

A Project Report on

# **Developing AI Based Comprehensive Framework for Online Assessments**

Submitted in partial fulfillment of the requirements for the award  
of the degree of

**Bachelor of Engineering**

in

**Information Technology**

by

**Swapnil Sapre(18104027)**

Under the Guidance of

**Prof. Vishal Badgujar**



**Department of Information Technology**  
**NBA Accredited**

A.P. Shah Institute of Technology  
G.B.Road,Kasarvadavli, Thane(W), Mumbai-400615  
UNIVERSITY OF MUMBAI  
**Academic Year 2021-2022**

## Approval Sheet

This Project Report entitled “***Developing AI Based Comprehensive Framework for Online Assessments***” Submitted by “***Swapnil Sapre (18104027)***” is approved for the partial fulfillment of the requirement for the award of the degree of ***Bachelor of Engineering in Information Technology*** from ***University of Mumbai***.

Prof. Vishal Badgujar  
Guide

Prof. Kiran Deshpande  
Head Department of Information Technology

Place: A.P. Shah Institute of Technology, Thane

Date:

## CERTIFICATE

This is to certify that the project entitled “*Developing AI Based Comprehensive Framework for Online Assessments*” submitted by “*Swapnil Sapre (18104027)*” for the partial fulfillment of the requirement for award of a degree *Bachelor of Engineering* in *Information Technology*, to the University of Mumbai, is a bonafide work carried out during academic year 2021-2022.

Prof. Vishal Badgujar  
Guide

Prof. Kiran Deshpande  
Head Department of Information Technology

Dr. Uttam D.Kolekar  
Principal

External Examiner(s)

1.

2.

Place: A.P. Shah Institute of Technology, Thane

Date:

## Acknowledgement

We have great pleasure in presenting the report on **Developing AI Based Comprehensive Framework for Online Assessments**. We take this opportunity to express our sincere thanks towards our guide **Prof. Vishal Badgujar** Department of IT, APSIT thane for providing the technical guidelines and suggestions regarding line of work. We would like to express our gratitude towards his constant encouragement, support and guidance through the development of project.

We thank **Prof. Kiran B. Deshpande** Head of Department, IT, APSIT for his encouragement during progress meeting and providing guidelines to write this report.

We thank **Prof. Vishal S. Badgujar** BE project co-ordinator, Department of IT, APSIT for being encouraging throughout the course and for guidance.

We also thank the entire staff of APSIT for their invaluable help rendered during the course of this work. We wish to express our deep gratitude towards all our colleagues of APSIT for their encouragement.

**Student Name1:** Swapnil Sapre

**Student ID1:** 18104027

## **Declaration**

We declare that this written submission represents our ideas in our own words and where others' ideas or words have been included, We have adequately cited and referenced the original sources. We also declare that We have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

(Swapnil Sapre - 18104027)

Date:

## **Abstract**

Online examinations are the way of conducting examinations on the user's mobile devices or laptops rather than actual paper. During situations when physical exams cannot be conducted online exams have been the preferred choice. In physical examinations the students doing malpractices reduces by a great amount as there are examiners physically present to monitor every student. In the same way the idea of Smart AI-ML based Online Framework is to reduce the malpractices done by the students in the online mode also as far as the current platforms in use are concerned. For this, there are some methods used which are based on the machine learning algorithms. The main task during the exam time is the continuous live detection of every candidate and for this facial detection using webcam is necessary. Before this, a proper pose setup and facial illumination needs to be ensured.

With the live detection happening the examiner can track the student from their end as well as get the alerts based on student's illegal conduct. Overall, the system is able to perform the proctoring tasks in a fully automated fashion and thereby requiring very minimum efforts from the examiner in monitoring the candidates. We have arrived at a conclusion that Smart Online Examination platform is a much viable solution to the existing platforms for conduction of the exams and doing the proctoring.

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.0.1	Facial Recognition and ML . . . . .	1
<b>2</b>	<b>Literature Review</b>	<b>3</b>
<b>3</b>	<b>Objectives</b>	<b>4</b>
<b>4</b>	<b>Project Design</b>	<b>5</b>
4.0.1	Existing System Architecture . . . . .	5
4.0.2	Proposed System Architecture . . . . .	6
<b>5</b>	<b>Project Implementation</b>	<b>10</b>
<b>6</b>	<b>Testing</b>	<b>20</b>
6.0.1	Functional Testing . . . . .	20
6.0.2	Non Functional Testing . . . . .	21
<b>7</b>	<b>Result</b>	<b>22</b>
<b>8</b>	<b>Conclusions and Future Scope</b>	<b>33</b>
	<b>Bibliography</b>	<b>34</b>
	<b>Appendices</b>	<b>36</b>
	Appendix-A . . . . .	36
	<b>Publication</b>	<b>38</b>

# List of Figures

4.1	Authentication of users . . . . .	5
4.2	Workflow for Student's End . . . . .	6
4.3	Workflow for teacher's end . . . . .	7
5.1	Register . . . . .	10
5.2	Login . . . . .	11
5.3	Create test . . . . .	12
5.4	Update Questions . . . . .	13
5.5	Give Test . . . . .	13
5.6	Randomize Questions in test . . . . .	14
5.7	Generation of secret password (OTP) . . . . .	14
5.8	Applying boundary box around captured face . . . . .	15
5.9	Mobile detection ML code . . . . .	16
5.10	User movement detection ML code . . . . .	17
5.11	Eye tracking detection ML code . . . . .	18
5.12	Libraries used . . . . .	19
7.1	Registered User in system . . . . .	22
7.2	Different User logging in system . . . . .	22
7.3	Teacher Dashboard . . . . .	23
7.4	Face validation error message . . . . .	23
7.5	Importing questions . . . . .	24
7.6	View Questions . . . . .	25
7.7	Update Questions . . . . .	26
7.8	Student's Dashboard . . . . .	26
7.9	Exam Instructions . . . . .	27
7.10	Exam Window . . . . .	28
7.11	Student Results . . . . .	29
7.12	Logs generation on teacher's dashboard . . . . .	30
7.13	Detection of mobile . . . . .	31
7.14	Alert on tab switching . . . . .	31
7.15	Tab switching activity . . . . .	32



# List of Abbreviations

AI:	Artificial Intelligence
ML:	Machine Learning
PDF:	Portable Document Format
YOLO:	You Only Look Once - Machine Learning Algorithm
ESC:	Escape key
CNN:	Convolutional Neural Networks algorithm
OTP:	One Time Password
CSV:	Comma Separated Value file

# Chapter 1

## Introduction

The traditional examination system process is long and time-consuming whereas the online examination system provides a quick and accurate solution within the specified time frame. Online exams require fewer resources and are more reliable and accurate as compared to offline tests. Students have quick access to online exams thereby reducing the necessity for faculty and staff for managing a large number of teaching materials and dealing with seating arrangements. The system provides a fast and accurate solution within the specified time frame and the scoring process is simpler and saves teachers time. The assessment is quick, reliable, and accurate. The online exam provides flexibility and security within it, as each student can receive random questions from an equivalent sample test. The online exam allows one to add videos, images, audio, PDF files, and more to the tests. Online exams allow for several features to stop malpractices, like locking the browser, disabling the print, arrow, and ESC keys [2]. These options provided by the online exams ensures the diversity of questions and proctoring which otherwise would not have been possible to that extent in the offline mode of examination.

Proctored exams are a type of online exam to protect the integrity and prevent malpractices of students. When such an online examination system is combined with automated proctoring it adds more value to the entire system as the teacher can conduct exams in a much efficient manner by getting all the activity of candidates at one place [4, 7]. A web proctored exam, involves a proctor overseeing an exam and monitoring the students online by using a webcam and microphone-compatible computer and an uninterrupted internet connection. Such kind of proctored systems provide multiple advantages to the teacher as they can now quickly create the tests, share them with the students and at the same time also get the automatic monitoring done by the system itself thereby saving their time from individually monitoring the students.

### 1.0.1 Facial Recognition and ML

Facial Recognition is a process that maps an individual's face and saves it as a face print. The software program makes use of algorithms to look at a captured face and compare it to the saved one to verify one's identity. Machine learning algorithms best recognize the numbers so it is pretty challenging. This numerical illustration of a face is named a function vector. As a result, using machine learning the function vector can detect the following aspects in a person's face:

1. Height of face (cm)
2. Width of the face (cm)
3. Average color of face (R, G, B)
4. Width of lips (cm)
5. Height of nose (cm)

By combining machine learning and making it applicable for online examinations has done betterment in this field. They yield a system that is fully automated in conducting examinations, monitoring each candidate without the examiner needing to do it individually, and grading the students along with getting automated attendance while also maintaining the records for future reference [6]. So in this work we have developed a web based proctoring system that has two aspects student and teacher dashboard both designed by taking help from [18].

# Chapter 2

## Literature Review

In [1] Facial feature detection algorithm is needed to detect the face of the candidate from several image sets. It is based on the average variance calculation of the three components in the face image like eyes, nose, and mouth and the extraction of these features.

From [2] A automated system is the one that completely does the tasks on its own. Such a system is required in terms of conducting the examinations and monitoring each candidate without the examiner needing to do it individually.

In paper [3] Verification of the users is required to give access to appropriate users only in the system. Authors has described about verifying the users on the day of the examination as the students have to enter their username as they are recognized during the registration process along with some other documents as proof.

In paper [4]The author says, while attempting the exam students will get proctored till the exam gets over.They wont be able to switch their tabs and all the logs will be generated at the admins end.

In [5] Tab locking mechanisms are necessary to ensure that the candidates do not navigate away from the exam window because it is locked while the exam is in progress. By having randomized system for question papers the students do not get the same questions and the questions are also shuffled.

In [6] the students have to enter their username as they are recognized during the registration process along with some other documents as proof and grading the students along with getting automated attendance while also maintaining the records for future reference.

From paper [7] The admin have to register themselves as they will be able to take the exam, generate the examination paper,The logs generated by the system will be displayed at admin side and any malpractice found they can terminate the exam.

Authors in [8] present face landmark detection which is a computer vision-based method to detect and track the user's eye movement by forming key points on it. Deepface is a deep learning python-based framework that can detect faces from images. It is a neural network-based and has already been trained with sample images of other people. To capture the candidate's eye position, eye tracking and pupil detection are necessary.

In [9] YOLO is a machine learning algorithm that is used to perform object detection. It can be used to detect objects in still images, provided sample video, or from a real-time webcam which is used in this system as discussed.

Authors in [10] have discussed CNN (Convolutional Neural Networks) a deep learning algorithm that helps in face verification and detection. The system requires training samples by taking a large number of sample data of students' faces obtained from online examinations.

# Chapter 3

## Objectives

1. To set up multi factor authentication mechanism in exam by creating a system that validates each user well before the exam using facial recognition.
2. To allow the faculty to alert warnings or end the exam if students found performing malpractices.
3. To perform object tracking (person tracking) throughout the exam.
4. To provide a platform for the faculty to easily setup the exam and get detailed reporting.
5. To send attendance and mis-activity report details to faculty.
6. To provide tab locking features to not allow students to navigate away from exam screen.
7. To deploy our framework on the cloud to achieve load balancing and scalability.

# Chapter 4

## Project Design

### 4.0.1 Existing System Architecture

There are many tools and software available on the market for taking tests online. However, most of them are hosted inside the system and are not intended for free use. Monitoring is not enabled on other widely used open-source evaluation platforms. This allows students to engage in professional illicit activity, such as talking to others or minimizing the screen to find answers while taking the test. The existing system includes the following aspects:

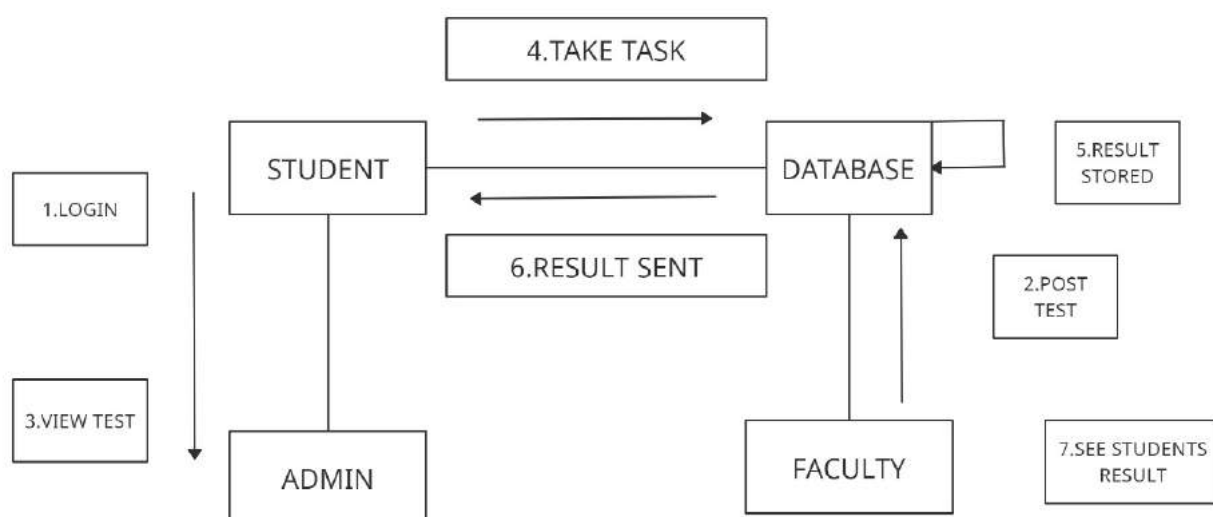


Figure 4.1: Authentication of users

1. The system includes a simple username and password combination to log into the exam portal.
2. For taking the attendance, the faculty has to supervise the student by calling out their name.
3. There is no diversity in questions as there are a series of similar questions available in the questionnaire for all the students taking the exam.

This project aims to solve the limitations of the existing systems and further develop a new system.

## 4.0.2 Proposed System Architecture

The proposed system is explained by dividing it into two blocks of sequence diagrams, one for the student end and another one for the examiner end.

### Student Block

Before the candidate begins to attempt the examination they must register and login into the exam portal with their credentials.

### Username and password

This combination is one of the most widespread authentication methods used due to its relatively low cost of deployment and convenience to the users. However, users can impersonate another person and there would be no means to detect if an online student is the one he or she claims to be. For this purpose, two-factor authentication mechanism is used. During the registration process, the student's mobile number or email plays a very important role. Every time the student logs in for an exam an OTP (one-time-password) will be sent to the registered mobile number or email. As a result, one student cannot take the exam on the other's behalf as one student can register only with one mobile number/email at a time.

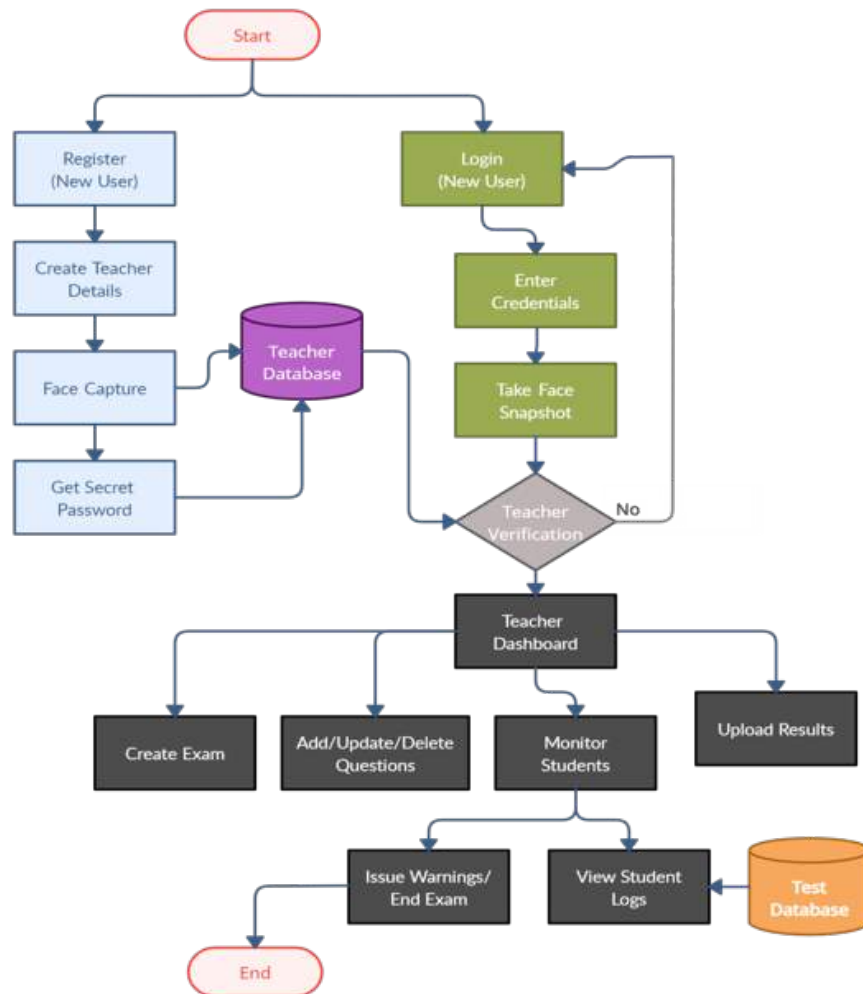


Figure 4.2: Workflow for Student's End

Figure 4.2 illustrates the process followed by the system from the students' perspective. The student taking up the examination needs to set up a profile in advance. The profile will contain all the personal information along with the pictures of them taken by varying distances and light variations. For this, the system requires the use of a webcam of the device to take pictures of the user's face. On the day of the examination, the students have to enter their username as they are recognized during the registration process along with some other documents as proof [3]. Upon successful login, they are directed to the facial recognition page. New images of the user are taken and compared against the ones stored in the database to establish if they match within suitable bounds. Once the student is validated by this method [17], they are allowed to proceed with their examinations. This completes the two-factor authentication process as now the students are verified both by secret password and face recognition.

### Examiner Block

Figure 4.2 below illustrates the process followed by the system from the examiner's perspective.

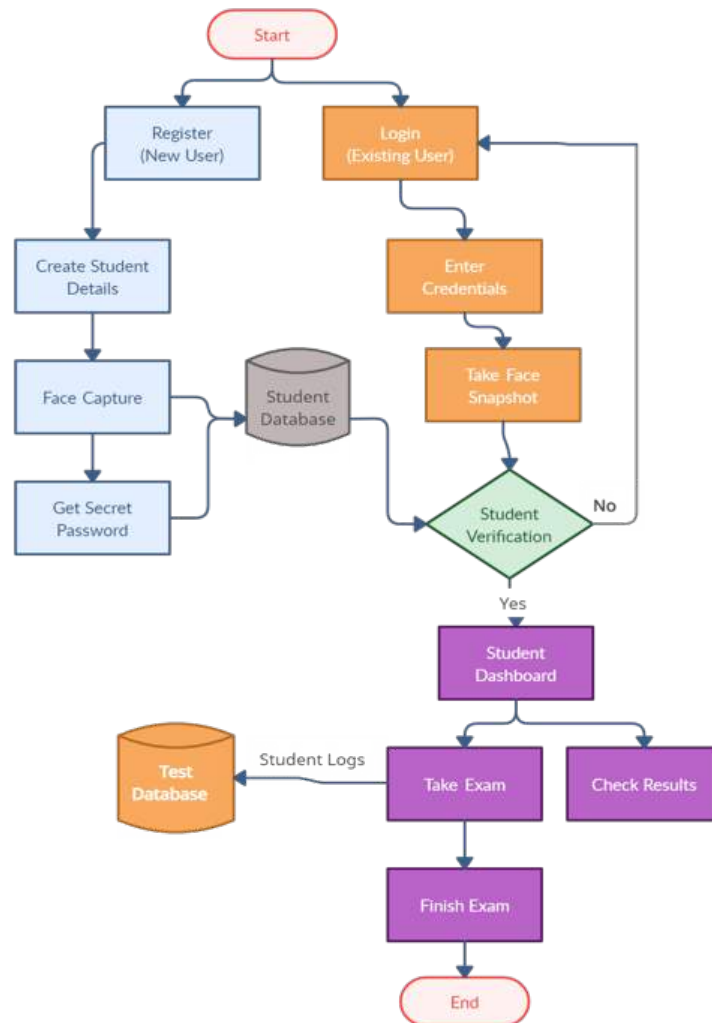


Figure 4.3: Workflow for teacher's end



The examiner will be using this system for exam generation and monitoring the students while they are giving the exam. Another feature the proposed system provides is the ability to issue warnings to the students doing malpractices and eventually ending the exam for those who are repeatedly found doing the same. The exam generation process and the activity report details methods are demonstrated in the next subsections.

### **Question paper generation process**

The most crucial task that the examiner needs to carry out is the creation of the question paper. The system has been developed to increase diversity and avoid all the students getting the same questions and in the same order [5]. At the time of creation the examiner can specify the type of question from different options provided as per their choice. The proposed system makes all examiners mandatory to display the questions in the form of images rather than plain text. This ensures that students do not copypaste the text as it is in another tab to get the answers. The system also allows the examiner to create a CSV/doc file-based question paper and then simply import it in.

### **Alert generation**

Alert is an important feature in this system. The webcam and microphone are used to capture the live images and audio from the student's end. The examiner is provided with a dashboard that enables them to see live data of all the students. Monitoring each individual is not possible in a real-time scenario and the alert based system generates such alerts. The alerts are displayed to the examiner containing the activity carried out [13]. Based on the above report, the examiner can send warnings to the student. After issuing multiple warnings if the student is still found doing malpractices then the examiner can decide to end the exam for that particular student. The next section discusses some of the methods used to detect the face, video, and audio of the candidates.

### **Object tracking and face recognition**

Dlib is a c++ based library consisting of machine learning algorithms and trained data sets that are capable of performing face recognition. The machine learning algorithm makes use of predefined data sets to track the object. For the dlib tracker to track the same object we need to define the boundary around our target specified in four coordinates (top, right, bottom, left). The sample input is passed to the dlib object tracker which tracks it in further frames. The updating process of the tracker is performed at the back end. The process will be looped throughout the examination process to track the students. The candidate will be allowed to give the examination only when they are verified by using face recognition and validation. The OpenCV library of python is capable of processing the images and videos to identify the faces and objects that appear in the samples. In the actual process of examination, the images of candidates are detected, and then a box is applied around the face of the person (as marked by a white rectangle in the image above) based on the four coordinates [16].

We attempt to match each face in the input image to the stored ones in the database. The result of this process will be as follows: The distance is within an acceptable value indicating the faces match or else the faces do not match. The object tracking algorithm also performs the process of object detection in the candidate's live environment simultaneously. The object detection happens in the video frames in combination with the YOLO real-time object detection algorithm [9].

### **Tab locking**

The last thing the system prevents the students from doing is accessing other windows to fetch the answers [5, 13]. If any user navigates away from the exam screen then an alert gets displayed on the screen and if the student gets 5 such alerts then automatically their test will be terminated.

# Chapter 5

## Project Implementation

This chapter contains the snapshots of the code written to implement different modules of the system. It starts with the login-register part and describes each functionality covering till student and teacher dashboard.

```
@app.route('/register', methods=['GET', 'POST'])
def register():
    cur = mysql.connection.cursor()
    #form = RegisterForm(request.form)
    if request.method == 'POST':
        cur = mysql.connection.cursor()
        name = request.form['name']
        email = request.form['email']
        username = request.form['username']
        imgdata1 = request.form['image_hidden']
        ut = request.form['user_type']
        sesOTP = generateOTP()
        session['secretpassword'] = sesOTP
        cur = mysql.connection.cursor()
        cur.execute('SELECT * from users')
        data = cur.fetchone()
        cur.execute('INSERT INTO users(username,name,email,secretpassword,user_type,user_image) values(%s,%s,%s,%s,%s,%s)', (username,name,email,sesOTP,ut,imgdata1))
        msg1 = Message('From Smart E-Exam', sender = sender, recipients = [email])
        msg1.body = "Thanks for registering. Please keep this secret code with you as it is needed for future logins. Your secret code is "+sesOTP+"."
        mail.send(msg1)
        mysql.connection.commit()
        cur.close()
        flash('Thanks for registering! Please check your email to confirm your email address.', 'success')
        # change in login function to redirect to warning page
    elif request.method == 'POST':
        flash('Thanks for registering! Please check your email to confirm your email address.', 'success')
    return render_template('register.html')
```

Figure 5.1: Register

For Figure 5.1 all the values from the user submitted form at the time of registration are stored in the database. It also makes use of ‘Message’ function that sends the secret password to the email.

```

@app.route('/login', methods=['GET','POST'])
def login():
    if request.method == 'POST':
        username = request.form['username']
        ut = request.form['user_type']
        password_candidate = request.form['secretpassword']
        imgdata1 = request.form['image_hidden']
        ut = request.form['user_type']
        cur = mysql.connection.cursor()
        cur.execute('SELECT * from users where username = %s and user_type=%s', (username,ut))
        data = cur.fetchone()
        if data:
            password = data['secretpassword']
            username = data['username']
            email = data['email']
            name = data['name']
            imgdata2 = data['user_image']
            nparr1 = np.frombuffer(base64.b64decode(imgdata1), np.uint8)
            nparr2 = np.frombuffer(base64.b64decode(imgdata2), np.uint8)
            image1 = cv2.imdecode(nparr1, cv2.COLOR_BGR2GRAY)
            image2 = cv2.imdecode(nparr2, cv2.COLOR_BGR2GRAY)
            img_result = DeepFace.verify(image1, image2, enforce_detection = False)
            if img_result["verified"] == True and password_candidate == password:
                session['logged_in'] = True
                session['username'] = username
                session['name'] = name
                session['user_type'] = ut
                session['email'] = email
                if ut == 'student':
                    return redirect(url_for('student_dashboard'))
                else:
                    return redirect(url_for('teacher_dashboard'))
            else:
                error = 'Either image not verified or Invalid password'
                return render_template('login.html', error=error)
        cur.close()
    else:
        error = 'Username not found'
        return render_template('login.html', error=error)
    return render_template('login.html')

```

Figure 5.2: Login

Figure 5.2 is the code of login function. In this all the details of the user which are already stored in the database are fetched. It also fetches the stored image of the user and decodes it first by doing computations using the numpy library. By using 'DeepFace' method, it then compares the two images of the user along with the secret password which validates whether the images and password match or not [17].

If all credentials and the face validation part is successfully matched then the redirect statement takes the user to their respective dashboard.

```

app.route('/create-test', methods = ['GET', 'POST'])
@is_logged
def create_test():
    form = UploadForm()
    if request.method == 'POST' and form.validate_on_submit():
        f = form.doc.data
        filename = secure_filename(f.filename)
        f.save('questions/' + filename)
        cur = mysql.connection.cursor()
        d = doctodict('questions/' + f.filename.replace(' ', '_').replace('.', '').replace('/', ''))
        wtest_id = generate_slug(2)
        test_id = form.testid.data
        try:
            for no, data in d.items():
                marks = data[('MARKS')] (1/2/3...)
                a = data[('OPTION A')]
                b = data[('OPTION B')]
                c = data[('OPTION C')]
                d = data[('OPTION D')]
                question = data[('QUESTION')]
                correct_ans = data[('CORRECT CHOICE')] (A/B/C/D)
                explanation = data[('EXPLANATION')] (OPTIONAL)

                cur.execute('INSERT INTO questions(test_id,qid,q,a,b,c,d,ans,marks,explanation) values(%s,%s,%s,%s,%s,%s,%s,%s,%s)',
                    (test_id,no,question,a,b,c,d,correct_ans,marks,explanation))
            mysql.connection.commit()

            start_date = form.start_date.data
            end_date = form.end_date.data
            start_time = form.start_time.data
            end_time = form.end_time.data
            start_date_time = str(start_date) + " " + str(start_time)
            end_date_time = str(end_date) + " " + str(end_time)
            show_result = form.show_result.data
            neg_mark = form.neg_mark.data

            duration = int(form.duration.data)*60
            password = form.password.data
            passp = form.pass.percentaga.data
            subject = form.subject.data
            topic = form.topic.data
            proctor_type = form.proctor_type.data
            cur.execute('INSERT INTO teachers (username, test_id, start, end, duration, show_ans, password, passp, subject, topic, neg_mark, proctoring_type) values(%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s)',
                (dict(session)['username'], test_id, start_date_time, end_date_time, duration, show_result, password, passp, subject, topic, neg_mark, proctor_type))
            mysql.connection.commit()
            cur.close()
            flash(f'Test ID: {test_id}', 'success')
            return redirect(url_for('create_test'))
        except Exception as e:
            print(e)
            flash('Invalid Input File Format', 'danger')
            return redirect(url_for('create_test'))
    return render_template('create_test.html', form = form)

```

Figure 5.3: Create test

Figure 5.3 represents code for create test function. It first verifies whether all the fields necessary have been provided or not. If it is validated then all the details like question description, options, correct answer and marks from the document file which the teacher has uploaded are extracted and stored in the database.

```

@app.route('/update/<testid>/<qid>', methods=['GET','POST'])
@is_logged
def update_quiz(testid, qid):
    if request.method == 'GET':
        cur = mysql.connection.cursor()
        cur.execute('SELECT * FROM questions where test_id = %s and qid = %s', (testid,qid))
        ureresults = cur.fetchall()
        mysql.connection.commit()
        return render_template("updateQuestions.html", ureresults=ureresults)
    if request.method == 'POST':
        ques = request.form['ques']
        ao = request.form['ao']
        bo = request.form['bo']
        co = request.form['co']
        do = request.form['do']
        anso = request.form['anso']
        markso = request.form['mko']
        cur = mysql.connection.cursor()
        cur.execute('UPDATE questions SET q = %s, a = %s, b = %s, c = %s, d = %s, ans = %s, marks = %s where test_id = %s and qid = %s', (ques,ao,bo,co,do,anso,markso,testid,qid))
        cur.connection.commit()
        flash('Updated successfully.', 'success')
        cur.close()
        return redirect(url_for('updateetidlist'))
    else:
        flash('ERROR OCCURED.', 'error')
        return redirect(url_for('updateetidlist'))

```

Figure 5.4: Update Questions

Figure 5.4 contains code for updating the questions. The code first checks whether the test has been started or not. If the test has not started then in this case the test ID whose questions the teacher wants to update is passed as a parameter and then the respective values which the teacher has updated using the form gets extracted and updated in the database under that test ID.

```

@app.route('/give-test/<testid>', methods=['GET','POST'])
@is_logged
def test(testid):
    global duration,marked_ans,proctortype
    tid = testid
    if request.method == 'GET':
        try:
            data = {'duration': duration, 'marks': '', 'q': '', 'a': '', 'b': '', 'c': '', 'd': ''}
            return render_template("quiz.html", **data, answers=marked_ans, proctortype=proctortype,tid=tid)
        except:
            return redirect(url_for("give_test"))
    else:
        cur = mysql.connection.cursor()
        flag = request.form['flag']
        if flag == 'get':
            num = request.form['no']
            results = cur.execute('SELECT * from questions where test_id = %s and qid = %s', (testid, num))
            if results > 0:
                data = cur.fetchone()
                del data['ans']
                cur.close()
                return json.dumps(data)
            elif flag == 'mark':
                qid = request.form['qid']
                print(qid)
                ans = request.form['ans']
                print(ans)
                results = cur.execute('SELECT * from students where test_id = %s and qid = %s and username = %s', (testid, qid, session['username']))
                if results > 0:
                    cur.execute('UPDATE students set ans = %s where test_id = %s and qid = %s and username = %s', (ans,testid, qid, session['username']))
                    mysql.connection.commit()
                    cur.close()
                else:
                    cur.execute('INSERT INTO students(username,test_id,qid,ans) values(%s,%s,%s,%s)', (session['username'], testid, qid, ans))
                    mysql.connection.commit()
                    cur.close()
            elif flag == 'time':
                cur = mysql.connection.cursor()
                time_left = request.form['time']
                try:
                    cur.execute('UPDATE studenttestinfo set time_left=SEC_TO_TIME(%s) where test_id = %s and username = %s and completed=0', (time_left, testid, session['username']))
                    mysql.connection.commit()
                    cur.close()
                    return json.dumps({'time':'fired'})
                except:
                    pass
            else:
                cur = mysql.connection.cursor()
                cur.execute('UPDATE studenttestinfo set completed=1 where test_id = %s and username = %s', (testid, session['username']))
                mysql.connection.commit()
                cur.close()
                flash('Test submitted successfully', 'Info')
                return json.dumps({'sql':'fired'})

```

Figure 5.5: Give Test

Figure 5.5 shows the code for giving the test. It makes use of a dictionary having various fields such as question and marked answers. This dictionary is updated whenever student marks the answer in the test. It fetches the time left in the test and updates this dictionary in the backend. The fetched answers and time are sent as the json response from the server and this code attempts to save the values in the database.

```
@app.route('/randomize', methods = ['POST'])
def random_gen():
    if request.method == "POST":
        id = request.form['id']
        cur = mysql.connection.cursor()
        results = cur.execute('SELECT count(*) from questions where test_id = %s', [id])
        if results > 0:
            data = cur.fetchone()
            total = data['count(*)']
            nos = list(range(1,int(total)+1))
            random.Random(id).shuffle(nos)
            cur.close()
            return json.dumps(nos)
```

Figure 5.6: Randomize Questions in test

Figure 5.6 is code for having randomized questions. It first counts the total questions present in that test and then applies a 'Random' function to get random questions and 'shuffle' function to shuffle the order of questions displayed in the test.

```
def generateOTP():
    otp=str(randint(00000,99999))
    return otp
```

Figure 5.7: Generation of secret password (OTP)

Figure 5.7 contains code for generation of secret password (OTP). It just includes a 'randint' function that generates a string of random numbers between the given ranges. In this code the OTP generated will be between 00000 to 99999. Once generated that number is passed to the register function as a parameter which will send it to the user on their email during registration process.

```

wf.close()

def draw_outputs(img, outputs, class_names):
    boxes, objectness, classes, nums = outputs
    boxes, objectness, classes, nums = boxes[0], objectness[0], classes[0], nums[0]
    wh = np.flip(img.shape[0:2])
    for i in range(nums):
        x1y1 = tuple((np.array(boxes[i][0:2]) * wh).astype(np.int32))
        x2y2 = tuple((np.array(boxes[i][2:4]) * wh).astype(np.int32))
        img = cv2.rectangle(img, x1y1, x2y2, (255, 0, 0), 2)
        img = cv2.putText(img, '{} {:.4f}'.format(
            class_names[int(classes[i])], objectness[i]),
            x1y1, cv2.FONT_HERSHEY_COMPLEX_SMALL, 1, (0, 0, 255), 2)
    return img

yolo_anchors = np.array([(10, 13), (16, 30), (33, 23), (30, 61), (62, 45),
                        (59, 119), (116, 90), (156, 198), (373, 326)],
                        np.float32) / 416

yolo_anchor_masks = np.array([[6, 7, 8], [3, 4, 5], [0, 1, 2]])

```

Figure 5.8: Applying boundary box around captured face

Figure 5.8 contains code for the machine learning algorithm used to apply a box around the student's face captured from the webcam. It requires the input as captured image 'img' along with certain parameters such as 'classnames' and 'outputs' which has been defined under YOLO training model. It then detects the coordinates in the user's face and then applies a box using cv2.rectangle method. Using cv2.putText() method it displays the label of the identified object in the image.



```

def get_frame(imgData):
    nparr = np.frombuffer(base64.b64decode(imgData), np.uint8)
    image = cv2.imdecode(nparr, cv2.COLOR_BGR2GRAY)
    ret = True

    size = image.shape
    font = cv2.FONT_HERSHEY_SIMPLEX
    model_points = np.array([
        (0.0, 0.0, 0.0),          # Nose tip
        (0.0, -330.0, -65.0),     # Chin
        (-225.0, 170.0, -135.0),  # Left eye left corner
        (225.0, 170.0, -135.0),   # Right eye right corne
        (-150.0, -150.0, -125.0), # Left Mouth corner
        (150.0, -150.0, -125.0)   # Right mouth corner
    ])

    focal_length = size[1]
    center = (size[1]/2, size[0]/2)
    camera_matrix = np.array(
        [[focal_length, 0, center[0]],
         [0, focal_length, center[1]],
         [0, 0, 1]], dtype = "double"
    )

    img = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
    img = cv2.resize(img, (320, 320))
    img = img.astype(np.float32)
    img = np.expand_dims(img, 0)
    img = img / 255
    class_names = [c.strip() for c in open("models/classes.TXT").readlines()]
    boxes, scores, classes, nums = yolo(img)
    count=0
    mob_status = ""
    person_status = ""
    for i in range(nums[0]):
        if int(classes[0][i] == 0):
            count +=1
        if int(classes[0][i] == 67):
            print('Mobile Phone detected')
            mob_status = 1
        else:
            print('Mobile Phone not detected')
            mob_status = 0
    print(mob_status)

```

Figure 5.9: Mobile detection ML code

Figure 5.9 represents code of mobile detection. In this once again the student's captured image from live environment is used. It first does some filtering on the image by changing the type and its size for easier application of the algorithm processes. Then based on the available class labels, it checks if the obtained values from the provided image match those class values [12]. If value of the class is 67: then mobile is detected else it is not detected. Once the mobile detection is found it increments the counter of mobile detections whose value is then stored in the database.

```

user_move1=""
user_move2=""
if ret == True:
    faces = find_faces(image, face_model)
    for face in faces:
        marks = detect_marks(image, landmark_model, face)
        image_points = np.array([
            marks[30], # Nose tip
            marks[8], # Chin
            marks[36], # Left eye left corner
            marks[45], # Right eye right corner
            marks[48], # Left Mouth corner
            marks[54], # Right mouth corner
        ], dtype="double")
        dist_coeffs = np.zeros((4,1)) # Assuming no lens distortion
        (success, rotation_vector, translation_vector) = cv2.solvePnP(model_points, image_points, camera_matrix, dist_coeffs, flags=cv2.SOLVEPNP_UPNP)

        (nose_end_point2D, jacobian) = cv2.projectPoints(np.array([(0.0, 0.0, 1888.0)]), rotation_vector, translation_vector, camera_matrix, dist_coeffs)

        for p in image_points:
            cv2.circle(image, (int(p[0]), int(p[1])), 3, (0,0,255), -1)

        p1 = ( int(image_points[0][0]), int(image_points[0][1]))
        p2 = ( int(nose_end_point2D[0][0]), int(nose_end_point2D[0][1]))
        x1, x2 = head_pose_points(image, rotation_vector, translation_vector, camera_matrix)

        try:
            m = (p2[1] - p1[1])/(p2[0] - p1[0])
            ang1 = int(math.degrees(math.atan(m)))
        except:
            ang1 = 90

        try:
            m = (x2[1] - x1[1])/(x2[0] - x1[0])
            ang2 = int(math.degrees(math.atan(-1/m)))
        except:
            ang2 = 90

        if ang1 >= 48:
            user_move1 = 2
            print('Head is down')
        elif ang1 <= -48:
            user_move1 = 1
            print('Head is up')
        else:
            user_move1 = 0

        if ang2 >= 48:
            print('Head is right')
            user_move2 = 4
        elif ang2 <= -48:
            print('Head is left')
            user_move2 = 3
        else:
            user_move2 = 0

```

Figure 5.10: User movement detection ML code

Figure 5.10 represents code of user movement detection. In this part, to detect the user head movement first face is to be detected. For this 'findfaces()' function is used which detects and marks points on nose, chin, etc. Using the headposepoints it determines the movement of the head based on particular angle values whether the head is down, up, right or moving left [15].

```

ret, jpeg = cv2.imencode('.jpg', image)
jpg_as_text = base64.b64encode(jpeg)

gaze.refresh(image)

frame = gaze.annotated_frame()
eye_movements = ""

if gaze.is_blinking():
    eye_movements = 1
    print("Blinking")
elif gaze.is_right():
    eye_movements = 4
    print("Looking right")
elif gaze.is_left():
    eye_movements = 3
    print("Looking left")
elif gaze.is_center():
    eye_movements = 2
    print("Looking center")
else:
    eye_movements = 0
    print("Not found!")
print(eye_movements)

```

Figure 5.11: Eye tracking detection ML code

Figure 5.11 shows code for eye tracking. In this, gaze tracking function is used which analyzes the user's eye movement. It first requires the input image that has been marked with the eye points on it. Then using different functions, the eye points are analysed and found out whether the user is blinking, looking left, looking right, etc. The count of the respective eye positions is taken, stored and then displayed later on in the logs.

```

from flask import Flask, request, render_template, flash, redirect, url_for, session, logging, send_file, jsonify
from flask_mysql import MySQL
from wtforms import Form, StringField, TextAreaField, PasswordField, validators, DateTimeField, BooleanField, IntegerField, DecimalField, HiddenField, SelectField, RadioField
from flask_wtf import FlaskForm
from flask_wtf.file import FileField, FileRequired, FileAllowed
from passlib.hash import sha256_crypt
from functools import wraps
from werkzeug.utils import secure_filename
from docx import Document
from coolname import generate_slug
from datetime import timedelta, datetime
from random import randint
import numpy as np
import base64
import cv2
from deepface import DeepFace
from flask_mail import Mail, Message
from threading import Thread
from flask import render_template_string
from itsdangerous import URLSafeTimedSerializer
from validate_email import validate_email
import random
import json
import csv
import math
import operator
import pandas as pd
from wtforms.components import TimeField, DateField
# from wtforms.fields.html5 import DateField
from wtforms.validators import ValidationError, NumberRange, InputRequired
import socket
import camera
# from emailverifier import Client
from waitress import serve

```

Figure 5.12: Libraries used

Figure 5.12 shows the list of libraries used for the project. These all libraries are necessary in order to implement different web and machine learning (ML) related functions. Out of all the libraries included the most important ones are:

1. **numpy**: It is very useful for performing mathematical and logical operations on Arrays. So it is used to compute values from the two image arrays of the user for comparing purpose.
2. **cv2**: It is a openCV library that can process images and videos to identify objects and faces in it.
3. **deepface**: It is a lightweight face recognition library used to work with user face images and extract key-points of eye and mouth movement.
4. **dlib**: It's a landmark's facial detector with pre-trained models, the dlib is used to estimate the location of 68 coordinates (x, y) that map the facial points on a person's face.
5. **flask**: It is used for developing web applications using python,

# Chapter 6

## Testing

### 6.0.1 Functional Testing

#### Unit Testing

Out of different standard testing methods available, unit testing has been used as our main testing method. It is a testing process that tests individual components of the application in terms of its functionality and working. In our online examination application we have several such units in place that perform their respective action based on the user's input. These small units include creating an exam, adding new questions, updating questions, monitoring the students from the teacher's end. From the student's end it includes attempting exam, saving the answers, checking the results and generation of logs. And in common, the registration and login component includes taking user's snapshots and verifying them later.

For all the above mentioned components we found testing them individually has best suited here. Unit testing has also been used because it helps in finding any issues at earlier stage and in the individual component rather than finding it later in the entire application.

#### Integration Testing

After each unit is tested individually, it is integrated with other units to create a functioning module that can perform specific tasks. These are then tested as a combined group through integration testing to ensure whole segment of the application behave as expected. So in this system once the individual components like creation and management of exams is completed it is combined and developed as one single teacher module which is then tested as a whole. Similarly individual units like attempting the exam and checking the results is combined as a single student module.

## **6.0.2 Non Functional Testing**

### **Compatibility Testing**

Compatibility testing is used to see how an application will work in different environments. It is used to check that the application is compatible with different operating systems, browsers and platforms. The framework we used to develop this application is Flask - Python. It is framework that is used to create web applications. This system being an online proctoring web application is compatible on any operating system and also supported on different web browsers and devices that have a working webcam and microphone.

### **Load Testing**

Load testing checks how the software behaves under normal and peak conditions. This is done to determine how much work the software can handle before performance is affected. The load can be checked by allowing multiple users to use the system simultaneously to see how the system is performing.

# Chapter 7

## Result

This chapter includes the results which has been produced by the system in terms of different functionalities. These results have been demonstrated in the form of snapshots taken from the working system and describing what each one of them does. The flow in this chapter is then divided into three parts: Login and register component, Student dashboard component and teacher dashboard component. All the functionalities that come under the respective components has also been described in this chapter.

### 1) Login - Register



Figure 7.1: Registered User in system

The user in this figure has done the registration process by filling all the details. So their image has been stored in the database.



Figure 7.2: Different User logging in system

Another user as shown in this figure is now trying to login in the system. However this

user has not done the registration yet.

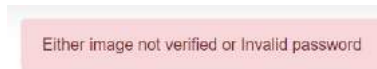


Figure 7.3: Teacher Dashboard

As the two users are different their face validation does not give positive result. This means that the user in 7.2 does not get access to the system. This has been shown as face validation error in figure 7.3.

## 2) Teacher Dashboard

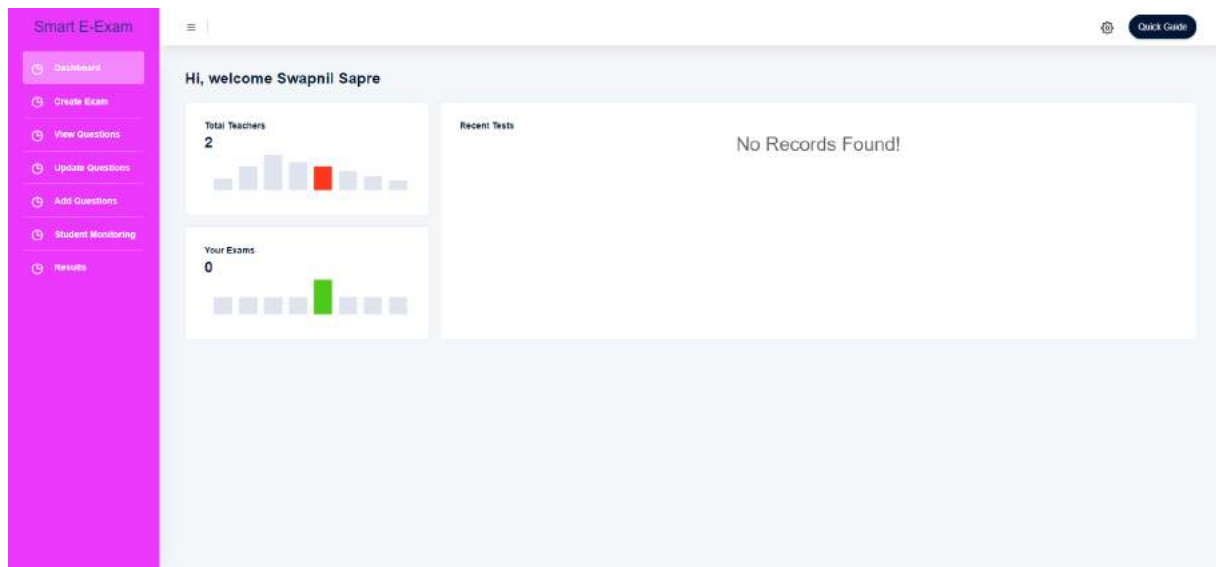


Figure 7.4: Face validation error message

Teacher dashboard provides different options to choose from like creating tests, viewing the questions, updating and adding questions and student monitoring. These options can be selected from the menu on the left. It also shows the recent tests created by that teacher. There is also a quick guide button on the top that will show a tutorial of what different options are provided for.



### a) Importing Questions

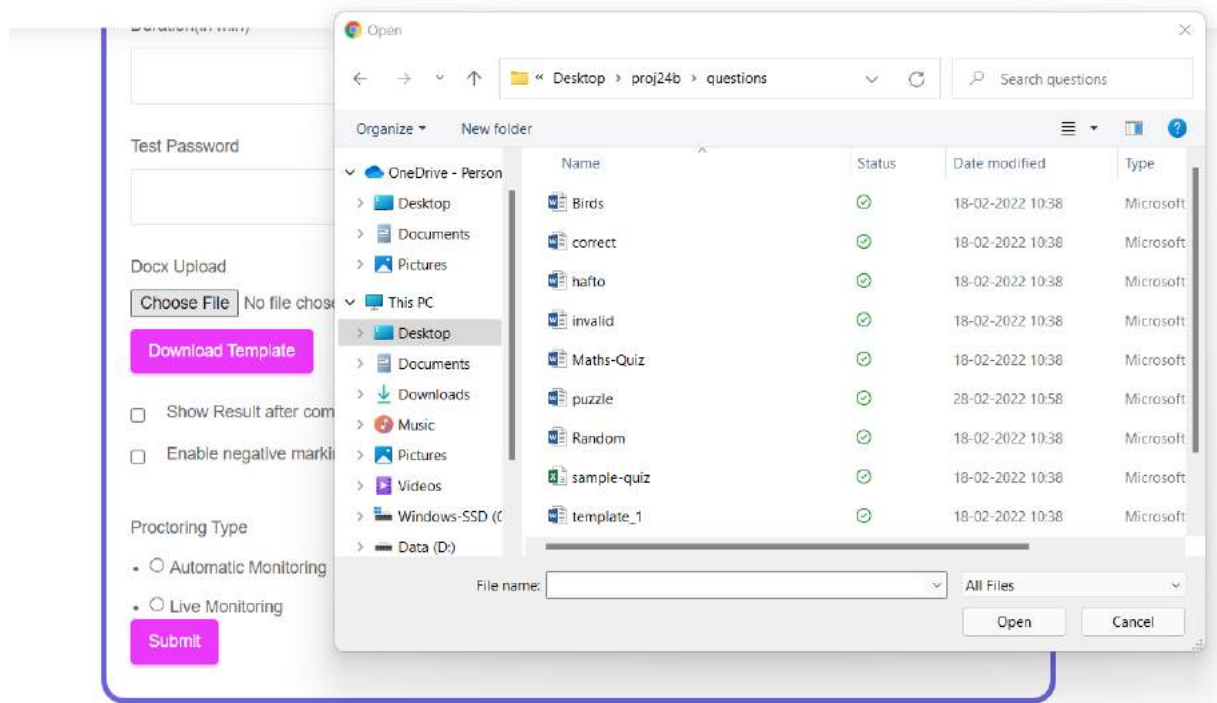
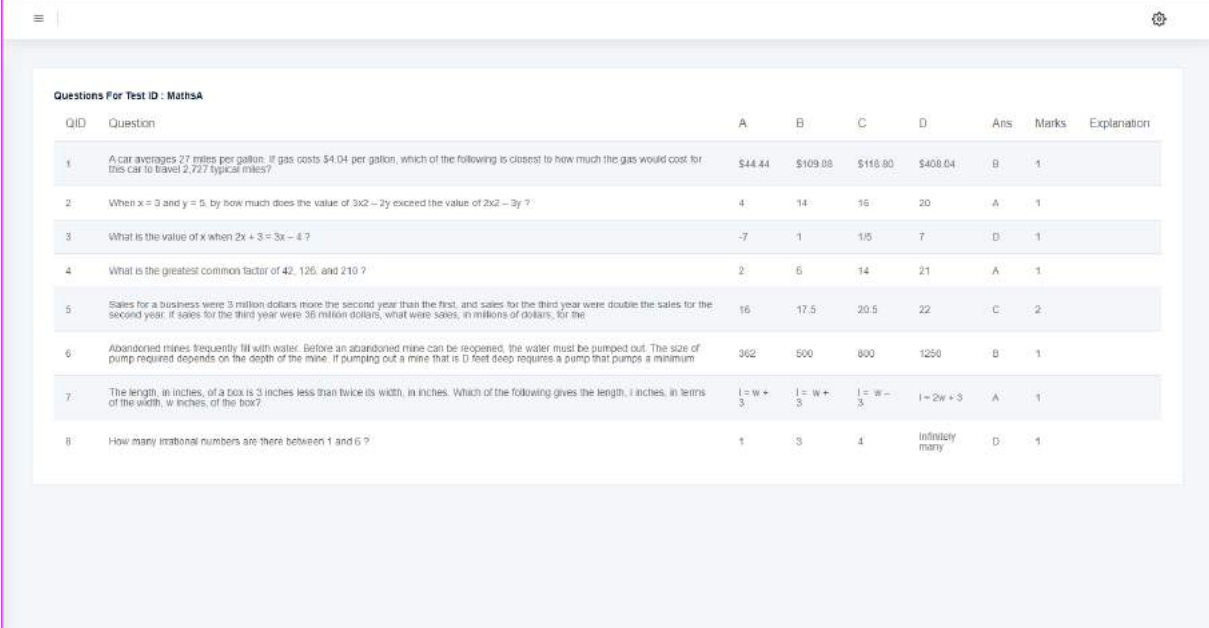


Figure 7.5: Importing questions

In this part, a document file containing all the questions for the test is being selected and imported. The teacher has to choose a word file containing the questions of the test and simply upload it in the examination form. If the teacher is using it for the first time then a sample template has been provided which can be downloaded and then the questions can be added.

## b) View Questions



Questions For Test ID : MathsA							
QID	Question	A	B	C	D	Ans	Marks
1	A car averages 27 miles per gallon. If gas costs \$4.04 per gallon, which of the following is closest to how much the gas would cost for this car to travel 2,727 typical miles?	\$44.44	\$109.08	\$118.80	\$408.04	B	1
2	When $x = 3$ and $y = 5$ , by how much does the value of $3x^2 - 2y$ exceed the value of $2x^2 - 3y$ ?	4	14	16	20	A	1
3	What is the value of $x$ when $2x + 3 = 3x - 4$ ?	-7	1	1/5	7	D	1
4	What is the greatest common factor of 42, 126, and 210?	2	6	14	21	A	1
5	Sales for a business were 3 million dollars more the second year than the first, and sales for the third year were double the sales for the second year. If sales for the third year were 36 million dollars, what were sales, in millions of dollars, for the	16	17.5	20.5	22	C	2
6	Abandoned mines frequently fill with water. Before an abandoned mine can be reopened, the water must be pumped out. The size of pump required depends on the depth of the mine. If pumping out a mine that is 0 feet deep requires a pump that pumps a minimum	362	500	800	1290	B	1
7	The length, in inches, of a box is 3 inches less than twice its width, in inches. Which of the following gives the length, $l$ inches, in terms of the width, $w$ inches, of the box?	$l = w + 3$	$l = w + \frac{1}{3}$	$l = w - \frac{1}{3}$	$l = 2w + 3$	A	1
8	How many irrational numbers are there between 1 and 6?	1	3	4	Infinitely many	D	1

Figure 7.6: View Questions

The teacher can see all the questions set for that particular test by clicking on View questions from the navigation tab. The questions for the selected test ID will be displayed in the form of a table.

## c) Update Questions

**Q 2: Which two cities are historical places?**

A: A and C

C: C and F

ANSWER: C

UPDATE

B: B and F

D: B and E

MARKS: 2

---

**Q 3: Which two cities are hill stations?**

A: A and B

C: B and D

ANSWER: C

UPDATE

B: C and A

D: A and F

MARKS: 2

In part (b) below the updating process of the question is being done which allows the teacher to edit the question description, the options or the marks.

## UPDATE QUESTIONS

Question 2 of MathsA

Question

When  $x = 3$  and  $y = 5$ , by how much does the value of  $3x^2 - 2y$  exceed the value

OPTION A

A

4

OPTION B

B

14

OPTION C

C

16

OPTION D

D

20

ANSWER

Answer

A

MARKS

Marks

1

UPDATE!

Figure 7.7: Update Questions

### 3) Student Dashboard

Menu

Dashboard

Attempt Test

Results

Options

Logout

Welcome back Swapnil Sapre

Given Exams

0

Total Students

3

Recent Tests

No Records Found!

Figure 7.8: Student's Dashboard

This is the student dashboard that is given to user when they log in into the system. It has 2 options present which includes

1. Attempting the test: The teacher needs to share the exam ID and the password to the students.
2. Checking the results after the test is completed. If the teacher has set the option to check the results after completing the test then only it is possible for students to check.

#### a) Exam Instructions

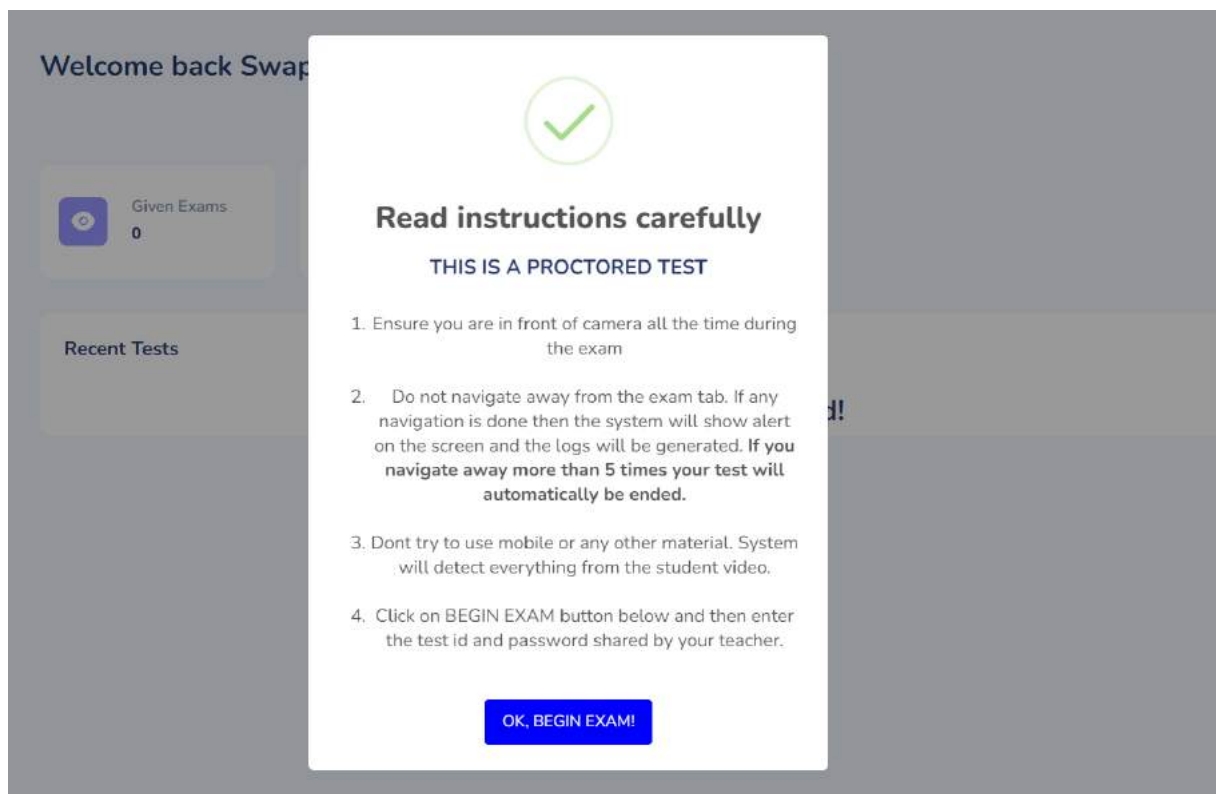


Figure 7.9: Exam Instructions

When the student clicks on attempt test button, these are the instruction that are displayed to make them aware about the exam process. Once the student has finished reading the details they can click on 'OK BEGIN EXAM' button to redirect to a page in which they will be required to enter test ID and the password.

## b) Exam Window

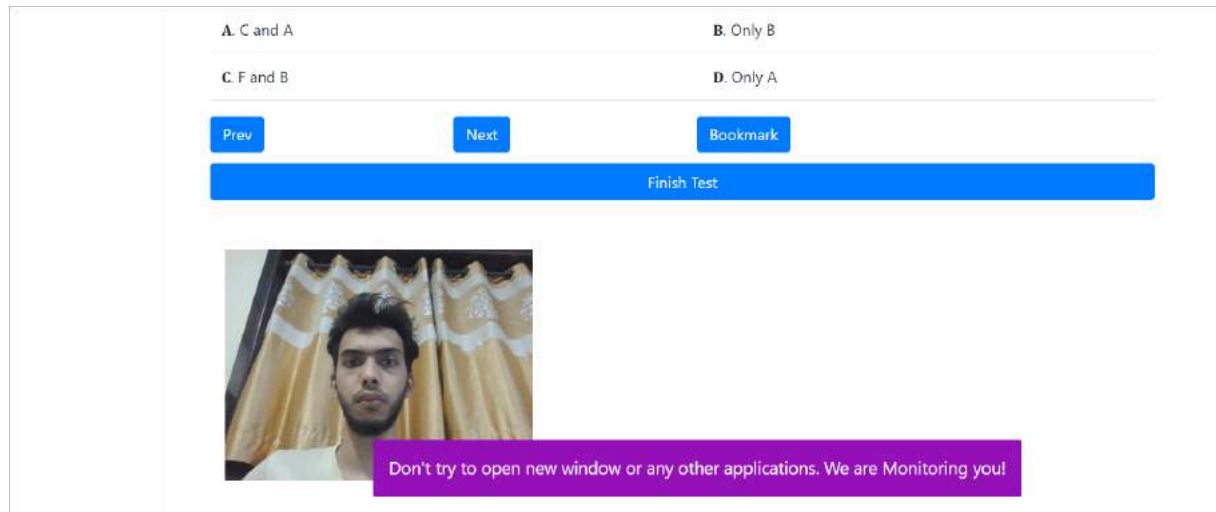


Figure 7.10: Exam Window

This is the examination window in which student will be appearing for the test.

1. It displays the questions in the form of navigation grid at the top. The student can navigate from one question to another by just clicking on the button.
2. then displays the actual question with description and the marks at the bottom
3. Below the question are the options out of which one option is correct. The student needs to select the option and then click the 'Next' button. By clicking 'Next' button the answer gets saved and it moves to the next question.
4. There is also option of Finish Test that student can click once they finish with all the questions.
5. At the last there is the student window streamed from their device's webcam. This is the live environment from which the images are captured and analyzed to generate the logs on the teacher's dashboard.

### c) Student Results

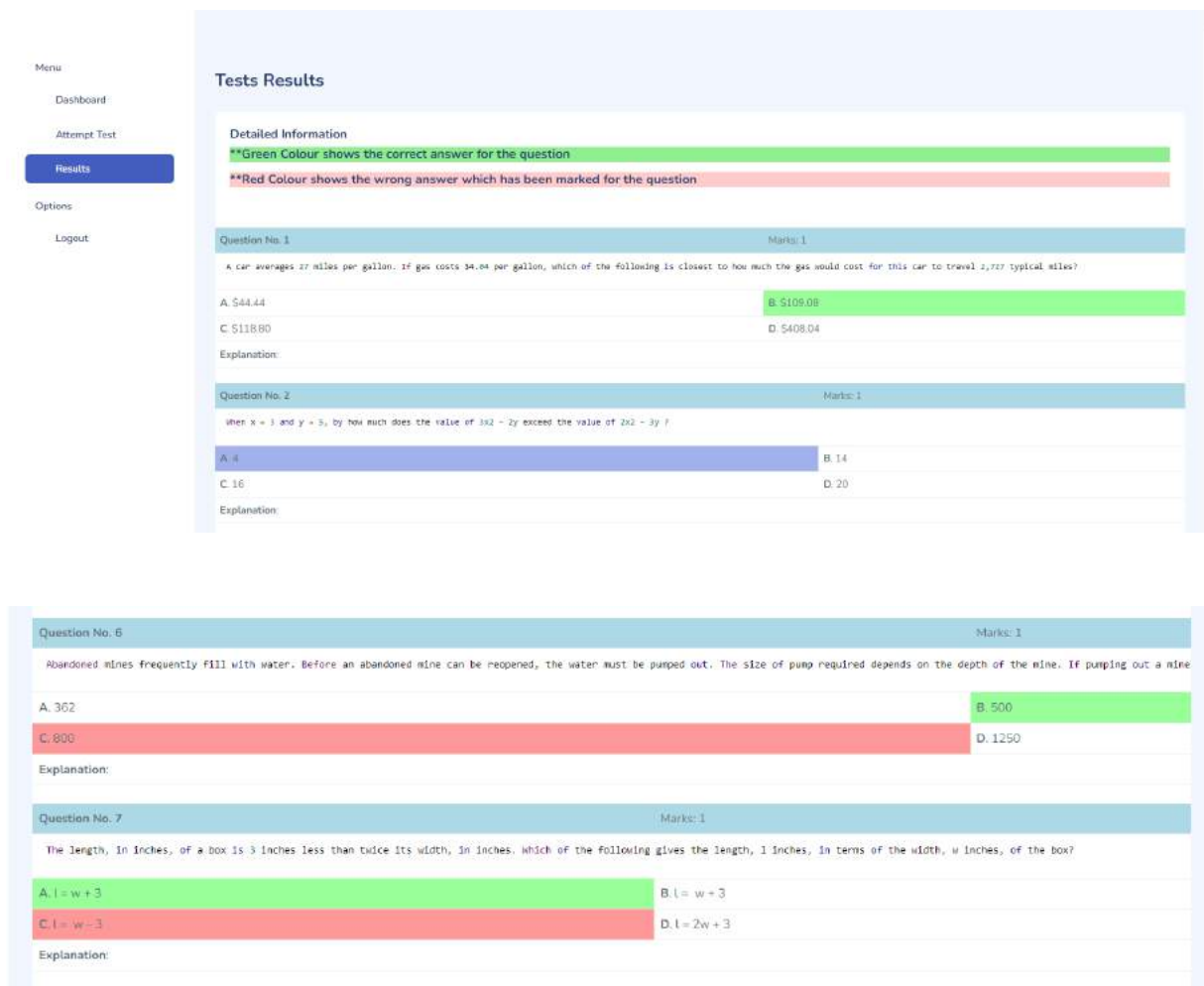


Figure 7.11: Student Results

After the student has attempted the test, they can check their marks and in detailed review of their attempt. The attempt shows:

1. Green color shows that option denotes correct answer for the question.
2. Red color shows the wrong answer which has been marked for the question.
3. Blue color shows that question has been answered by the student.

#### d) Log Generation

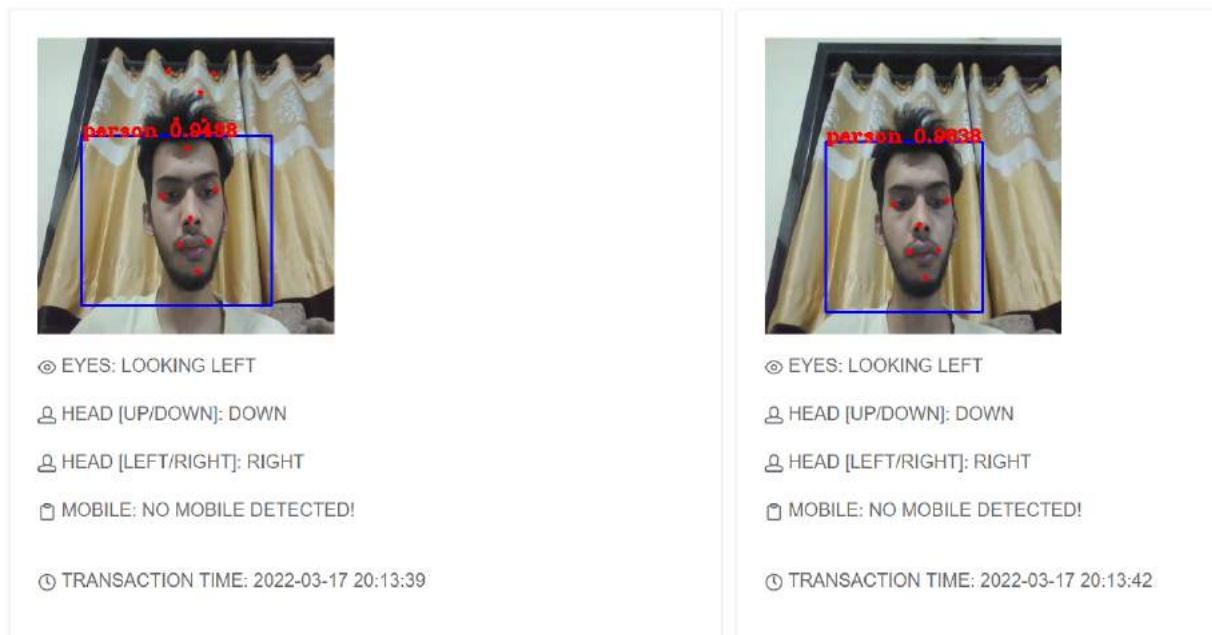


Figure 7.12: Logs generation on teacher's dashboard

The logs are displayed after collecting the images from student's live environment. The logs contain information about where the student is looking (left, right, up, down), the head movement and whether any person or mobile detections are found along with the time and date when that event has occurred.

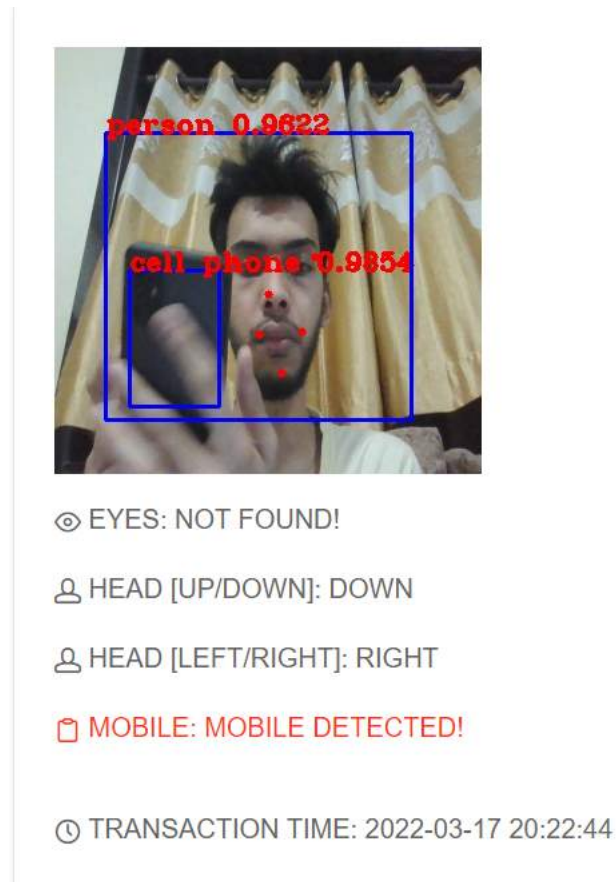


Figure 7.13: Detection of mobile

In the above figure the student is using mobile and it is detected by the system which displays it in logs.

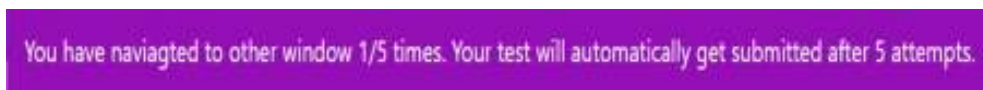


Figure 7.14: Alert on tab switching

As the student has navigated from the exam window, the system detected that event and displayed the alert message regarding the total attempts done and how many maximum attempts are left after which the test will be submitted automatically.



WINDOW LOGS of melodic-mandrill of swapsap09		
NAME	WINDOW EVENT	TIME
Swapnil Sapre	WINDOW EVENT WAS OCCURRED!	2022-03-17 20:14:23
Swapnil Sapre	WINDOW EVENT WAS OCCURRED!	2022-03-17 20:14:34
Swapnil Sapre	WINDOW EVENT WAS OCCURRED!	2022-03-17 20:14:53
Swapnil Sapre	WINDOW EVENT WAS OCCURRED!	2022-03-17 20:15:01
Swapnil Sapre	WINDOW EVENT WAS OCCURRED!	2022-03-17 20:15:08
Swapnil Sapre	WINDOW EVENT WAS OCCURRED!	2022-03-17 20:15:16
Swapnil Sapre	WINDOW EVENT WAS OCCURRED!	2022-03-17 20:15:18
Swapnil Sapre	WINDOW EVENT WAS OCCURRED!	2022-03-17 20:15:27
Swapnil Sapre	WINDOW EVENT WAS OCCURRED!	2022-03-17 20:15:28
Swapnil Sapre	WINDOW EVENT WAS OCCURRED!	2022-03-17 20:15:30
Swapnil Sapre	WINDOW EVENT WAS OCCURRED!	2022-03-17 20:15:56

Figure 7.15: Tab switching activity

The tab navigation events which the student does also gets captured in the form of logs and are displayed on the teacher's dashboard. It contains the names of all those students who have done tab switching along with the time on which it occurred for a particular test.

# Chapter 8

## Conclusions and Future Scope

In our system, face detection is achieved by first capturing the face snapshot of the user and then verifying it with the stored images in the database. The system is also capable of performing student tracking throughout the examination. As the input facial image is captured in real-time during examinations there is a variance in image quality and the illumination of the face. On this aspect, the YOLO algorithm has provided very accurate results by detecting the objects in the candidate's live environment. The designed system is also able to automate the attendance process by generating the data of the students who have appeared for the exam. Our examination framework currently handles only objective-based questions. Hence the future scope of this work is to make the entire system a part of the e-learning framework and also provide support for practical as well as subjective-based examinations.

# Bibliography

- [1] Walaa M. Abd-Elhafiez, Mohamed Heshmat, and Seham Elaw. Efficient Method for Face Recognition and Its Role in Supporting E-Learning Systems. 2015. doi: 10.1109/ECONF.2015.21
- [2] Wang Aimin and Wang Jipeng. Design and Implementation of Web-Based Intelligent Examination System. 2009. doi: 10.1109/WCSE.2009.77.
- [3] Hadian S. G. Asep and Yoanes Bandung. A Design of Continuous User Verification for Online Exam Proctoring on M- Learning. 2019. doi: 10.1109/ICEEI47359.2019.8988786.
- [4] Yousef Atoum et al. Automated Online Exam Proctoring. 2017. doi: 10.1109/TMM.2017.2656064.
- [5] Samuel S. Chua et al. Online Examination System with Cheating Prevention Using Question Bank Randomization and Tab Locking. 2019.
- [6] Radhika C. Damale and Bazeshree. V Pathak. Face Recognition Based Attendance System Using Machine Learning Algorithms. 2018. doi: 10.1109/ICCONS.2018.8662938.
- [7] Jegatha Deborah L et al. Secure Online Examination System for e-learning. 2019. doi: 10.1109/CCECE43985.2019. 9052408.
- [8] Ondrej Kainz et al. Enhancing Attention through the Eye Tracking. 2020. doi: 10.1109/ICETA51985.2020.9379229.
- [9] Mansi Mahendru and Sanjay Kumar Dubey. Real Time Object Detection with Audio Feedback using Yolo vs. YOLOv3. 2021. doi: 10.1109/Confluence51648.2021.9377064.
- [10] Arief Agus Sukmandhani and Indraajani Sutedja. Face Recognition Method for Online Exams. 2019. doi: 10.1109/ICIMTech. 2019.8843831.
- [11] How to use Flask-Python : <https://www.freecodecamp.org/news/how-to-build-a-web-application-using-flask-and-deploy-it-to-the-cloud-3551c985e492/> Accessed on 18/10/22 at 11:00 AM.
- [12] Deep Learning based Object Detection using YOLOv3 with OpenCV : <https://learnopencv.com/deep-learning-based-object-detection-using-yolov3-with-opencv-python-c/> Accessed on 20/10/22 Accessed on 19/10/22 at 06:15 PM.
- [13] Eliminate physical exams with auto proctoring: [https://www.eklavya.in/online-exam-remote-proctoring.aspx#Eliminate\\_Physical\\_Exams](https://www.eklavya.in/online-exam-remote-proctoring.aspx#Eliminate_Physical_Exams) Accessed on 22/10/22 at 05:00 PM.

- [14] Automating Online Proctoring Using AI: <https://towardsdatascience.com/automating-online-proctoring-using-ai-e429086743c8> Accessed on 22/10/22 at 05:15 PM.
- [15] Object Detection using Tensorflow: [https://www.tensorflow.org/hub/tutorials/object\\_detection](https://www.tensorflow.org/hub/tutorials/object_detection) Accessed on 01/11/22 at 07:30 PM.
- [16] Real time object detection with deep learning and OpenCV: <https://pyimagesearch.com/2017/09/18/real-time-object-detection-with-deep-learning-and-opencv/> Accessed on 09/11/22 at 10:00 AM.
- [17] Deep Face Recognition in Python: <https://medium.com/nerd-for-tech/deep-face-recognition-in-python-41522fb47028> Accessed on 09/11/22 at 06:00 PM.
- [18] Themewagon HTML Templates for Web application: <https://themewagon.com/themes/bootstrap-4-html5-admin-dashboard-template-regal/> Accessed on 10/10/22 at 06:00 PM.

# Appendices

## Appendix-A: Installation and Running the project

1. Download xampp server required for running the system on localhost from <https://www.apachefriends.org/index.html>.

1.1) **Run .exe file:** Once the software bundle has been downloaded, you can start the installation by double clicking on the file with the ending .exe.

1.2) One may get a warning message saying User Account Control (UAC) can interfere with the XAMPP installation because it limits writing access to the C: drive. Just click on OK button and proceed ahead.

1.3) **Start the setup wizard:** Next the start screen of the XAMPP setup wizard should appear automatically. Click on 'Next' to configure the installation settings.

1.4) **Choose software components:** Under 'Select Components', you have the option to exclude individual components of the XAMPP software bundle from the installation. But for a full local test server, it is recommended to install the standard setup and all available components. Then click 'Next'.

1.5) **Choose the installation directory:** In this next step, you can choose where you'd like the XAMPP software. If you keep the standard setup, then a folder with the name XAMPP will be created under C: drive. After you've chosen a location, click 'Next'.

1.6) **Start the installation process:** The setup wizard will unpack and install the selected components and save them to the designated directory. This process can take several minutes

1.7) Once all the components are unpacked and installed, you can close the setup wizard by clicking on 'Finish'.

1.8) After installation is completed open the Xampp Control panel and start the services of apache and mysql.

2. It also requires anaconda terminal to be installed. It can be done from <https://www.anaconda.com/products/distribution>.

3. Now we require the YOLO v3 weights. It can be downloaded from: <https://pjreddie.com/darknet/yolo/>

4. Go to anaconda terminal. We need to import the python environment into our system. from sample file (project-env.yml). For this in the terminal type the command:

**conda env create - -name testenv - - filename=project-env.yml.**

This will create a python (project) environment named 'testenv'.

5. Next place all the files such as code file, requirements file and static templates of the application in any desired folder for example Desktop/BEproject. Here BEproject is name of the folder created on Desktop of the system.

6. Now navigate into that folder. For this in the same anaconda terminal type following command:

**cd Desktop/BEproject**

7. We need to activate the environment which we have created before in order to start the application. For this type command:

**activate testenv**

Our starting file for the application is app.py which contains the driver code. So to execute enter:

**python app.py**

The python program will execute and we will get a localhost url of <http://127.0.0.1:5000> Now the application is running which can be accessed in browser.

# Publication

Paper entitled “**AI-ML Based Smart Online Examination Framework**” is presented at “**3rd International Conference on Deep Learning, Artificial Intelligence**” and published in “**Robotics (ICDLAIR) 2021 / Journal Name: Springer LNNS**” by “**Swapnil Sapre, Kunal Shinde, Keval Shetta**”.