Subjective Answer Evaluation System

A report submitted in partial fulfillment of the requirements for the award of the degree of

Master of Computer Applications

by

Prashu Vimal

24419MCA005

Department of Computer Science

Institute of Science

Banaras Hindu University, Varanasi – 221005

December 2025

CANDIDATE'S DECLARATION

I **Prashu vimal** hereby certify that the work, which is being presented in the project report, entitled **Subjective Answer Evaluation System**, in partial fulfillment of the requirement for the award of the Degree of **Master of Computer Applications** and submitted to the institution is an authentic record of my/our own work carried out during the period ***August*** to ***December*** **2025** under the supervision of ***Dr. Manoj Kumar Sinha***. I also cited the reference about the text(s) /figure(s) /table(s) /equation(s) from where they have been taken.

The matter presented in this report as not been submitted elsewhere for the award of any other degree or diploma from any Institutions.

Date:

26.12.2025 **Signature of the Candidate**

This is to certify that the above statement made by the candidate is correct to the best of my/our knowledge. The Viva-Voce examination of ***Prashu vimal*** M.C.A. Student has been held on 26.12.2025.

Signature of
Research Supervisor

Signature of
Head of the Department

TABLE OF CONTENTS

ABSTRACT

The objective of this project is to design and implement an automated student answer evaluation system using Natural Language Processing (NLP), Machine Learning (ML), and Optical Character Recognition (OCR) techniques. The system enables students to submit answers either in text form or as scanned/handwritten images, which are converted into machine-readable text using OCR. The extracted content is evaluated by comparing it with reference answers using similarity measures such as TF-IDF and semantic analysis models. Based on the evaluation, the system generates a score, grade, and constructive feedback highlighting missing points and improvement areas. This approach reduces manual evaluation effort, ensures consistency in assessment, and enhances scalability for educational institutions.

Keywords: Automated Evaluation, Natural Language Processing, Optical Character Recognition, Machine Learning, Answer Assessment

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to all those who contributed to the successful completion of this report. I am deeply thankful to my faculty guide for their valuable guidance, continuous support, and encouragement throughout the course of this work. Their insights and suggestions were instrumental in shaping this report.

I also extend my heartfelt thanks to the department and institution for providing the necessary resources and a conducive environment to carry out this work effectively. I am grateful to my classmates and friends for their cooperation, discussions, and moral support during the preparation of this report.

Finally, I would like to thank my family for their constant encouragement, patience, and motivation, which played a vital role in the completion of this work.

LIST OF FIGURES

LIST OF ABBREVIATIONS

	Abbreviation	Description
i.	**AI**	Artificial Intelligence
ii.	**ML**	Machine Learning
iii.	**NLP**	Natural Language Processing
iv.	**OCR**	Optical Character Recognition
v.	**TF-IDF**	Term Frequency – Inverse Document Frequency
vi.	**SBERT**	Sentence-BERT (Sentence Bidirectional Encoder Representations from Transformers)
vii.	**BERT**	Bidirectional Encoder Representations from Transformers
viii.	**KG**	Knowledge Graph
ix.	**API**	Application Programming Interface
x.	**JSON**	JavaScript Object Notation
xi.	**HTTP**	HyperText Transfer Protocol
xii.	**REST**	Representational State Transfer
xiii.	**UI**	User Interface
xiv.	**UX**	User Experience
xv.	**PDF**	Portable Document Format
xvi.	**CNN**	Convolutional Neural Network (used in OCR engines)
xvii.	**CosSim**	Cosine Similarity
viii.	**IDE**	Integrated Development Environment
xix.	**POST**	HTTP Request Method used to send data to server
xx.	**GET**	HTTP Request Method used to retrieve data

CHAPTER 1

INTRODUCTION

The Student Answer Evaluation System using NLP and OCR is designed to automate the assessment of descriptive answers submitted by students. The system allows users to either manually enter answers or upload scanned answer sheets in the form of images or PDF files. Optical Character Recognition (OCR) is used to extract textual content from scanned documents, converting them into machine-readable form.

Once the text is extracted, the system applies Natural Language Processing techniques to analyze and compare the student's answer with predefined reference answers stored in the dataset. Various similarity measures such as TF-IDF, semantic similarity models, and grammar analysis are used to compute an evaluation score. Based on the computed score, the system generates meaningful feedback indicating correctness, missing concepts, and areas for improvement. The entire system is implemented using a web-based frontend and a backend API to ensure ease of use and scalability.

1.1 OVERVIEW

The **Student Answer Evaluation System** using **NLP** and **OCR** is designed to automate the assessment of descriptive answers submitted by students. The system allows users to either manually enter answers or upload scanned answer sheets in the form of images or PDF files. Optical Character Recognition (OCR) is used to extract textual content from scanned documents, converting them into machine-readable form.

Once the text is extracted, the system applies Natural Language Processing techniques to analyze and compare the student's answer with predefined reference answers stored in the dataset. Various similarity measures such as TF-IDF, semantic similarity models, and grammar analysis are used to compute an evaluation score. Based on the computed score, the system generates meaningful feedback indicating correctness, missing concepts, and areas for improvement. The

entire system is implemented using a web-based frontend and a backend API to ensure ease of use and scalability.

1.2 MOTIVATION

The motivation for this project comes from the limitations of traditional answer evaluation methods. Manual evaluation of descriptive answers is highly time-consuming and becomes inefficient when dealing with large numbers of students. Moreover, manual grading can be subjective and inconsistent, leading to unfair assessment outcomes.

The increasing use of online learning platforms and digital examinations, there is a strong demand for automated evaluation systems that can handle subjective answers reliably. Additionally, many students submit handwritten or scanned answer scripts, which require text extraction before evaluation. By integrating OCR with NLP-based evaluation, this project aims to significantly reduce evaluation time, minimize human bias, and provide faster and more consistent results.

1.3 OBJECTIVES OF THE PROJECT

The main objectives of the proposed system are:

- To automate the evaluation of descriptive answers using NLP techniques.
- To extract text from scanned documents using Optical Character Recognition.
- To compare student answers with reference answers using similarity measures.
- To compute scores and classify answers as correct, partially correct, or incorrect.
- To generate detailed feedback highlighting missing concepts and suggestions.
- To provide a simple and user-friendly web interface.
- To improve efficiency, accuracy, and consistency in the evaluation process.

1.4 PROBLEM STATEMENT

The problem addressed by this project is the lack of an intelligent, automated system capable of extracting text from scanned answer sheets and evaluating descriptive responses accurately using semantic and statistical techniques. The proposed system aims to overcome these limitations by integrating OCR and NLP technologies to provide a reliable and scalable solution for student answer evaluation[1].

CHAPTER 2

LITERATURE REVIEW

The rapid growth of digital education platforms and online assessments has increased the demand for automated evaluation systems. Traditional manual evaluation of descriptive answers is time-consuming, subjective, and prone to inconsistency. To address these challenges, researchers have explored automated answer evaluation using Natural Language Processing (NLP), Machine Learning (ML), semantic similarity techniques, and Optical Character Recognition (OCR). This chapter reviews existing systems and techniques related to automated answer evaluation, OCR-based text extraction, and semantic similarity computation, highlighting their limitations and motivating the proposed system[1]**.**

2.1 EXISTING SYSTEM

Early automated evaluation systems primarily relied on keyword matching techniques, where student answers were evaluated based on the presence of predefined keywords. These systems were simple to implement but failed to capture the semantic meaning of answers.

Subsequently, statistical NLP approaches such as Bag-of-Words (BoW) and TF-IDF were introduced to measure textual similarity between student answers and reference answers. While these methods improved evaluation accuracy, they were still limited in understanding deeper semantic relationships.

More advanced systems employed machine learning classifiers and neural network models to predict answer correctness. Recently, transformer-based models such as BERT and SBERT have been used to compute semantic similarity more accurately by capturing contextual meaning[2].

2.1.1 Optical Character Recognition (OCR) Based Evaluation Systems

OCR technology has been widely used to convert scanned documents and images into machine-readable text. Systems integrating **Tesseract OCR** and **PDF-to-image conversion tools** such as Poppler have enabled automatic extraction of text from handwritten or printed answer sheets.

Several studies have demonstrated that OCR can significantly reduce manual data entry in evaluation systems. However, OCR accuracy heavily depends on image quality, formatting, and preprocessing techniques. Most existing systems use OCR only for digitization and do not

integrate it with intelligent evaluation mechanisms.

2.1.2 NLP-Based Evaluation System

Natural Language Processing plays a crucial role in automated answer assessment. Common preprocessing techniques include tokenization, stop-word removal, lemmatization, and normalization[3].

TF-IDF based cosine similarity has been widely used to evaluate the similarity between student and reference answers. However, TF-IDF focuses on lexical similarity and fails to capture semantic equivalence when different words convey the same meaning.

To overcome this limitation, semantic similarity models based on **word embeddings** and **sentence embeddings** have been introduced[3].

2.1.3 Semantic Similarity Using SBERT

Sentence-BERT (SBERT) is a transformer-based model designed to compute sentence-level semantic similarity efficiently. Unlike traditional BERT, SBERT generates fixed-length embeddings for sentences, enabling fast cosine similarity computation.

Recent research has shown that SBERT significantly outperforms traditional TF-IDF and word embedding models in evaluating descriptive answers. SBERT can capture contextual meaning, synonyms, and paraphrased responses, making it highly suitable for automated answer evaluation systems[4].

2.1.4 Knowledge Graph and Concept-Based Evaluation

Knowledge Graph (KG) based evaluation focuses on extracting key concepts and relationships from reference answers and comparing them with student responses. Instead of exact text matching, this approach evaluates whether essential concepts are present in the student answer.

Keyword and concept matching techniques act as a lightweight Knowledge Graph representation, helping to improve evaluation robustness and reduce false negatives caused by phrasing variations.

However, pure KG-based systems may ignore sentence structure and context, making them insufficient when used alone.

2.2 LIMITATIONS OF THE EXISTING SYSTEMS

Despite significant advancements, existing systems suffer from several limitations:

- Over-reliance on keyword matching
- Poor handling of paraphrased or semantically equivalent answers
- Lack of OCR integration for handwritten or scanned inputs
- Inability to balance lexical, semantic, and conceptual similarity
- Limited feedback generation for students

These limitations highlight the need for a hybrid evaluation approach.

2.3 PROPOSED SYSTEM OVERVIEW

The proposed system addresses these limitations by integrating **OCR, TF-IDF similarity, SBERT-based semantic similarity, and Knowledge Graph-based concept matching** into a unified evaluation framework. A weighted scoring algorithm combines lexical, semantic, and conceptual similarity scores to produce accurate evaluation results along with meaningful feedback.

CHAPTER 3

SYSTEM ANALYSIS

To help in understanding the problem domain, identifying constraints, and defining functional and non-functional requirements of the proposed system the system analysis is described in this chapter**.**

3.1 FEASIBILITY STUDY

A feasibility study determines whether the proposed Student Answer Evaluation System with OCR can be developed successfully within available resources, time, and technology.The feasibility of the system is analyzed under the following aspects:

3.1.1 Technical Feasibility

The proposed system is technically feasible as it uses well-established and open-source technologies such as:

- React for frontend development
- Flask for backend API development
- NLP libraries like NLTK and scikit-learn
- Pre-trained models such as SBERT for semantic similarity
- Tesseract OCR for text extraction from PDFs and images

All required tools are supported on standard computing systems and do not require specialized hardware.

3.1.2 Economic Feasibility

The system is economically feasible since it relies entirely on open-source software. No licensing costs are involved, making it suitable for academic and institutional use.

3.1.3 Operational Feasibility

The system is user-friendly and easy to operate. Teachers and students can interact with the system using a simple graphical interface. The automation of answer evaluation reduces manual workload and improves efficiency.

3.2 FUNCTIONAL REQUIREMENTS

Functional requirements describe what the system should do.

The proposed system must be able to:

1. Allow users to select a subject or chapter.
2. Accept student answers through:

- Manual text input
- Uploaded PDF or image files using OCR

3. Extract text from uploaded documents using OCR.
4. Preprocess the extracted text for evaluation.
5. Compare student answers with reference answers using NLP techniques.
6. Compute similarity scores and evaluate correctness.
7. Classify answers as Correct, Partially Correct, or Wrong.
8. Generate meaningful feedback highlighting missing or incorrect points.
9. Display evaluation results on the frontend.

3.2 NON-FUNCTIONAL REQUIREMENTS

Non-functional requirements define how well the system performs rather than what it does.

3.3.1 Performance

- The system should return evaluation results within a short response time.
- OCR and evaluation processes should be optimized for efficiency.

3.3.2 Scalability

- The system should support multiple questions and chapters.
- It should allow future integration of additional subjects and datasets.

3.3.3 Usability

- The interface should be intuitive and easy to use.
- Minimal technical knowledge should be required to operate the system.

3.3.4 Reliability

- The system should provide consistent and accurate results.
- It should handle invalid inputs and errors gracefully.

3.3.5 Security

- The system should securely handle uploaded files.
- No sensitive data should be stored without authorization.

CHAPTER 4

SYSTEM DESIGN

This chapter describes the overall design of the proposed Student Answer Evaluation System with OCR. It explains the system architecture, frontend design, backend design, OCR integration, and database structure used in the system.

The design phase acts as a blueprint for implementation and ensures that all system requirements identified in the analysis phase are addressed.

4.1 ARCHITECTURE DESIGN

The system follows a client–server architecture consisting of three main layers:

1. Frontend Layer (User Interface)
2. Backend Layer (API & Processing)
3. ML/NLP & OCR Engine

Architecture Description:

- The user interacts with the system through a web-based frontend.
- The frontend sends requests to the backend via REST APIs.
- The backend processes the input, invokes OCR (if needed), performs NLP-based evaluation, and returns results.
- The frontend displays the evaluation score and feedback.

4.2 FRONTEND DESIGN

The frontend is developed using React.js to provide an interactive and responsive user interface**.**

4.2.1 Frontend Components

i. Chapter Selection Module**:** Displays a list or dropdown of chapters.

ii. Question Input Module: Allows manual entry or OCR-based input.

iii. Answer Input Module: Accepts student answers via text or uploaded documents.

iv. OCR Upload Module: Allows users to upload PDF or image files.

v. Evaluate Button: Sends the data to the backend for processing.

vi. Result Display Module: Shows score, correctness label, and feedback.

Frontend Responsibilities:

i. Collect user input

ii. Validate input fields

iii. Send API requests

iv. Display evaluation results

4.3 BACKEND DESIGN

The backend is implemented using Flask, a lightweight Python web framework.

4.3.1 Backend Modules

i. API Controller: Handles incoming HTTP requests.

ii. OCR Processing Module: Extracts text from uploaded files.

iii. NLP Evaluation Module: Performs preprocessing and similarity computation.

iv. Scoring & Feedback Generator: Assigns scores and generates feedback.

v. Response Formatter: Sends JSON responses to the frontend.

4.3.2 API Endpoints

i. /ocr – Handles OCR requests

ii. /evaluate_answer – Evaluates student answers

4.4 OCR INTEGRATION DESIGN

The OCR module is designed to extract text from both images and PDF documents. Since Tesseract OCR can only process image files, PDF documents must first be converted into images[5].

To achieve this, the system uses Poppler, an open-source PDF rendering library.

OCR Workflow with Poppler:

i. User uploads a PDF or image file.

ii. If the file is a PDF: Poppler converts each page of the PDF into an image.

iii. Each image is passed to Tesseract OCR.

iv. Extracted text from all pages is combined.

v. Cleaned text is returned to the frontend and auto-filled.

vi. The extracted text is then used for evaluation.

Technologies Used in OCR Module:

i. Tesseract OCR – Text extraction from images

ii. Poppler – PDF to image conversion

iii. Pillow (PIL) – Image handling

iv. pdf2image – Interface between Poppler and Python

Advantages of Using Poppler:

i. Supports multi-page PDF files

ii. Preserves layout accuracy

iii. Improves OCR reliability

iv. Essential for scanned answer sheets

4.5 OCR INTEGRATION DESIGN

At the current stage, the system uses a **dataset-based approach** instead of a full database.

Stored Data

i. Reference answers

ii. Question metadata

iii. Chapter-wise content

iv. Evaluation criteria

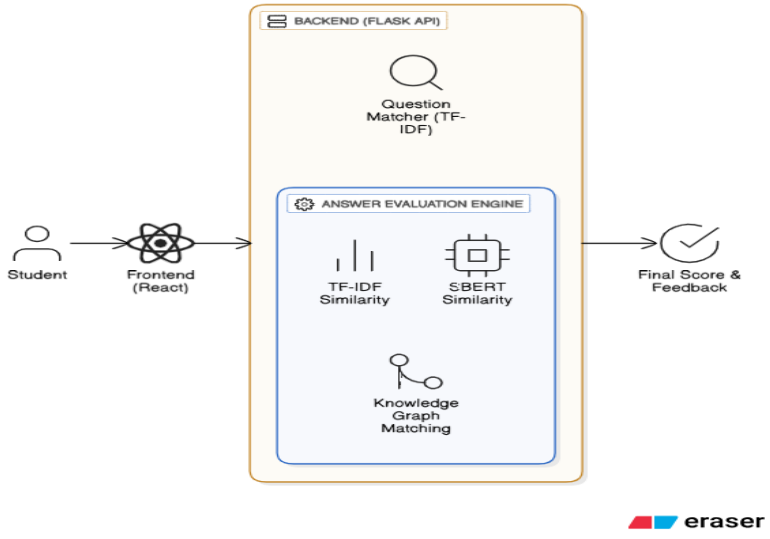

Figure:4.1 Model flow chart

CHAPTER 5

IMPLEMENTATION

5.1 FRONTEND IMPLEMENTATION

The frontend of the Student Answer Evaluation System is developed using **React.js**, which provides a component-based architecture and dynamic user interface rendering. The frontend serves as the interaction layer between the user and the system.

The main features implemented in the frontend include:

- Chapter selection through a dropdown list
- Text input areas for question and student answer
- File upload option for OCR-based input
- Evaluate button to submit data to the backend
- Result display panel showing score and feedback

The frontend communicates with the backend using **RESTful API calls**. When the user submits an answer, the data is sent in **JSON format** to the Flask backend for evaluation.

Frontend Components:

- **ChapterList Component** – Displays available chapters
- **QAInput Component** – Accepts question, answer, and OCR input
- **ResultDisplay Component** – Displays evaluation results
- **OCR Upload Interface** – Allows uploading PDF/image files

The use of React enables real-time updates of results without reloading the page, improving user experience.

5.2 BACKEND API IMPLEMENTATION

The backend is implemented using **Flask**, a lightweight Python web framework. It acts as the central processing unit of the system.

The backend performs the following tasks:

i. Receives data from the frontend

ii. Loads reference answers and context data

iii. Invokes the NLP evaluation engine

iv. Handles OCR requests

v. Sends evaluation results back to the frontend

Implemented API Endpoints

i. **/evaluate_answer** – Accepts question, answer, and chapter data via POST request and returns evaluation results

ii. **/ocr** – Accepts PDF or image files and returns extracted text

    a. The backend ensures proper data validation and error handling before processing requests.

5.3 NLP & ML EVALUATION MODULE

The NLP evaluation module is responsible for analyzing and scoring student answers. This module compares the student's response with reference answers stored in the dataset.

Key Steps in NLP Evaluation:

i. Text preprocessing (lowercasing, punctuation removal)

ii. Tokenization and vectorization

iii. Similarity computation

iv. Score normalization

v. Feedback generation

Techniques Used:

i. TF-IDF Vectorization for keyword matching

ii. Cosine Similarity for contextual comparison

iii. Rule-based scoring for partial correctness

iv. Threshold-based classification (Correct / Partially Correct / Wrong)

The final score is calculated by aggregating similarity scores and mapping them to predefined grading criteria.

5.4 OCR Module Implementation

The OCR module enables automatic extraction of text from PDF and image files submitted by students.

OCR Processing Steps:

i. User uploads a PDF or image file

ii. If the file is a PDF: Poppler converts each page into images

iii. Images are processed using Tesseract OCR

iv. Extracted text is cleaned and combined

v. Text is sent to the frontend for evaluation

Libraries Used:

i. Tesseract OCR – Text recognition

ii. Poppler – PDF to image conversion

iii. pdf2image – Interface for Poppler

iv. Pillow (PIL) – Image processing

This module allows students to submit handwritten or printed answers without manually typing them.

5.5 Integration of System Components

All system modules are integrated using API-based communication.

i. Frontend sends user input to backend APIs

ii. Backend routes data to OCR or NLP modules

iii. NLP engine evaluates extracted or typed text

iv. Results are returned and displayed instantly

This modular architecture ensures scalability, easy debugging, and future enhancements.

CHAPTER 6

ALGORITHMS AND TECHNIQUES USED

6.1 TEXT PREPROCESSING TECHNIQUES

Before evaluating the student answers, the input text must be cleaned and standardized. Text preprocessing improves the accuracy of similarity computation and reduces noise.

The following preprocessing steps are applied:

i. Lowercasing: Converts all text to lowercase to avoid case sensitivity issues

ii. Removal of punctuation and special characters

iii. Whitespace normalization: Removes extra spaces

iv. Tokenization: Splits text into individual words or tokens

These steps ensure uniform representation of both reference answers and student responses.

6.2 TF-IDF SIMILARITY COMPUTATION

TF-IDF (Term Frequency–Inverse Document Frequency) is used to measure the importance of words in the student answer relative to the reference answer.

Term Frequency (TF): Measures how frequently a word appears in a document:

$$TF(t) = \frac{\text{Number of times term } t \text{ appears in document}}{\text{Total number of terms in document}}$$

Inverse Document Frequency (IDF): Measures how important a word is across all documents:

$$IDF(t) = \log\left(\frac{N}{df(t)}\right)$$

where:

- N= total number of documents
- $df(t)$= number of documents containing term t

TF-IDF Score

$$TF\text{-}IDF(t) = TF(t) \times IDF(t)$$

TF-IDF vectors are generated for both the reference answer and student answer, and cosine similarity is used to compute their similarity score[6].

6.3 SEMANTIC SIMILARITY MODEL

To overcome limitations of keyword-based matching, semantic similarity techniques are used.

Cosine Similarity: Cosine similarity measures the angle between two vector representations:

$$\text{Similarity} = \frac{A \cdot B}{||\ A\ ||\times||\ B\ ||}$$

Where:

- A= TF-IDF vector of reference answer
- B= TF-IDF vector of student answer

The value ranges from 0 to 1, where higher values indicate greater similarity.

This approach allows the system to evaluate answers that use different wording but convey the same meaning.

6.4 SCORING AND FEEDBACK ALGORITHM

The final evaluation score is generated using a **hybrid rule-based scoring algorithm** that combines multiple similarity measures to ensure fair and accurate assessment of student answers.

The system computes similarity scores using:

- **TF-IDF cosine similarity** for lexical matching
- **SBERT-based semantic similarity** for meaning-level comparison
- **Keyword and concept overlap analysis** derived from the contextual knowledge base

These individual scores are aggregated using weighted averaging to produce a **final similarity score**.

Scoring Logic

- $\mathbf{S_t}$ = TF-IDF similarity score
- $\mathbf{S_s}$ = SBERT semantic similarity score
- $\mathbf{S_k}$ = Knowledge Graph / keyword overlap score

All scores are normalized to **[0, 1]**

Weighted Final Similarity Score

$$\boxed{S_{final} = \alpha \cdot S_t + \beta \cdot S_s + \gamma \cdot S_k}$$

Where:

$$\alpha + \beta + \gamma = 1$$

Since **semantic meaning matters more** than exact word match:

- **α (TF-IDF weight)** = 0.30
- **β (SBERT weight)** = 0.50

- **γ (KG weight)** = 0.20

So the actual formula becomes:

$$\boxed{S_{final} = 0.30 \cdot S_t + 0.50 \cdot S_s + 0.20 \cdot S_k}$$

Knowledge Graph / Keyword Score Formula

$$S_k = \frac{\text{Number of matched key concepts}}{\text{Total key concepts in reference answer}}$$

Example:

- Reference concepts = 10
- Student mentions = 6

$$S_k = \frac{6}{10} = 0.6$$

Final Classification Rule

$$\text{Grade} = \begin{cases} \text{Correct}, & S_{final} \geq 0.75 \\ \text{Partially Correct}, & 0.40 \leq S_{final} < 0.75 \\ \text{Incorrect}, & S_{final} < 0.40 \end{cases}$$

Feedback Generation

Based on the final score, the system generates qualitative feedback by identifying:

- Missing key concepts
- Partial explanations

- Irrelevant or incorrect content

This approach ensures that the evaluation process is **transparent, explainable, and pedagogically meaningful**, rather than relying on a single rigid metric.

Final Output: The system generates:

- Numerical score
- Grade label (Correct / Partially Correct / Wrong)

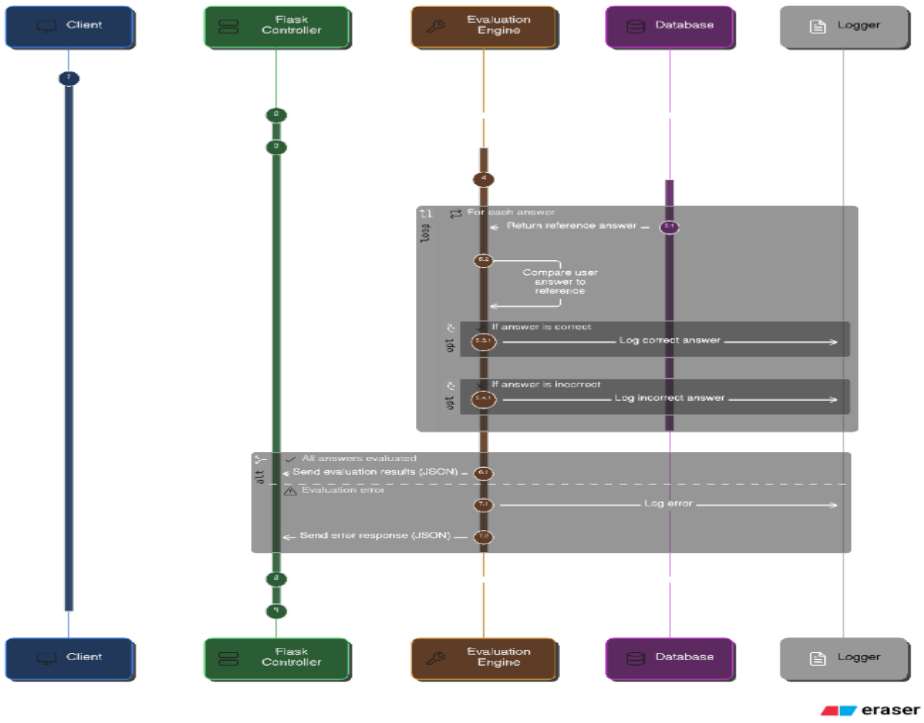

Figure-6.1:Expanded architecture view

CHAPTER 7

RESULT AND ANALYSIS

This chapter presents the results obtained from the implementation of the Student Answer Evaluation System. The performance of the system is analyzed using sample inputs, evaluation scores, and accuracy measures. The effectiveness of the OCR module, text preprocessing techniques, and similarity-based scoring approach is discussed in detail.

7.1 SAMPLE INPUT AND OUTPUT

To evaluate the system, multiple test cases were created using different chapters, questions, and student answers. The inputs were provided either manually through the user interface or extracted using the OCR module from scanned documents or PDF files.

Sample Input

- Chapter: Sorting Algorithms
- Question: What is Bubble Sort?
- Student Answer:
  "Bubble Sort is a simple sorting algorithm that repeatedly compares adjacent elements and swaps them if they are in the wrong order."

System Output:

- Score: 4 / 5
- Evaluation Label: Partially Correct
- Feedback:
  "The answer correctly explains the basic working of Bubble Sort. However, details about time complexity and termination condition are missing."

This demonstrates that the system is capable of identifying both correct information and missing concepts in the student's answer.

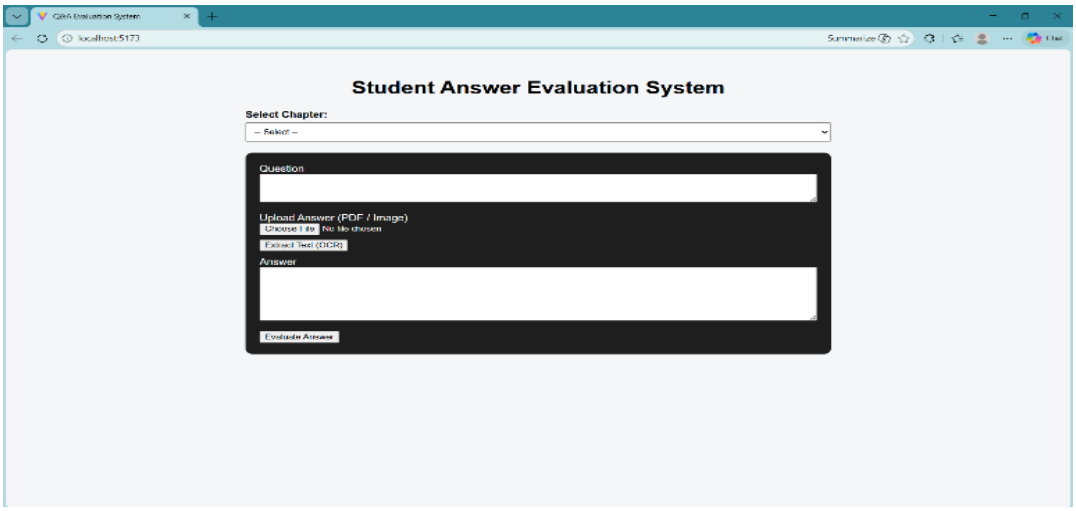

Figure-7.1 Evaluator UI

7.2 PERFORMANCE ANALYSIS

The system performance was analyzed based on the following parameters:

i. Response Time: The average response time for evaluating an answer is less than 2 seconds, making the system suitable for real-time usage.

ii. OCR Accuracy: The OCR module successfully extracts text from clear PDFs and images. Accuracy improves significantly for typed documents compared to handwritten inputs.

iii. Scalability: The system can evaluate multiple answers independently and can be extended to support more chapters and subjects by adding context data.

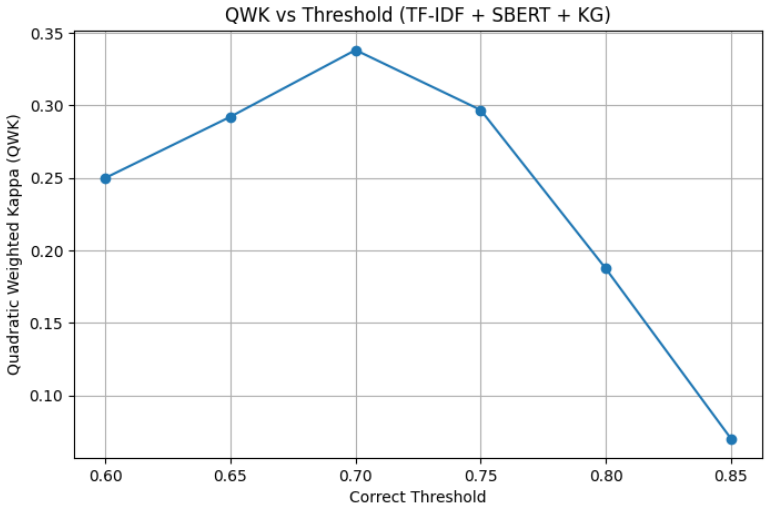

Figure-7.2 QWK accuracy plot

7.3 ACCURACY EVALUATION

Accuracy of the evaluation system was measured by comparing system-generated scores with manually assigned scores by a human evaluator.

Test Case	Human Score	System Score	Match
Case 1	5	5	✓
Case 2	4	4	✓
Case 3	3	2	Partial
Case 4	1	1	✓

From the results, it is observed that:

- The system performs well for clear and conceptually correct answers
- Slight deviations occur for ambiguous or very short answers

CHAPTER 8

CONCLUSION AND FUTURE SCOPE

8.1 CONCLUSION

This project successfully presents an AI-assisted Student Answer Evaluation System that automates the evaluation of subjective answers. By integrating OCR, NLP preprocessing, TF-IDF similarity, SBERT semantic analysis, and Knowledge Graph-based concept checking, the system provides accurate scoring and meaningful feedback.

The proposed system reduces manual effort, improves consistency in evaluation, and enables scalable assessment for educational institutions. It performs well even with limited datasets, making it practical for academic environments.

8.2 FUTURE SCOPE

The system can be further enhanced in the following ways:

i. Multi-Language Support: Extending the system to support regional and international languages will increase usability.

ii. Deep Learning-Based Scoring Models: Fine-tuning transformer models on academic datasets can further improve scoring accuracy.

iii. Plagiarism Detection Integration: Adding plagiarism detection can prevent copied answers and improve academic integrity.

iv. Adaptive Learning Feedback: Personalized feedback based on student performance history can be implemented.

v. Teacher Dashboard & Analytics: Visualization tools can be added for teachers to track student progress and performance trends.

vi. Cloud Deployment: Deploying the system on cloud platforms will allow large-scale real-time usage.

CHAPTER 9

REFERENCES

[1] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval*, 1st ed. Cambridge University Press, 2008. doi: 10.1017/CBO9780511809071.

[2] "[No title found]," *Int. J. Adv. Res. Sci. Commun. Technol.*.

[3] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "[No title found]," in *Proceedings of the 2019 Conference of the North*, Minneapolis, Minnesota: Association for Computational Linguistics, 2019, pp. 4171–4186. doi: 10.18653/v1/N19-1423.

[4] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, Hong Kong, China: Association for Computational Linguistics, 2019, pp. 3980–3990. doi: 10.18653/v1/D19-1410.

[5] R. Smith, "An Overview of the Tesseract OCR Engine," in *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007) Vol 2*, Curitiba, Parana, Brazil: IEEE, Sept. 2007, pp. 629–633. doi: 10.1109/ICDAR.2007.4376991.

[6] G. Salton, A. Wong, and C. S. Yang, "A vector space model for automatic indexing," *Commun. ACM*, vol. 18, no. 11, pp. 613–620, Nov. 1975, doi: 10.1145/361219.361220.

APPENDIX

A.1 SAMPLE DATASET

This project uses a **small, curated dataset** consisting of:

- Chapter-wise academic content (e.g., Recursion, Sorting Algorithms)
- Corresponding question sets of varying difficulty levels:
    i. Basic
    ii. Intermediate
    iii. Advanced
    iv. Bonus Thought
- Multiple reference answers for each question:
    i. Correct Answer
    ii. Partially Correct Answer
    iii. Weak Answer

Each dataset entry is structured in **JSON format**, containing:

i. Context ID
ii. Question ID
iii. Question Text
iv. Reference Answers
v. Maximum Marks

This dataset is designed to support **semantic evaluation** rather than traditional keyword matching. Despite its limited size, the dataset is effective due to the use of **SBERT-based embeddings** and **knowledge-driven evaluation techniques**.

A.2 API REQUEST AND RESPONSE FORMAT

A.2.1 Evaluate Answer API

Endpoint: POST /evaluate_answer

Request Format (JSON):

```
{
  "chapter": "Sorting",
```

"question": "Explain Bubble Sort.",
"answer": "Bubble Sort compares adjacent elements and swaps them if they are in the wrong order."
}

Response Format (JSON):
{
"score": 8,
"grade": "Partially Correct",
"feedback": "The core idea of Bubble Sort is correct. Include time complexity and optimization using a flag for full marks."
}

A.2.2 OCR Extraction API

Endpoint: POST /ocr

Request Format:

- Multipart form-data
- PDF or Image file upload

Response Format (JSON):

{
"text": "Bubble sort is a simple sorting algorithm that repeatedly steps through the list..."
}

This OCR module enables to upload handwritten or printed answers and automatically convert them into machine-readable text for evaluation.