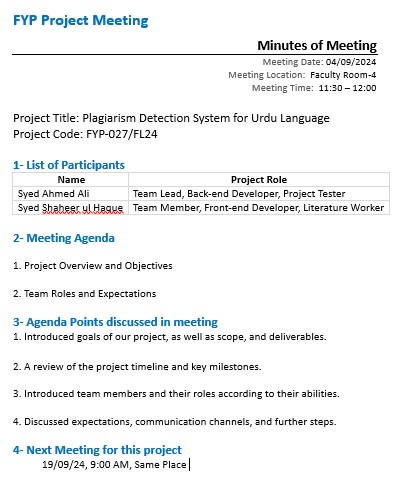


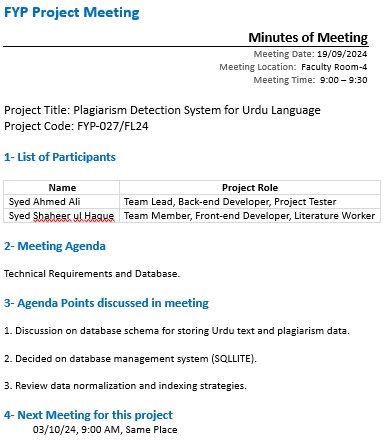
# A6. COPY OF EVALUATION COMMENTS

## COPY OF EVALUATION COMMENTS BY JURY FOR PROJECT – I END SEMESTER EVALUATION

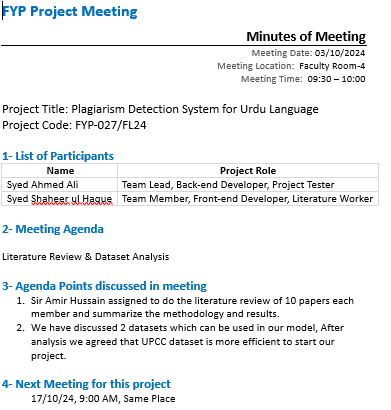


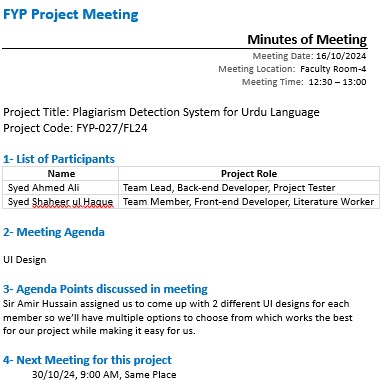
# A7. MEETINGS’ MINUTES & Sign-Off Sheet

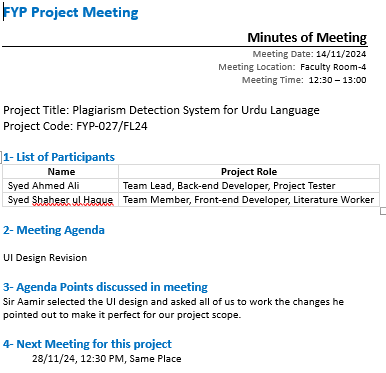
**Meeting 1:**

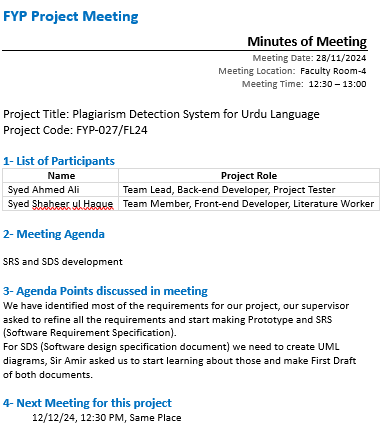


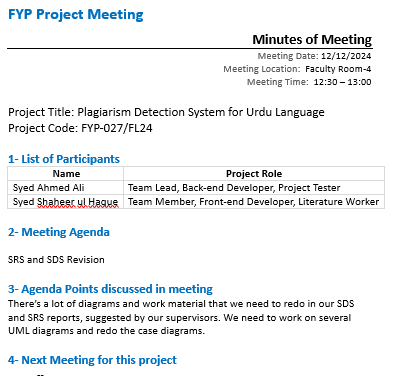
**Meeting 2:**

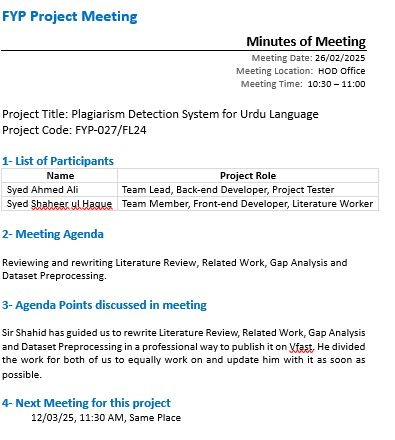
**Meeting 3:**

**Meeting 4:**

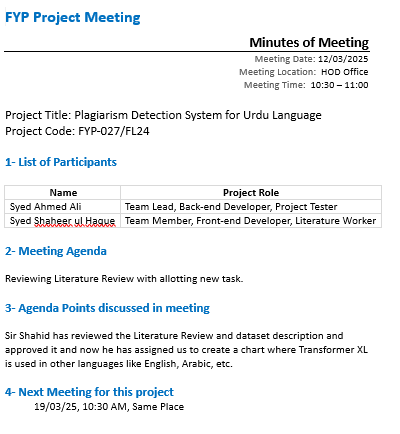
**Meeting 5:**

**Meeting 6:**

**Meeting 7:**



**Meeting 8:**

**Meeting 9:**

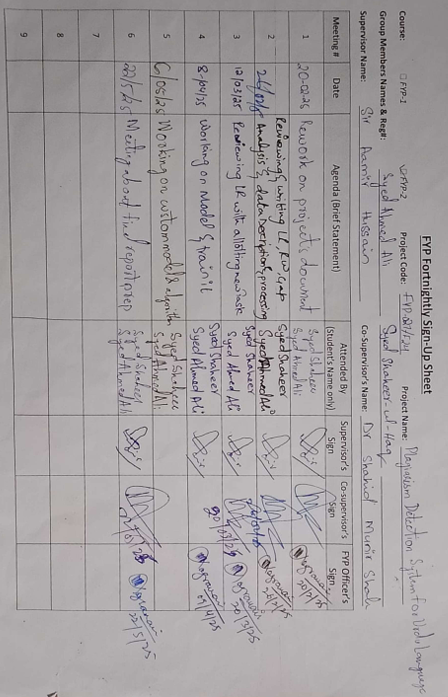


**Meeting 10:**

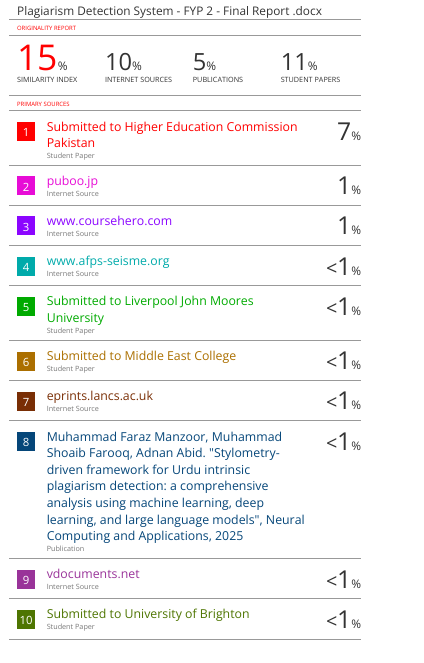
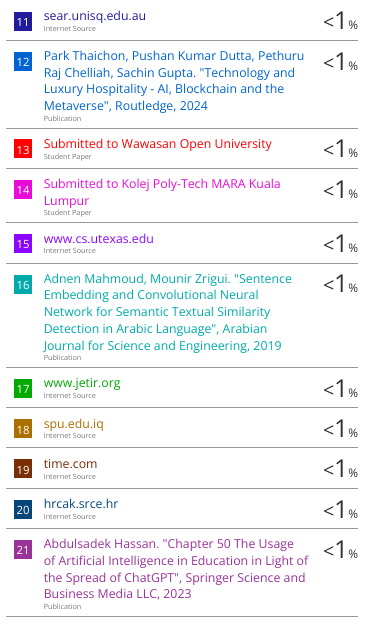
# A8. DOCUMENT CHANGE RECORD

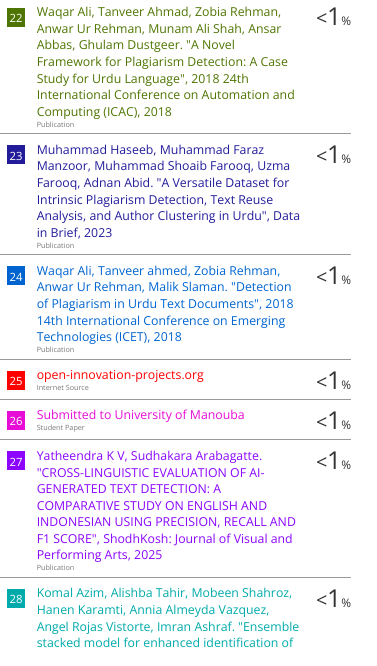
|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Author** | **Change Details** |
| 6/7/25 | 1 | Syed Shaheer ul Haque | Added 4 chapters after fyp 1. |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

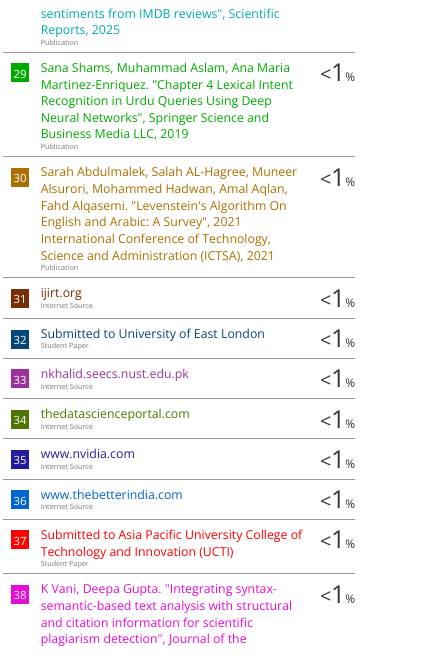
# A9. PROJECT PROGRESS

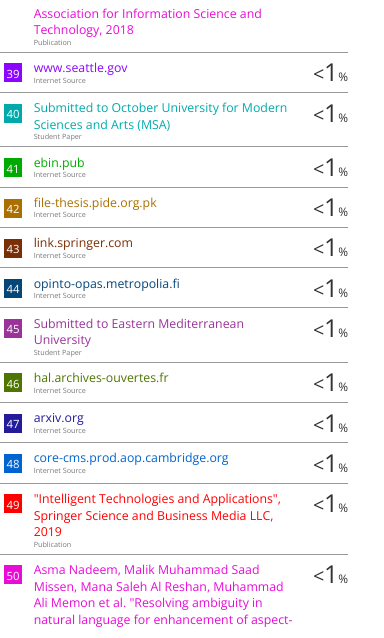


# A10. Plagiarism Test Summary Report

**** ****

****

****

****

****

****