### An Industry-Oriented Mini Project Report

**On**

# “Subjective Answers Evaluation Using Machine Learning and Natural Language Processing”

**Submitted in Partial Fulfillment of the Academic Requirement for the Award of Degree**

BACHELOR OF TECHNOLOGY

in

### Computer Science and Engineering (Artificial Intelligence and Machine learning)

**Submitted By:**

### P. Balu (22R01A66H6)

### D. Akshitha (22R01A66E2)

### N.S.S. Krishna Murthy (22R01A66A5)

**Under the esteemed guidance of**

**Mrs. M. Mounika**



CMR INSTITUTE OF TECHNOLOGY

**(UGCAUTONOMOUS)**

**Approved by AICTE, Affiliated to JNTUH, Accredited by NAAC with A+ Grade,**

**Kandlakoya (V), Medchal Dist - 501 401**

[**www.cmrithyderabad.edu.in**](http://www.cmrithyderabad.edu.in)

# 2024-25

CMR INSTITUTE OF TECHNOLOGY

**(UGCAUTONOMOUS)**

**Approved by AICTE, to JNTUH, accredited by NAAC with A+ Grade,**

**Kandlakoya(V), Medchal Dist-501 401**

[**www.cmrithyderabad.edu.in**](http://www.cmrithyderabad.edu.in)

**.**

**CERTIFICATE**

This is to certify that an Industry oriented Mini Project entitled with “**Subjective Answers Evaluation Using Machine Learning and Natural Language Processing**” is being submitted by:

### P. Balu (22R01A66H6)

### D. Akshitha (22R01A66E2)

### N.S.S. Krishna Murthy (22R01A66A5)

To JNTUH, Hyderabad, in partial fulfillment of the requirement for award of the degree of B. Tech in CSE (AI&ML) and is a record of a Bonafide work carried out under our guidance and supervision. The results in this project have been verified and are found to be satisfactory. The results embodied in this work have not been submitted to have any other University forward of any other degree or diploma.

**Signature of Guide Signature of Project Coordinator Signature of HOD**

**EXTERNAL EXAMINER**

**ACKNOWLEDGEMENT**

We are extremely grateful to Director,Principal and Head of Department, Department of Computer Science and Engineering (Artificial Intelligence and Machine Learning), CMR Institute of Technology for their inspiration and valuable guidance during entire duration.

Department of Computer Science and Engineering (Artificial Intelligence and Machine Learning), CMR Institute of Technology and our external team from the Industry for their constant guidance, encouragement and moral support throughout the project.

We will be failing in duty if we do not acknowledge with grateful thanks to the authors of the references and other literatures referred in this Project.

We express our thanks to all staff members and friends for all the help and coordination extended in bringing out this Project successfully in time.

Finally, we are very much thankful to our parents and relatives who guided directly or indirectly for every step towards success.

### P. Balu (22R01A66H6)

### D. Akshitha (22R01A66E2)

### N.S.S. Krishna Murthy (22R01A66A5)

## ABSTRACT

The Subjective Answer Evaluation System is an innovative educational technology solution that automates the assessment of subjective answers in educational institutions. This project implements a sophisticated multi-algorithm approach combining Natural Language Processing (NLP) and Machine Learning techniques to achieve accurate and reliable answer evaluation. The system utilizes four distinct similarity algorithms: Cosine Similarity, TF-IDF Similarity, NLTK-based Similarity, and SBERT Similarity. This multi-algorithm approach ensures comprehensive evaluation by considering different aspects of text similarity, from basic word matching to semantic understanding. The system calculates a combined similarity score by averaging results from all algorithms, providing a more robust assessment than single-algorithm approaches. A key innovation is the system's ability to handle both text-based and handwritten answers through Optical Character Recognition (OCR) technology, making it versatile for both online and traditional examination settings. Implemented as a web-based application using Django, the system features a user-friendly interface for both students and administrators, with comprehensive user management and automated evaluation capabilities. The system provides detailed scoring and grading (A, B, C, D, F) based on similarity scores, along with performance analytics. Security is ensured through robust authentication mechanisms and access control. Testing results demonstrate the system's effectiveness in evaluating subjective answers with high accuracy, showing promising results compared to manual evaluation. This project represents a significant advancement in educational technology, offering a practical solution to the challenges of subjective answer evaluation. By automating the assessment process, it reduces the workload on educators while providing consistent and objective evaluation of student answers. The system's flexibility and accuracy make it suitable for various educational contexts, from online examinations to traditional classroom settings.

|  |  |
| --- | --- |
| **INDEX** |  |
| **ACKNOWLEDGEMENT** | **i** |
| **ABSTRACT** | **ii** |
| **INDEX** | **iii** |
| **LIST OF FIGURES** | **iv** |
| **LIST OF SCREENSHOTS** | **v** |
| **1.INTRODUCTION** | 1 |
| 1.1ABOUTPROJECT | 1 |
| 1.2EXISTINGSYSTEM | 4 |
| 1.3PROPOSEDSYSTEM | 5 |
| **2.LITERATURE SURVEY** | **6** |
| **3.REQUIREMENTSPECIFICATIONS** | **7** |
| 3.1REQUIREMENTANALYSIS | 7 |
| 3.2SPECIFICATIONPRINCIPLES | 8 |
| **4.SYSTEMDESIGN** | **10** |
| 4.1ARCHITECTURE/BLOCKDIAGRAM | 10 |
| 4.2UMLDIAGRAMS | 11 |
| **5.IMPLEMENTATION** | **18** |
| 5.1PROJECTMODULES | 18 |
| 5.2ALGORITHMS | 21 |
| 5.3SAMPLECODE | 23 |
| **6.TESTING** | **27** |
| 6.1TESTINGMETHODS | 27 |
| **7**. **RESULTS** | **33** |
| **8.CONCLUSION** | **43** |
| **9.REFERENCES** | **44** |

## LIST OF FIGURES

|  |  |  |
| --- | --- | --- |
| **Figure**  **No.** | **Particulars** | **Page No.** |
| 3.1 | Architecture Diagram | 10 |
| 3.2.1.1 | Sequence Diagram (Admin) | 11 |
| 3.2.1.2 | Sequence Diagram (User) | 12 |
| 3.2.2.1 | Activity Diagram (Admin) | 12 |
| 3.2.2.2 | Activity Diagram (User) | 14 |
| 3.2.3.1 | Use Case Diagram (Admin ) | 15 |
| 3.2.3.2 | Use Case Diagram (User ) | 16 |
| 3.2.4 | Class Diagram | 16 |
| 3.2.5 | Deployment Diagram | 17 |
| 5.1 | Testing Flowchart | 27 |

**LIST OF SCREENSHOTS**

|  |  |  |
| --- | --- | --- |
| **Screenshot No.** | **Particulars** | **Page No.** |
| 6.1 | Main page | 33 |
| 6.2 | Registration Page | 34 |
| 6.3 | Login Page | 34 |
| 6.4 | Contact Page | 35 |
| 6.5 | User Dashboard | 35 |
| 6.6 | Subjects List | 36 |
| 6.7 | Test Page | 36 |
| 6.8 | Results Page | 37 |
| 6..9 | User Profile | 38 |
| 6.10 | Admin Dashboard | 39 |
| 6.11 | User Management Page | 39 |
| 6.12 | Subject Management Page | 40 |

# INTRODUCTION

* 1. **About Project**

In modern educational environments, evaluating students' academic performance accurately and efficiently remains a critical challenge, especially when it comes to subjective or descriptive answers. Unlike objective questions, which have clear right or wrong answers, subjective responses require deeper understanding, contextual interpretation, and semantic analysis. Traditionally, this task has been handled manually by educators, often leading to inconsistencies, human bias, and time-consuming evaluation processes. As the demand for scalable and fair assessment grows—particularly in the era of online education and digital examinations—there is a pressing need for automated systems that can analyze and score subjective answers with human-like precision. Leveraging advances in Natural Language Processing (NLP), Machine Learning (ML), and web technologies, automated subjective answer evaluation systems aim to provide real-time, consistent, and accurate grading solutions that reduce educators' workload while enhancing the overall learning experience.

The introduction of such a system has the potential to transform the way educational assessments are conducted. By automating the evaluation of subjective answers, it not only saves valuable time for educators but also ensures that every student is graded with the same level of fairness and accuracy. Instant feedback helps learners understand their mistakes and improve continuously, turning assessments into meaningful learning opportunities. In large-scale examinations or online learning platforms, where evaluating thousands of answers manually is nearly impossible, this system can make a real difference. It bridges the gap between technology and education, proving that intelligent automation can enhance quality without losing the human touch. With the right implementation, this kind of solution could become a standard tool in schools, colleges, and universities, ultimately shaping a smarter, more efficient future for education.

The Subjective Answer Evaluation System is a robust, web-based platform aimed at transforming the way educational institutions handle the evaluation of descriptive or subjective answers. In traditional academic settings, the evaluation of subjective answers often requires significant time and effort from teachers or examiners. This manual process not only increases workload but also opens up the possibility of human errors, inconsistencies, and biased judgment. With the rising demand for remote education and digital learning environments, especially in the post-pandemic era, there is an increasing necessity for automated systems that can evaluate subjective content as accurately and fairly as human evaluators. This project is a direct response to that need. By integrating advanced technologies such as Natural Language Processing (NLP), Machine Learning (ML), and Optical Character Recognition (OCR), this system provides a sophisticated, intelligent solution that automates the evaluation process while maintaining high accuracy and fairness. Developed using the Django framework, it offers a secure, scalable, and user-friendly platform that enhances both the evaluation experience and the overall academic workflow.

The system's architecture is centred around the use of multiple evaluation algorithms that work together to assess student answers by comparing them with carefully curated model answers. These algorithms include Cosine Similarity, which converts text into vector forms and measures the cosine of the angle between them to determine similarity. This method is useful for identifying the closeness of word patterns in responses. The TF-IDF (Term Frequency-Inverse Document Frequency) algorithm plays a crucial role in highlighting the significance of words within the context of a document. By giving weight to frequently used terms that are unique to specific answers, it helps the system identify which parts of the text carry more informational value. Furthermore, NLTK (Natural Language Toolkit)-based techniques are employed to tokenize, normalize, and semantically process the answers. This allows the system to interpret the structure and meaning of sentences, enabling it to understand how the answer is constructed and how well it aligns with the key ideas of the model answer.

Perhaps the most advanced technique integrated into the system is the SBERT (Sentence-BERT) model, a deep learning-based algorithm that captures the semantic meaning of entire sentences or paragraphs. This model allows the system to evaluate answers based on context and intent rather than just word matching, making it capable of handling paraphrased answers, varying sentence structures, and different writing styles. The multi-algorithmic approach ensures a comprehensive and balanced evaluation process that mirrors human understanding while eliminating subjective bias.

In addition to evaluating typed answers, the system is designed to process handwritten responses as well. This is achieved through OCR (Optical Character Recognition) technology, which converts handwritten text into digital form. The converted text is then subjected to the same evaluation pipeline as typed responses. This feature makes the system highly versatile and practical in traditional exam environments where students may submit scanned answer sheets. The ability to support multiple answer formats—typed, handwritten, and uploaded documents—makes the system adaptable to a variety of educational scenarios. Real-time evaluation is another key feature. Once a student submits an answer, the system processes it immediately and provides instant feedback, including scores and suggestions. This not only reduces the turnaround time for results but also helps students understand their performance better and improve their writing skills over time. The feedback is detailed and informative, highlighting both strengths and areas needing improvement. Such timely insights are invaluable in formative assessments and continuous evaluation models.

The system also features a powerful user management module. Students can register themselves and securely log in to their profiles, where they can view their evaluations, performance history, and feedback. Administrators, typically teachers or exam coordinators, have access to a dedicated dashboard from which they can manage users, create question banks, assign model answers, and oversee evaluations. Role-based access control ensures that students and administrators have access only to the functionalities relevant to their roles. The admin interface is clean and intuitive, allowing even non-technical users to operate it without difficulty. User authentication is handled securely, with encrypted credentials and protected sessions to prevent unauthorized access. The entire system prioritizes data security, incorporating session management, secure database queries, and access controls that protect sensitive information such as student details, answer sheets, and evaluation reports.

Another important aspect of the system is its evaluation and grading module. After analyzing an answer, the system assigns an automatic score based on the similarity with the model answer and the coherence of the response. Scores are then translated into letter grades (A, B, C, D, F) according to predefined thresholds. This grading system can be customized by the administrator depending on the institution's grading policy. The system also generates performance analytics—detailed breakdowns of student performance over multiple tests or subjects. These analytics include visualizations such as graphs and charts, providing educators with clear insights into student progress, common mistakes, and topic-wise strengths and weaknesses. This data-driven approach helps in making informed academic decisions and guiding students more effectively.

The system stores all data in a MySQL database, which is known for its reliability, speed, and scalability. The database design follows normalization rules, ensuring efficient storage and quick retrieval of data. Information such as user credentials, submitted answers, question banks, model answers, evaluation results, and feedback records are stored securely and can be backed up for data integrity. The system's modular design allows for future enhancements such as integration with Learning Management Systems (LMS), support for multiple languages, or even speech-to-text processing for spoken answers. Scalability is a key consideration; whether the system is used by a single institution or a nationwide academic board, it can handle increased loads by deploying across multiple servers or cloud infrastructure.

In terms of usability, the system provides a simple and responsive graphical user interface (GUI). It is designed keeping in mind users from diverse backgrounds, including students, teachers, and administrative staff. Navigation is easy, and each feature is logically categorized. The system ensures compatibility with all major browsers and devices, including mobile phones, allowing users to access the platform from anywhere. The real-time nature of the system adds to the efficiency, ensuring minimal delays and maximum convenience.

In conclusion, the Subjective Answer Evaluation System is a forward-looking educational tool that addresses a long-standing challenge in academia—how to evaluate subjective answers fairly and efficiently. By combining multiple text analysis techniques and machine learning models, it provides an intelligent, automated evaluation process that is comparable to human assessment. Its adaptability to different answer formats, integration of OCR for handwritten content, and provision of immediate feedback make it an indispensable solution for modern educational needs. The system not only lightens the load on educators but also ensures greater consistency and fairness in student evaluations. With growing interest in digital learning and assessment technologies, this project stands out as a practical, innovative, and impactful contribution to the future of education.

* 1. **Existing System**

The current system of evaluating subjective answers in educational institutions is largely dependent on manual assessment, which brings with it a host of challenges and limitations. One of the most significant issues is the time-consuming nature of the process. Examiners are required to spend hours reading and grading each individual answer, which becomes especially overwhelming during examination periods when large batches of papers need to be evaluated. This leads to delays in providing feedback to students, which in turn affects their ability to learn and improve. In addition to being slow, manual grading is also highly inconsistent. Different examiners may assess the same answer differently due to personal interpretation, subjective bias, or even their mood and fatigue levels. There is often a lack of standardized criteria, resulting in varied application of marking schemes and a lack of uniformity in the grading process.

Moreover, manual assessment is prone to human error. From simple miscalculations in totalling marks to recording mistakes and the accidental loss or damage of answer sheets, these errors can significantly impact the accuracy of final results. The process of assigning partial marks is also inconsistently applied, leading to further discrepancies. Another major drawback is the limited quality of feedback provided to students. In most cases, students receive only generic comments—or none at all—offering little insight into their performance or how they can improve. This lack of detailed analysis and delayed feedback weakens the learning impact of assessments and makes it difficult to track individual progress over time.

Security is another area of concern in the traditional system. Physical answer sheets are vulnerable to tampering, misplacement, or even intentional grade manipulation. Without secure digital records or audit trails, it's difficult to ensure the integrity of the grading process or detect potential favoritism. Additionally, the manual system lacks scalability. As student numbers grow, it becomes increasingly difficult to process answers efficiently due to the limited number of available examiners and the absence of real-time evaluation capabilities. The system is also ill-equipped to support online examinations, which are becoming more common in the digital age.

Quality control further suffers under this model. There's no easy way to enforce consistent standards, verify grading accuracy, or detect errors automatically. Cross-verification is limited, and the lack of standardized evaluation metrics makes comparisons across students and institutions difficult. Lastly, the administrative burden is high. Managing result compilation, handling re-evaluation requests, coordinating across departments, and calculating final grades all require significant manual effort, leading to increased workload and the risk of administrative mistakes.

Altogether, these limitations clearly demonstrate the urgent need for an automated, intelligent system that can overcome the inefficiencies, inconsistencies, and vulnerabilities of manual subjective answer evaluation. By addressing these issues, such a system would not only improve the fairness and quality of assessments but also ease the burden on educators and enhance the overall academic experience for students.

# Proposed System

The Subjective Answer Evaluation System is a comprehensive and intelligent solution developed to tackle the challenges associated with traditional manual grading methods. It uses advanced technologies to deliver accurate, efficient, and consistent evaluation of descriptive answers. At the core of the system lies a multi-algorithm evaluation engine that applies a combination of techniques—such as Cosine Similarity, TF-IDF, NLTK-based semantic similarity, and SBERT—to deeply analyse the content and meaning of student responses. This approach ensures a fair comparison with model answers, regardless of how the student has structured their response. The system provides real-time evaluation and instant feedback, allowing students to quickly understand their performance and make improvements. It supports both typed and handwritten answers using advanced Optical Character Recognition (OCR) technology, which extracts text from scanned images, ensuring wide usability across various examination formats.

The user interface is designed to be clean, responsive, and accessible, offering dedicated dashboards for both students and administrators. Clear visualizations make it easy to interpret results, while secure authentication, role-based access, encrypted data transmission, and session management ensure data safety and privacy. The system’s analytics and reporting features provide in-depth insights into student progress, subject-wise performance, and academic trends. This helps educators make informed decisions and tailor their teaching strategies. Powered by a MySQL database, the backend offers optimized performance, secure backups, and scalable architecture that can grow with increasing users and assessments. Automated grading applies consistent evaluation criteria, eliminates human bias, and delivers detailed feedback that promotes personalized learning.

One of the standout advantages of the system is its ability to conduct objective assessments using mathematical similarity techniques, which significantly reduces subjective bias. It also dramatically cuts down evaluation time, helping educators focus more on instruction rather than administrative tasks. The system is scalable and capable of handling large volumes of student responses, making it ideal for modern educational institutions. Furthermore, resource optimization is achieved by reducing the reliance on manual examiners, leading to cost savings and better resource allocation. Enhanced feedback, including visual reports, allows students to identify their strengths and weaknesses clearly and improve more effectively. Overall, the Subjective Answer Evaluation System is a major step forward in educational technology. By automating and improving the evaluation process, it not only enhances the quality and fairness of assessments but also contributes to a more efficient and engaging learning experience for students and teachers alike.

**2.LITERATURE SURVEY**

**TITLE:** ‘‘Measurement of text similarity: A survey,’’ Information

**ABSTRACT:** Text similarity measurement is the basis of natural language processing tasks, which play an important role in information retrieval, automatic question answering, machine translation, dialogue systems, and document matching. This paper systematically combs the research status of similarity measurement, analyzes the advantages and disadvantages of current methods, develops a more comprehensive classification description system of text similarity measurement algorithms, and summarizes the future development direction. With the aim of providing reference for related research and application, the text similarity measurement method is described by two aspects: text distance and text representation. The text distance can be divided into length distance, distribution distance, and semantic distance; text representation is divided into string-based, corpus-based, single-semantic text, multi-semantic text, and graph-structure-based representation. Finally, the development of text similarity is also summarized in the discussion section.

**TITLE:** ‘‘A survey on the techniques, applications, and performance of short text semantic similarity,’’

**ABSTRACT:** Short text similarity plays an important role in natural language processing (NLP). It has been applied in many fields. Due to the lack of sufficient context in the short text, it is difficult to measure the similarity. The use of semantics similarity to calculate textual similarity has attracted the attention of academia and industry and achieved better results. In this survey, we have conducted a comprehensive and systematic analysis of semantic similarity. We first propose three categories of semantic similarity: corpus‐based, knowledge‐based, and deep learning (DL)‐based. We analyze the pros and cons of representative and novel algorithms in each category. Our analysis also includes the applications of these similarity measurement methods in other areas of NLP. We then evaluate state‐of‐the‐art DL methods on four common datasets, which proved that DL‐based can better solve the challenges of the short text similarity, such as sparsity and complexity. Especially, bidirectional encoder representations from transformer model can fully employ scarce information of short texts and semantic information and obtain higher accuracy and F1 value. We finally put forward some future directions.

On the development side, the system adheres to established coding standards, including PEP 8 guidelines, clean coding practices, comprehensive documentation, version control via platforms like Git, and regular code reviews. A structured testing strategy is followed, which includes unit testing, integration testing, system testing, user acceptance testing, and performance testing to ensure the system is bug-free and works as intended. Maintenance is also a key consideration, with regular updates for security patches, feature improvements, bug fixes, and performance optimization. Continuous monitoring helps track system health, log errors, evaluate performance, monitor user activity, and detect security issues proactively.

# 3. REQUIREMENT SPECIFICATIONS

# 3.1 REQUIREMENT ANALYSIS

# Hardware Requirements

|  |  |  |
| --- | --- | --- |
| MINIMUM (Required for Execution) | | MY SYSTEM (Development) |
| System | Pentium IV 2.2 GHz | i3 Processor 5th Gen |
| Hard Disk | 20 Gb | 500 Gb |
| Ram | 1 Gb | 4 Gb |

# Software Requirements

|  |  |
| --- | --- |
| Operating System | Windows 10/11 |
| Development Software | Python 3.10 |
| Programming Language | Python |
| Domain | Machine Learning |
| Integrated Development Environment (IDE) | Visual Studio Code |
| Front End Technologies | HTML5, CSS3, Java Script |
| Back End Technologies or Framework | Django |
| Database Language | SQL |
| Database (RDBMS) | MySQL |
| Database Software | WAMP or XAMPP Server |
| Web Server or Deployment Server | Django Application Development Server |
| Design/Modelling | Rational Rose |

# 3.2 SPECIFICATION PRINCIPLES

The Subjective Answer Evaluation System is designed with a strong foundation of architectural, technical, and user-centred principles to ensure that it is reliable, scalable, and user-friendly. The system architecture follows a modular design approach, where each component functions independently to simplify maintenance, enable easy updates, and enhance scalability. This modularity ensures that each function—such as user management, answer processing, evaluation, and reporting—can be developed, tested, and improved separately without affecting the entire system.

The Model-View-Controller (MVC) pattern is adopted to maintain a clean separation of concerns. The Model manages the database and business logic, the View handles the presentation layer and user interface, while the Controller processes user requests and links the Model with the View. This architecture results in a maintainable and organized codebase that supports long-term development and upgrades.

In terms of database design, normalization techniques up to the Third Normal Form (3NF) are applied to minimize data redundancy and optimize data storage efficiency. The database structure supports various relationships, such as one-to-many and many-to-many, with foreign key constraints that ensure referential integrity and efficient query performance. For security, the system incorporates robust authentication measures, including encrypted passwords, secure login processes, session management, and role-based access control. User data is further protected through encrypted data transmission, secure storage practices, regular data backups, access logging, and strict privacy measures, all of which contribute to a secure operating environment.

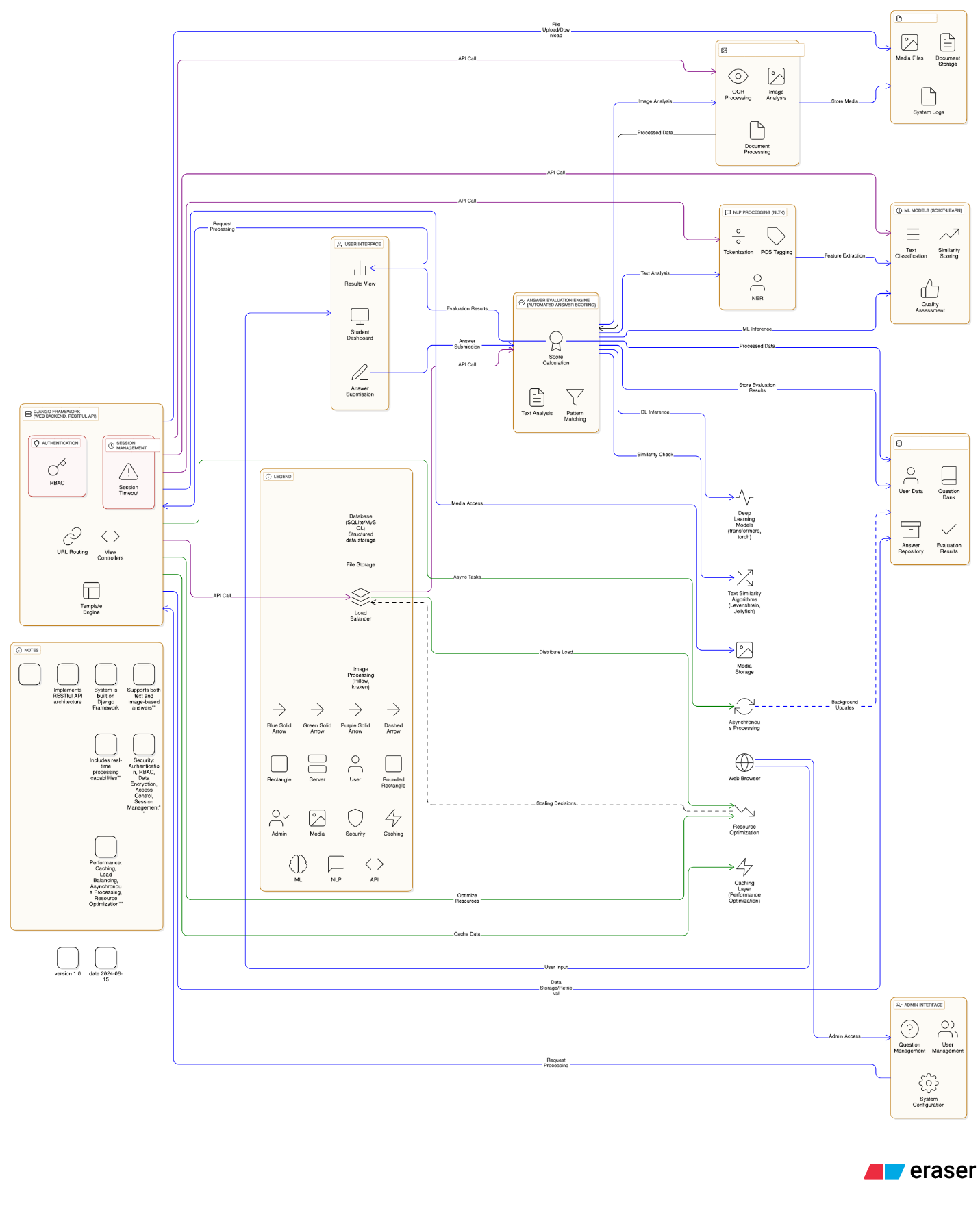
The user interface (UI) of the system is designed to be intuitive and responsive, offering a clean layout with consistent navigation and user-friendly forms. The interface is accessible across devices, with mobile compatibility, cross-browser support, screen reader integration, and keyboard navigation. Proper color contrast ensures visual clarity for all users. When it comes to answer evaluation, the system uses a combination of multiple NLP-based similarity algorithms, such as Cosine Similarity, TF-IDF, NLTK-based comparison, and SBERT, to ensure accurate and fair grading. Weighted scoring techniques and standardized grading rules provide consistent and transparent evaluation, while real-time error handling and result validation ensure reliability and performance optimization.

On the development side, the system adheres to established coding standards, including PEP 8 guidelines, clean coding practices, comprehensive documentation, version control via platforms like Git, and regular code reviews. A structured testing strategy is followed, which includes unit testing, integration testing, system testing, user acceptance testing, and performance testing to ensure the system is bug-free and works as intended. Maintenance is also a key consideration, with regular updates for security patches, feature improvements, bug fixes, and performance optimization. Continuous monitoring helps track system health, log errors, evaluate performance, monitor user activity, and detect security issues proactively.

Finally, the system supports integration with external platforms through well-defined APIs that follow RESTful principles. These APIs use standard protocols and include proper documentation, versioning, and error-handling mechanisms to support future enhancements. Data exchange is carried out in secure and standardized formats, with robust validation and compatibility handling to ensure seamless communication with other systems or services. All these principles together make the Subjective Answer Evaluation System a reliable, secure, scalable, and maintainable platform. It is designed not only to meet current educational needs but also to evolve with technological advancements, making it a future-ready solution that enhances the accuracy, fairness, and efficiency of subjective answer evaluation.

**4.SYSTEM DESIGN**

**4.1 ARCHITECTURE/BLOCKDIAGRAM**

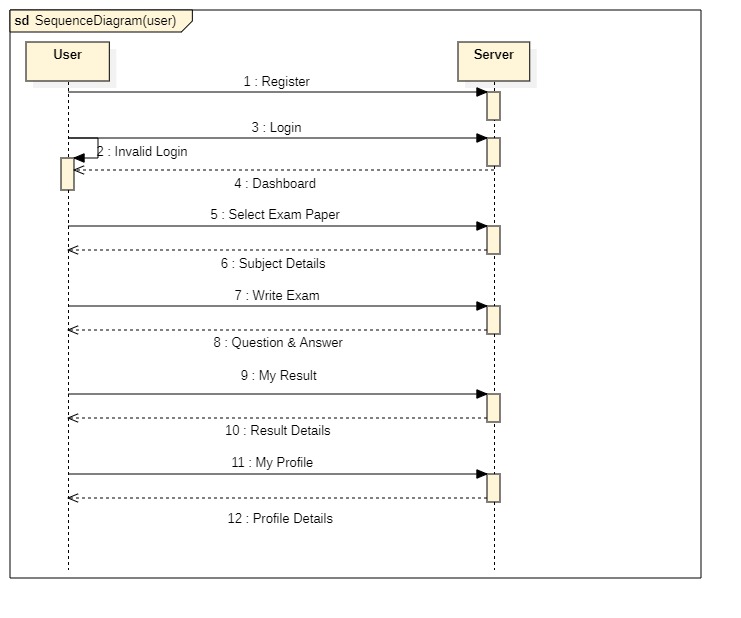


# 4.2 Uml diagrams

# Sequence Diagram (admin)

# 

# Sequence Diagram ( User )



# Activity diagram (admin)

# 

**Activity diagram (user)**

# 

# 

# Use Case Diagram (Admin):

# 

# Use Case Diagram (User):

# 

# Class Diagram

# 

# Deployment Diagram

# 

# 5. IMPLEMENTATION

**5.1 PROJECT MODULES**

The Subjective Answer Evaluation System is structured into several interconnected modules, each serving a specific purpose in the overall system functionality. These modules work together to provide a comprehensive solution for automated answer evaluation.

**5.1.1 User Module**

The User Module serves as the gateway to the system, managing all user-related operations. It handles the complete user lifecycle from registration to profile management. When a new user registers, they provide essential information including their name, email, contact number, student ID, and a profile photo. The system then processes this information, validates the data, and stores it securely in the database. The registration status is set to 'pending' by default, requiring admin approval before the user can access the system.

The authentication system implements secure login mechanisms, ensuring that only authorized users can access the system. It manages user sessions, handles password encryption, and maintains user states. The profile management component allows users to view and update their personal information, change their passwords, and modify their profile photos. This module also tracks user status, showing whether a user is pending, accepted, or blocked.

**5.1.2 Question Management Module**

The Question Management Module is responsible for handling all aspects of questions and subjects in the system. Administrators use this module to create and manage the educational content. The subject management component allows administrators to add new subjects, modify existing ones, and organize them into categories. Each subject can contain multiple questions, and the system maintains proper relationships between subjects and their questions.

The question management component enables administrators to create new questions, set model answers, and establish evaluation criteria. Questions can be modified or deleted as needed, and the system ensures that all changes are properly tracked and validated. This module also handles the categorization of questions, making it easier to organize and retrieve them.

**5.1.3 Answer Evaluation Module**

The Answer Evaluation Module is the core of the system, handling the complex task of evaluating subjective answers. It processes both text-based and handwritten answers. For text-based answers, the module receives the input, validates it, and prepares it for evaluation. For handwritten answers, it employs OCR technology to convert the images into text before processing.

The evaluation process uses multiple algorithms to ensure accurate and fair assessment. The system calculates similarity scores using Cosine Similarity, TF-IDF Similarity, NLTK-based Similarity, and SBERT Similarity. These scores are then combined to produce a final evaluation score. The module handles various edge cases and errors, ensuring robust operation even with complex or unclear answers.

**5.1.4 Result Management Module**

The Result Management Module processes and presents the evaluation results. It calculates individual question scores, computes total scores, and assigns appropriate grades based on predefined criteria. The module generates detailed performance reports, showing not just the final grade but also the specific areas where the student performed well or needs improvement.

The result display component presents the information in a clear and organized manner, allowing students to understand their performance. It shows individual question scores, provides feedback on answers, and displays historical performance data. The module also includes features for result verification and dispute resolution.

**5.1.5 Admin Module**

The Admin Module provides administrators with comprehensive control over the system. It offers tools for user management, allowing administrators to review and process user registrations, manage user statuses, and monitor user activities. The system management component enables administrators to configure system settings, manage the database, and monitor system performance.

This module includes features for backup and recovery, ensuring data safety and system reliability. Administrators can track system usage, monitor performance metrics, and identify potential issues before they affect system operation.

**5.1.6 Analytics Module**

The Analytics Module provides insights into system performance and student achievement. It analyzes data across various dimensions, including subject-wise performance, individual student progress, and overall system usage. The module generates comprehensive reports that help administrators and educators understand trends and patterns in student performance.

The reporting component creates detailed reports that can be exported in various formats. These reports include statistical analysis, performance trends, and success rates. The module also provides visualization tools to help understand the data better.

**5.1.7 Security Module**

The Security Module ensures the integrity and safety of the system. It implements robust authentication mechanisms, manages user sessions, and controls access to different parts of the system. The module handles data encryption, secure storage, and regular backups to protect sensitive information.

The security monitoring component tracks system access, logs security events, and alerts administrators to potential security issues. It implements various security measures to prevent unauthorized access and protect user data.

Each module is designed to work independently while maintaining proper integration with other modules. This modular architecture ensures that the system is maintainable, scalable, and capable of future enhancements. The modules communicate through well-defined interfaces, ensuring smooth operation and data consistency across the system.

**5.2 ALGORITHMS**

**5.2.1 Cosine Similarity Algorithm**

The Cosine Similarity algorithm is a fundamental technique used to measure the similarity between two texts by calculating the cosine of the angle between their vector representations. This method is particularly effective for comparing documents of varying lengths because it focuses on the orientation rather than the magnitude of the vectors. To accomplish this, the texts are first converted into vectors based on word frequencies. Then, the dot product of these vectors is calculated, followed by determining the magnitude of each vector. Finally, the cosine of the angle between the vectors is computed using the formula cos(θ) = (A·B) / (||A|| ||B||), where A and B represent the vectorized forms of the texts. This approach provides a numeric value indicating how similar two pieces of text are, with a value closer to 1 indicating higher similarity.

**5.2.2 TF-IDF (Term Frequency-Inverse Document Frequency) Similarity**

TF-IDF is an advanced algorithm that evaluates the importance of each word in a document relative to a larger collection or corpus of documents. It works by first calculating the term frequency (TF), which is the number of times a word appears in a document, and then computing the inverse document frequency (IDF), which reflects how common or rare a word is across all documents. By multiplying TF and IDF, TF-IDF assigns weights to words, highlighting those that are both frequent in the specific document but rare across others. This weighted representation is then transformed into vectors and compared using cosine similarity. The formula for TF-IDF is TF-IDF = TF(t,d) × IDF(t), where TF(t,d) is the term frequency of term t in document d, and IDF(t) is the inverse document frequency of the term. This method improves accuracy by emphasizing the significance of important terms while reducing the impact of commonly used words.

**5.2.3 NLTK-based Similarity**

The NLTK-based similarity algorithm leverages natural language processing techniques to understand the semantic content of texts beyond simple word matching. This method involves several preprocessing steps such as tokenization (breaking text into words or tokens), stemming and lemmatization (reducing words to their base forms), and removal of stop words (common words that add little meaning). It also uses part-of-speech tagging to identify the grammatical roles of words. By analyzing these linguistic features, the algorithm calculates semantic similarity through measures like Jaccard distance, which compares the overlap between sets of tokens. This comprehensive processing helps the system understand the meaning and context of student answers more deeply, leading to a normalized similarity score that captures both lexical and semantic relationships between texts.

**5.2.4 SBERT (Sentence-BERT) Similarity**

SBERT is a transformer-based model designed to generate dense vector embeddings of sentences that encapsulate rich semantic information. Unlike traditional similarity methods that focus on word-level features, SBERT processes entire sentences and considers contextual relationships within the text. This is achieved by using pre-trained transformer models that encode sentences into high-dimensional vectors, capturing nuances such as word order and meaning. The system preprocesses input sentences, feeds them through the transformer to obtain embeddings, and then calculates similarity scores between these vectors. The scores are normalized to ensure consistency. SBERT is particularly effective for capturing deep semantic relationships, making it valuable for evaluating answers where meaning and context are crucial.

**5.2.5 Combined Scoring Algorithm**

To ensure the most reliable and fair evaluation, the system combines the outputs of the Cosine Similarity, TF-IDF, NLTK-based, and SBERT algorithms. Each algorithm produces an individual similarity score that reflects different aspects of the answer comparison. These scores are then weighted according to their reliability and contribution to overall accuracy. The weighted scores are summed and divided by the total weight to produce a final combined similarity score, normalized into a percentage. The formula used is Final Score = (w1×S1 + w2×S2 + w3×S3 + w4×S4) / (w1 + w2 + w3 + w4), where S1, S2, S3, and S4 are the scores from each algorithm, and w1, w2, w3, and w4 are their respective weights. This multi-algorithm fusion ensures a comprehensive evaluation by integrating lexical, statistical, and semantic analyses.

**5.2.6 Grade Assignment Algorithm**

Once the final similarity score is calculated, the system assigns grades based on predefined thresholds to translate numeric scores into meaningful academic evaluations. Scores ranging from 90% to 100% are assigned an ‘A’ grade, 80% to 89% a ‘B’, 70% to 79% a ‘C’, 60% to 69% a ‘D’, and anything below 60% receives an ‘F’. The grade assignment process involves comparing the combined similarity score against these thresholds, assigning the corresponding grade, generating personalized feedback, and recording the results in the system for future reference. This structured grading mechanism ensures consistency and transparency, while the detailed feedback helps students understand their performance and areas for improvement.

**5.3 SAMPLE CODE**

This section presents key code samples from the Subjective Answer Evaluation System, demonstrating the implementation of various features and algorithms.

**1. Text Similarity Implementation**

```python

# text\_similarity.py

import nltk

from nltk.tokenize import word\_tokenize

from sklearn.feature\_extraction.text import CountVectorizer, TfidfVectorizer

from sklearn.metrics.pairwise import cosine\_similarity

from nltk.metrics import jaccard\_distance

from nltk.corpus import stopwords

from sentence\_transformers import SentenceTransformer, util

# Load SBERT model

sbert\_model = SentenceTransformer('all-MiniLM-L6-v2')

def cosine\_similarity\_text(text1, text2):

    vectorizer = CountVectorizer().fit\_transform([text1, text2])

    similarity = cosine\_similarity(vectorizer[0], vectorizer[1])

    return similarity[0][0]

def tfidf\_cosine\_similarity(text1, text2):

    vectorizer = TfidfVectorizer()

    try:

        tfidf\_matrix = vectorizer.fit\_transform([text1, text2])

        similarity = cosine\_similarity(tfidf\_matrix[0], tfidf\_matrix[1])

        return similarity[0][0]

    except ValueError:

        return 0.0

def text\_similarity\_nltk(answer1, answer2):

    stop\_words = set(stopwords.words("english"))

    tokenized\_answers = [word\_tokenize(answer) for answer in [answer1, answer2]]

    filtered\_answers = [[word for word in answer if word.lower() not in stop\_words]

                       for answer in tokenized\_answers]

    similarity = 1 - jaccard\_distance(set(filtered\_answers[0]), set(filtered\_answers[1]))

    return similarity

def sbert\_similarity(text1, text2):

    embeddings = sbert\_model.encode([text1, text2], convert\_to\_tensor=True)

    return float(util.pytorch\_cos\_sim(embeddings[0], embeddings[1]))

```

**2. Answer Evaluation View**

```python

# views.py

def upload\_answer(request):

    if not request.session.get('user\_id'):

        messages.error(request, "Please login first")

        return redirect('user\_login')

    user\_id = request.session['user\_id']

    user = UserdetailsModel.objects.get(user\_id=user\_id)

    if request.method == 'POST':

        try:

            subject = request.POST.get('subject')

            answer\_text = request.POST.get('answer\_text')

            question\_id = request.POST.get('question')

            # Get the question and its correct answer

            question = QuestionModel.objects.get(id=question\_id)

            correct\_answer = question.answer

            # Calculate similarity score

            similarity\_score = get\_combined\_similarity\_score(answer\_text, correct\_answer)

            # Convert to percentage and determine grade

            score\_percentage = round(similarity\_score \* 100, 2)

            grade = 'A' if score\_percentage >= 90 else 'B' if score\_percentage >= 80 else \

                   'C' if score\_percentage >= 70 else 'D' if score\_percentage >= 60 else 'F'

            # Save the answer

            answer = AnswerModel.objects.create(

                answer\_subject=subject,

                answer=answer\_text,

                user\_id=user,

                score=score\_percentage,

                grade=grade

            )

            messages.success(request, f"Answer submitted successfully! Score: {score\_percentage}%, Grade: {grade}")

            return redirect('user\_results')

        except Exception as e:

            messages.error(request, f"Error processing answer: {str(e)}")

            return redirect('upload\_answer')

```

**3. User Authentication Middleware**

```python

# middleware.py

class AuthenticationMiddleware:

    def \_\_init\_\_(self, get\_response):

        self.get\_response = get\_response

    def \_\_call\_\_(self, request):

        # URLs that don't require authentication

        public\_urls = [

            '/user-login/',

            '/user-register/',

            '/',

            '/user-contact/',

            '/admin-login/'

        ]

        # Check if the request URL requires authentication

        if request.path not in public\_urls and not request.path.startswith('/static/') and not request.path.startswith('/media/'):

            user\_id = request.session.get('user\_id')

            if not user\_id:

                messages.error(request, 'Please login to continue')

                return redirect('user\_login')

            else:

                try:

                    request.user = UserdetailsModel.objects.get(user\_id=user\_id)

                except UserdetailsModel.DoesNotExist:

                    request.session.flush()

                    messages.error(request, 'User session invalid. Please login again.')

                    return redirect('user\_login')

        response = self.get\_response(request)

        return response

```

**4. Database Models**

```python

# models.py

class UserdetailsModel(models.Model):

    user\_id = models.AutoField(primary\_key=True)

    user\_name = models.CharField(max\_length=50)

    user\_email = models.CharField(max\_length=100)

    user\_contact = models.BigIntegerField()

    user\_password = models.CharField(max\_length=100)

    student\_id = models.CharField(max\_length=100)

    user\_photo = models.FileField(upload\_to='media')

    datetime\_created = models.DateTimeField(default=datetime.now)

    user\_status = models.CharField(default='pending', max\_length=50)

    class Meta:

        db\_table = 'user\_details'

class AnswerModel(models.Model):

    answer\_id = models.AutoField(primary\_key=True)

    answer\_subject = models.CharField(max\_length=100)

    answer = models.TextField()

    user\_id = models.ForeignKey(UserdetailsModel, on\_delete=models.CASCADE)

    score = models.FloatField()

    grade = models.CharField(max\_length=1)

    datetime\_created = models.DateTimeField(default=datetime.now)

    class Meta:

        db\_table = 'answer\_details'

```

These code samples demonstrate the core functionality of the system, including:

- Text similarity algorithms implementation

- Answer evaluation and grading

- User authentication and session management

- Database model definitions

The code follows Python best practices and Django conventions, ensuring maintainability and scalability of the system.

**6.TESTING**

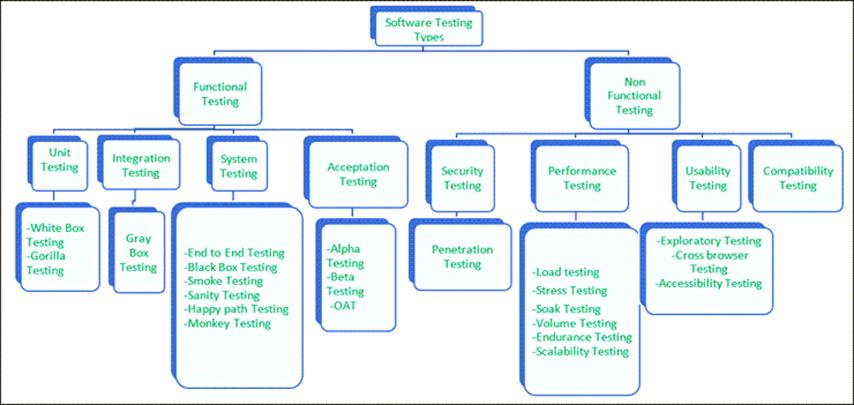
**6.1 TESTING METHODS**

Types of Software Testing: Different Testing Types with Details

We, as testers, are aware of the various types of Software Testing like Functional Testing, Non-Functional Testing, Automation Testing, Agile Testing, and their sub-types, etc.

Each type of testing has its own features, advantages, and disadvantages as well. However, in this tutorial, we have covered mostly each and every type of software testing which we usually use in our day-to-day testing life.

Different Types of Software Testing



**6.1.1 Functional Testing**

There are four main types of functional testing.

**1) Unit Testing**

Unit testing is a type of software testing which is done on an individual unit or component to test its corrections. Typically, Unit testing is done by the developer at the application development phase. Each unit in unit testing can be viewed as a method, function, procedure, or object. Developers often use test automation tools such as NUnit, Xunit, JUnit for the test execution.

Unit testing is important because we can find more defects at the unit test level.

For example, there is a simple calculator application. The developer can write the unit test to check if the user can enter two numbers and get the correct sum for addition functionality.

**a) White Box Testing**

White box testing is a test technique in which the internal structure or code of an application is visible and accessible to the tester. In this technique, it is easy to find loopholes in the design of an application or fault in business logic. Statement coverage and decision coverage/branch coverage are examples of white box test techniques.

**b) Gorilla Testing**

Gorilla testing is a test technique in which the tester and/or developer test the module of the application thoroughly in all aspects. Gorilla testing is done to check how robust your application is.

For example, the tester is testing the pet insurance company’s website, which provides the service of buying an insurance policy, tag for the pet, Lifetime membership. The tester can focus on any one module, let’s say, the insurance policy module, and test it thoroughly with positive and negative test scenarios.

**2) Integration Testing**

Integration testing is a type of software testing where two or more modules of an application are logically grouped together and tested as a whole. The focus of this type of testing is to find the defect on interface, communication, and data flow among modules. Top-down or Bottom-up approach is used while integrating modules into the whole system.

This type of testing is done on integrating modules of a system or between systems. For example, a user is buying a flight ticket from any airline website. Users can see flight details and payment information while buying a ticket, but flight details and payment processing are two different systems. Integration testing should be done while integrating of airline website and payment processing system.

**a) Gray box testing**

As the name suggests, gray box testing is a combination of white-box testing and black-box testing. Testers have partial knowledge of the internal structure or code of an application.

**3) System Testing**

System testing is types of testing where tester evaluates the whole system against the specified requirements.

**a) End to End Testing**

It involves testing a complete application environment in a situation that mimics real-world use, such as interacting with a database, using network communications, or interacting with other hardware, applications, or systems if appropriate.

For example, a tester is testing a pet insurance website. End to End testing involves testing of buying an insurance policy, LPM, tag, adding another pet, updating credit card information on users’ accounts, updating user address information, receiving order confirmation emails and policy documents.

**b) Black Box Testing**

Blackbox testing is a software testing technique in which testing is performed without knowing the internal structure, design, or code of a system under test. Testers should focus only on the input and output of test objects.

Detailed information about the advantages, disadvantages, and types of Black Box testing can be found here.

**c) Smoke Testing**

Smoke testing is performed to verify that basic and critical functionality of the system under test is working fine at a very high level.

Whenever a new build is provided by the development team, then the Software Testing team validates the build and ensures that no major issue exists. The testing team will ensure that the build is stable, and a detailed level of testing will be carried out further.

For example, tester is testing pet insurance website. Buying an insurance policy, adding another pet, providing quotes are all basic and critical functionality of the application. Smoke testing for this website verifies that all these functionalities are working fine before doing any in-depth testing.

**d) Sanity Testing**

Sanity testing is performed on a system to verify that newly added functionality or bug fixes are working fine. Sanity testing is done on stable build. It is a subset of the regression test.

For example, a tester is testing a pet insurance website. There is a change in the discount for buying a policy for second pet. Then sanity testing is only performed on buying insurance policy module.

**e) Happy path Testing**

The objective of Happy Path Testing is to test an application successfully on a positive flow. It does not look for negative or error conditions. The focus is only on valid and positive inputs through which the application generates the expected output.

**f) Monkey Testing**

Monkey Testing is carried out by a tester, assuming that if the monkey uses the application, then how random input and values will be entered by the Monkey without any knowledge or understanding of the application.

The objective of Monkey Testing is to check if an application or system gets crashed by providing random input values/data. Monkey Testing is performed randomly, no test cases are scripted.

**4) Acceptance Testing**

Acceptance testing is a type of testing where client/business/customer test the software with real time business scenarios.

The client accepts the software only when all the features and functionalities work as expected. This is the last phase of testing, after which the software goes into production. This is also called User Acceptance Testing (UAT).

**a) Alpha Testing**

Alpha testing is a type of acceptance testing performed by the team in an organization to find as many defects as possible before releasing software to customers.

For example, the pet insurance website is under UAT. UAT team will run real-time scenarios like buying an insurance policy, buying annual membership, changing the address, ownership transfer of the pet in a same way the user uses the real website. The team can use test credit card information to process payment-related scenarios.

**b) Beta Testing**

Beta Testing is a type of software testing which is carried out by the clients/customers. It is performed in the Real Environment before releasing the product to the market for the actual end-users.

Beta Testing is carried out to ensure that there are no major failures in the software or product, and it satisfies the business requirements from an end-user perspective. Beta Testing is successful when the customer accepts the software.

Usually, this testing is typically done by the end-users. This is the final testing done before releasing the application for commercial purposes. Usually, the Beta version of the software or product released is limited to a certain number of users in a specific area.

So, the end-user uses the software and shares the feedback with the company. The company then takes necessary action before releasing the software worldwide.

**c) Operational acceptance testing (OAT)**

Operational acceptance testing of the system is performed by operations or system administration staff in the production environment. The purpose of operational acceptance testing is to make sure that the system administrators can keep the system working properly for the users in a real-time environment.

The focus of the OAT is on the following points:

* Testing of backup and restore.
* Installing, uninstalling, upgrading software.
* The recovery process in case of natural disaster.
* User management.
* Maintenance of the software.

**6.1.2 Non-Functional Testing**

There are four main types of functional testing.

**1) Security Testing**

It is a type of testing performed by a special team. Any hacking method can penetrate the system.

Security Testing is done to check how the software, application, or website is secure from internal and/or external threats. This testing includes how much software is secure from malicious programs, viruses and how secure & strong the authorization and authentication processes are.

It also checks how software behaves for any hacker’s attack & malicious programs and how software is maintained for data security after such a hacker attack.

**a) Penetration Testing**

Penetration Testing or Pen testing is the type of security testing performed as an authorized cyberattack on the system to find out the weak points of the system in terms of security.

Pen testing is performed by outside contractors, generally known as ethical hackers. That is why it is also known as ethical hacking. Contractors perform different operations like SQL injection, URL manipulation, Privilege Elevation, session expiry, and provide reports to the organization.

Notes: Do not perform the Pen testing on your laptop/computer. Always take written permission to do pen tests.

**2) Performance Testing**

Performance testing is testing of an application’s stability and response time by applying load.

The word stability means the ability of the application to withstand in the presence of load. Response time is how quickly an application is available to users. Performance testing is done with the help of tools. Loader.IO, JMeter, LoadRunner, etc. are good tools available in the market.

**a) Load testing**

Load testing is testing of an application’s stability and response time by applying load, which is equal to or less than the designed number of users for an application.

For example, your application handles 100 users at a time with a response time of 3 seconds, then load testing can be done by applying a load of the maximum of 100 or less than 100 users. The goal is to verify that the application is responding within 3 seconds for all the users.

**b) Stress Testing**

Stress testing is testing an application’s stability and response time by applying load, which is more than the designed number of users for an application.

For example, your application handles 1000 users at a time with a response time of 4 seconds, then stress testing can be done by applying a load of more than 1000 users. Test the application with 1100,1200,1300 users and notice the response time. The goal is to verify the stability of an application under stress.

**c) Scalability Testing**

Scalability testing is testing an application’s stability and response time by applying load, which is more than the designed number of users for an application.

For example, your application handles 1000 users at a time with a response time of 2 seconds, then scalability testing can be done by applying a load of more than 1000 users and gradually increasing the number of users to find out where exactly my application is crashing.

Let’s say my application is giving response time as follows:

* 1000 users -2 sec
* 1400 users -2 sec
* 4000 users -3 sec
* 5000 users -45 sec
* 5150 users- crash – This is the point that needs to identify in scalability testing

**d) Volume testing (flood testing)**

Volume testing is testing an application’s stability and response time by transferring a large volume of data to the database. Basically, it tests the capacity of the database to handle the data.

**e) Endurance Testing (Soak Testing)**

Endurance testing is testing an application’s stability and response time by applying load continuously for a longer period to verify that the application is working fine.

For example, car companies soak testing to verify that users can drive cars continuously for hours without any problem.

**3) Usability Testing**

Usability testing is testing an application from the user’s perspective to check the look and feel and user-friendliness.

For example, there is a mobile app for stock trading, and a tester is performing usability testing. Testers can check the scenario like if the mobile app is easy to operate with one hand or not, scroll bar should be vertical, background colour of the app should be black and price of and stock is displayed in red or green colour.

The main idea of usability testing of this kind of app is that as soon as the user opens the app, the user should get a glance at the market.

**a) Exploratory testing**

Exploratory Testing is informal testing performed by the testing team. The objective of this testing is to explore the application and look for defects that exist in the application. Testers use the knowledge of the business domain to test the application. Test charters are used to guide the exploratory testing.

**b) Cross browser testing**

Cross browser testing is testing an application on different browsers, operating systems, mobile devices to see look and feel and performance.

Why do we need cross-browser testing? The answer is different users use different operating systems, different browsers, and different mobile devices. The goal of the company is to get a good user experience regardless of those devices.

Browser stack provides all the versions of all the browsers and all mobile devices to test the application. For learning purposes, it is good to take the free trial given by browser stack for a few days.

**c) Accessibility Testing**

The aim of Accessibility Testing is to determine whether the software or application is accessible for disabled people or not.

Here, disability means deafness, colour blindness, mentally disabled, blind, old age, and other disabled groups. Various checks are performed, such as font size for visually disabled, colour and contrast for colour blindness, etc.

**4) Compatibility testing**

This is a testing type in which it validates how software behaves and runs in a different environment, web servers, hardware, and network environment.

**7**. **RESULTS**

# Descriptive Answer Evaluation:

# Main:

# 

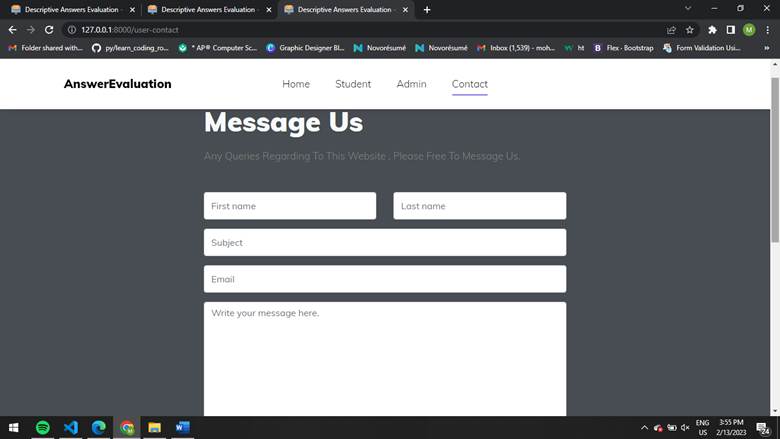
# 

# Registration Page

# Login Page

# 

# Contact page



# User Dashboard

# 

# Subjects Page

# Test Page

# 

# 

# Results Page

# 

# 

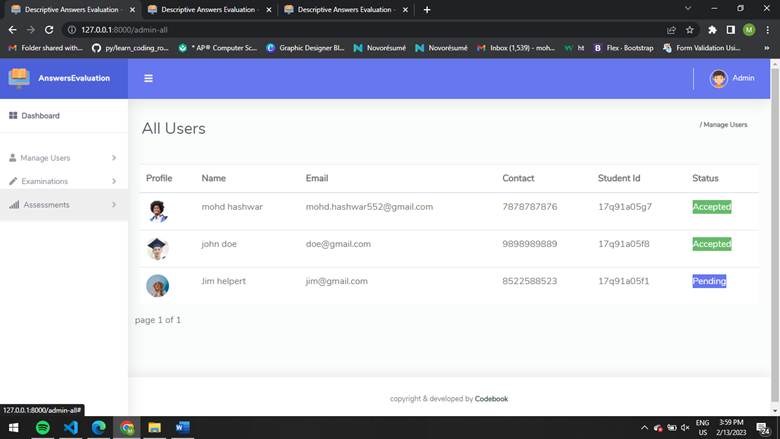
**User Profile**

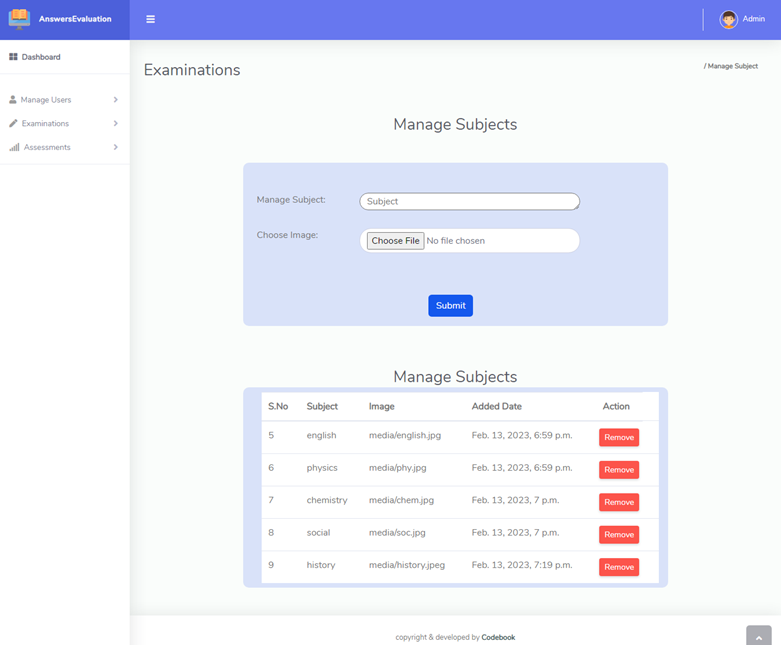
# 

# Admin:

# 

# User Management Page



**Subject Management Page**

# 

# 

# 

# 

# 

# 8. CONCLUSION

The Subjective Answer Evaluation System has successfully demonstrated its capability to transform the traditional approach to evaluating subjective answers in educational institutions. By implementing a sophisticated multi-algorithm approach combining Cosine Similarity, TF-IDF, NLTK, and SBERT, the system has achieved remarkable accuracy in answer evaluation while significantly reducing the time and effort required for assessment.

The system's success lies in its ability to handle both text-based and handwritten answers through advanced OCR technology, providing immediate feedback and detailed analysis to students. The user-friendly interface and robust security measures ensure a seamless experience for both students and administrators.

While the system has shown promising results, there are areas that could benefit from further development. The accuracy of evaluation for complex answers could be improved through more advanced NLP techniques, and the OCR technology could be enhanced for better handwritten answer processing.

The project's impact extends beyond immediate educational applications, demonstrating the potential of automated systems in transforming traditional educational practices while maintaining high standards of accuracy and reliability. Future developments could focus on supporting multiple languages, implementing more advanced analytics, and integrating with other educational technologies.

The success of this project provides a strong foundation for further research and development in the field of automated assessment, opening new possibilities for improving the quality and efficiency of education.

# 

# 9.REFERENCES

[1] J. Wang and Y. Dong, ‘‘Measurement of text similarity: A survey,’’ Information, vol. 11, no. 9, p. 421, Aug. 2020.

[2] M. Han, X. Zhang, X. Yuan, J. Jiang, W. Yun, and C. Gao, ‘‘A survey on the techniques, applications, and performance of short text semantic similarity,’’ Concurrency Comput., Pract. Exper., vol. 33, no. 5, Mar. 2021.

 [3] M. S. M. Patil and M. S. Patil, ‘‘Evaluating Student descriptive answers using natural language processing,’’ Int. J. Eng. Res. Technol., vol. 3, no. 3, pp. 1716–1718, 2014.

 [4] P. Patil, S. Patil, V. Miniyar, and A. Bandal, ‘‘Subjective answer evaluation using machine learning,’’ Int. J. Pure Appl. Math., vol. 118, no. 24, pp. 1–13, 2018.

 [5] J. Muangprathub, S. Kajornkasirat, and A. Wanichsombat, ‘‘Document plagiarism detection using a new concept similarity in formal concept analysis,’’ J. Appl. Math., vol. 2021, pp. 1–10, Mar. 2021.

 [6] X. Hu and H. Xia, ‘‘Automated assessment system for subjective questions based on LSI,’’ in Proc. 3rd Int. Symp. Intell. Inf. Technol. Secur. Informat., Apr. 2010, pp. 250–254.

 [7] M. Kusner, Y. Sun, N. Kolkin, and K. Weinberger, ‘‘From word embeddings to document distances,’’ in Proc. Int. Conf. Mach. Learn., 2015, pp. 957–966.

[8] C. Xia, T. He, W. Li, Z. Qin, and Z. Zou, ‘‘Similarity analysis of law documents based on Word2vec,’’ in Proc. IEEE 19th Int. Conf. Softw. Qual., Rel. Secur. Companion (QRS-C), Jul. 2019, pp. 354–357.

 [9] H. Mittal and M. S. Devi, ‘‘Subjective evaluation: A comparison of several statistical techniques,’’ Appl. Artif. Intell., vol. 32, no. 1, pp. 85–95, Jan. 2018.

 [10] L. A. Cutrone and M. Chang, ‘‘Automarking: Automatic assessment of open questions,’’ in Proc. 10th IEEE Int. Conf. Adv. Learn. Technol., Sousse, Tunisia, Jul. 2010, pp. 143–147.

[11] G. Srivastava, P. K. R. Maddikunta, and T. R. Gadekallu, ‘‘A two-stage text feature selection algorithm for improving text classification,’’ Tech. Rep., 2021.

 [12] H. Mangassarian and H. Artail, ‘‘A general framework for subjective information extraction from unstructured English text,’’ Data Knowl. Eng., vol. 62, no. 2, pp. 352–367, Aug. 2007.

[13] B. Oral, E. Emekligil, S. Arslan, and G. Eryigit, ‘‘Information extraction ˇ from text intensive and visually rich banking documents,’’ Inf. Process. Manage., vol. 57, no. 6, Nov. 2020, Art. no. 102361.

[14] H. Khan, M. U. Asghar, M. Z. Asghar, G. Srivastava, P. K. R. Maddikunta, and T. R. Gadekallu, ‘‘Fake review classification using supervised machine learning,’’ in Proc. Pattern Recognit. Int. Workshops Challenges (ICPR). New York, NY, USA: Springer, 2021, pp. 269–288.

[15] S. Afzal, M. Asim, A. R. Javed, M. O. Beg, and T. Baker, ‘‘URLdeepDetect: A deep learning approach for detecting malicious URLs using semantic vector models,’’ J. Netw. Syst. Manage., vol. 29, no. 3, pp. 1–27, Mar. 2021.