جامعة عجمان

AJMAN UNIVERSITY

College	College of Engineering and Information Technology
Title	Camera Cheating Behavior Detection System
Department	Department of Information Technology
Course	IT Project (Final Year Project)
Major	Networking and Security
Supervisor	Usman Javed Butt
Semester/Session	2024-2025

Names: -	**ID: -**
Anas Ahmed Hamood	202110589
Abdulrahman Mohamed Hammad	202110647
Omar Issam	202111117
Ali Ahmed	202111259

Acknowledgment

We extend our sincerest gratitude and appreciation to our supervisor, Usman Javed Butt, whose guidance, continuous motivation, and critical feedback played an instrumental role in the success of our project. His input was essential in refining our methodology and ensuring the practical viability of our work. We also extend special thanks to each member of our project team for their tireless dedication during the data collection process. Their commitment was demonstrated by contributing numerous personal photographs, tediously recreating a multitude of cheating behaviors to successfully train our AI detection model. Not only did this collaboration enhance our model's performance, it also provided invaluable practical experience in implementation. The team's diligent effort in gathering and labeling data was crucial for overcoming early technical challenges and facilitated the initial development of our project. We also wish to thank our families and friends for their moral encouragement and patience throughout this challenging project phase.

Abstract

This malpractice plague in exams continues to rage on in educational institutions despite a continued high level of integrity challenge. It has been demonstrated that traditional systems of invigilating examinations are incapable, because they are flawed, firstly, by inherent flaws in the system design, second, with inherent biases of the human being, and third, in resources available at a certain time. This current report provides our new solution, the Camera Cheating Behaviour Detection System which is based on AI for identification of behavioural patters that are suspiciously observed and thus related to cheating. Our solution approach focuses on motion patterns and body orientations unlike the use of face recognition techniques, which makes privacy paramount on students while taking their examinations. The machine learning based AI system was developed and trained on behind the complex compiled dataset that represents cheating behaviours emulated by our project members while working on the project. Additionally, a web based exam management system has been developed to provide live color-coded alerts when suspicious activities are detected. Future works include expansion of the dataset to improve detection abilities further, more advanced functionalities for the system itself and compatibility with currently deployed classroom CCTV systems. Our greatest war is on a surveillance system that is efficient, effective and above all impartial to ensure academic integrity within learning boards.

List of Abbreviations

AI: Artificial Intelligence
CCTV: Closed-Circuit Television
YOLO: You Only Look Once (an object detection model family)
GPU: Graphics Processing Unit

Table of Contents

Table of Figures

Table of Tables

Chapter 1: Introduction

1.1 Background

Academic institutions are ever faced with the consistent challenge of sustaining examination process and practices.
It has been proved that traditional surveillance has been ineffective, and its detection capacities have been limited, hence allowing some mischiefs of academic dishonesty remain undetected.

1.2 Motivation and Problem Statement

The foregoing project is driven by the need to enhance the efficiency and precision of academic cheatable detection systems. Human surveillance is constrained by subjective bias, fatigue, few-eyedness, and problems of monitoring large testing environments. It calls for an immediate solution of creating a technological solution that can plug on those loopholes and install accurate and unbiased sensors for cheating detection in real time. The aim here therefore would be to imagine and integrate such a system which would reduce human inconveniences in surveillance for an effective mechanism on speedy detection of cheating malpractices for timely remedial actions.

1.3 Objectives

The main objective with this project is to come up with AI vision system for capturing academic fraud, hence helping the collar suspicious acts in real-time when an examination is in process. Timely notification to an examiner should occur in case an alleged cheating is found, and it will pave the way for timely intervention. All for the secure logging of reported cases of cheating to analyse incidents and record-keeping after the investigation is over. Do the analysis of behaviours without the use of face recognition and protect the privacy of the students.

1.4 Project Contributions

The innovation in this research effort is the designing of an innovative AI solution to monitor exams while maintaining some of the privacy parameters intact. When given priority over identification parameters, the behavioral traces of body movement and pose enable the solution in this system to offer additional efficiency in detection, while adhering to the privacy of the candidate. In addition, this full solution also favors automated, objective monitoring and therefore, there is a reduction in dependency on human proctors. Intensive use of AI technology and its implementation on exams are brought together through the development of a series of intuitive web interfaces for examiners. Aggregated, these contributions build a foundation to piece together improvements in surveillance effectiveness and fairness for exams.

1.5 Documentation Overview

This report outlines the phases through which implementation of the project went though. The first question that the report addresses is planning and requirements analysis-specification of scope and objectives of the system. Following this is the literature review of other works. After design and analysis of the system, charting and design choices. The following chapters are dedicated to the subsequent description of implementation progress including AI model training and web interface development. Finally, the report announces results, as well as a discussion of the challenges encountered and improvements in future. Complementary technical details among other supporting data are available in the appendices.

Chapter 2: Literature Review

(This literature review examines recent developments in automated cheating detection and related AI technologies, highlighting the gap addressed by our project.)

2.1 Existing AI Surveillance Systems

For such papers, systems that view scholarly assessment sometimes employ detection frameworks for objects that are already old, say YOLOv3, for example. Such detection frameworks do not provide for real-time execution when considered for resource-constrained devices. On the other hand, a purely cloud-based proctoring approach leads to long delays and more privacy considerations concerning the examinee.

2.2 Pose Estimation Models

Complex pose estimation techniques, including OpenPose, are used to detect undesirable movements by students. These are top-notch for recognizing postures but require a whole lot of computational resources and would not operate in real-time on normal hardware without significant optimization.

2.3 Advancements in YOLOv11

YOLOv11-nano, the most recent version of YOLO models, is especially made for edge devices and performs even better than its predecessors, like the YOLOv8n, both in speed and accuracy. By incorporating Multi-Task Learning Optimization and Grouped Efficient Layer Aggregation Network into YOLOv10 and YOLOv11, inference speed and accuracy are further increased, thus greatly facilitating real-time cheating detection application during exams.

2.4 Hybrid Models

Others combine both artificial intelligence methods, such as audio and image processing in order to enhance detection efficacy. However, the amalgamation of these models more often than not presents additional structural complexity. While highly promising this additional complexity may prevent real time deployment with standard examination room equipment.

2.5 Conclusion

In general, the state of the art has neither provided a successful efficient real-time cheating detection system that could operate on local devices, a laptop of a teacher, for instance. Previous iterations of YOLO models and other visual-based approaches either had poor results or needed cloud-based systems, which created a gap that newer advances in artificial intelligence can fill.
This project is undertaken in an effort to bridge this gap using the best approaches that are in use today. The creation of a practical, on the site monitoring system for exams by vision models on AI.

Chapter 3: Planning and Requirements

The Camera Cheating Behavior Detection System is intended for exam supervisors and academic staff that requires a reliable tool to help support the examination surveillance. Other than that, IT support staff are considered as users who can then monitor and troubleshoot the technical issue for the deployment of the system. Practically, the system identifies the cheating behaviors on the go, sends on-screen alerts to exam supervisors in real time, and generates copies of incidents on supervisors' laptops as evidence. Major non-functional objectives are high detection accuracy for suspicious activities, an intuitive and friendly-to-use interface for the supervisors, fast response with minimal lag; reliable operation during exam sessions – no sensitive identification data is stored; and rigid privacy protection for students.

Several project assumptions and constraints were established from the beginning. The first deployment is executed as a laptop setup for testing as it operates outside of cloud services to eliminate internet reliability concerns and privacy issues. It is assumed that high resolution CCTV, or 'webcam' cameras will be utilized here which are connected via Bluetooth or wired USBs to feedback high quality video for effective detection. The system is developed with the knowledge that it will later be incorporated with standard classroom CCTV infrastructure.

Users	Functional Requirements	Users	Functional Requirements
Exam	Detect cheating behaviours	High detection	Detection initially runs on
Supervisors	in real time	accuracy	laptop (for testing)
Academic Staff	Notify supervisors immediately of incidents	User-friendly interface	No cloud dependency (local processing only)
IT Support	Log and store cheating incident records	Strict student privacy (no facial ID)	Use high-resolution CCTV camera (Bluetooth or cable)

Table 3.1 *Users, Functional and Non-functional Requirements, and Project Constraints*

Chapter 4: Project Analysis and Design

The Camera Cheating Behaviours Detection System enables exam supervisors to log into a specific website for live monitoring as well as immediate cheating behaviour alerts. A CCTV camera in the exam room is linked to the supervisor's laptop (with Bluetooth or via cable) that broadcasts video to an artificial intelligence module that tracks the movements of the students. The system is a system which will help to streamline exam proctoring where one supervisor can monitor all the exam hall. Major features include the real time detection of suspicious behaviour, instant alert notifications, ability to playback previously recorded evidence (after the exam).

When an exam session begins, the supervisor switches on the system; he or she also connects the camera of the classroom. The AI element continually looks at the live footage video, which is being sent onto the laptop, for any irregularities in movement or filmed activities. In case the behaviour of cheating is identified, the system generates an alert to the supervisor through the web interface which triggers immediate action (interventions among students). This approach recognizes that within a given team, there may be efforts by an individual to cheat, and for this reason, all the steps of such are flagged immediately, enhancing integrity without completely depending on human caution.

4.1 Context Diagram

The context diagram is a depiction of system components and users through which they get involved. The exam supervisor provides access to the system via a secure web portal that sends instantaneous information updates on any incidents of cheating picked up. The AI module processes the live video that is streamed to the supervisor's computer from the classroom camera. When the suspicious behaviour is detected, the web portal is triggered to register an alert of the sort to the supervisor.

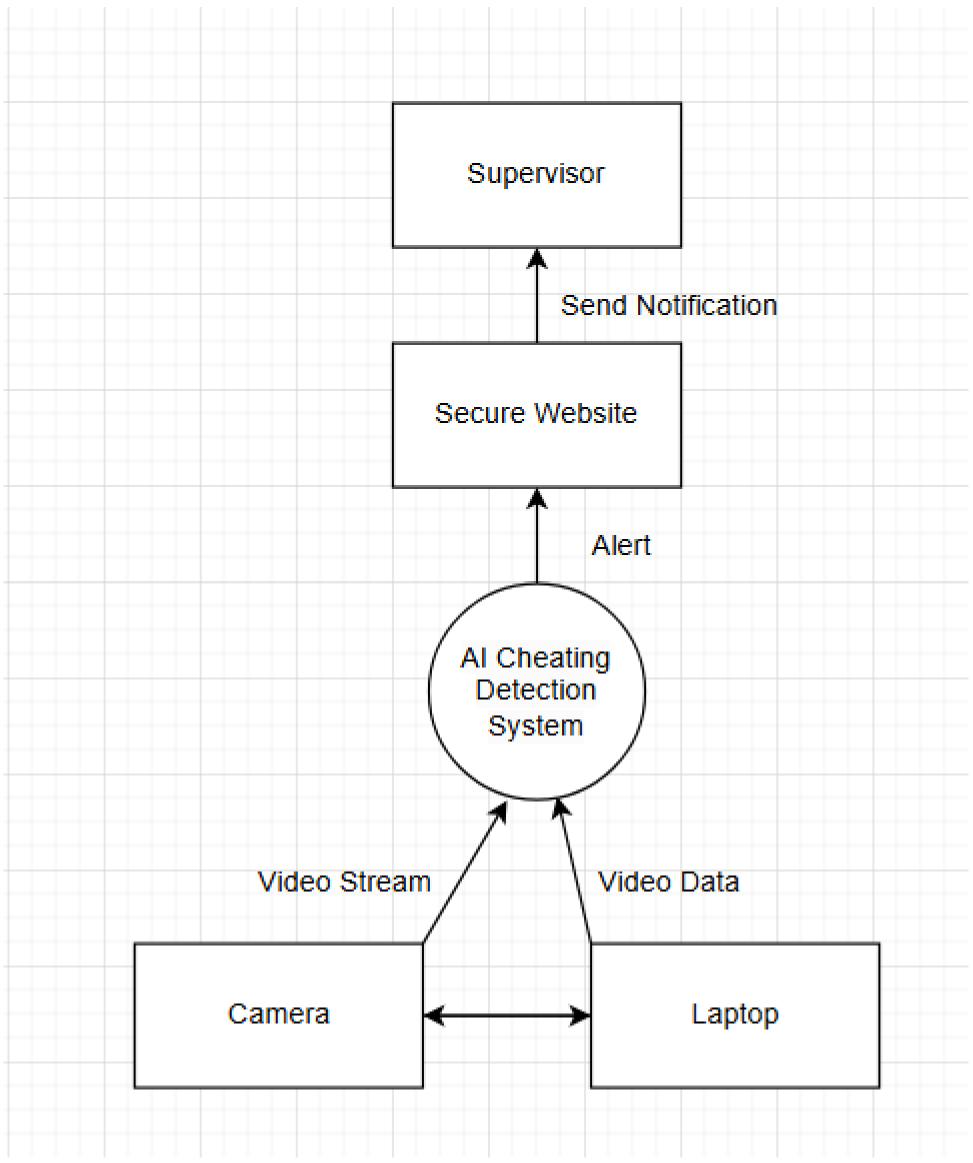


Figure 4.1.1 *Context Diagram*

The video taken by the camera is sent to the artificial intelligence system by the laptop to be analysed. The moment that it finds a suspected case of cheating the system generates an alert to the web application – after which the supervisor is then notified.

4.2 Use Cases

- Monitor Examination
- Detect Cheating Behavior
- View Cheating Notifications
- View Recording History

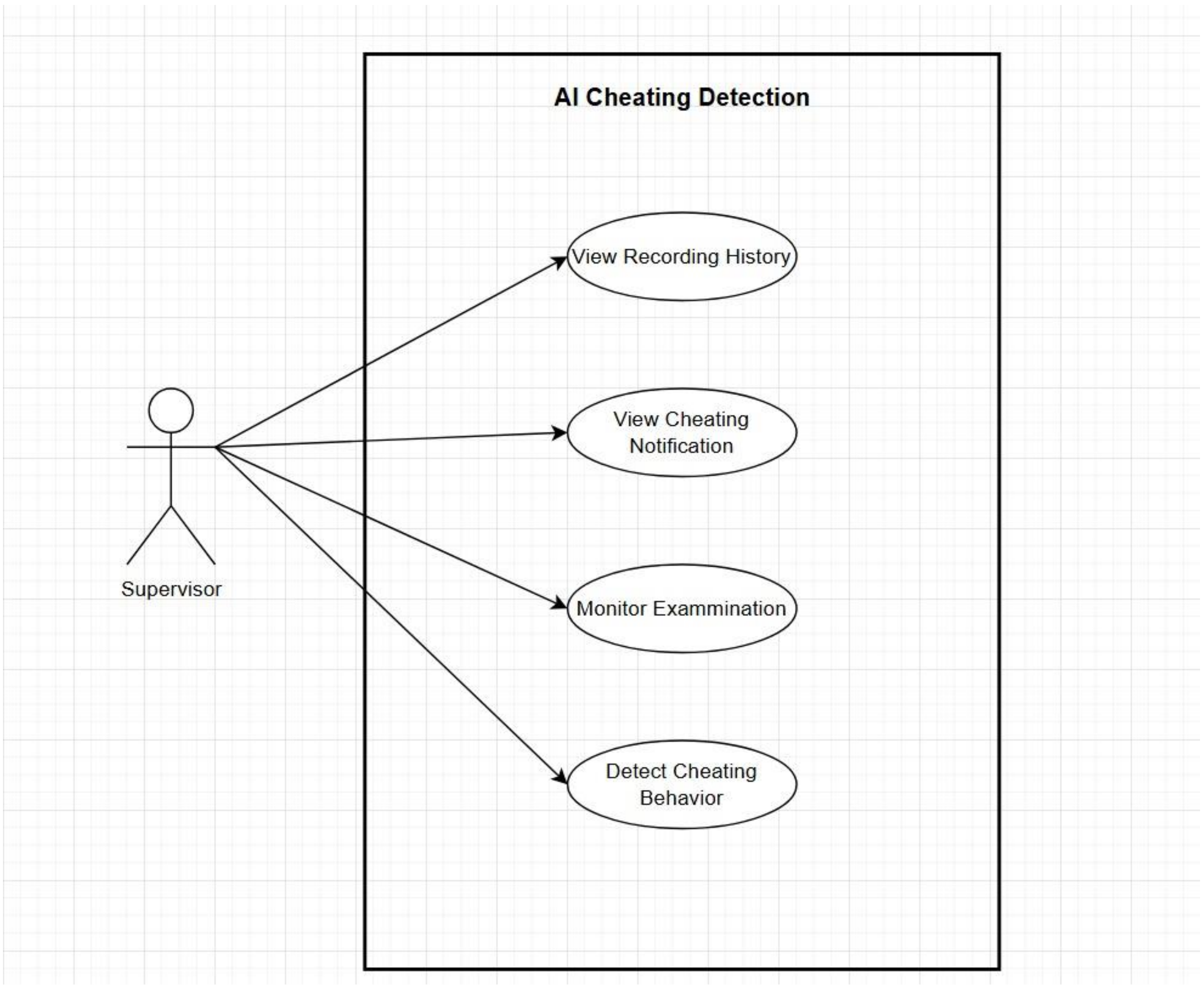


Figure 4.2.1 *Use Case Diagram*

The Supervisor has access to 4 elements which are View Recording History and View Cheating Notification and Monitor Examination and Detect Cheating Behavior.

4.3 Activity Diagram

System Activation → Video Capture → AI Movement Analysis → Suspicious Behavior Detection → Supervisor Alert → Supervisor Action (e.g., relocate student, confiscate exam paper)

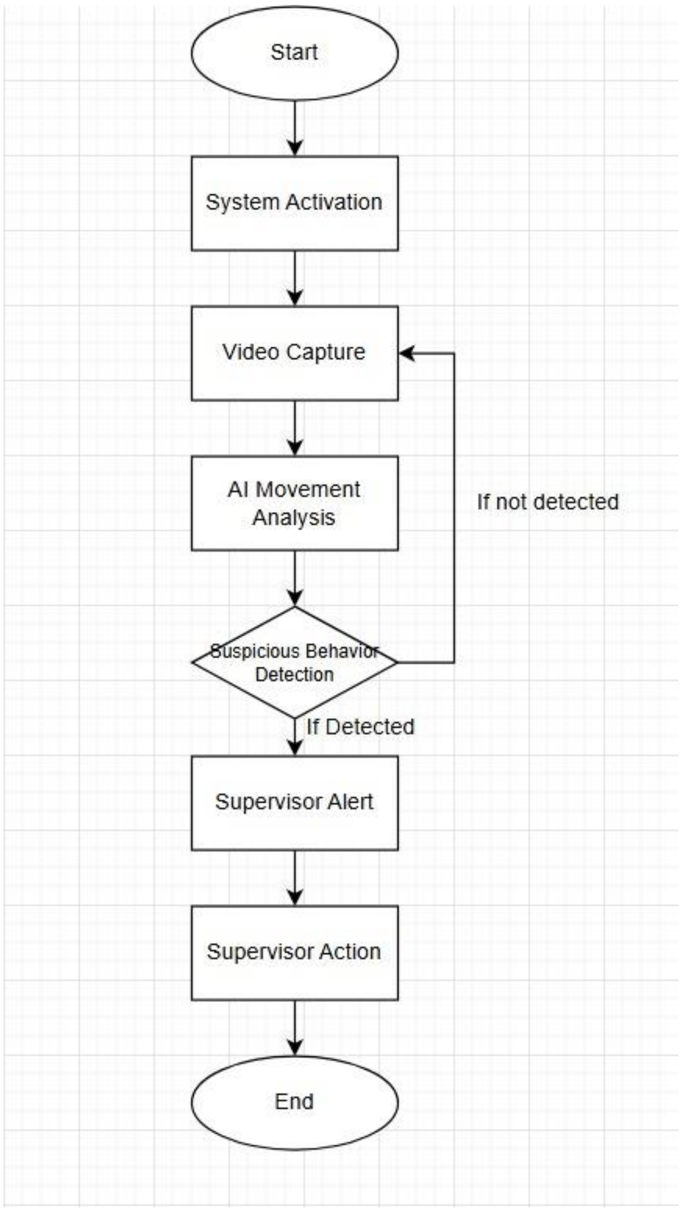


Figure 4.3.1 *Activity Diagram*

First the supervisor will start the system and activate it, then the system will start capturing the video and do the AI Movement Analysis based on the AI trained model and it will check for any suspicious behavior, if it detects any then it will notify or will alert the supervisor, based on the alert the supervisor will take action, and if not detected anything it will continue capturing the video and make the AI Analysis.

4.4 Sequence Diagram

Supervisor accesses website → Connects camera to laptop → Camera streams video → AI analyzes video → Suspicious activity identified → Notification sent to supervisor via website

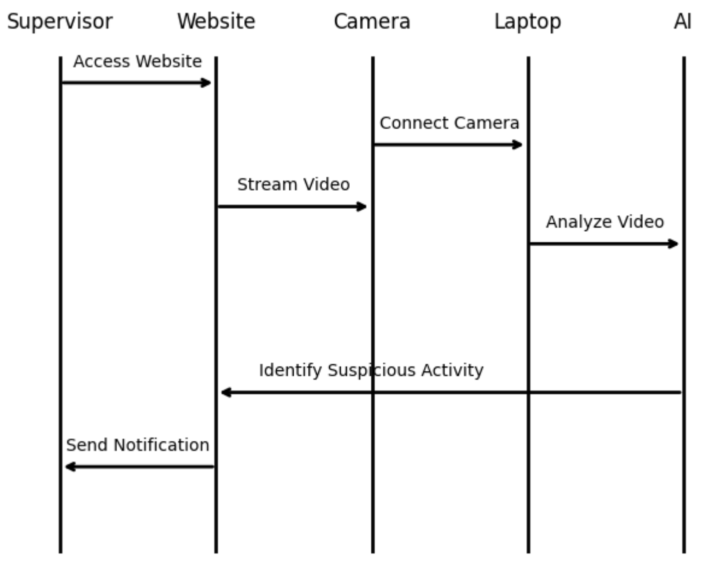


***Figure 4.4.1** Sequence Diagram*

The supervisor will access the website in order to get the session started. At that point, the camera will be connected to the laptop and the video content will start streaming to the website directly. The laptop transfers the video stream to the AI module for the purpose of the analysis. In case the AI detects any criminal activities, a notification will be sent to the supervisor instantly via the website interface. This automatic procedure ensures instant watching of the happenings and immediate action towards likely security risks.

Chapter 5: Implementation

To make sure the Camera Cheating Behavior Detection System project completes successfully, we have planned implementation strategy. The project timeline has been segmented into clear phases, with specific checkpoints and estimated finish dates so as to track progress while also maintaining a smooth workflow. The actual handbook is therefore as shown below:

Data Gathering (Deadline: 20 March 2025))
The task was to succeed by collecting enough of the necessary visual data through stunting and capturing the different cheating behaviors. The entire team worked closely together, thus developing the necessary varied dataset eventually for the model to be of use in the best way.

Image Annotation (Deadline: 23 March 2025)
One of the most significant tasks in this stage is to annotate the collected images. The most important components that stand as the proof of cheating behaviors like bizarre body movements and objects in the room that are not allowed, will be clearly labeled. Every worker must devote their full attention to this step to guarantee accuracy, consistency, and trustworthiness of the training data for all the models.

AI Model Training (Deadline: 25 March 2025)
We will employ the labeled dataset to raise the AI detection model's level using the YOLOv10 technique. Instead of focusing on a particular stage of the process, we will utilize a cyclical system to ensure that we fully develop and then train the AI model. That's how we will reach the maximum accuracy and build the best detections, by changing the parameters each time and turning them properly.

Testing and Validation (Deadline: 1 April 2025)
After finishing the initial training, specific, traceable tests will be set up. Our goal is to test the ability of our model to recognize and correct cheating behaviors in most real cases by examining accuracy, precision, recall, and overall dependability.

Embedding AI Model into the Webpage (Deadline: 7 April 2025)
The success of the implementation phase rests exclusively on the tested and verified AI model being entered into the web page smoothly in such a way that it integrates well with our existing site look and feel. This integrated system on the web portal not only makes exam supervisors aware of the situation in real-time but also allows them to be able to track this entire process in a very user-friendly way during the real-time of the operations.

Implementing HD Camera Integration (Deadline: 10 April 2025)
The thing that should be mentioned is the last in a chain of activities and it can be said that it means consolidating the machine learning AI system with the high-resolution CCTV camera feed showing the video over the internet directly into an AI-based detection system. A further stage in this is the final adjustments at the two ends of the communication choir that will realize clear, uninterrupted data between the camera, laptop, the AI system, and the supervisor.

Chapter 6: Web interface Design and Implementation

6.1 Overview

The AI-based Cheating Detection System for this website is the major point of convergence for teachers and administrative personnel to observe, supervise, and respond to online test cheating cases in real-time. It acts as a go-between between the back-end AI-monitoring engine and end-users to deliver real-time interaction and comments through a friendly interface.

Web applications developed with the user in focus allow the teachers to set up a camera, start and finish the examination monitoring, and get real-time notifications for strange activity that might be reported by the AI algorithm. Also, the platform supports reviewing recorded sessions and viewing incident logs to allow examination assessment and documentation post facto.

Deployment of the website is most crucial to the system's success in general because it makes advanced backend processes—such as video feed analysis and monitoring of behavior—a non-technician-friendly feature. It equates to trainers not having to get too technologically involved just to be able to monitor accordingly, leading to a solution deployable institution-wide, easily up scalable, and maintainable.

6.2 Front-End Design

HTML5, CSS3, and Vanilla JavaScript were used to develop the front-end of the site in the simplest way possible for it to be responsive and easily usable mainly.

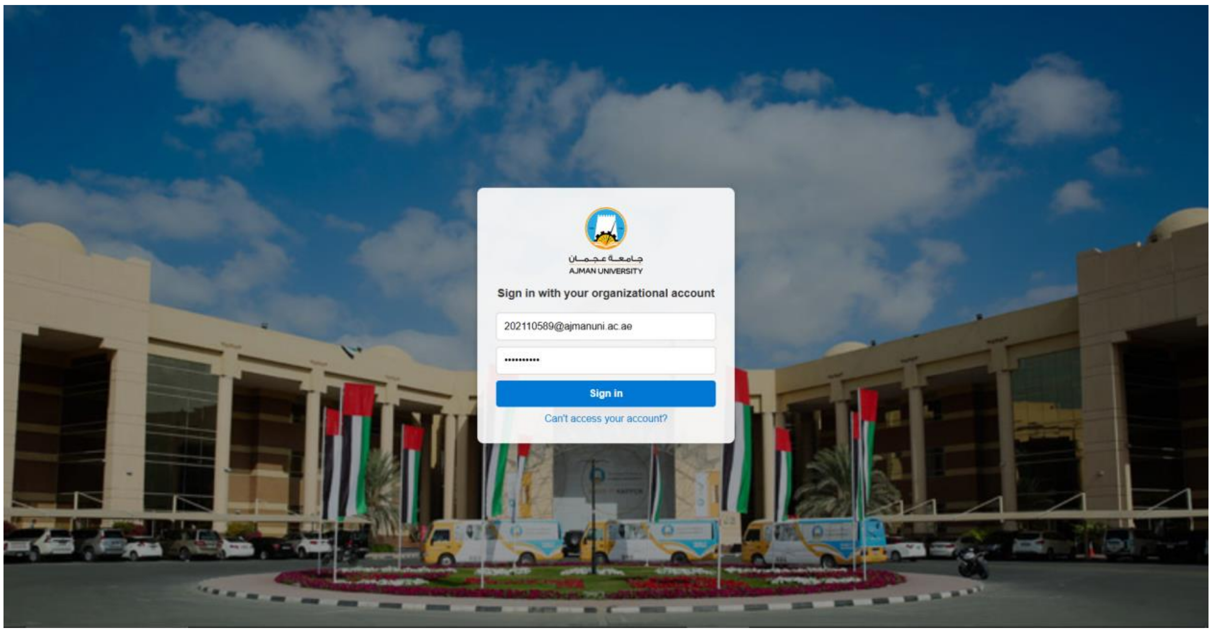
The layout is designed to follow the university's visual style and teachers can easily manage students' performance during tests because it is structured that way.

The top factors taken into consideration were:

- Responsiveness: Built to function flawlessly on desktops and tablets.
- UI/UX: A simple interface, easy to navigate, and consistent styling to reduce cognitive load.
- Feedback & Alerts: Status messages in real-time and redirecting alerts enhance situational awareness.

Here some of the website snapshots: -

6.2.1 Login Page



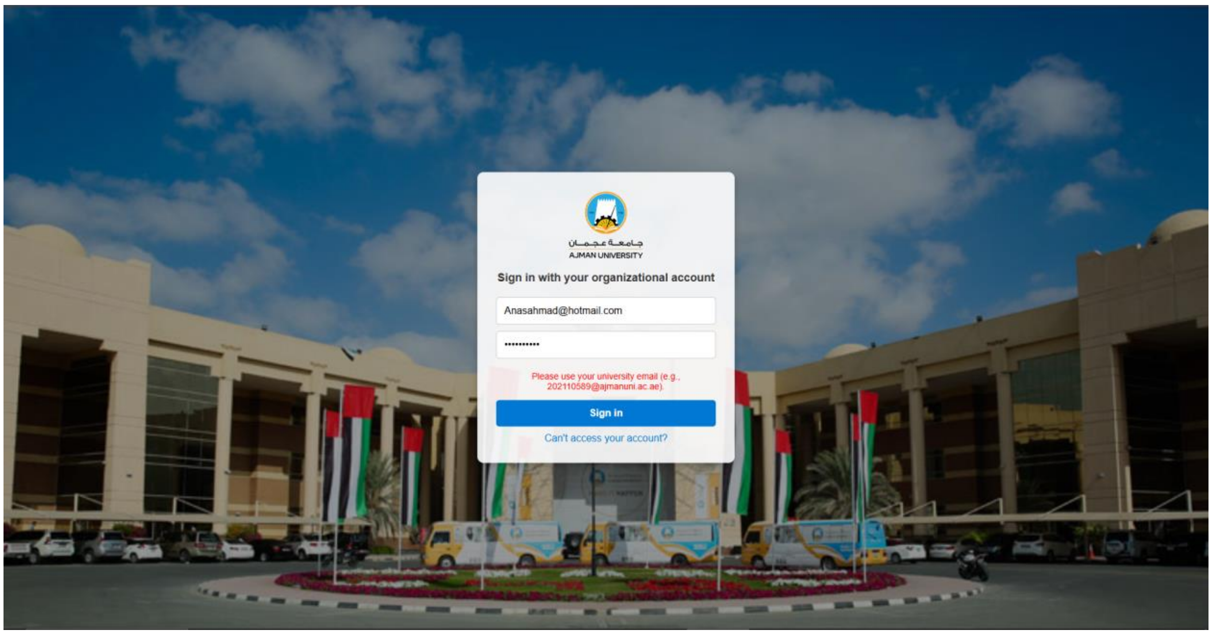
Sign in with your organizational account

202110589@ajmanuni.ac.ae

••••••••••

Figure 6.2.1.1 *Login Interface for Authorized Users*

The login page is the system gateway where only valid instructors or staff can see the monitoring dashboard.



Sign in with your organizational account

Anasahmad@hotmail.com

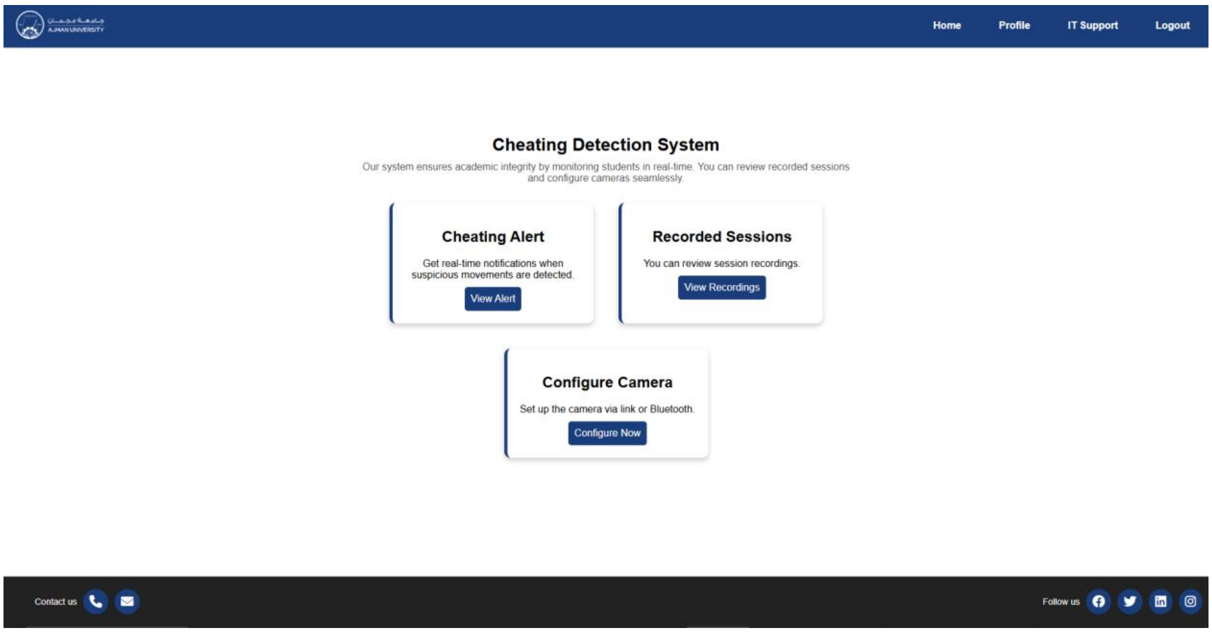
••••••••••

Please use your university email (e.g., 202110589@ajmanuni.ac.ae)

Figure 6.2.1.2 *Login Interface for Unauthorized Users*

It has secure password and email fields with visual feedback on error if the user enters a wrong password and email.

6.2.2 Home Dashboard



Cheating Detection System

Our system ensures academic integrity by monitoring students in real-time. You can review recorded sessions and configure cameras seamlessly.

Cheating Alert

Get real-time notifications when suspicious movements are detected.

View Alert

Recorded Sessions

You can review session recordings.

View Recordings

Configure Camera

Set up the camera via link or Bluetooth.

Configure Now

Figure 6.2.2.1 *Main Dashboard with Feature Access*

Upon login, a one-dashboard is displayed to users, and it contains fast reach of core functions composed of alert detection for cheating cases, camera settings to configure the camera to the supervisor's laptop, and recordings so he is able so look back if any cheating cases happen or if he finds any to similar papers he can check the recording and then he will have proof that the cheater where cheating during the exam and the proof is the recording.

6.2.3 Camera Configuration Page

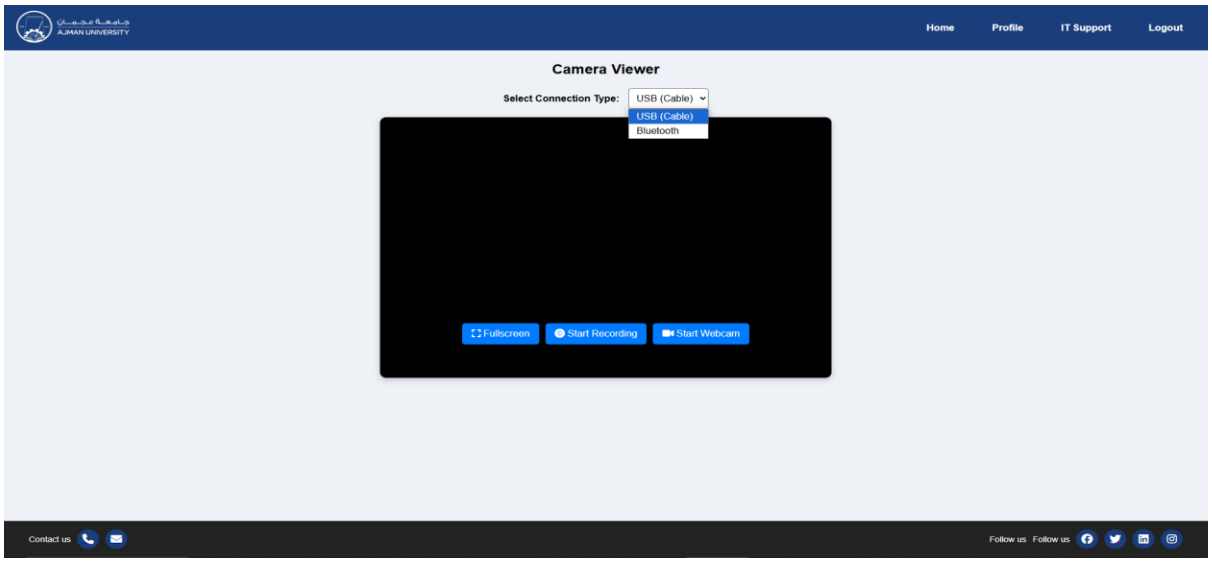


Figure 6.2.3.1 *Camera Configuration with Connection Type Selection*

This page provides an option to use USB connection and start webcam streams for testing purposes because we do not have the budget for connected using an actual camera which is preferred a CCTV camera. Controls also provide full screen toggling and recording capability.

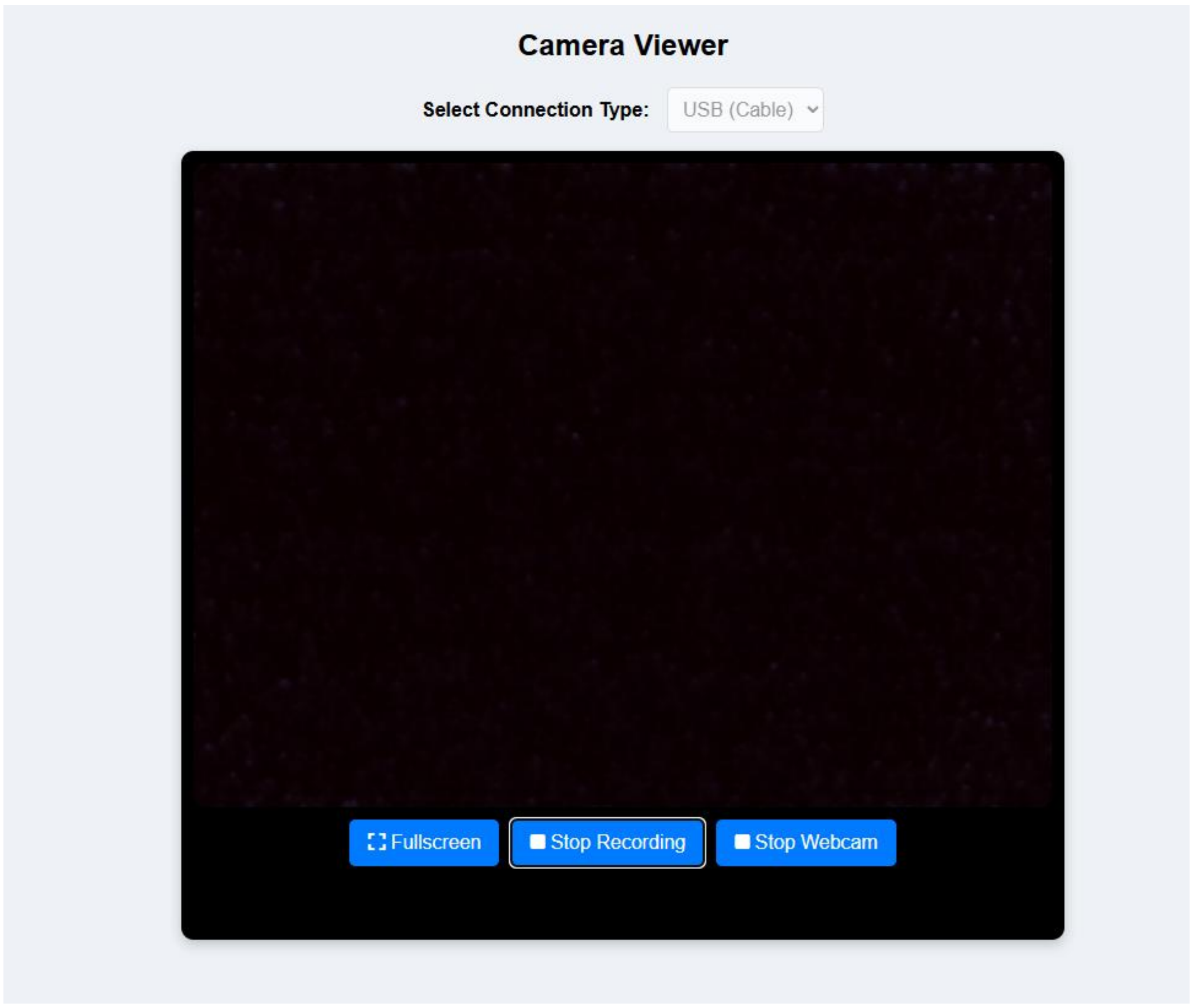
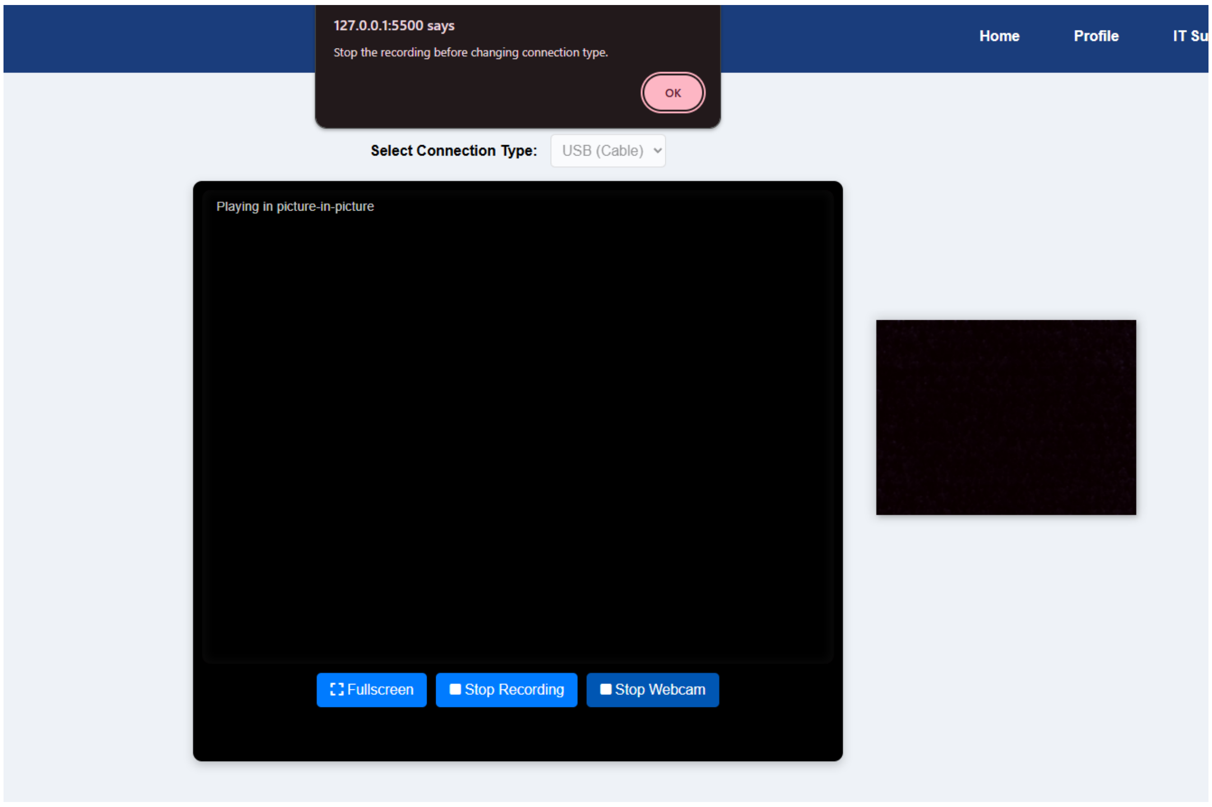


Figure 6.2.3.2 *Recording Features*

Here the user or supervisor can start the recoding manually and have another feature which is the Fullscreen where the user can adjust the screen as he like.

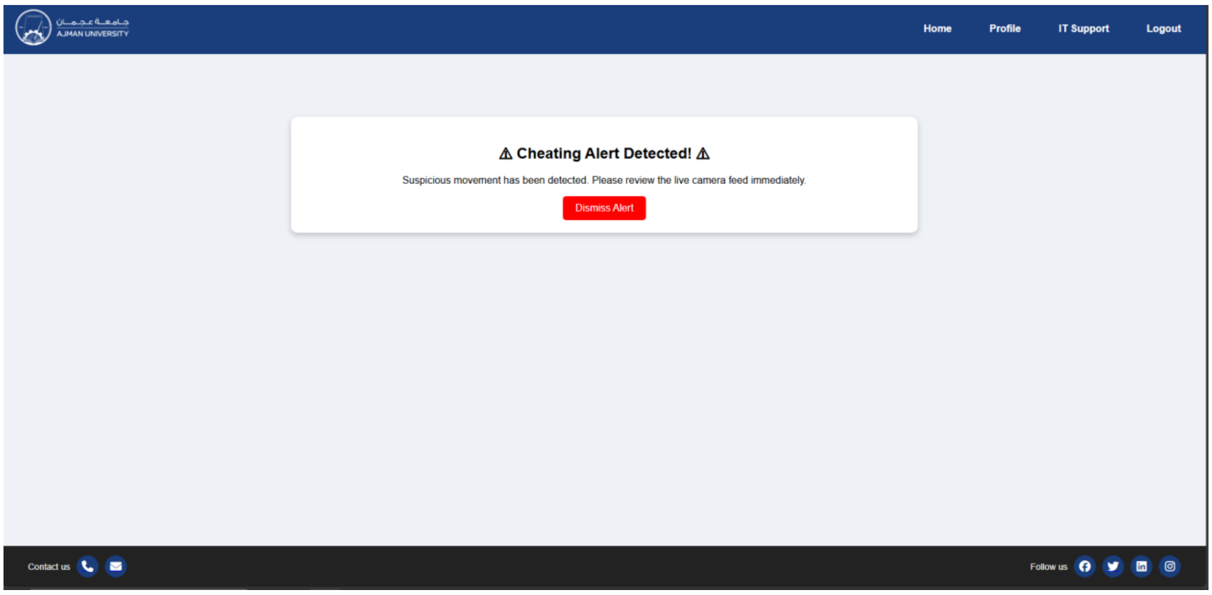

127.0.0.1:5500 says

Stop the recording before changing connection type.

Figure 6.2.3.3 *View Adjustment and Recording Error*

Here the user can go and browse or do any activity with he's laptop as he wishes because we have a great feature which is picture-in-picture where it creates small square, and he can watch the class from there. Additionally, if the user by mistake clicks on stop webcam there is an error message will appear and tell the user to close the camera first, so he's saved recoding don't disappear and be saved in the local machine.

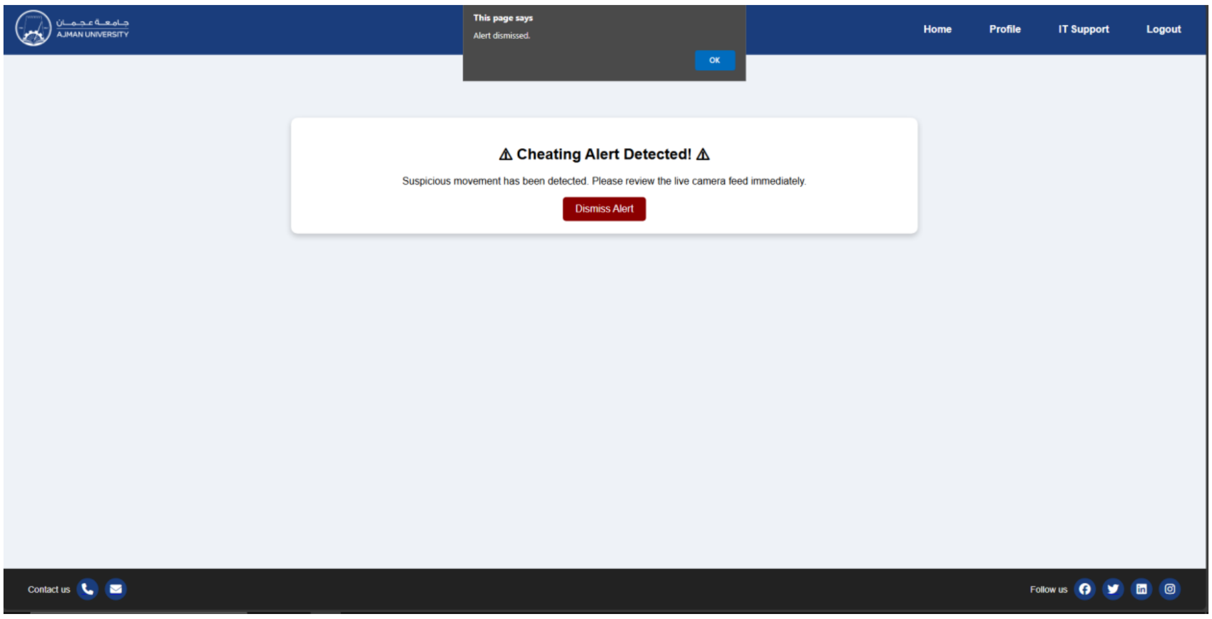
6.2.4 Alert Detection Interface


⚠ Cheating Alert Detected! ⚠

Suspicious movement has been detected. Please review the live camera feed immediately.

Figure 6.2.4.1 *View Adjustment and Recording Error*

Here the user can access cheating alerts in an organized structure for review and confirmation. Alerts can be dismissed by clicking "Dismiss Alert" to save space and get rid of unnecessary storage


AJMAN UNIVERSITY

Home Profile IT Support Logout

This page says

Alert dismissed.

OK

⚠ Cheating Alert Detected! ⚠

Suspicious movement has been detected. Please review the live camera feed immediately.

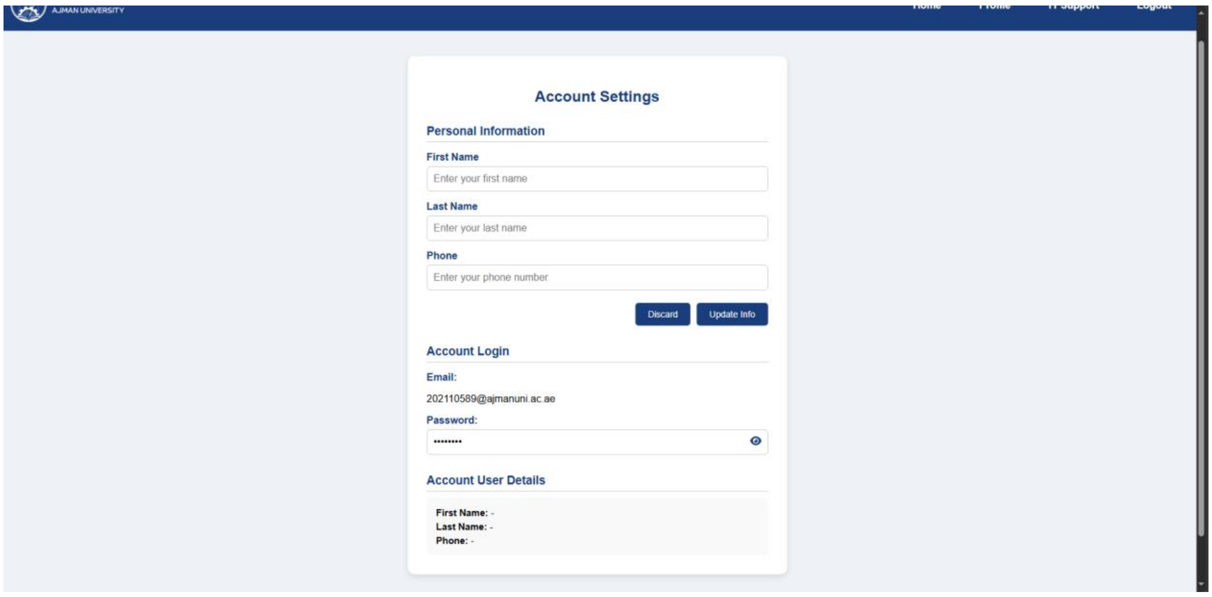
Dismiss Alert

Contact us

Follow us

Figure 6.2.4.2 *Cheating Alert dismissal*

6.2.5 Profile Settings Page


Account Settings

Personal Information

First Name

Enter your first name

Last Name

Enter your last name

Phone

Enter your phone number

Discard Update Info

Account Login

Email:

202110589@ajmanuni.ac.ae

Password:

••••••••

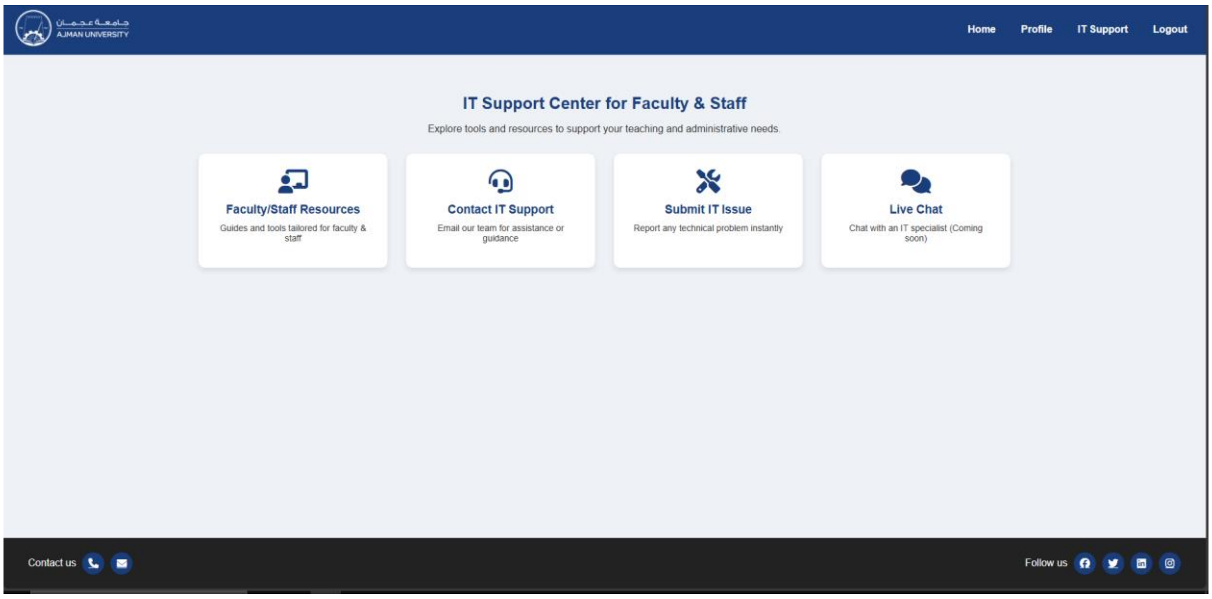
Account User Details

First Name: -
Last Name: -
Phone: -

Figure 6.2.5.1 *Account settings configuration*

This page allows the user to update and confirm any changes to his account settings and credentials, changes can either be discarded or confirmed by clicking “Update Info”. Current details are display in the Account User Details section.

6.2.6 IT Support Page



IT Support Center for Faculty & Staff

Explore tools and resources to support your teaching and administrative needs.

Faculty/Staff Resources: Guides and tools tailored for faculty & staff

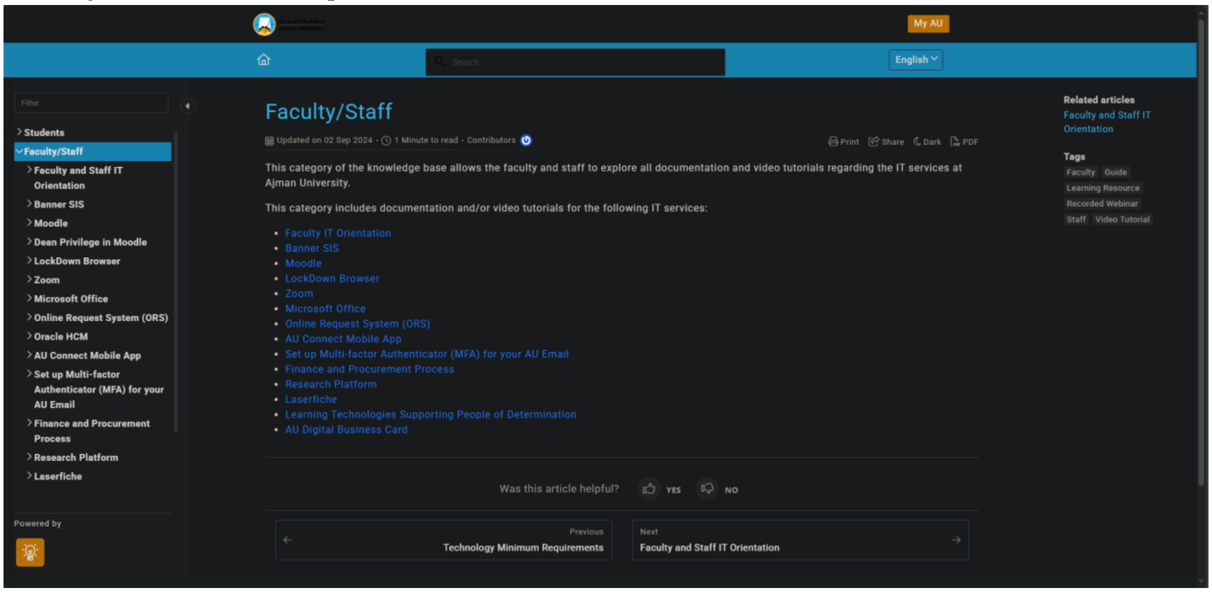
Contact IT Support: Email our team for assistance or guidance

Submit IT Issue: Report any technical problem instantly

Live Chat: Chat with an IT specialist (Coming soon)

Figure 6.2.6.1 *IT Support Page Overview*

Aligning with Ajman University's excellent Student Support, we have implemented an IT Support Centre providing clear instructions, resources for the user to take advantage of and receive directly from Ajman Universities repositories.



Faculty/Staff

Updated on 02 Sep 2024 · 1 Minute to read · Contributors

This category of the knowledge base allows the faculty and staff to explore all documentation and video tutorials regarding the IT services at Ajman University.

This category includes documentation and/or video tutorials for the following IT services:

- Faculty IT Orientation
- Banner SIS
- Moodle
- LockDown Browser
- Zoom
- Microsoft Office
- Online Request System (ORS)
- AU Connect Mobile App
- Set up Multi-factor Authenticator (MFA) for your AU Email
- Finance and Procurement Process
- Research Platform
- Laserfiche
- Learning Technologies Supporting People of Determination
- AU Digital Business Card

Was this article helpful? YES NO

Previous: Technology Minimum Requirements

Next: Faculty and Staff IT Orientation

Figure 6.2.6.2 *IT Support Faculty/Staff Resources Link*

The "**Faculty/Staff Resources"** button is a hyperlink to the official Ajman University Faculty/Staff section.

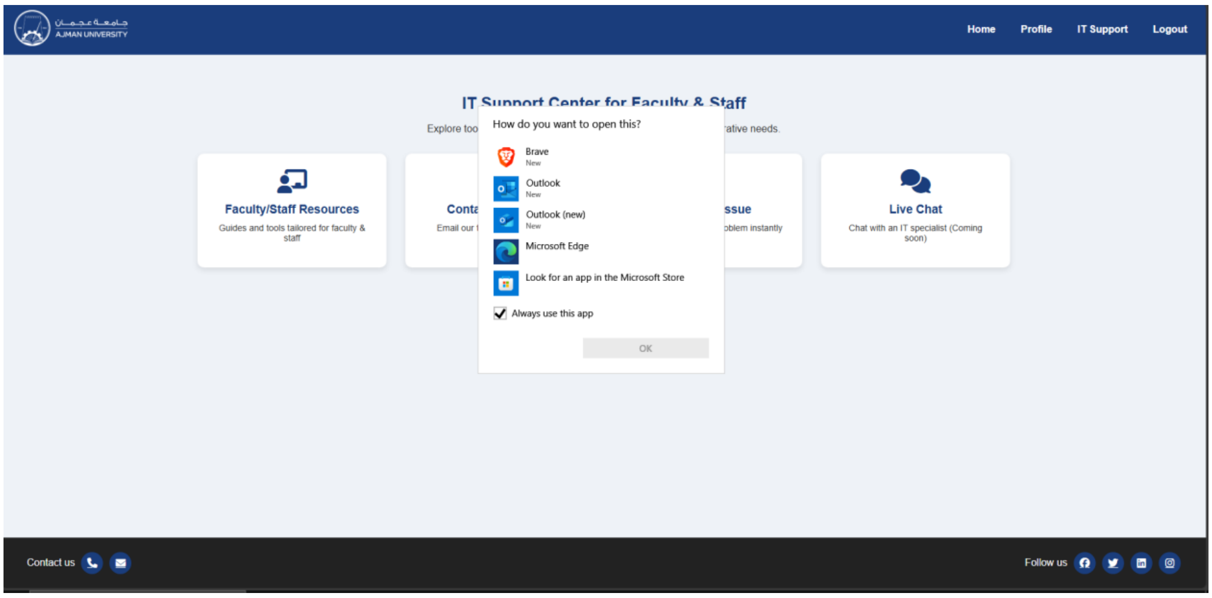

AJMAN UNIVERSITY
Home
Profile
IT Support
Logout
IT Support Center for Faculty & Staff
How do you want to open this?
Brave
Outlook
Outlook (new)
Microsoft Edge
Look for an app in the Microsoft Store
Always use this app
OK
Faculty/Staff Resources
Guides and tools tailored for faculty & staff
Live Chat
Chat with an IT specialist (Coming soon)
Contact us
Follow us

Figure 6.2.6.3 *IT Support Contact Link*

This button launches outlook or other default email programs to send an email ticket directly to IT Support.

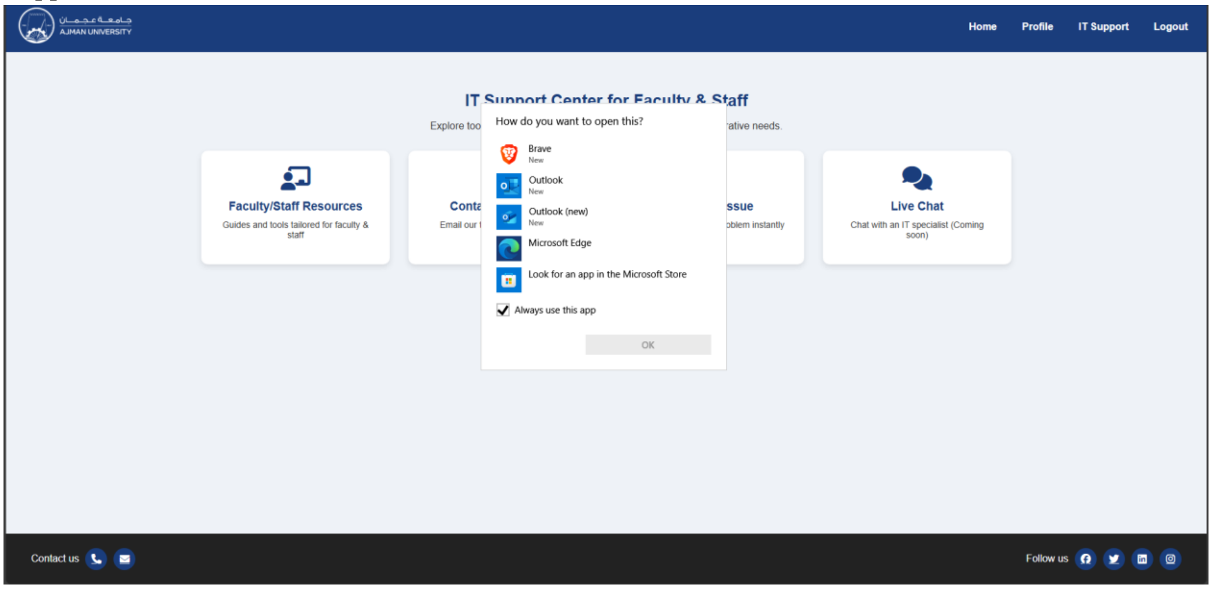
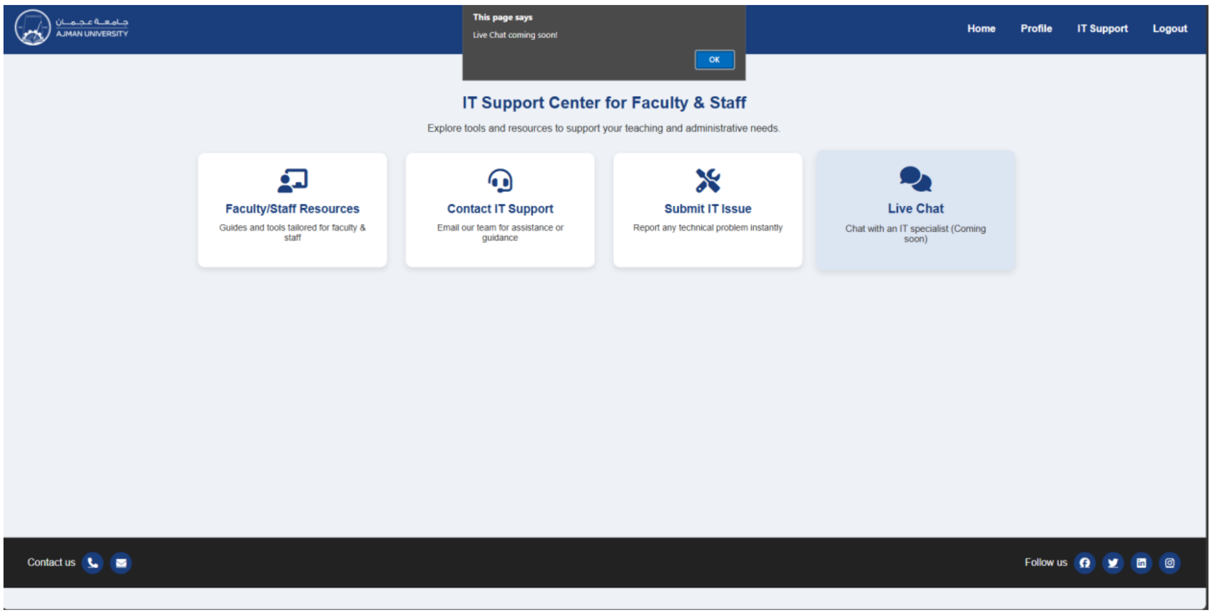

AJMAN UNIVERSITY
Home
Profile
IT Support
Logout
IT Support Center for Faculty & Staff
How do you want to open this?
Brave
Outlook
Outlook (new)
Microsoft Edge
Look for an app in the Microsoft Store
Always use this app
OK
Faculty/Staff Resources
Guides and tools tailored for faculty & staff
Live Chat
Chat with an IT specialist (Coming soon)
Contact us
Follow us

Figure 6.2.6.4 *IT Support Technical Issue*

Similarly, Technical Issue button is used to report bugs or other technical difficulties directly to our IT Team.



This page says
Live Chat coming soon!

IT Support Center for Faculty & Staff

Explore tools and resources to support your teaching and administrative needs.

Faculty/Staff Resources
Guides and tools tailored for faculty & staff

Contact IT Support
Email our team for assistance or guidance

Submit IT Issue
Report any technical problem instantly

Live Chat
Chat with an IT specialist (Coming soon)

Figure 6.2.6.5 *Live Chat with an IT Specialist*

Lastly, our Work-In-Progress feature which we hope to deliver in the near future, a Live Chatting system that facilitates direct instructions and guidance to the user in an automated and user-friendly manner.

6.3 Back-End Development & Model training

6.3.1 Back-End Development

The processing of detection results and communications between the web interface and the AI model are the responsibilities of the back end of the system.
This, however, includes getting outputs from the AI model (a "cheating detected" event) and logging them to a database and using them to trigger events to the front-end for the supervisor to see. We coded the back-end logic in Python which interfaced with the front-end through a light weighted web framework and implemented a real time database (Firebase) to send notification messages to web client instantly.

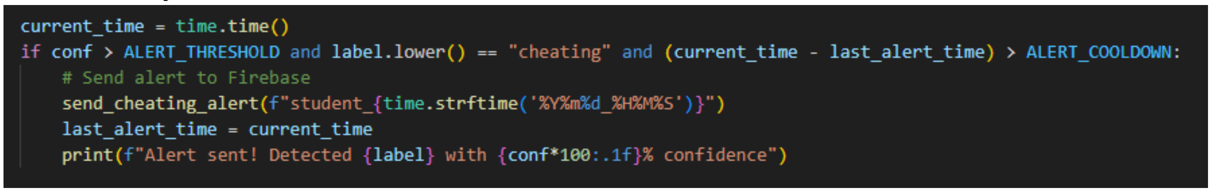
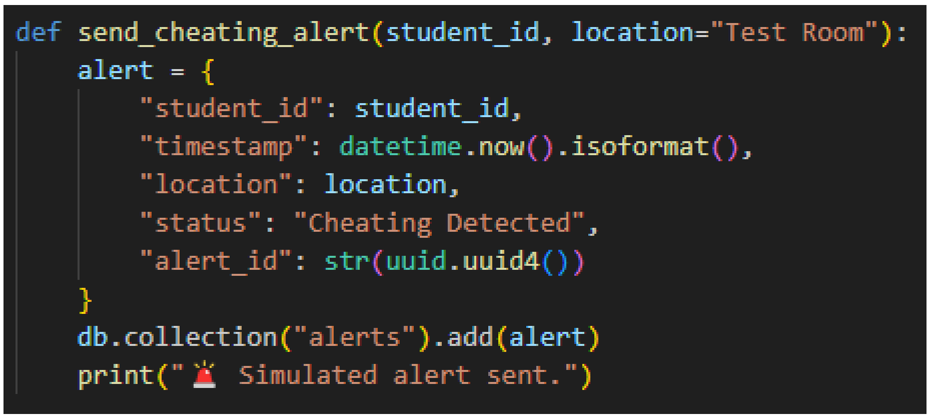

```
current_time = time.time()
if conf > ALERT_THRESHOLD and label.lower() == "cheating" and (current_time - last_alert_time) > ALERT_COOLDOWN:
    # Send alert to Firebase
    send_cheating_alert(f"student_{time.strftime('%Y%m%d_%H%M%S')}")
    last_alert_time = current_time
    print(f"Alert sent! Detected {label} with {conf*100:.1f}% confidence")
```

Figure 6.3.1.1 *Cheating Alert Pseudocode*

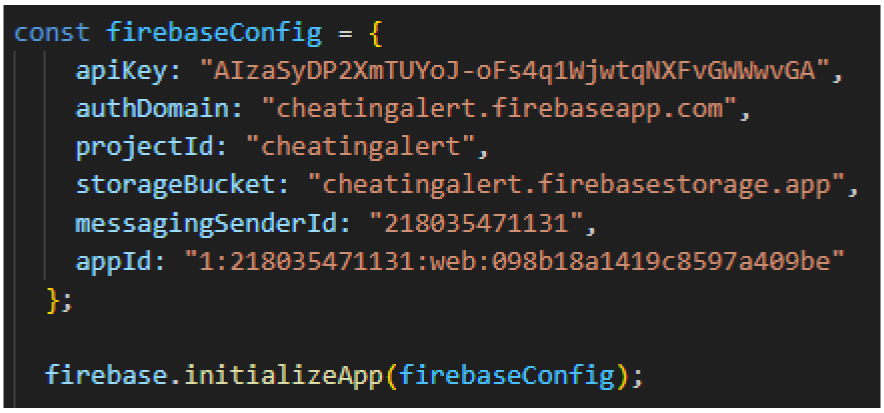
First of all, the system checks the actual time and evaluates the output of the AI model. If the confidence at which the model detects behaviour is above 85% and cheating is recognized, the behaviour is triggered by back-end; function to initiate an alert.



```python
def send_cheating_alert(student_id, location="Test Room"):
    alert = {
        "student_id": student_id,
        "timestamp": datetime.now().isoformat(),
        "location": location,
        "status": "Cheating Detected",
        "alert_id": str(uuid.uuid4())
    }
    db.collection("alerts").add(alert)
    print("🚨 Simulated alert sent.")
```

Figure 6.3.1.2 *Cheating Alert Function*

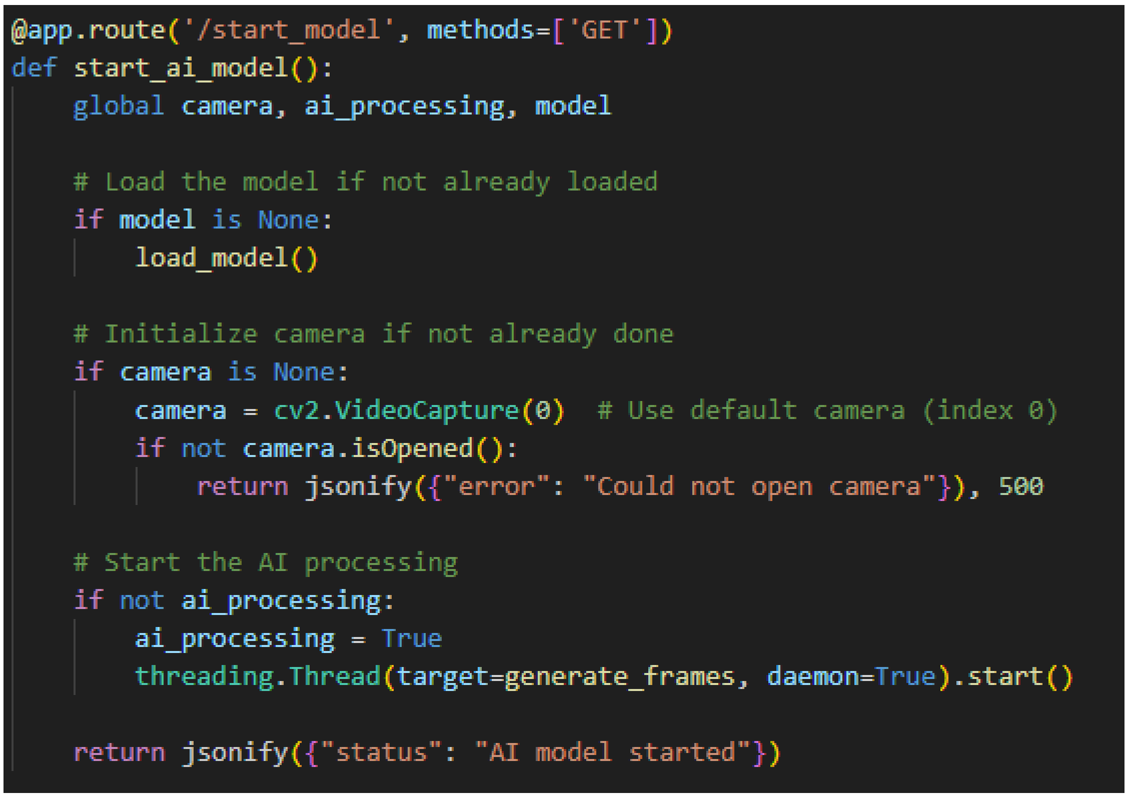
This function takes care of the sending out of the cheating alert. It makes an alert entry with details (timestamp, type of cheating strings) and adds it to the database. It also prepares erroneously the notification payload response that will be transported to the web interface for the supervisor.



```javascript
const firebaseConfig = {
    apiKey: "AIzaSyDP2XmTUYoJ-oFs4q1WjwtqNXFvGWWwvGA",
    authDomain: "cheatingalert.firebaseapp.com",
    projectId: "cheatingalert",
    storageBucket: "cheatingalert.firebasestorage.app",
    messagingSenderId: "218035471131",
    appId: "1:218035471131:web:098b18a1419c8597a409be"
  };

  firebase.initializeApp(firebaseConfig);
```

Figure 6.3.1.3 *Firebase Database Configuration*

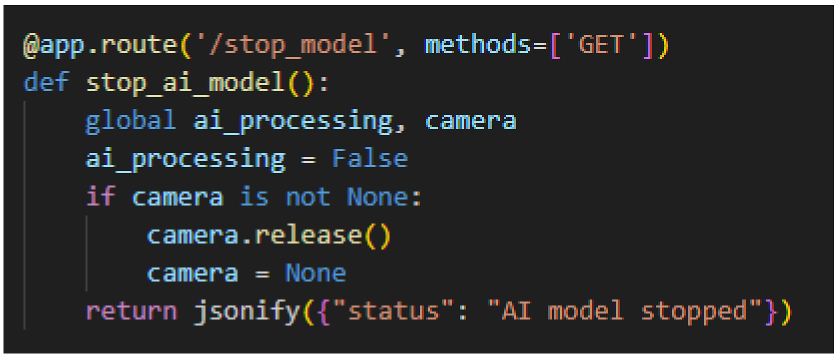
It has an internal back end which consists of a Firebase real-time database. This code illustrates steps used for initialisation and connection set up in Firebase. It produces all required database keys and security protocols for an application to save and retrieve alert record data. Once configured, once any notification enters the database it gets synchronized and is made available for any display in the client-side interface for the supervisor to view.



```python
@app.route('/start_model', methods=['GET'])
def start_ai_model():
    global camera, ai_processing, model

    # Load the model if not already loaded
    if model is None:
        load_model()

    # Initialize camera if not already done
    if camera is None:
        camera = cv2.VideoCapture(0)  # Use default camera (index 0)
        if not camera.isOpened():
            return jsonify({"error": "Could not open camera"}), 500

    # Start the AI processing
    if not ai_processing:
        ai_processing = True
        threading.Thread(target=generate_frames, daemon=True).start()

    return jsonify({"status": "AI model started"})
```

Figure 6.3.1.4 *Flask Route to Start the AI Model and Camera Feed*

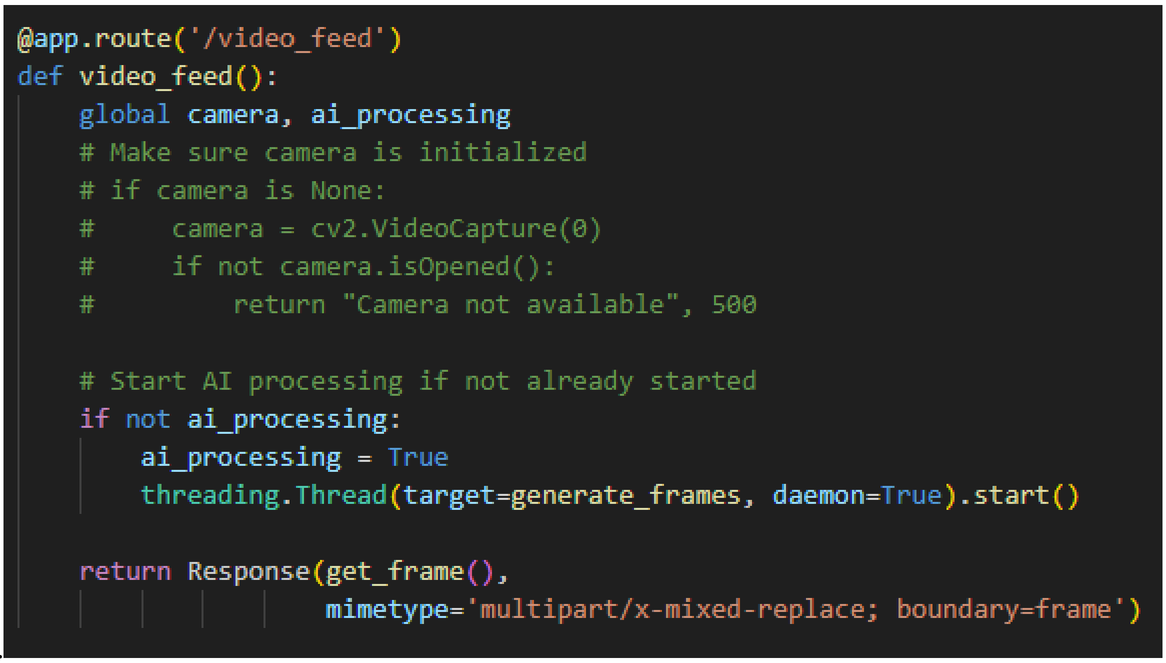
The Python function presented is responsible for uploading the artificial intelligence model and opening the system's camera, if it has never been used. The GET request is being listened for at the endpoint **/start_model**. The function when called first returns if the AI model is successfully loaded by calling **load_model()**. Then it initiates the camera via **cv2.VideoCapture(0)** and examines whether it works. In the event of failure of the camera to open, the system returns a 500-error. After successful initiation of the model and the camera, the function begins the thread of frame generation, which allows for real time processing. This enables the system to begin detecting behaviour from live video streams.



```python
@app.route('/stop_model', methods=['GET'])
def stop_ai_model():
    global ai_processing, camera
    ai_processing = False
    if camera is not None:
        camera.release()
        camera = None
    return jsonify({"status": "AI model stopped"})
```

Figure 6.3.1.5 *Flask Route to Stop the AI Model and Release Resources*

This step, properly, brings an end to the artificial intelligence stream processing, and frees up the hardware resources that have been reserved for use in the camera. It answers the **/stop_model** GET endpoint by rendering the **ai_processing** flag FALSE hence terminating any active AI inference procedures. When the camera is called in some circumstances it is deactivated by use of the OpenCV release function (**camera.release ()**) and the camera reference destroyed. The process guarantees that the system reverts to the normal unrestricted state, removes memory leaks and makes future sessions easy.

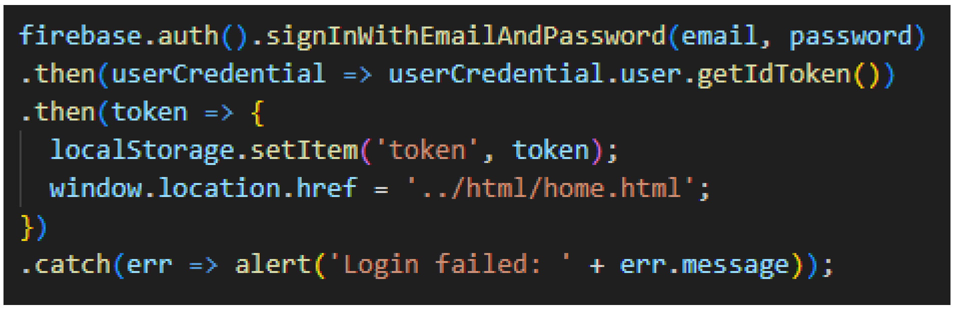


```python
@app.route('/video_feed')
def video_feed():
    global camera, ai_processing
    # Make sure camera is initialized
    # if camera is None:
    #     camera = cv2.VideoCapture(0)
    #     if not camera.isOpened():
    #         return "Camera not available", 500

    # Start AI processing if not already started
    if not ai_processing:
        ai_processing = True
        threading.Thread(target=generate_frames, daemon=True).start()

    return Response(get_frame(),
                    mimetype='multipart/x-mixed-replace; boundary=frame')
```

.

Figure 6.3.1.6 *Video Feed Streaming Route for Web Display*

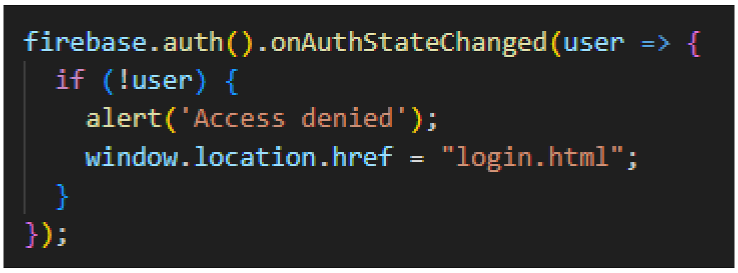
The **/video_feed** route allows the streaming of camera frames to the web interface streaming continuously. It response with **multipart/x-mixed – replace** HTTP response, which browsers will utilize for the rendering of a live video feed. Even though camera initialization is commented out in this function it is assumed that the camera is already initialized. Unless there is already running AI processing, a separate thread is started for frame generation. The backend stream has a straight connection with the user interface element used for the live video rendition for behavioural analysis.



```javascript
firebase.auth().signInWithEmailAndPassword(email, password)
.then(userCredential => userCredential.user.getIdToken())
.then(token => {
  localStorage.setItem('token', token);
  window.location.href = '../html/home.html';
})
.catch(err => alert('Login failed: ' + err.message));
```

Figure 6.3.1.7 *Firebase Email/Password Login and Token Generation*

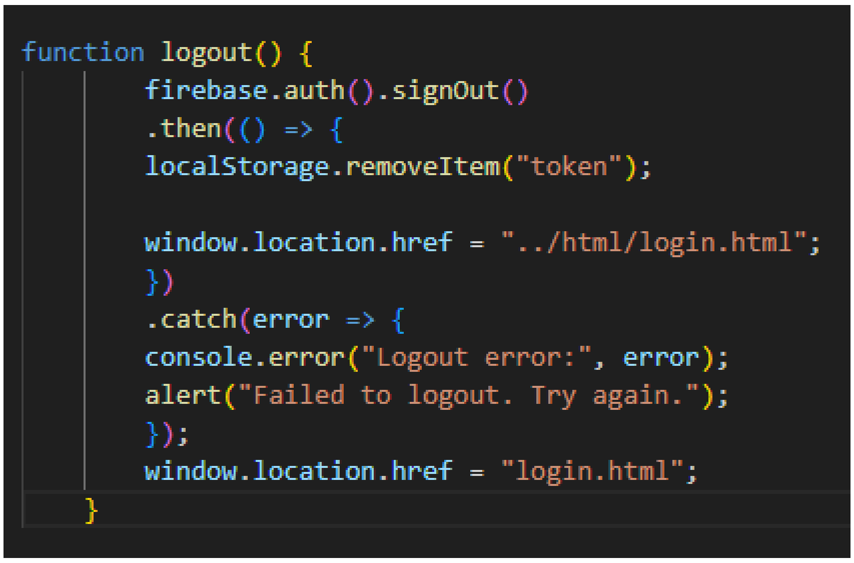
This JavaScript function will log-in users by using email and password through the Firebase Authentication. It retrieves a session token, (**getIdToken()**), saves it in the local storage in the browser then redirects the user to a home page after successful authentication. This token is used to verify user identity for secure websites and to prevent unauthorized posted in the AI dashboards.



```
firebase.auth().onAuthStateChanged(user => {
  if (!user) {
    alert('Access denied');
    window.location.href = "login.html";
  }
});
```

Figure 6.3.1.8 *Firebase Auth State Monitoring for Session Validation*

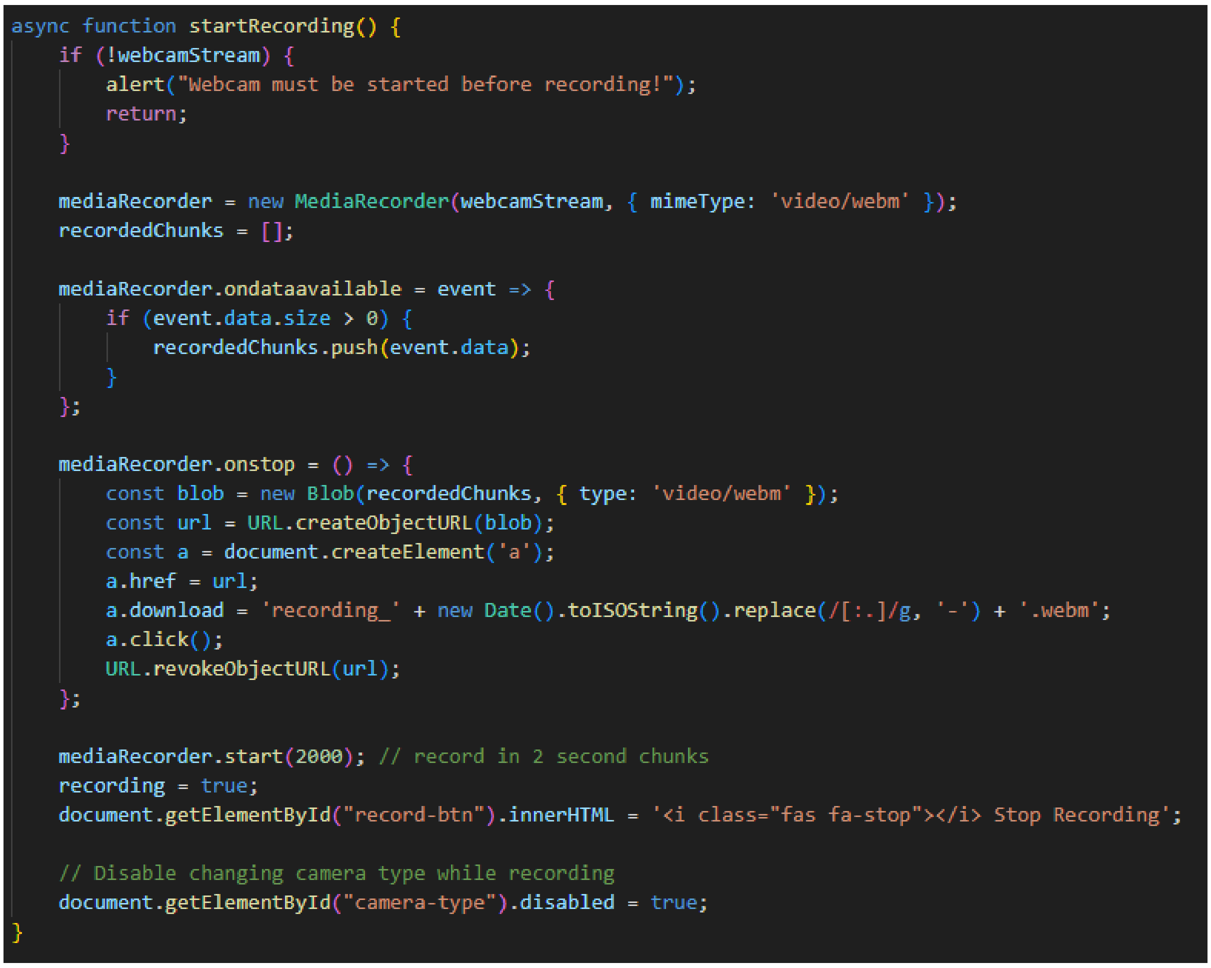
This way ensures that access to protected pages is only accessible by the authorized users. If an opening user lacks a valid session, the function will redirect to the login page and the message access denied will be shown. This is a basic mechanism of security for ensuring the integrity of the maintaining session throughout the application life cycle.



```
function logout() {
        firebase.auth().signOut()
        .then(() => {
        localStorage.removeItem("token");

        window.location.href = "../html/login.html";
        })
        .catch(error => {
        console.error("Logout error:", error);
        alert("Failed to logout. Try again.");
        });
        window.location.href = "login.html";
    }
```

Figure 6.3.1.9 *Firebase Logout Function with Token Destruction*

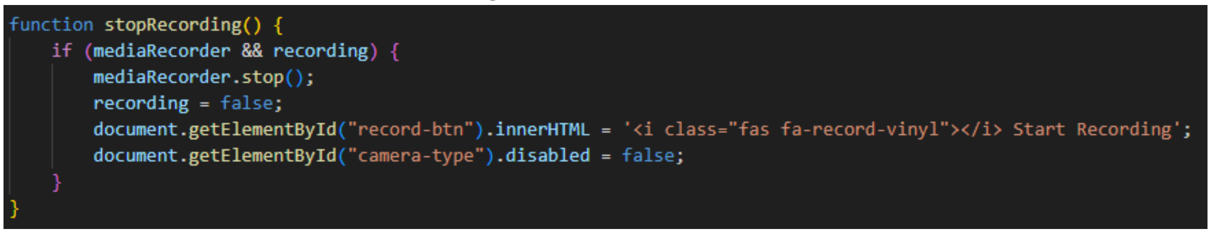
Upon calling this function, it logs out user session using Firebase's **signOut()** function and removes the token from the local storage. Upon the logout the user is redirected to the login page. In addition, it has error handling capacities built in it that provide it with the ability to catch and illustrate any logout errors that may be experienced enhancing the robust end user experience.



```
async function startRecording() {
    if (!webcamStream) {
        alert("Webcam must be started before recording!");
        return;
    }

    mediaRecorder = new MediaRecorder(webcamStream, { mimeType: 'video/webm' });
    recordedChunks = [];

    mediaRecorder.ondataavailable = event => {
        if (event.data.size > 0) {
            recordedChunks.push(event.data);
        }
    };

    mediaRecorder.onstop = () => {
        const blob = new Blob(recordedChunks, { type: 'video/webm' });
        const url = URL.createObjectURL(blob);
        const a = document.createElement('a');
        a.href = url;
        a.download = 'recording_' + new Date().toISOString().replace(/[:.]/g, '-') + '.webm';
        a.click();
        URL.revokeObjectURL(url);
    };

    mediaRecorder.start(2000); // record in 2 second chunks
    recording = true;
    document.getElementById("record-btn").innerHTML = '<i class="fas fa-stop"></i> Stop Recording';

    // Disable changing camera type while recording
    document.getElementById("camera-type").disabled = true;
}
```

Figure 6.3.1.10 *Web-Based Screen Recording Initialization Using MediaRecorder*

This asynchronous function makes it possible to record a screen in a session. It initiates a new instance of a **MediaRecorder**, whose MIME type will be **video/webm**. The mentioned function is always on the lookout for incoming segments of data. After recording has finished, it combines all segments into a downloadable file in **WebM** format and initiates its automatic download. The function also disables switching cameras while recording to maintain the capture integrity of data during an examination session.



```
function stopRecording() {
    if (mediaRecorder && recording) {
        mediaRecorder.stop();
        recording = false;
        document.getElementById("record-btn").innerHTML = '<i class="fas fa-record-vinyl"></i> Start Recording';
        document.getElementById("camera-type").disabled = false;
    }
}
```

Figure 6.3.1.11 *Stop Recording Function and UI State Reset*

This JavaScript function stops the active screen recording session and resets the UI. It reverts the recording button's icon and re-enables the camera selection input. By halting the **MediaRecorder** instance and allowing user interface updates, this ensures the system provides clear feedback and prepares the interface for future recording sessions.

Chapter 7: AI Model

7.1 AI Model Training process

In this chapter I will talk about the steps we took to train our model; first step is data gathering.

7.2 Data Gathering

For data gathering we divided the data into 2 parts:

1. Generated our own data (by taking pictures and videos)
2. Get data from open sources

Example of generating our own data is the figures below:

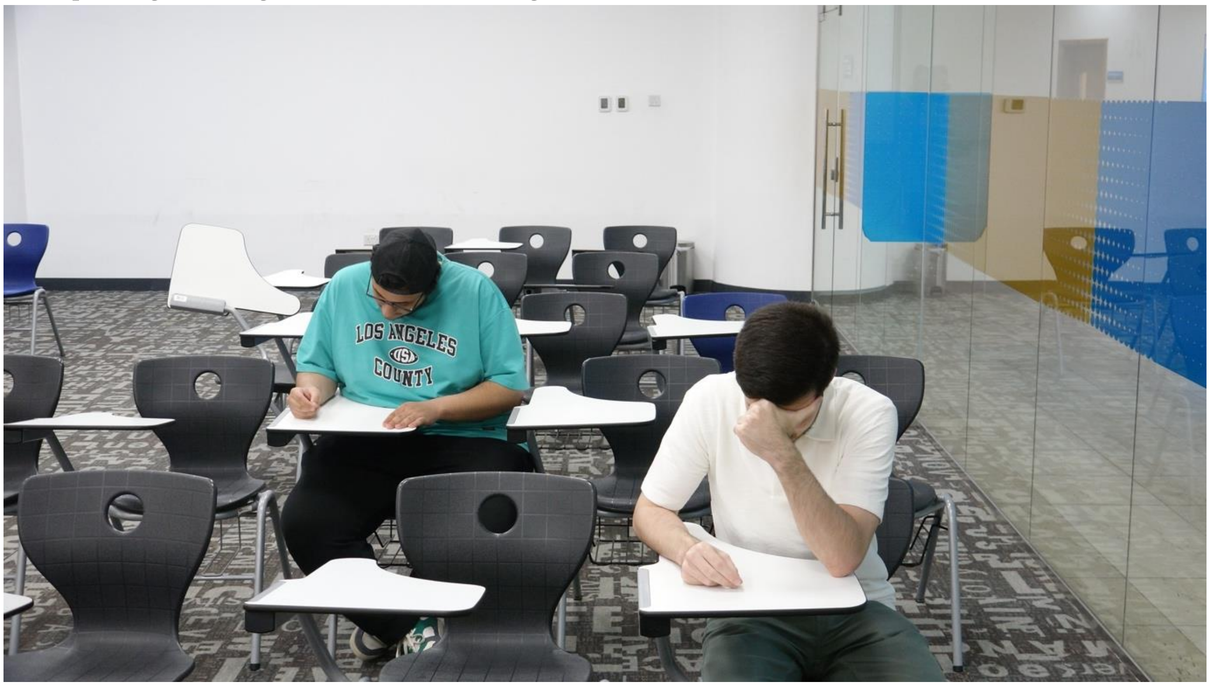


Figure 7.2 1 *Picture of Normal Posture*

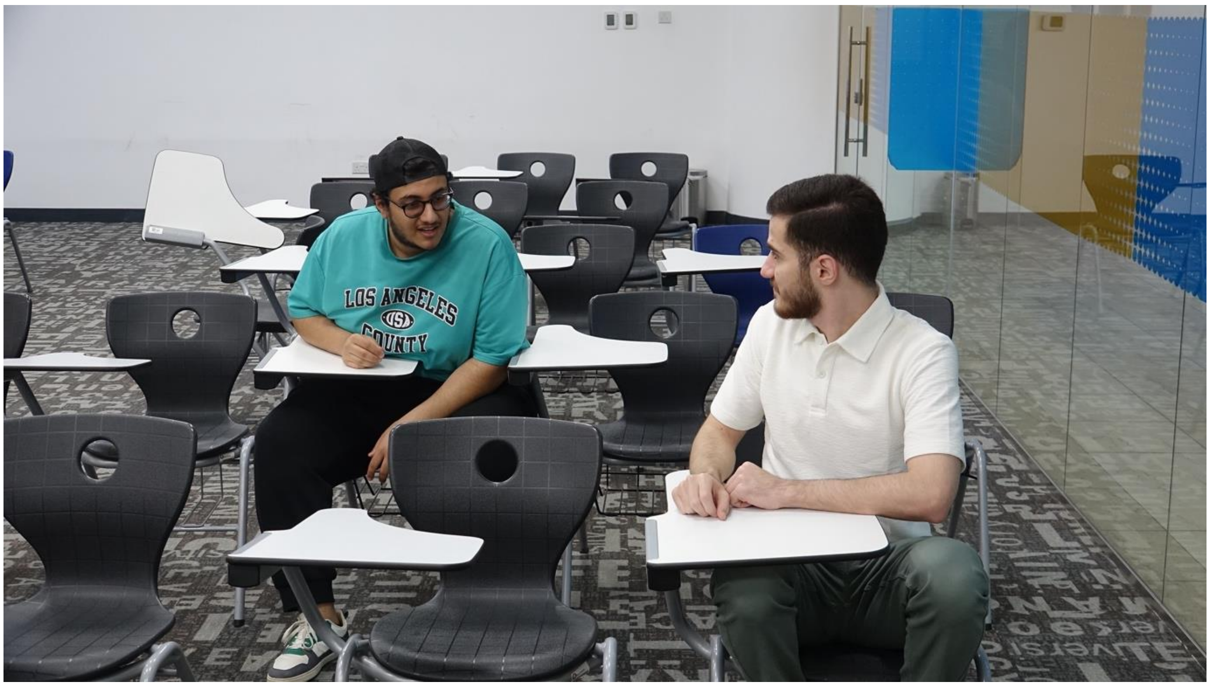


Figure 7.2.2 *Picture of Cheating Posture*

For the second part we took this data from Kaggle which is an open-source dataset website which you can download images or another dataset that you would like to use, this is an example of the dataset available online:

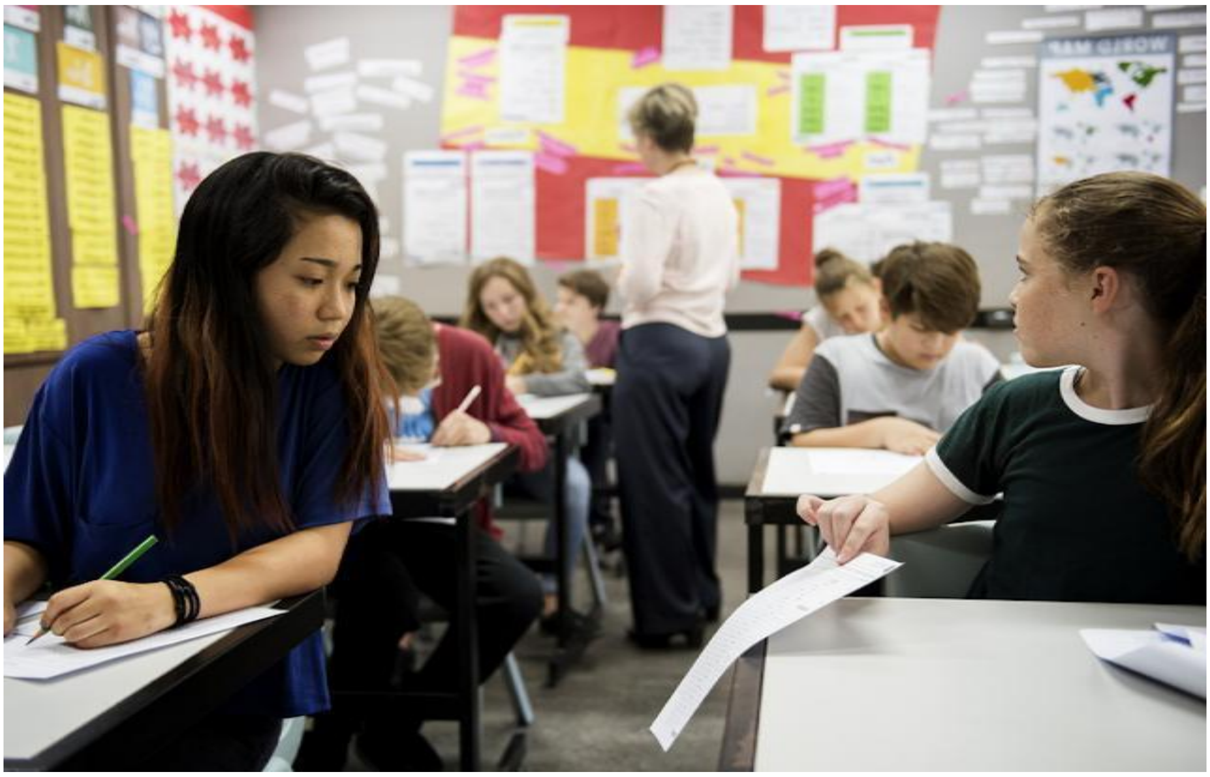


Figure 7.2.3 *Example of a Picture from Open Source*

7.3 Data Annotations

After gathering the data now we have to annotate it so the model can understand what data it is getting, to do this step we have used Roboflow, Roboflow is a free tool available online, this tools makes it so much easier to split the work between me and my colleagues, so in the figures below I will show the steps to how to annotate the data and make them ready to train the model.

7.3.1 Create a new project

First step is to login, and create a new workspace, in that workspace you should click on create a new project:

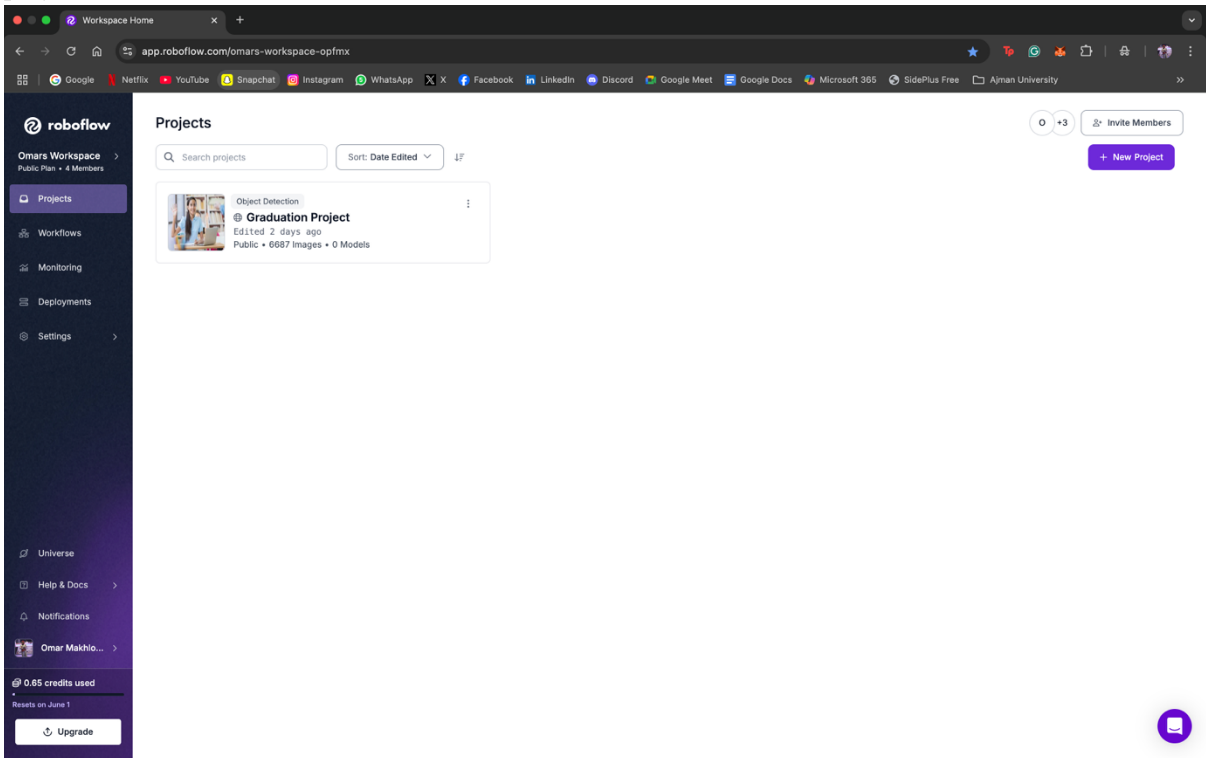
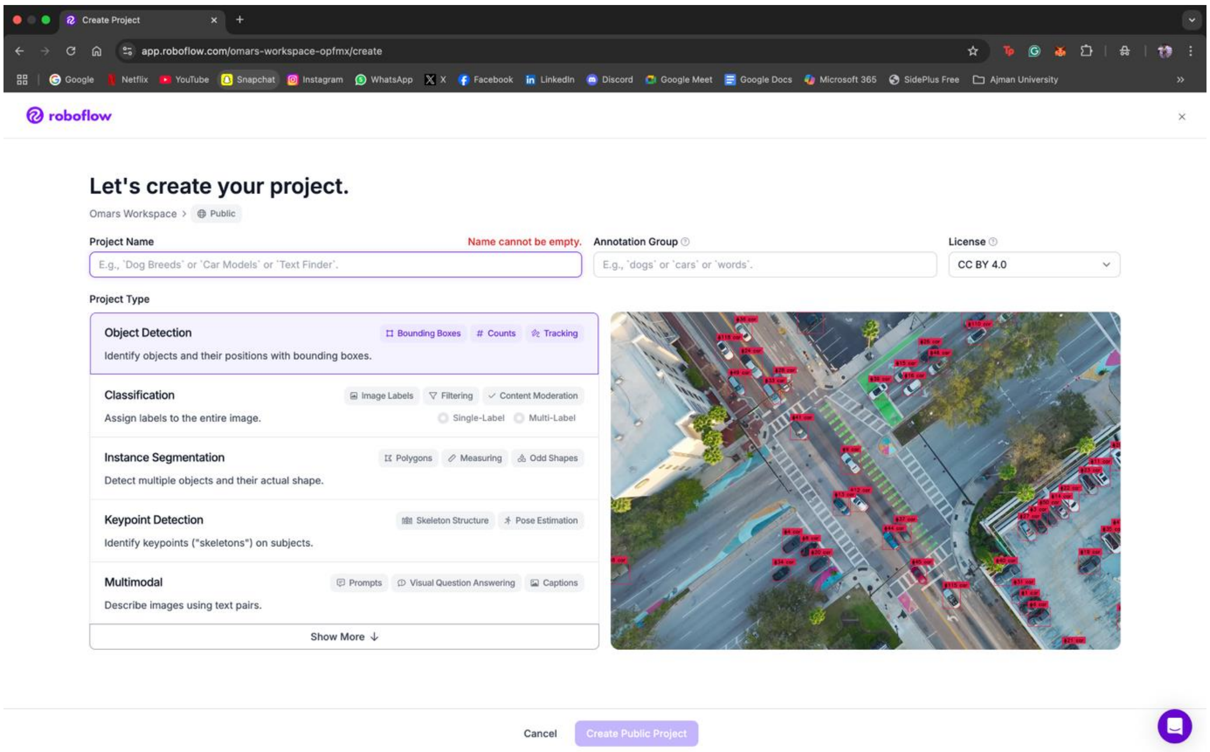


Figure 7.3.1.1 *Create New Project*

Then we should name the project, specify the Annotation group, and specify for what are we doing this project, for us we are using Object detection:


Create Project
app.roboflow.com/omars-workspace-opfmx/create
Google Netflix YouTube Snapchat Instagram WhatsApp X Facebook LinkedIn Discord Google Meet Google Docs Microsoft 365 SidePlus Free Ajman University
roboflow

Let's create your project.

Omars Workspace > Public

Project Name	E.g., 'Dog Breeds' or 'Car Models' or 'Text Finder'. (Name cannot be empty.)
Annotation Group	E.g., 'dogs' or 'cars' or 'words'.
License	CC BY 4.0

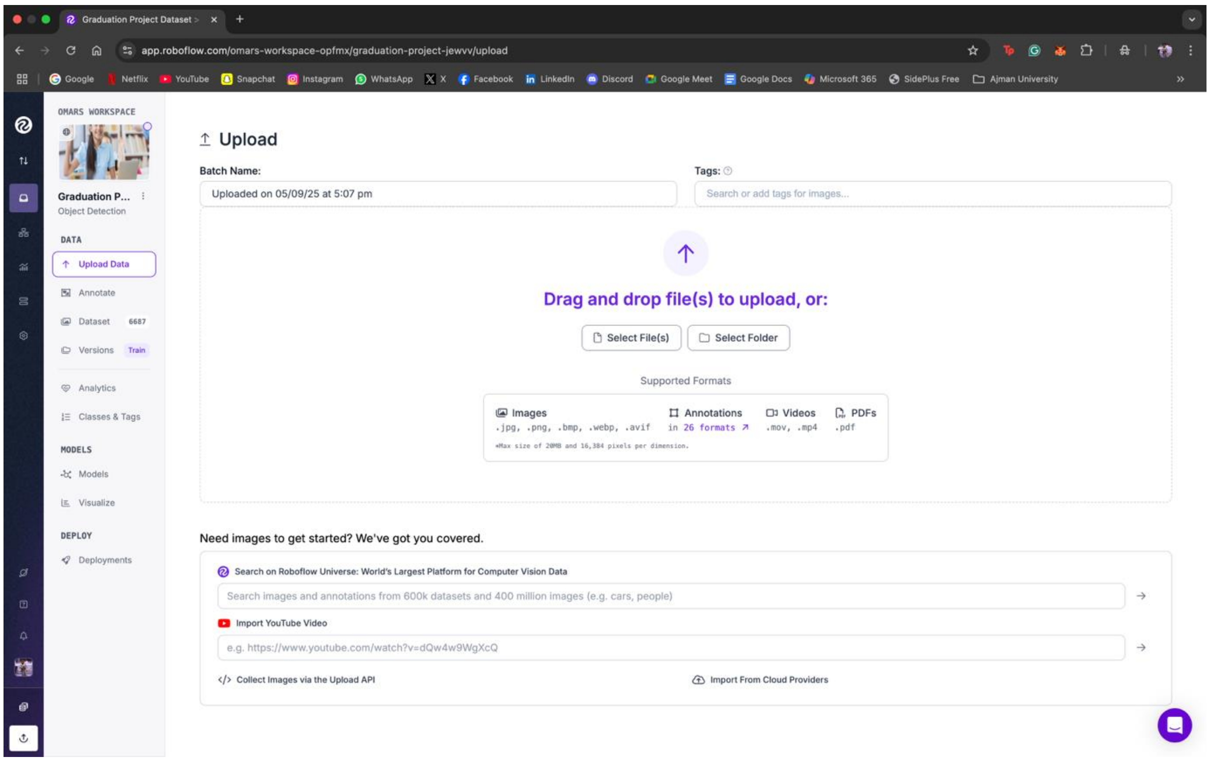
Project Type

- Object Detection — Identify objects and their positions with bounding boxes. (Bounding Boxes, Counts, Tracking)
- Classification — Assign labels to the entire image. (Image Labels, Filtering, Content Moderation; Single-Label, Multi-Label)
- Instance Segmentation — Detect multiple objects and their actual shape. (Polygons, Measuring, Odd Shapes)
- Keypoint Detection — Identify keypoints ("skeletons") on subjects. (Skeleton Structure, Pose Estimation)
- Multimodal — Describe images using text pairs. (Prompts, Visual Question Answering, Captions)

Figure 7.3.1.2 *Process of Creating Project*

After that we create the project, after creating the project we are going to upload the data into the project in this page.

7.3.2 Upload the images


Graduation Project Dataset >
app.roboflow.com/omars-workspace-opfmx/graduation-project-jewvv/upload
Google Netflix YouTube Snapchat Instagram WhatsApp X Facebook LinkedIn Discord Google Meet Google Docs Microsoft 365 SidePlus Free Ajman University
OMARS WORKSPACE
Graduation P...
Object Detection
DATA
Upload Data
Annotate
Dataset 6687
Versions Train
Analytics
Classes & Tags
MODELS
Models
Visualize
DEPLOY
Deployments

Upload

Batch Name: Uploaded on 05/09/25 at 5:07 pm

Tags: Search or add tags for images...

Drag and drop file(s) to upload, or:

Select File(s)
Select Folder

Supported Formats

Images	Annotations	Videos	PDFs
.jpg, .png, .bmp, .webp, .avif	in 26 formats	.mov, .mp4	.pdf

*Max size of 20MB and 16,384 pixels per dimension.

Need images to get started? We've got you covered.

Search on Roboflow Universe: World's Largest Platform for Computer Vision Data

Search images and annotations from 600k datasets and 400 million images (e.g. cars, people)

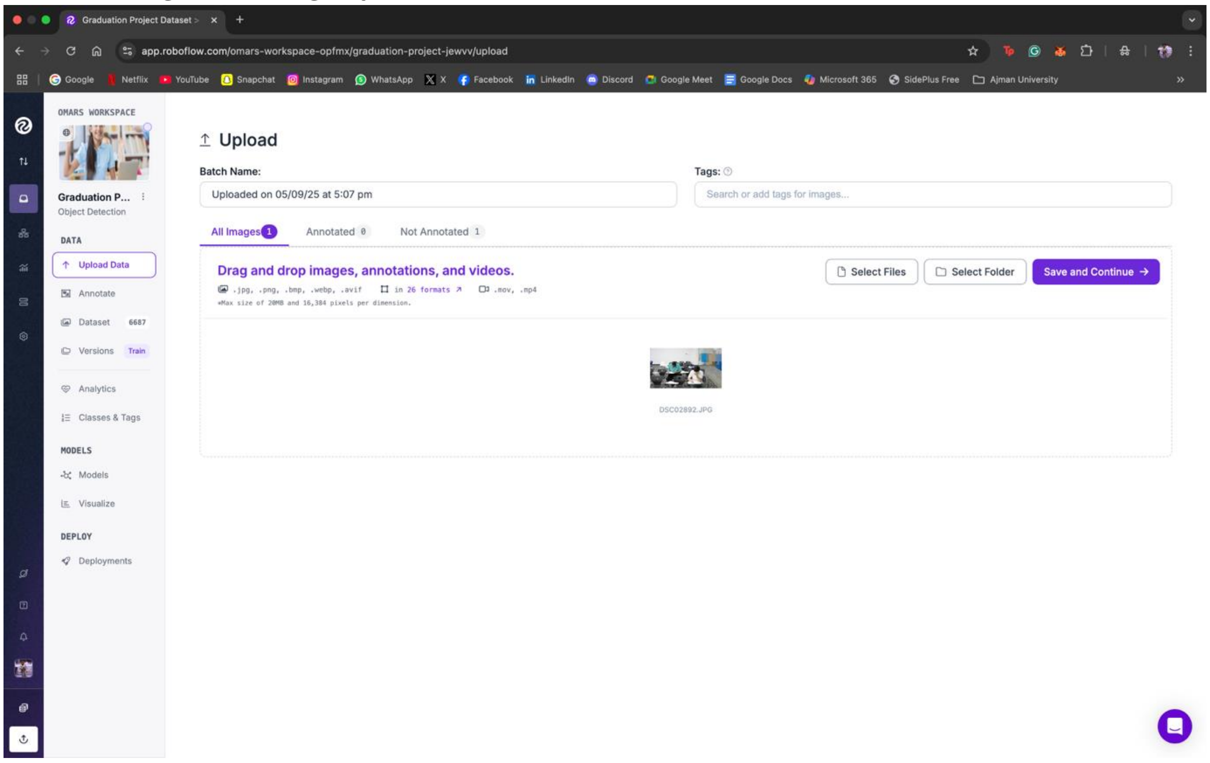
Import YouTube Video

e.g. https://www.youtube.com/watch?v=dQw4w9WgXcQ

Collect Images via the Upload API
Import From Cloud Providers

Figure 7.3.2.1 *Upload the Images*

After inserting all the images, you click on save and continue:



Batch Name: Uploaded on 05/09/25 at 5:07 pm

Tags: Search or add tags for images...

All Images 1 | Annotated 0 | Not Annotated 1

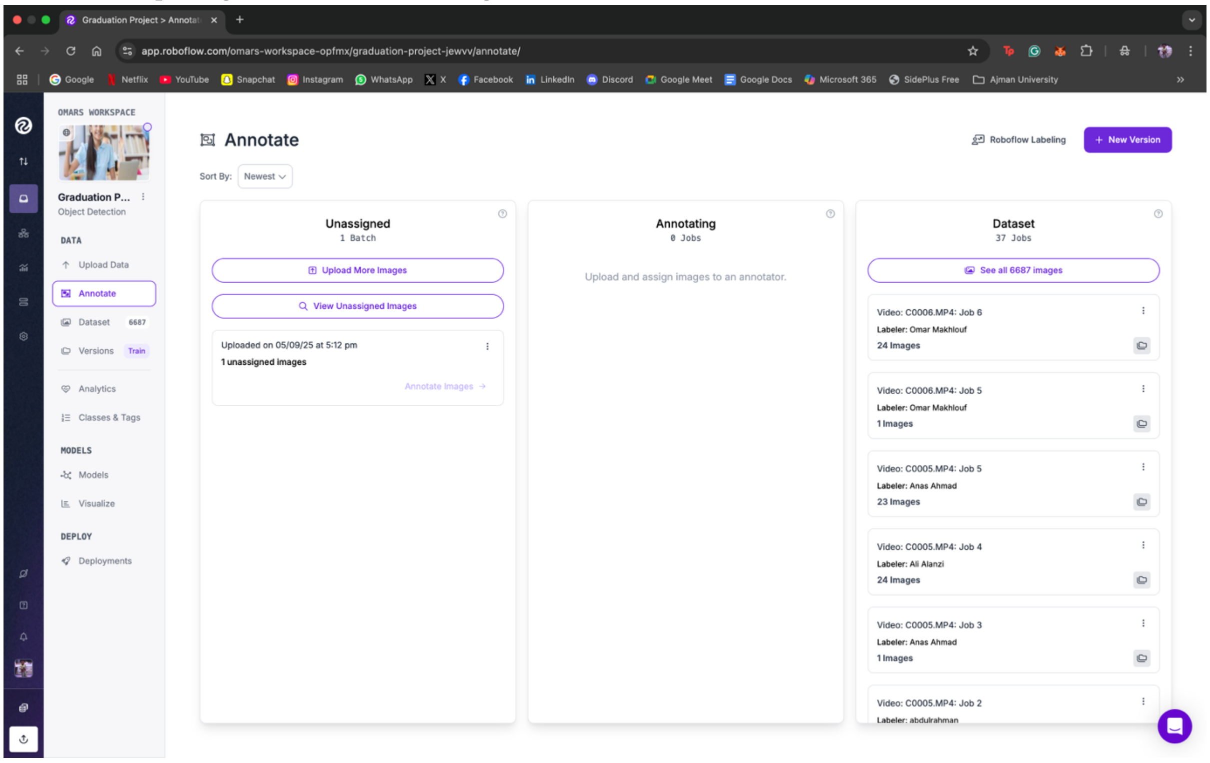
Drag and drop images, annotations, and videos.

.jpg, .png, .bmp, .webp, .avif in 26 formats .mov, .mp4

*Max size of 20MB and 16,384 pixels per dimension.

Figure 7.3.2.2 *Images are Uploaded*

After this step we go to annotate the images.



Unassigned — 1 Batch

Uploaded on 05/09/25 at 5:12 pm — 1 unassigned images

Annotating — 0 Jobs

Upload and assign images to an annotator.

Dataset — 37 Jobs

Video: C0006.MP4: Job 6 — Labeler: Omar Makhlouf — 24 Images

Video: C0006.MP4: Job 5 — Labeler: Omar Makhlouf — 1 Images

Video: C0005.MP4: Job 5 — Labeler: Anas Ahmad — 23 Images

Video: C0005.MP4: Job 4 — Labeler: Ali Alanzi — 24 Images

Video: C0005.MP4: Job 3 — Labeler: Anas Ahmad — 1 Images

Video: C0005.MP4: Job 2 — Labeler: abdulrahman

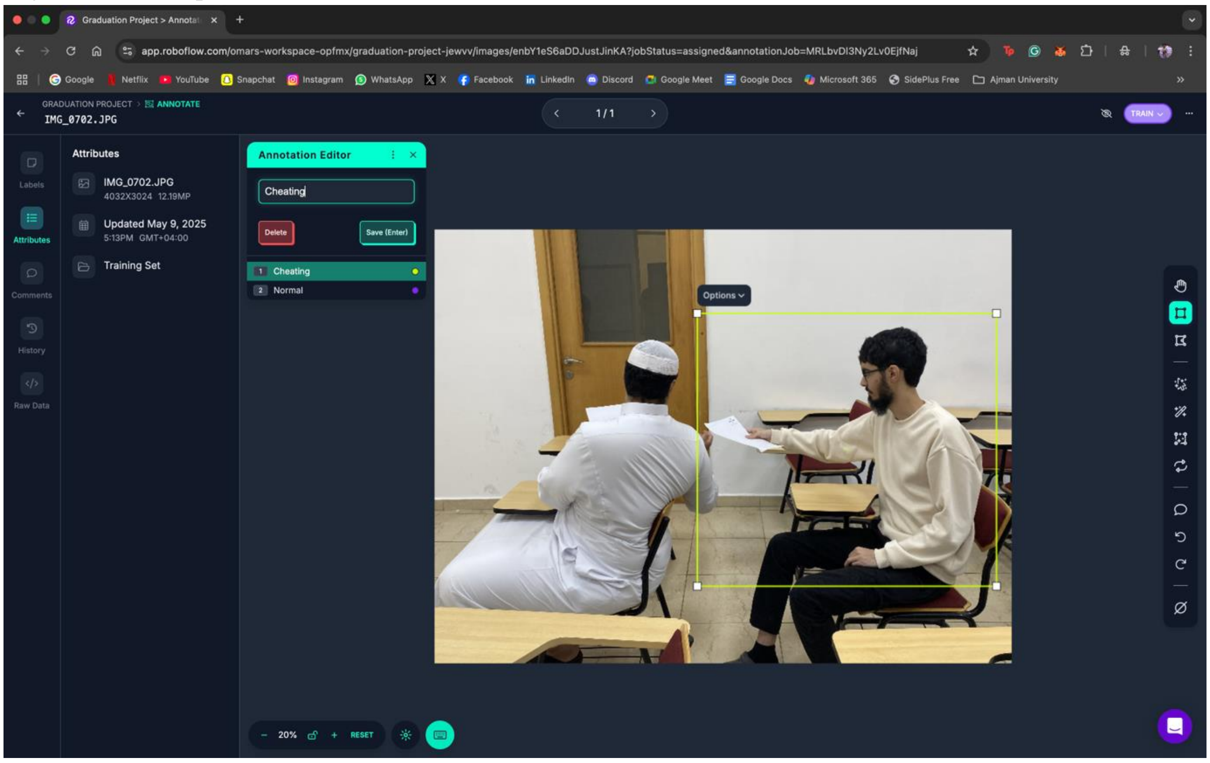
Figure 7.3.2.3 *Annotate the Images*

7.3.3 Annotate the images

Now we are going to annotate the images for us we have 2 labels:

1. Normal
2. Cheating

We are going to use these 2 labels to distinguish between cheating posture and normal postures of students, now how do we do it, first we use the Bounding box tool to draw a bounding box around the object, for example like this:



Graduation Project > Annotate
app.roboflow.com/omars-workspace-opfmx/graduation-project-jewvv/images/enbY1eS6aDDJustJinKA?jobStatus=assigned&annotationJob=MRLbvDI3Ny2Lv0EjfNaj
GRADUATION PROJECT > ANNOTATE
IMG_0702.JPG
1 / 1
TRAIN
Labels
Attributes
Comments
History
Raw Data
Annotation Editor
Delete
Save (Enter)
Options
20%
RESET

Attributes
IMG_0702.JPG 4032X3024 12.19MP
Updated May 9, 2025 5:13PM GMT+04:00
Training Set

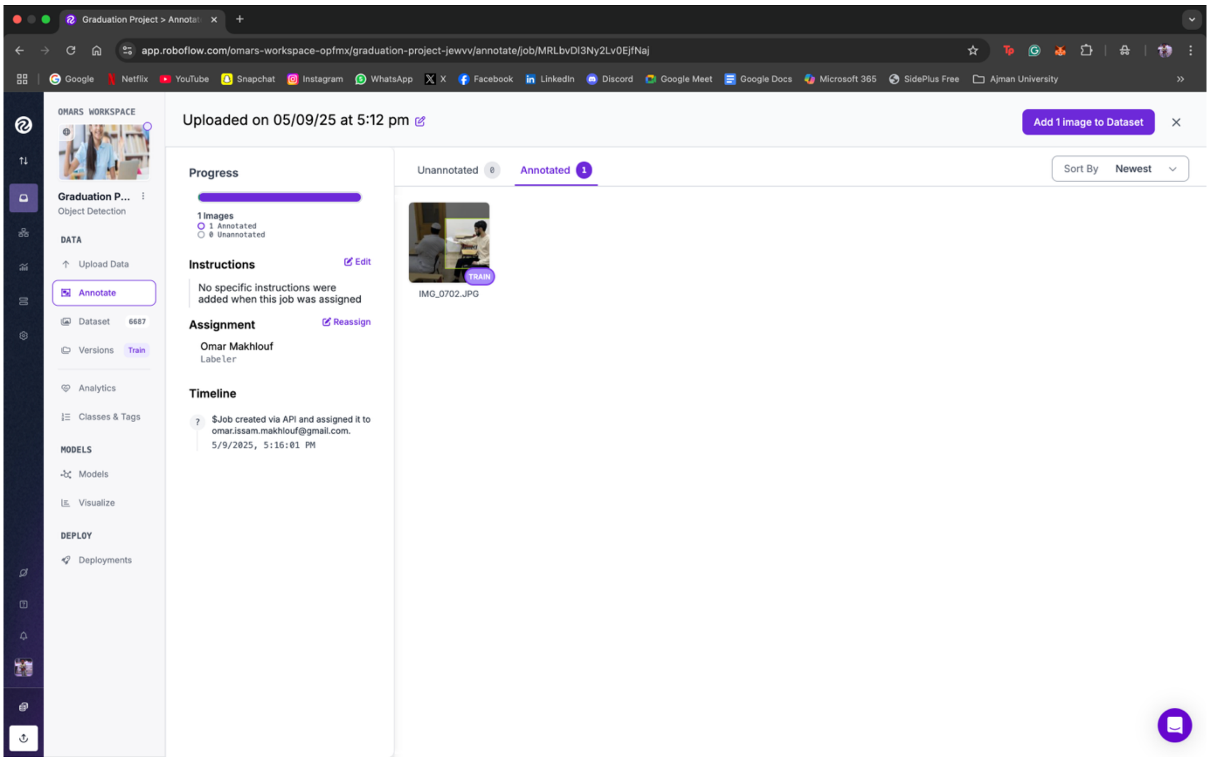
Cheating

1 Cheating
2 Normal

Figure 7.3.3.1 Drawing Bounding Box

In this image for example we drew a bounding box around my colleague and specified the label into cheating because he is cheating after that we save it, after saving it should look something like this:

After annotating all the images, we are going to add the images to the datasets:



No specific instructions were added when this job was assigned

$Job created via API and assigned it to omar.issam.makhlouf@gmail.com.
5/9/2025, 5:16:01 PM

***Figure 7.3.3.2** Add Annotated Image to Dataset*

Now to the next step we check on all the images we have annotated to see if there are any mistakes that happened:

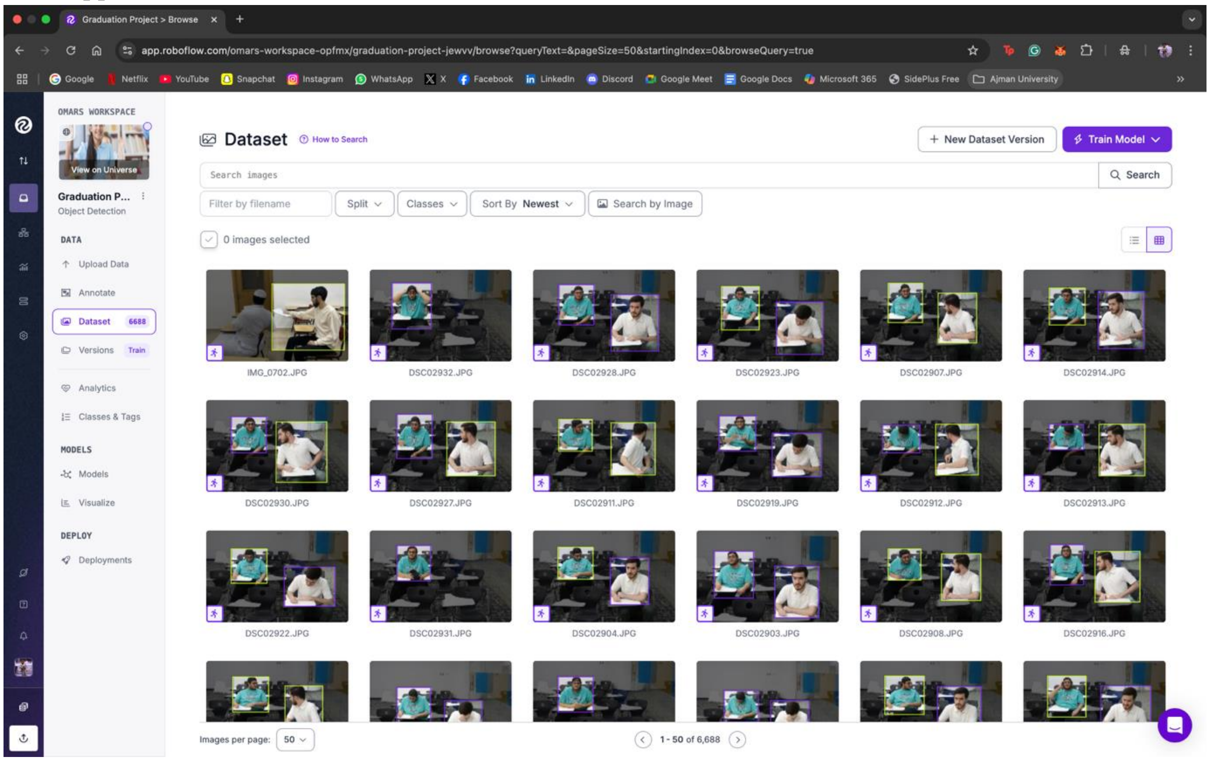
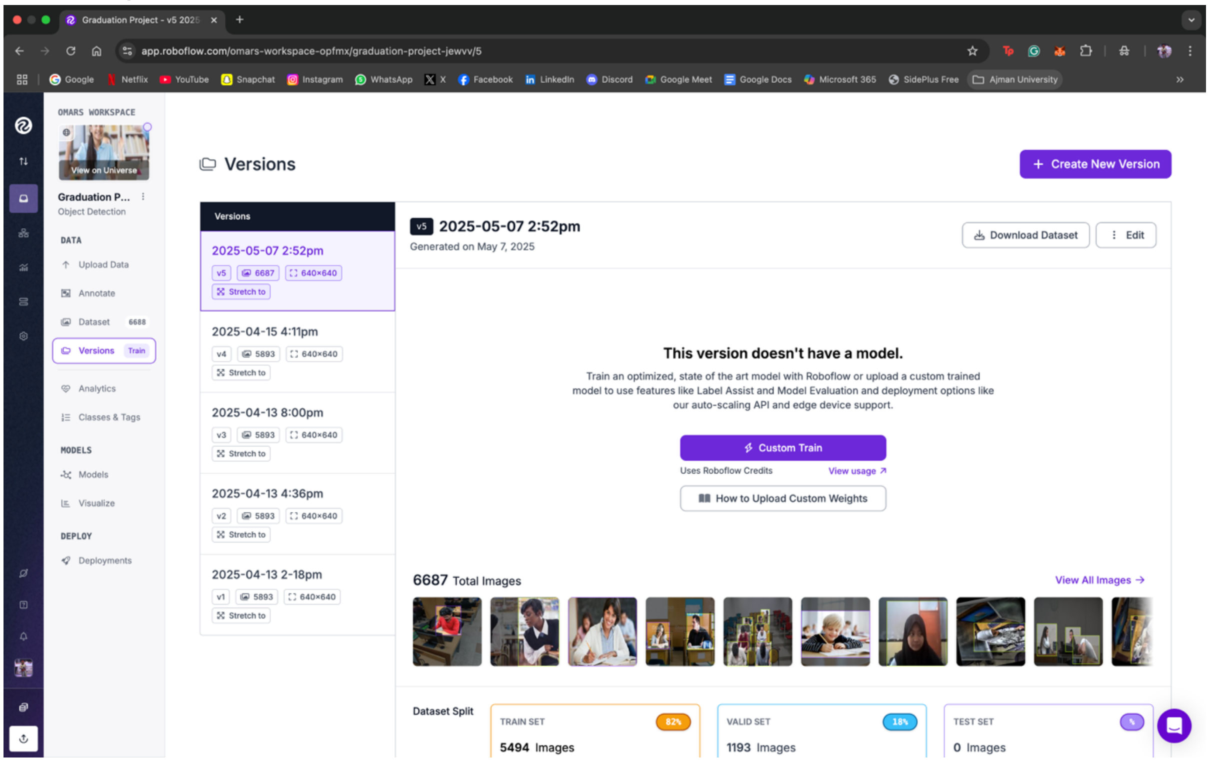


***Figure 7.3.3.3** Here is How the Dataset Looks like*

After that we are going to create a version of this dataset and specify how much training, validation, and testing do we want.

7.3.4 Create a version

In the image below we click on create version to create a version of the dataset:



v5 2025-05-07 2:52pm

Generated on May 7, 2025

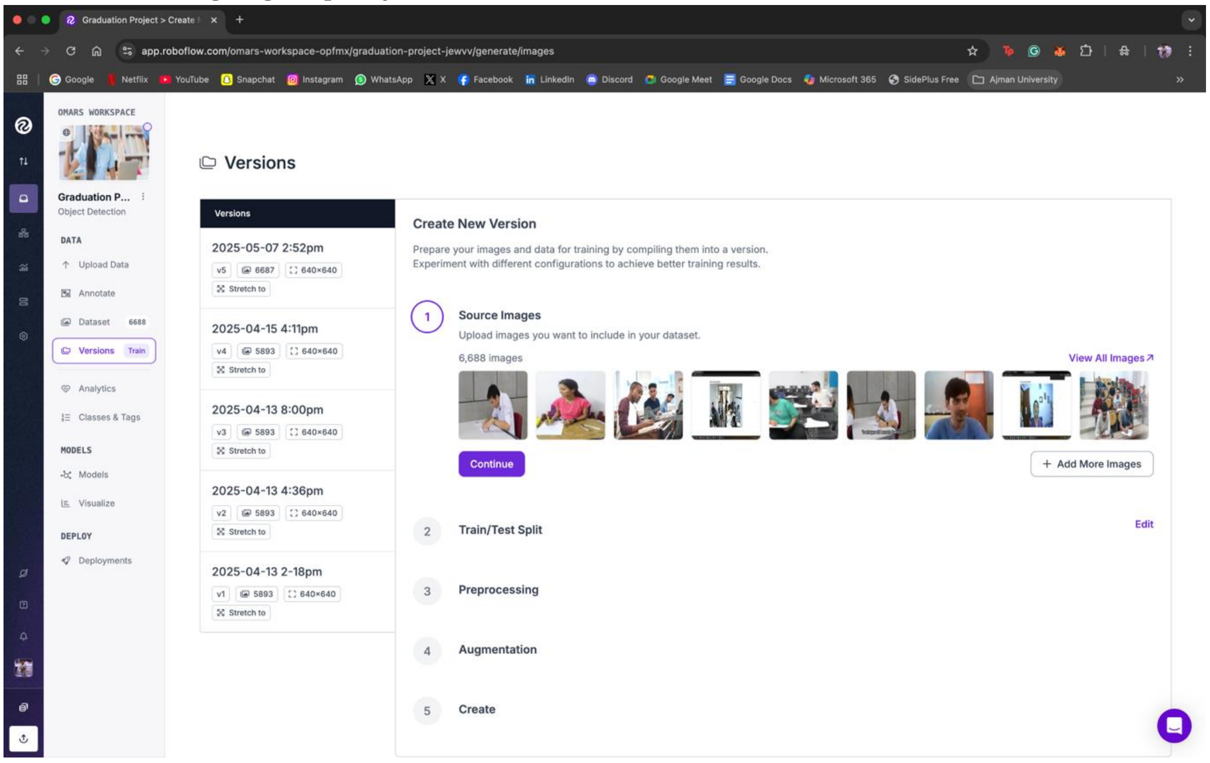
This version doesn't have a model.

Train an optimized, state of the art model with Roboflow or upload a custom trained model to use features like Label Assist and Model Evaluation and deployment options like our auto-scaling API and edge device support.

6687 Total Images

Figure 7.3.4.1 *Create a Version*

After that we are going to specify the dataset, we want to use which is the whole dataset:



Create New Version

Prepare your images and data for training by compiling them into a version. Experiment with different configurations to achieve better training results.

1 Source Images

Upload images you want to include in your dataset.

6,688 images

2 Train/Test Split

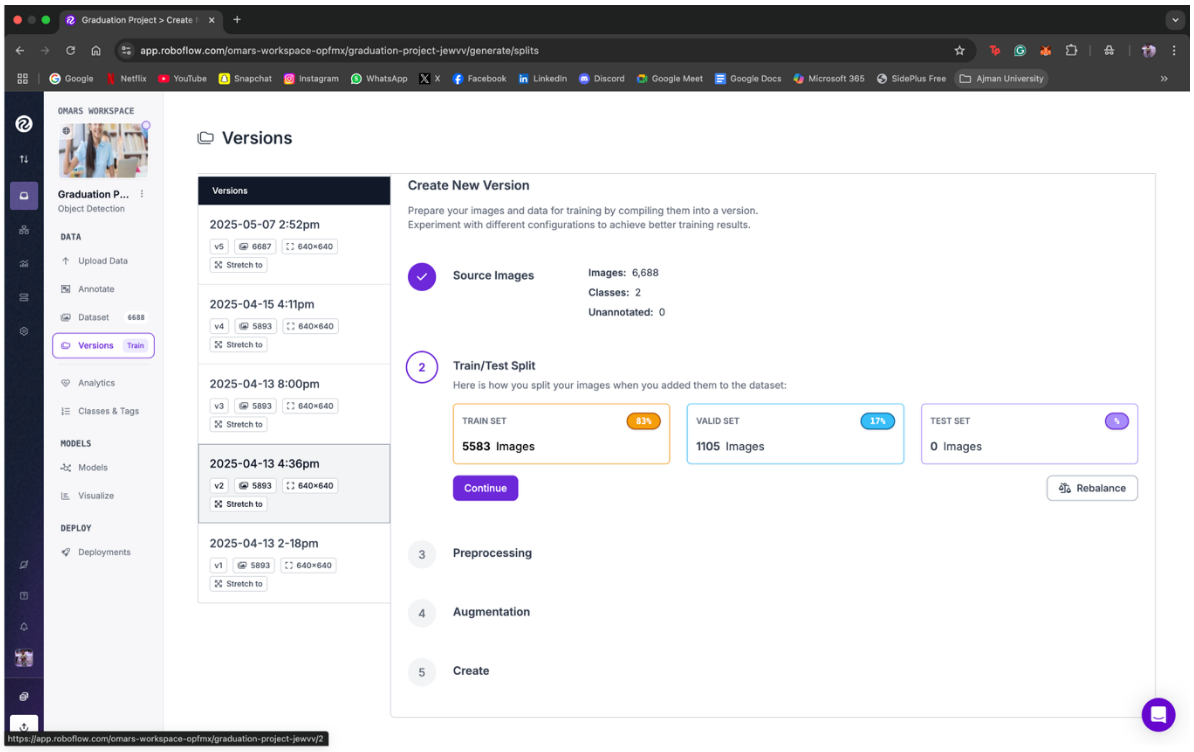
3 Preprocessing

4 Augmentation

5 Create

Figure 7.3.4.2 *Select Dataset*

Then we should choose the split as I mentioned before, how much I need for trainings, validation, and testing, for us we went with 82% training, 18% validation, and 0% testing because we will test this model on live cameras.


Versions

Create New Version

Prepare your images and data for training by compiling them into a version.
Experiment with different configurations to achieve better training results.

Source Images — Images: 6,688 Classes: 2 Unannotated: 0

2 Train/Test Split

Here is how you split your images when you added them to the dataset:

TRAIN SET 83%	VALID SET 17%	TEST SET %
5583 Images	1105 Images	0 Images

3 Preprocessing

4 Augmentation

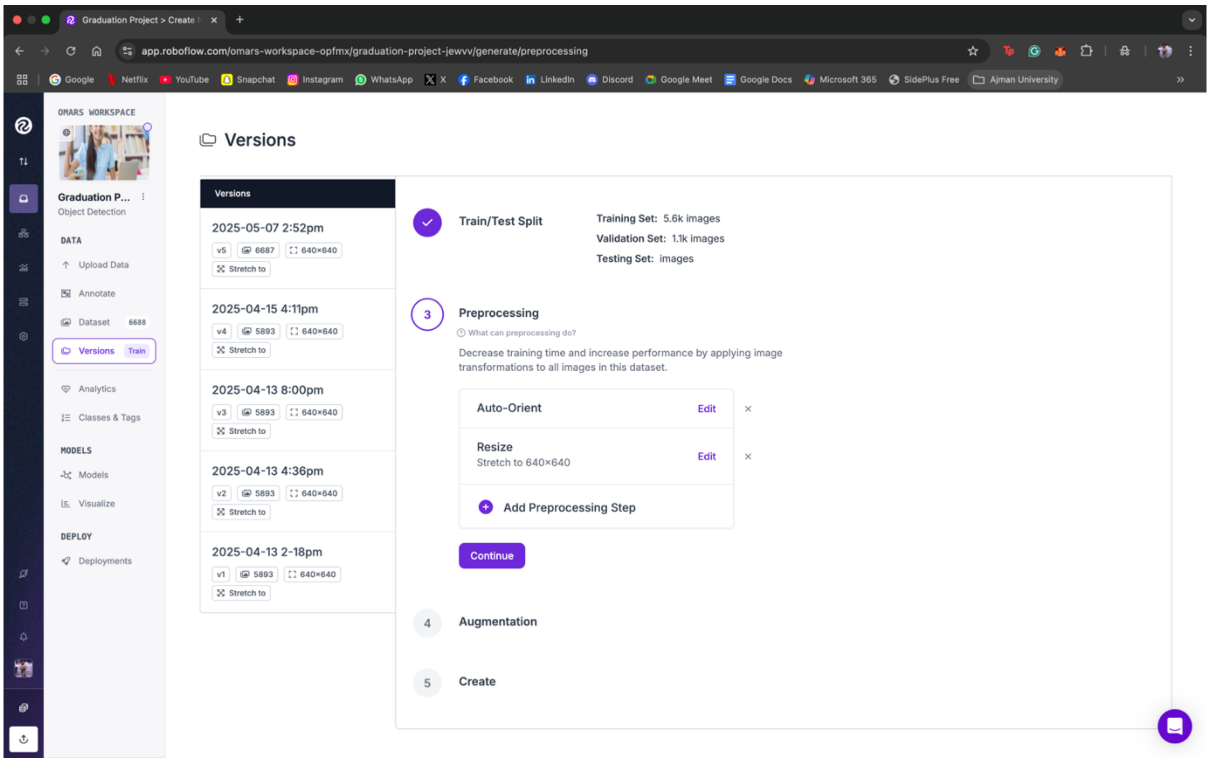
5 Create

Versions			
2025-05-07 2:52pm	v5	6687	640×640 Stretch to
2025-04-15 4:11pm	v4	5893	640×640 Stretch to
2025-04-13 8:00pm	v3	5893	640×640 Stretch to
2025-04-13 4:36pm	v2	5893	640×640 Stretch to
2025-04-13 2-18pm	v1	5893	640×640 Stretch to

Figure 7.3.4.3 *Train/Test Split*

After that we are going to do some preprocessing, we will choose 2 things:

1. Auto-Orient, which orients the images to be all vertical.
2. Resize, this will resize the images to 640x640 (which is the best for Computer vision AI models.



Versions
2025-05-07 2:52pm · v5 · 6687 · 640×640 · Stretch to
2025-04-15 4:11pm · v4 · 5893 · 640×640 · Stretch to
2025-04-13 8:00pm · v3 · 5893 · 640×640 · Stretch to
2025-04-13 4:36pm · v2 · 5893 · 640×640 · Stretch to
2025-04-13 2-18pm · v1 · 5893 · 640×640 · Stretch to

Train/Test Split
Training Set: 5.6k images
Validation Set: 1.1k images
Testing Set: images

3 Preprocessing
What can preprocessing do?
Decrease training time and increase performance by applying image transformations to all images in this dataset.

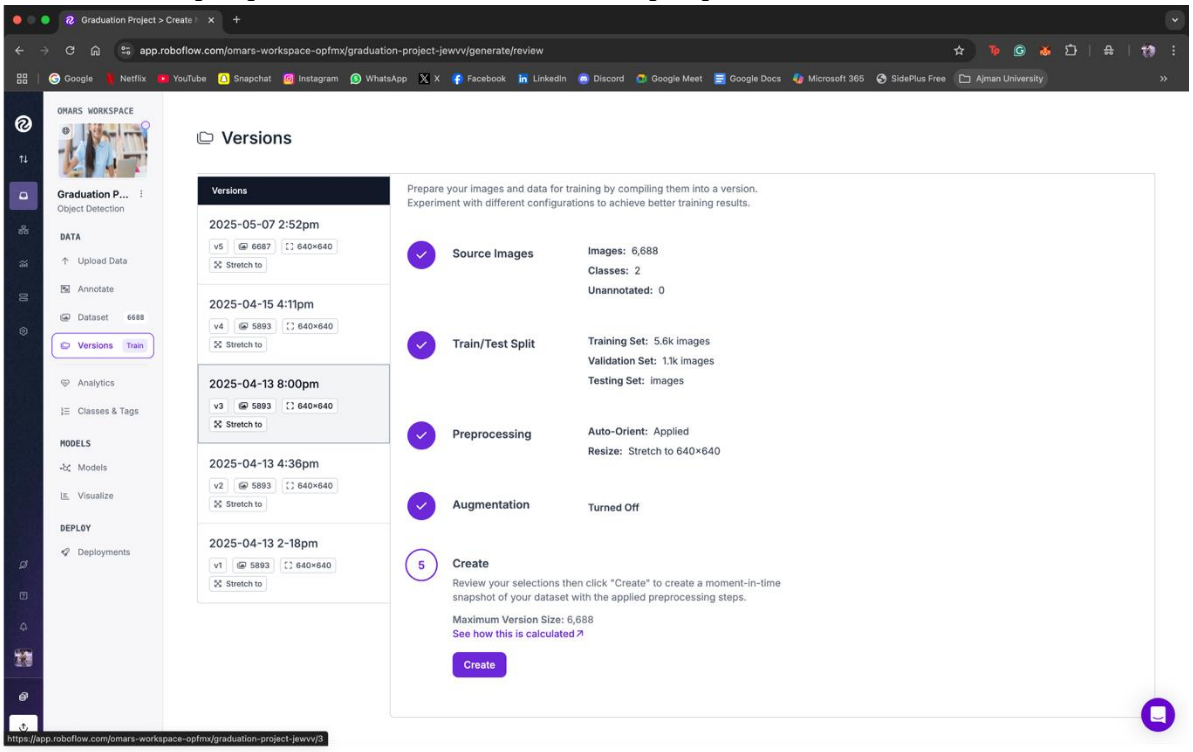
Auto-Orient — Edit
Resize — Stretch to 640×640 — Edit
Add Preprocessing Step
Continue

4 Augmentation

5 Create

Figure 7.3.4.4 Preprocessing

After that we are going to create the version that we are going to use to train our model:


Graduation Project > Create
app.roboflow.com/omars-workspace-opfmx/graduation-project-jewvv/generate/review
OMARS WORKSPACE
Graduation P...
Object Detection
DATA
Upload Data
Annotate
Dataset 6688
Versions Train
Analytics
Classes & Tags
MODELS
Models
Visualize
DEPLOY
Deployments
Versions
https://app.roboflow.com/omars-workspace-opfmx/graduation-project-jewvv/3

Versions
2025-05-07 2:52pm · v5 · 6687 · 640×640 · Stretch to
2025-04-15 4:11pm · v4 · 5893 · 640×640 · Stretch to
2025-04-13 8:00pm · v3 · 5893 · 640×640 · Stretch to
2025-04-13 4:36pm · v2 · 5893 · 640×640 · Stretch to
2025-04-13 2-18pm · v1 · 5893 · 640×640 · Stretch to

Prepare your images and data for training by compiling them into a version. Experiment with different configurations to achieve better training results.

Source Images
Images: 6,688
Classes: 2
Unannotated: 0

Train/Test Split
Training Set: 5.6k images
Validation Set: 1.1k images
Testing Set: images

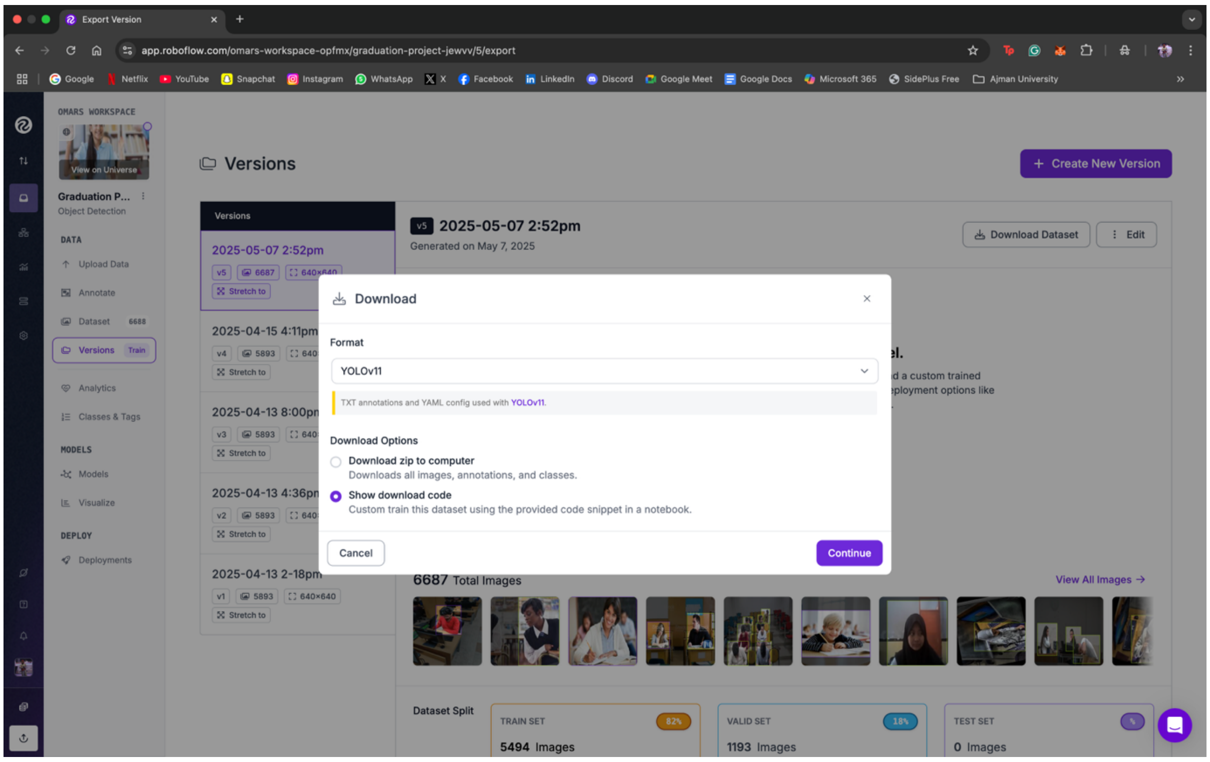
Preprocessing
Auto-Orient: Applied
Resize: Stretch to 640×640

Augmentation
Turned Off

5 Create
Review your selections then click "Create" to create a moment-in-time snapshot of your dataset with the applied preprocessing steps.
Maximum Version Size: 6,688
See how this is calculated
Create

Figure 7.3.4.5 Create Version

After creating the version, we can download the labelled dataset to be used to train the model, when downloading the dataset we have to specify the format, for us the format is YOLOv11 and then download the dataset as a zip file to the computer.



Download

Format

YOLOv11

TXT annotations and YAML config used with YOLOv11.

Download Options

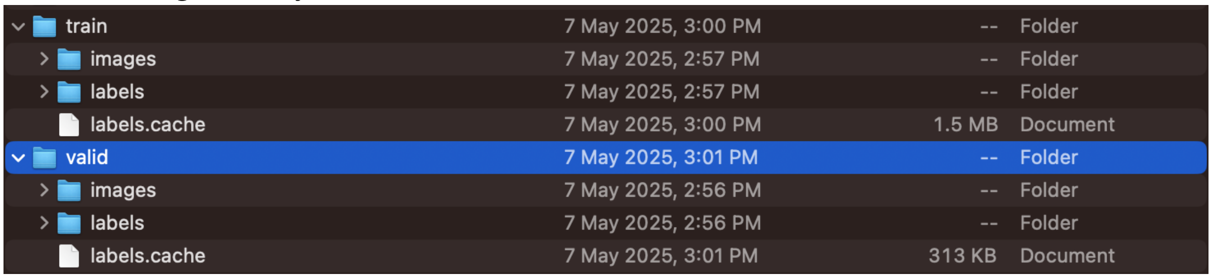
[ ] Download zip to computer
Downloads all images, annotations, and classes.

[✓ Show download code]
Custom train this dataset using the provided code snippet in a notebook.

Cancel Continue

Figure 7.3.4.6 *Specify Exporting Format*

After extracting the file, you will notice that the format of the folders look like this:



train	7 May 2025, 3:00 PM	--	Folder
images	7 May 2025, 2:57 PM	--	Folder
labels	7 May 2025, 2:57 PM	--	Folder
labels.cache	7 May 2025, 3:00 PM	1.5 MB	Document
valid	7 May 2025, 3:01 PM	--	Folder
images	7 May 2025, 2:56 PM	--	Folder
labels	7 May 2025, 2:56 PM	--	Folder
labels.cache	7 May 2025, 3:01 PM	313 KB	Document

Figure 7.3.4.7 *Folders format*

And if we go look at the images, there will be all the images in jpg format but if we look at the labels we can see that all the files are in txt format, but they have the same name, so when we open the files (one jpg and one txt with the same name) usually looks like this:

This is the image:

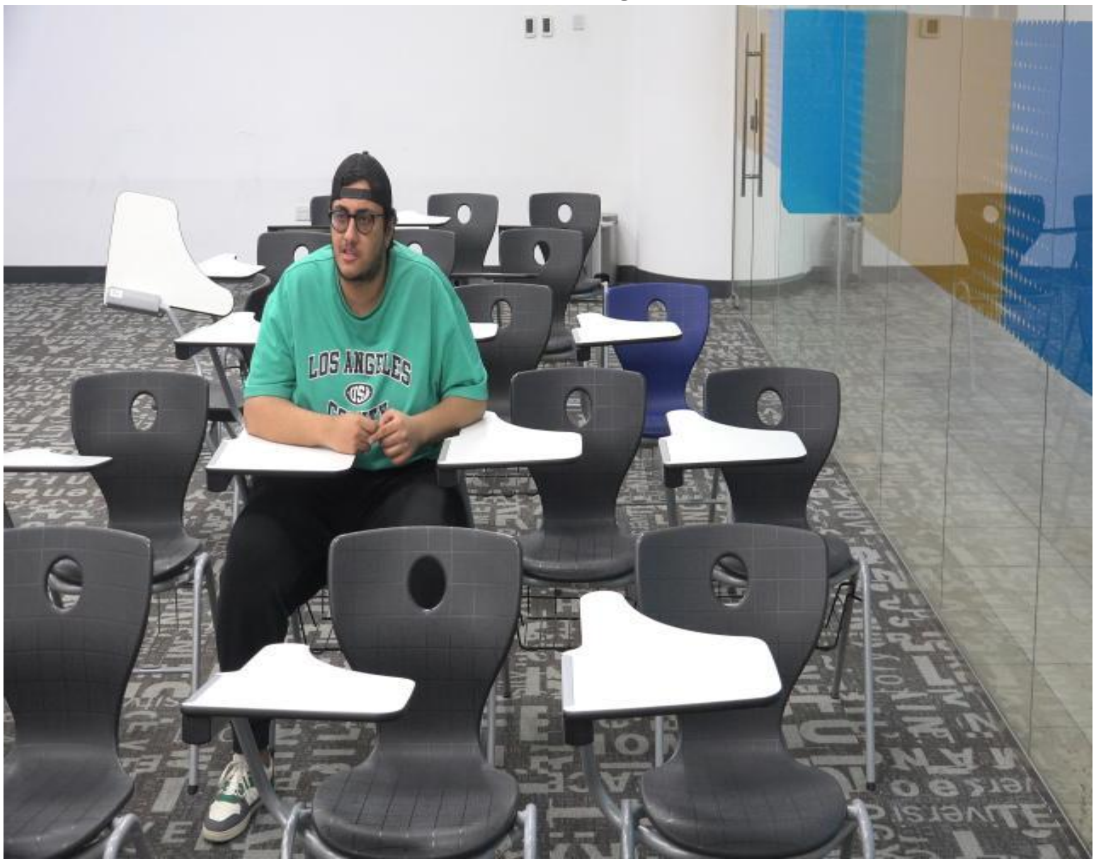
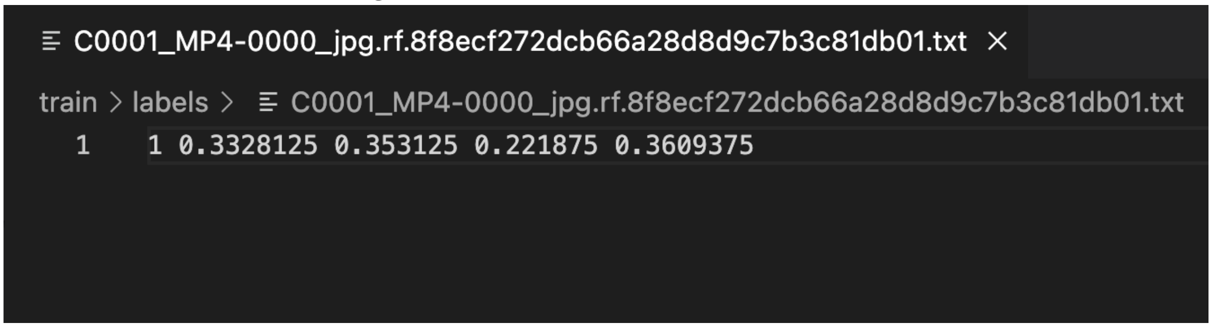


Figure 7.3.4.8 *Image Example*

And this is the label of this image:



```
C0001_MP4-0000_jpg.rf.8f8ecf272dcb66a28d8d9c7b3c81db01.txt
train > labels > C0001_MP4-0000_jpg.rf.8f8ecf272dcb66a28d8d9c7b3c81db01.txt
1 0.3328125 0.353125 0.221875 0.3609375
```

Figure 7.3.4.9 *Text File Format Example*

All the time the name of the image file and the label of that file are the same, what is different is the file type so we can see that in the first image it is .jpg but in the second it is .txt, now let's talk about the txt file and what does the numbers mean.
The format of these numbers read like this:
<class_id> <x_center> <y_center> <width> <height>
1: class_id – This object belongs to class 1 (the first class in the dataset which means normal).

0.3328125: x_center – X coordinate of the center of the bounding box, normalized.

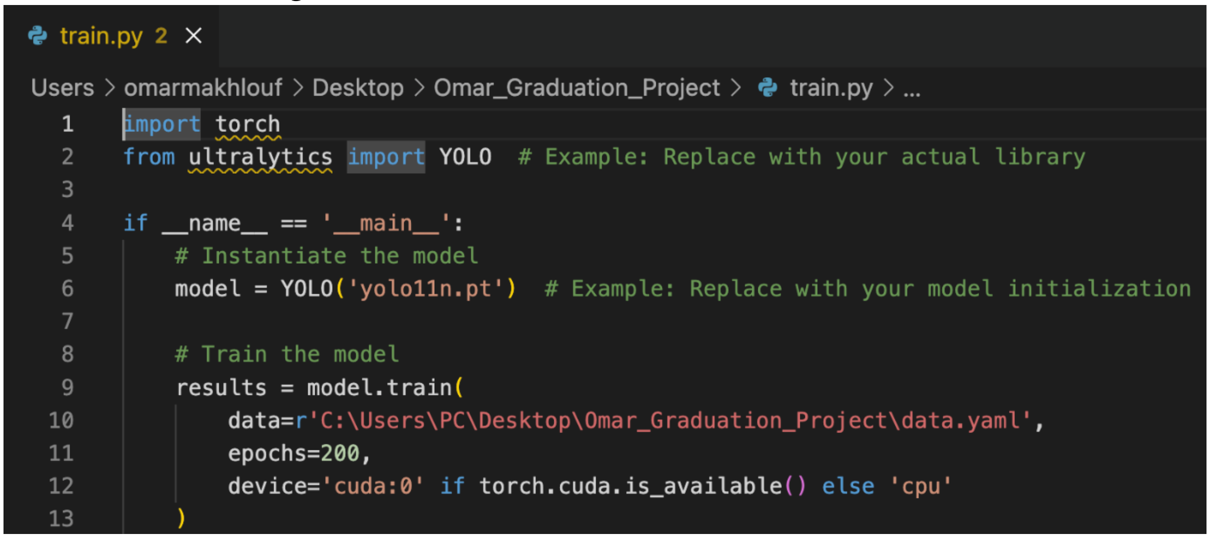
0.353125: y_center – Y coordinate of the center of the bounding box, normalized.

0.221875: width – Width of the bounding box, normalized to image width.

0.3609375: height – Height of the bounding box, normalized to image height.

7.4 Train the Model

For training the model I have used VS Code to write and execute the code, here is a sample of the code used for the training:

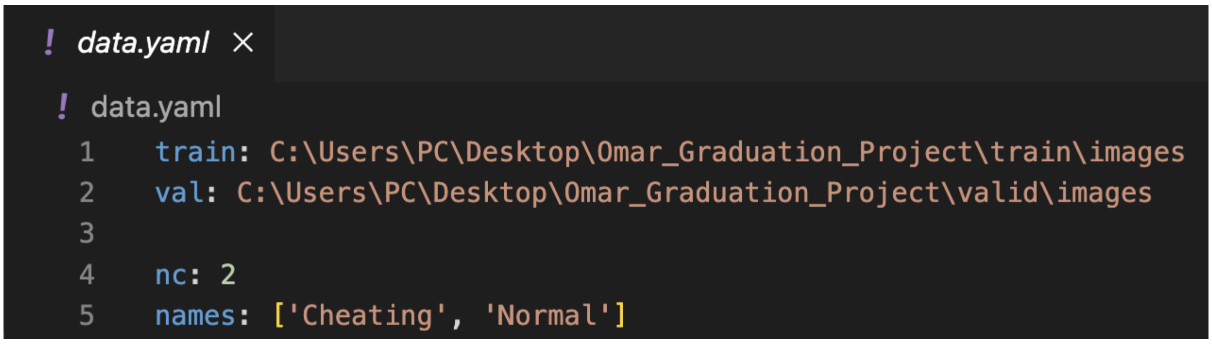

train.py 2 ×
Users > omarmakhlouf > Desktop > Omar_Graduation_Project > train.py > ...

```python
import torch
from ultralytics import YOLO  # Example: Replace with your actual library

if __name__ == '__main__':
    # Instantiate the model
    model = YOLO('yolo11n.pt')  # Example: Replace with your model initialization

    # Train the model
    results = model.train(
        data=r'C:\Users\PC\Desktop\Omar_Graduation_Project\data.yaml',
        epochs=200,
        device='cuda:0' if torch.cuda.is_available() else 'cpu'
    )
```

Figure 7.4.1 *Training the Model Code*

First we import torch and yolo, after that we are going to use a pretrained model to help with the accuracy of the model, for us we have used yolo11n (YOLOv11 nano), after that we specify the path of the data.yaml file, specify how many epochs we want to train and lastly we want to use the GPU to train the model, if it is not available then we want to use the CPU.

This is a screenshot of the data.yaml file:



```yaml
train: C:\Users\PC\Desktop\Omar_Graduation_Project\train\images
val: C:\Users\PC\Desktop\Omar_Graduation_Project\valid\images

nc: 2
names: ['Cheating', 'Normal']
```

***Figure 7.4.2** data.yaml file code*

In the data.yaml file we should specify the training path and the validation path.
After doing this then we can train the model by running the python file.

7.5 Results

After the training is done, we can view the results:

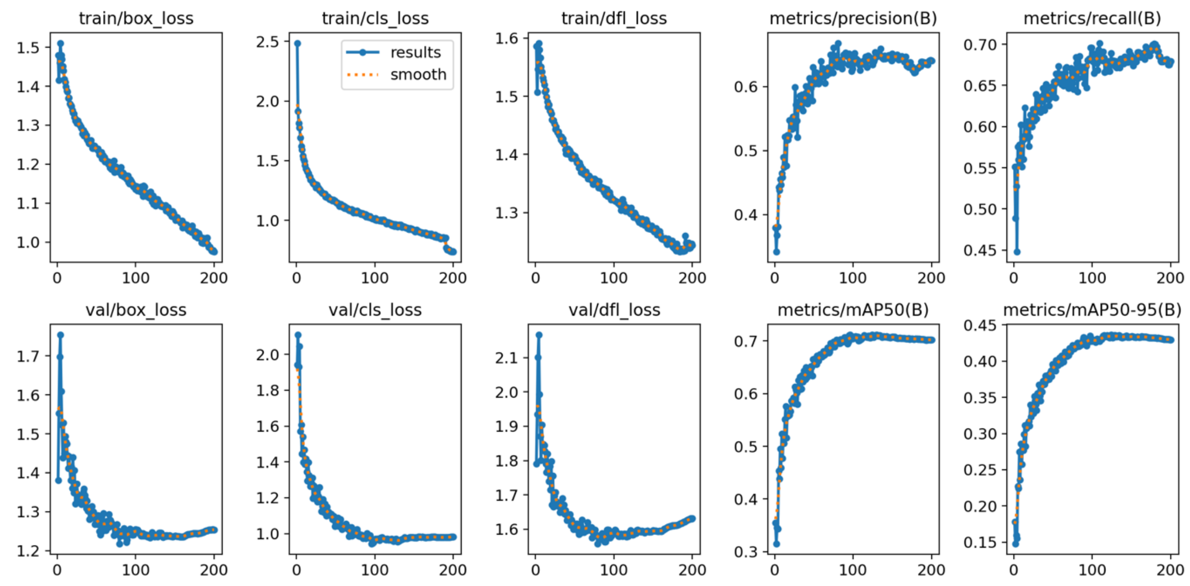


***Figure 7.5.1** Training Results*

These are the results of our training, we can see the accuracy kept increasing, and the loss kept decreasing, which means that the model is improving, and that the training is going to the right direction.

7.6 Testing the model

Last step is to test the model, we used some images to test the model, here are some testing results of the images:

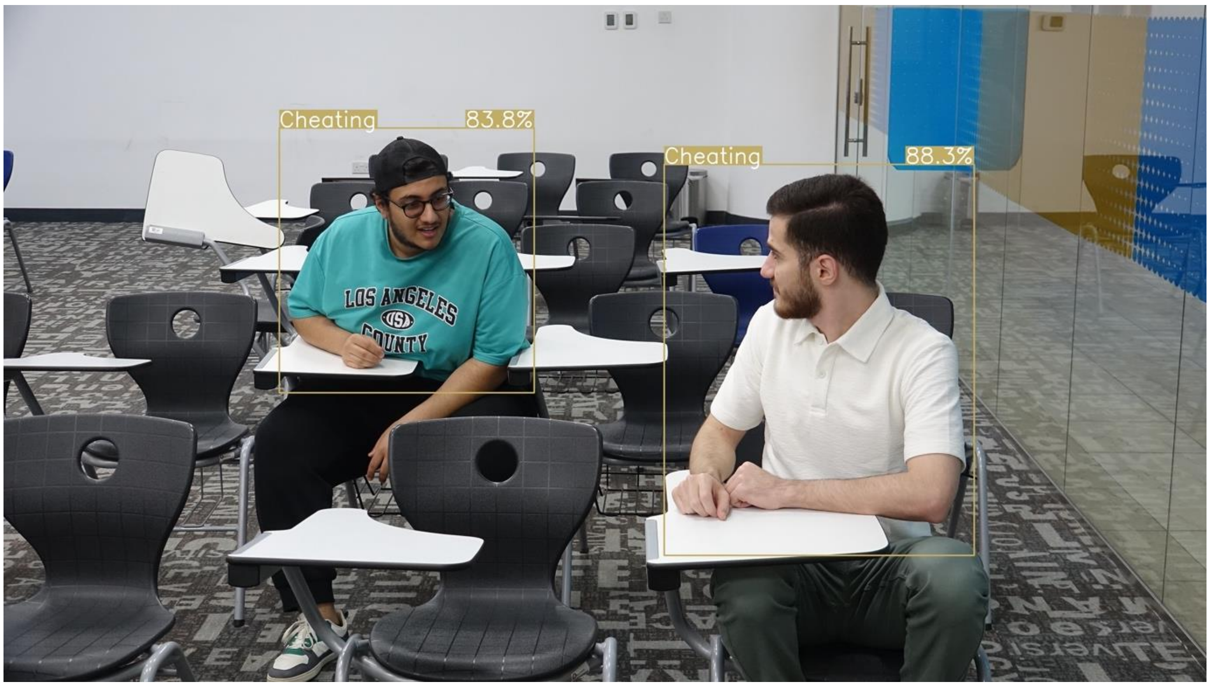


Figure 7.6.1 *Cheating Case*

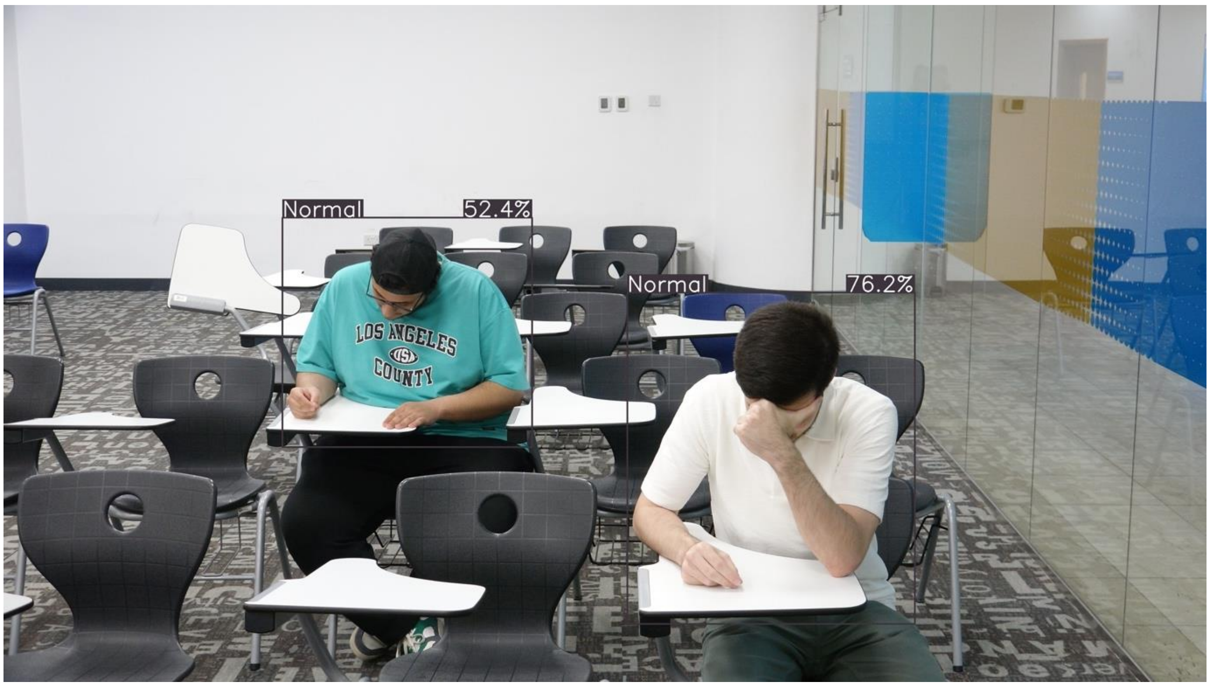


Figure 7.6.2 *Normal Case*

Chapter 8: Conclusion

Generally speaking, Camera Cheating Behavior Detection System is a suitable example of how the application of artificial intelligence can improve surveillance during exams. The outcome of the project was the implementation of a YOLO trained model to identify specific forms of cheating based on body movement and postures and interactive web interface to offer real time warning to the invigilators. Also, the system values the privacy of students by excluding facial recognition and addressing behaviors that represent cheating activity.

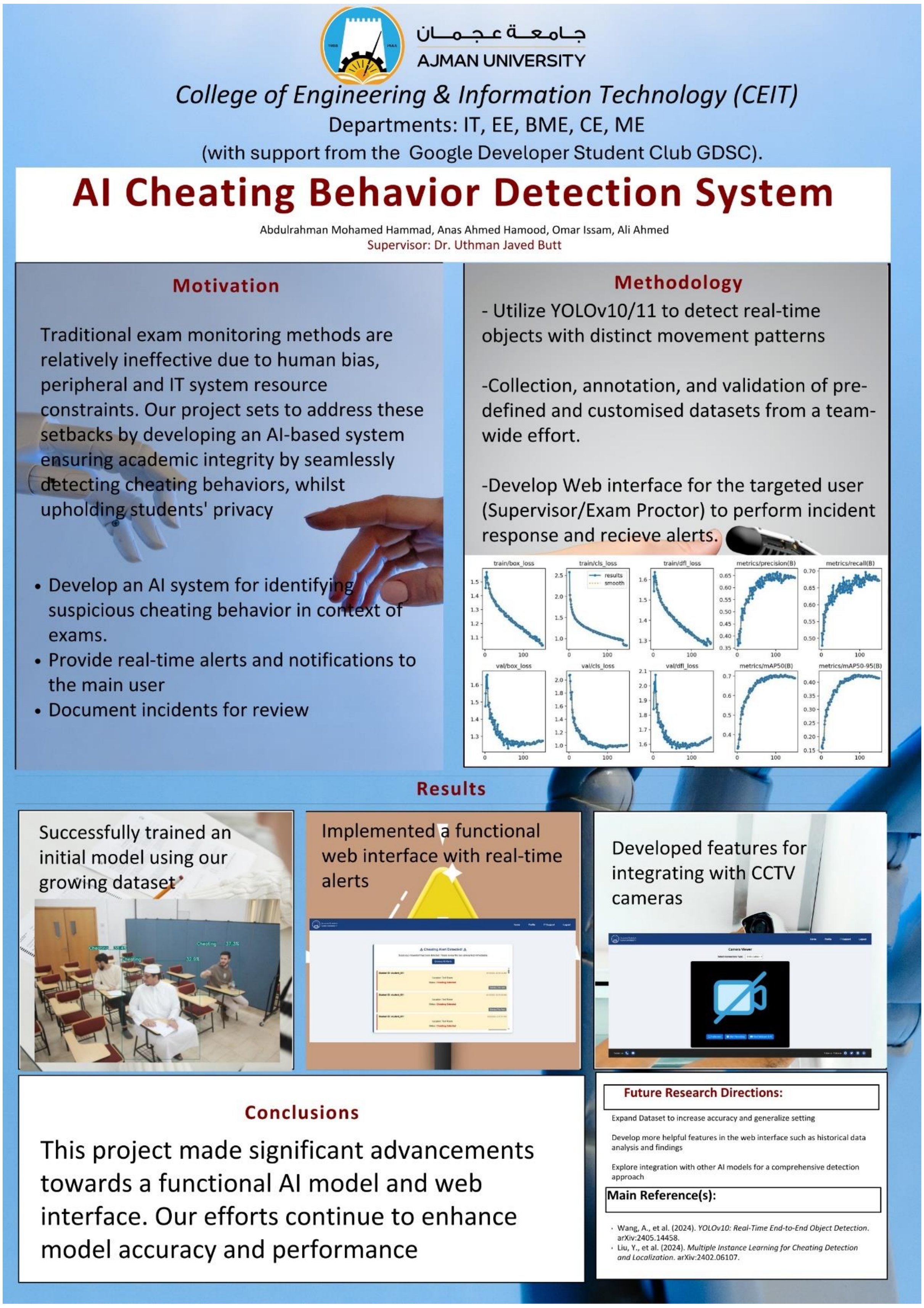
Initial evaluations of the system identified how it could detect open cases of academic fraud and notify relevant stakeholders promptly, hence allowing for human surveillance of suspected incidents. There are areas for improvement; for example, increasing the size of the dataset by adding more honest and more cheaters to expand the range of behaviors that the model will be trained to detect so as to increase the detection of subtler behaviors and wide scale field tests in live examination rooms. Further, integration of the system with existing classroom monitoring tools, as well as tuning of the alert interface using user feedback, improve adoption. Generally, such an attempt creates a good base to AI-enhanced examination monitoring and provides a practical framework to safeguard academic integrity in academia.

References

1. Aaronson, S. (2025, January 22). *AI expert Scott Aaronson on fighting plagiarism*. Axios. https://www.axios.com/local/austin/2025/01/22/artificial-intelligence-university-of-texas-scott-aaronson
2. AI detectors: An ethical minefield. (2024, December 12). *Northern Illinois University Center for Innovative Teaching and Learning*. https://citl.news.niu.edu/2024/12/12/ai-detectors-an-ethical-minefield/
3. Wang, A., Chen, H., Liu, L., Chen, K., Lin, Z., Han, J., & Ding, G. (2024). *YOLOv10: Real-time end-to-end object detection* (arXiv:2405.14458). arXiv. https://arxiv.org/abs/2405.14458
4. AI cheating is overwhelming the education system – but teachers shouldn't despair. (2024, August 24). *The Guardian*. https://www.theguardian.com/commentisfree/2024/aug/24/ai-cheating-chat-gpt-openai-writing-essays-school-university
5. AI-detectors biased against non-native English writers. (2023). *Stanford Human-Centered AI (HAI)*. https://hai.stanford.edu/news/ai-detectors-biased-against-non-native-english-writers
6. AI detectors don't work. Here's what to do instead. (2024). *MIT Sloan EdTech*. https://mitsloanedtech.mit.edu/ai/teach/ai-detectors-dont-work/
7. YOLOv10: Real-time object detection evolved. (2024). *viso.ai*. https://viso.ai/deep-learning/yolov10/
8. Sundaresan Geetha, A., Alif, M. A. R., Hussain, M., & Allen, P. (2024). Comparative analysis of YOLOv8 and YOLOv10 in vehicle detection: Performance metrics and model efficacy. *Vehicles*, *6*(3), 1364–1382. https://www.mdpi.com/2624-8921/6/3/65
9. Unmasking academic cheating behavior in the artificial intelligence era: Evidence from a list experiment. (2024). *Education and Information Technologies*. https://link.springer.com/article/10.1007/s10639-024-12495-4
10. Kundu, D., Mehta, A., Kumar, R., Lal, N., Anand, A., Singh, A., & Shah, R. R. (2024). *Keystroke dynamics against academic dishonesty in the age of LLMs* (arXiv:2406.15335). arXiv. https://arxiv.org/abs/2406.15335
11. Liu, Y., Ren, J., Xu, J., Bai, X., Kaur, R., & Xia, F. (2024). *Multiple instance learning for cheating detection and localization in online examinations* (arXiv:2402.06107). arXiv. https://arxiv.org/abs/2402.06107
12. Evaluating the efficacy of AI content detection tools in differentiating human and AI-authored content. (2023). *International Journal for Educational Integrity*, *19*, Article 5. https://edintegrity.biomedcentral.com/articles/10.1007/s40979-023-00140-5
13. Reassessing academic integrity in the age of AI: A systematic review of existing literature. (2025). *International Journal of Educational Research Open*, *6*, 100282. https://www.sciencedirect.com/science/article/pii/S2590291125000269
14. Teachers still can't trust AI text checkers. (2024, September 3). *Axios*. https://www.axios.com/2024/09/03/teachers-still-cant-trust-ai-text-checkers
15. There's a good chance your kid uses AI to cheat. (2025). *The Wall Street Journal*. https://www.wsj.com/tech/ai/chatgpt-ai-cheating-students-97075d3c
16. There's a fix for AI-generated essays. Why aren't we using it? (2024). *Vox*. https://www.vox.com/future-perfect/370419/chatgpt-schools-ai-cheating-plagiarism-detection

17. Does A.I. really encourage cheating in schools? (2024). *The New Yorker*. https://www.newyorker.com/news/fault-lines/does-ai-really-encourage-cheating-in-schools
18. How acceptable is it to use ChatGPT for homework? (2024). *The Times*. https://www.thetimes.co.uk/article/homework-cheats-parents-need-to-know-mnjdwjmjl
19. New data reveal how many students are using AI to cheat. (2024). *Education Week*. https://www.edweek.org/technology/new-data-reveal-how-many-students-are-using-ai-to-cheat/2024/04
20. Academic dishonesty using generative AI. (2024). *Northern Michigan University Center for Teaching and Learning*. https://nmu.edu/ctl/academic-dishonesty-using-generative-ai
21. diagrams.net. (2024). *Online diagram software*. https://app.diagrams.net/
22. Rakibul Hasan Shaon. (2023). *Exam Cheating1 Dataset* [Data set]. *Kaggle*. https://www.kaggle.com/datasets/rakibulhasanshaon69/exam-cheating1
23. Muhammed Ashiq K.M. (2023). *Exam Cheating Detection* [Data set]. *Kaggle*. https://www.kaggle.com/datasets/muhammedashiqkm/exam-cheating-detection

Appendix



AJMAN UNIVERSITY

جـامعـة عجـمـان

College of Engineering & Information Technology (CEIT)

Departments: IT, EE, BME, CE, ME

(with support from the Google Developer Student Club GDSC).

AI Cheating Behavior Detection System

Abdulrahman Mohamed Hammad, Anas Ahmed Hamood, Omar Issam, Ali Ahmed

Supervisor: Dr. Uthman Javed Butt

Motivation

Traditional exam monitoring methods are relatively ineffective due to human bias, peripheral and IT system resource constraints. Our project sets to address these setbacks by developing an AI-based system ensuring academic integrity by seamlessly detecting cheating behaviors, whilst upholding students' privacy

- Develop an AI system for identifying suspicious cheating behavior in context of exams.
- Provide real-time alerts and notifications to the main user
- Document incidents for review

Methodology

- Utilize YOLOv10/11 to detect real-time objects with distinct movement patterns

-Collection, annotation, and validation of pre-defined and customised datasets from a team-wide effort.

-Develop Web interface for the targeted user (Supervisor/Exam Proctor) to perform incident response and recieve alerts.

Results

Successfully trained an initial model using our growing dataset

Implemented a functional web interface with real-time alerts

Developed features for integrating with CCTV cameras

Conclusions

This project made significant advancements towards a functional AI model and web interface. Our efforts continue to enhance model accuracy and performance

Future Research Directions:

Expand Dataset to increase accuracy and generalize setting

Develop more helpful features in the web interface such as historical data analysis and findings

Explore integration with other AI models for a comprehensive detection approach

Main Reference(s):

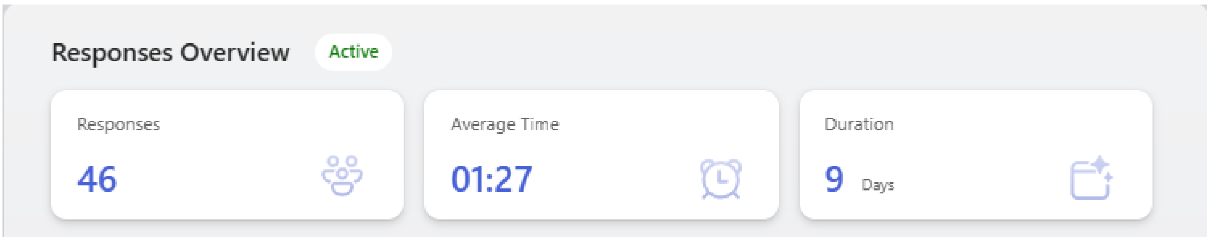
- Wang, A., et al. (2024). *YOLOv10: Real-Time End-to-End Object Detection.* arXiv:2405.14458.
- Liu, Y., et al. (2024). *Multiple Instance Learning for Cheating Detection and Localization.* arXiv:2402.06107.

Figure A.1 *Project Poster*

Appendix A.1 – Poster Design Overview

This poster describes the AI Cheating Behavior Detection System project in its general picture, its justification, and method and the obtained results and general conclusion. The justification part describes limitations in relation to traditional examination monitoring and emphasizes the need for artificially intelligent substitutes. Methodological approach shows the usage of YOLOv10/11 for object detection, usage of customized data sets and adding real-time notifications to web-based platform. The results report a successful prototype development with real-time monitoring, alerting, and initial CCTV systems integration. This poster was a deceptive instrument for presentations and displays, clearly enumerating the large benefits and achievements of this system.

Appendix A.2 – Survey Summary: -

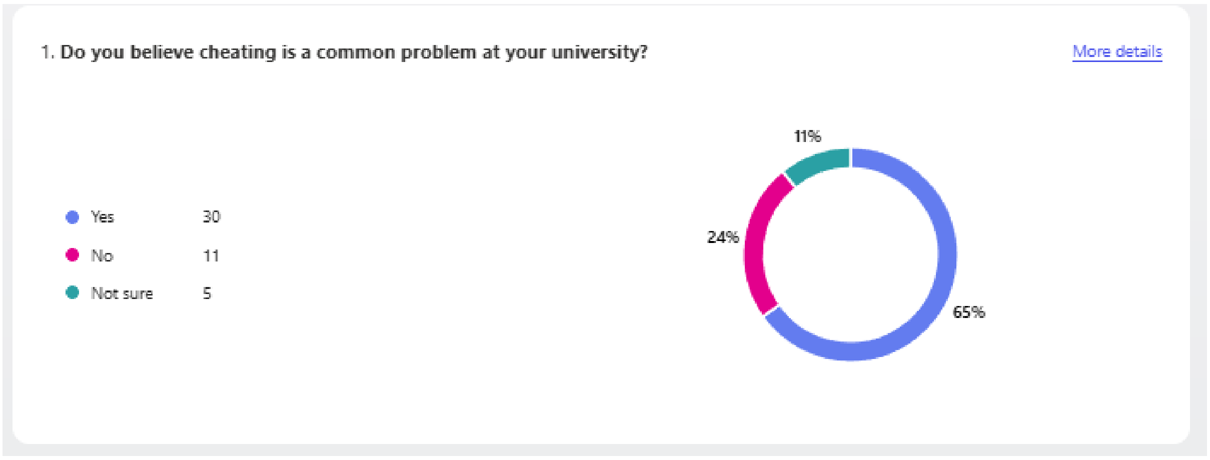


Responses Overview Active

Responses	Average Time	Duration
46	01:27	9 Days

Figure A.2.1 *Responses Overview*

This figure displays the aggregated responses from 46 participants over 9 days, collected via Microsoft Forms.



1. Do you believe cheating is a common problem at your university? More details

●	Yes	30
●	No	11
●	Not sure	5

Figure A.2.2 *Survey Results: Perceived Frequency of Cheating on Campus*

Question 1: Do you believe cheating is a common problem at your university?
Analysis:
By out of the 46 participants, 65% responded positively, this would therefore speak volumes when it comes to the incidence of cheating. To the contrary, 24% of the respondents disagreed and 11% were not sure.
The evidence presented herein provides eminent basis for developing AI powered interventions for improving academic integrity. The high percentages of participants who identify cheating indicate that students are aware that this is an obvious problem which is widespread.

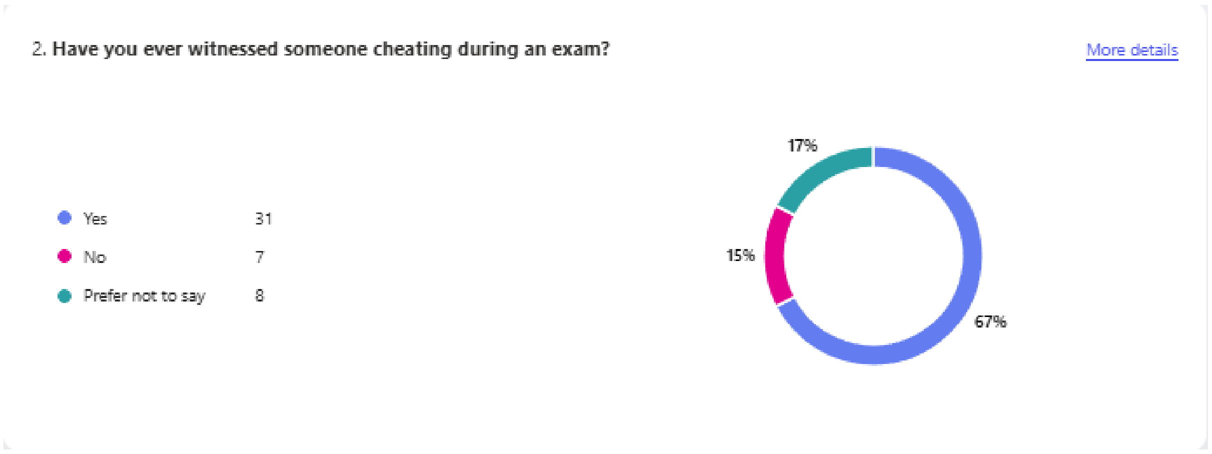


Figure A.2.3 Survey Results: Student Witnessing of Cheating Behaviours

Question 2: Have you ever witnessed someone cheating during an exam?
Analysis:
A big 67% of students reported first-hand observation of cases of academic dishonesty hence a testament that cheating is a lived experience and not a perception, 17% of recipients could not answer and 15% gave “No” answers which may mean either of non-sureness or indirect acknowledgment of cheating existence. Based on the figures, it can be seen that throughout the country there are widespread cheating incidents and there are gaps in the traditional monitoring strategies.

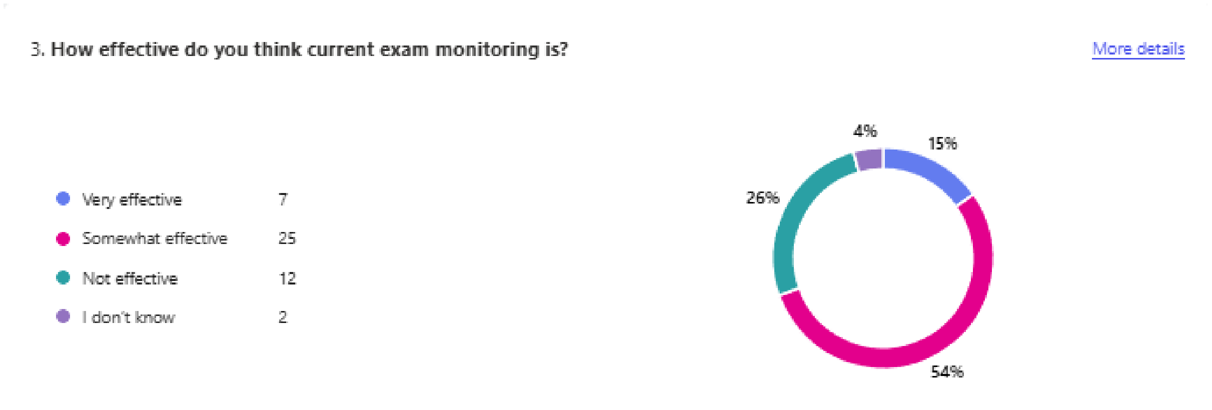
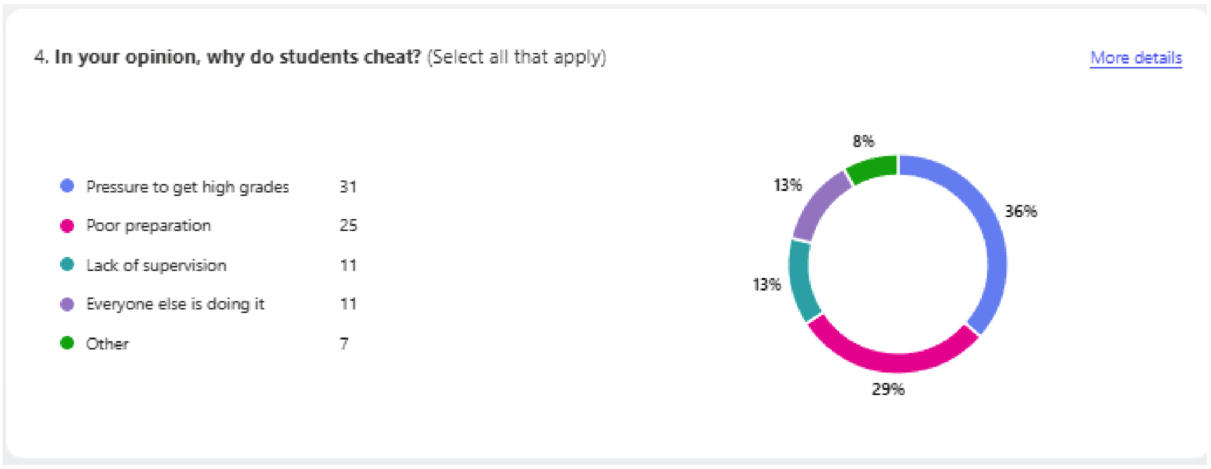


Figure A.2.4 Survey Results: Student Opinions on Monitoring Effectiveness

Question 3: How effective do you think current exam monitoring is?
Analysis:
Only 15% found it “very effective”; 54% “somewhat effective” – an obvious moderate confidence in tools we already had. Meanwhile 26% had answered “not effective” and 4% didn’t know. These results point to dissatisfaction with the current system and confirm the project’s aim of introducing AI driven improvements of increased reliability and objectivity.

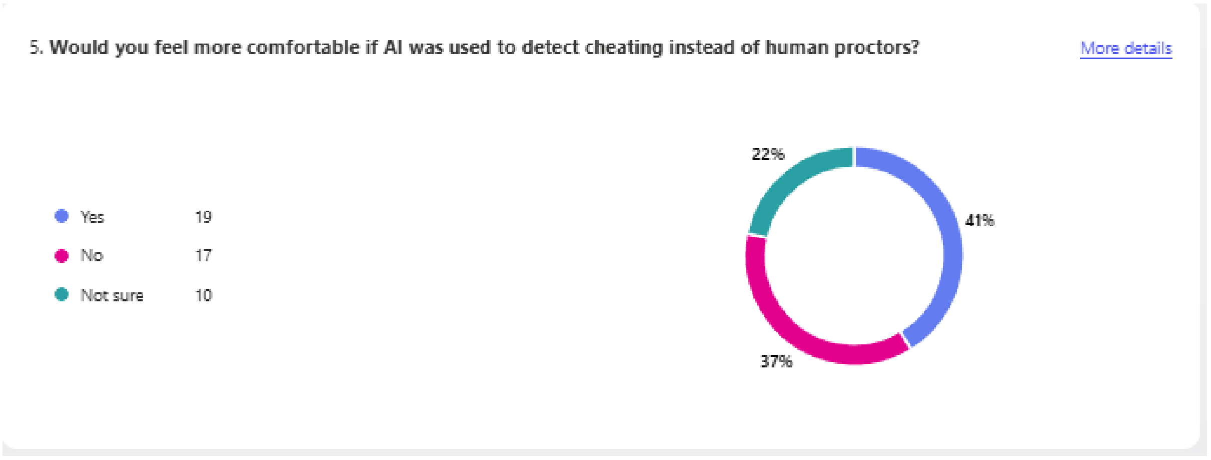


● Pressure to get high grades	31
● Poor preparation	25
● Lack of supervision	11
● Everyone else is doing it	11
● Other	7

Figure A.2.5 *Survey Results: Reasons Students Admit to Cheating*

Question 4: In your opinion, why do students cheat? ***(Select all that apply)***
Analysis:
Major identified factors were "Pressure for top grades" at 36%, "Not being properly prepared" at 29%, which says that high academic pressure and inadequate preparation are influential factors; 13% of subjects reported "Lack of supervision" and cited "The behavior of other pupils" while 8% of subjects selected the category "Other".

These findings show that academic cheating is not only aided by the perception of opportunity, but also by fundamental problems of education and psychology that can be solved with the use of improved monitoring methods.



● Yes	19
● No	17
● Not sure	10

Figure A.2.6 *Survey Results: Student Comfort with AI-Based Proctoring*

Question 5: Would you feel more comfortable if AI was used to detect cheating instead of human proctors?
Analysis:
Responses were evenly distributed with 41% comfortable with YES, 37% uncomfortable with YES and 22% unsure. This demonstrates an optimism surrounded by an ongoing lack of trust. Their pupils can describe phobias about fairness, open-mindedness or privacy. The need for a rationale behind what drives its mechanism in the system, what data, which its relies upon, and what protections against erroneous allegations create an interesting contrast.