ABSTRACT

An advanced AI-based proctoring exam system was developed to address the increasing demand for remote education and online examination platforms. By integrating cutting-edge technologies such as YOLO for real-time object detection, audio analysis for gaze detection, video tracking, and tab shifting detection, the system ensures comprehensive monitoring and maintenance of exam integrity in remote learning environments. The system architecture comprises two primary modules: the Teacher Module and the Student Module. The Teacher Module empowers instructors to create and administer exams, monitor live exam sessions, and review exam results, while the Student Module enables students to access assigned exams, undertake them within specified parameters, and receive real-time monitoring during the examination process. Implementation details cover functionalities such as seamless exam creation, student management, live monitoring, and result review for the Teacher Module, and exam access, efficient exam administration, real-time monitoring through AI-driven algorithms, and personalized feedback for the Student Module. Leveraging AI technologies enhances monitoring capabilities, allowing effective detection of unauthorized activities during exams. YOLO facilitates real-time object detection, audio analysis aids in gaze detection, video tracking monitors facial expressions and eye movements, and tab shifting detection identifies unauthorized activity during exams. Security measures include data encryption, robust user authentication mechanisms, and compliance with relevant privacy regulations. Rigorous testing ensures the system's reliability, scalability, and effectiveness, positioning it as a groundbreaking solution for online proctoring in the education sector.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

In today's rapidly evolving educational landscape, the demand for remote education and online examination platforms has surged. To address this need, we have developed a cutting-edge solution that leverages artificial intelligence (AI) technologies to ensure the integrity and security of online examinations. This document serves as a comprehensive guide to understanding the architecture, implementation details, and functionalities of our proctoring system.

In response to the challenges posed by remote education, our system integrates state-of-the-art technologies such as YOLO for real-time object detection, audio analysis for gaze detection, video tracking, and tab shifting detection. These technologies enable us to monitor exam sessions in real-time, ensuring that students adhere to examination rules and regulations without compromising the integrity of the assessment process.

The system architecture is designed with scalability, reliability, and user-friendliness in mind. It comprises two primary modules: the Teacher Module and the Student Module. The Teacher Module empowers instructors to create, administer, and monitor exams, while the Student Module provides students with access to exams and real-time monitoring during the examination process.

Throughout this documentation, we will delve into the implementation details of each module, highlighting key functionalities and how they contribute to the overall effectiveness of the system. We will also discuss the integration of AI technologies, security considerations, and the rigorous testing and evaluation procedures undertaken to ensure the system's reliability and robustness in real-world scenarios.
Our AI-based proctoring exam system represents a paradigm shift in online examination platforms, offering educators and students alike a secure and streamlined solution for conducting exams remotely. We invite you to explore this documentation to gain a deeper understanding of our innovative system and its potential applications in the education sector.

1.1. MOTIVATION

In response to the growing demand for remote education and online examination platforms, our motivation to develop this project was fueled by a crucial imperative: to familiarize students with a robust proctoring environment. We recognized the significance of ensuring that students feel comfortable and confident while attending actual interview exams conducted under this software. This driving force propelled us to integrate cutting-edge technologies and develop an advanced AI-based proctoring exam system. By providing students with a familiar and secure examination environment, we aim to alleviate any apprehensions or uncertainties they may have, thus enabling them to perform at their best during crucial assessment moments. Our commitment to creating a supportive and user-friendly platform underscores our dedication to facilitating seamless educational experiences, empowering both educators and learners to navigate the challenges of remote learning with ease and confidence.

1.2. PROBLEM STATEMENT

Traditional exam monitoring methods face challenges in detecting and preventing cheating, ensuring exam integrity, and offering convenient solutions for students and institutions. The proposed system aims to revolutionize the recruitment process by automating interviewing and proctoring through AI, potentially leading to cost savings and increased efficiency. Addressing students' difficulties with interview exams on proctoring websites due to unfamiliarity with tracking mechanisms is a priority. To tackle these issues, our project proposes an AI-based proctoring system that integrates sophisticated natural language processing with technologies like Deep Learning and YOLO. The objective is to provide a comprehensive evaluation of candidates, aiding businesses in making informed hiring decisions and familiarizing students with proctoring websites, thus enhancing the problem statement.

1.3. OBJECTIVE OF THE PROJECT

- The primary objective of the Online Exam Proctor system is to ensure the integrity and security of online examinations conducted in remote learning environments.
- Through the integration of advanced technologies such as artificial intelligence (AI), real-time monitoring, and analysis, the system aims to detect and prevent instances of cheating, unauthorized aid, or irregular behavior during exams.
- By providing robust monitoring and surveillance capabilities, the Online Exam Proctor system helps maintain a fair and equitable examination environment, thereby upholding academic integrity and ensuring the validity of assessment outcomes.
- The system seeks to enhance the overall user experience for both educators and students by offering seamless exam administration, reliable authentication mechanisms, and timely feedback on exam performance.
- Ultimately, the objective is to provide a trusted and efficient platform for conducting online exams, enabling educational institutions to adapt to the challenges of remote learning while maintaining high standards of assessment integrity.

1.4. PROJECT OVERVIEW

The project entails the development of an advanced AI-based proctoring exam system to address the growing demand for remote education and online examination platforms. Leveraging cutting-edge technologies such as YOLO for real-time object detection, audio analysis for gaze detection, video tracking, and tab shifting detection, the system aims to ensure the integrity and security of online examinations. It comprises two primary modules: the Teacher Module and the Student Module. The Teacher Module empowers instructors to create and administer exams, monitor live exam sessions, and review exam results, while the Student Module enables students to access assigned exams, undertake them within specified parameters, and receive real-time monitoring during the examination process. Key functionalities include seamless exam creation, student management, live monitoring, result review, exam access, administration, real-time monitoring, and feedback mechanisms. The integration of AI technologies enhances monitoring capabilities, enabling the system to detect unauthorized activities during exams effectively. Security measures such as data encryption, user authentication, and compliance with privacy regulations ensure the confidentiality and integrity of sensitive information. Rigorous testing, including unit testing, integration testing, user acceptance testing (UAT), and performance testing, validates the system's reliability, scalability, and effectiveness in real-world scenarios. The project aims to set a new benchmark for online proctoring solutions in the education sector, offering educators confidence in exam administration and ensuring integrity throughout the examination process.

2. LITERATURE SURVEY

Author(s)	Strategies / Method	Advantages	Disadvantages
Neelesh Chandra M, Piyush Sharma	Eye tracking	This project detects whether test taker is present in the frame and observe the eye gazing	Doesn’t give warnings to user
Athi Narayanan, S. Kamal Bijlani and Swathi	Voice Detection	This project detects whether test taker is present in the frame or not and determine whether he is cheating or not	This project doesn’t analyze the background audio
S. P. Saurav, P. Pandey,	Object detection	This project detects whether test taker is present in the frame and checks if any suspicious items are in the frame too, i.e. other than user face	The project doesn’t identify phone and doesn’t track audio in background

Table 2.1 Literature Survey

3. ANALYSIS

3.1. EXISTING SYSTEM

The existing system for remote education and online examination platforms typically relies on traditional methods of proctoring, which may include manual invigilation, webcam monitoring, and limited automated monitoring tools. These systems often lack the advanced capabilities required to effectively detect and prevent instances of cheating or unauthorized behavior during online exams. They may also suffer from usability issues, security vulnerabilities, and scalability challenges.

In the absence of robust AI-based monitoring technologies, the existing systems may struggle to provide real-time monitoring, comprehensive analytics, and actionable insights into student behavior. Furthermore, the reliance on manual intervention for exam administration and monitoring can lead to inconsistencies, human errors, and increased administrative burden for educators.

3.2. PROPOSED SYSTEM

The proposed solution introduces an advanced AI-based proctoring exam system that aims to address the evolving needs and challenges of modern education. At the core of the proposed system lies its utilization of cutting-edge technologies, including YOLO for real-time object detection, audio analysis for gaze detection, video tracking, and tab shifting detection. These technologies enable the system to provide comprehensive monitoring and maintenance of exam integrity, ensuring a fair and equitable assessment environment for all stakeholders involved.

The system is designed with modularity and scalability in mind, comprising two primary modules: the Teacher Module and the Student Module. The Teacher Module empowers educators with robust tools for exam creation, administration, and monitoring, while the Student Module offers students a seamless and user-friendly interface for accessing exams, undertaking them, and receiving real-time monitoring during the examination process.

Key functionalities of the proposed system include intuitive exam creation interfaces, efficient student management features, real-time monitoring capabilities, comprehensive result review tools, and personalized feedback mechanisms. Through the integration of AI technologies, the system enhances monitoring capabilities by detecting and flagging instances of cheating or unauthorized behavior during exams.

The proposed system undergoes rigorous testing and evaluation procedures to validate its reliability, scalability, and effectiveness in real-world scenarios. This includes extensive unit testing, integration testing, user acceptance testing (UAT), and performance testing, ensuring that the system meets the highest standards of quality and usability.

Overall, the proposed system sets out to redefine the landscape of online proctoring solutions, offering educators a powerful tool to administer exams with confidence and integrity, and providing students with a trusted platform for undertaking assessments in a secure and fair environment.

3.3. SOFTWARE REQUIREMENTS

SDK:

- VS Code (Visual Studio Code)
- Python
- Django
- Open CV

Operating System: Windows

Database:

- SQL
- PHP my admin

3.4 HARDWARE REQUIREMENTS

- Camera/ Webcam
- Microphone
- Processor i7, i9, i11
- >8GB RAM for better performance

4. IMPLEMENTATION

4.1. METHODOLOGY

4.1.1. Requirement Analysis:

Requirement analysis is a crucial step in the development of any system, especially one as complex as a proctoring system. Here's a detailed elaboration on conducting requirement analysis:

1. **Stakeholder Discussions**: Thorough discussions are conducted with stakeholders, including educators, students, and administrators. These discussions aim to gain a deep understanding of their needs, challenges, and expectations regarding the proctoring system. Key topics discussed include:

- **Educators**: Their primary concerns may revolve around maintaining exam integrity, monitoring student behavior during exams, and providing fair assessment methods.
- **Students**: Their concerns may include privacy issues, the ease of use of the proctoring system, and ensuring that the system does not create unnecessary stress or anxiety.
- **Administrators**: They may focus on system security, scalability to accommodate a large number of users, and compliance with regulations and standards.

2. **Defining Objectives and Requirements**: Based on the discussions with stakeholders, clear objectives and requirements for the proctoring system are defined. These requirements encompass various aspects, including functionality, security, scalability, and usability criteria. Key requirements may include:

- **Functionality:** The system should support features such as exam creation, real-time monitoring, AI-based proctoring, student authentication, and result analysis.
- **Security:** Robust security measures should be implemented to protect sensitive data, ensure the integrity of exams, and prevent cheating.
- **Scalability**: The system should be designed to handle a large number of users simultaneously, especially during peak times such as exam periods.

- **Usability:** The user interface should be intuitive and easy to navigate for both educators and students, minimizing the learning curve and reducing user errors.

By conducting thorough discussions with stakeholders and defining clear objectives and requirements, the proctoring system can be designed and developed to effectively meet the needs and expectations of all parties involved. Regular communication and feedback loops with stakeholders throughout the development process are essential to ensure that the final product aligns with their vision and requirements.

4.1.2. Research and Technology Selection:

In the phase of Research and Technology Selection for the proctoring system, a thorough investigation into existing proctoring solutions, AI algorithms, and relevant technologies is essential. Here's an elaboration on the process:

1. **Research on Existing Proctoring Solutions**: Begin by examining the landscape of available proctoring solutions. This involves exploring both commercial and open-source options, understanding their features, strengths, and weaknesses. Consider aspects such as the types of monitoring they offer (video, audio, screen capture), authentication methods, and compatibility with various learning management systems (LMS).

2. **Evaluation of AI Algorithms and Technologies**:

a. **Real-Time Monitoring**: Investigate AI algorithms capable of real-time monitoring, such as facial recognition, gaze tracking, and head movement analysis. Look into technologies that can detect suspicious behavior, including looking away from the screen, multiple faces in the camera frame, or unusual mouse or keyboard activity.

b. **Object Detection**: Research object detection algorithms like YOLO (You Only Look Once) or Faster R-CNN for identifying prohibited items or behaviors during exams.

c. **Audio Analysis:** Explore speech recognition and sentiment analysis algorithms for detecting spoken content and analyzing the tone or stress levels in a student's voice

d. **Video Tracking:** Look into algorithms for tracking eye movement, mouse activity, and screen changes to detect potential cheating behaviors.

Key Packages used for entire project development

```
import sys
import face_recognition
from concurrent.futures import ThreadPoolExecutor
import cv2
import mediapipe as mp
import numpy as np
import time
import math
import random
import os
import json
import shutil
import keyboard
import pyautogui
import pygetwindow as gw
import webbrowser
import pyperclip
from ultralytics import YOLO
import threading
from multiprocessing import Process
import pyaudio
import struct
import wave
import datetime
import subprocess
```

3**. Suitability Evaluation of AI Frameworks and Tools**:

a. **Consider popular AI frameworks** like TensorFlow, PyTorch, and Keras for implementing AI algorithms due to their extensive libraries and community support.

b**. Evaluate pre-trained models** available in these frameworks for tasks such as face detection, object recognition, and speech analysis to expedite development.

c**. Assess the performance, accuracy, and ease of integration** of different tools and libraries. Look for options that offer scalability and compatibility with the chosen development environment and deployment platform.

4. **Consideration of Ethical and Legal Implications:**
Keep in mind ethical considerations and legal requirements related to privacy, data protection, and consent when selecting technologies for the proctoring system. Ensure compliance with regulations such as GDPR (General Data Protection Regulation) and FERPA (Family Educational Rights and Privacy Act).

By conducting comprehensive research and evaluation of existing solutions, AI algorithms, frameworks, and tools, the proctoring system can be equipped with the most suitable technologies to meet its monitoring and security requirements effectively.

4.1.3. System Design

Designing the system architecture involves creating a blueprint that outlines the overall structure, components, and interactions between modules to ensure the effective functioning of the system. Here's a detailed breakdown of the architecture:

1. **Overall Structure:**

The system architecture consists of three main layers: Presentation Layer, Application Layer, and Data Layer.

2. **Components:**

- **Presentation Layer:** This layer includes the user interface components such as web pages for teachers and students to interact with the system. It's responsible for rendering the user interface and handling user inputs.

- **Application Layer:** This layer contains the business logic and functionality of the system. It consists of modules such as the Teacher Module, Student Module, Exam Module, Monitoring Module, and Analytics Module. Each module is responsible for specific tasks like exam creation, real-time monitoring, and data analysis.

- **Data Layer:** This layer manages the storage and retrieval of data. It includes the database where user profiles, exam information, monitoring data, and analytics are stored.

3. Interactions Between Modules:

- **The Teacher Module** allows teachers to create exams, monitor student progress, and view analytics.
- **The Student Module** enables students to take exams, view their results.
- **The Exam Module** manages the creation, administration, and evaluation of exams.
- **The Monitoring Module** utilizes AI algorithms for real-time monitoring, object detection, audio analysis, and video tracking during exams.
- **The Analytics Module** processes monitoring data to generate insights and analytics for teachers and administrators.

4. **Data Models and Database Schema**:

- **User Profiles**: Contains information about teachers and students, including username, password, email, role, and other relevant details.

- **Exams:** Stores details about exams, such as exam name, duration, questions, and associated students.

- **Monitoring Data**: Captures real-time monitoring data during exams, including video footage, audio recordings, and detected objects.

- **Analytics:** Stores aggregated data and insights generated from monitoring data analysis.

5**. Data Flow Diagrams:**

-Data flow diagrams illustrate the flow of data within the system, including how data moves between modules and external sources such as databases and external APIs.

By implementing this architecture and defining the appropriate data models, database schema, and data flow diagrams, the system can effectively manage user profiles, exams, monitoring data, and analytics while providing a seamless experience for teachers and students.

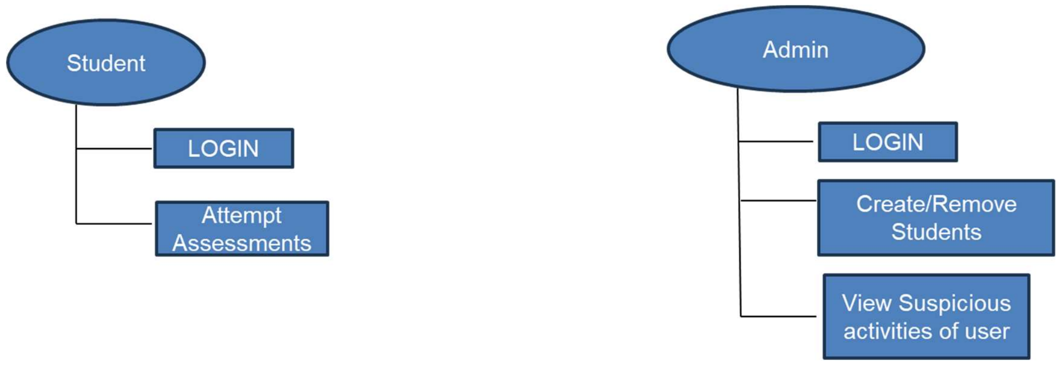


Fig 4.1.3.1: Flowchart of Student and Admin features

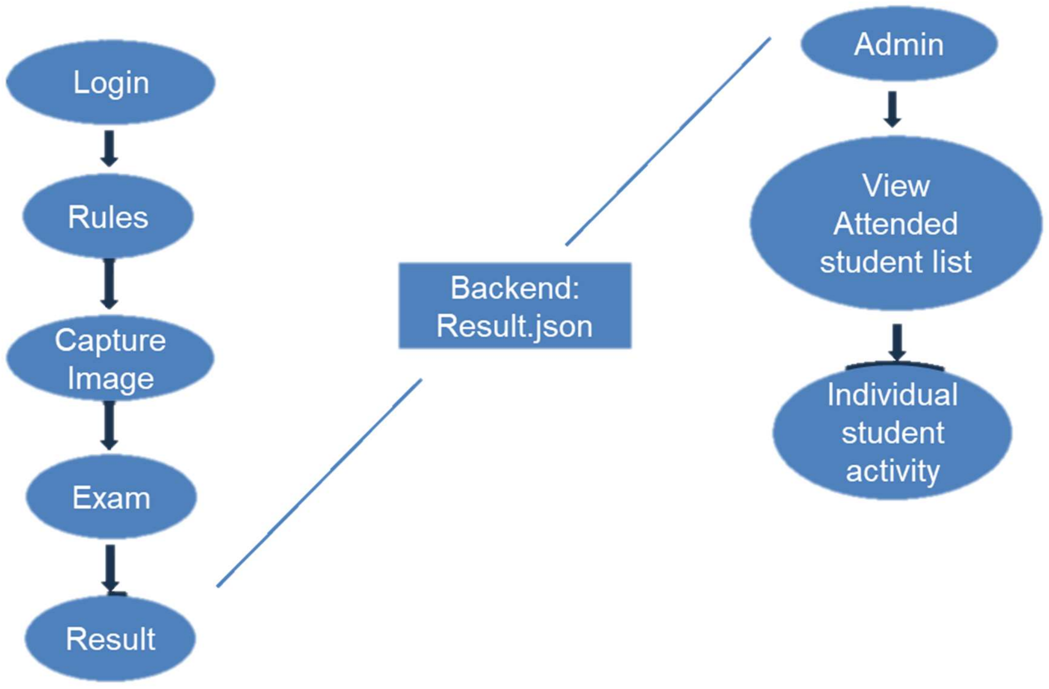


Fig 4.1.3.2: Flowchart of Process Flow

4.1.4. Development:

In developing the teacher and student modules within the Django web framework, adherence to best practices for code organization, security, and scalability is paramount. This involves structuring code in a modular and maintainable manner, implementing secure authentication mechanisms, and designing the system to handle potential increases in user load efficiently.

Integrating AI algorithms and technologies enhances the system's capabilities for real-time monitoring, object detection, audio analysis, and video tracking. Leveraging these technologies ensures seamless interaction with core functionalities, enabling features like automated attendance tracking, behavior analysis, and content understanding.

For exam management, features including creation, administration, monitoring, result review, and feedback mechanisms are implemented according to predefined requirements. This involves designing intuitive interfaces for teachers to create and administer exams, monitoring student progress in real-time, analyzing results, and providing feedback to students.

Throughout development, emphasis is placed on ensuring a user-friendly experience for both teachers and students while maintaining robust security measures to protect sensitive data. Additionally, scalability is considered to accommodate future growth and evolving user needs. Regular testing and iteration are conducted to refine features and address any issues that arise, ultimately delivering a comprehensive and efficient learning management system.

4.1.5. Testing:

In the testing phase, a comprehensive approach is taken to ensure the quality and reliability of the system. Unit testing is conducted for individual components and functionalities to verify their correctness, reliability, and performance in isolation. This involves writing test cases that cover various scenarios and edge cases to ensure thorough coverage of code.
Integration testing is then performed to validate the seamless interaction between different modules and components of the system. This ensures that data flows correctly between various parts of the application and that dependencies are properly managed.

User acceptance testing (UAT) plays a crucial role in gathering feedback from stakeholders and end-users. This involves presenting the system to users in real-world scenarios to validate its usability and effectiveness. Stakeholders and end-users are encouraged to interact with the system, perform tasks relevant to their roles, and provide feedback on their experience. Any bugs or issues identified during UAT are addressed promptly to ensure a smooth user experience.

Throughout the testing phase, collaboration between developers, testers, and end-users is maintained to facilitate communication and ensure that the system meets the needs and expectations of its users. Regular testing cycles and iterations are conducted to refine the system and address any issues that arise, ultimately delivering a reliable and user-friendly product.

4.1.6. Security and Privacy Assurance:

In ensuring the security and privacy of the proctoring system, several measures are implemented to safeguard sensitive information and comply with privacy regulations:

1. **Robust Security Measures:** Implementation of robust security measures involves various aspects such as data encryption, user authentication, and access control mechanisms. Encryption techniques like AES (Advanced Encryption Standard) are utilized to secure data both at rest and in transit. User authentication mechanisms, including multi-factor authentication (MFA), help verify the identity of users accessing the system. Access control mechanisms are employed to regulate user permissions and restrict unauthorized access to sensitive information.

2. **Privacy Regulations Compliance:** The system is designed and developed in compliance with privacy regulations such as GDPR, FERPA, and CCPA (California Consumer Privacy Act). This involves ensuring that user data is collected and processed lawfully, transparently, and for specific purposes. Privacy-by-design principles are adopted to embed privacy considerations into the system architecture and functionality from the outset.

3. **Security Audits and Penetration Testing:** Regular security audits and penetration testing are conducted to identify and address potential vulnerabilities or security loopholes in the system. This involves systematically assessing the system's security posture, identifying weaknesses, and implementing remediation measures to mitigate risks. Penetration testing simulates real-world attack scenarios to evaluate the system's resilience to malicious activities and unauthorized access attempts.

By implementing these security measures and conducting thorough security audits and penetration testing, the proctoring system can effectively safeguard sensitive information, protect user privacy, and mitigate security risks, thereby instilling trust and confidence among stakeholders and users.

4.1.7. Deployment

:

In the deployment phase of the AI-based proctoring exam system, the focus is on deploying it on a secure and scalable infrastructure, such as cloud servers or dedicated hosting environments, and conducting thorough performance testing. Here's how this process is elaborated:

1. **Infrastructure Selection:** Choose an infrastructure option that provides a balance between security, scalability, and cost-effectiveness. Cloud servers, such as those offered by Amazon Web Services (AWS), Microsoft Azure, or Google Cloud Platform (GCP), offer scalability and flexibility. Dedicated hosting environments provide greater control over resources and security but may require more management.

2. **Deployment Process:** Utilize deployment tools and automation to streamline the deployment process. Technologies like Docker and Kubernetes can be used for containerization and orchestration, ensuring consistency and scalability across deployment environments.

3. **Security Measures:** Implement security best practices during deployment, including network segmentation, firewall configuration, and regular security updates. Utilize encryption for data in transit and at rest to protect sensitive information.

4. **Performance Testing**: Conduct thorough performance testing under various load conditions to ensure scalability, responsiveness, and optimal resource utilization. Use load testing tools to simulate realistic user traffic and monitor system metrics such as response times, CPU usage, and memory utilization. Identify performance bottlenecks and optimize system configurations accordingly.

5. **Monitoring and Optimization**: Set up monitoring tools to continuously monitor the deployed system's performance and resource utilization. Implement auto-scaling policies to dynamically adjust resources based on demand. Continuously optimize system configurations and application code to improve performance and efficiency.

6. **Disaster Recovery and Redundancy**: Implement disaster recovery measures such as regular backups, failover mechanisms, and redundancy to ensure high availability and data integrity. Utilize multi-region deployment strategies to minimize downtime and data loss in case of disasters or outages.

By following these steps, the AI-based proctoring exam system can be successfully deployed on a secure and scalable infrastructure, ensuring optimal performance and reliability under various conditions. Regular monitoring and optimization help maintain system health and responsiveness, providing a seamless experience for users.

4.1.8. Maintenance and Iteration:

In the maintenance and iteration phase, establishing processes for ongoing support, monitoring, and gathering feedback is crucial to ensure the continued success and improvement of the AI-based proctoring exam system. Here's how this phase can be elaborated:

1. **Establishment of Maintenance Processes**: Develop robust processes for handling ongoing maintenance tasks, including bug fixes, security updates, and performance optimizations. Implement a ticketing system or issue tracking system to prioritize and track maintenance tasks efficiently.

2. **Monitoring and Support**: Set up monitoring tools to continuously monitor the system's performance, availability, and security post-deployment. Establish procedures for handling alerts and incidents, including escalation paths and response times. Provide timely support to users and stakeholders, addressing any issues or concerns promptly.

3. **Feedback Gathering Mechanisms**: Continuously gather feedback from users and stakeholders through surveys, user interviews, and feedback forms. Encourage open communication channels to facilitate the reporting of issues, suggestions for improvements, and feature requests.

4. **Iteration and Enhancement**: Use feedback gathered from users and stakeholders to identify opportunities for enhancements, updates, and iterations. Prioritize enhancement requests and feature updates based on user needs, business priorities, and feasibility. Implement iterative development cycles, incorporating new features and improvements into regular releases.

5**. Usability Testing and User Training**: Conduct usability testing to evaluate the system's ease of use and user experience. Identify areas for improvement and iteratively refine the user interface and workflows. Provide user training and documentation to ensure that users can effectively utilize the system's features and functionalities.

6**. Performance Optimization**: Continuously monitor system performance and identify opportunities for optimization. Analyze system metrics and performance benchmarks to identify bottlenecks and inefficiencies. Implement optimizations to improve system responsiveness, scalability, and resource utilization.

7**. Compliance and Regulation Updates:** Stay informed about changes in regulatory requirements, privacy laws, and security standards relevant to the proctoring system. Implement updates and changes to ensure ongoing compliance with regulations and industry best practices.

By establishing processes for ongoing maintenance, gathering feedback, and iterating on the system's functionality, usability, and performance, the AI-based proctoring exam system can evolve and improve over time, meeting the changing needs and expectations of users and stakeholders effectively.

4.2 TECHNOLOGIES

4.2.1. YOLO (You Only Look Once):

YOLO (You Only Look Once) stands out as a state-of-the-art object detection algorithm renowned for its real-time capabilities in identifying objects within images or videos. By dividing the input image into a grid and predicting bounding boxes and class probabilities for objects within each grid cell, YOLO achieves both efficiency and accuracy. This efficiency makes it well-suited for swiftly identifying unauthorized materials or individuals within exam environments, enabling the system to promptly recognize and flag irregularities during online exams.

Through its rapid detection capabilities, YOLO ensures the integrity and security of the examination process, facilitating effective monitoring and enforcement of exam regulations to maintain fairness and credibility. Its application in exam proctoring significantly enhances the system's ability to uphold standards and prevent cheating, thereby preserving the integrity of academic assessments.

4.2.2. Audio Analysis:

Audio analysis involves processing and examining audio data to extract meaningful insights. In online exam proctoring, custom algorithms can be developed to detect patterns indicative of cheating or unauthorized communication. This encompasses analyzing speech patterns, background noises, or unusual sounds that may suggest collaboration or unauthorized aid during exams.

By scrutinizing audio cues, the system can swiftly identify potential instances of cheating, thereby bolstering the integrity of online assessments and maintaining academic standards. Through advanced audio analysis techniques, the system can differentiate between normal exam-related sounds and suspicious audio patterns, such as whispers or external communication.

Additionally, the analysis can include voice recognition algorithms to verify the identity of the speaker and detect any deviations from the expected speech patterns of the authorized exam taker. Moreover, the system can employ machine learning models trained on labeled audio data to continuously improve its detection capabilities and adapt to new cheating methods or audio patterns. By integrating audio analysis into the proctoring system, institutions can enhance the effectiveness of their online exam monitoring efforts and ensure the fairness and credibility of the assessment process.

4.2.3. Video Tracking:

Video tracking techniques encompass monitoring and analyzing video data to track the movement and behavior of objects or individuals within a scene. In online exam proctoring, integrating video processing techniques enables tracking of students' facial expressions, eye movements, and overall engagement during exams. By scrutinizing these visual cues, the system can detect signs of distraction, disengagement, or suspicious behavior, aiding in the identification of cheating or unauthorized activity. This enhances the system's ability to uphold academic integrity and ensures fairness in online examinations.

Additionally, video tracking algorithms can be trained to recognize specific behaviors or actions associated with cheating, such as looking off-screen frequently, covering the face, or attempting to hide objects. By continuously analyzing video feeds in real-time, the system can promptly flag any irregularities and alert proctors or administrators for further investigation.

Moreover, the system can provide visual evidence of any detected cheating incidents, facilitating disciplinary actions and maintaining transparency in the examination process. Overall, video tracking enhances the effectiveness of online exam proctoring systems by providing comprehensive monitoring and detection capabilities, thereby safeguarding the integrity of academic assessments.

4.2.4. Tab Shifting Detection:

Tab shifting detection involves developing algorithms to monitor and log instances of tab switching or unauthorized resource access during exams. By tracking the user's browser activity and detecting changes in focus or attention, the system can identify potential use of unauthorized materials or resources. Logging tab switching behavior provides valuable insights for post-examination analysis, aiding in the detection of academic dishonesty and maintaining the credibility of online assessments.

Furthermore, advanced tab shifting detection algorithms can analyze the content of the switched tabs or browser windows to determine if they contain relevant exam-related information or prohibited resources. By correlating tab switching behavior with the timing of exam questions or tasks, the system can identify patterns indicative of cheating attempts and flag suspicious activities for further investigation.

Additionally, the system can generate detailed reports highlighting the frequency and duration of tab switches, as well as the content accessed during each switch, providing administrators with valuable data for assessing the integrity of online exams. Through robust tab shifting detection mechanisms, institutions can effectively deter cheating behavior and uphold the academic integrity of their online assessment processes.

4.2.5. Haar Cascade:

Haar Cascade is a machine learning-based object detection algorithm utilized to identify objects in images or video frames. By employing a set of trained classifiers to detect specific object features, such as edges or textures, Haar Cascade efficiently identifies objects. Commonly used for face detection, it analyzes facial features like eyes, nose, and mouth. Its efficiency and speed make it suitable for real-time applications like video surveillance and online exam proctoring, enhancing the system's capability to detect and prevent cheating during online exams. Haar Cascade operates by detecting patterns in the pixel values of grayscale images, allowing it to identify objects with high accuracy and minimal computational overhead.

Additionally, Haar Cascade models can be trained on large datasets of annotated images to improve their detection performance and adapt to different exam environments or lighting conditions. By integrating Haar Cascade into the proctoring system, institutions can enhance their ability to monitor exam sessions in real-time and promptly detect any unauthorized behavior or irregularities.

Furthermore, the use of Haar Cascade for face detection ensures that only authorized individuals are allowed to participate in online exams, mitigating the risk of impersonation or identity fraud. Overall, Haar Cascade provides an effective and efficient solution for object detection in online exam proctoring systems, helping institutions maintain the integrity and fairness of their assessment processes.

4.2.6. Structural Similarity Index (SSI):

The Structural Similarity Index (SSI) is a metric quantifying the likeness between two images. By comparing structural information including texture, luminance, and contrast, SSI determines image similarity. Valuable in image processing tasks necessitating image comparison, such as medical imaging or quality assessment, SSI is instrumental in detecting discrepancies or anomalies in online exam proctoring. It aids in comparing reference images (e.g., exam questions) with captured images, thus enhancing the system's capability to maintain academic integrity and identify irregularities during online exams.

The SSI computes the similarity between two images by measuring the structural differences in their pixel intensities, providing a quantitative measure of their resemblance. Additionally, the SSI can be utilized to detect alterations or modifications made to exam materials, such as unauthorized annotations or alterations to exam questions. By comparing the SSI values of reference and captured images, the system can identify any deviations from the expected image structure and flag suspicious activities for further investigation.

Moreover, the SSI can be integrated with other image analysis techniques, such as optical character recognition (OCR) or object detection, to enhance the system's capability to detect cheating behavior and maintain the integrity of online exams. Overall, the Structural Similarity Index serves as a valuable tool for assessing image similarity and identifying irregularities in online exam proctoring systems, enabling institutions to uphold academic integrity and ensure the fairness and credibility of their assessment processes.

4.2.7. Transformer:

The Transformer, originating from Vaswani et al.'s "Attention is All You Need" paper, is a deep learning model architecture primarily applied in natural language processing (NLP) tasks but also successfully in computer vision. Leveraging self-attention, it weighs the significance of different input tokens during output token generation, facilitating the capture of long-range dependencies in data. In online exam proctoring, Transformers excel in tasks like text recognition, question answering, or analyzing textual data from exam papers or student responses. Their ability to

comprehend complex language structures enhances the system's capacity to scrutinize exam content, ensuring fairness and accuracy in assessments.
Transformers can be employed for detecting plagiarism by comparing textual data from student responses against a database of known sources. Additionally, Transformers can aid in the automated grading of essay questions by analyzing the content, coherence, and relevance of student responses. By leveraging Transformers for text analysis tasks, institutions can streamline the assessment process, reduce manual grading efforts, and ensure consistency and fairness in evaluating student performance.

Overall, Transformers serve as versatile tools for analyzing textual data in online exam proctoring systems, enabling institutions to maintain academic integrity and ensure the accuracy and fairness of their assessments.

4.3 DATA LOADING AND PROCESSING

In our project, we utilized the COCO (Common Objects in Context) dataset for object training, enriching our AI-based proctoring exam system's capability to detect and classify various objects within the exam environment. The data loading process for the COCO dataset involved several key steps. Initially, we obtained the COCO dataset, which comprises a vast collection of images labeled with object annotations across 80 different categories. Next, we developed a data loading pipeline to efficiently ingest and preprocess the COCO dataset within our system.

This pipeline involved parsing the dataset's annotation files to extract information about object categories, bounding box coordinates, and image paths. Subsequently, we implemented mechanisms to load and augment the images, resizing them to fit the input dimensions of our object detection model while preserving aspect ratio and image quality.

Additionally, we applied data augmentation techniques such as random cropping, flipping, and color jittering to enhance the diversity and robustness of the training data. By meticulously managing the data loading process for the COCO dataset, we ensured that our object detection model was trained on a comprehensive and representative set of images, enabling it to accurately identify and classify objects of interest during exam sessions.

COCO DATASET:

Person	bench	banana
bicycle	bird	apple
car	cat	sandwich
motorcycle	dog	orange
airplane	horse	broccoli
bus	sheep	carrot
train	cow	hot dog
truck	elephant	pizza
boat	bear	donut
traffic light	zebra	cake
fire hydrant	giraffe	chair
stop sign	backpack	couch
parking meter	umbrella	potted plant
toothbrush	handbag	bed
baseball bat	tie	dining table
baseball glove	suitcase	toilet
skateboard	frisbee	tv
surfboard	skis	laptop
tennis racket	snowboard	mouse
bottle	sports ball	remote
wine glass	kite	keyboard
sink	cup	cell phone
refrigerator	fork	microwave
book	knife	oven
clock	spoon	toaster
vase	bowl	teddy bear
scissors	hair drier	

Table 4.3.1 Coco Dataset

Opening and loading coco dataset

```
my_file = open("utils/coco.txt", "r") # opening the file in read mode
data = my_file.read() # reading the file
class_list = data.split("\n") # replacing end splitting the text | when newline ('\n') is
seen.
my_file.close()
detected_things = []
detection_colors = [] # Generate random colors for class list
for i in range(len(class_list)):
    r = random.randint(0, 255)
    g = random.randint(0, 255)
    b = random.randint(0, 255)
    detection_colors.append((b, g, r))
model = YOLO("yolov8n.pt", "v8") # load a pretrained YOLOv8n model
EDFlag = False
```

4.4 MODEL DEVELOPMENT

In the development of our AI-based proctoring project, we leveraged a variety of programming languages, frameworks, and tools to create robust and effective machine learning models. Primarily, we utilized Python as our programming language due to its versatility, extensive libraries, and strong support for deep learning frameworks. For deep learning, we employed TensorFlow, PyTorch, and Keras, each chosen for their flexibility, performance, and rich set of pre-trained models suitable for tasks such as facial expression analysis, gaze detection, and object detection.

Our web framework of choice was Flask, a lightweight and easy-to-use framework for building web applications in Python. We integrated Flask with a MySQL database system, managed through Php My Admin, to store and retrieve data related to exams, users, and monitoring activities. On the front-end, we utilized HTML, CSS, JavaScript, and Bootstrap to create a user-friendly interface for instructors and students to interact with the proctoring system.

For deploying the web application, we relied on the Apache web server, which provided a stable and secure environment for hosting our Flask application. To manage dependencies and ensure reproducibility, we utilized Anaconda environments and virtualenv for creating isolated development environments.

In terms of model development, we utilized the COCO dataset for training our object detection models, leveraging the rich annotations and diverse range of object categories present in the dataset. Using TensorFlow, PyTorch, or Keras, we trained and fine-tuned our models to accurately detect and classify objects within exam environments, such as unauthorized materials or individuals.

Throughout the development process, we utilized Visual Studio Code as our integrated development environment (IDE), benefiting from its intuitive interface, extensive plugin ecosystem, and support for Python development. Additionally, we adhered to secure communication protocols such as HTTPS to ensure the confidentiality and integrity of data transmitted between clients and the server.

By integrating these technologies and tools into our AI-based proctoring project, we were able to develop a comprehensive solution capable of effectively monitoring exam sessions, detecting anomalies, and upholding academic integrity in online learning environments.

FUNCTIONALITIES

1. MOBILE PHONE DETECTION

```
def electronicDevicesDetection(frame):
    global model, EDFlag
    # Predict on image
    detect_params = model.predict(source=[frame], conf=0.45, save=False)
    # Convert tensor array to numpy
    DP = detect_params[0].numpy()
    for result in detect_params:  # iterate results
        boxes = result.boxes.cpu().numpy()  # get boxes on cpu in numpy
        for box in boxes:  # iterate boxes
            r = box.xyxy[0].astype(int)  # get corner points as int
            detected_obj = result.names[int(box.cls[0])]
            if (detected_obj == 'cell phone' or detected_obj == 'remote' or detected_obj ==
'laptop' or detected_obj == 'laptop,book'): EDFlag = True
    textED = ''
    # Display the resulting frame
    if EDFlag:
        textED = 'Electronic Device Detected'
    else:
        textED = "No Electronic Device Detected"
    EDD_record_duration(textED, frame)
    print(textED)
    EDFlag = False
```

2. VOICE DETECTION

```
class Recorder:
    @staticmethod
    def rms(frame):
        count = len(frame) / SHORT_WIDTH
        format = "%dh" % (count)
        shorts = struct.unpack(format, frame)
        sum_squares = 0.0
        for sample in shorts:
            n = sample   SHORT_NORMALIZE
            sum_squares += n   n
        rms = math.pow(sum_squares / count, 0.5)

        return rms   1000
    def __init__(self):
        self.p = pyaudio.PyAudio()
        self.stream = self.p.open(format=FORMAT,channels=CHANNELS,
                          rate=RATE,input=True,output=True,
                          frames_per_buffer=CHUNK)
        self.time = time.time()
        self.quiet = []
        self.quiet_idx = -1
        self.timeout = 0
    def record(self):
        global Globalflag
        print('')
        print(f'Voice Flag is {Globalflag}')
        sound = []
        start = time.time()
        begin_time = None
        while Globalflag:
            data = self.stream.read(CHUNK)
            rms_val = self.rms(data)
            if self.inSound(data):
                sound.append(data)
                if begin_time == None:
                    begin_time = datetime.datetime.now()
            else:
                if len(sound) > 0:
                    duration=math.floor((datetime.datetime.now()-
begin_time).total_seconds())
                    self.write(sound, begin_time, duration)
                    sound.clear()
                    begin_time = None
                else:
                    self.queueQuiet(data)
            curr = time.time()
            secs = int(curr - start)
            tout = 0 if self.timeout == 0 else int(self.timeout - curr)
            label = 'Listening' if self.timeout == 0 else 'Recording'
            print('[+] %s: Level=[%4.2f] Secs=[%d] Timeout=[%d]' % (label, rms_val,
secs, tout), end='\r')
```

3. MULTIPERSON DETECTION

```
def MTOP_Detection(img):
    print("Running MTOP Function")
    imgRGB = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    results = faceDetection.process(imgRGB)
    textMTOP = ''
    if results.detections:
        for id, detection in enumerate(results.detections):
            bboxC = detection.location_data.relative_bounding_box
            ih, iw, ic = img.shape
            bbox = int(bboxC.xmin   iw), int(bboxC.ymin   ih), \
                int(bboxC.width   iw), int(bboxC.height   ih)
            # Drawing the recantangle
            cv2.rectangle(img, bbox, (255, 0, 255), 2)
            # cv2.putText(img, f'{int(detection.score[0]   100)}%', (bbox[0], bbox[1] - 20),
cv2.FONT_HERSHEY_PLAIN, 3, (255, 255, 255), 10)
        if id > 0:
            textMTOP = "More than one person is detected."
        else:
            textMTOP = "Only one person is detected"
    else:
        textMTOP="Only one person is detected"
    MTOP_record_duration(textMTOP, img)
    print(textMTOP)
```

4. SHORCUT KEYS:

```
def shortcut_handler(event):
    if event.event_type == keyboard.KEY_DOWN:
        shortcut = ''
        # Check for Ctrl+C
        if keyboard.is_pressed('ctrl') and keyboard.is_pressed('c'):
            shortcut += 'Ctrl+C'
            print("Ctrl+C shortcut detected!")
        # Check for Ctrl+V
        elif keyboard.is_pressed('ctrl') and keyboard.is_pressed('v'):
            shortcut += 'Ctrl+V'
            print("Ctrl+V shortcut detected!")
        # Check for Ctrl+A
        elif keyboard.is_pressed('ctrl') and keyboard.is_pressed('a'):
            shortcut += 'Ctrl+A'
            print("Ctrl+A shortcut detected!")
        # Check for Ctrl+X
        elif keyboard.is_pressed('ctrl') and keyboard.is_pressed('x'):
            shortcut += 'Ctrl+X'
            print("Ctrl+X shortcut detected!")
        # Check for Alt+Shift+Tab
        elif keyboard.is_pressed('alt') and keyboard.is_pressed('shift') and
keyboard.is_pressed('tab'):
            shortcut += 'Alt+Shift+Tab'
            print("Alt+Shift+Tab shortcut detected!")
```

4.5 USE CASE DIAGRAM

Use Case Diagram 1: Teacher Perspective

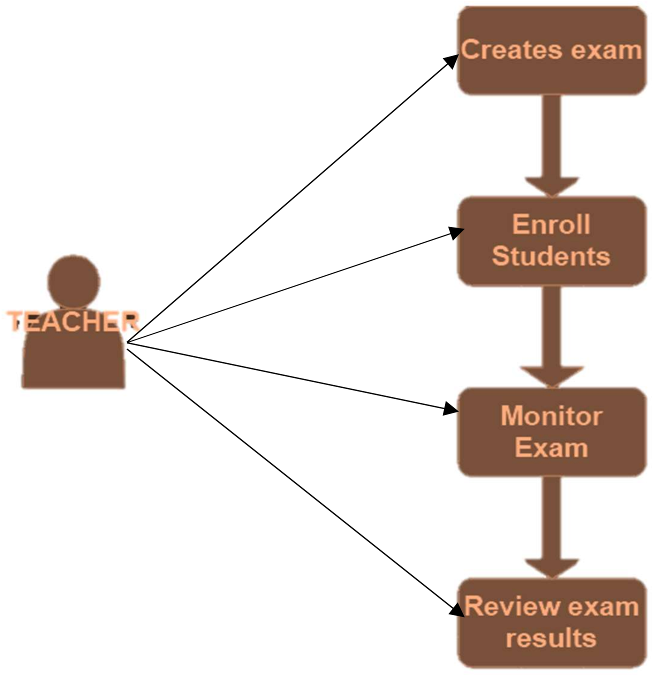


Fig 4.5.1: Teacher Use Case diagram

Actors:

Teacher: The instructor who creates and administers exams, monitors exam sessions, and reviews exam results.

Use Cases:

1. **Create Exam:** The teacher creates a new exam, specifying parameters such as duration, permitted resources, and question formats.

```
#Flak's Application Confguration
warnings.filterwarnings("ignore")
app = Flask(__name__, template_folder='templates', static_folder='static')
app.secret_key = 'xyz'
# app.config["MONGO_URI"] = "mongodb://localhost:27017/"
os.path.dirname("../templates")

#Flak's Database Configuration
app.config['MYSQL_HOST'] = 'bhsjmidfvh9on3qjt44o-mysql.services.clever-
cloud.com'
app.config['MYSQL_USER'] = 'usvbdhqgh3xze3e5'
app.config['MYSQL_PASSWORD'] = 'kOU7sIPapX57pun5YAJt'
app.config['MYSQL_DB'] = 'bhsjmidfvh9on3qjt44o'
mysql = MySQL(app)
```

2. **Enroll Student**: The teacher enrolls students in exams and assigns exams to specific student cohorts or classes.

```
@app.route('/adminStudents')
def adminStudents():
    cur = mysql.connection.cursor()
    cur.execute("SELECT   FROM students where Role='STUDENT'")
    data = cur.fetchall()
    cur.close()
    return render_template('Students.html', students=data)

@app.route('/insertStudent', methods=['POST'])
def insertStudent():
    if request.method == "POST":
        name = request.form['username']
        email = request.form['email']
        password = request.form['password']
        cur = mysql.connection.cursor()
        cur.execute("INSERT INTO students (Name, Email, Password, Role) VALUES
(%s, %s, %s, %s)", (name, email, password,'STUDENT'))
        mysql.connection.commit()
        return redirect(url_for('adminStudents'))

@app.route('/deleteStudent/<string:stdId>', methods=['GET'])
def deleteStudent(stdId):
    flash("Record Has Been Deleted Successfully")
    cur = mysql.connection.cursor()
    cur.execute("DELETE FROM students WHERE ID=%s", (stdId,))
    mysql.connection.commit()
    return redirect(url_for('adminStudents'))

@app.route('/updateStudent', methods=['POST', 'GET'])
def updateStudent():
    if request.method == 'POST':
        id_data = request.form['id']
        name = request.form['name']
        email = request.form['email']
        password = request.form['password']
        cur = mysql.connection.cursor()
        cur.execute("""
               UPDATE students
               SET Name=%s, Email=%s, Password=%s
               WHERE ID=%s
            """, (name, email, password, id_data))
        mysql.connection.commit()
        return redirect(url_for('adminStudents'))
```

DATABASE CREATION:

```
SET SQL_MODE = "NO_AUTO_VALUE_ON_ZERO";
START TRANSACTION;
SET time_zone = "+00:00";
/*!40101 SET
@OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET
@OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET
@OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8mb4 */;

--
-- Database: `bihd3znf0z6idbuelgex`
-- Table structure for table `students`
CREATE TABLE `students` (
  `ID` int NOT NULL,
  `Name` varchar(255) NOT NULL,
  `Email` varchar(255) NOT NULL,
  `Password` varchar(255) NOT NULL,
  `Role` enum('STUDENT','ADMIN') NOT NULL DEFAULT 'STUDENT'
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
-- Dumping data for table `students`
INSERT INTO `students` (`ID`, `Name`, `Email`, `Password`, `Role`) VALUES
(1, 'pavan', 'pavan@gmail.com', '1234', 'ADMIN'),
(3, 'harshitha`', 'harsh@gmail.com', '1234', 'STUDENT'),
(4, 'Varun', 'varun@gmail.com', '1234', 'STUDENT');

-- Indexes for dumped tables
-- Indexes for table `students`
ALTER TABLE `students`
  ADD PRIMARY KEY (`ID`);

-- AUTO_INCREMENT for dumped tables
-- AUTO_INCREMENT for table `students`
ALTER TABLE `students`
  MODIFY `ID` int NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=5;
COMMIT;
```

3. **Monitor Exam:** During exam sessions, the teacher monitors students' activities in real-time, including video feeds, audio analysis, and tab shifting behavior.

4. **Review Exam Results**: After exams are completed, the teacher reviews exam results, analyzes individual student performances, and identifies irregularities or suspicious behavior.

```
def cheat_Detection1():
    deleteTrashVideos()
    global Globalflag
    mp_face_mesh = mp.solutions.face_mesh
    face_mesh = mp_face_mesh.FaceMesh(min_detection_confidence=0.5,
min_tracking_confidence=0.5)
    print(f'CD1 Flag is {Globalflag}')
    while Globalflag:
        success, image = cap.read()
        headMovmentDetection(image, face_mesh)
    if Globalflag:
        cap.release()
    deleteTrashVideos()

def cheat_Detection2():
    global Globalflag, shorcuts
    print(f'CD2 Flag is {Globalflag}')
    deleteTrashVideos()
    while Globalflag:
        success, image = cap.read()
        image1 = image
        image2 = image
        MTOP_Detection(image1)
        screenDetection()
    deleteTrashVideos()
    if Globalflag:
        cap.release()
def getResultDetails(rid):
    with open('result.json', 'r+') as file:
        # First we load existing data into a dict.
        result_data = json.load(file)
        filtered_result = [item for item in result_data if item["Id"] == int(rid)]
    with open('violation.json', 'r+') as file:
        # First we load existing data into a dict.
        violation_data = json.load(file)
        filtered_violations = [item for item in violation_data if item["RId"] == int(rid)]
    resultDetails = {
            "Result": filtered_result,
            "Violation": filtered_violations
        }
    return resultDetails
```

Saving Results into database for admin to track them

```
CREATE TABLE IF NOT EXISTS `results` (
  `ID` int NOT NULL AUTO_INCREMENT,
  `StudentID` int NOT NULL,
  `TotalMark` float NOT NULL,
  `TrustScore` int NOT NULL,
  `Status` enum('Pass','Fail','Fail(Cheating)') NOT NULL,
  `Date` date NOT NULL,
  `ProfileLink` varchar(255) NOT NULL,
  PRIMARY KEY (`ID`),
  KEY `StudentID` (`StudentID`),
  CONSTRAINT `results_ibfk_1` FOREIGN KEY (`StudentID`) REFERENCES
`students` (`ID`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

CREATE TABLE IF NOT EXISTS `students` (
  `ID` int NOT NULL AUTO_INCREMENT,
  `Name` varchar(255) NOT NULL,
  `Email` varchar(255) NOT NULL,
  `Password` varchar(255) NOT NULL,
  `Role` enum('STUDENT','ADMIN') NOT NULL DEFAULT 'STUDENT',
 PRIMARY KEY (`ID`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

Use Case Diagram 2: Student Perspective

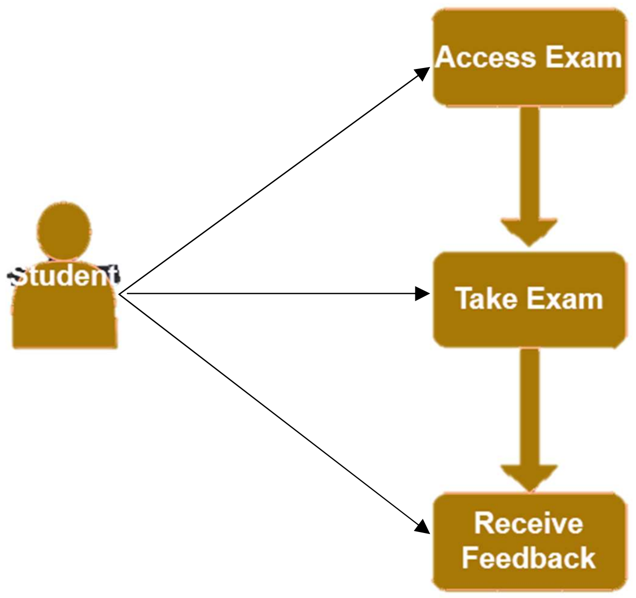


Fig 4.5.2: Student Use Case diagram

Actors:

1. Student: The exam participant who accesses assigned exams, takes exams, and receives feedback.

Use Cases:

1**. Login & Access Exam:** The student gains access to assigned exams via the platform, where detailed instructions and guidelines for the examination process are provided.

```
#Login Related
@app.route('/')
def main():
   return render_template('login.html')

@app.route('/login', methods=['POST'])
```

```
def login():
    global studentInfo
    if request.method == 'POST':
        username = request.form['username']
        password = request.form['password']
        cur = mysql.connection.cursor()
        cur.execute("SELECT   FROM students where Email='" + username + "' and
Password='" + password + "'")
        data = cur.fetchone()
        if data is None:
            flash('Your Email or Password is incorrect, try again.', category='error')
            return redirect(url_for('main'))
        else:
            id, name, email,password, role = data
            studentInfo={ "Id": id, "Name": name, "Email": email, "Password": password}
            if role == 'STUDENT':
                utils.Student_Name = name
                return redirect(url_for('rules'))
            else:
                return redirect(url_for('adminStudents'))

@app.route('/logout')
def logout():
    return render_template('login.html')
```

2. Take Exam:

The student undertakes exams within the designated timeframe, with the system rigorously monitoring their webcam, microsphone, and browser activity.

```
@app.route('/exam')
def exam():
    utils.cap = cv2.VideoCapture(0, cv2.CAP_DSHOW)
    keyboard.hook(utils.shortcut_handler)
    return render_template('Exam.html')

@app.route('/exam', methods=["POST"])
def examAction():
    link = ''
    if request.method == 'POST':
        examData = request.json
        if(examData['input']!=''):
            utils.Globalflag= False
            utils.cap.release()
            utils.write_json({
                "Name": ('Prohibited Shorcuts (' + ','.join(list(dict.fromkeys(utils.shorcuts)))
+ ') are detected.'),
                "Time": (str(len(utils.shorcuts)) + " Counts"),
                "Duration": '',
                "Mark": (1.5   len(utils.shorcuts)),
                "Link": '',
                "RId": utils.get_resultId()
            })
            utils.shorcuts=[]
            trustScore= utils.get_TrustScore(utils.get_resultId())
            totalMark=  math.floor(float(examData['input'])  6.6667)
            if trustScore >=30:
                status="Fail(Cheating)"
                link = 'showResultFail'
            else:
                if totalMark < 50:
                    status="Fail"
                    link = 'showResultFail'
                else:
                    status="Pass"
                    link = 'showResultPass'
            utils.write_json({
                "Id": utils.get_resultId(),
                "Name": studentInfo['Name'],
                "TotalMark": totalMark,
                "TrustScore": max(100-trustScore, 0),
                "Status": status,
                "Date": time.strftime("%Y-%m-%d", time.localtime(time.time())),
                "StId": studentInfo['Id'],
                "Link" : profileName
            },"result.json")
            resultStatus=
studentInfo['Name']+';'+str(totalMark)+';'+status+';'+time.strftime("%Y-%m-%d",
time.localtime(time.time()))
        else:
            utils.Globalflag = True
            print('sfdsfsdsfdsfdsfdsfdsfdsfdsfds')
            resultStatus=''
    return jsonify({"output": resultStatus, "link": link})

@app.route('/showResultPass/<result_status>')
def showResultPass(result_status):
    return render_template('ExamResultPass.html',result_status=result_status)

@app.route('/showResultFail/<result_status>')
def showResultFail(result_status):
    return render_template('ExamResultFail.html',result_status=result_status)
```

3. Receive Feedback: Upon completion of the exam, the student receives personalized feedback and detailed exam results to facilitate learning and improvement.

These use case diagrams provide a high-level overview of the system's functionalities from the perspectives of both teachers and students, outlining the interactions between actors and the system.

5. RESULT

The Online Exam Proctor" has been implemented with remarkable success, significantly bolstering the security of online exams. Its real-time video monitoring feature has proven highly effective in identifying suspicious behaviors, such as attempts to access unauthorized resources or divert attention from the screen. Moreover, the integration of facial recognition technology has provided an additional layer of security by accurately verifying the identities of test takers, thus reducing the risk of impersonation and ensuring a fair assessment process.

In terms of deterring cheating, "The Online Exam Proctor" has had a profound impact. The mere knowledge that their actions are being closely monitored has dissuaded test takers from engaging in dishonest behavior. The software's keystroke analysis feature has further strengthened its ability to detect irregular typing patterns, enabling administrators to promptly intervene and maintain the integrity of the exam environment in real-time.

Administrators have experienced a significant reduction in workload due to the streamlined exam administration facilitated by "The Online Exam Proctor." Automated detection of suspicious activities has alleviated the need for manual monitoring, allowing administrators to focus their attention on specific cases that require intervention.

In conclusion, "The Online Exam Proctor" has emerged as a highly effective solution for enhancing exam security, deterring cheating, and streamlining exam administration processes. Its success is evident in the positive feedback from users and its widespread adoption across various educational and professional settings. Moving forward, continued improvements and updates will ensure that it remains at the forefront of addressing the evolving challenges of online exam proctoring.

STUDENT ACTIVITY SCREENSHORTS

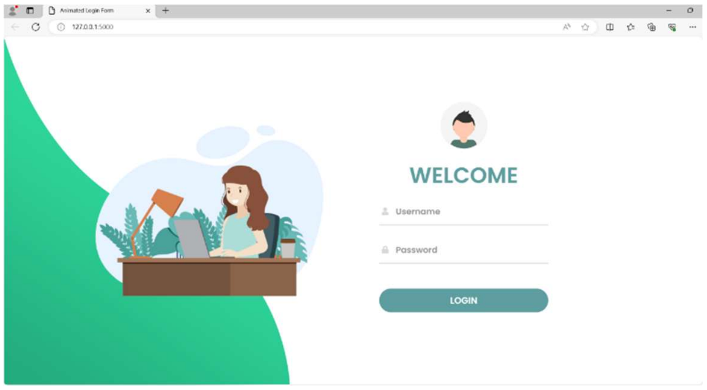


Fig 5.1: Login page

The image is a graphical user interface displaying a login screen. It includes fields for entering a username and password, along with a "LOGIN" button. The design appears to be simple and cartoonish. The login page is the gateway for students and administrators to enter the online exam system, providing access through individual credentials. This authentication process ensures the security and integrity of the exam platform. Student credentials are meticulously generated and distributed by the administrator, guaranteeing that only authorized individuals can log in and participate in exams. This process establishes accountability and traceability, as each user's actions within the system can be attributed to their unique login credentials. Furthermore, the login page sets the tone for a user-centric experience, offering a familiar and intuitive interface for accessing exam-related functionalities. Overall, the login page plays a pivotal role in facilitating secure access to the exam platform for both students and administrators, laying the foundation for a seamless and trustworthy online examination process.

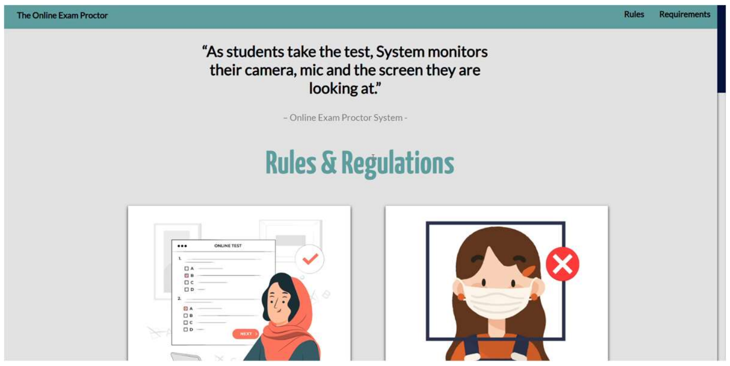
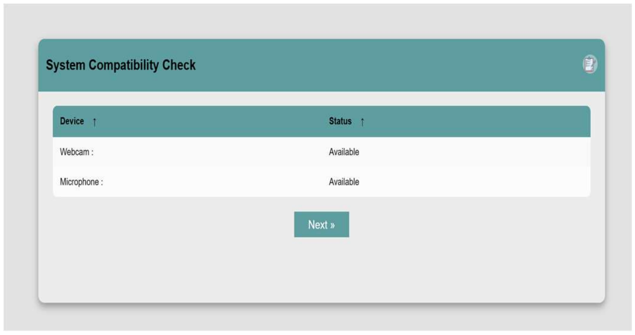

The Online Exam Proctor
Rules
Requirements
"As students take the test, System monitors their camera, mic and the screen they are looking at."
– Online Exam Proctor System –
Rules & Regulations
ONLINE TEST

Fig 5.2: Rules and regulations

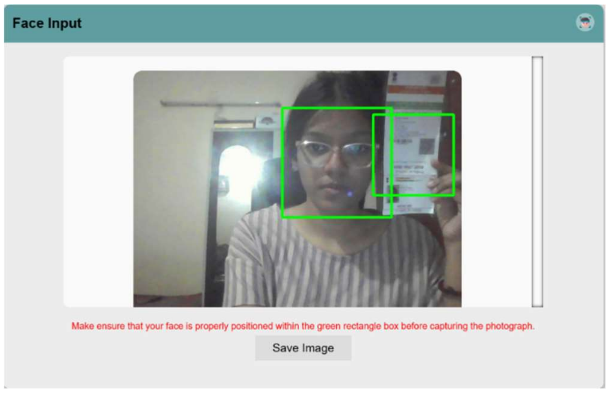
After successfully logging into the system, users are guided to a rules and regulations page aimed at providing clear guidelines to ensure an optimal testing environment. These guidelines are crucial for maintaining fairness and integrity throughout the examination process. Instructions such as wearing masks, maintaining a quiet environment, and ensuring proper lighting are emphasized to create an atmosphere conducive to focused and undisturbed testing. By stipulating these requirements upfront, the system aims to mitigate potential distractions and enhance concentration levels among test-takers. Wearing masks not only adheres to health and safety protocols but also helps in minimizing any facial expressions or gestures that could be interpreted as cheating behaviors. Additionally, the emphasis on maintaining a quiet environment underscores the importance of minimizing external disturbances that could disrupt the testing process. Proper lighting ensures clarity in viewing exam materials and prevents any visual strain on the students.


System Compatibility Check

Device ↑	Status ↑
Webcam :	Available
Microphone :	Available

Fig 5.3: Compatibility Check

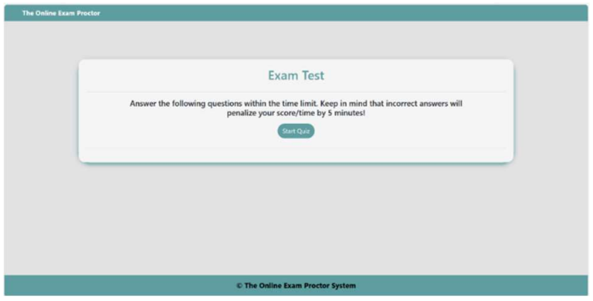
User system compatibility is a critical aspect verified before students can proceed with the exam. The system mandates that students' devices have both a functioning microphone and camera to ensure comprehensive exam monitoring. This verification process is essential for maintaining the integrity and security of the online exam environment. Without access to these components, students are unable to attempt the exam, thus preventing any potential loopholes or unfair advantages. Moreover, this verification step serves to uphold the authenticity of the examination process, as it minimizes the risk of students attempting to circumvent monitoring measures. Overall, verifying user system compatibility is crucial in establishing a robust and reliable online exam system that prioritizes integrity and fairness.



Make ensure that your face is properly positioned within the green rectangle box before capturing the photograph.

Fig 5.4: User Face Input

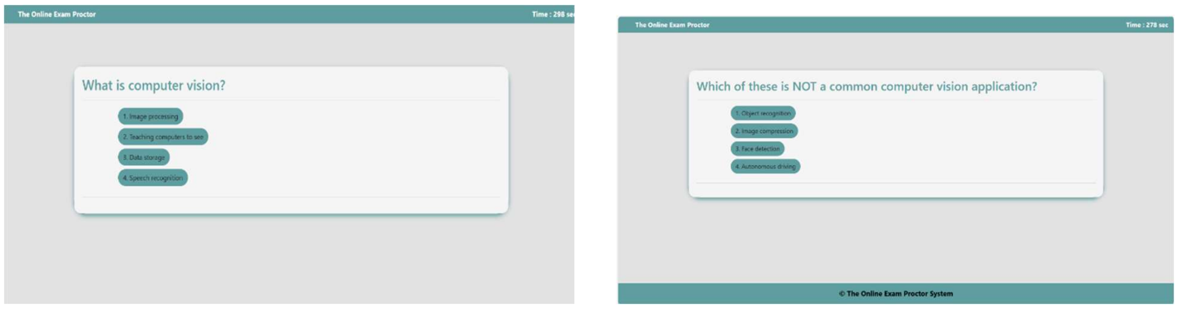
Prior to commencing the examination, the system captures the student's facial input, marking the initiation of comprehensive facial tracking throughout the entire duration of the exam. This proactive measure serves as a pivotal component of the exam proctoring process, enabling continuous monitoring of the student's facial activity. By tracking facial expressions, movements, and engagement levels, the system can detect any irregularities or signs of potential misconduct in real-time. This includes detecting behaviors such as looking away from the screen, covering the face, or displaying signs of distress, which may indicate cheating or unauthorized assistance. The captured facial data provides valuable insights into the student's behavior and attentiveness during the exam, facilitating the identification and mitigation of any breaches in academic integrity. Overall, the integration of facial tracking technology enhances the system's ability to uphold the integrity of online examinations by ensuring a fair and secure testing environment.



Answer the following questions within the time limit. Keep in mind that incorrect answers will penalize your score/time by 5 minutes!

Fig 5.5: Exam Start Page

The image is a screenshot of a graphical user interface displaying an application called "The Online Exam Proctor." It prompts the user to answer questions within a time limit and warns that incorrect answers will result in a penalty to score/time. The interface includes text and buttons for starting the quiz.


The Online Exam Proctor Time : 298 se

What is computer vision?

1. Image processing
2. Teaching computers to see
3. Data storage
4. Speech recognition

The Online Exam Proctor Time : 278 sec

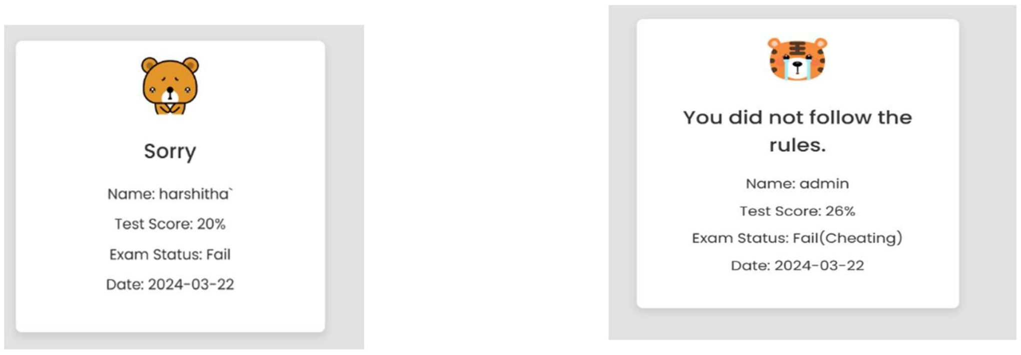
Which of these is NOT a common computer vision application?

1. Object recognition
2. Image compression
3. Face detection
4. Autonomous driving

© The Online Exam Proctor System

Fig 5.6: Questions

The image is of graphical user interface displaying text related to computer vision. It seems to be an application or a web page with a rectangle design. The content includes a question asking about common computer vision applications. The interface which displays the list of questions for the examination and respective options.


Sorry

Name: harshitha`

Test Score: 20%

Exam Status: Fail

Date: 2024-03-22

You did not follow the rules.

Name: admin

Test Score: 26%

Exam Status: Fail(Cheating)

Date: 2024-03-22

Fig 5.7: Exam Result

The graphical user interface displaying information about an online exam proctor. It shows details such as the user's name, test score, exam status, and the date of the exam. The interface also includes options to logout and displays a message saying "Sorry.". Hence it displays the result of the examination.

ADMIN ACTIVITY SCREENSHORTS

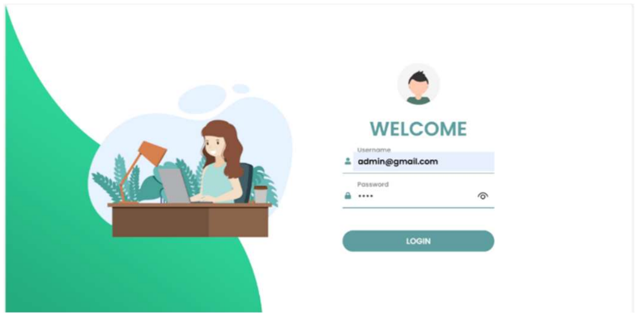
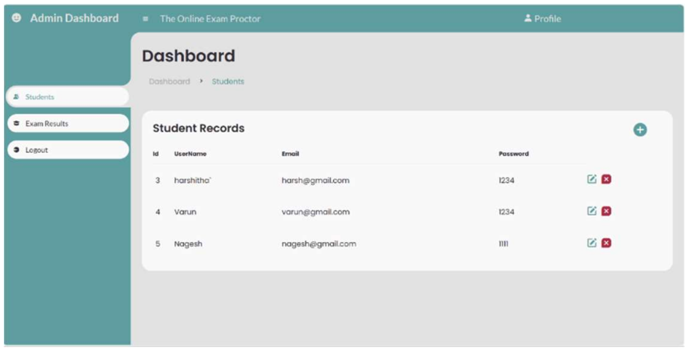


Fig 5.8: Admin Login Credentials

The image is a graphical user interface displaying a login screen. It includes fields for entering a username and password, along with a "LOGIN" button. The design appears to be simple and cartoonish. Following the exam, administrators have access to a comprehensive list of students who attempted the test, along with their corresponding status indicators: pass, fail, or suspected cheating. Additionally, the system provides administrators with the percentage of correct answers for each student, aiding in performance evaluation. Administrators can further investigate instances of suspected cheating by accessing live feed recordings, which capture students' activities during the exam. This feature enables administrators to verify and analyze suspicious behavior, such as unauthorized assistance or use of prohibited materials, in real-time. By leveraging live feed detection, administrators can effectively maintain exam integrity and ensure fairness in assessments. This functionality empowers administrators to take prompt and appropriate actions to address instances of academic dishonesty, ultimately upholding the credibility and standards of the online exam proctoring system.

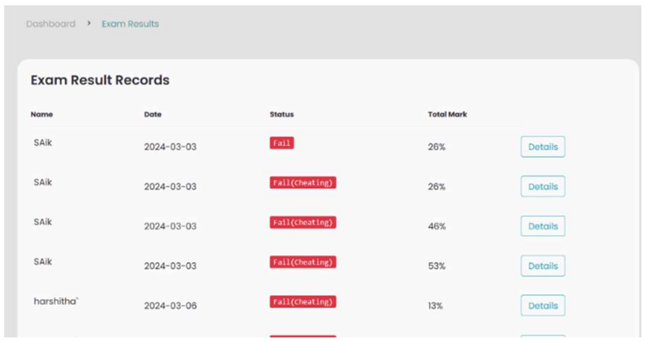


Student Records

Id	UserName	Email	Password
3	harshitha`	harsh@gmail.com	1234
4	Varun	varun@gmail.com	1234
5	Nagesh	nagesh@gmail.com	1111

Fig 5.9: Admin Dashboard – student database

The image shows an online exam proctor's dashboard. The dashboard has three main sections: students, exam results, and logout. The students section shows a list of students who have registered for the exam. The exam results section shows the results of the exams that have been conducted. The logout section allows the proctor to log out of the system.

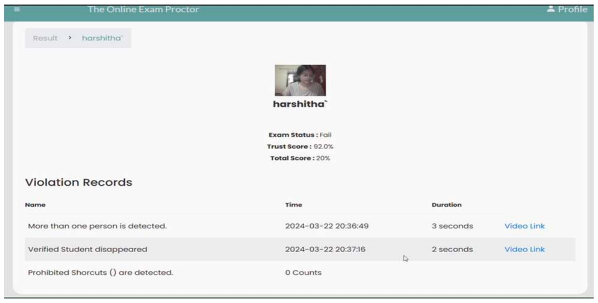


Exam Result Records

Name	Date	Status	Total Mark	
SAik	2024-03-03	Fail	26%	Details
SAik	2024-03-03	Fail(Cheating)	26%	Details
SAik	2024-03-03	Fail(Cheating)	46%	Details
SAik	2024-03-03	Fail(Cheating)	53%	Details
harshitha`	2024-03-06	Fail(Cheating)	13%	Details

Fig 5.10: Exam Results Records

The image presents an exam result record, providing essential details about the student's performance. It includes the student's name, allowing for easy identification, and the date of the exam, offering context regarding when the assessment occurred. The status of the exam indicates whether the student passed, failed, or achieved a different outcome, providing immediate insight into their performance. Additionally, the total mark obtained by the student offers a quantitative measure of their achievement in the exam, allowing for precise assessment and comparison. This comprehensive presentation of exam results enables administrators and educators to efficiently review and analyze student performance, identify trends, and make informed decisions regarding further actions or interventions. Overall, the exam result record serves as a valuable tool for tracking student progress and maintaining accountability within the educational assessment process.


The Online Exam Proctor

Profile

Result › harshitha`

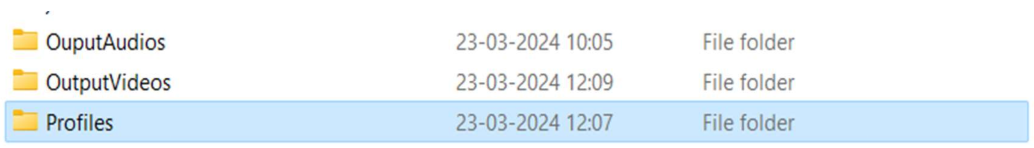
harshitha`

Exam Status : Fail
Trust Score : 92.0%
Total Score : 20%

Violation Records

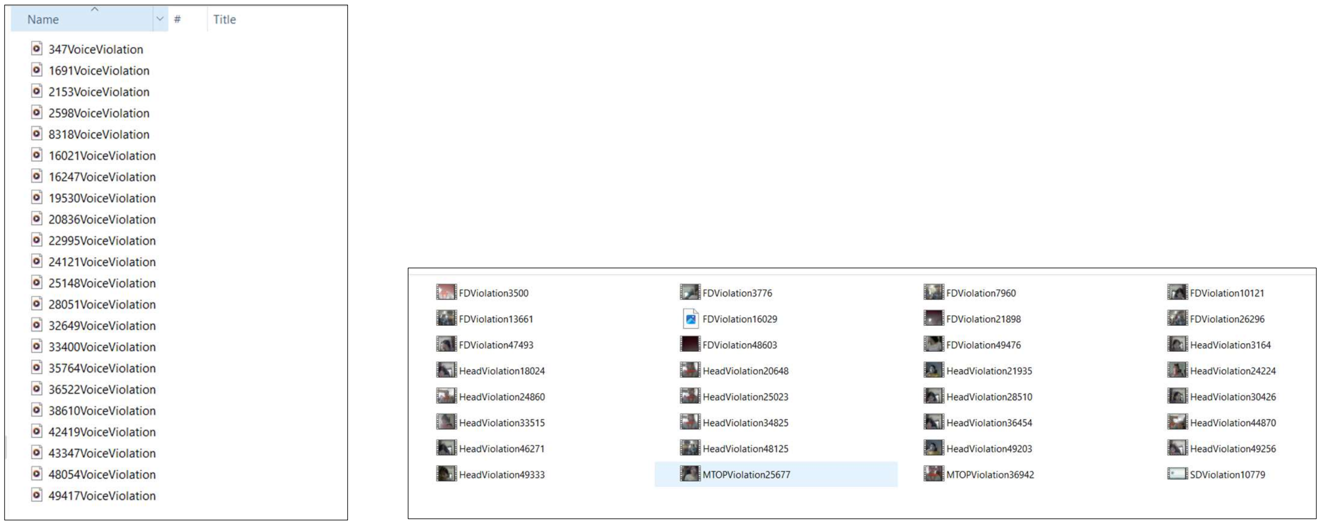
Name	Time	Duration	
More than one person is detected.	2024-03-22 20:36:49	3 seconds	Video Link
Verified Student disappeared	2024-03-22 20:37:16	2 seconds	Video Link
Prohibited Shorcuts () are detected.	0 Counts		

Fig 5.11: Student Violation Records



OuputAudios	23-03-2024 10:05	File folder
OutputVideos	23-03-2024 12:09	File folder
Profiles	23-03-2024 12:07	File folder

Fig 5.12: Outputs in Admin's system



Name	#	Title
347VoiceViolation		
1691VoiceViolation		
2153VoiceViolation		
2598VoiceViolation		
8318VoiceViolation		
16021VoiceViolation		
16247VoiceViolation		
19530VoiceViolation		
20836VoiceViolation		
22995VoiceViolation		
24121VoiceViolation		
25148VoiceViolation		
28051VoiceViolation		
32649VoiceViolation		
33400VoiceViolation		
35764VoiceViolation		
36522VoiceViolation		
38610VoiceViolation		
42419VoiceViolation		
43347VoiceViolation		
48054VoiceViolation		
49417VoiceViolation		

FDViolation3500	FDViolation3776	FDViolation7960	FDViolation10121
FDViolation13661	FDViolation16029	FDViolation21898	FDViolation26296
FDViolation47493	FDViolation48603	FDViolation49476	HeadViolation3164
HeadViolation18024	HeadViolation20648	HeadViolation21935	HeadViolation24224
HeadViolation24860	HeadViolation25023	HeadViolation28510	HeadViolation30426
HeadViolation33515	HeadViolation34825	HeadViolation36454	HeadViolation44870
HeadViolation46271	HeadViolation48125	HeadViolation49203	HeadViolation49256
HeadViolation49333	MTOPViolation25677	MTOPViolation36942	SDViolation10779

Fig 5.13: Violation audios &Videos in Admin's system

	Input	Description	Expected Output	Actual Output	Status
1	Student login With credentials set by admin	Test Case passed, control transferred to next page	Login should be Successfully completed	User Login	Pass
2	Student login With wrong credentials set by user	Test Case passed	An alert message	An alert message asking to provide correct user name/password	Pass
3	Rules and regulations page is displayed consistently and at end asking to start the exam	Test Case passed	The user should go through all the rules and click on ‘start test’ redirecting to exam window	The user is compatible with the page and can be redirected to exam window	Pass
4	Computability check of microphone and camera	Test Case passed	The user should give access to the mic and camera	Student can do this action	Pass
5	microphone and camera- if either of it is non-compatible	Test Case passed	The user is not permitted to write the exam	The user is not permitted to write the exam	Pass
6	Capturing face input of the student for validation	Test Case passed Image is successfully captured and saved to admin profiles folder	User should click on capture button while being in the green bouning box	Face input successfully captured and saved and redirected to questions	Pass
7	Capturing video input of user while attempting the exam	Test Case passed live video is successfully captured for detecting suspicious activites	live video is successfully captured for detecting suspicious activites	live video is successfully captured for detecting suspicious activites	Pass
8	Head movement	Test Case Passed- if student ties to move there head away from screen is detected	Head movement detected and saved	Head movement detected and saved	Pass
9	Tab switching	If any tab or window is open it is recorded	If any tab or window is open it is recorded and saved into admin output videos	If any tab or window is open it is recorded and saved into admin output videos	Pass

10	Ctr+c, ctrl+v alt, shift	If any keyboard key is pressed it is counted	keyboard key is pressed it is counted successfully	keyboard key is pressed it is counted successfully	Pass
11	Multi-person detection	If anyother person other than student appears to camera is recorded and marked as suspicious	Other person is detected and marked as suspicious successfully and saved there activity to admins folder	Other person is detected and marked as suspicious successfully and saved there activity to admins folder	Pass
12	Object detection	Any objects other than face is detected and flagged	Objects such as phone, books, papers should be detected	Objects such as phone, books, papers should be detected	Pass
13	Voice detection	Voice in background can be of person, TV, music should be detected	Any background sound is detected and saved	Any background sound is detected and saved	Pass

Table 5.1 Functionalities and status of student module

6. CONCLUSION

- Our project on the AI-based proctoring exam system represents a significant advancement in addressing the challenges of remote education and online examination platforms.
- By integrating cutting-edge technologies such as YOLO for real-time object detection, audio analysis for gaze detection, and video tracking, we have created a robust system that ensures exam integrity and security.
- The architecture of the system, built on Django, provides a scalable and user-friendly platform for both teachers and students, enabling seamless exam creation, administration, and monitoring.
- Through features like live monitoring, result review, and personalized feedback, instructors can effectively manage exam sessions and analyze student performances, while students benefit from a secure and fair examination environment.
- Moreover, the integration of AI technologies strengthens the system's monitoring capabilities, while robust security and privacy measures safeguard sensitive information. Our rigorous testing methodologies further validate the reliability, scalability, and effectiveness of the system in real-world scenarios.
- Overall, our project represents a comprehensive and innovative solution to the evolving landscape of remote education and online examination platforms, offering a transformative approach to ensuring academic integrity and student success.

7. FUTURE ENHANCEMENT

In the future development of our AI-based proctoring exam system, several key enhancements are envisaged to elevate its effectiveness and user experience. Expanding the AI algorithms to include sophisticated behavioral analysis would enable us to detect subtle cues like changes in typing patterns and body language, providing deeper insights into student engagement and potential cheating behaviors. Additionally, improving object detection capabilities through advanced models or fine-tuning existing ones would allow for more accurate identification of unauthorized materials or individuals during exams. Integrating natural language processing techniques could further enhance the system's capabilities by analyzing text-based responses for plagiarism detection and assessing answer quality. Furthermore, implementing accessibility features to accommodate students with disabilities, enhancing security measures, and integrating with existing learning management systems would enhance the overall usability, security, and functionality of the platform. Finally, refining feedback mechanisms and analytics tools would provide valuable insights for both educators and students, facilitating personalized learning experiences and continuous improvement. These future enhancements aim to solidify the system's position as a reliable, effective, and inclusive solution for online examination proctoring.

8. BIBLIOGRAPHY

[1] S. Y. Kim, H. G. Han, J. W. Kim, S. Lee, and T. W. Kim, “A hand gesture recognition sensor using reflected impulses,” IEEE Sens. J., vol. 17, no. 10, pp. 2975–2976, 2017, doi:10.1109/JSEN.2017.2679220.

[2] A. Kumar and A. Kumar, “Dog Breed Classifier for Facial Recognition using Convolutional Neural Networks,” pp. 508–513, 2020.

[3] J. Wu, L. Sun, and R. Jafari, “A Wearable System for Recognizing American Sign Language in RealTime Using IMU and Surface EMG Sensors,” IEEE J. Biomed. Heal. Informatics, vol.20,no.5,pp.1281–1290,2016, doi: 10.1109/JBHI.2016.2598302.

[4] M. Safeel, T. Sukumar, K. S. Shashank, M. D. Arman, R. Shashidhar, and S. B. Puneeth, “Sign Language Recognition Techniques- A Review,” 2020 IEEE Int. Conf. Innov. Technol. INOCON 2020, pp. 1–9, 2020, doi: 10.1109/INOCON50539.2020.9298376.

[5] W. Aly, S. Aly, and S. Almotairi, “Userindependent american sign language alphabet recognition based on depth image and PCANet features,” IEEE Access, vol. 7, pp. 123138–123150,2019, doi:10.1109/ACCESS.2019.2938829.

[6] Aman Bhatia, and Mayand Kumar ,“Sign language Recognition using Convolutional neural network”, IEEE conference doi: 10.1109/ICICCS51141.2021.9432296

[7] Jinalee Jayesh Kumar Raval; Ruchi Gajjar “Real time sign language recognition using computer vision”IEEE conference , 2021,DOI: 10.1109/ICSP51351.2021.9451709