MBEYA UNIVERSITY OF SCIENCE AND TECHNOLOGY

MUST

2012

CHUO KIKUU CHA SAYANSI NA TEKNOLOJIA MBEYA

COURSE NAME	BACHELOR DEGREE OF COMPUTER ENGINEERING.
PROJECT TITLE	SMART INVIGILATION SYSTEM.
MODULE NAME	SENIOR PROJECT 2
MODULE CODE	COB 4117
STUDENT NAME	SHUKURU NICOLAUS DIMOSO
REGISTRATION NUMBER	19100534050012
EXAMINATION NUMBER	UE/BEC/22/17256
SUPERVISOR	LIHENDIME MADEMBWE
LEVEL	UQF-8-SEMESTER-II
ACADEMIC YEAR	2022-20223

DECLARATION

I, SHUKURU NICOLAUS DIMOSO, hereby declare that the work titled “Smart Invigilation System” has been carried out by me in the department of Computer Science and Engineering at Mbeya University of Science and Technology.

…..…………..……………

Shukuru N. Dimoso (STUDENT).

…………………………..

Date

CERTIFICATION

The undersigned certify that, he has read and hereby recommend for acceptance by MBEYA UNIVERSITY OF SCIENCE AND TECHNOLOGY(MUST) a project titled ‘SMART INVIGILATION SYSTEM in the fulfillment for the Bachelor of Computer Engineering at Mbeya University of Science and Technology (MUST).

…………..………………

Lihendime Madembwe (SUPERVISOR).

…………………………

Date

ACKNOWLEDGEMENT

I would like to thank my ALMIGHTY GOD for his life grace which enabled me to have time for project study and presentation. I would like also to thank my project coordinator, SHADRACK NJUGUNYA for his advices and allowance to work on this project.

I would also thank my project supervisor ‘LIHENDIME MADEMBWE for directing, advising me with this project. I would also want to thank to the department staffs of COMPUTER SCIENCE AND ENGINEERING for their help and advices on this project.

ABSTRACT

Examination cheating is the situation which occurs when a student attempts to get academic credit in a way that is dishonest, disrespectful, irresponsible or unfair. Examination cheating activities like face movement, head movements, hand movements, or hand contact are extensively involved, and the rectitude and worthiness of fair and unbiased examination are prohibited by such cheating activities. The aim of this project is to develop a system to supervise or control unethical activities in real-time examinations. Exam supervision is fallible due to limited human abilities and capacity to handle students in examination rooms, and these errors can be reduced with the help of the Smart Invigilation System.

This work presents an automated system for exams invigilation using deep learning and computer vision approaches i.e., MTCNN and PyTorch . MTCNN is an object detection model that is implemented to detect the suspicious activities of students during examinations based on their face movements, and for starting capturing the video of students Opencv is used.

The system is fully efficient in detecting and monitoring students in one frame during examinations. Different real-time scenarios are considered to evaluate the performance of the Automatic Invigilation System. The proposed invigilation system can be implemented in colleges, universities, and schools to detect and alert student suspicious activities. Hopefully, through the implementation of the proposed invigilation system, we can prevent and solve the problem of cheating because it is unethical.

COPYRIGHT

Table of Contents

LIST OF ABBREAVIATIONS

OPENCV	Open Source Computer Vision.
SSD	Single Short Detector.
YOLO	You Only Look Once.
CNN	Convolution Neural Networks.
DLIB	Digital Library.
HOG	Histogram of Oriented Gradient
MTCNN	Multi-Task Cascading Convolutional Neural Network

CHAPTER ONE

1.1. Introduction

Smart Invigilator System is a project implemented to alert and detect unethical activities of students in a real time examination. Unethical can be defined as a situation when a person does anything which is considered unacceptable.

So during exams, cheating is the among of the unethical activities because is the against the examination rule. Cheating occurs when a student attempts to get academic credit in a way that is dishonest, disrespectful, irresponsible or unfair.

During exams a student can cheat using eyes movement when student is copying or looking from another's sheet, movement of the head from one side to another (left, right, or upward), lip movement when a student is trying to talk with another student during exam, movement of the face from one side to another (left, right, or upward), when a student uses electronic gadgets such as mobile phone that can be used for malpractice, also motion of the students from one place to another.

1.2. Background information

In a traditional examination and formal examination, question papers are provided to students where student respond in the form of answers in a limited time period as we do now.

But in traditional and formal examination, the invigilators (examination supervisors) their job is to prevent any kind of cheating in a examination room such as gesture communications, movements of students during examinations and monitor students from cheating by observing the uses of notes or any cheating materials.

As a time goes on, now we have so many invigilation systems which have designed to detect, to monitor, to supervise and to alert cheating activities of students during exams.

Example some of universities now days' use closed circuit television to monitor or detect cheating activities in an examination room with the aid of human invigilators who are checking student when conducting their exams through closed circuit television.

Examples of other systems which have been implemented for invigilation purpose are;

1. Realization of intelligent invigilation system based on adaptive threshold.
2. Multi-index examination cheating detection method based on neural network.
3. Real time automated invigilator in classroom monitoring using computer vision.

But both of those systems are not more efficiently for offline examinations, aim was for online examination which can monitor and detect one student at a time and each student must have his or her camera.

So to overcome the problem of offline examinations monitoring and to reduce a load to a invigilation staffs, the proposed system Smart Invigilation System based on computer vision that can detect and alert suspicious activities of students in a real time examinations is implemented.

This project is time constraints, so it focuses on the analysis of the face movement of each student during exams.

By analyzing the movement or orientation of the face, a cheating label is considered for the following face movements left, right, or upward and no cheating is considered only for down movement of the face when student is doing his/her exam.

1.3. Problem Statement

Academic dishonesty and cheating have constantly been worrying factors for the educational institutes in the education system worldwide. The fact is that the students adopt different cheating approaches during the examination. There are psychological and social reasons why students opt to cheat in exams, such as parent’s pressure, feeling of incompetency, want better grades, time constraints, fear of failure, and take a risk with less fear of being caught. So invigilation process becomes a problem during exams due limited human ability and capacity to handle students in examination room.

A good invigilation system must detect and alert cheating activities in a real time and checks must be promptly, accurately and automatically.

In the past few years, the process of detecting cheating activities of students during online and offline exams have been detected multiple times by using various techniques. Some of the related works developed for detecting cheating activities of students are;

1.3.1. Automated Invigilation System for detection of suspicious activities during examination.

Implemented a system using spatial-temporal features to detect student’s abnormal behavior, this model based on the movements of the students like turning around, movements of the head, movements of hands and other part of the body.

➢ Strength of the system

- The system helps to detect cheating in real time examinations.
- It was used for both online and offline examinations.

➢ Weaknesses of the system

- The problem with this system was overlapping of detection boxes when students sitting arrangement is conducted or when students sitting arrangement is very close or during the collection of the papers.

1.3.2. Real Time Automated invigilator in Classroom Monitoring using Computer Vision

Also detecting the cheating of students using the openpose module in which the poses of the students were extracted by fetching the multiple joints of the body, and then cheating label drawn on the faces of the students who are trying to make movements during exams.

➢ Strength of the system

- The system helps to detect cheating in a real time examination.

➢ Weaknesses of the system

- Due to this, increase the computation of the system and hence not suitable for real time.

1.3.3. Closed circuit Television and manual systems

Globally even also in Tanzania, some of universities use human invigilators and some of the universities use closed circuit television camera to monitor students during examinations.

- Strength of the system
  - The system helps to detect cheating in examination rooms.
- Weaknesses of the system
  - Both of those systems do not alert suspicious activities automatically instead use human being to detect suspicious activities.

1.3.4. Expected solution

Therefore, the proposed model which is smart invigilation system is focused to come up with a smart invigilation system which will monitor and alert cheating activities in a real time during exams.

1.4. Project objectives

1.4.1. Main objective

The main objective is to design a Smart Invigilation System.

1.4.2. Specific objectives

1. To create invigilator interface for capturing live video.

2. To create an algorithm that will extract facial images from the webcam.

3. To create an algorithm that will check the angle of face orientation to determine whether exceeding or not.

4. To link a software part with hardware part.

1.5. Scope of the project

Prototyping, the system will use webcam or mobile phone camera for capturing video.

1.6. Significance

1. To reduce dishonesty and cheating among the students during examinations.

2. To reduce the burden on the invigilation staff members.

3. To improve fairness and integrity of examinations.

1.7. Limitation

The process of detecting cheating activities of students will be done only if the camera is in front of the student face and not behind him/her.

CHAPTER TWO

2. Literature Review

2.1. Introduction

This part focuses on the looking of different solutions implemented by researches to remove or reduce the problem of cheating of students during exams.

The review is comprised of works that have been done to find the solution to the problem, the systems that were closely related to the proposed model are reviewed and noted the strengths and weaknesses of the solutions.

By studying these systems, we were able to find the deficiency and stated ways in which the solutions can be implemented.

2.2. Related work

The following are common systems (related works) as elaborated below; -

2.2.1. Realization of intelligent invigilation system based on adaptive threshold.

Adil, (2022) created a system that will help schools and universities to monitor cheating activities such as whispering, hand contact in the classroom during examinations. The system based on certain threshold levels, in which a box is drawn around the student and whenever a student moves his hand beyond this level, it will be detected. They used the Voila jones algorithm which used to recognize face and Adaboost for their proposed model

Drawback: The drawback of this algorithm is that, the overlapping may occur when there is less space in students sitting arrangement.

2.2.2. Automation of traditional exam invigilation using closed circuit television and bio-metric.

Ketab, (2022) created an invigilation system with the authentication process of students. The proposed system was used for both offline and online examination.

It uses the technique of multi- system bio-metrics for the student’s identification along with a 3D facial recognition method. But to monitor students during examinations, the model used eye movement and speech recognition module is implemented to detect any improper voice

Drawback: The drawback of this algorithm is that, it requires multiple cameras to capture students.

2.2.3. Automatic invigilation using computer vision.

Zhenhong and wan, (2022) created a deep learning module for the identification of cheating during exams, also another algorithm which is YOLO (You Only Look Once) algorithm is implemented to detect the boundary box for each student along with Openpose algorithm which used to identify and recognize student correct posture and position and label them as suspected or non-suspected.

YOLO is an algorithm that uses neural networks to provide real time object detection. This means that prediction in the entire image is done in a single algorithm run. The CNN is used to predict class probabilities and bounding boxes simultaneously.

The Yolo algorithm consists of various classes. Some of the common ones include tiny YOLO and YOLOV3.

Drawback: This system works fine for online examinations and not suitable for offline examinations, because there is overlapping of detection boxes.

2.2.4. Application of a single- shot Multi-core detection algorithm in intelligent visual monitoring of examination room.

Xiao, (2022) created an automatic real time based invigilation system by using a single-shot Multi-box (ssd) detector and then comparing the accuracy of the system with Yolo and other object detection deep learning modules.

SSD is designed for object detection in real time . Faster R-CNN uses a region proposal network to create boundary boxes and utilizes those boxes to classify objects.

This system used to detect and identify cheating activities of students during the exam with an accuracy of 88%, so it has been adapted to the education sector.

Drawback: Requires multiple cameras to supervise and monitor each student.

2.2.5. Real time automatic invigilator using inception V3 CNN algorithm.

Kulkami, Rutuja, (2022) created a method which used V3CNN for the classification of body movements into legal and illegal activities during examination.

In their system, the automatic invigilation system was proposed in which students are being monitored and supervised. The backbone of this architecture was the inception V3CNN model.

The inception V3 model used several techniques for optimizing the network for better model adaptation. This is a deep learning model based on Convolution Neural Networks, which is used for image classification.

Drawback: But the problem with this invigilation system is that, it is not enables the capture of more than 15 students during the examination while live stream.

2.3. Literature review summary

S/NO	AUTHOR	SYSTEM	DRAWBACKS
1	Adil, (2022)	Realization of intelligent invigilation system based on adaptive threshold	Drawback of this algorithm is that, the overlapping may occur when there is less space in students sitting arrangement.
2	Ketab, (2022)	Automation of traditional exam invigilation using closed circuit television and bio-metric	Drawback of this algorithm is that, it requires multiple cameras to capture students

3	Zhenhong and wan, (2022)	Automatic invigilation using computer vision	This system works fine for online examinations and not suitable for offline examinations, because there is overlapping of detection boxes
4	Xiao, (2022)	Application of ssd core detection algorithm in intelligent visual monitoring of examination room	Requires multiple cameras to supervise and monitor each student
6	Kulkami, Rutuja, (2022)	Real time automatic invigilator using inception V3 CNN algorithm	The problem with this invigilation system is that, it is not enables the capture of more than 15 students during the examination while live streaming

2.4. Proposed system

So a proposed system which is Smart Invigilation System is focused to come up with simple solution to detect and alert cheating activities of students in real time examinations.

This system can be used in both offline and online examinations. This system uses two algorithms which are MTCNN and PyTorch which is more accurate for facial features extraction.

It has no limit for capturing number of students in examination room, but it depends on the device used to record live video. But to monitor students during examinations, the system used face movement with aid of boundary boxes and utilizes those boxes to classify

student’s activities.

The system based on certain levels (angles), in which a box is drawn around the student and whenever a student moves his face beyond this level, it will be detected as a cheating.

2.4.1. Block diagram of the proposed system

v

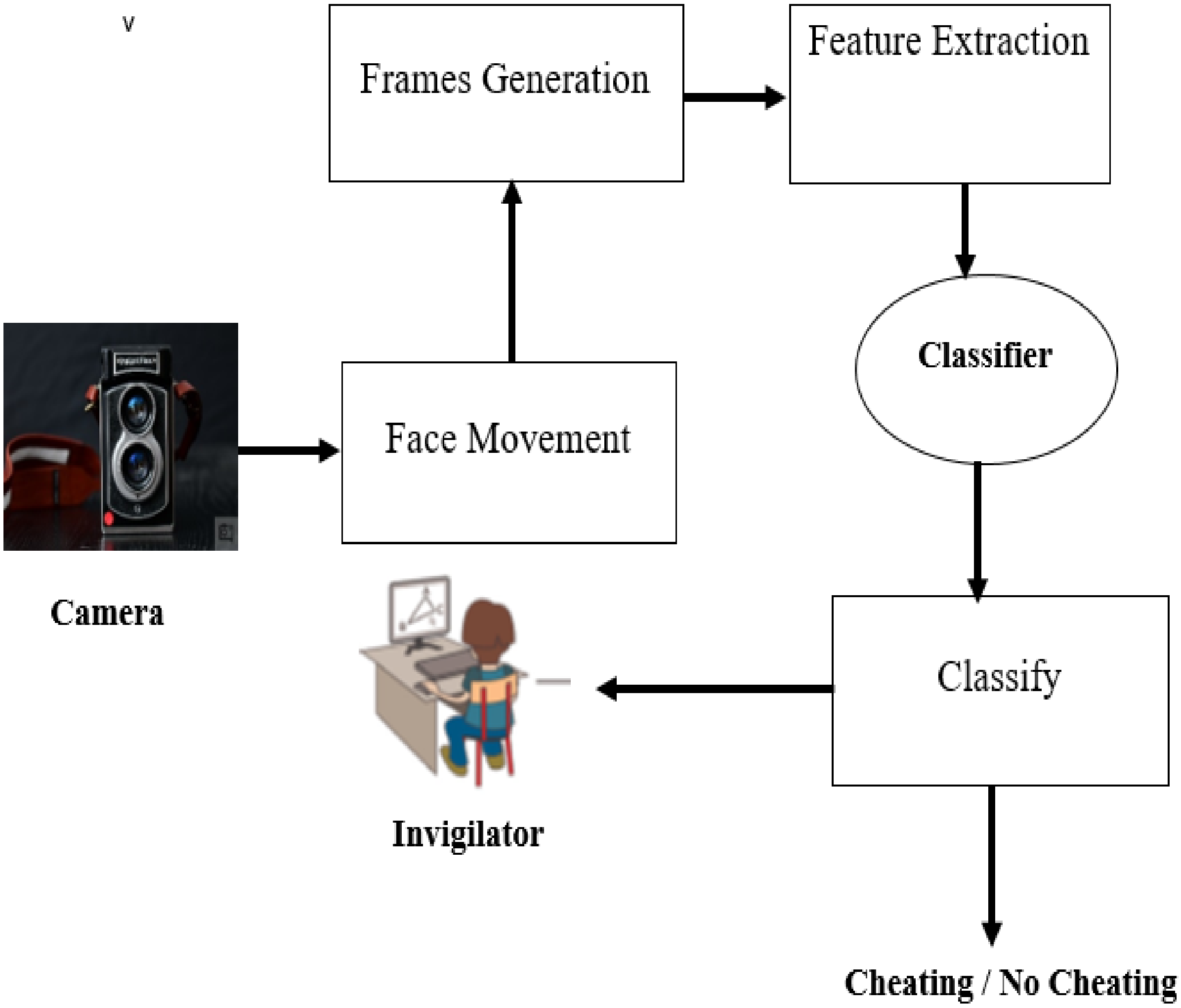


Figure 2.4.1: Block diagram of the proposed system

2.4.2. Working principle of the proposed model

In the proposed system two modules are implemented, the first is opencv and second is MTCNN model. Opencv (open source computer vision library) is an open source computer vision and machine learning software library.

It is the huge library for the computer vision, machine learning and image processing. It plays a major role in real-time operation which is very important in today’s systems. But in a proposed system opencv is used only to start the camera for capturing live video.

MTCNN is a deep learning model implementing a variety of deep learning algorithms, including classification, regression, clustering, data transformation and structured prediction. In a proposed system MTCNN used to estimate the location of 5 coordinates (x, y) that map the facial points on a person’s face and then predict the orientation of the face from live video.

In a proposed system surveillance camera is used for live video recording of students in the examination room, video is then converted into frames by using python code.

MTCNN used for facial features extraction with more than 90 % accuracy. In this project it is implemented to detect unethical or cheating activities in the examination room. The system is trained on two types one is labeled as cheating and the other one is labeled as no cheating.

CHAPTER THREE

3. METHODOLOGY

3.1. Introduction

In this article, data collection method, sampling techniques, methodology approach are elaborated. According to the project management institute (PMI), the methodology is defined as a system of practices, techniques, procedures and rules used by those who work in a discipline (zenkit.com, 2018).

3.2. Data collection

Data collection Is a process of gathering and analyzing specific information to proffer solutions to relevant questions and evaluate the results. There are two types of data collection namely primary data collection and secondary data collection (www.formplus.us, 2020).

3.2.1. Primary data collection

Type of data that collected by or that has been generated by the researchers directly through interviews, surveys, experiments, Observations, questionnaires especially designed for understanding and solving the research problem at hand. A primary data collected directly from the original source, so it’s the first-hand data collected from a field.

The proposed system will use this kind of method to collect information through questionnaires. This method takes much time and money because it needs to travel to interview them, and in case of observation it needs more much time to observe until information needed are acquired.

3.2.2. Secondary data collection

Secondary data collection, on the other hand, is referred to as the gathering of second-hand data collected by an individual who is not the original user. It is the process of collecting data that is already existing, whether published books, journals, and/or online portals.

In terms of ease, it is much less expensive and easier to collect. Through it the data from internet, books and other journals will be gathered so that to be used whenever needed in this project.

3.3. Design approach

The important concept of this project can be explained in two main phases: image acquisition, face detection and feature extraction.

3.3.1. Image acquisition

In this part which is data acquisition; camera is used to capture the video of students, and then the video is converted in to multiple frames or images to detect the students.

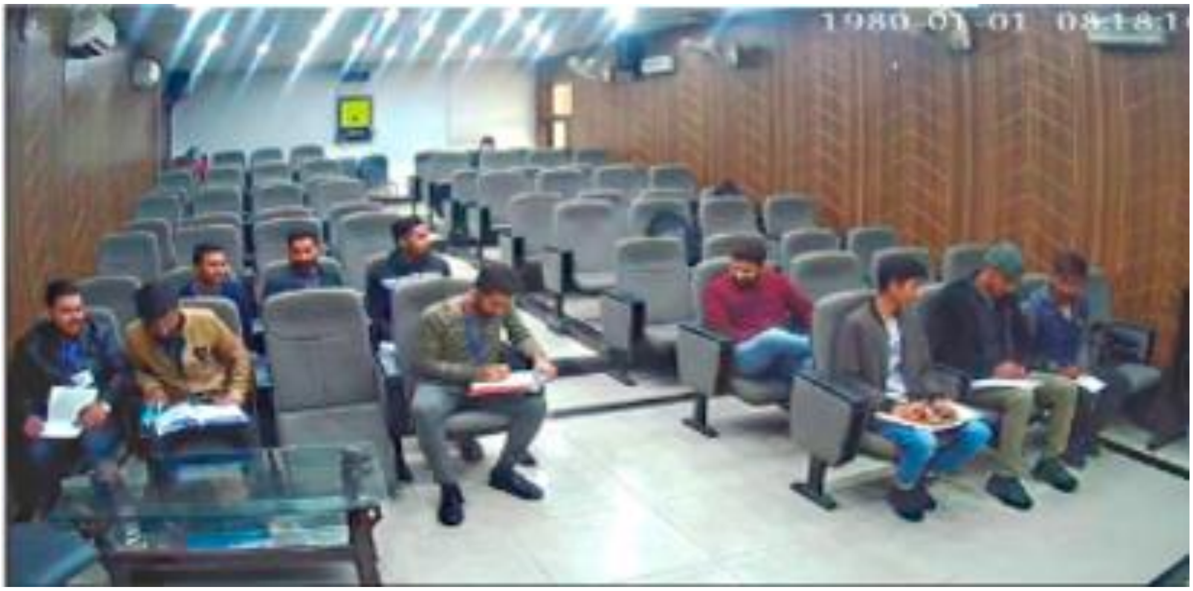


Figure 3.3.1: image acquisition

3.3.2. Face detector

MTCNN (Multi-Task Convolutional Neural Network) is implemented to detect students faces.The algorithm detects student faces along with 5 points facial landmarks.

Face detection is generally divided in to 3 steps by using MTCNN model which are as follow:

1. **Face detection:** The step is to detect all the faces in the live video from the camera. This can be done using a face detector called HOG (Histogram of Oriented Gradients).
2. **Face alignment using facial landmarks:** Second step is to align or normalize the face using facial landmarks in order to improve the accuracy of the face orientation.

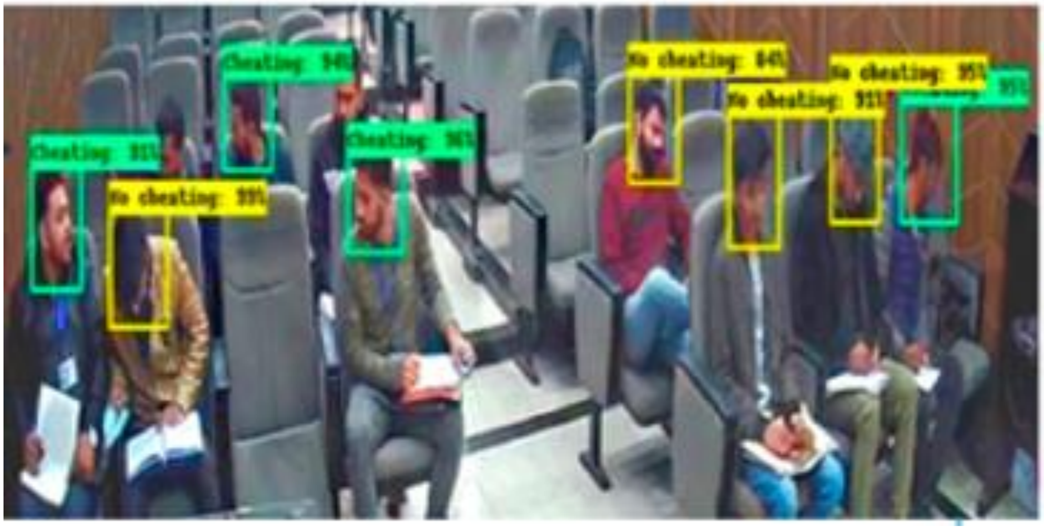


Figure 3.3.2: Face alignment

3. **Face encoding:** In this step the face images are passed to the model and extract the facial features.

3.4. Sampling technique.

Sampling technique is the selection of sample components that will provide representative perspective of the entire sample. Simple random sampling, cluster sampling, stratified random sampling, and systematic sampling are some of the techniques used.

For this project the simple random sampling will be used. Simple random sampling is the one which individual is chosen entirely by chance and each member of the population has an equal chance of being included in the sample.

3.5. Data analysis

This process will involve evaluation of data collected, with the aim of obtaining useful information which will help in drawing conclusion on what to be used in system design. Data analysis will aid in deciding the type of software and hardware to be used.

3.6. Methodology approach

3.6.1. RAD

Actually, in this project Rapid Application Development (RAD) as the type of prototyping model will be used. RAD is a development model prioritizes rapid prototyping and quick feedback over long drawn out development and testing cycles. With RAD, developers can make multiple iterations and updates to a software rapidly without needing to start a development schedule from scratch each time kissflow.com, (2021).

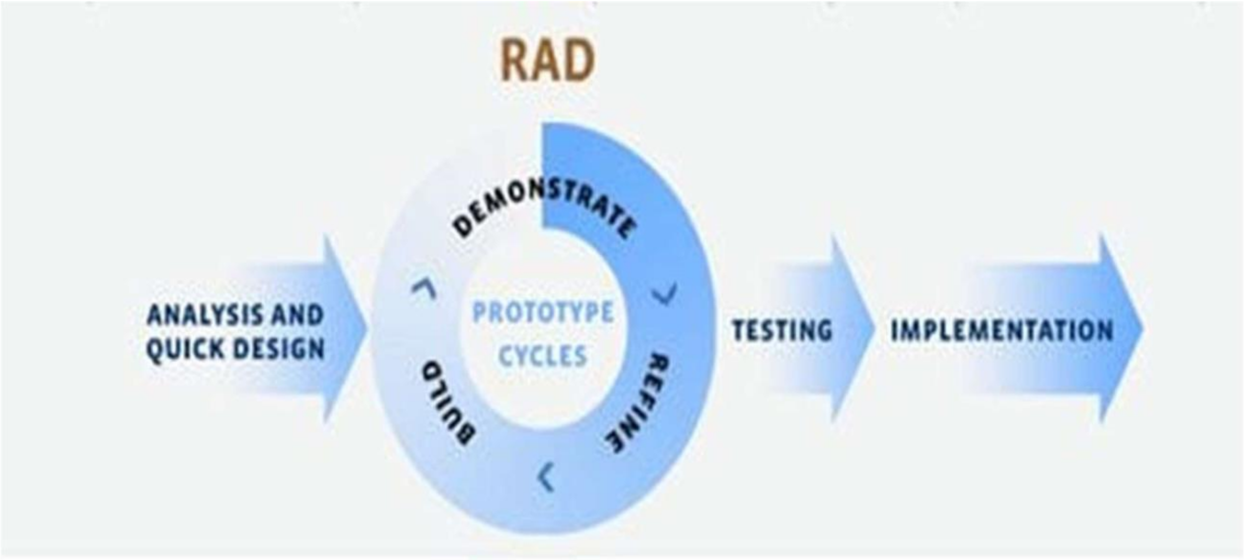


Figure 3.6.1: Steps of Rapid application development

3.6.2. Why RAD preferred

1. Enables face-to-face interactions.
2. Support, trust, and motivate the people involved.
3. It encourages Customer satisfaction through early and continuous software delivery.
4. Quicker customer feedback provides a better idea of customer needs.
5. Requirements can be changed at any time.
6. Encourages and priorities customer feedback.
7. Reviews are quick.
8. Development time is drastically reduced.

3.7. Testing

The prototype will then be tested to check its functionality with reference to specified performance parameters.

CHAPTER FOUR

4. Data Collection and Analysis

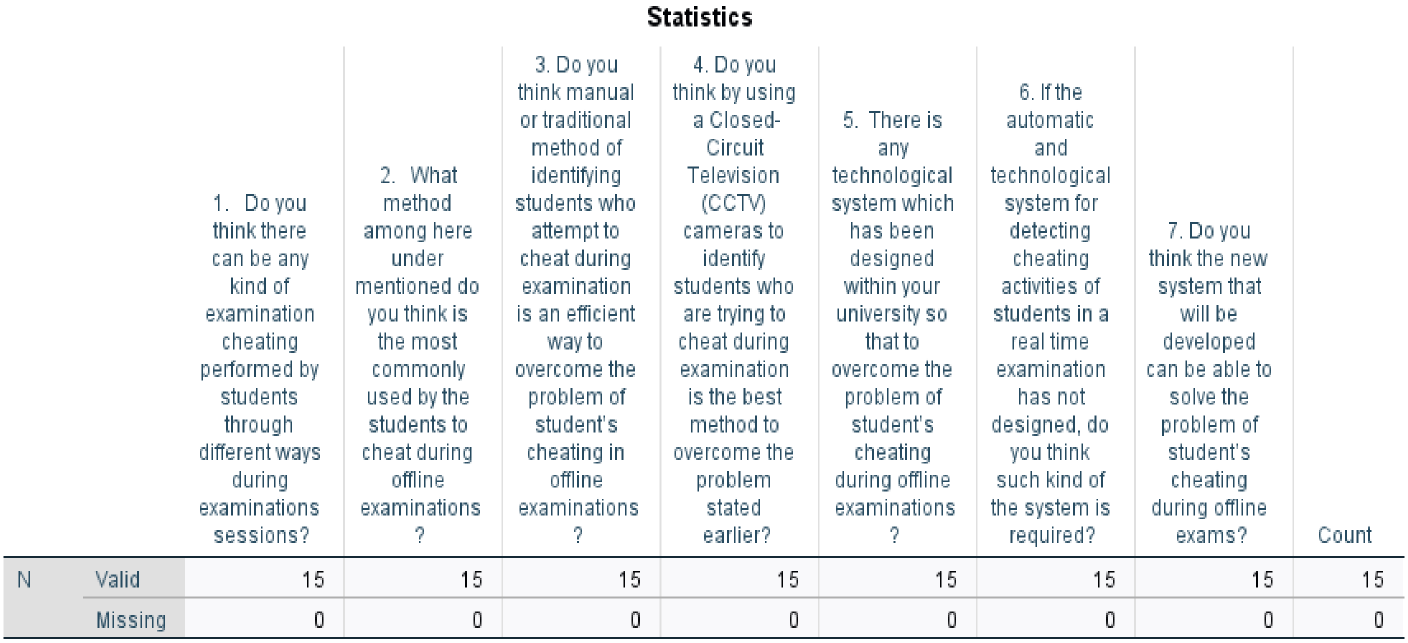
4.1. Data Collection

Data collection Is a process of gathering and analyzing specific information to proffer solutions to relevant questions and evaluate the results. There are two types of data collection namely primary data collection and secondary data collection (www.formplus.us, 2020). In this project both primary and secondary data is collected. The primary data were collected through questionnaire and the secondary data were gathered through reading different articles from the online publications.

4.1.1. Primary data collection

Type of data that collected by or that has been generated by the researchers directly through interviews, surveys, experiments, Observations, questionnaires especially designed for understanding and solving the research problem at hand. A primary data collected directly from the original source, so it's the first-hand data collected from a field.

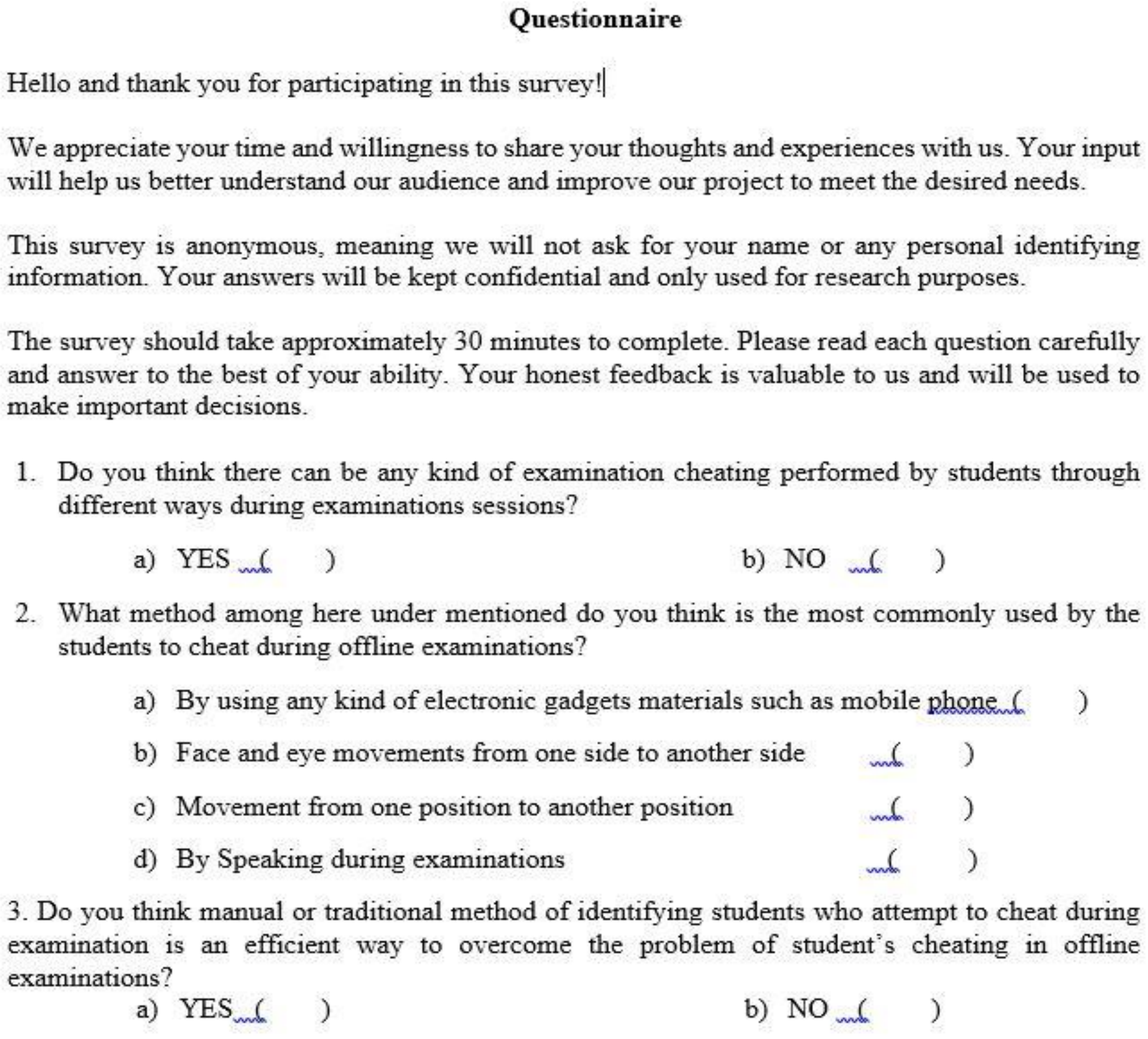
The proposed system will use this kind of method to collect information through questionnaires.



Statistics

		1. Do you think there can be any kind of examination cheating performed by students through different ways during examinations sessions?	2. What method among here under mentioned do you think is the most commonly used by the students to cheat during offline examinations ?	3. Do you think manual or traditional method of identifying students who attempt to cheat during examination is an efficient way to overcome the problem of student's cheating in offline examinations ?	4. Do you think by using a Closed-Circuit Television (CCTV) cameras to identify students who are trying to cheat during examination is the best method to overcome the problem stated earlier?	5. There is any technological system which has been designed within your university so that to overcome the problem of student's cheating during offline examinations ?	6. If the automatic and technological system for detecting cheating activities of students in a real time examination has not designed, do you think such kind of the system is required?	7. Do you think the new system that will be developed can be able to solve the problem of student's cheating during offline exams?	Count
N	Valid	15	15	15	15	15	15	15	15
	Missing	0	0	0	0	0	0	0	0

4.1.2. Questionnaire Questions



Questionnaire

Hello and thank you for participating in this survey!

We appreciate your time and willingness to share your thoughts and experiences with us. Your input will help us better understand our audience and improve our project to meet the desired needs.

This survey is anonymous, meaning we will not ask for your name or any personal identifying information. Your answers will be kept confidential and only used for research purposes.

The survey should take approximately 30 minutes to complete. Please read each question carefully and answer to the best of your ability. Your honest feedback is valuable to us and will be used to make important decisions.

1. Do you think there can be any kind of examination cheating performed by students through different ways during examinations sessions?

   a) YES ( ) b) NO ( )

2. What method among here under mentioned do you think is the most commonly used by the students to cheat during offline examinations?

   a) By using any kind of electronic gadgets materials such as mobile phone ( )

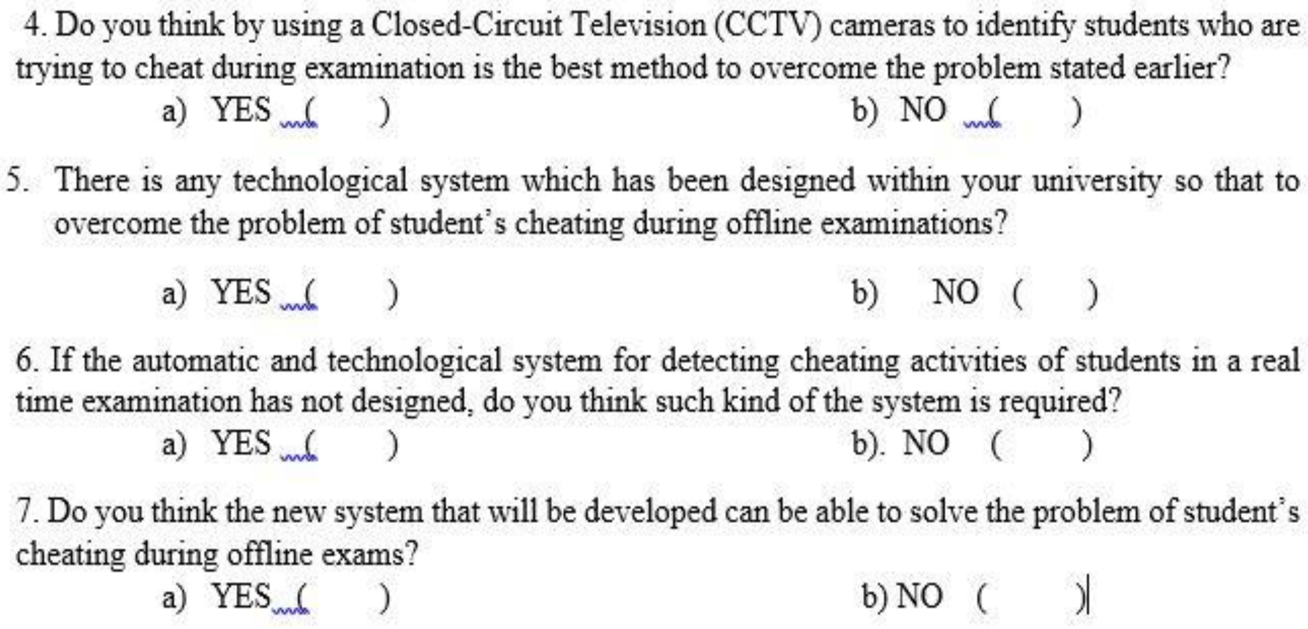
   b) Face and eye movements from one side to another side ( )

   c) Movement from one position to another position ( )

   d) By Speaking during examinations ( )

3. Do you think manual or traditional method of identifying students who attempt to cheat during examination is an efficient way to overcome the problem of student's cheating in offline examinations?

   a) YES ( ) b) NO ( )



4. Do you think by using a Closed-Circuit Television (CCTV) cameras to identify students who are trying to cheat during examination is the best method to overcome the problem stated earlier?

a) YES ( ) b) NO ( )

5. There is any technological system which has been designed within your university so that to overcome the problem of student's cheating during offline examinations?

a) YES ( ) b) NO ( )

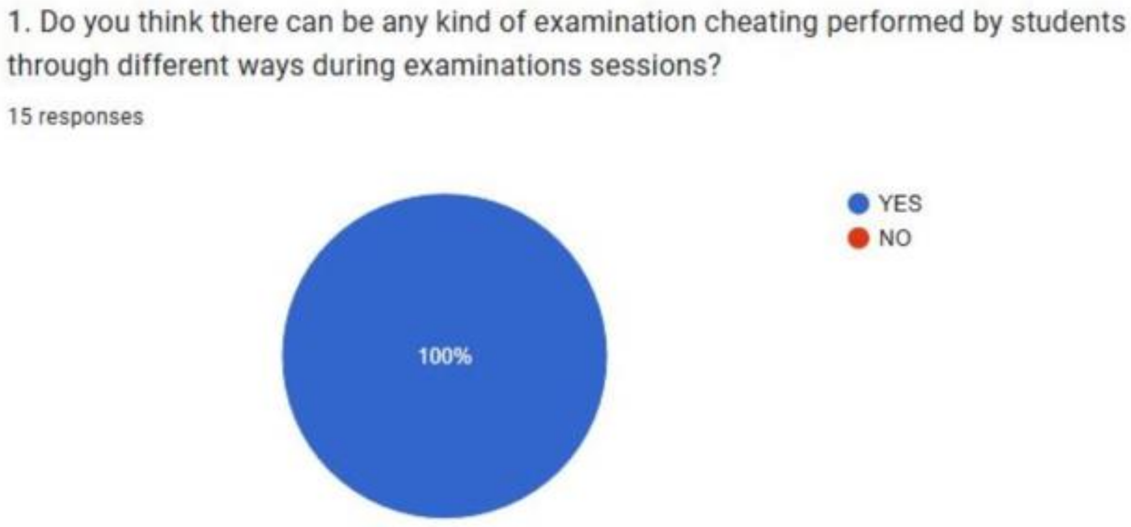
6. If the automatic and technological system for detecting cheating activities of students in a real time examination has not designed, do you think such kind of the system is required?

a) YES ( ) b). NO ( )

7. Do you think the new system that will be developed can be able to solve the problem of student's cheating during offline exams?

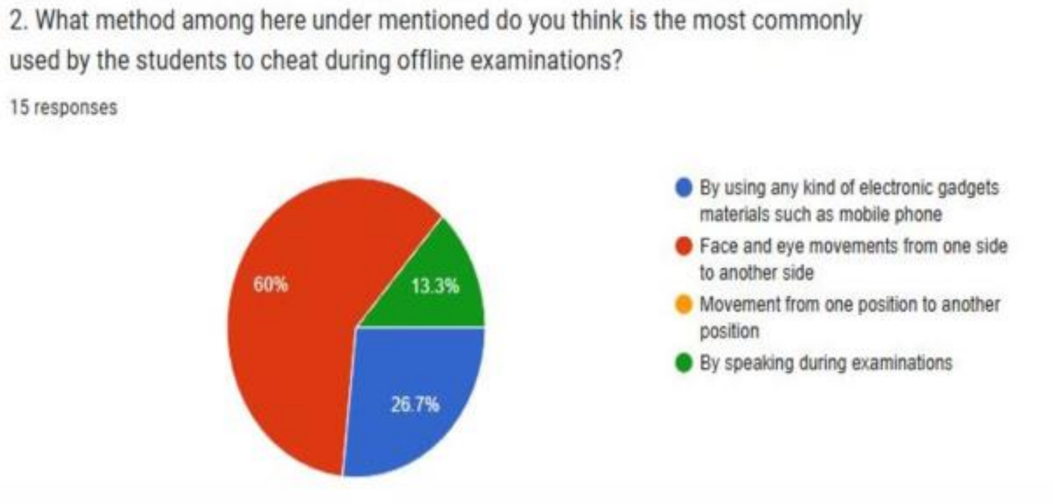
a) YES ( ) b) NO ( )

4.1.3. Response to Questionnaire Questions from the Respondents



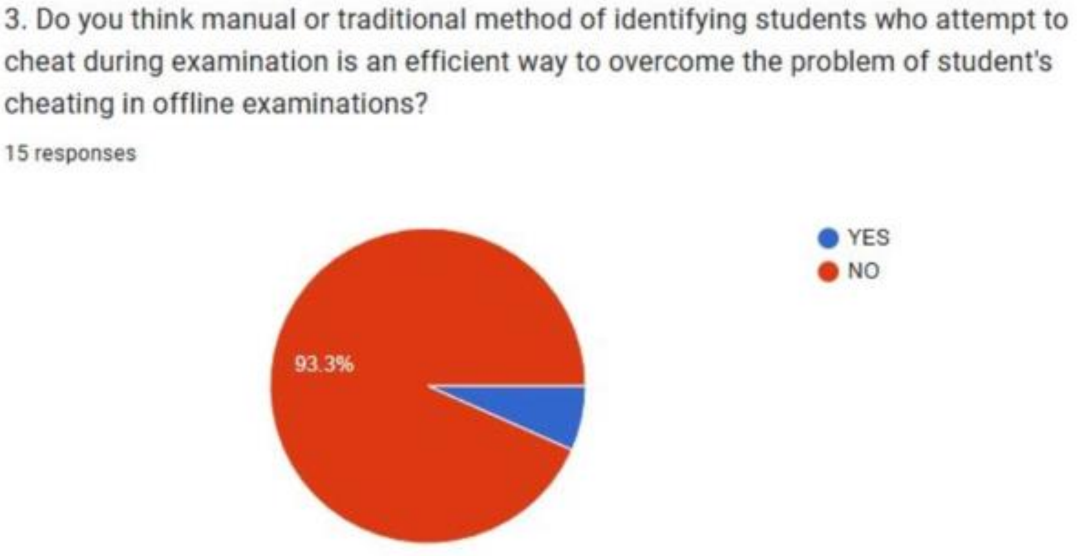
1. Do you think there can be any kind of examination cheating performed by students through different ways during examinations sessions?

15 responses



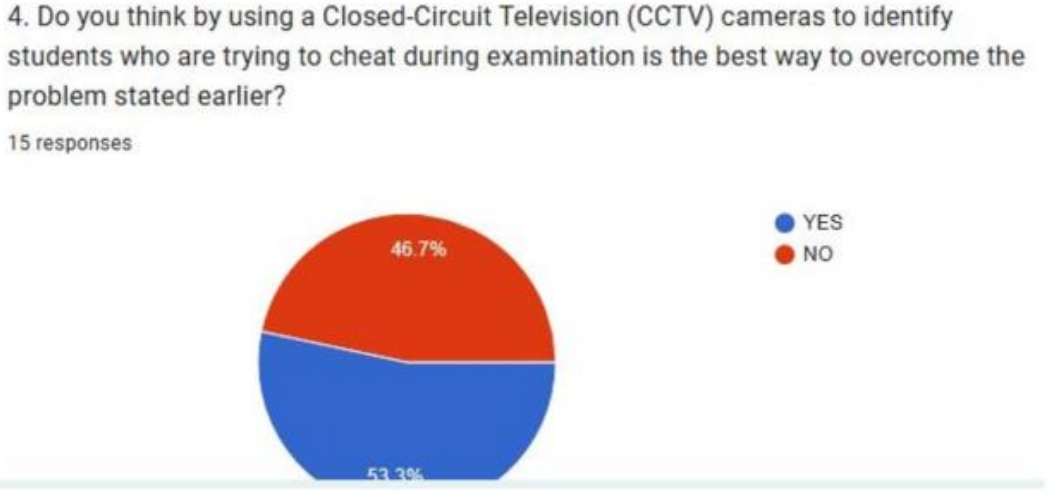
2. What method among here under mentioned do you think is the most commonly used by the students to cheat during offline examinations?

15 responses



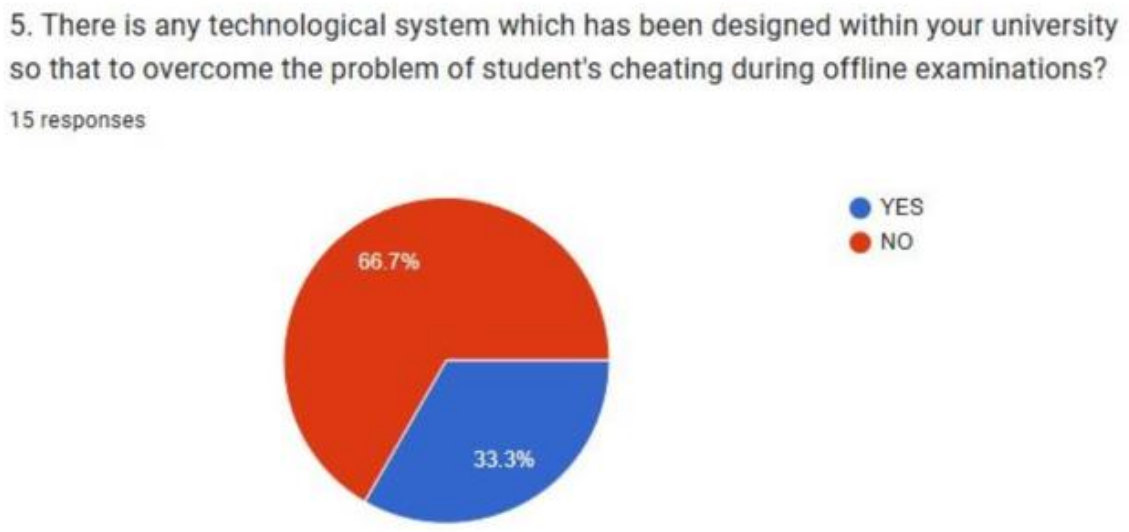
3. Do you think manual or traditional method of identifying students who attempt to cheat during examination is an efficient way to overcome the problem of student's cheating in offline examinations?

15 responses



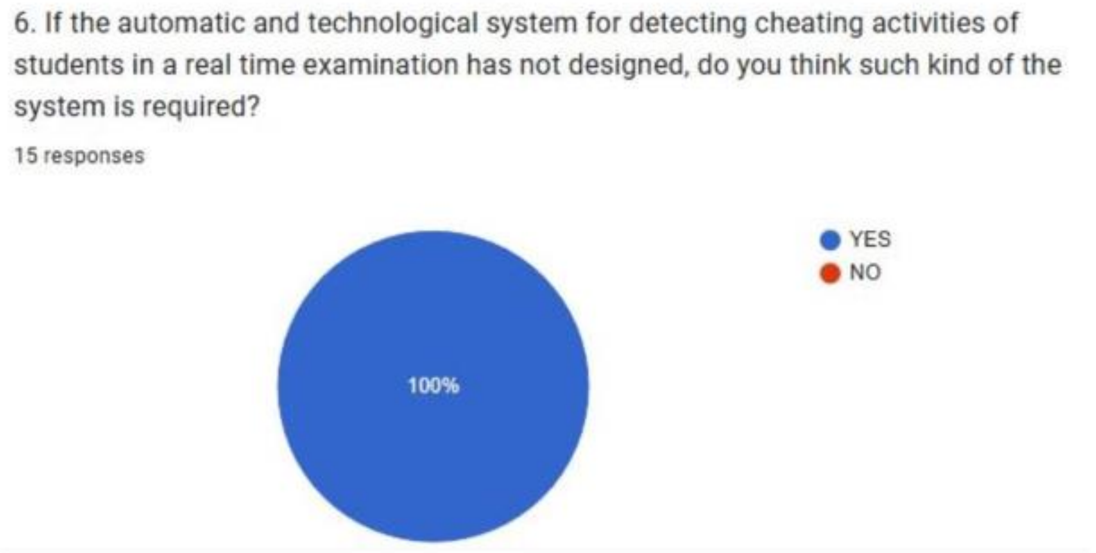
4. Do you think by using a Closed-Circuit Television (CCTV) cameras to identify students who are trying to cheat during examination is the best way to overcome the problem stated earlier?

15 responses



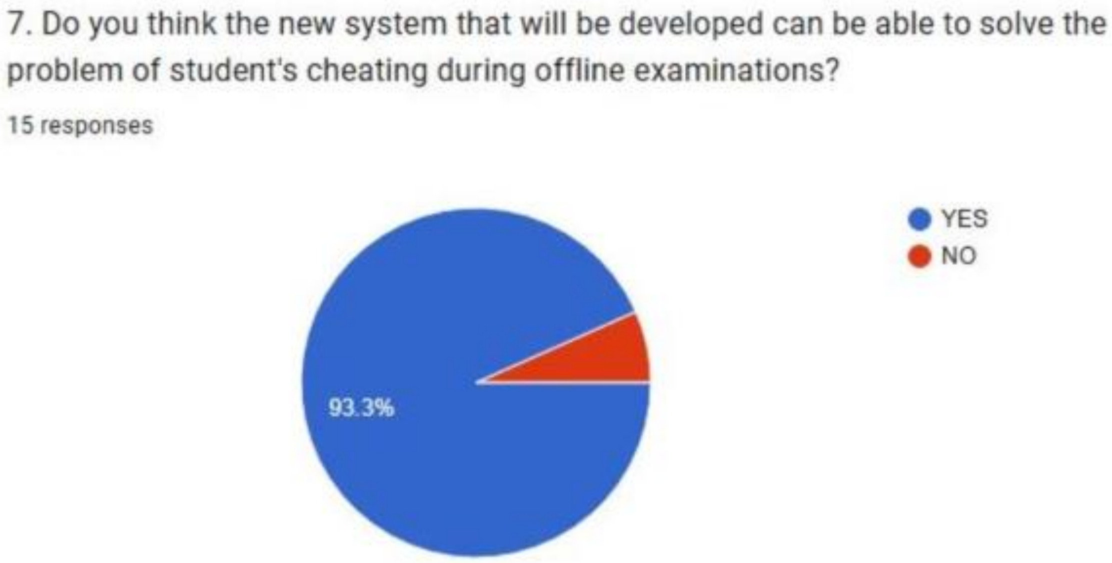
5. There is any technological system which has been designed within your university so that to overcome the problem of student's cheating during offline examinations?

15 responses



6. If the automatic and technological system for detecting cheating activities of students in a real time examination has not designed, do you think such kind of the system is required?

15 responses



7. Do you think the new system that will be developed can be able to solve the problem of student's cheating during offline examinations?

15 responses

4.1.4. Secondary data collection

Secondary data collection, on the other hand, is referred to as the gathering of second-hand data collected by an individual who is not the original user. It is the process of collecting data that is already existing, whether published books, journals, and/or online portals.

In terms of ease, it is much less expensive and easier to collect. Through it the data from internet, books and other journals will be gathered so that to be used whenever needed in this project.

In secondary data collection, the following were collected;

- ❖ Libraries that will be used for Smart Invigilation System

  By passing through different articles and projects done by the other people which are concerning with Smart Invigilation System, the following are libraries that are used for making this kind of this project.

  1. Historical Oriented Gradient (HOG)
  2. DLIB
  3. Facenet-Pytorch
  4. Multi-Task Cascaded Convolutional Neural Network (MTCNN)
  5. Open Source Computer Vision Library (OPENCV)
- ❖ Programming language that will be used for Smart Invigilation System
  1. Python programming language

Disadvantage of using DLIB

The major drawback of using DLIB is that, it does not detect small faces, as it is trained for a minimum face size of 80 X 80.

Disadvantage of using HOG

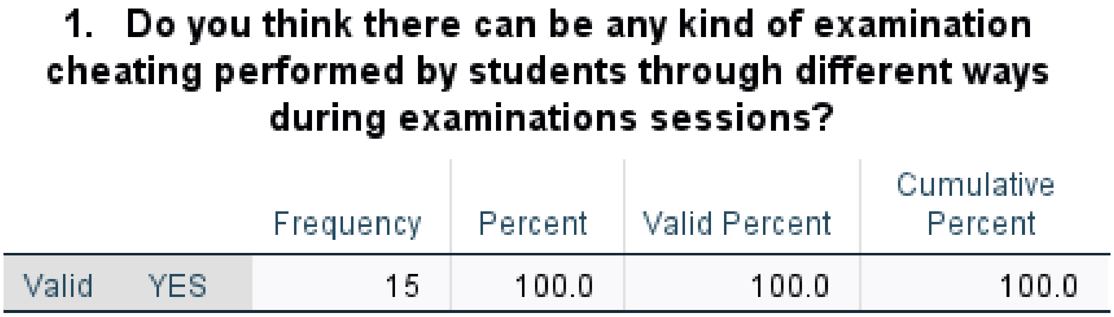
The computation speed is tardy while detecting an object for large-scaled images. Hence, the accuracy is not highly reliable compared to the current convolutional neural networks.

4.2. DATA ANALYSIS

Data analysis: is the process of systematically applying statistical and logical techniques to describe and illustrate and evaluate data.

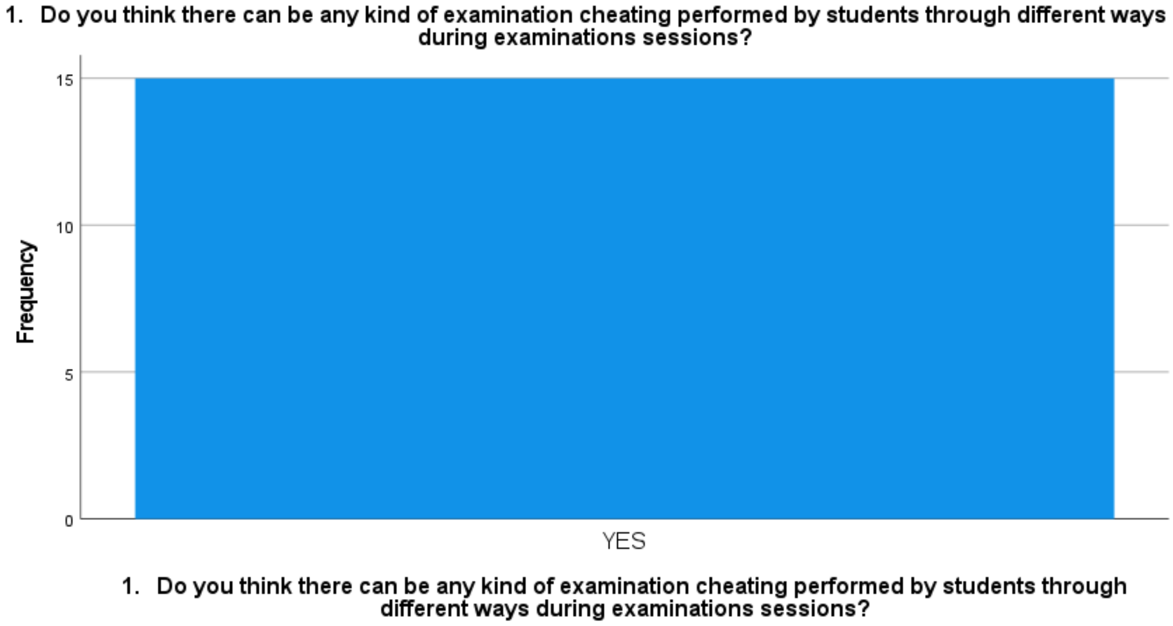
In this project SPSS was used to analyze primary data in tabular form and graphically.

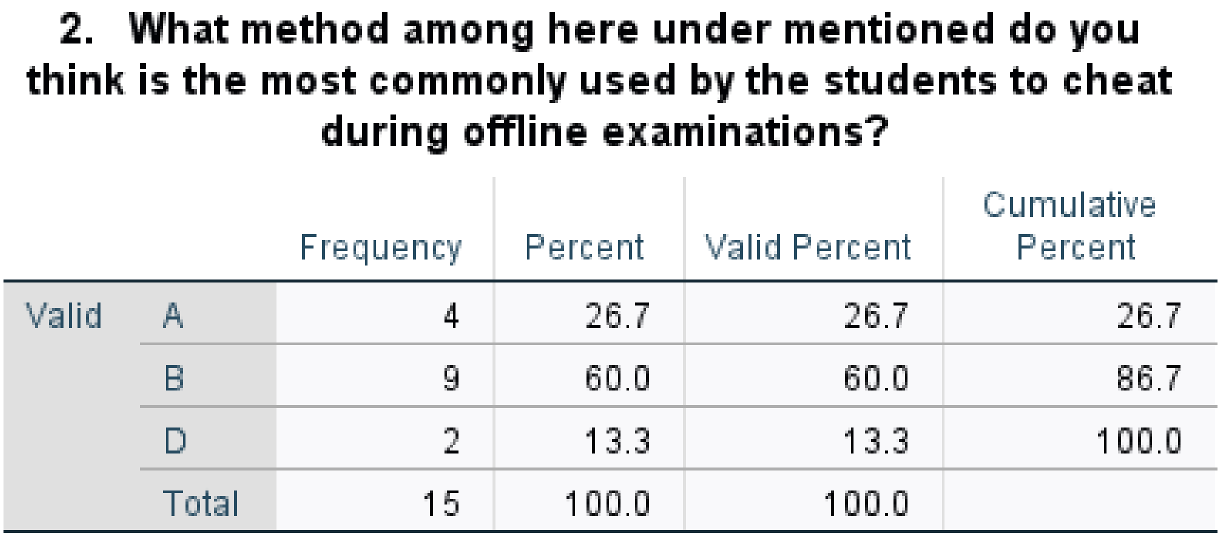
4.2.1. Primary Data Analysis in Tabular and Graphical Forms



1. Do you think there can be any kind of examination cheating performed by students through different ways during examinations sessions?

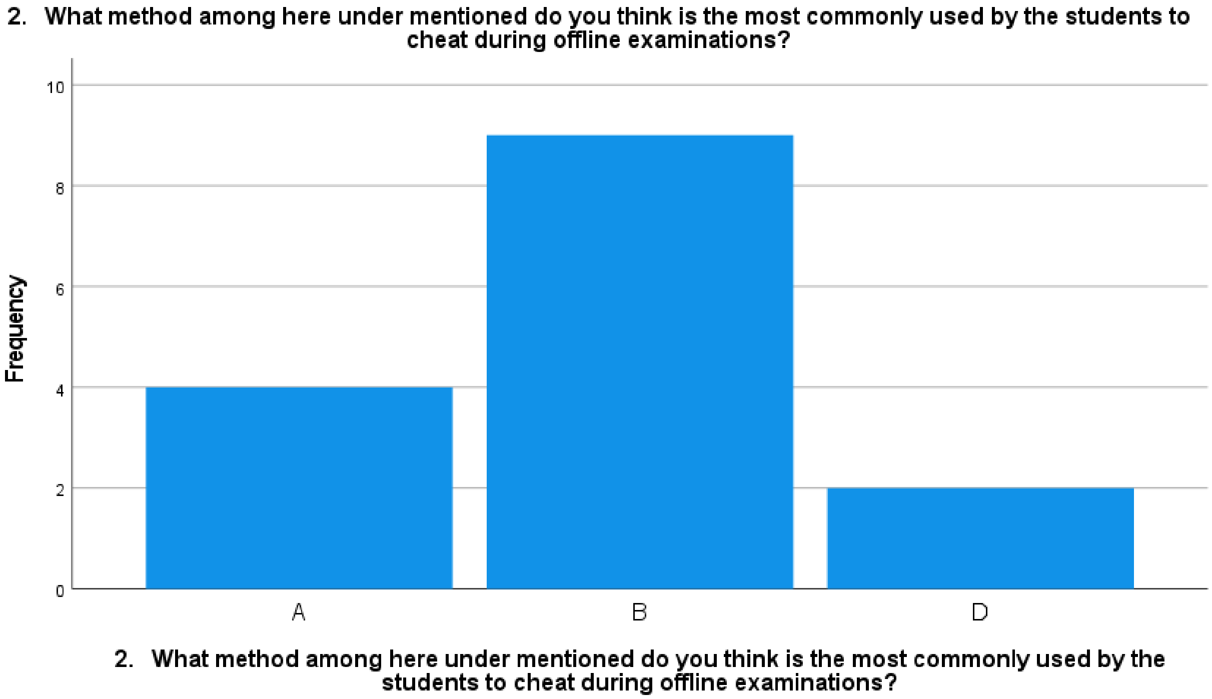
		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	YES	15	100.0	100.0	100.0

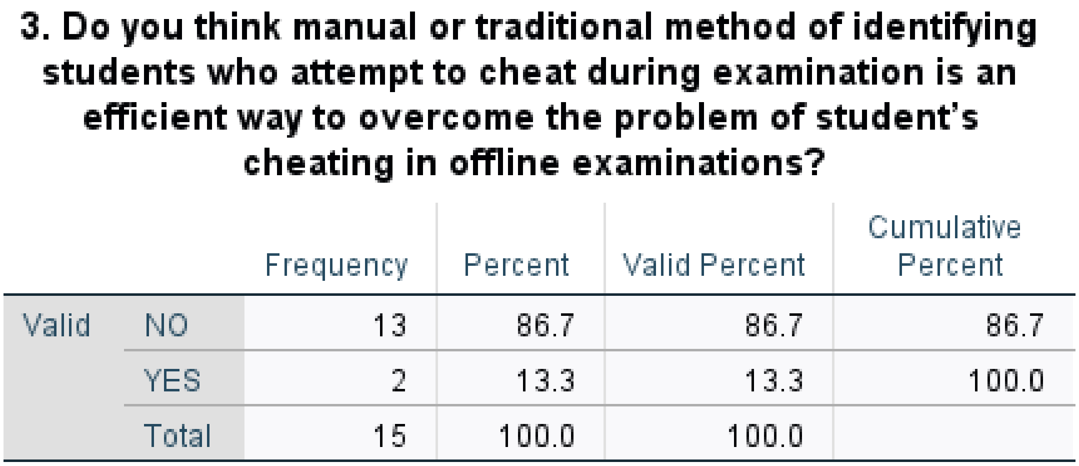




2. What method among here under mentioned do you think is the most commonly used by the students to cheat during offline examinations?

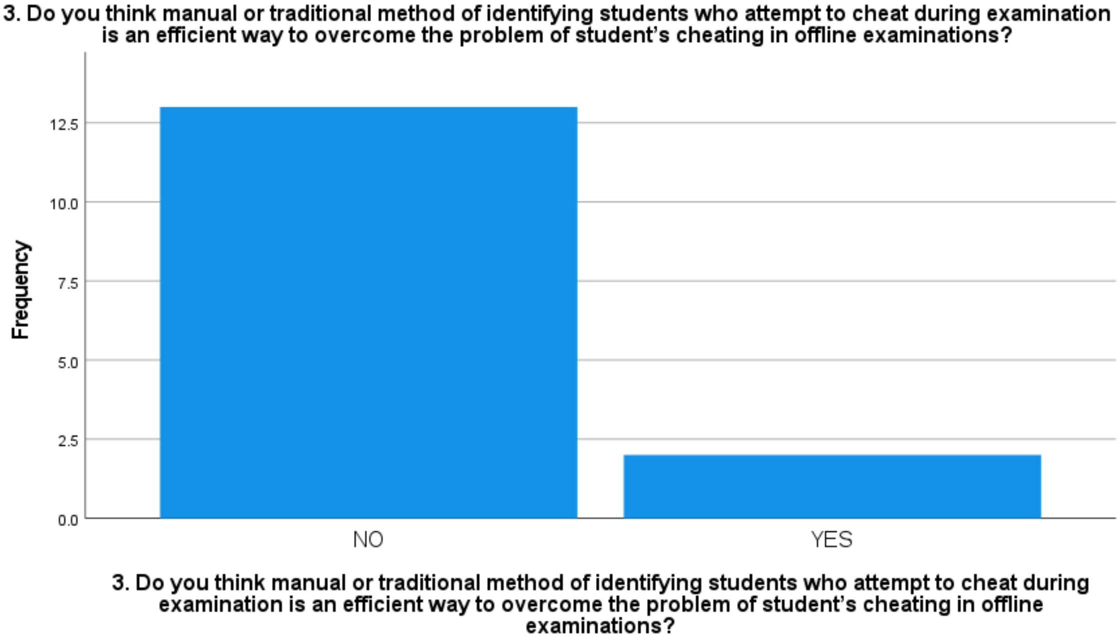
		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	A	4	26.7	26.7	26.7
	B	9	60.0	60.0	86.7
	D	2	13.3	13.3	100.0
	Total	15	100.0	100.0	



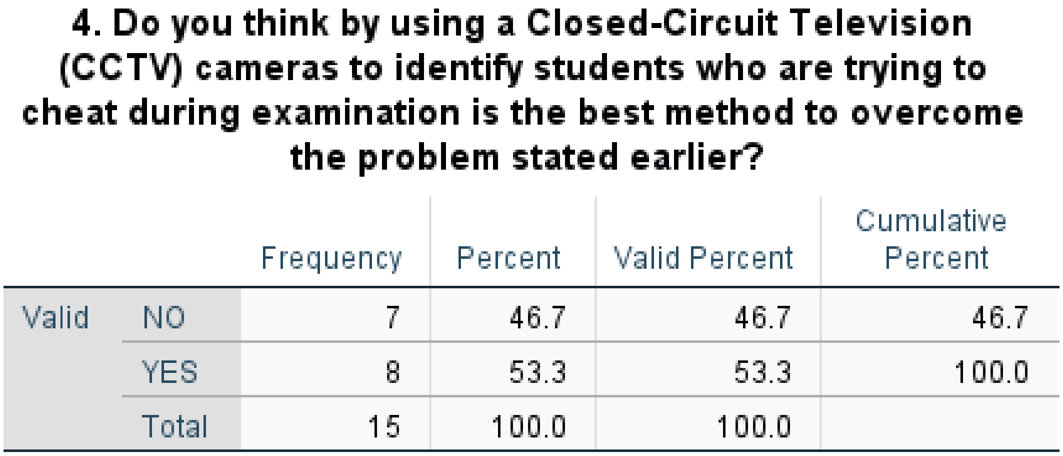


3. Do you think manual or traditional method of identifying students who attempt to cheat during examination is an efficient way to overcome the problem of student's cheating in offline examinations?

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	NO	13	86.7	86.7	86.7
	YES	2	13.3	13.3	100.0
	Total	15	100.0	100.0	

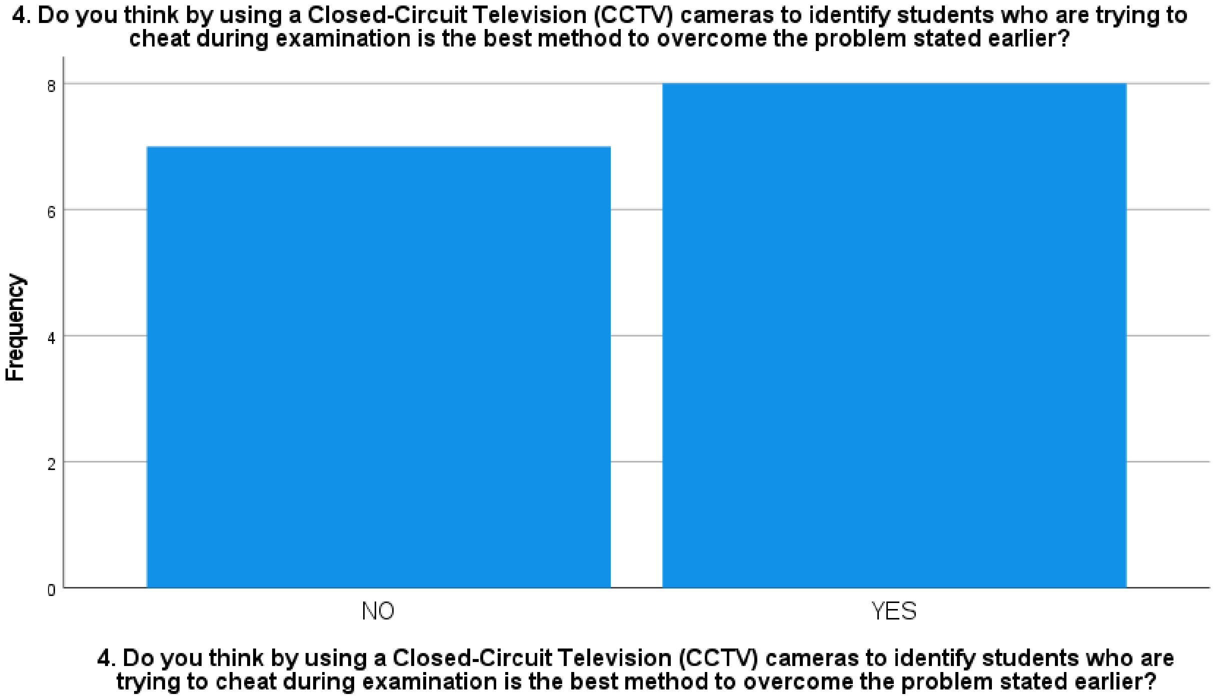


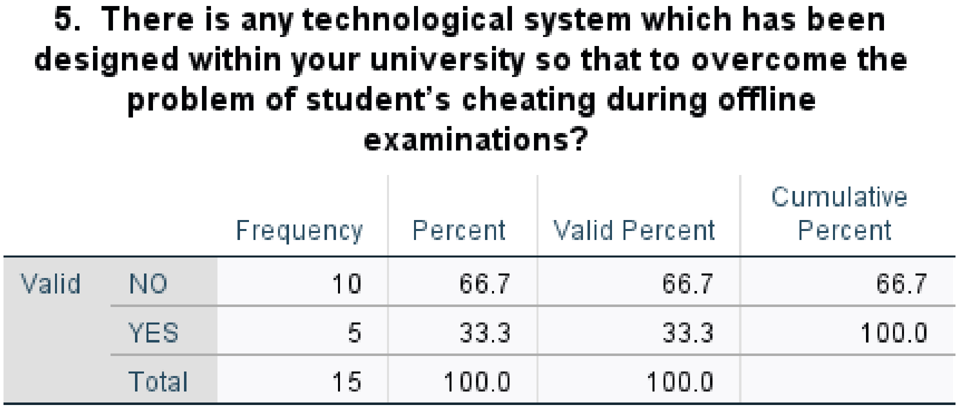
3. Do you think manual or traditional method of identifying students who attempt to cheat during examination is an efficient way to overcome the problem of student's cheating in offline examinations?



4. Do you think by using a Closed-Circuit Television (CCTV) cameras to identify students who are trying to cheat during examination is the best method to overcome the problem stated earlier?

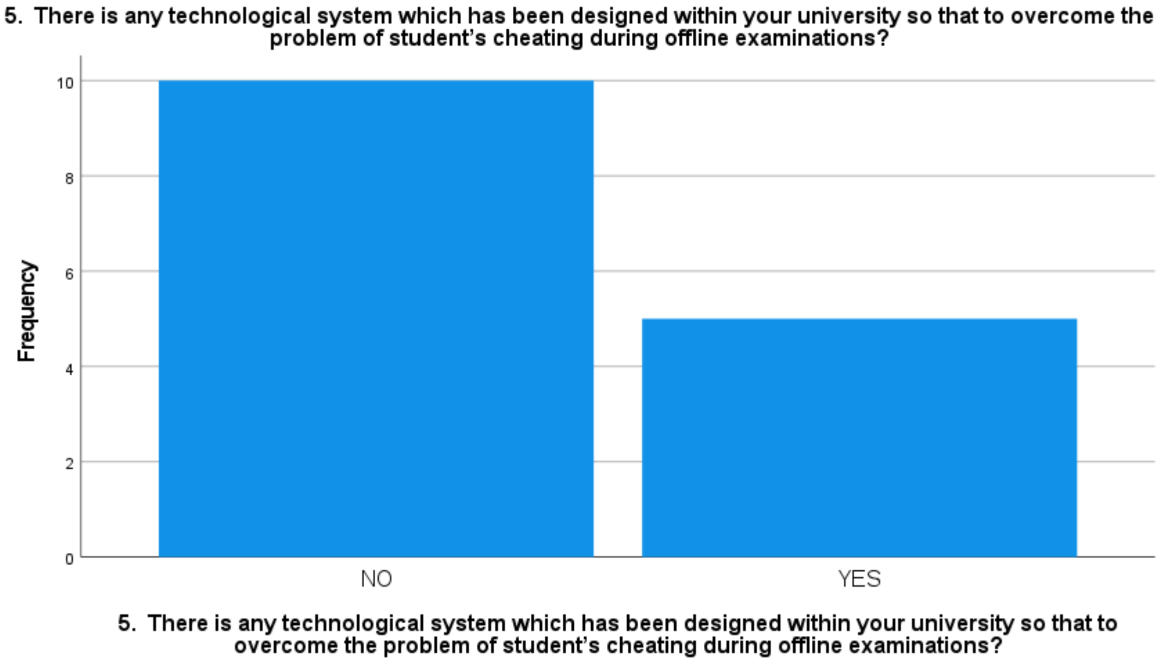
		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	NO	7	46.7	46.7	46.7
	YES	8	53.3	53.3	100.0
	Total	15	100.0	100.0	

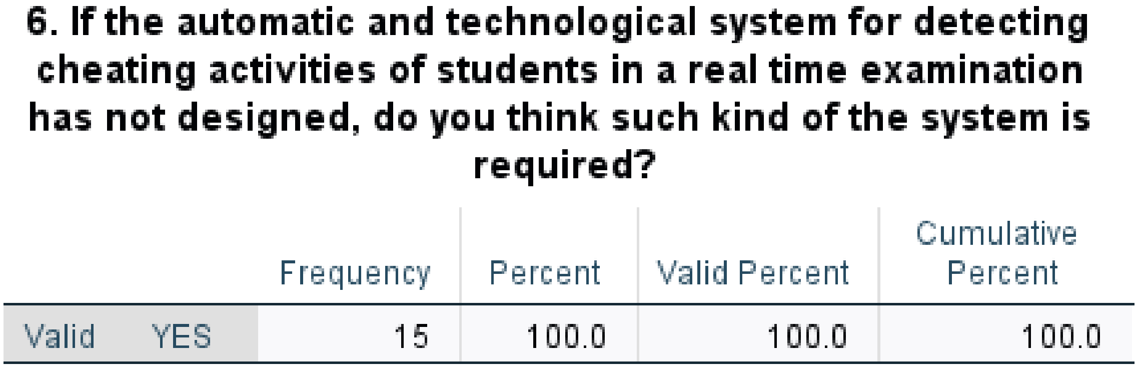




5. There is any technological system which has been designed within your university so that to overcome the problem of student’s cheating during offline examinations?

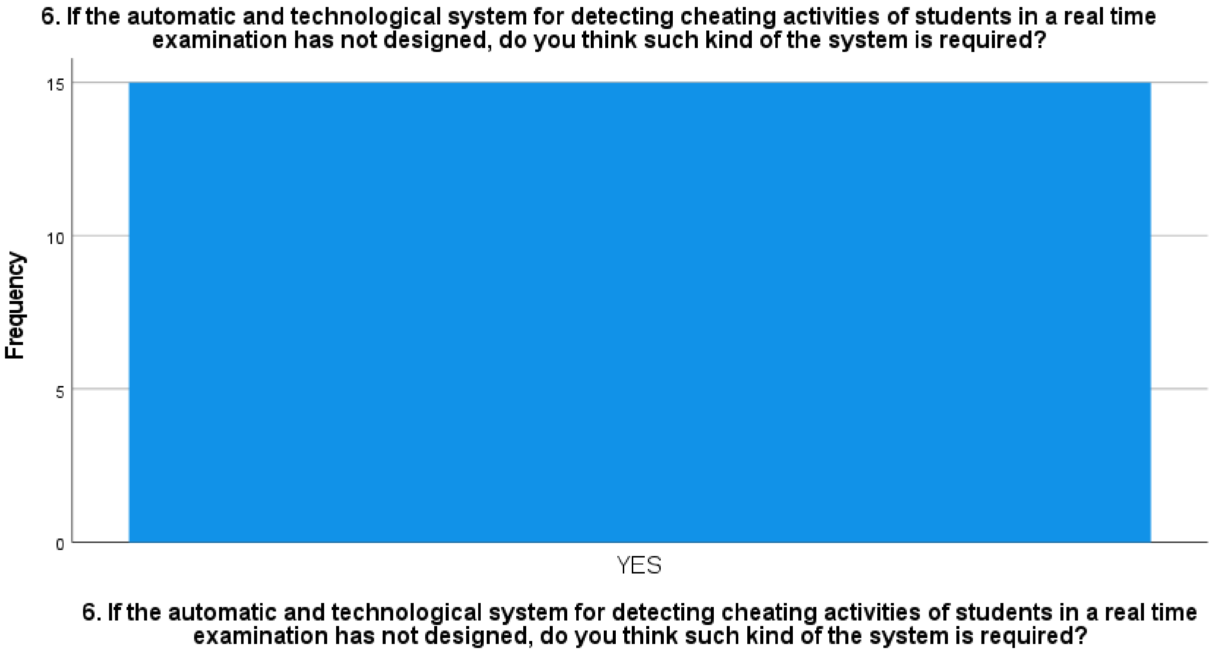
		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	NO	10	66.7	66.7	66.7
	YES	5	33.3	33.3	100.0
	Total	15	100.0	100.0	

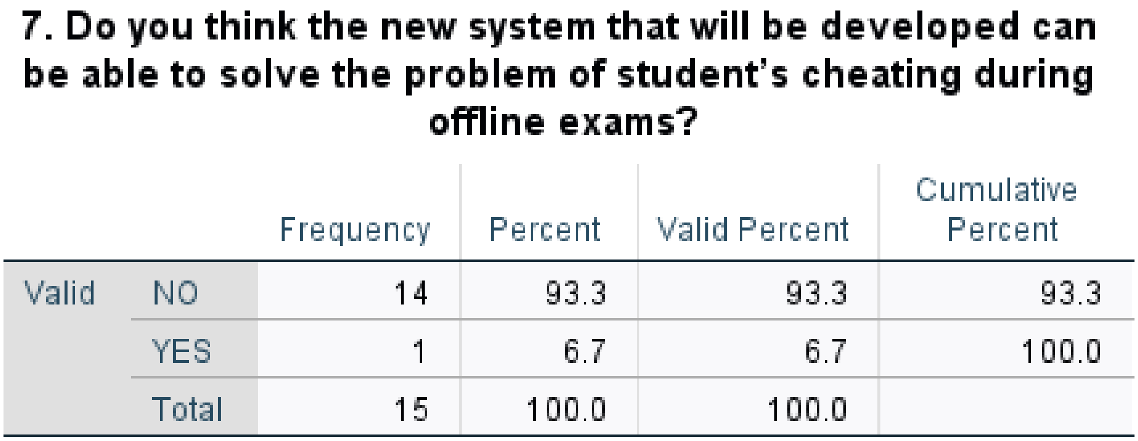




6. If the automatic and technological system for detecting cheating activities of students in a real time examination has not designed, do you think such kind of the system is required?

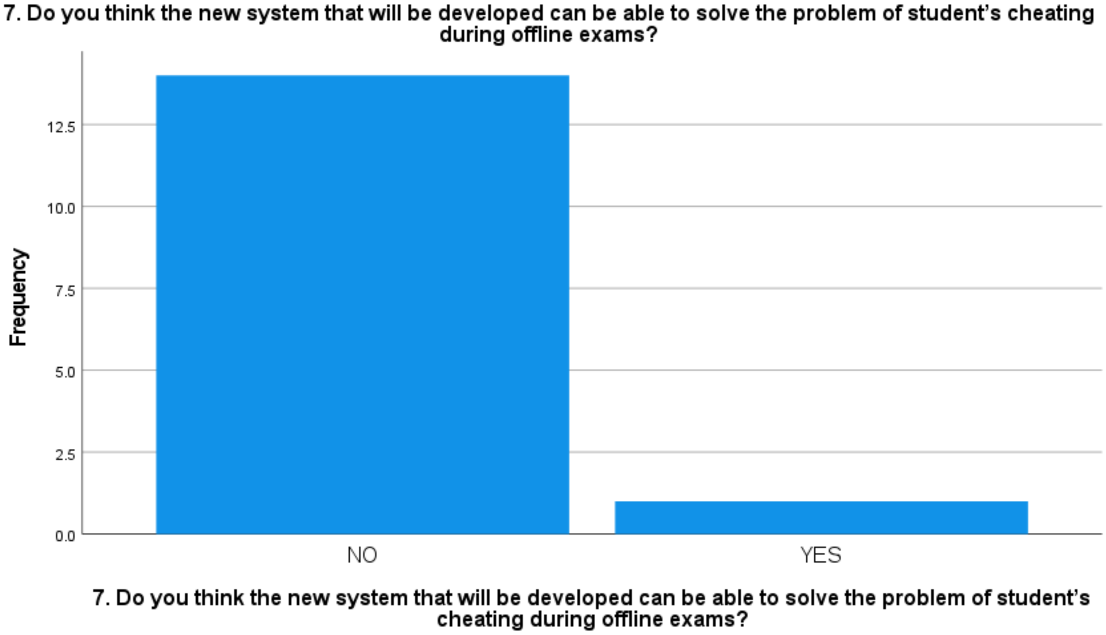
		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	YES	15	100.0	100.0	100.0





7. Do you think the new system that will be developed can be able to solve the problem of student's cheating during offline exams?

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	NO	14	93.3	93.3	93.3
	YES	1	6.7	6.7	100.0
	Total	15	100.0	100.0	



7. Do you think the new system that will be developed can be able to solve the problem of student's cheating during offline exams?

4.2.2. Secondary Data Analysis

- Libraries that will be used for Smart Invigilation System

1. Multi-Task Cascaded Convolutional Neural Network (MTCNN)

This is a deep learning algorithm used for face detection and recognition

Why this kind of technology is selected?

I. MTCNN is very accurate and robust to perform three tasks simultaneously: face detection, landmark localization, and face alignment.

II. MTCNN is able to detect faces at different scales and orientations, making it robust to variations in image quality and lighting conditions.

III. The MTCNN algorithm can be used very well for detecting non-frontal faces.

IV. It is more accurate for face angles calculation.

- Comparison between Detection models and MTCNN algorithm

Models	**Implementation**	**Backbone**	**Avg Time (sec/img)**	**Accuracy**
MTCNN	Keras Pytorch	-	2.4 0.085	96.42% 96.42%
HOG	Dlib & other	-	0.3 0.2	60.71% 71.42%
CNN	Dlib & other	-	0.3 0.2	60.71% 89.28%
Haar Cascade	-	-	0.005	62.5%
RetinaFace	MXNet MXNet Pytorch Pytorch	ResNet MobileNet ResNet MobileNet	1.24 0.515 1.0975 0.205	92.85% 78.57% 62.5% 62.5%

2. PyTorch

Is an open-source machine learning library that is widely used for developing and training deep learning models.

Why this kind of technology is selected?

I. Pytorch supports GPU acceleration, which enables faster training and inference of deep learning models.
II. Pytorch provides automatic differentiation, which simplifies the process of computing gradients for optimizing model parameters.
III. Pytorch is built to be Pythonic, which means it integrates well with the Python programming language and can be easily used with other Python libraries.

3. Open-Source Computer Vision (OpenCv)

- Is an open-source computer vision library that provides a wide range of image and video processing functions.

Why this kind of technology is selected?

I. Cross-platform support: OpenCv is available on various platforms including Windows, Linux, macOS, Android,and iOS.
II. Machine learning support: It includes machine learning algorithms like support vector machines (SVM), K-nearest neighbors (KNN), and neural networks.
III. OpenCv is designed to be highly optimized, making it an ideal choice for real-time computer vision applications.

- Devices that will be used

Component	**Specification**
Processor	I5, i7 or i9 Processor (GPU)
Operating System	Windows 7, 8, 10 or Windows 11 Home, Pro, Enterprise or Education version
Memory (RAM)	8 GB of RAM or larger
Storage	500 GB solid state drive, or larger
Webcam	Either external USB device or built-in
Network	802.11ac (or better) Wi-Fi capability

- Language that will be used for Smart Invigilation System

1. Python Programming Language

Why this language is selected?

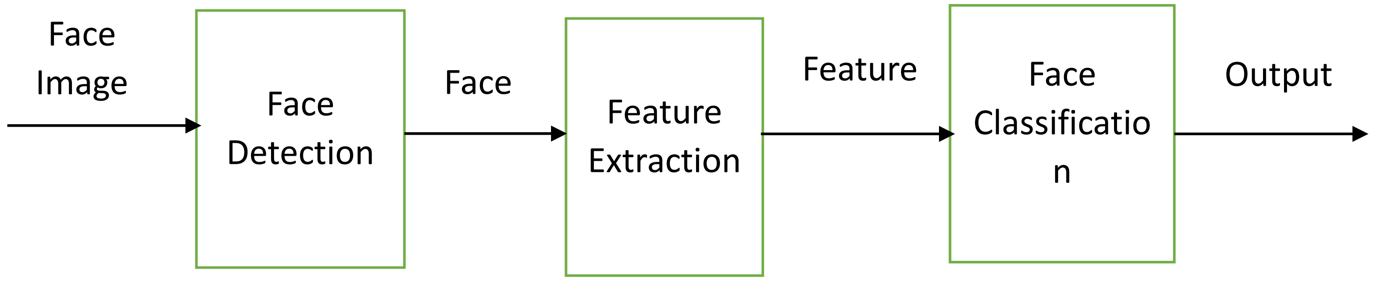
I. Python supports object-oriented programming, which allows for the creation of reusable code and modular design.

II. Cross Platform: Python can run on a variety of operating systems, including Windows, macOS, and Linux.

III. Python has a vast ecosystem of third-party libraries and packages that extend its functionality for specific use cases.

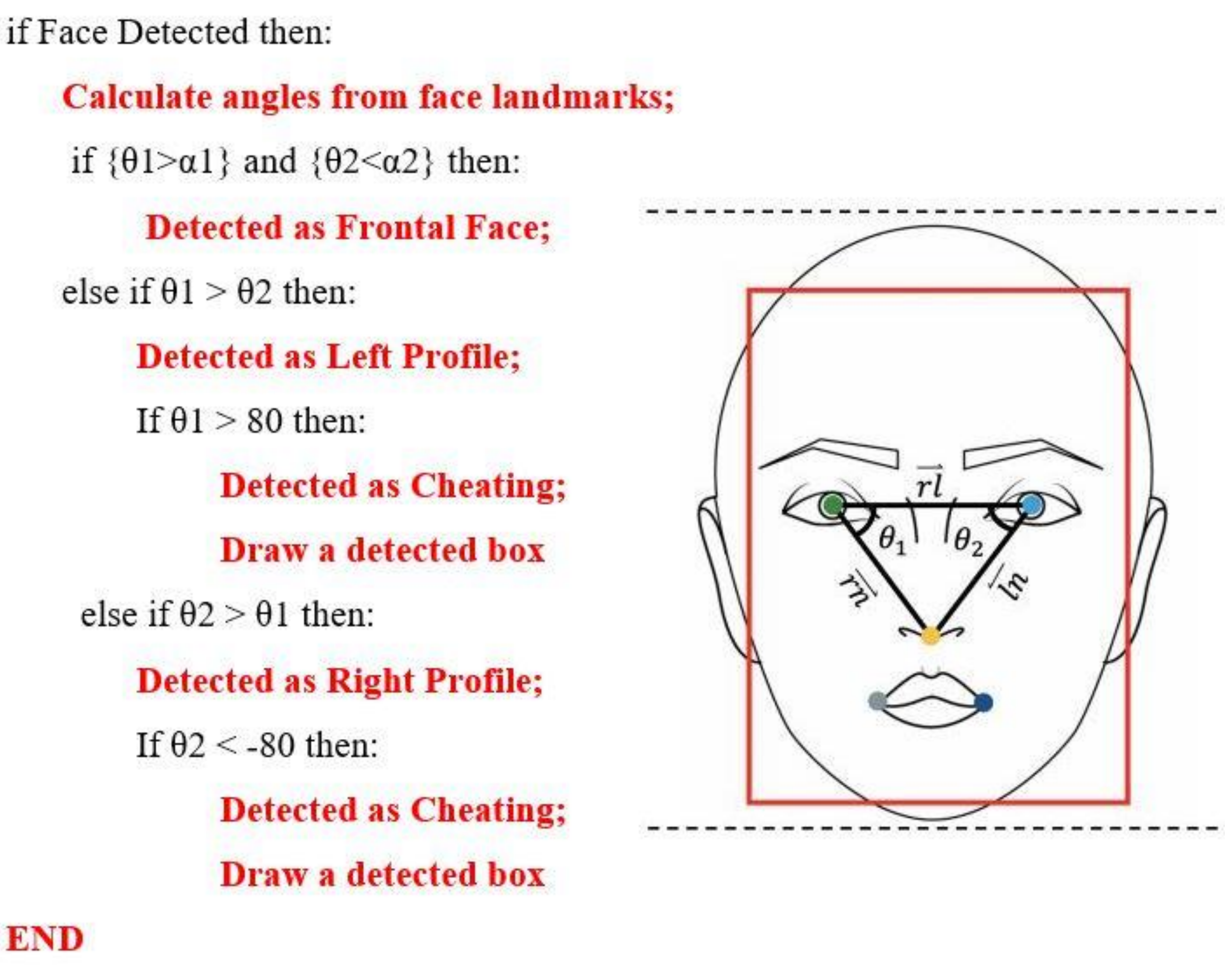
5. System Design

5.1. Cheating detection pipeline



1. Face Detection: Here one or more faces are detected from a camera.
2. Feature Extraction: In this step the most important features are extracted from an image of the face.
3. Face Classification: Face is classified based on extracted features.

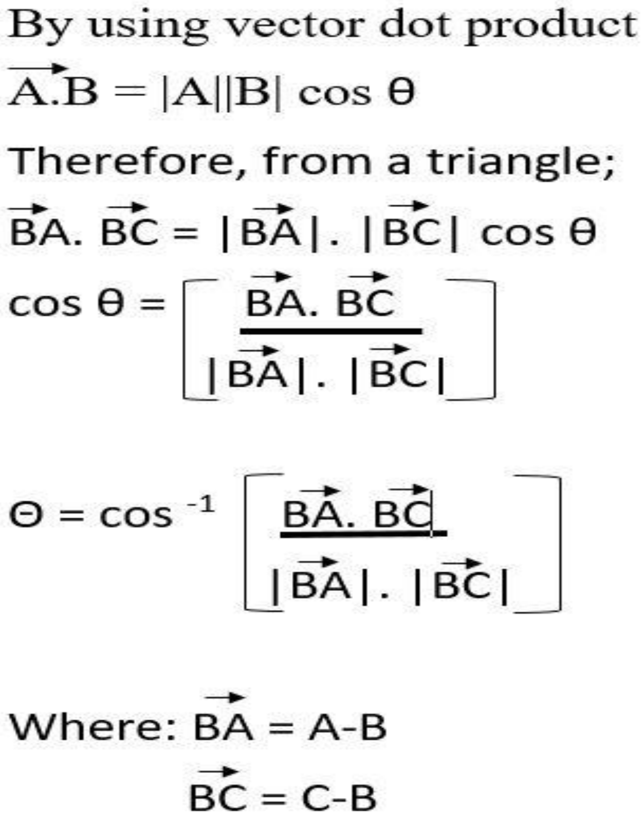
5.2. To classify a face based on extracted features



```
if Face Detected then:
    Calculate angles from face landmarks;
    if {θ1>α1} and {θ2<α2} then:
        Detected as Frontal Face;
    else if θ1 > θ2 then:
        Detected as Left Profile;
        If θ1 > 80 then:
            Detected as Cheating;
            Draw a detected box
    else if θ2 > θ1 then:
        Detected as Right Profile;
        If θ2 < -80 then:
            Detected as Cheating;
            Draw a detected box
END
```

5.3. Angle Calculation

- In this part, we see how we can use the angles to find the face pose or orientation. Face orientation, or pose, is determined by 2 angles, θ1 and θ2.
- After the model detected the face and the landmarks, then the next step is to calculate the angle between the vector ||rl|| and ||rn||, similarly between the vector ||rl|| and ||ln||.
- Hence, we produced both θ1 and θ2. After calculating the angles, θ1 and θ2 act as an input to the algorithm, if the value of θ1 and θ2 between certain thresholds then we consider the detected face as frontal pose. Otherwise if θ1 < θ2 then it is left profile and the opposite is the right profile



By using vector dot product

$$\overrightarrow{A.}B = |A||B| \cos \theta$$

Therefore, from a triangle;

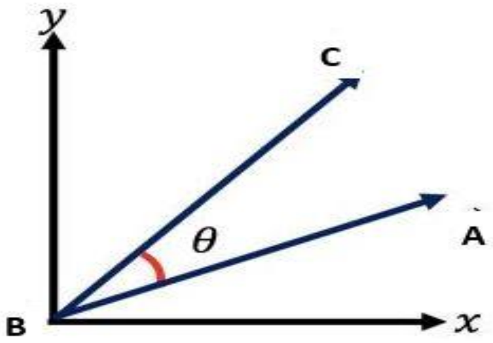
$$\overrightarrow{BA}.\ \overrightarrow{BC} = |\overrightarrow{BA}|.\ |\overrightarrow{BC}| \cos \theta$$

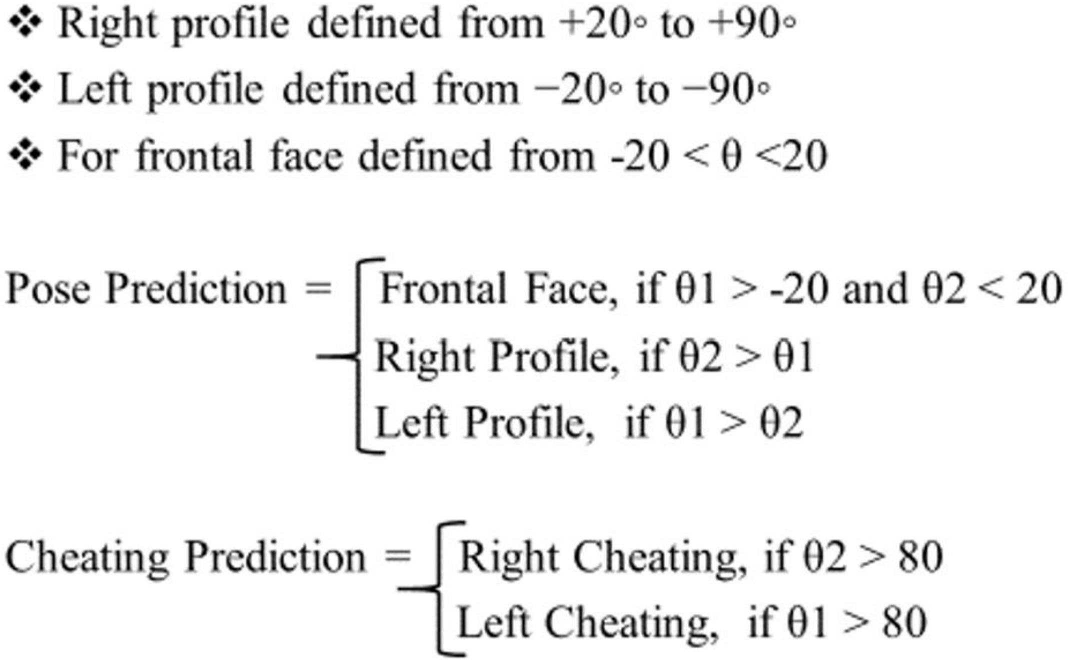
$$\cos \theta = \left[\frac{\overrightarrow{BA}.\ \overrightarrow{BC}}{|\overrightarrow{BA}|.\ |\overrightarrow{BC}|} \right]$$

$$\Theta = \cos^{-1} \left[\frac{\overrightarrow{BA}.\ \overrightarrow{BC}}{|\overrightarrow{BA}|.\ |\overrightarrow{BC}|} \right]$$

Where: $\overrightarrow{BA} = A-B$

$\overrightarrow{BC} = C-B$





- ❖ Right profile defined from +20◦ to +90◦
- ❖ Left profile defined from −20◦ to −90◦
- ❖ For frontal face defined from $-20 < \theta < 20$

$$\text{Pose Prediction} = \begin{cases} \text{Frontal Face, if } \theta 1 > -20 \text{ and } \theta 2 < 20 \\ \text{Right Profile, if } \theta 2 > \theta 1 \\ \text{Left Profile, if } \theta 1 > \theta 2 \end{cases}$$

$$\text{Cheating Prediction} = \begin{cases} \text{Right Cheating, if } \theta 2 > 80 \\ \text{Left Cheating, if } \theta 1 > 80 \end{cases}$$

5.4. System Flow Chart

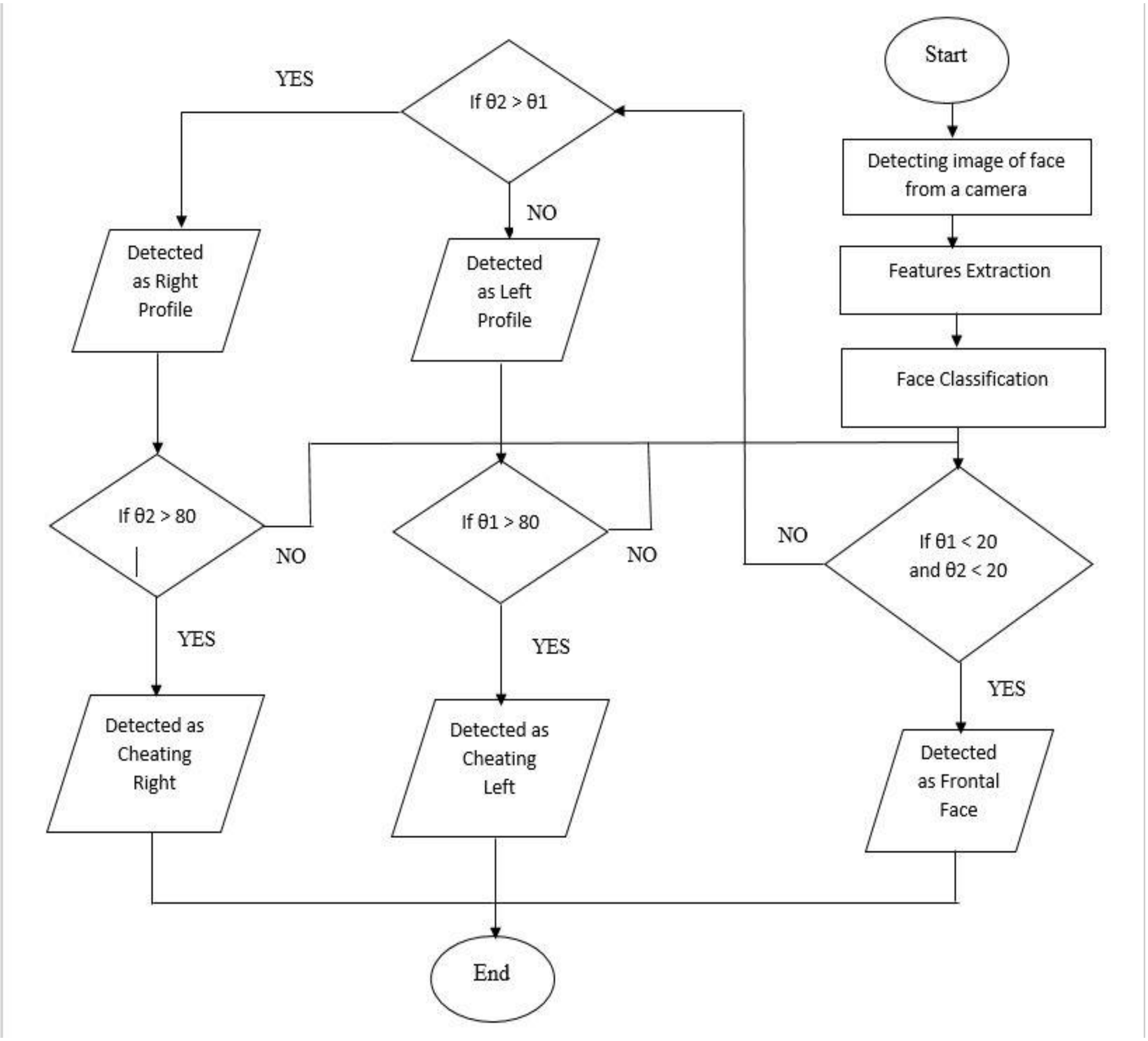


Figure 5.4: System Flow Chart

5.5. System Usecase Diagram

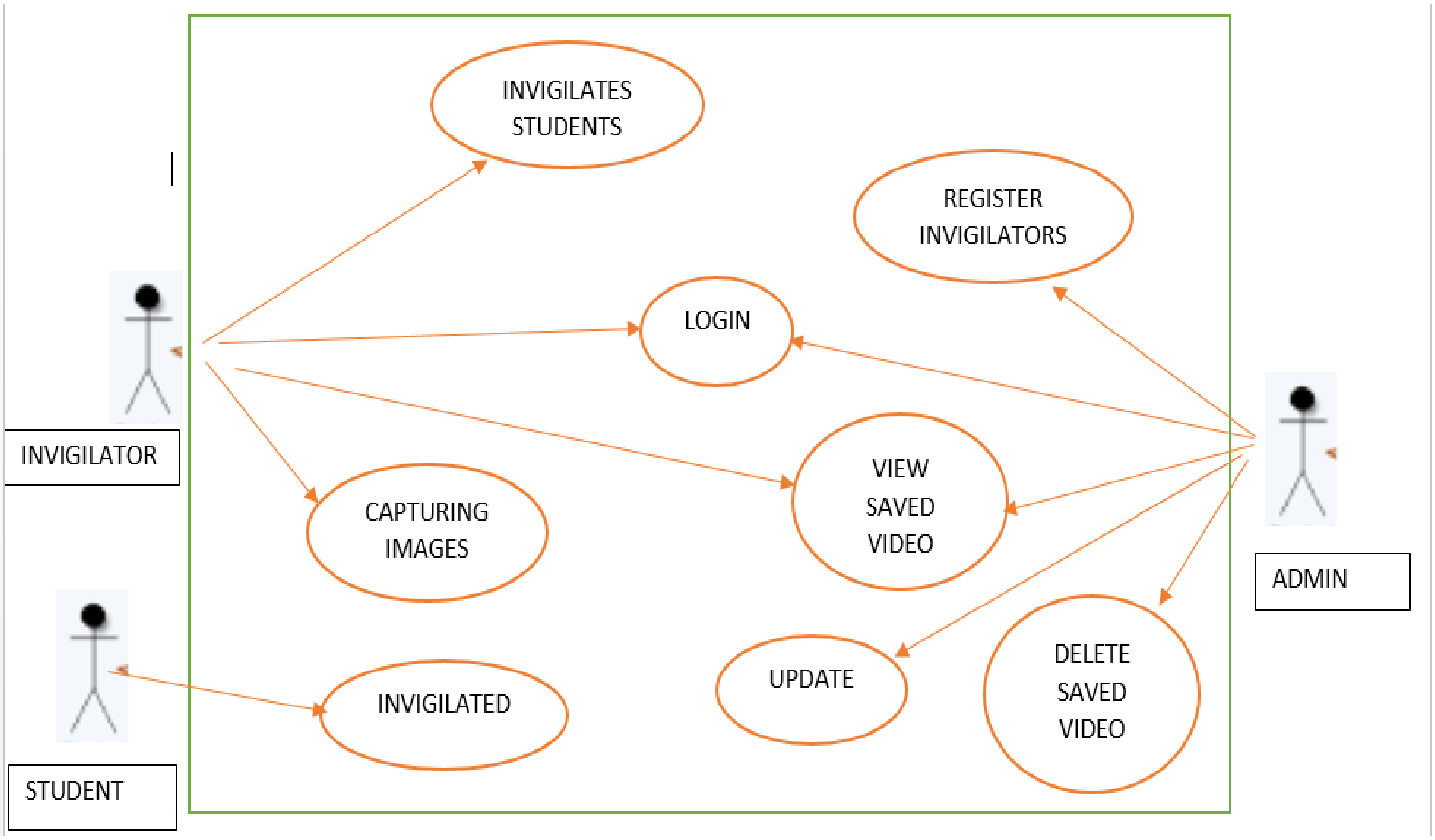


Figure 5.5: System UseCase Diagram

CHAPTER FIVE

6. Simulation

A Smart Invigilation System by using Multi-Task Cascaded Convolutional Neural Network (MTCNN) can be simulated and prototyped using various programming languages and frameworks.

The Smart Invigilation System involves the following steps;

1. Designing of Invigilator interface
2. Data collection
3. Training the MTCNN model
4. Developing the invigilation system
5. Testing and refining the system for simulation by using terminal

6.1. Designing of Invigilator Interface

The invigilator interface consists of the following functions

I. Login function
II. Recording Live Video in a real time examination
III. Starting Invigilation process
IV. Help function
V. Logout function

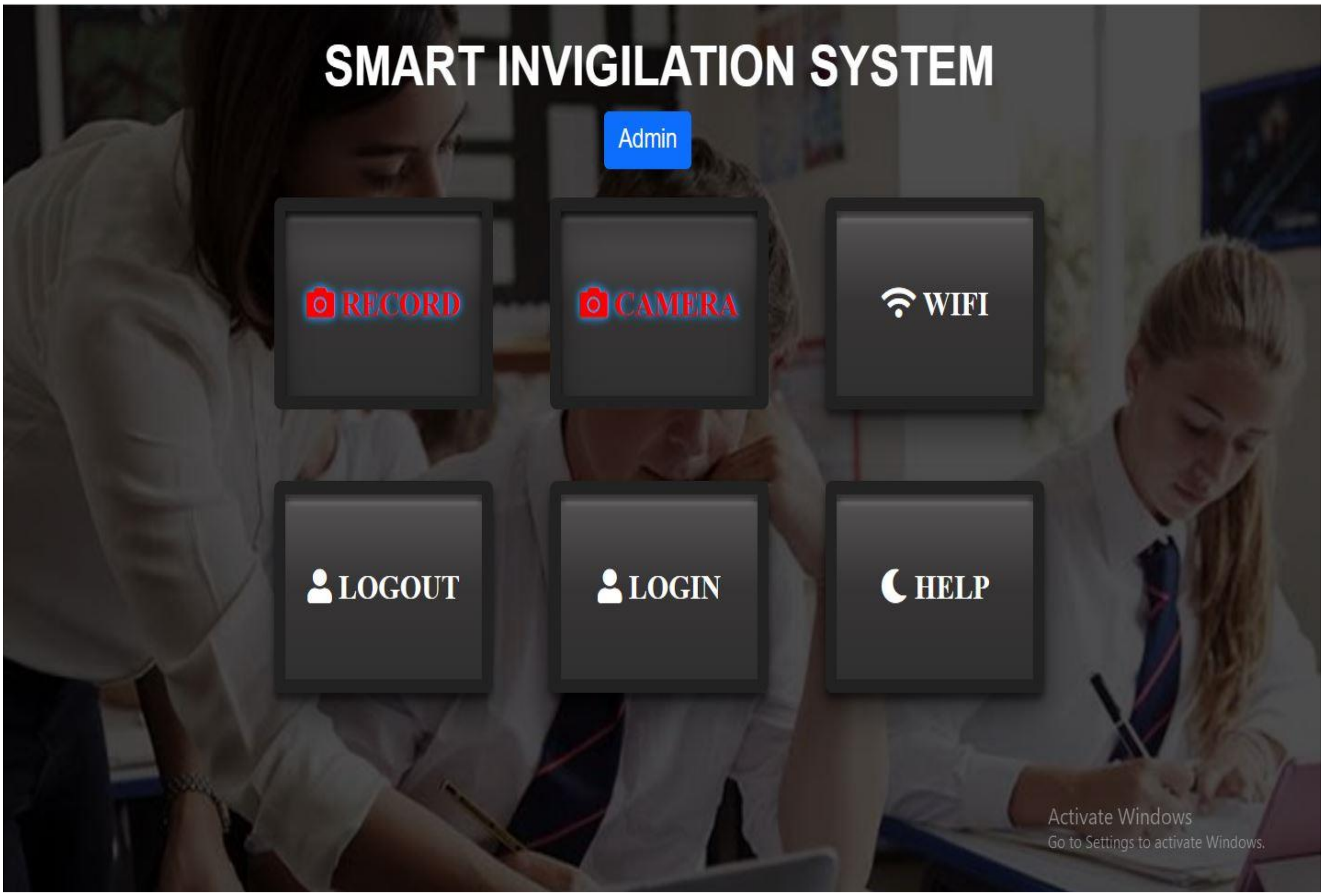

SMART INVIGILATION SYSTEM
Admin
RECORD
CAMERA
WIFI
LOGOUT
LOGIN
HELP
Activate Windows
Go to Settings to activate Windows.

Figure 6.1: Smart invigilation Interface

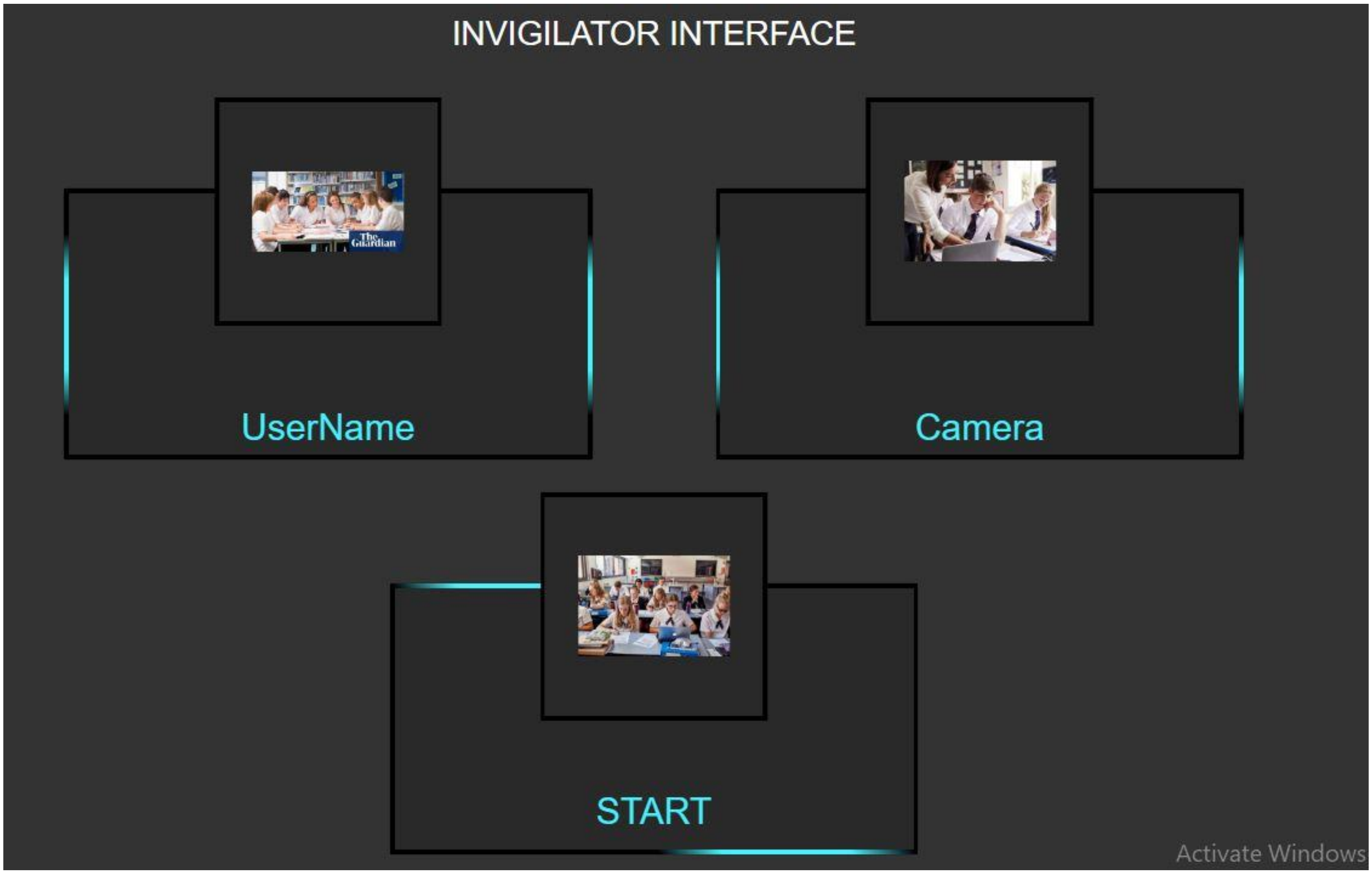


Figure 6.1: Smart invigilation Interface

6.2. Data collection

Here the process of gathering a dataset of images that contain faces for training the MTCNN model is done. This dataset can be obtained through a live video of students in a real time examination.

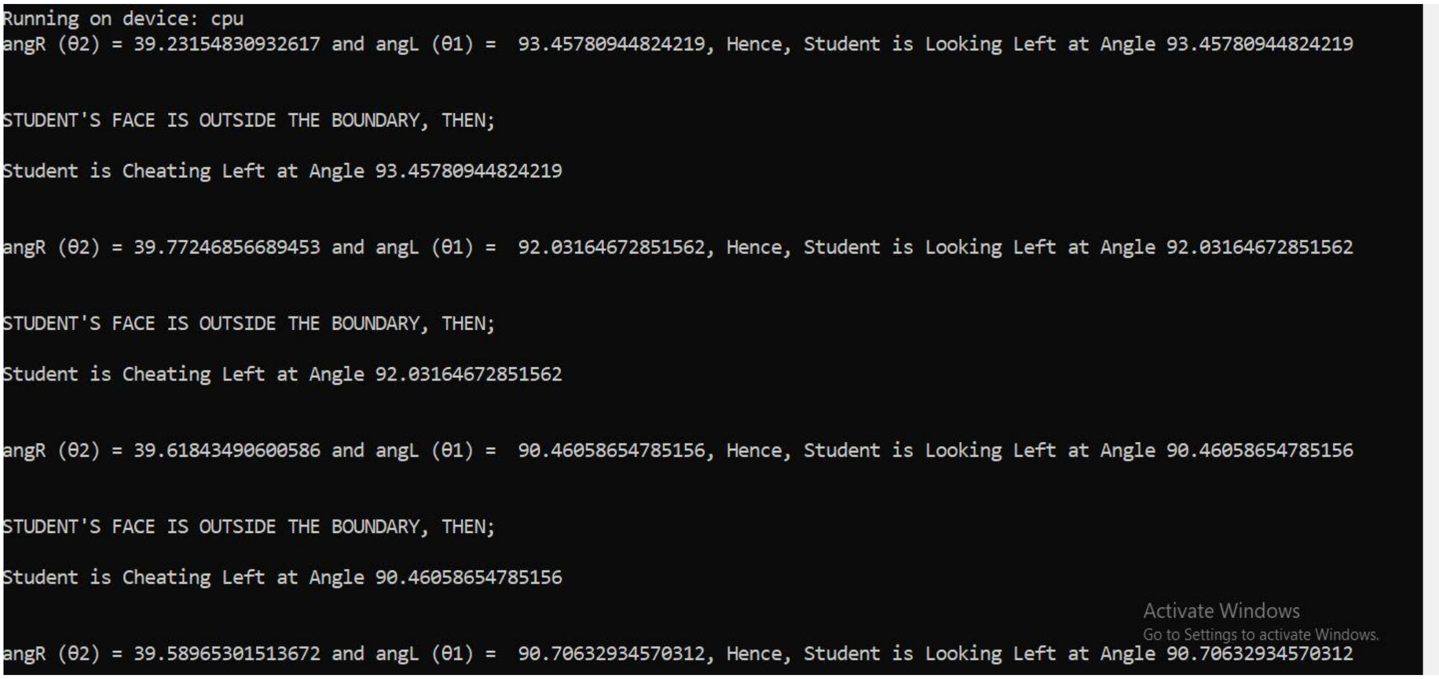
6.3. Training the MTCNN model

By using collected data, the MTCNN model is trained. This can be done using deep learning frameworks that is PyTorch. The trained model should be able to detect faces and extract their features accurately.

6.4. Developing the invigilation system

By using a programming language like Python programming language, the Smart Invigilation System is developed that utilizes the MTCNN model to detect and classify faces in real time video streams. The system is designed to detect any suspicious behavior (students cheating) during exams by detecting face orientation for each student in a real time examination.

6.5. Testing and refining the system for simulation by using terminal



```
Running on device: cpu
angR (θ2) = 39.23154830932617 and angL (θ1) =  93.45780944824219, Hence, Student is Looking Left at Angle 93.45780944824219


STUDENT'S FACE IS OUTSIDE THE BOUNDARY, THEN;

Student is Cheating Left at Angle 93.45780944824219


angR (θ2) = 39.77246856689453 and angL (θ1) =  92.03164672851562, Hence, Student is Looking Left at Angle 92.03164672851562


STUDENT'S FACE IS OUTSIDE THE BOUNDARY, THEN;

Student is Cheating Left at Angle 92.03164672851562


angR (θ2) = 39.61843490600586 and angL (θ1) =  90.46058654785156, Hence, Student is Looking Left at Angle 90.46058654785156


STUDENT'S FACE IS OUTSIDE THE BOUNDARY, THEN;

Student is Cheating Left at Angle 90.46058654785156


angR (θ2) = 39.58965301513672 and angL (θ1) =  90.70632934570312, Hence, Student is Looking Left at Angle 90.70632934570312
```

Figure 6.5: Test and refining the system by using terminal

7. Prototype

The prototype consists of several components working together:

Surveillance Cameras: High-resolution cameras are strategically placed throughout the examination hall to capture video of the entire area. These cameras are allowing invigilators to monitor different sections of the room.

Facial Detection: The system utilizes facial detection technology to identify and verify the identity of each student’s face. During the exam, the cameras continuously capture and analyze the students' faces to detect any suspicious activity basing on face orientation.

Live Streaming and Recording: The video feed from the surveillance cameras is streamed live and recorded. This enables remote invigilators to monitor the examination room in real-time. Additionally, all videos are recorded for later review and analysis if necessary.

Alerts and Notifications: The system generates real-time alerts and notifications for invigilators when potential cheating or irregularities are detected. These alerts can be delivered through a text message, or email, ensuring invigilators can quickly respond to any issues.

The prototype of the smart invigilation system is designed to enhance the efficiency and effectiveness of exam invigilation by combining various technologies, aims to reduce the workload on invigilators while maintaining the integrity of the examination process.

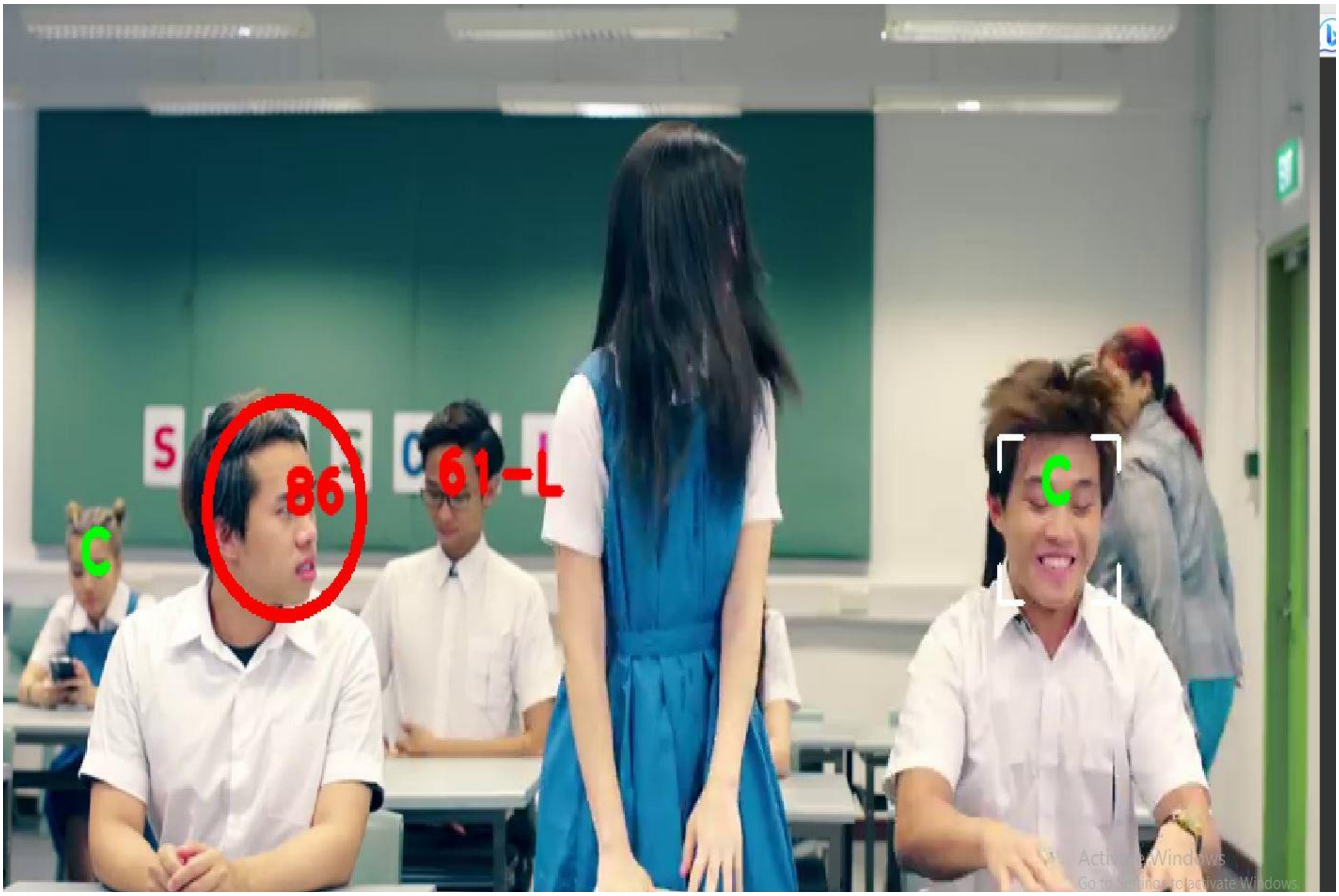


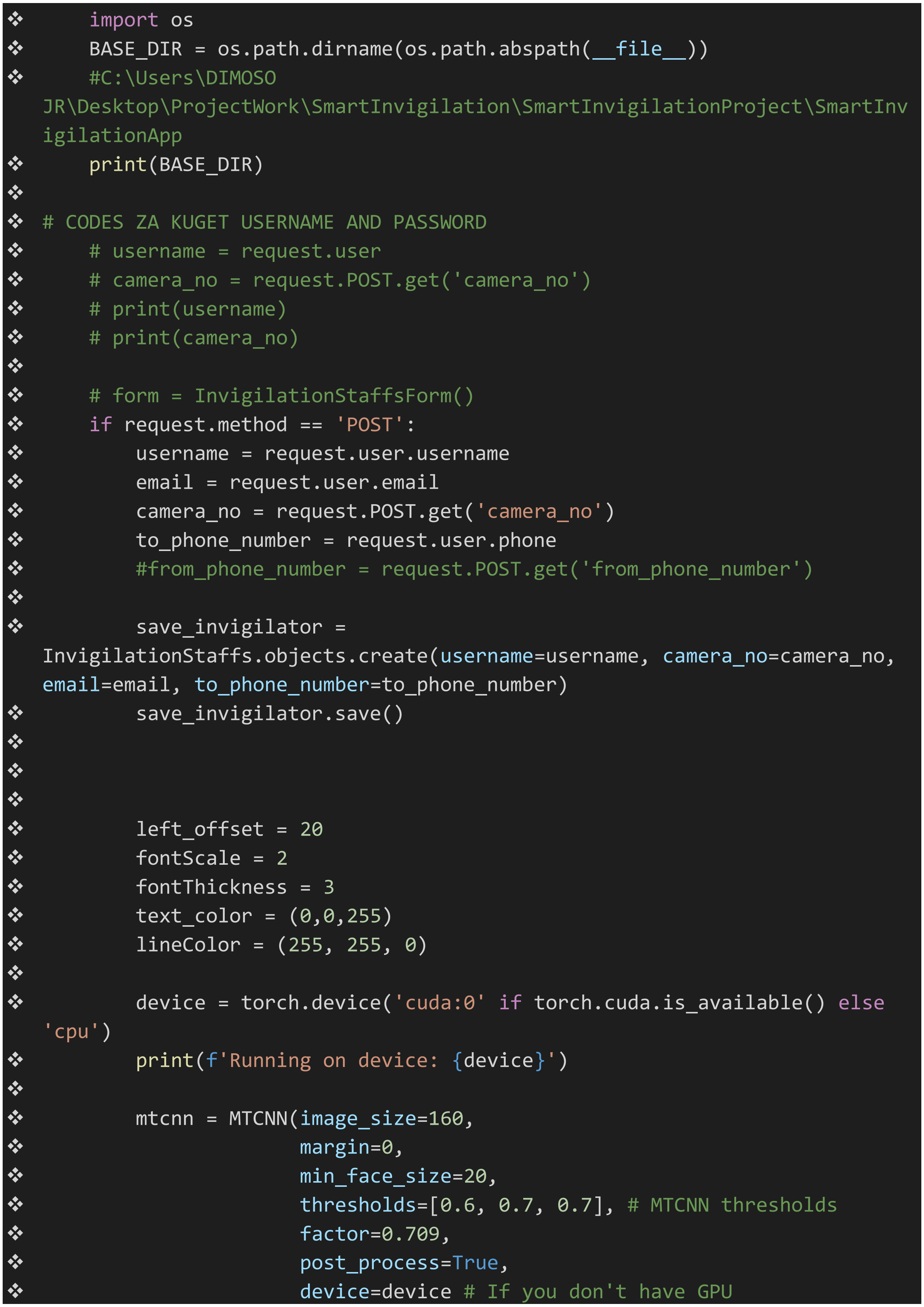
Figure 7: Final Prototype of Smart Invigilation System

7.1. System Codes

- Views.py File

```
from django.db.models.query import QuerySet
from django.http.response import HttpResponse, HttpResponseRedirect
from django.shortcuts import render, redirect, reverse, get_object_or_404
from django.contrib import messages
from django.contrib.auth import login, authenticate
from django.contrib.auth.forms import UserCreationForm
from django.contrib.auth.models import User, auth
from django.core.mail import send_mail
from django.conf import settings
from django.contrib.auth.decorators import login_required
from .models import *
from .forms import *

# project 2
from facenet_pytorch import MTCNN
from PIL import Image
from matplotlib import pyplot  as plt
import numpy as np
import math
import requests

import argparse
import torch
import cv2
#from .face import detect_faces
from . import NameFind


import random
# import predFacePoseApp
# Create your views here.

#SEND EMAIL AND SMS

from twilio.rest import Client
from django.core.mail import EmailMultiAlternatives
from django.template.loader import render_to_string
from django.utils.html import strip_tags
from django.core.mail import send_mail
from django.conf import settings

```

```python
def homePage(request):

    return render(request, 'SmartInvigilationApp/homePage.html')

def signin(request):

    if request.method == 'POST':
        email = request.POST.get('email')
        password = request.POST.get('password')

        user = auth.authenticate(email=email, password=password)

        if user is not None:
            auth.login(request, user)
            return redirect('homePage')
        else:
            messages.info(request, 'Credentials Invalid, Email or Password is incorrect')
            return redirect('signin')

    else:
        return render(request, 'SmartInvigilationApp/homePage.html')

def logout(request):
    auth.logout(request)
    return redirect('homePage')


def SmartInvigilationProject(request):
```



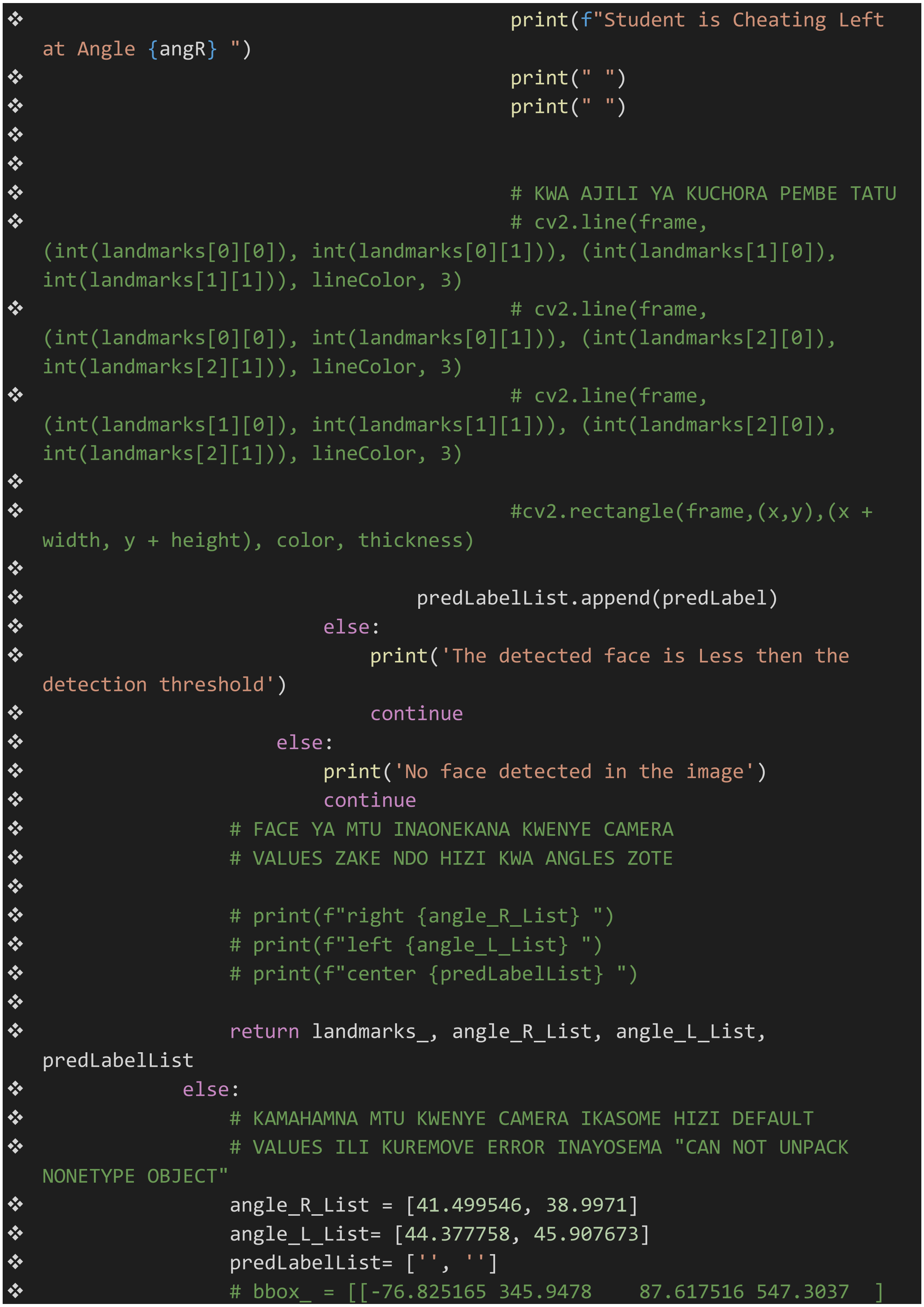
```
❖       import os
❖       BASE_DIR = os.path.dirname(os.path.abspath(__file__))
❖       #C:\Users\DIMOSO
   JR\Desktop\ProjectWork\SmartInvigilation\SmartInvigilationProject\SmartInv
   igilationApp
❖       print(BASE_DIR)
❖
❖   # CODES ZA KUGET USERNAME AND PASSWORD
❖       # username = request.user
❖       # camera_no = request.POST.get('camera_no')
❖       # print(username)
❖       # print(camera_no)
❖
❖       # form = InvigilationStaffsForm()
❖       if request.method == 'POST':
❖           username = request.user.username
❖           email = request.user.email
❖           camera_no = request.POST.get('camera_no')
❖           to_phone_number = request.user.phone
❖           #from_phone_number = request.POST.get('from_phone_number')
❖
❖           save_invigilator =
   InvigilationStaffs.objects.create(username=username, camera_no=camera_no,
   email=email, to_phone_number=to_phone_number)
❖           save_invigilator.save()
❖
❖
❖
❖           left_offset = 20
❖           fontScale = 2
❖           fontThickness = 3
❖           text_color = (0,0,255)
❖           lineColor = (255, 255, 0)
❖
❖           device = torch.device('cuda:0' if torch.cuda.is_available() else
   'cpu')
❖           print(f'Running on device: {device}')
❖
❖           mtcnn = MTCNN(image_size=160,
❖                         margin=0,
❖                         min_face_size=20,
❖                         thresholds=[0.6, 0.7, 0.7], # MTCNN thresholds
❖                         factor=0.709,
❖                         post_process=True,
❖                         device=device # If you don't have GPU
❖                 )
❖ 
❖         # Landmarks: [Left Eye], [Right eye], [nose], [left mouth], [right mouth]
❖         def npAngle(a, b, c):
❖             ba = a - b
❖             bc = c - b
❖ 
❖             cosine_angle = np.dot(ba, bc)/(np.linalg.norm(ba)*np.linalg.norm(bc))
❖             angle = np.arccos(cosine_angle)
❖ 
❖             return np.degrees(angle)
❖ 
❖         # def visualize(image, landmarks_, angle_R_, angle_L_, pred_):
❖         #     fig , ax = plt.subplots(1, 1, figsize= (8,8))
❖ 
❖         #     leftCount = len([i for i in pred_ if i == 'L'])
❖         #     rightCount = len([i for i in pred_ if i == 'R'])
❖         #     frontalCount = len([i for i in pred_ if i == ''])
❖         #     facesCount = len(pred_) # Number of detected faces (above the threshold)
❖         #     ax.set_title(f"Number of detected faces = {facesCount} \n frontal = {frontalCount}, left = {leftCount}, right = {rightCount}")
❖         #     for landmarks, angle_R, angle_L, pred in zip(landmarks_, angle_R_, angle_L_, pred_):
❖ 
❖         #         if pred == 'C':
❖         #             color = 'red'
❖         #         elif pred == 'R':
❖         #             color = 'blue'
❖         #         else:
❖         #             color = 'green'
❖ 
❖         #         point1 = [landmarks[0][0], landmarks[1][0]]
❖         #         point2 = [landmarks[0][1], landmarks[1][1]]
❖ 
❖         #         point3 = [landmarks[2][0], landmarks[0][0]]
❖         #         point4 = [landmarks[2][1], landmarks[0][1]]
❖ 
❖         #         point5 = [landmarks[2][0], landmarks[1][0]]
❖         #         point6 = [landmarks[2][1], landmarks[1][1]]
❖         #         for land in landmarks:
❖         #             pass
❖         #         #TO PRINT TRIANGLE AND CIRCLES ON A FACE
❖       #      #ax.scatter(land[0], land[1])
❖       #    # plt.plot(point1, point2, 'y', linewidth=3)
❖       #    # plt.plot(point3, point4, 'y', linewidth=3)
❖       #    # plt.plot(point5, point6, 'y', linewidth=3)
❖       #    #looking_center = int(pred)
❖       #    looking_right = int(math.floor(angle_R))
❖       #    looking_left = int(math.floor(angle_L))
❖
❖       #    plt.text(point1[0], point2[0], f"{pred} \n
   {looking_left}, {looking_right}",
❖       #        size=20, ha="center", va="center", color=color)
❖       #    ax.imshow(image)
❖       #    fig.savefig(BASE_DIR+'/OutputImages/Output_detection.jpg
   ')
❖       #  return print('Done detect')
❖
❖       def visualizeCV2(frame, landmarks_, angle_R_, angle_L_, pred_):
❖
❖           for landmarks, angle_R, angle_L, pred in zip(landmarks_,
   angle_R_, angle_L_, pred_):
❖
❖               if pred == 'C':
❖                   color = (0, 255, 0) #Green-BGR
❖               elif pred == 'Right Profile':
❖                   color = (255, 0, 0)
❖               else:
❖                   color = (0, 0, 255)
❖
❖               point1 = [int(landmarks[0][0]), int(landmarks[1][0])]
❖               point2 = [int(landmarks[0][1]), int(landmarks[1][1])]
❖
❖               point3 = [int(landmarks[2][0]), int(landmarks[0][0])]
❖               point4 = [int(landmarks[2][1]), int(landmarks[0][1])]
❖
❖               point5 = [int(landmarks[2][0]), int(landmarks[1][0])]
❖               point6 = [int(landmarks[2][1]), int(landmarks[1][1])]
❖
❖               for land in landmarks:
❖                   pass
❖                   #UKITAKA KUWEKA LINE KWENYE FACE UNCOMMENT BELOW THEN
   TOA PASS HAPO JUU
❖                   # cv2.circle(frame, (int(land[0]), int(land[1])),
   radius=5, color=(0, 255, 255), thickness=-1)
❖
❖
```

```
❖                # cv2.line(frame, (int(landmarks[0][0]), int(landmarks[0][1])), (int(landmarks[1][0]), int(landmarks[1][1])), lineColor, 3)
❖                # cv2.line(frame, (int(landmarks[0][0]), int(landmarks[0][1])), (int(landmarks[2][0]), int(landmarks[2][1])), lineColor, 3)
❖                # cv2.line(frame, (int(landmarks[1][0]), int(landmarks[1][1])), (int(landmarks[2][0]), int(landmarks[2][1])), lineColor, 3)
❖
❖                text_sizeR, _ = cv2.getTextSize(pred, cv2.FONT_HERSHEY_PLAIN, fontScale, 4)
❖                text_wR, text_hR = text_sizeR
❖
❖                cv2.putText(frame, pred,(point1[0], point2[0]), cv2.FONT_HERSHEY_PLAIN, fontScale, color, fontThickness, cv2.LINE_AA)
❖


❖        def predFacePose(frame):
❖
❖            bbox_, prob_, landmarks_ = mtcnn.detect(frame, landmarks=True) # The detection part producing bounding box, probability of the detected face, and the facial landmarks
❖            angle_R_List = []
❖            angle_L_List = []
❖            predLabelList = []
❖
❖            x,y,width,height = 100,100,200,150
❖            color = (0,255,0)
❖            thickness = 2
❖
❖
❖
❖
❖
❖
❖
❖            if bbox_ is not None and prob_ is not None and landmarks_ is not None:
❖
❖                for bbox, landmarks, prob in zip(bbox_, landmarks_, prob_):
❖                    if bbox is not None: # To check if we detect a face in the image
❖                        if prob > 0.9:#0.9 # To check if the detected face has probability more than 90%, to avoid
❖                            angR = npAngle(landmarks[0], landmarks[1], landmarks[2]) # Calculate the right eye angle
❖                            angL = npAngle(landmarks[1], landmarks[0], landmarks[2])# Calculate the left eye angle
❖                            angle_R_List.append(angR)
❖                            angle_L_List.append(angL)
❖
❖
❖
❖
❖
❖                            if ((int(angR) in range(35, 57)) and (int(angL) in range(35, 58))):
❖                                predLabel='' #C 'Frontal'
❖                                predLabelList.append(predLabel)
❖                            else:
❖                                if angR < angL:
❖                                    LeftAngle = int(angL)
❖                                    predLabel= f'{LeftAngle}-R'
❖
❖                                    print(f"angR (θ2) = {angR} and angL (θ1) =  {angL}, Hence, Student is Looking Right at Angle {angL} ")
❖                                    print(" ")
❖                                    print(" ")
❖
❖
❖                                    if angL > 80: #-80
❖
❖                                        #HIZI NI KWA AJILI KUTUMA EMAIL KWA INVIGILATOR
❖
❖
❖                                        # subject = "SMART INVIGILATION SYSTEM"
❖                                        # message = f"Hey {username} Some of the students are cheating Right, Please! Check your screen"
❖                                        # from_email = settings.EMAIL_HOST_USER
❖                                        # recipient_list = [email]
❖                                        # send_mail(subject, message, from_email, recipient_list, fail_silently=True)
❖
```

```
❖                                       #MWISHO WA KWA AJILI KUTUMA EMAIL
   KWA INVIGILATOR
❖
❖                                       #HIZI NI KWA AJILI YA KUTUMA SMS
   KWA INVIGILATOOR
❖
❖
❖
❖                                       # account_sid =
   "ACec294a515a22883f25914c532734bf23"
❖                                       # auth_token =
   "0d924ac3503fe120262b106a4c7a0bd3"
❖                                       # client = Client(account_sid,
   auth_token)
❖                                       # message = client.messages \
❖                                       #              .create(
❖                                       #                  body="Some of
   the students are cheating Right, Please! Check your screen",
❖                                       #                  from_='+14068023
   763',
❖                                       #                  #to='+2556284315
   07'
❖                                       #                  to
   =to_phone_number
❖                                       #                          )
❖                                       # print("Some of the students are
   cheating Right, Please! Check your screen")
❖
❖
❖
❖                                       #MWISHO WA KUTUMA MSG KWA
   INVIGILATOR
❖
❖
❖                                       cv2.circle(frame,
   (int(landmarks[1][0]), int(landmarks[1][1])), radius=50, color=(0, 0,
   255), thickness=5)
❖                                       predLabel= f'{LeftAngle}'
   #'Cheating L'
❖
❖                                       print("LEFT ANGLE IS > 80:  THEN;
   ")
❖
❖
❖
```

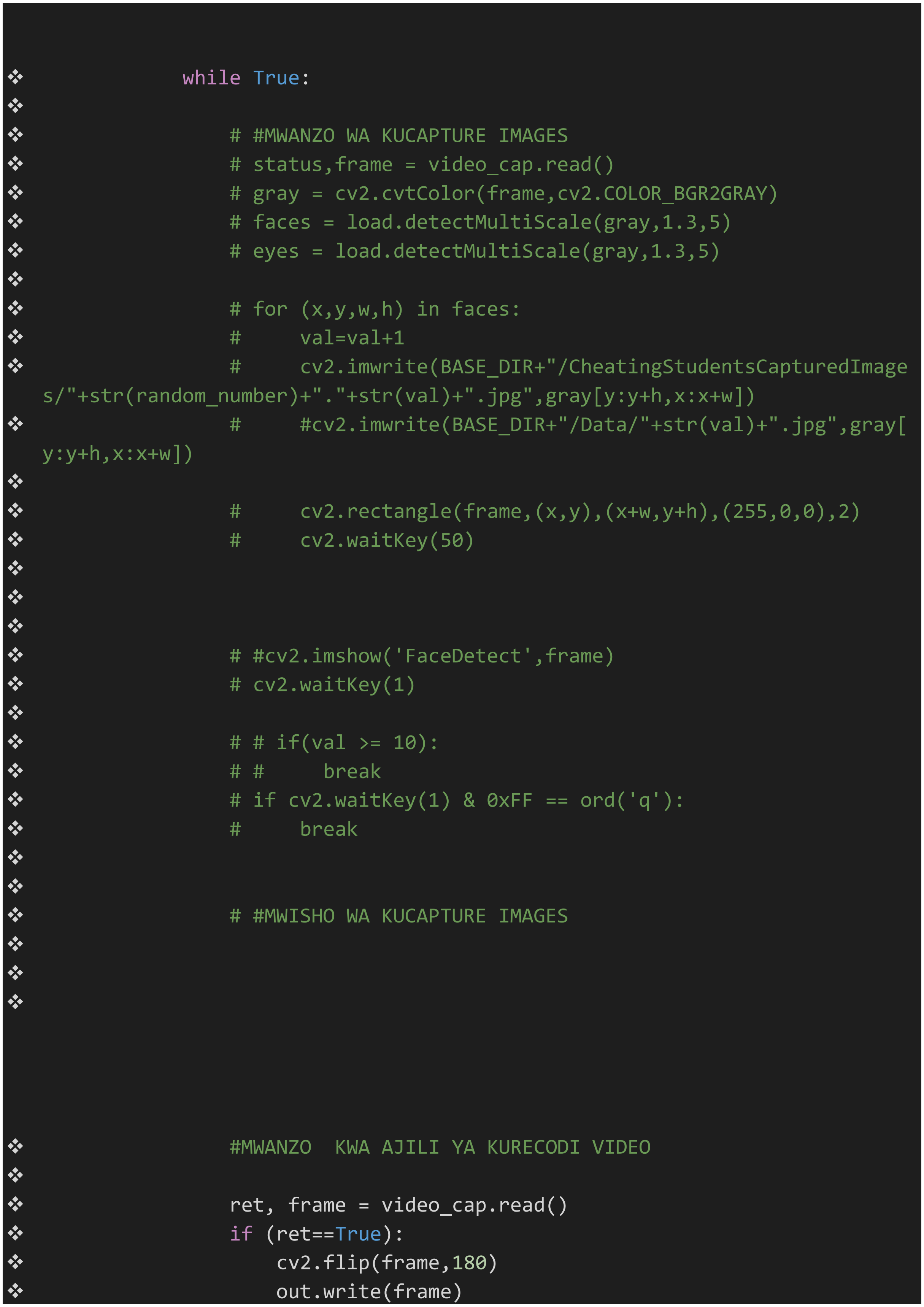
```
❖                     print(f"Student is Cheating Right at Angle {angL} ")
❖                     print(" ")
❖                     print(" ")
❖
❖
❖                     # KWA AJILI YA KUCHORA PEMBE TATU
❖                     # cv2.line(frame, (int(landmarks[0][0]), int(landmarks[0][1])), (int(landmarks[1][0]), int(landmarks[1][1])), lineColor, 3)
❖                     # cv2.line(frame, (int(landmarks[0][0]), int(landmarks[0][1])), (int(landmarks[2][0]), int(landmarks[2][1])), lineColor, 3)
❖                     # cv2.line(frame, (int(landmarks[1][0]), int(landmarks[1][1])), (int(landmarks[2][0]), int(landmarks[2][1])), lineColor, 3)
❖

❖                 else:
❖                   RightAngle = int(angR)
❖                   predLabel=f'{RightAngle}-L'
❖

❖                   print(f"angR (θ2) = {angR} and angL (θ1) =  {angL}, Hence, Student is Looking Left at Angle {angR} ")
❖                   print(" ")
❖
❖
❖                   if angR > 80: #80
❖
❖                     #HIZI NI KWA AJILI KUTUMA EMAIL KWA INVIGILATOR
❖
❖
❖                     # subject = "SMART INVIGILATION SYSTEM"
❖                     # message = f"Hey {username} Some of the students are cheating Left, Please! Check your screen"
❖                     # from_email = settings.EMAIL_HOST_USER
❖                     # recipient_list = [email]
❖                     # send_mail(subject, message, from_email, recipient_list, fail_silently=True)
❖
```

```
❖                                        #MWISHO KWA AJILI KUTUMA EMAIL KWA INVIGILATOR
❖
❖                                        #HIZI NI KWA AJILI YA KUTUMA SMS KWA INVIGILATOOR
❖
❖
❖
❖                                        # account_sid = "ACec294a515a22883f25914c532734bf23"
❖                                        # auth_token = "0d924ac3503fe120262b106a4c7a0bd3"
❖                                        # client = Client(account_sid, auth_token)
❖                                        # message = client.messages \
❖                                        #              .create(
❖                                        #                  body="Some of the students are cheating Left, Please! Check your screen",
❖                                        #                  from_='+14068023763',
❖                                        #                  #to='+255628431507'
❖                                        #                  to =to_phone_number
❖                                        #                              )
❖                                        # print("Some of the students are cheating Left, Please! Check your screen")
❖
❖                                        #MWISHO KWA AJILI YA KUTUMA SMS KWA INVIGILATOOR
❖
❖
❖
❖
❖                                        cv2.circle(frame, (int(landmarks[0][0]), int(landmarks[1][1])), radius=50, color=(0, 0, 255), thickness=5)
❖                                        predLabel=f'{RightAngle}' #'Cheating R'
❖
❖                                        print("RIGHT ANGLE IS > 80: THEN; ")
❖
❖
❖
```



```
❖                                         print(f"Student is Cheating Left at Angle {angR} ")
❖                                         print(" ")
❖                                         print(" ")
❖
❖
❖                                         # KWA AJILI YA KUCHORA PEMBE TATU
❖                                         # cv2.line(frame, (int(landmarks[0][0]), int(landmarks[0][1])), (int(landmarks[1][0]), int(landmarks[1][1])), lineColor, 3)
❖                                         # cv2.line(frame, (int(landmarks[0][0]), int(landmarks[0][1])), (int(landmarks[2][0]), int(landmarks[2][1])), lineColor, 3)
❖                                         # cv2.line(frame, (int(landmarks[1][0]), int(landmarks[1][1])), (int(landmarks[2][0]), int(landmarks[2][1])), lineColor, 3)
❖
❖                                         #cv2.rectangle(frame,(x,y),(x + width, y + height), color, thickness)
❖
❖                                 predLabelList.append(predLabel)
❖                         else:
❖                             print('The detected face is Less then the detection threshold')
❖                             continue
❖                     else:
❖                         print('No face detected in the image')
❖                         continue
❖                 # FACE YA MTU INAONEKANA KWENYE CAMERA
❖                 # VALUES ZAKE NDO HIZI KWA ANGLES ZOTE
❖
❖                 # print(f"right {angle_R_List} ")
❖                 # print(f"left {angle_L_List} ")
❖                 # print(f"center {predLabelList} ")
❖
❖                 return landmarks_, angle_R_List, angle_L_List, predLabelList
❖             else:
❖                 # KAMAHAMNA MTU KWENYE CAMERA IKASOME HIZI DEFAULT
❖                 # VALUES ILI KUREMOVE ERROR INAYOSEMA "CAN NOT UNPACK NONETYPE OBJECT"
❖                 angle_R_List = [41.499546, 38.9971]
❖                 angle_L_List= [44.377758, 45.907673]
❖                 predLabelList= ['', '']
❖                 # bbox_ = [[-76.825165 345.9478    87.617516 547.3037   ]
❖                   #           [302.04672  163.34067  380.0983   259.25977 ]
❖                   #           [120.40265  130.62308  186.46005  210.27997 ]]
❖                   landmarks_ = [[[-34.039093,   421.1461    ],
❖                                  [ 16.470798,   427.12183   ],
❖                                  [-35.07744,    464.84344   ],
❖                                  [-41.21428,    498.56042   ],
❖                                  [ -0.88659286, 504.5458    ]],
❖
❖                                 [[324.0439,     198.46864   ],
❖                                  [360.09473,    200.7623    ],
❖                                  [339.04395,    216.24658   ],
❖                                  [324.7361,     238.5289    ],
❖                                  [353.1474,     240.4968    ]],
❖
❖                                 [[133.74072,    161.84549   ],
❖                                  [163.7682,     158.71199   ],
❖                                  [145.17691,    176.29892   ],
❖                                  [138.92558,    193.42422   ],
❖                                  [164.13974,    191.53346   ]]]
❖                   # prob_ =[0.9998667, 0.9999882, 0.9963965]
❖                   #return landmarks_, angle_R_List, angle_L_List,
   predLabelList
❖                   print("There is no face to detect")
❖                   # print("None")
❖                   # print(f"r is {angle_R_List} ")
❖                   # print(f"l is {angle_L_List} ")
❖                   # print(f"prob_ is {predLabelList} ")
❖
❖                   return landmarks_, angle_R_List, angle_L_List,
   predLabelList
❖
❖
❖           def predFacePoseApp():
❖
❖
❖
❖               source = 0
❖
❖               mysource = camera_no #now naingiza video path lkn km unaingiza
   no?
❖               #mysource = int(camera_no)
❖               #print(camera_no)
❖               # Create a video capture object from the VideoCapture Class.
❖               #video_cap = cv2.VideoCapture(BASE_DIR+"/videos/4.mp4")
❖               video_cap = cv2.VideoCapture(BASE_DIR+mysource)
```

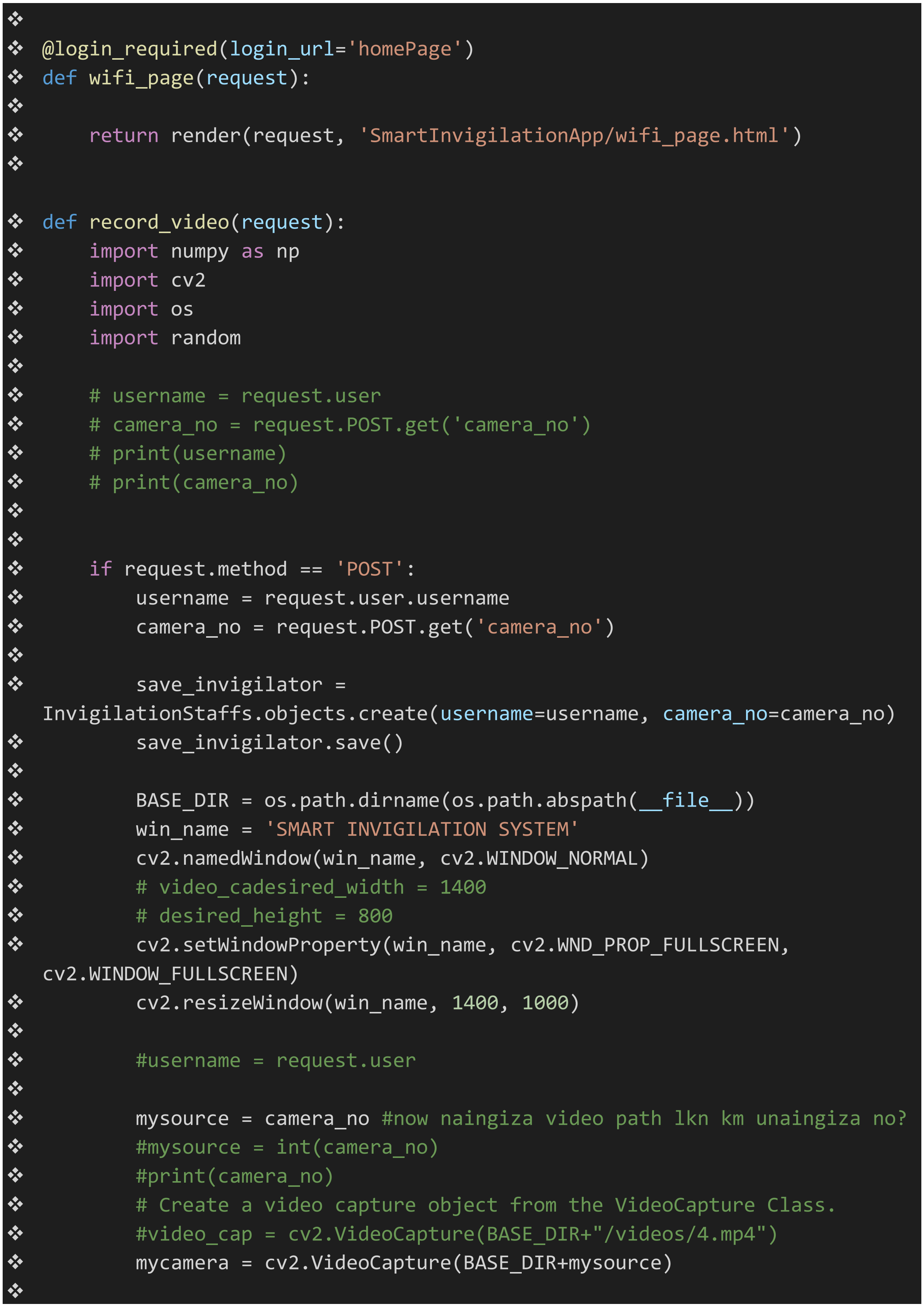
```
❖
❖            #km unaingiza no path itakuwa
❖            #video_cap = cv2.VideoCapture(mysource)
❖
❖            # Create a named window for the video display.
❖            win_name = 'SMART INVIGILATION SYSTEM'
❖            cv2.namedWindow(win_name, cv2.WINDOW_NORMAL)
❖            # video_cadesired_width = 1400
❖            # desired_height = 800
❖            cv2.setWindowProperty(win_name, cv2.WND_PROP_FULLSCREEN,
  cv2.WINDOW_FULLSCREEN)
❖            cv2.resizeWindow(win_name, 1400, 1000)
❖            #dim = (video_cadesired_width, desired_height)
❖            left_offset = 20
❖            fontScale = 2
❖            fontThickness = 3
❖            text_color = (255,0,0) #(0,0,255)
❖
❖            # #Mwisho wa full screen Model
❖
❖
❖            #MWANZO  KWA AJILI YA KURECODI VIDEO
❖
❖            random.seed()
❖            random_number = random.randint(1,100)
❖            print(random_number)
❖
❖            #video_cap = cv2.VideoCapture(BASE_DIR+"/videos/6.mp4")
❖            fourcc=cv2.VideoWriter_fourcc('M','J','P','G')
❖            capturing_win_name = 'CAPTURING LIVE VIDEO'
❖            cv2.namedWindow(capturing_win_name, cv2.WINDOW_NORMAL)
❖            # video_cadesired_width = 1400
❖            # desired_height = 800
❖            cv2.setWindowProperty(capturing_win_name,
  cv2.WND_PROP_FULLSCREEN, cv2.WINDOW_FULLSCREEN)
❖            cv2.resizeWindow(capturing_win_name, 1400, 1000)
❖            # out=cv2.VideoWriter(BASE_DIR+'/saved-
  media/my.mp4',fourcc,20.0,(640,480))
❖
❖            if (video_cap.isOpened() == False):
❖                print("Unable to read camera")
❖
❖            frame_width = int(video_cap.get(3))
❖            frame_height = int(video_cap.get(4))
❖             out=cv2.VideoWriter(BASE_DIR+f'/saved-media/SmartInvigilationVideo-{random_number}.avi',fourcc,10,(frame_width,frame_height))
❖
❖             #MWISHO  KWA AJILI YA KURECODI VIDEO
❖



❖
❖
❖             face_cascade = cv2.CascadeClassifier(BASE_DIR+'/cascades/data/haarcascade_frontalface_default.xml')
❖             eye_cascade = cv2.CascadeClassifier(BASE_DIR+'/cascades/data/haarcascade_eye.xml')
❖


❖             #MWANZO WA KUCAPTURE IMAGES
❖
❖
❖             load = cv2.CascadeClassifier(BASE_DIR+'/cascades/data/haarcascade_frontalface_default.xml')
❖
❖             #cap=cv2.VideoCapture("videos/4.mp4")
❖
❖             random.seed()
❖             random_number = random.randint(1,100)
❖             print(random_number)
❖
❖             # name = input("Enter Username : ")
❖             # id = input("Enter Unique id (maxLen = 4) : ")
❖
❖             val=0
❖

❖             #MWISHO WA KUCAPTURE IMAGES
❖
❖
```



```
while True:

    # #MWANZO WA KUCAPTURE IMAGES
    # status,frame = video_cap.read()
    # gray = cv2.cvtColor(frame,cv2.COLOR_BGR2GRAY)
    # faces = load.detectMultiScale(gray,1.3,5)
    # eyes = load.detectMultiScale(gray,1.3,5)

    # for (x,y,w,h) in faces:
    #     val=val+1
    #     cv2.imwrite(BASE_DIR+"/CheatingStudentsCapturedImages/"+str(random_number)+"."+str(val)+".jpg",gray[y:y+h,x:x+w])
    #     #cv2.imwrite(BASE_DIR+"/Data/"+str(val)+".jpg",gray[y:y+h,x:x+w])

    #     cv2.rectangle(frame,(x,y),(x+w,y+h),(255,0,0),2)
    #     cv2.waitKey(50)



    # #cv2.imshow('FaceDetect',frame)
    # cv2.waitKey(1)

    # # if(val >= 10):
    # #     break
    # if cv2.waitKey(1) & 0xFF == ord('q'):
    #     break


    # #MWISHO WA KUCAPTURE IMAGES




    #MWANZO  KWA AJILI YA KURECODI VIDEO

    ret, frame = video_cap.read()
    if (ret==True):
        cv2.flip(frame,180)
        out.write(frame)
```

```
❖
❖                 # Display the resulting frame
❖                     cv2.imshow(capturing_win_name, frame)
❖                     if cv2.waitKey(1) & 0xFF == ord('q'):
❖                         break
❖                 else:
❖                     continue
❖
❖                 #MWISHO  KWA AJILI YA KURECODI VIDEO
❖
❖
❖
❖                 # Read one frame at a time using the video capture object.
❖                 has_frame, frame = video_cap.read()
❖                 if not has_frame:
❖                     #break
❖                     print("It is not possible to read image from the camera")
❖                     continue
❖

❖
❖

❖                 #DRAWING BOX ON STUDENT'S FACE
❖
❖                 faces = face_cascade.detectMultiScale(frame, 1.3, 5)
❖                 for (x, y, w, h) in faces:
❖                     NameFind.draw_box(frame, x, y, w, h)
❖
❖                 #cv2.imshow('Face Detection Using Haar-Cascades ', frame)
❖                 if cv2.waitKey(1) & 0xFF == ord('q'):
❖                     break
❖
❖                 #MWISHO WA  DRAWING BOX
❖

❖                 landmarks_, angle_R_List, angle_L_List, predLabelList = predFacePose(frame)
❖
❖                 # print(angle_R_List)
❖                # print(angle_L_List)
❖                # angle_L_List and angle_R_List are list inwhich
❖                # the left and right angles are appended
❖
❖                # Annotate each video frame.
❖                visualizeCV2(frame, landmarks_, angle_R_List, angle_L_List, predLabelList)
❖
❖                #To draw a graphy
❖                #visualize(frame, landmarks_, angle_R_List, angle_L_List, predLabelList)
❖
❖                for Left_angle in angle_R_List:
❖                    if Left_angle > 80:
❖                        #MWANZO WA KUCAPTURE IMAGES
❖
❖                        gray2 = cv2.cvtColor(frame,cv2.COLOR_BGR2GRAY)
❖                        faces2 = load.detectMultiScale(gray2,1.3,5)
❖
❖
❖                        for (x,y,w,h) in faces2:
❖                            val=val+1
❖                            cv2.imwrite(BASE_DIR+"/LeftCapturedImages/"+str(Left_angle)+"."+str(val)+".jpg",gray2[y:y+h,x:x+w])
❖
❖
❖                        # if(val >= 3):
❖                        #     break
❖                        # if cv2.waitKey(1) & 0xFF == ord('q'):
❖                        #     break
❖                        #MWISHO WA KUCAPTURE IMAGES
❖
❖                for Right_angle in angle_L_List:
❖                    if Right_angle > 80:
❖                        #MWANZO WA KUCAPTURE IMAGES
❖
❖                        gray2 = cv2.cvtColor(frame,cv2.COLOR_BGR2GRAY)
❖                        faces2 = load.detectMultiScale(gray2,1.3,5)
❖
❖
❖                        for (x,y,w,h) in faces2:
❖                            val=val+1
❖                            cv2.imwrite(BASE_DIR+"/RightCapturedImages/"+str(Left_angle)+"."+str(val)+".jpg",gray2[y:y+h,x:x+w])
❖
```

```
❖
❖                        # if(val >= 3):
❖                        #     break
❖                        # if cv2.waitKey(1) & 0xFF == ord('q'):
❖                        #     break
❖                        #MWISHO WA KUCAPTURE IMAGES
❖
❖
❖
❖
❖
❖
❖                cv2.imshow(win_name, frame)
❖
❖                key = cv2.waitKey(1)
❖
❖                # You can use this feature to check if the user selected the `q` key to quit the video stream.
❖                if key == ord('Q') or key == ord('q') or key == 27:
❖                    # Exit the loop.
❖                    break
❖
❖            video_cap.release()
❖            cv2.destroyWindow(win_name)
❖
❖
❖
❖        messages.success(request, f"Invigilation Completed Successfully By - {username} ")
❖        predFacePoseApp()
❖


❖        return redirect('starting_page')
❖
❖
❖                #return HttpResponse("welll")
❖                #return render(request, 'SmartInvigilationApp/homePage.html')
❖
❖
❖ @login_required(login_url='homePage')
❖ def starting_page(request):
❖
❖     return render(request, 'SmartInvigilationApp/starting_page.html')
```



```
❖
❖ @login_required(login_url='homePage')
❖ def wifi_page(request):
❖
❖     return render(request, 'SmartInvigilationApp/wifi_page.html')
❖

❖ def record_video(request):
❖     import numpy as np
❖     import cv2
❖     import os
❖     import random
❖
❖     # username = request.user
❖     # camera_no = request.POST.get('camera_no')
❖     # print(username)
❖     # print(camera_no)
❖
❖
❖     if request.method == 'POST':
❖         username = request.user.username
❖         camera_no = request.POST.get('camera_no')
❖
❖         save_invigilator =
  InvigilationStaffs.objects.create(username=username, camera_no=camera_no)
❖         save_invigilator.save()
❖
❖         BASE_DIR = os.path.dirname(os.path.abspath(__file__))
❖         win_name = 'SMART INVIGILATION SYSTEM'
❖         cv2.namedWindow(win_name, cv2.WINDOW_NORMAL)
❖         # video_cadesired_width = 1400
❖         # desired_height = 800
❖         cv2.setWindowProperty(win_name, cv2.WND_PROP_FULLSCREEN,
  cv2.WINDOW_FULLSCREEN)
❖         cv2.resizeWindow(win_name, 1400, 1000)
❖
❖         #username = request.user
❖
❖         mysource = camera_no #now naingiza video path lkn km unaingiza no?
❖         #mysource = int(camera_no)
❖         #print(camera_no)
❖         # Create a video capture object from the VideoCapture Class.
❖         #video_cap = cv2.VideoCapture(BASE_DIR+"/videos/4.mp4")
❖         mycamera = cv2.VideoCapture(BASE_DIR+mysource)
❖
```

```
❖
❖
❖
❖           #km unaingiza no path itakuwa
❖           #mycamera = cv2.VideoCapture(mysource)
❖
❖           random.seed()
❖           random_number = random.randint(1,100)
❖           print(random_number)
❖

❖           fourcc=cv2.VideoWriter_fourcc('M','J','P','G')
❖           # out=cv2.VideoWriter(BASE_DIR+'/saved-media/my.mp4',fourcc,20.0,(640,480))
❖
❖           if (mycamera.isOpened() == False):
❖               print("Unable to read camera")
❖
❖           frame_width = int(mycamera.get(3))
❖           frame_height = int(mycamera.get(4))
❖           out=cv2.VideoWriter(BASE_DIR+f'/saved-media/SmartInvigilationVideo-{random_number}.avi',fourcc,10,(frame_width,frame_height))
❖
❖           while(True):
❖               # mycamerature frame-by-frame
❖               ret, frame = mycamera.read()
❖               if (ret==True):
❖                   cv2.flip(frame,180)
❖                   out.write(frame)
❖
❖               # Display the resulting frame
❖                   cv2.imshow(win_name, frame)
❖                   if cv2.waitKey(1) & 0xFF == ord('q'):
❖                       break
❖               else:
❖                   break
❖
❖           # When everything done, release the mycamerature
❖           mycamera.release()
❖           cv2.destroyAllWindows()
❖
❖           messages.success(request, f"Video Recorded Successfully By {username} ")
❖           return redirect('recording_video_page')
```

```
❖
❖       messages.success(request, f"Error! Form is not valid")
❖       return redirect('recording_video_page')
❖
❖       #return render(request,
    'SmartInvigilationApp/recording_video_page.html')
❖

❖   @login_required(login_url='homePage')
❖   def recording_video_page(request):
❖
❖       return render(request,
    'SmartInvigilationApp/recording_video_page.html')
❖



❖
❖
```

CHAPTER 6

8. Conclusion and Recommendation

8.1. Conclusion

This project proposed a system for students' invigilation in the examination using the Deep Learning and computer vision approach. In this work, implemented the MTCNN with Opencv for automatic invigilation. The detection of the cheating was done on the base of the neck and face movement of the students. The data set was generated in a local environment for the experiments. The results show that proposed model achieved more accuracy as compared to other one. In future work, the system will also able to detect the other methods of cheating like exchanging sheet, head movement of the students, lip movement of the student and gesture detection.

8.2. Recommendation

A smart invigilation system for offline examinations based on face orientation can greatly enhance the integrity and security of the examination process.

By implementing advanced facial recognition to monitor and detect the face orientation of each student with her or his name on the face during the exam will improve the invigilation process.

This feature ensures that students maintain proper eye contact with their own answer sheets and discourages any attempts to seek external assistance or engage in malpractice. Moreover, the system can provide real-time alerts to invigilators if any student's face orientation deviates from the predefined guidelines, allowing them to take immediate action.

This recommendation provides a robust solution for maintaining the fairness and transparency of offline examinations, giving educational institutions the confidence to conduct assessments with increased efficiency and integrity.

9. Appendices A. Time Frame

ACTIVITIES	Duration in Months 2022/2023								
Selection of project title	NOV	DEC	JAN	FEB	MAR	APR	MAY	JUN	JULY
Title defending	🟦								
Consultation with supervisor	🟧	🟧							
Literature review	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩
Data collection	🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪
Data analysis and problem justification				🟨	🟨	🟨			
Designing circuit for a prototype						▨	▨	▨	
Building and testing a prototype							⬛	⬛	⬛
Project report writing and submission								🟥	🟥

10. References

1. Mahmood, F., Arshad, J., Ben Othman, M. T., Hayat, M. F., Bhatti, N., Jaffery, M. H., Rehman, A. U., & Hamam, H. (2022). Implementation of an Intelligent Exam Supervision System Using Deep Learning Algorithms. *Sensors*, *22*(17), 6389.
2. Khan, A. R., Saba, T., Khan, M. Z., Fati, S. M., & Khan, M. U. G. (2022). Classification of human's activities from gesture recognition in live videos using deep learning. *Concurrency and Computation: Practice and Experience*, *34*(10), e6825.
3. Thampan, N., & Arumugam, S. (2022). Smart Online Exam Invigilation using AI based Facial Detection and Recognition Algorithms. 2022 2nd Asian Conference on Innovation in Technology (ASIANCON),
4. Adil, Md, Rajbala Simon, and Sunil Kumar Khatri. "Automated invigilation system for detection of suspicious activities during examination." In 2019 Amity International Conference on Artificial Intelligence (AICAI), pp. 361-366. IEEE, 2019.
5. Kulkarni, Rutuja. "Real Time Automated Invigilator in Classroom Monitoring Using Computer Vision." In 2nd International Conference on Advances in Science & Technology (ICAST). 2019.
6. Li, Zhizhuang, Zhengzhou Zhu, and Teng Yang. "A Multi-index Examination Cheating Detection Method Based on Neural Network." In 2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI), pp. 575-581. IEEE, 2019