A detailed report on

Proctoring techniques during virtual interviews

By:

PRANJAL LAL (2021B3A71055G)

Prepared in fulfillment of the requirements of Practice School-1

At

iQuadra Information Services Pvt. Ltd, Nellore

A Practice School – 1 Station of

BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI

July 2023

Acknowledgment

I am incredibly indebted to my Industry Mentor, Srihari Rao Nannuru, for his precious guidance and encouragement. Without his guidance and support, the report would have been incomplete.

I would also like to express gratitude towards my Faculty Mentor, Dr. Arnab Guha, for his support and constant monitoring of the work on this project and the Practice School division for providing the fantastic opportunity to work at iQuadra Information Services Pvt. Ltd., Nellore.

Lastly, I want to thank my Friends who constantly supported me, provided crucial inputs, and made the report more reliable.

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE PILANI

(RAJASTHAN)

Practice School Division

Station: iQuadra Information Services Pvt. Ltd, Nellore

Duration: 2 months

Date of Start: 30th May 2023

Date of Submission: 20th July 2023

Title of the Project: Proctoring techniques during virtual interviews

ID No: 2021B3A71055G

Name: **Pranjal Lal**

Discipline: M.Sc. in Economics + B.E in Computer Science

Name of Industry Mentor:

Srihari Rao Nannuru

iQuadra Information Services Pvt. Ltd, Nellore

Name of PS Faculty Mentor:

Dr. Amab Guha

Department of Mechanical Engineering,

BITS-Pilani, Hyderabad Campus

Abstract

The main overview of the project is about proctoring methods and their significance in preventing malpractice during online video interviews. There are many tricks that the candidates use during interviews to get an unfair advantage. I am trying to create a prototype of the proctoring system that can counter those. The system includes the AI models like face detection, ID detection, noise detection, etc. It also acknowledges the effectiveness of current techniques while recognizing the need for ongoing research and development to advance proctoring technology. It will also give an insight into different software that we will use to create the prototype, like AWS, OpenCV, etc. It will also include the progress made by me in the making of a face recognition/detection model. and an eye-tracking model that can tell if a person is looking away from the camera.

Keywords used- Proctoring methods, AI models, AWS, OpenCV.

Project Area- AI/ML, Python, Data analysis

Table of contents

Introduction:

- Problem statement- The problem statement consists of four parts:

1. We must identify and describe effective proctoring measures, such as face detection, noise detection, and violation identification (e.g., camera rotation), which can be utilized during online video interviews. These measures are intended to ensure the authenticity and fairness of the interview process, minimizing the risk of unethical practices.

2. We must develop an online interview system prototype that incorporates the identified proctoring measures and techniques. The prototype will use Python and AWS, utilizing AI/ML techniques and data analysis to implement the proctoring functionalities. Additionally, understanding security and privacy concerns will be applied to ensure the system's robustness and protection of sensitive information.

   Models that must be developed are:

   a. **Facial Recognition: Identifying, Verifying, and Counting Persons in Images and Videos using Stored Faces** - Facial Recognition technology enables the identification and verification of individuals in images and videos by comparing their facial features with a stored set of faces. This advanced technology uses sophisticated algorithms and deep learning models to analyze facial characteristics, allowing for accurate and real-time recognition. Additionally, the technology can count the number of people present in a scene, providing valuable insights into whether someone else is sitting in the room with the candidate or not.
   b. **Eye Tracking: Analyzing how many times a person looks away from a camera in a recorded video -** Eye Tracking technology analyzes a recorded video to detect how many times a person diverts their gaze away from the camera. This can be used for various purposes like to suspect a person of cheating or even to judge their attention span or engagement patterns.

3. Once the prototype is developed, it will be thoroughly tested and evaluated to ensure its effectiveness in preventing malpractice and maintaining the integrity of the interview process ad also the quality of codes. The evaluation process will consider the system's accuracy, efficiency, and reliability in detecting and preventing malpractice.

4. Then, we must deploy the model using AWS for market use and showcase it on Git Hub to provide documentation, code samples, and instructions for users interested in utilizing the model or understanding its implementation.

- Need for a proctoring system- We need to understand the advantages of online exams over classroom exams. Online exams offer flexibility, accessibility, immediate feedback, enhanced question variety, efficient evaluation processes, and valuable data analytics. For example, if a person wants a certificate from a course at an institution in a different country, they do not need to go there for the examination and can take an online exam. Online exams are also helpful in keeping the records of those exams without any additional effort. But online exams are also very susceptible to many malpractices. That is the reason we need proctoring methods to assist in the process.

OBJECTIVE:

The landscape of tech hiring has witnessed a profound shift, with take-home assessments emerging as a favored approach over conventional methods. However, ensuring these assessments' integrity is paramount, as even one case of candidate malpractice can undermine the effectiveness of the entire testing process. The credibility and widespread adoption of take-home assessments and remote interviews hinge upon the existence of a robust and infallible online proctoring mechanism. Such a mechanism is imperative to maintain the authenticity and trustworthiness of the assessment process, guaranteeing fair evaluation and reliable results.

Thus, to be able to create such a mechanism, we need to know the ways in which the candidate tries to outsmart the system. According to Hackerearth.com, a candidate uses six major tricks to cheat., which are:

1. Surfing on another tab or engine for answers- It involves navigating away from the test platform or browser window to seek external information, which undermines the integrity and fairness of the evaluation process.

2. Using another device- Many candidates come prepared with a secondary device strategically positioned out of the web camera's view, allowing them to access hidden resources without detection by the system. This method provides them with a convenient means to obtain unauthorized assistance, remaining undetected by the proctoring system in place.

3. Copying and pasting codes from the web- Thanks to the recent advancements in Windows and Mac operating systems, candidates now have the capability to store and maintain a collection of copied texts. With just a few keystrokes, they can easily access these stored pieces of code. This functionality enables them to retrieve and utilize the copied texts during assessments swiftly.

4. Impersonation- Impersonation refers to the act of pretending to be someone else during an online assessment or exam. It involves assuming the identity of another individual, typically to gain an unfair advantage or to have more knowledge about that test.

5. Seating someone else in the room- Candidates exploit the absence of audio recording devices within the test system by strategically placing individuals in the room who possess greater knowledge or even talk to someone else taking the same exam.

6. Restroom breaks- Candidates go to the bathroom break and during the break, when they are out of the camera's reach, they seek some help and get the answers to the questions.

EXISTING METHODS AND THEIR PROS/CONS:

Different proctoring service providers, like Mercer Mettl, Proctorio, Proctoru, etc., are using various methods that have very high success but still have some demerits. Let us discuss some of the already existing methods:

1. **Face Recognition**- Face recognition is a proctoring measure that involves matching the face of a test-taker with an image in a database. It offers several advantages in preventing impersonation during assessments. Verifying the individual's identity reduces the risk of someone else taking the test on behalf of the intended candidate. Additionally, incorporating face recognition into the proctoring system enables the collection of another picture of the person, further enhancing the database for future identification purposes. However, there are some drawbacks to consider. Face recognition can be misleading or inaccurate if the lighting conditions or camera quality are subpar, leading to potential false positives or negatives. Privacy concerns are also a consideration, as the use of face recognition involves capturing and storing individuals' facial images, raising questions about data security and consent.
2. **Signature and ID matching**- ID and signature matching is a proctoring technique that verifies a test-taker ID document and signature by comparing them with the corresponding data stored in a database. It offers advantages such as enhanced identity verification and scalability. The software can be scaled to using it at airports, railway stations, hotels, etc., for identity-checking devices. However, challenges may arise due to the quality of ID documents and variations in signature styles, which can impact matching accuracy. Privacy concerns related to data security and consent also need to be addressed.
3. **Noise Detection**- Noise detection is a proctoring measure that identifies suspicious sounds or glitch noises during online assessments. It serves several benefits, including detecting the presence of someone else in the room or the use of unauthorized devices. However, there are some limitations to consider. False detections can occur, leading to unnecessary interruptions or distractions for test-takers. Additionally, noise detection is

primarily focused on identifying audible sounds and may not be effective in detecting nonverbal malpractices or subtle forms of cheating that do not produce noticeable sounds

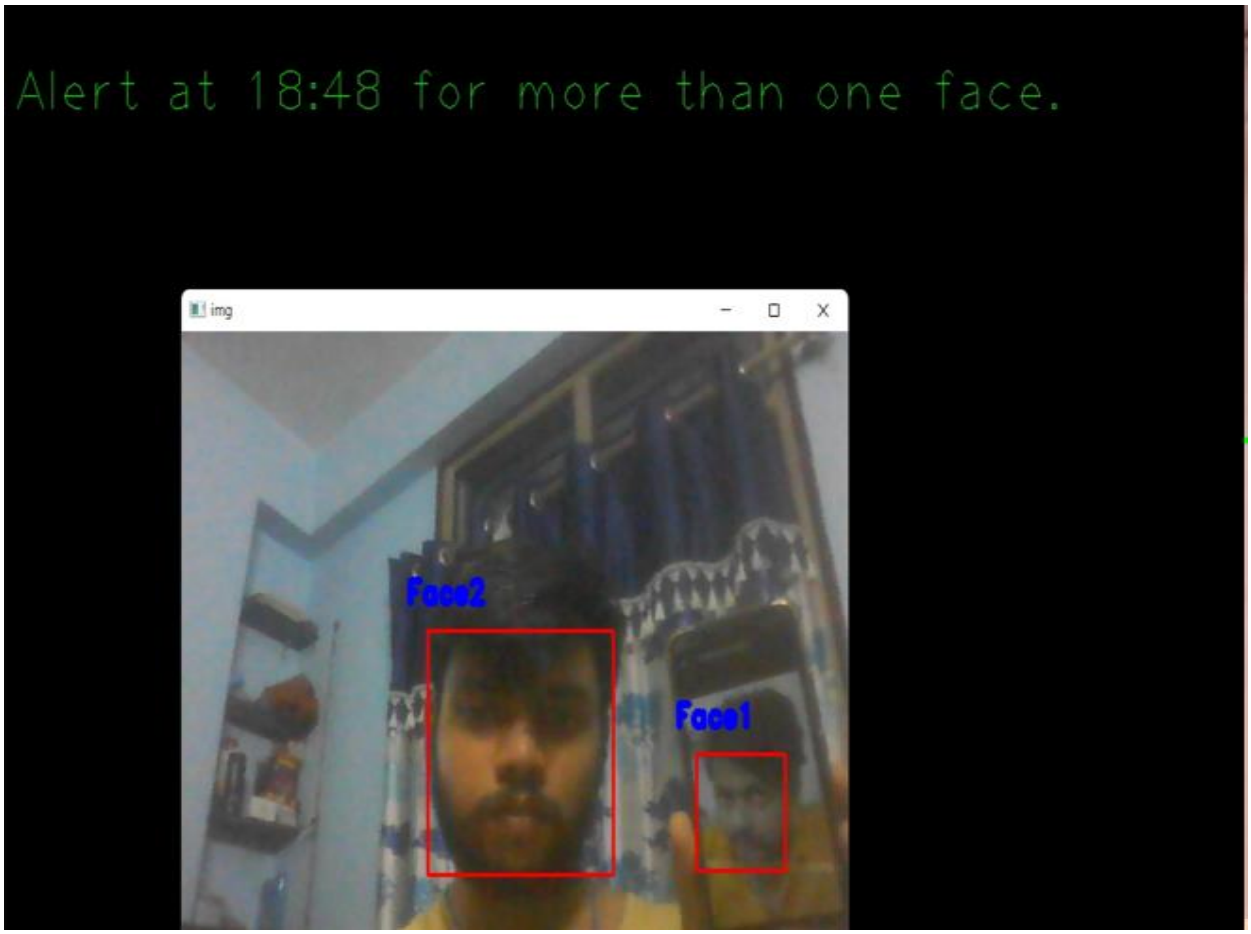
4. **Camera Rotation Detector**- Camera rotation detection is a proctoring measure that monitors the entire room for any suspicious activities during online assessments. Its primary purpose is to ensure that the candidate is not attempting to hide or manipulate any unauthorized materials or resources. One of the advantages of camera rotation detection is that it provides a comprehensive view of the surroundings, allowing proctors to identify any potential malpractices or prohibited actions. By monitoring the camera feed, proctors can detect if the candidate is attempting to change the camera angle or hide certain objects. However, there are certain limitations to consider. The effectiveness of camera rotation detection relies heavily on the development and accuracy of the underlying AI algorithms. The system must quickly capture and process information within a limited time frame to detect suspicious activities. Therefore, the efficiency of the AI system plays a crucial role in the success of camera rotation detection.
5. **Eye and hand movement tracker**- This measure aims to identify any suspicious or unauthorized activities that may indicate cheating or malpractice. One of the key advantages of eye and hand movement tracking is its ability to provide insights into the candidate's behavior and actions during the assessment. By analyzing the movements of their eyes and hands, proctors can identify patterns or irregularities that may indicate potential cheating, such as frequent glances away from the screen or unusual hand gestures. However, there are certain considerations when implementing eye and hand movement tracking. The accuracy and reliability of the tracking system are crucial for its effectiveness. The technology should be capable of accurately capturing and interpreting the candidate's movements, considering variations in hand gestures and eye-tracking algorithms.

MY PROPOSED SOLUTION AND FINAL PROGRESS:

❖ Following are the solutions that I proposed:

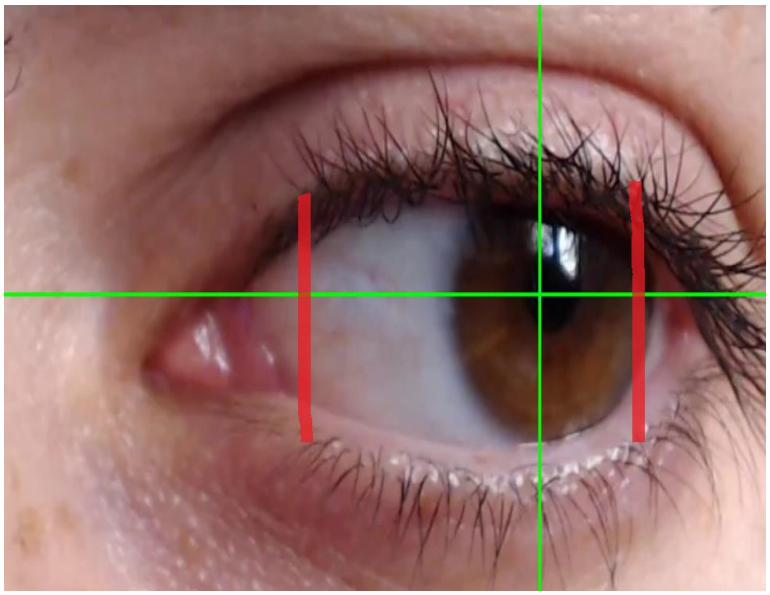
1. **Face detection and recognition-**

My proposed solution includes a system that has a classifier that detects more than one face, and if it does, it alerts the system with a message that multiple people are in the camera and can also flag the interview for that. Moreover, I will also create a model that does facial recognition based on the photo repository, and if the face in the video does not match the one in the database, it alerts the system with a different message that the person in the camera cannot be identified.



2. **Eye-tracking**-

For this problem, I used a more advanced model which can count the number of times the person has looked away from the camera. But the hurdle we have here is that to track each eye, we need a high-quality cropped eye video for better accuracy, and for that, we need a high-quality camera. But, the idea is that we create two sensitive lines, which can also be a circle surrounding the eye, either of which, when get crossed by the intersection point of the lines, can be counted as (the number of times the person crossed looked away+1) and finally we can give the number of times person looked away easily.



❖ For the completion of these, I used and learned various software. Following are the names of those:

1. **AWS-** Amazon Web Services (AWS) is a comprehensive cloud computing platform providing on-demand computing resources and services. Some AWS tools I learned about are:
   a. Amazon EC2 (Elastic Compute Cloud): EC2 provides scalable virtual servers in the cloud, allowing you to deploy your proctoring model on reliable and resizable instances.
   b. Amazon S3: S3 is a highly scalable object storage service that can be used to store the necessary data for your proctoring models, such as training data, models, and configuration files.
   c. Amazon Rekognition: Rekognition is a powerful AWS service that offers computer vision capabilities, including facial analysis and recognition.
   d. AWS Lambda: Lambda is a serverless computing service that allows you to run your code without provisioning or managing servers.
   e. Amazon CloudWatch: CloudWatch provides monitoring and observability services for your AWS resources. It can be utilized to collect and analyze logs, set up alarms, and gain insights into the performance and behavior of your proctoring system, ensuring its efficient operation.
2. **OpenCV**- OpenCV (Open Source Computer Vision) is an open-source computer vision and machine learning library. It offers over 2500 optimized algorithms for image and video processing, object detection, feature extraction, and more. OpenCV was written in C++ but supports various programming languages, including C++, Python, and Java, making it accessible to a broad community of developers.
3. **Haar Cascade Classifiers-** These are pre-trained models given by OpenCV for specific purposes. They are computationally efficient, allowing real-time or near real-time performance on various devices. It is trained by giving it many positive images and can be done in multiple stages. The limitation they have over Dlib models is that they are less

general-purpose and require external software to train them as they are not capable of storing images.

4. **datetime/time modules-** The datetime module in Python provides classes for manipulating dates, times, and time intervals. It allows formatting and parsing of dates and times, enabling convenient handling of time-related data. The time module provides functions to work with time-related operations, such as measuring execution durations and creating delays in Python programs.
5. **NumPy-** NumPy library can add matrices, n-dimensional arrays, and functions to operate on these matrices. NumPy's ndarray facilitates seamless integration with OpenCV, as both libraries treat images as matrices, allowing easy manipulation and processing of image data. It also helps in accessing different parts of an image.
6. **OS module-** The OS module in Python provides a way to interact with the operating system and perform various system-related tasks. It enables functionalities such as file and directory operations, environment variables management, and process management. Haar Cascade Classifiers use it to train their models.
7. **GitHub-** GitHub is a web-based platform that facilitates version control and collaborative software development using Git. We can upload their code to GitHub repositories, enabling easy sharing and collaborative development with others.

❖ Following are the four steps I took to do the above-mentioned projects:

1. **Preprocessing-** Preprocessing includes webcam setup or video import, for which we use the syntax ‘cv2.videocapture(), and data collection from the videos or even the real-time videos like clicking pictures, grabbing audio, etc. In the eye-tracking project, preprocessing also includes cropping the eye area, which can be done by using an eye-detection model.
2. **Training-** Training includes importing a model from Haar Cascade, Dlib, or TensorFlow and then encoding generation, which uses many mathematical functions, but we get it integrated with models. During this process, the model makes itself familiar with the faces it can find in images.
3. **Detection/Recognition-** For the facial recognition project, this step includes the actual

mechanism of face recognition from the set of images and detecting multiple faces. For the other project, it includes eye tracking and boundary creation.

4. **Output generation-** Output generation includes creating alert creation, counting the number of times the candidate looked away, and adding necessary delays in printing alerts so that the system does not bug out.

CONCLUSION AND FUTURE WORK:

- Following are the project deliverables and learning outcomes from the fulfillment of the project:
    1. **Prototype of the Online Interview System**: This functional prototype will incorporate the identified proctoring measures and techniques discussed earlier. It should demonstrate the ability to prevent malpractice during online video interviews, including cheat detection, face detection, noise detection, and other violations.
    2. **Project Reports and Presentations**: Comprehensive project reports and presentations that I prepared, documenting the development process, methodology, findings, and outcomes.
    3. **AWS Configuration and Model Deployment**: The configuration files and settings used to deploy the proctoring model on AWS should be provided. This includes details on setting up virtual instances, storage configurations, and any necessary services or resources required for the system to run successfully.
    4. **GitHub Repository Setup and Collaborative Development**: A GitHub repository should be established to host the project's source code, documentation, and related materials. I will also learn how to do collaborative development on GitHub.
    5. **Python Libraries and Their Applications**: A compilation of the Python libraries used in the project and their applications should be documented. This would provide insights into the specific libraries leveraged for implementing the proctoring measures, data analysis, AI/ML techniques, and other functionalities within the Online Interview System.
    6. **ML Model Creation**: I learned how to create machine learning models for tasks such as facial recognition and eye tracking, using libraries like OpenCV and NumPy to process and analyze image data.

In the future, I wish to learn how to train models using Dlib. I would also like to include face identification from a set of faces in my first project. I also wish to complete the eye-tracking model by counting the line triggers. I would also like to add API endpoints, enabling seamless integration with other applications or systems. Finally, I will refine these models and upload it to my GitHub and resume.

APPENDIX A:

Codes for detection of multiple faces in a room using a pre-trained model:

(Note: I still have to add the face recognition model to it)

```
import cv2
import numpy as np
from datetime import datetime
import time

# Load the cascade
face_detect = cv2.CascadeClassifier('haarcascade_frontalface_alt.xml')
# Read the input image
img = cv2.VideoCapture(0)
ytest_pos = 40
last_alert_time = time.time()
ytest_pos = 40

# Text output window
outp = np.zeros((800, 1200, 3))

while True:
    _, img1 = img.read()
    gray = cv2.cvtColor(img1, cv2.COLOR_BGR2GRAY)
    faces = face_detect.detectMultiScale(gray, 1.1, 4)
    i = 0
    for (x, y, w, h) in faces:
        rect = cv2.rectangle(img1, (x, y), (x + w, y + h), (0, 0, 250), 2)
        i = i + 1
        cv2.putText(img1, "Face" + str(i), (x - 20, y - 20),
cv2.FONT_HERSHEY_DUPLEX, 0.8, (255, 0, 0), 3)

    cv2.imshow('img', img1)


    if i > 1:
        current_time = time.time()
        if current_time - last_alert_time > 10:
            ytest_pos += 40
```

```
            cv2.putText(outp, "Alert at {} for more than one
face.".format(datetime.now().strftime("%H:%M")),
                        (10, ytest_pos), cv2.FONT_HERSHEY_PLAIN, 3, (0, 255, 0))
            last_alert_time = current_time

    cv2.imshow('final', outp)

    if cv2.waitKey(50) & 0xff == 27:
        break

cv2.destroyAllWindows()
img.release()
```

APPENDIX B:

Codes for eye tracking project to count the number of times a person looked away from the camera using a pre-trained model:

(Note: I am still trying to figure out on how to use those lines to count the number of gazes away from the camera.)

```
import cv2
import numpy as np

video_capture = cv2.VideoCapture("eye_recording.flv")

# Sensitive line positions (percentage of the region of interest)
left_sensitive_line = 0  # 15% from the left
right_sensitive_line = 1  # 85% from the left

while True:
    ret, frame = video_capture.read()
    if not ret:
        break

    region_of_interest = frame[269: 795, 537: 1416]
    rows, cols, _ = region_of_interest.shape
    gray_roi = cv2.cvtColor(region_of_interest, cv2.COLOR_BGR2GRAY)
    gray_roi = cv2.GaussianBlur(gray_roi, (7, 7), 0)

    _, threshold = cv2.threshold(gray_roi, 3, 255, cv2.THRESH_BINARY_INV)
    contours, _ = cv2.findContours(threshold, cv2.RETR_TREE,
cv2.CHAIN_APPROX_SIMPLE)
    contours = sorted(contours, key=lambda x: cv2.contourArea(x), reverse=True)

    for cnt in contours:
        (x, y, w, h) = cv2.boundingRect(cnt)

        #cv2.drawContours(region_of_interest, [cnt], -1, (0, 0, 255), 3)
        cv2.rectangle(region_of_interest, (x, y), (x + w, y + h), (255, 0, 0), 2)
        cv2.line(region_of_interest, (x + int(w/2), 0), (x + int(w/2), rows), (0,
255, 0), 2)
        cv2.line(region_of_interest, (0, y + int(h/2)), (cols, y + int(h/2)), (0,
255, 0), 2)

        # Draw sensitive lines on both left and right ends of the eye
```

```
        left_line_x = int(x + left_sensitive_line * w)
        right_line_x = int(x + right_sensitive_line * w)
        cv2.line(region_of_interest, (left_line_x, 0), (left_line_x, rows), (0, 0,
255), 2)  # Left sensitive line
        cv2.line(region_of_interest, (right_line_x, 0), (right_line_x, rows), (0,
0, 255), 2)  # Right sensitive line
        break

    cv2.imshow("Threshold", threshold)
    cv2.imshow("Gray ROI", gray_roi)
    cv2.imshow("Region of Interest", region_of_interest)
    key = cv2.waitKey(30)
    if key == 27:
        break

cv2.destroyAllWindows()
video_capture.release()
```

References

1. **"Online Proctored Exam | Online Exam Proctoring | Remote Proctoring - Mettl." https://mettl.com/en/online-remote-proctoring/.**
2. **"The ProctorU Proctoring Platform - Advanced Exam Technology Backed by ...." https://www.proctoru.com/.**
3. **"6 Ways Candidates Try To Outsmart A Remote Proctored Assessment." 20 Nov. 2020, https://www.hackerearth.com/blog/talent-assessment/6-ways-candidates-think-they-are-smarter-than-your-remote-assessment-tool/.**
4. **"AWS Certified Cloud Practitioner Certification Course (CLF-C01) - Pass ...." https://www.youtube.com/watch?v=SOTamWNgDKc.**
5. **"Facial Recognition using OpenCV (Python) | by Aditya Dhapola ... - Medium." 07 Nov. 2019, https://medium.com/analytics-vidhya/facial-recognition-using-opencv-python-b90c94fbac8f.**
6. **"Face Recognition with Python, Dlib, and Deep Learning." 10 Oct. 2022, https://dontrepeatyourself.org/post/face-recognition-with-python-dlib-and-deep-learning/.**
7. **"opencv/haarcascade_frontalface_default.xml at 4.x · opencv/opencv - GitHub." https://github.com/opencv/opencv/blob/4.x/data/haarcascades/haarcascade_frontalface_default.xml.**