3. Implementation

3.1 Implementation Functionality

- **User Authentication:** Implement user authentication mechanisms for proctors, students, and administrators to ensure secure access to the application.
- **Dashboard for Proctors:** Develop a dashboard interface for proctors to manage exams, monitor live feeds, communicate with students, and intervene when necessary.
- **Dashboard for Students:** Create a user-friendly dashboard for students to access exams, view instructions, submit answers, and communicate with proctors.
- **AI-Powered Suspicious Activity Detection:** Integrate AI models using OpenCV and TensorFlow/PyTorch for real-time monitoring of exam sessions, detecting suspicious activities such as looking away from the screen or multiple faces in the camera view.
- **Live Camera Feed Monitoring:** Implement functionality to stream live camera feeds from students' devices to proctors' dashboards for real-time monitoring and supervision.
- **Real-Time Communication:** Enable instant messaging functionality between proctors and students within the exam interface using WebSockets or Socket.io for quick assistance, clarification of instructions, or issue resolution.
- **Exam Management:** Develop features for creating, scheduling, and managing exams, including setting time limits, configuring exam settings, and generating unique exam links for students.
- **Submission and Grading:** Implement mechanisms for students to submit exam answers securely and for proctors to grade and provide feedback on submitted exams.
- **Security Measures:** Implement robust security measures to protect user data, prevent unauthorized access, and ensure the integrity of exam sessions.
- **Reporting and Analytics:** Develop reporting and analytics features to track exam performance, identify trends, and generate insights for administrators and educators.
- **User Profile Management:** Enable users to create and manage their profiles, update personal information, and configure notification preferences.

3.2 Results and Reports

Result Generation: Upon completion of an exam, the system automatically generates results based on the student's responses and performance. Result generation includes calculating scores, identifying correct and incorrect answers, and determining overall performance metrics.

Real-time Feedback: During the exam, students may receive real-time feedback on their performance, such as notifications for unanswered questions or time warnings. Proctors can also provide immediate feedback or instructions to students through the chat functionality.

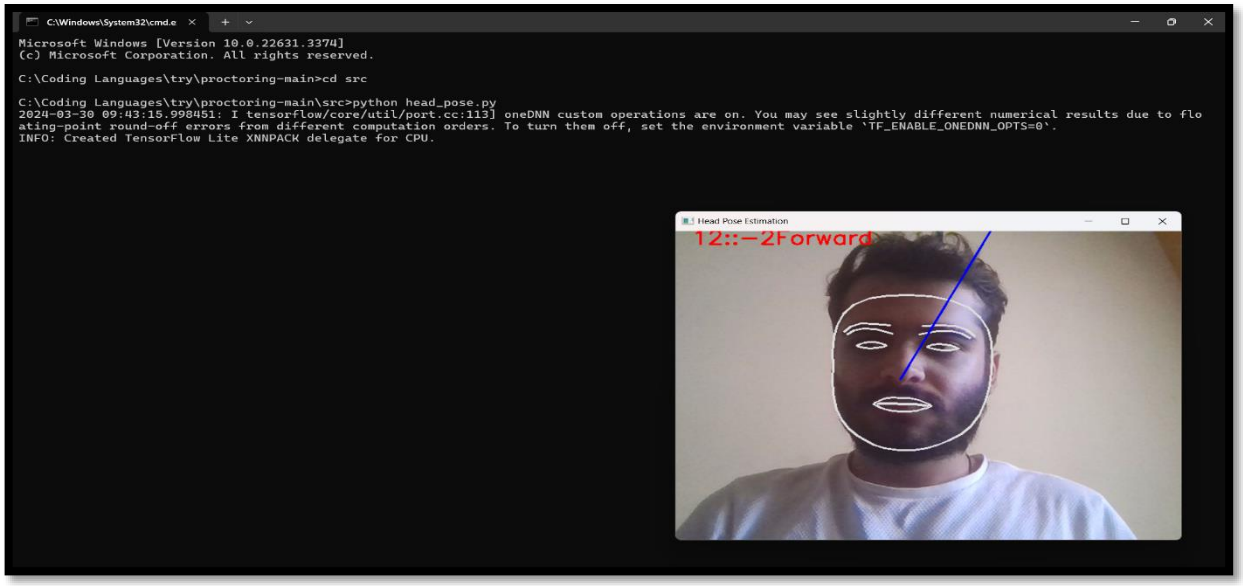
Customization Options: Administrators have the ability to customize the format and content of reports based on institutional requirements or preferences.

Accessibility and Security: Exam results and reports are securely stored and accessible only to authorized users, ensuring data privacy and confidentiality.

3.3 Snapshots

Backend Snapshots:

- **Head Position Detection Model**

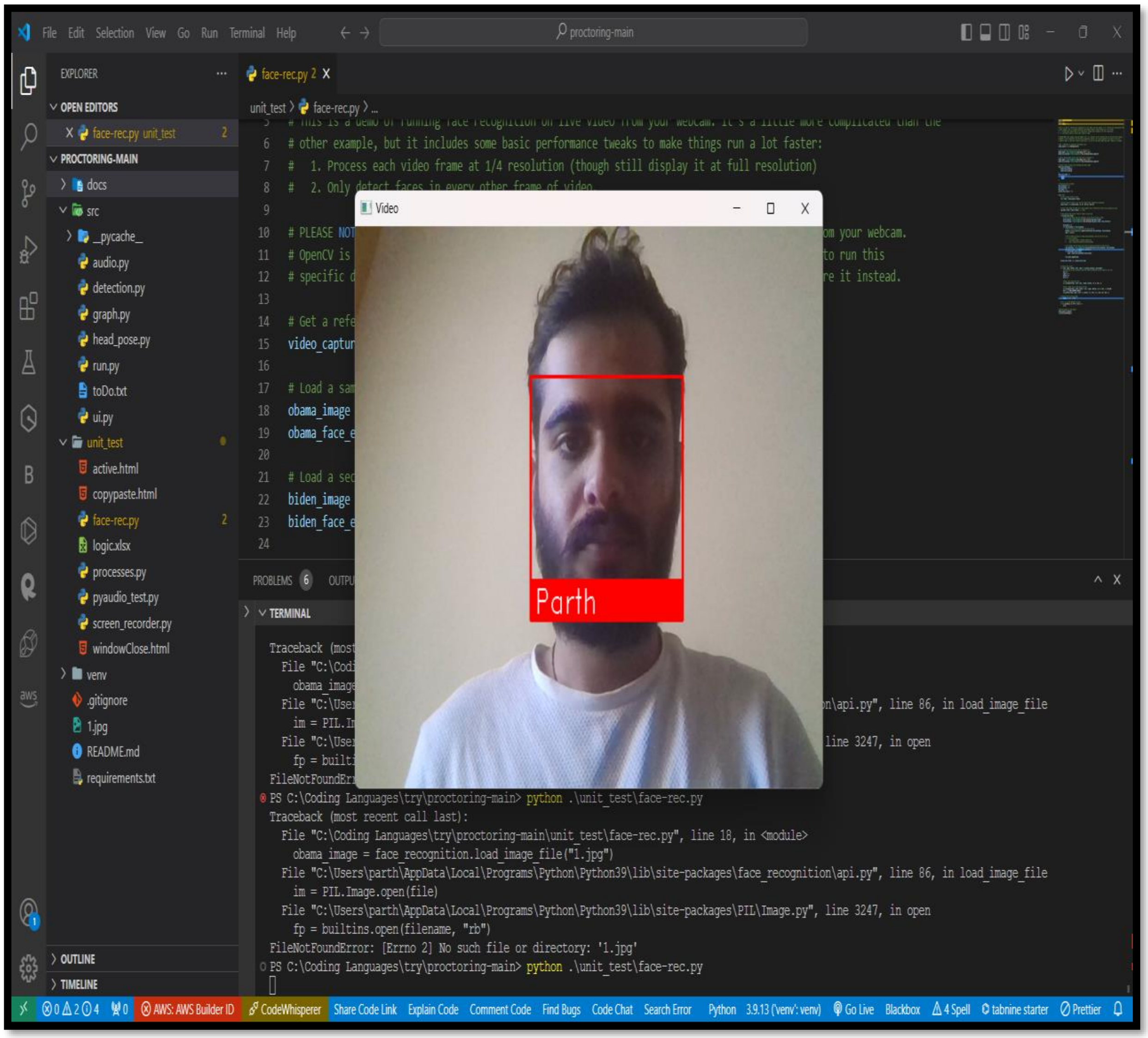


```
Microsoft Windows [Version 10.0.22631.3374]
(c) Microsoft Corporation. All rights reserved.

C:\Coding Languages\try\proctoring-main>cd src

C:\Coding Languages\try\proctoring-main\src>python head_pose.py
2024-03-30 09:43:15.998451: I tensorflow/core/util/port.cc:113] oneDNN custom operations are on. You may see slightly different numerical results due to flo
ating-point round-off errors from different computation orders. To turn them off, set the environment variable `TF_ENABLE_ONEDNN_OPTS=0`.
INFO: Created TensorFlow Lite XNNPACK delegate for CPU.
```

Fig. 3.3 1 Head Position Model

The head position detection model utilizes advanced computer vision algorithms to accurately determine the orientation of human heads within images or video frames. By analyzing facial landmarks and geometric features, it precisely identifies head angles, aiding in applications such as gaze estimation, facial recognition, and augmented reality.

- **User Detection Model**



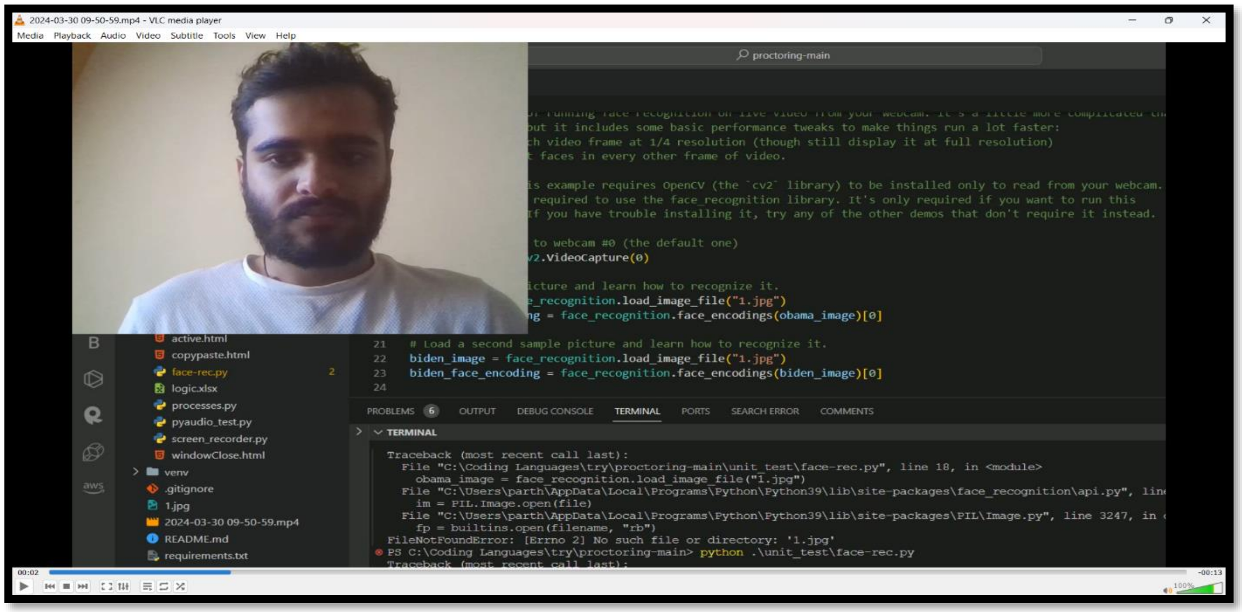
```
# This is a demo of running face recognition on live video from your webcam. It's a little more complicated than the
# other example, but it includes some basic performance tweaks to make things run a lot faster:
#   1. Process each video frame at 1/4 resolution (though still display it at full resolution)
#   2. Only detect faces in every other frame of video.

# PLEASE NOT                                                              om your webcam.
# OpenCV is                                                               to run this
# specific d                                                              re it instead.

# Get a refe
video_captur

# Load a sam
obama_image
obama_face_e

# Load a sec
biden_image
biden_face_e
```

```
Traceback (most
  File "C:\Codi
    obama_image
  File "C:\Use                                                            n\api.py", line 86, in load_image_file
    im = PIL.Im
  File "C:\Use                                                            line 3247, in open
    fp = builti
FileNotFoundErr
PS C:\Coding Languages\try\proctoring-main> python .\unit_test\face-rec.py
Traceback (most recent call last):
  File "C:\Coding Languages\try\proctoring-main\unit_test\face-rec.py", line 18, in <module>
    obama_image = face_recognition.load_image_file("1.jpg")
  File "C:\Users\parth\AppData\Local\Programs\Python\Python39\lib\site-packages\face_recognition\api.py", line 86, in load_image_file
    im = PIL.Image.open(file)
  File "C:\Users\parth\AppData\Local\Programs\Python\Python39\lib\site-packages\PIL\Image.py", line 3247, in open
    fp = builtins.open(filename, "rb")
FileNotFoundError: [Errno 2] No such file or directory: '1.jpg'
PS C:\Coding Languages\try\proctoring-main> python .\unit_test\face-rec.py
```

Fig. 3.3 2 Audio Detection Model

The audio detection model employs machine learning techniques to identify and classify various sounds within audio recordings or live streams. By analyzing spectral features and temporal patterns, it can distinguish between different audio events, enabling applications like speech recognition, environmental monitoring, and anomaly detection in audio streams.

- **Screen Recording Model**



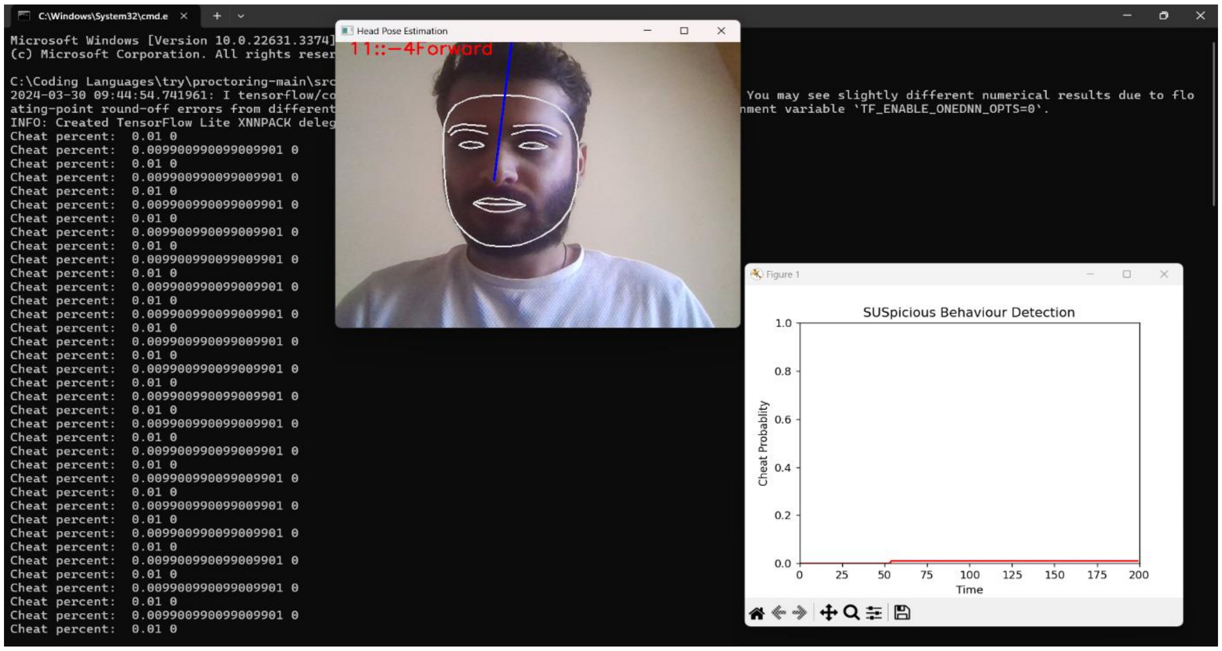
```
but it includes some basic performance tweaks to make things run a lot faster:
ch video frame at 1/4 resolution (though still display it at full resolution)
t faces in every other frame of video.

is example requires OpenCV (the `cv2` library) to be installed only to read from your webcam.
 required to use the face_recognition library. It's only required if you want to run this
If you have trouble installing it, try any of the other demos that don't require it instead.

 to webcam #0 (the default one)
v2.VideoCapture(0)

icture and learn how to recognize it.
e_recognition.load_image_file("1.jpg")
ng = face_recognition.face_encodings(obama_image)[0]

# Load a second sample picture and learn how to recognize it.
biden_image = face_recognition.load_image_file("1.jpg")
biden_face_encoding = face_recognition.face_encodings(biden_image)[0]
```

```
TERMINAL
Traceback (most recent call last):
  File "C:\Coding Languages\try\proctoring-main\unit_test\face-rec.py", line 18, in <module>
    obama_image = face_recognition.load_image_file("1.jpg")
  File "C:\Users\parth\AppData\Local\Programs\Python\Python39\lib\site-packages\face_recognition\api.py", line
    im = PIL.Image.open(file)
  File "C:\Users\parth\AppData\Local\Programs\Python\Python39\lib\site-packages\PIL\Image.py", line 3247, in
    fp = builtins.open(filename, "rb")
FileNotFoundError: [Errno 2] No such file or directory: '1.jpg'
PS C:\Coding Languages\try\proctoring-main> python .\unit_test\face-rec.py
Traceback (most recent call last):
```

Fig. 3.3 3 Screen Recording Model

The screen recording model leverages deep learning architectures to capture and encode visual information displayed on computer screens in real-time. By analyzing pixel data and frame sequences, it accurately records screen activities, facilitating applications such as tutorial creation, software testing, and user behavior analysis in digital environments.

- **Malpractice Detection Model**

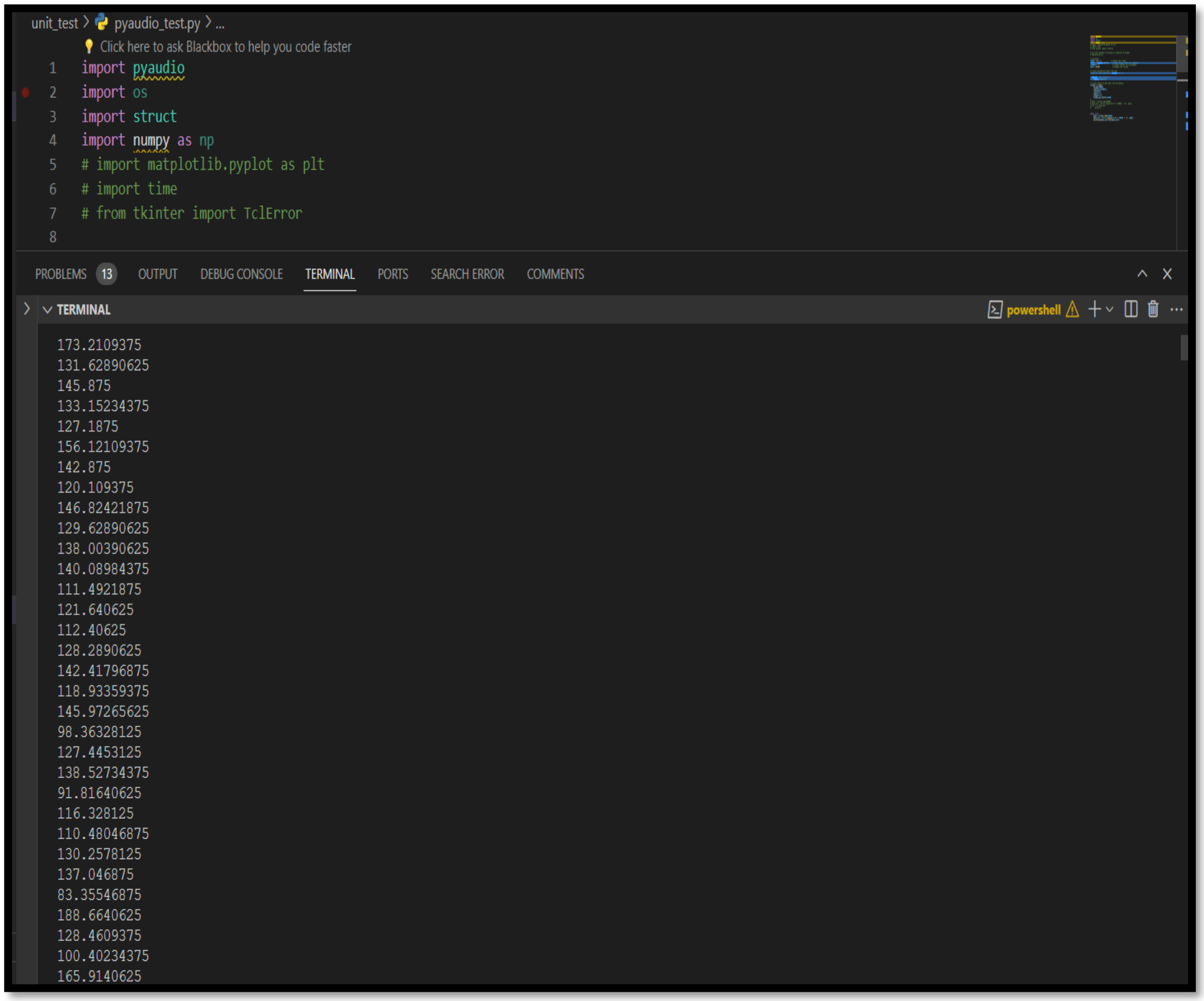


```
Microsoft Windows [Version 10.0.22631.3374]
(c) Microsoft Corporation. All rights reser

C:\Coding Languages\try\proctoring-main\src
2024-03-30 09:44:54.741961: I tensorflow/co                                          You may see slightly different numerical results due to flo
ating-point round-off errors from different                                          nment variable `TF_ENABLE_ONEDNN_OPTS=0`.
INFO: Created TensorFlow Lite XNNPACK deleg
Cheat percent:  0.01 0
Cheat percent:  0.009900990099009901 0
Cheat percent:  0.01 0
Cheat percent:  0.009900990099009901 0
Cheat percent:  0.01 0
Cheat percent:  0.009900990099009901 0
Cheat percent:  0.01 0
Cheat percent:  0.009900990099009901 0
Cheat percent:  0.01 0
Cheat percent:  0.009900990099009901 0
Cheat percent:  0.01 0
Cheat percent:  0.009900990099009901 0
Cheat percent:  0.01 0
Cheat percent:  0.009900990099009901 0
Cheat percent:  0.01 0
Cheat percent:  0.009900990099009901 0
Cheat percent:  0.01 0
Cheat percent:  0.009900990099009901 0
Cheat percent:  0.01 0
Cheat percent:  0.009900990099009901 0
Cheat percent:  0.01 0
Cheat percent:  0.009900990099009901 0
Cheat percent:  0.01 0
Cheat percent:  0.009900990099009901 0
Cheat percent:  0.01 0
Cheat percent:  0.009900990099009901 0
Cheat percent:  0.01 0
Cheat percent:  0.009900990099009901 0
Cheat percent:  0.01 0
Cheat percent:  0.009900990099009901 0
Cheat percent:  0.01 0
Cheat percent:  0.009900990099009901 0
Cheat percent:  0.01 0
Cheat percent:  0.009900990099009901 0
Cheat percent:  0.01 0
Cheat percent:  0.009900990099009901 0
Cheat percent:  0.01 0
```

Fig. 3.3 4 Malpractice Detection Model

The malpractice detection model integrates machine learning algorithms to analyze various data sources, such as medical records and practitioner behavior patterns, to identify potential instances of malpractice or negligence. By detecting anomalies, discrepancies, and deviations from established standards, it aids in early intervention, risk mitigation, and quality assurance within professional settings like healthcare or legal domains.

- **Audio Detection Model**


unit_test > pyaudio_test.py > ...
Click here to ask Blackbox to help you code faster
PROBLEMS 13
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
SEARCH ERROR
COMMENTS
TERMINAL
powershell

```
import pyaudio
import os
import struct
import numpy as np
# import matplotlib.pyplot as plt
# import time
# from tkinter import TclError

```

```
173.2109375
131.62890625
145.875
133.15234375
127.1875
156.12109375
142.875
120.109375
146.82421875
129.62890625
138.00390625
140.08984375
111.4921875
121.640625
112.40625
128.2890625
142.41796875
118.93359375
145.97265625
98.36328125
127.4453125
138.52734375
91.81640625
116.328125
110.48046875
130.2578125
137.046875
83.35546875
188.6640625
128.4609375
100.40234375
165.9140625
```

Fig. 3.3 5 Audio Detection Model

The audio detection model employs deep learning techniques to identify and categorize different types of sounds within audio recordings or live streams. By analyzing spectral features, temporal patterns, and frequency characteristics, it can distinguish between various audio events such as speech, music, environmental noise, and anomalies, enabling applications like speech recognition, surveillance, and acoustic monitoring.

Frontend Snapshots:

- **Home Page**

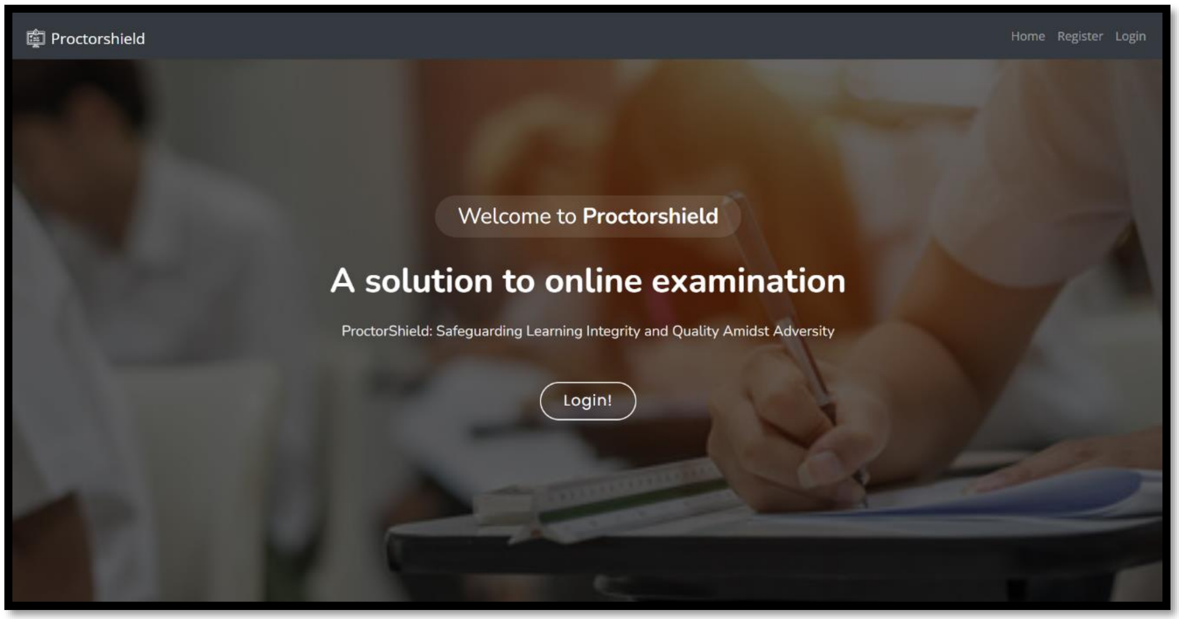

Proctorshield
Home Register Login
Welcome to Proctorshield
A solution to online examination
ProctorShield: Safeguarding Learning Integrity and Quality Amidst Adversity
Login!

Fig. 3.3 6 Home page

The frontend home page presents a visually captivating interface that welcomes users with dynamic content and intuitive navigation, guiding them seamlessly through the website's offerings. Through strategically placed elements and compelling visuals, it creates an immersive experience that encourages exploration and engagement from visitors.

- **Login**

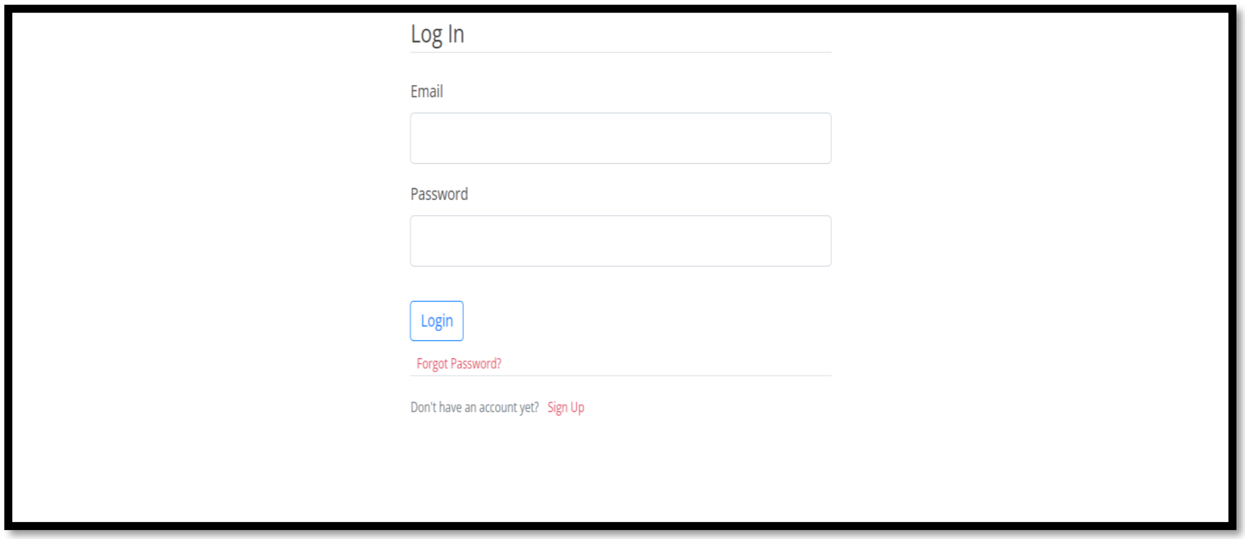
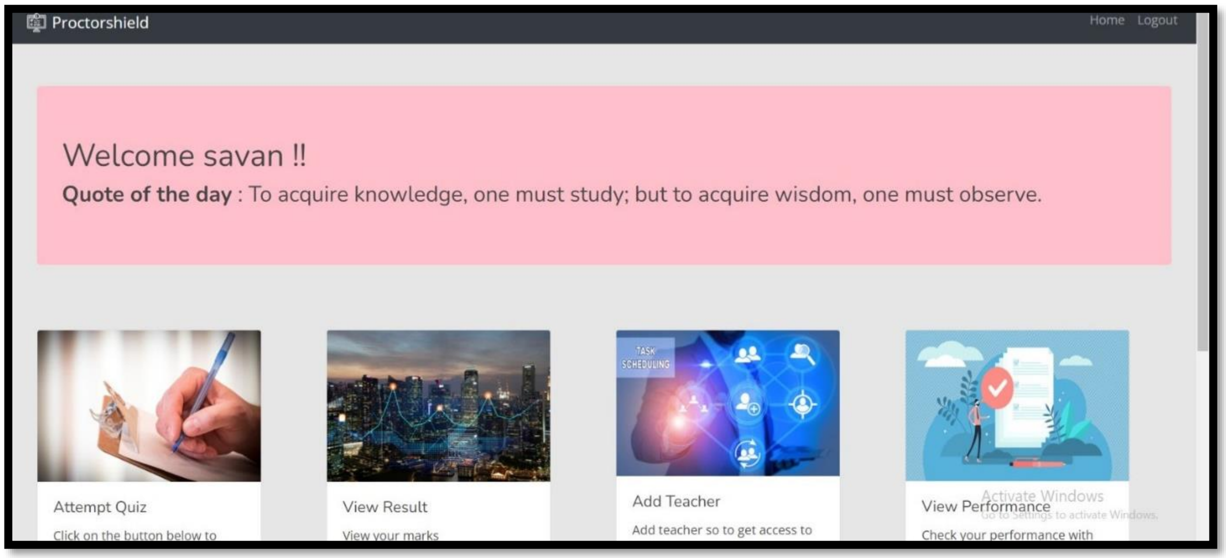


Fig. 3.3.7 Login Page

The login page serves as the entry point for users to access their accounts, requiring credentials such as usernames and passwords for authentication. Designed with user-friendliness and security in mind, it facilitates seamless access while implementing robust measures to protect user information.

- **Dashboard**



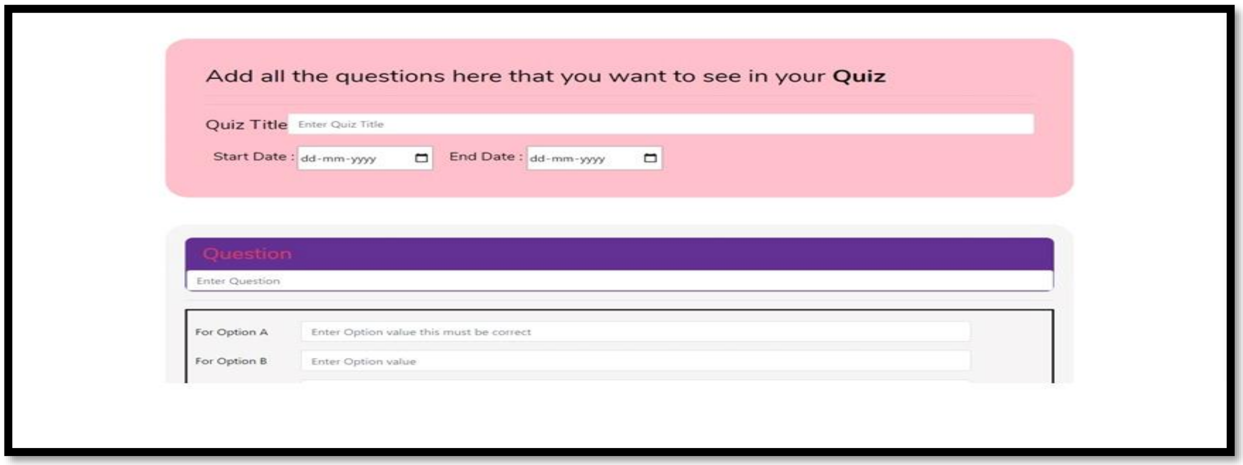
Welcome savan !!
Quote of the day : To acquire knowledge, one must study; but to acquire wisdom, one must observe.

Click on the button below to
View your marks
Add teacher so to get access to
Check your performance with

Fig. 3.3.8 DashBoard

The login page serves as the entry point for users to access their accounts, requiring credentials such as usernames and passwords for authentication. Designed with user-friendliness and security in mind, it facilitates seamless access while implementing robust measures to protect user information.

- **Quiz**

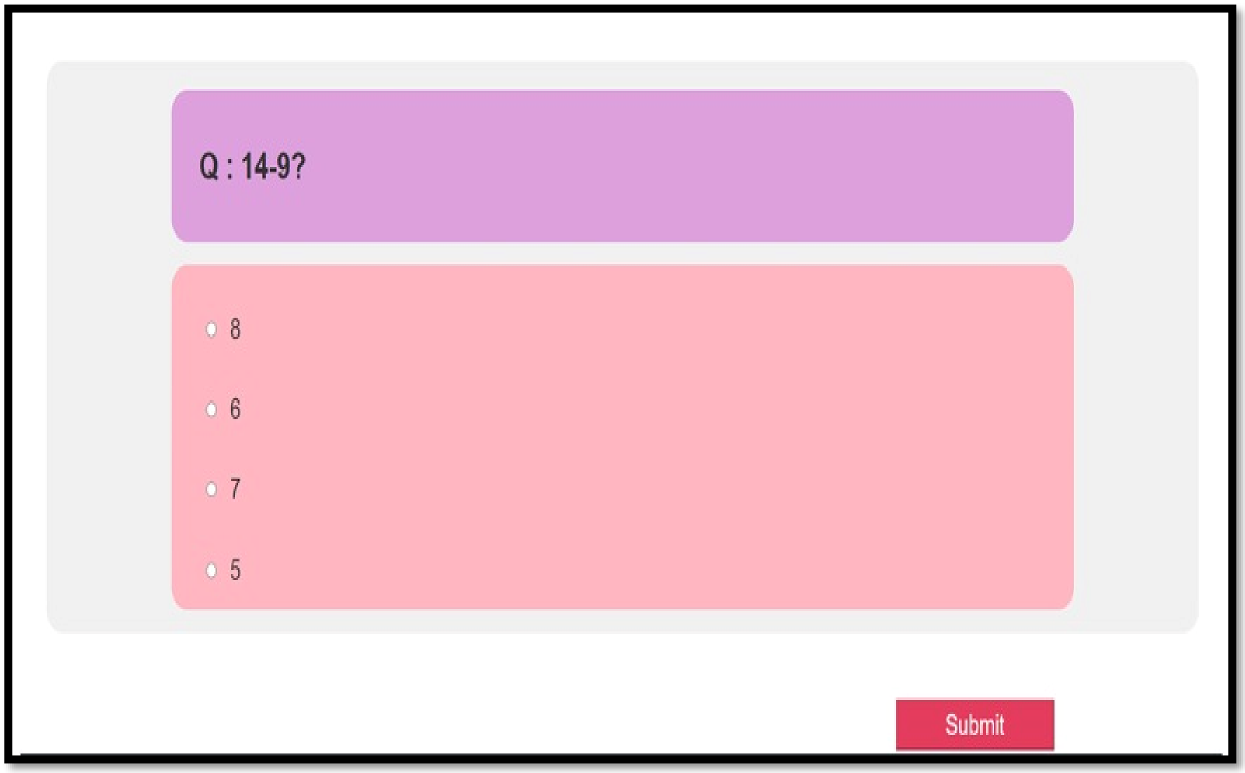

Add all the questions here that you want to see in your **Quiz**

Quiz Title: Enter Quiz Title
Start Date : dd-mm-yyyy End Date : dd-mm-yyyy

Question
Enter Question

For Option A: Enter Option value this must be correct
For Option B: Enter Option value

Fig. 3.3.9 Create Quiz


Q : 14-9?

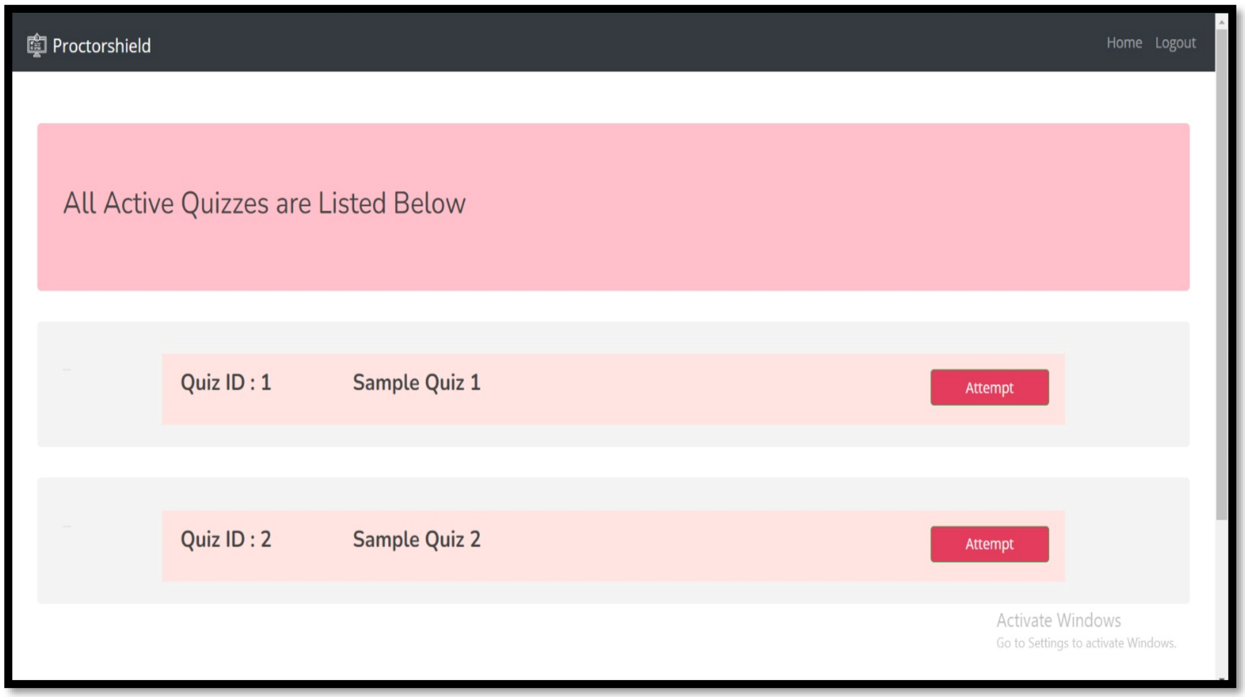
8

6

7

5

Fig. 3.3.10 Quiz

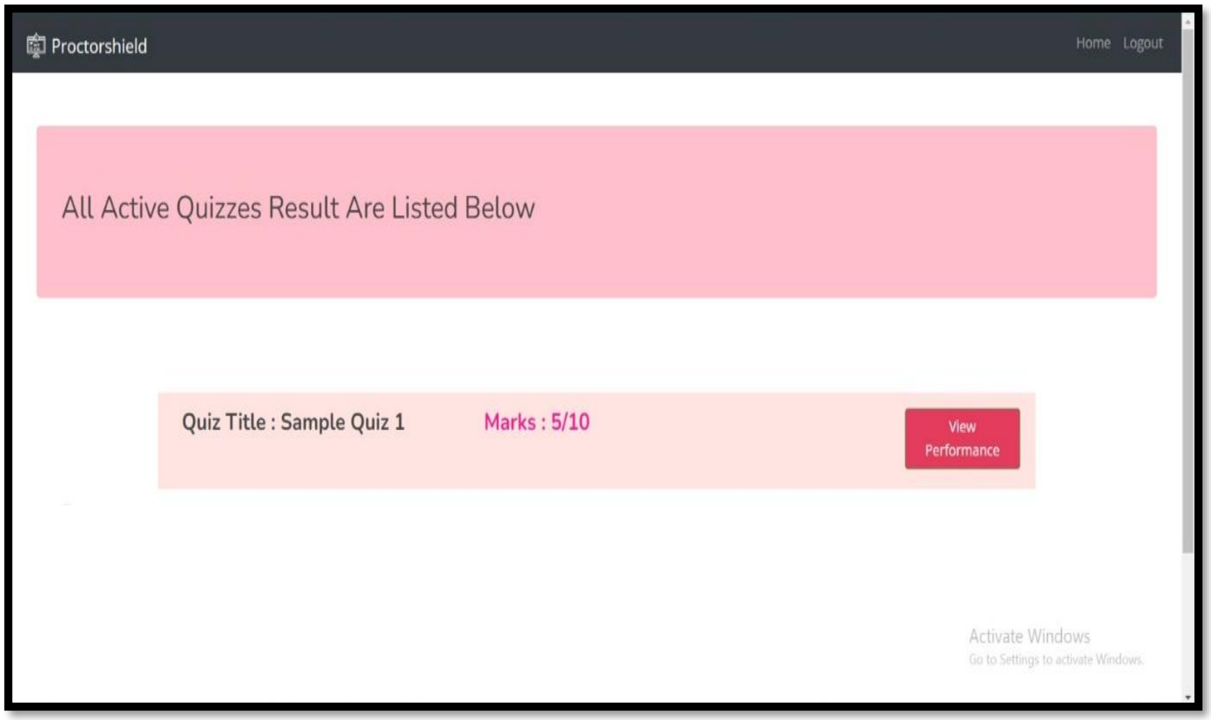


All Active Quizzes are Listed Below

Quiz ID : 1 Sample Quiz 1

Quiz ID : 2 Sample Quiz 2

Fig. 3.3.10 List of Quiz



Proctorshield

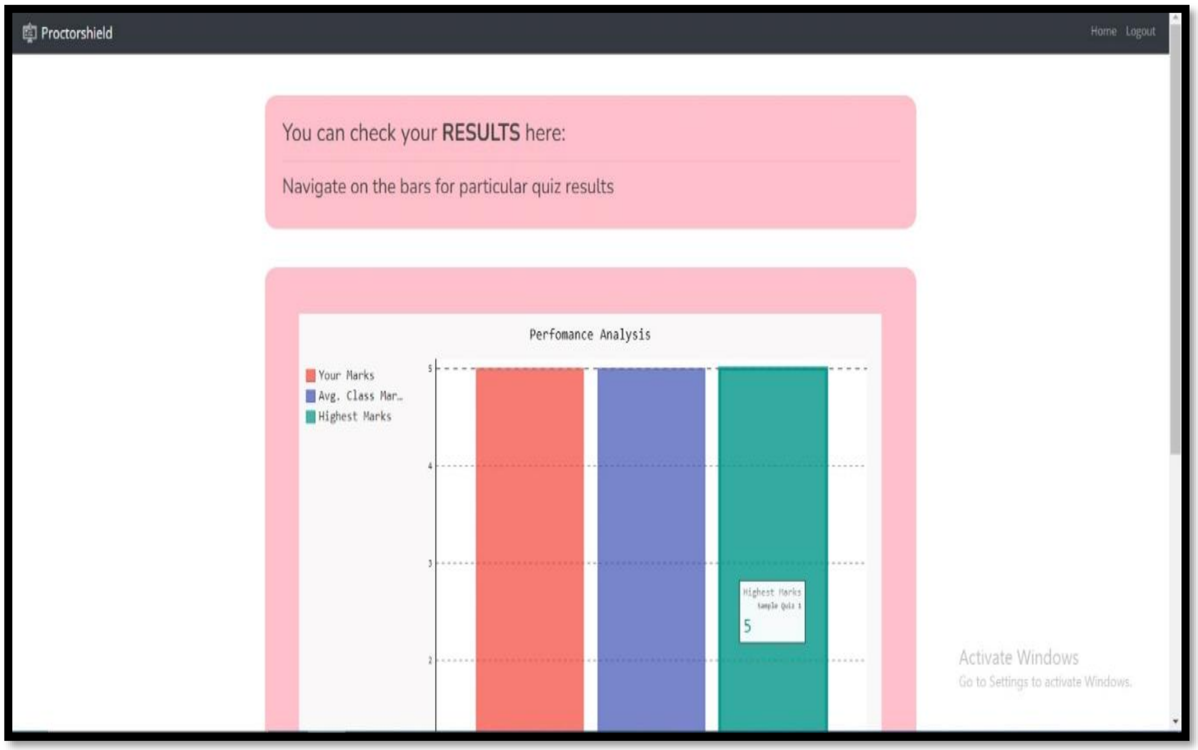
Home Logout

All Active Quizzes Result Are Listed Below

Quiz Title : Sample Quiz 1 Marks : 5/10

View Performance

Fig. 3.3.10 Result of Quiz



Proctorshield

Home Logout

You can check your **RESULTS** here:

Navigate on the bars for particular quiz results

Fig. 3.3.10 Performance Analysis

3.4 Testing and verification

Unit Testing: Individual components of the application, including backend logic, frontend interface elements, and AI algorithms, will undergo unit testing. Test cases will be designed to validate the behavior of each unit in isolation, ensuring that it functions as intended.

Integration Testing: Integrated modules and subsystems will be tested together to verify their interactions and compatibility. This phase will focus on testing the integration of backend and frontend components, communication channels, and database functionality.

System Testing: The entire system will be tested as a whole to evaluate its overall functionality, performance, and usability. Test scenarios will be designed to simulate real-world usage scenarios, including exam sessions with multiple users and varying conditions.

Security Testing: Security testing will be conducted to identify and mitigate potential vulnerabilities and threats to the application.

Usability Testing: Usability testing will be performed to assess the user experience and interface design of the application. Feedback from users will be collected to identify any usability issues or areas for improvement.

Performance Testing: Performance testing will evaluate the application's responsiveness, scalability, and reliability under different load conditions. Stress testing and load testing will be conducted to assess the application's ability to handle concurrent user activity and maintain performance.