***A Major Project******Report***

**ON**

**Artificial Intelligence based Interviewing and Proctoring System**

***Submitted in partial fulfillment for the Degree of B.Tech***

**in**

**Artificial Intelligence**

***By***

*RAYABARAPU PRABHATH KUMAR (19911A3543)*

*PASUPUNURI RAKESH ROHAN (19911A3540)*

*BOINI ARUN KUMAR(19911A3506)*

**Under the guidance of**

Ms. K. NIROSHA

Dept of AI



**DEPARTMENT OF ARTIFICIAL INTELLIGENCE**

**VIDYA JYOTHI INSTITUTE OF TECHNOLOGY**

(An Autonomous Institution)

**Accredited by NAAC & NBA, Approved by AICTE New Delhi & Permanently Affiliated to JNTUH**

2022 – 2023

****

***CERTIFICATE***

This is to certify that the project report entitled **Artificial Intelligence based Interviewing and Proctoring System** submitted by **Rayabarapu Prabhath Kumar (19911A3543),**  **Pasupunuri Rakesh Rohan (19911A3540)** and **Boini Arun Kumar (19911A3506)** to Vidya Jyothi Institute of Technology, Hyderabad, in partial fulfillment for the award of the degree of **B.Tech in Artificial Intelligence** a *bonafide* record of project work carried out by us under my supervision. The contents of this report, in full or in parts, have not been submitted to any other Institution or University for the award of any degree.

**Ms. K. Nirosha Dr.A.Obulesh**

**Supervisor HOD**

**Department of Artificial Intelligence Department of Artificial Intelligence**

***DECLARATION***

We declare that this project report titled **Artificial Intelligence based Interviewing and Proctoring System** submitted in partial fulfillment of the degree of **B.Tech in Artificial Intelligence** is a record of original work carried out under the supervision of  **Ms. K. Nirosha** , and has not formed the basis for the award of any other degree or diploma, in this or any other Institution or University. In keeping with the ethical practice in reporting scientific information, due acknowledgments have been made wherever the findings of others have been cited.

Rayabarapu Prabhath Kumar (19911A3543)

Pasupunuri Rakesh Rohan (19911A3540)

Boini Arun Kumar (199711A3506)

***ACKNOWLEDGEMENT***

Any endeavor in the field of development is a person’s intensive activity. A successful project is a fruitful culmination of efforts by many people, some directly involved and some others who have quietly encouraged and supported.

We would like to thank **Ms. K. Nirosha**, Department of Artificial Intelligence who has patiently guided and helped me throughout my project.

We wish to express profound gratitude to **Dr A.Obulesh**, Professor and Head of the Department, Artificial Intelligence, for his continuous encouragement to ensure the successful results in all my endeavors.

We wish to express our heart full thanks to the Principal **Dr. A.Padmaja** Madam, whose support was indispensable to me during the course.

We wish to express our heart full thanks to our Beloved Director  **Dr E. Sai Baba Reddy**, whose support was indispensable to me during the course.

We wish to express our heart full thanks to our **College Management,** for providing all the Infrastructure and facilities during the course.

By

Rayabarapu Prabhath Kumar (19911A3543)

Pasupunuri Rakesh Rohan (19911A3540)

Boini Arun Kumar (19911A3506)

***ABSTRACT***

The use of artificial intelligence (AI) in the hiring process is becoming increasingly common, as it offers a more efficient and objective way to assess candidates. However, current AI-based interview systems often lack the ability to fully evaluate a candidate's fit for a role and can be prone to bias.

Our project aims to address these issues by developing an AI-based interviewing and proctoring system that combines advanced natural language processing with proctoring technologies such as computer vision and machine learning.

The system will be able to conduct virtual interviews with candidates and assess their responses in real time, using a variety of metrics to evaluate their fit for the role. These metrics will include not just the content of the candidate's responses, but also non-verbal cues such as facial expressions and tone of voice.

In addition, the proctoring component of the system will monitor the candidate's behavior during the interview to ensure the integrity of the process. This will include detecting and identifying any unauthorized materials and tracking eye movement to ensure the candidate is fully engaged in the interview.

Overall, our AI-based interviewing and proctoring system will offer a more comprehensive and fair evaluation of candidates, helping organizations to make more informed hiring decisions.

Technologies like HTML, CSS, Javascript, Flask, Python, and machine learning were used in the development of this system.

**Keywords:** logistic regression, linear regression, Support Vector Machine, K Nearest Neighbors, heart disease, breast cancer, diabetes, calories

**TABLE OF CONTENTS**

| **DESCRIPTION** | **PAGE NUMBERS** |
| --- | --- |
| [CERTIFICATE](#_30j0zll) | i |
| [DECLARATION](#_1fob9te) | ii |
| ACKNOWLEDGEMENTS | iii |
| [ABSTRACT](#_3znysh7) | iv |
| [LIST OF FIGURES](#_2et92p0) | v |
| CHAPTER 1 INTRODUCTION   * 1. Introduction   1.2 Issues Identified  1.3 Objective  1.4 Motivation for work | 1 |
| CHAPTER 2 LITERATURE SURVEY  2.1 Introduction  2.2 Existing Model  2.3 Proposed Model  2.4 Conclusion | 3 |
| CHAPTER 3 METHODOLOGY  3.1 Introduction  3.2 Opencv Process  3.2.1 Haar Cascades  3.2.2 Dlib Frontend Face Detector  3.3 Technologies used  3.3.1 Python  3.3.2 Machine Learning  3.3.3 Opencv  3.3.4 HTML  3.3.5 CSS  3.3.6 Java Script | 5 |
| CHAPTER 4 System Design  4.1 Introduction  4.1.1 Why UML diagrams are used  4.1.2 Goals of UML  4.2 Types of UML  4.2.1 Class Diagram  4.2.2 Use case Diagram  4.2.3 Data Flow Diagram  4.3 Conclusion | 11 |
| CHAPTER 5 Implementation  5.1 Introduction  5.2 Installation  5.2.1 Python Installations  5.2.2 Scikit Learn  5.2.3 Install requirements  5.3 Implementing Models  5.3.1 Face Detection Model  5.3.2 Eye Tracking  5.3.3 Mouth Opening Detection  5.3.4 Head Pose Estimation  5.3.5 Audio Analysis  5.3.6 Face Recognition  5.3.7 Speech2Text & Text2Speech | 17 |
| CHAPTER 6 RESULTS  6.1 Introduction  6.2 Results  6.2.1 Home Page  6.2.2 Signup Recruiter page  6.2.3 Login Recruiter page  6.2.4 Signup Job Seeker page  6.2.5 Login Job Seeker page  6.2.6 After Login Recruiter page  6.2.7 After Login Job Seeker page  6.2.8 Exam Time Quiz  6.3 Conclusion | 40 |
| CHAPTER 7 CONCLUSION  7.1 Conclusion  7.2 Future Scope | 47 |
| CHAPTER 8 REFERENCES  8.1 References | 48 |

**LIST OF FIGURES**

| FIGURE  NO | TITLE | PAGE  NUMBER |
| --- | --- | --- |
| 3.1 | Ai based proctoring process | 5 |
| 3.2 | Workflow process | 5 |
| 3.3 | OpenCV process | 6 |
| 3.4 | Python | 7 |
| 3.5 | OpenCV | 8 |
| 3.6 | HTML | 9 |
| 3.7 | Java Script | 10 |
| 4.1 | Project Architecture | 11 |
| 4.2 | Class Diagram | 12 |
| 4.3 | Use case Diagram | 13 |
| 4.4 | Data Flow Diagram for interview system | 14 |
| 4.5 | Data Flow for information gathering | 14 |
| 4.6 | Data Flow of video analysis | 15 |
| 4.7 | Data flow of audio analysis | 15 |
| 4.8 | Data Flow for Interview process | 16 |
| 4.9 | Data flow of hiring process | 16 |
| 5.1 | Python Website | 17 |
| 5.2 | Download Python | 18 |
| 5.3 | Install Python | 18 |
| 5.4 | Python is installing | 19 |
| 5.5 | Successful Python installation | 19 |
| 5.6 | Running Python | 20 |
| 5.7 | Facial landmarks | 22 |
| 6.1 | Home page | 41 |
| 6.2 | Sign up recruiter page | 42 |
| 6.3 | Login recruiter page | 43 |
| 6.4 | Sign up job seeker page | 43 |
| 6.5 | Login job seek page | 44 |
| 6.6 | After login page recruiter side | 44 |
| 6.7 | After login page job seeker side | 45 |
| 6.8 | Exam demo 1 | 46 |
| 6.9 | Exam demo 2 | 46 |
|  |  |  |

**CHAPTER 1**

**INTRODUCTION**

**1.1 Introduction**

AI-based proctoring leverages Artificial Intelligence (AI) to predict, flag and register even the minutest of discrepancies and suspicion to prevent cheating during online examinations. Some of the many AI flags include distracted eye movements, the candidate not visible, detection of an unauthorized device, another person’s presence, etc.

AI-based proctoring is a process that assesses a test-taking individual’s behavior, environment and movement, similar to a human proctor. In AI-based proctoring, the AI algorithm is trained by experts to predict, flag and register even the minutest of discrepancies and suspicion. Certain AI flags include distracted eye movements, the candidate not visible, partial visibility, detection of an unauthorized device, another person’s presence, the person speaking not visible on the camera,opening any unauthorized website etc.

A proctored exam allows the invigilators to invigilate remotely. They use video, audio, and various anti-cheating features to maintain the exam’s credibility. Manual online proctoring in the remote examination is a difficult task as many students cannot be invigilated at the same time. During manually proctored examinations at the centers, a teacher can physically monitor students using all the senses. They can notice the sounds, movements of students and can easily ensure smooth conduct of the event. Online examinations restrict supervision as the teacher is not physically present at the location. A good remote online proctoring system should facilitate movement and sound detection.

**1.2 Issues Identified**

**Flexibility:** Online proctoring allows for much more flexibility. On the one hand, institutions can conduct and supervise assessment at a much larger scale, over different time zones. On the other hand, students are able to arrange their own preferred time slots to take the tests, and to break free from the need to travel for exams**.**

**Task automation:** Employing a virtual surveillance system helps institutions save time and effort in mundane tasks, such as printing, arranging test venues, and recruiting human invigilators**.**

**1.3 Objective**

We need to design an AI based architecture in which the whole recruitment process needs to be automated and computer vision models we use need to be efficient and cost effective. Finally we need to achieve a strong proctoring system and cost effective software.

**1.4 Motivation for work**

As a result of all the advancements and discoveries in the disciplines of AI and ML, doing in-depth research and offering a practical application has become an absolute joy. We believe AI would be extremely beneficial to the Proctoring Systems. Our goal was to minimize the chance of cheating for clients and try to make Strong AI models

**CHAPTER 2**

**LITERATURE SURVEY**

**2.1 Introduction**

Artificial intelligence (AI) in the Proctoring System is receiving attention from researchers and AI professionals. Few previous studies have investigated this topic from a multi-disciplinary perspective, including accounting, business and management, decision sciences and AI professions.

At the time of writing, academic literature on the specific influence AI will have on the recruitment industry is very limited. This review will be integrative in order to develop new perspectives on the topic (Saunders et al, 2016). To widen the array of literature, both scholarly literature and professional sources have been used. This is firstly because AI is a topic that is incredibly fast moving and as a result caused scholarly literature to fall behind. Secondly the area of HR is rarely covered by academics and thus better suited to professional reports. To develop a qualitative analysis of these sources thematic analysis was used. These themes were taken from arguments made in the literature surrounding and the impact the influence that AI could have in Recruitment, specifically on candidates and employers.

These are:

• Risks and Limitations

• Bias and Inclusion

• Technicalities and Opportunities

**2.2 Existing Model**

A lot of Research happened in this area where the main focus was on non verbal communication.in which opencv methods were really amazing and accurate to analyze video streaming. Many of those methods are based on facial characteristics. AI is able to find a person when a person is cheating or not.

Skillet,TurboHire,Talenture,Zoho and many other tools have been developed by using AI based proctoring systems where these tools are built based on opencv. These models are accurate but even it has problems faced and a lot of inaccurate results will happen.

**2.3 Proposed System**

Even we have lot existing opencv models,Research papers but Proctoring Systems are not completely rely on AI Software or else a machine because we can trust on AI completely. We can even observe that Strong AI proctoring systems in GRE,IELTS examinations were also cheated. Even a human we have a human invigilator monitoring along with software.

So we tried to make opencv models as lightweight as possible and try to mimic the behavior features of existing proctoring systems in a more accurate and cost effective software. We have made a prototype like a lot of people trying to make strong AI models but the fact is that they are not cost effective and every organization is not affortable.

Finally we prepared a prototype where non verbal communication is focussed and tested using open cv methods and we made an automatic process like everything in a recruitment process is automated.

**2.4 Conclusion**

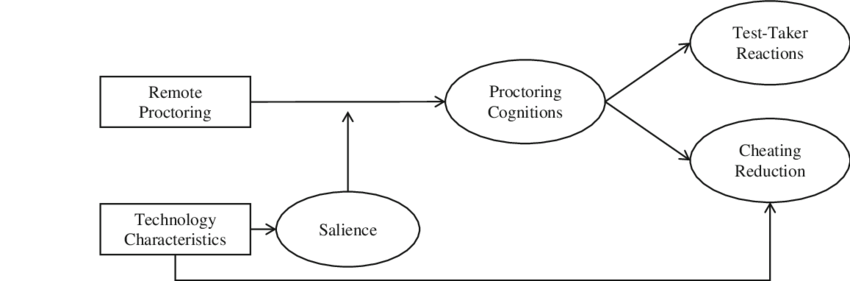
We achieved good accuracy for all our features like eye tracking, mouth tracking. head pose estimation and audio recording as well .

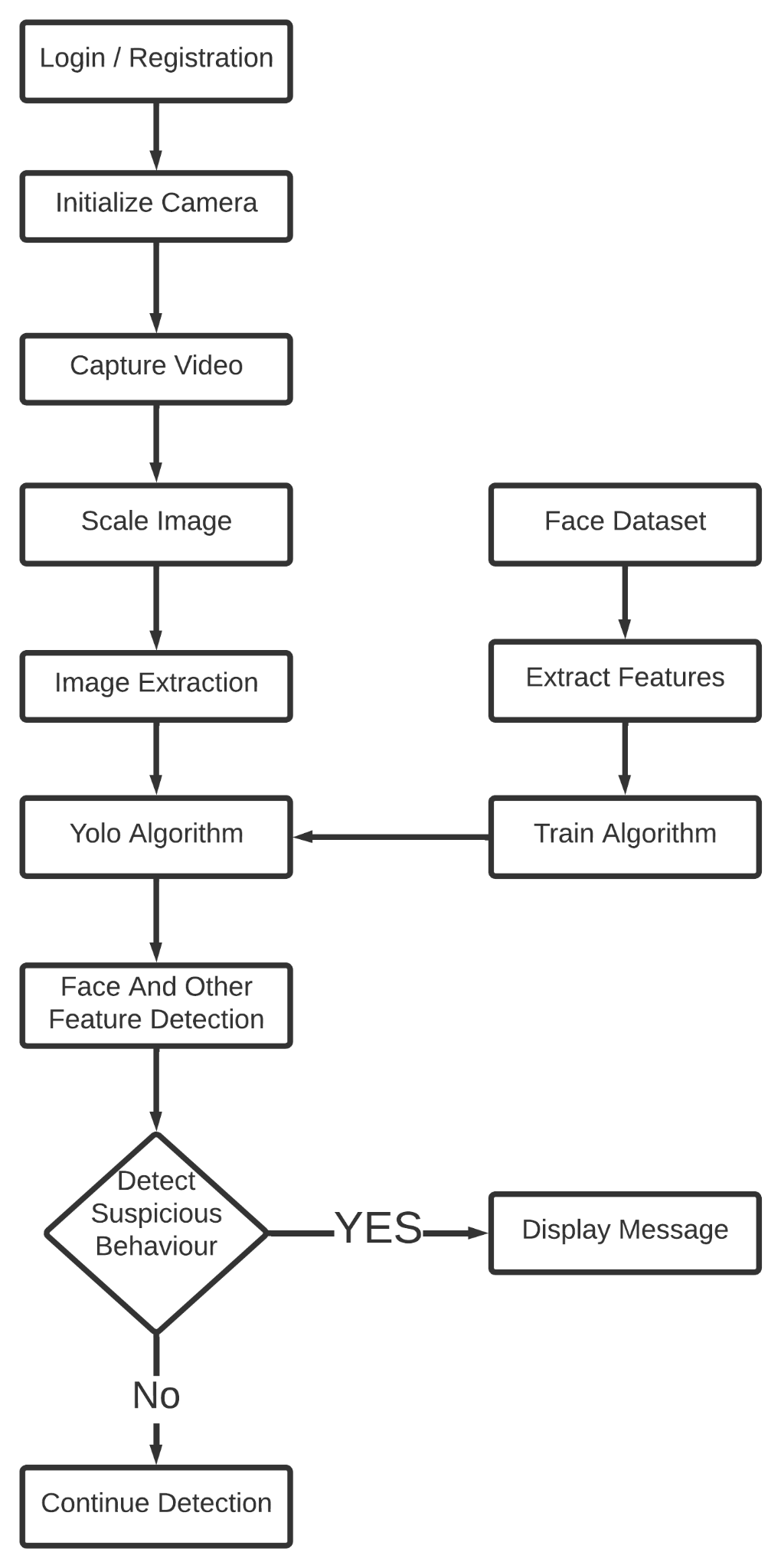
**CHAPTER 3**

**METHODOLOGY**

**3.1 Introduction**

To implement the opencv models to find features of faces using opencv algorithms as shown in the figure 3.1

  
**Fig 3.1:** AI based Proctoring Process



**Fig 3.2 Workflow Process**

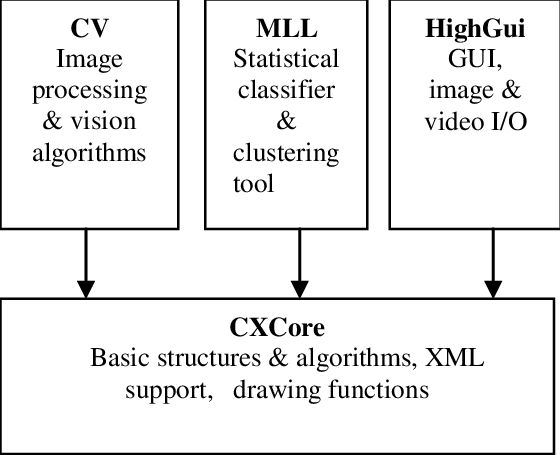


Fig 3.3 Opencv Process

**3.2 Opencv process**

## 3.2.1 Haar Cascades

They were proposed way back in 2001 by Paul Viola and Micheal Jones in their paper, “Rapid Object Detection using a Boosted Cascade of Simple Features.” It is super fast to work with and like the simple CNN, it extracts a lot of features from images. The best features are then selected via Adaboost. This reduces the original 160000+ features to 6000 features. But applying all these features in a sliding window will still take a lot of time. So they introduced a Cascade of Classifiers, where the features are grouped. If a window fails at the first stage, these remaining features in that cascade are not processed. If it passes then the next feature is tested and the same procedure is repeated. If a window can pass all the features then it is classified as a face region.

Haar cascades require a lot of positive and negative training images to train. Thankfully, these cascades come bundled with the OpenCV library along with the trained XML files.

## 3.2.2 Dlib Frontal Face Detector

Dlib is a C++ toolkit containing machine learning algorithms used to solve real-world problems. Although it is written in C++ it has python bindings to run it in python. It also has the great facial landmark keypoint detector which I used in one of my earlier articles to make a real-time gaze tracking system.

**3.3 Technologies Used**

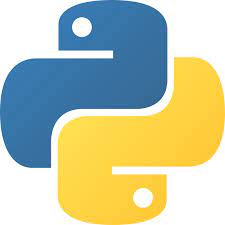
To develop the AI based Proctoring system we have used various technologies like python, HTML, CSS ,JS and opencv methods. We will understand each technology.

**3.3.1 Python**

Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace.

Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.   
• Python is Interpreted − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.   
• Python is Interactive − you can actually sit at a Python prompt and interact with the interpreter directly to write your programs.

Python also acknowledges that speed of development is important. Readable and terse code is part of this, and so is access to powerful constructs that avoid tedious repetition of code. Maintainability also ties into this may be an all but useless metric, but it does say something about how much code you have to scan, read and/or understand to troubleshoot problems or tweak behaviors. Python is used to create web and desktop applications, and some of the most popular web applications like Instagram, YouTube, Spotify all have been developed in Python. You can also develop the next big thing by using Python.

  
 **Fig 3.4** Python

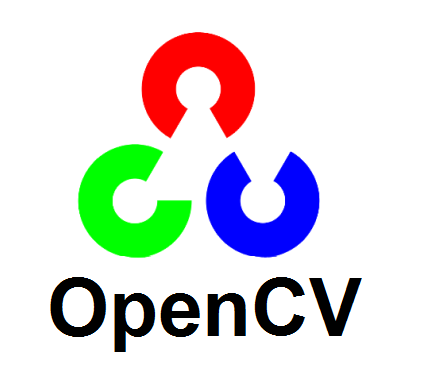
**3.3.2 Machine Learning**

Machine learning (ML) is a type of artificial intelligence (AI) that allows software applications to become more accurate at predicting outcomes without being explicitly programmed to do so. Machine learning algorithms use historical data as input to predict new output values.

Recommendation engines are a common use case for machine learning. Other popular uses include fraud detection, spam filtering, malware threat detection, business process automation (BPA) and Predictive maintenance.  
  
**3.3.3 Opencv**

OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products. Being an Apache 2 licensed product, OpenCV makes it easy for businesses to utilize and modify the code.

OpenCV is a huge open-source library for computer vision, machine learning, and image processing. OpenCV supports a wide variety of programming languages like Python, C++, Java, etc. It can process images and videos to identify objects, faces, or even the handwriting of a human. When it is integrated with various libraries, such as Numpy which is a highly optimized library for numerical operations, then the number of weapons increases in your Arsenal i.e whatever operations one can do in Numpy can be combined with OpenCV.

  
**Fig 3.5:** Opencv

Ease of use*:* OpenCV is easy and simple to learn

Availability of many tutorials: The fact that there are lots of tutorials available is a big plus as one can access many learning resources.

Compatibility with leading coding languages: OpenCV works with almost all the leading programming languages today, including Python, C++ and Java.

Free to use: Undoubtedly, a big plus is the fact that it is open source and hence free to use.

**3.3.4 HTML**

HyperText Markup Language (HTML) is a type of markup language. It is made by the World Wide Web Consortium (W3C). It is used to make webpages. Webpages can include writing, links, pictures, and even sound and video. HTML tells web browsers what webpages should look like. HTML also adds meta information to webpages. Meta information is information about a webpage e.g., the name of the person who created the page. Web browsers usually do not show meta information.

  
**Fig 3.6:** HTML version 5

**3.3.5 CSS**

Cascading Style Sheets fondly referred to as CSS, is a simply designed language intended to simplify the process of making web pages presentable. CSS allows you to apply styles to web pages. More importantly, CSS enables you to do this independent of the HTML that makes up each web page. CSS is easy to learn and understood, but it provides powerful control over the presentation of an HTML document.

We use CSS because of the following reasons:

* CSS saves time: You can write CSS once and reuse the same sheet on multiple HTML pages.
* Easy Maintenance: To make a global change simply change the style, and all elements in all the webpages will be updated automatically.
* Search Engines: CSS is considered a clean coding technique, which means search engines won’t have to struggle to “read” its content.
* Superior styles to HTML: CSS has a much wider array of attributes than HTML, so you can give a far better look to your HTML page in comparison to HTML attributes.
* Offline Browsing: CSS can store web applications locally with the help of an offline cache. Using this we can view offline websites.

**3.3.6 Javascript**

JavaScript is a text-based programming language used both on the client-side and server-side that allows you to make web pages interactive. Where HTML and CSS are languages that give structure and style to web pages, JavaScript gives web pages interactive elements that engage a user. Common examples of JavaScript that you might use every day include the search box on Amazon, a news recap video embedded on The New York Times, or refreshing your Twitter feed. Incorporating JavaScript improves the user experience of the web page by converting it from a static page into an interactive one. To recap, JavaScript adds behavior to web pages.

  
 **Fig 3.7:** Javascript

**CHAPTER 4**

**SYSTEM DESIGN**

**4.1 Introduction** We introduce you to the system architecture ,client server model along with UML diagrams.



**Fig 4.1:** Project architecture

**4.1.1 Why is the UML diagram used?**

As the strategic value of software increases for many companies, the industry looks for techniques to automate the production of software and to improve quality and reduce cost and time-to-market. These techniques include component technology, visual programming, patterns and frameworks. Businesses also seek techniques to manage the complexity of systems as they increase in scope and scale.

**4.1.2 Goals of UML**

The primary goals in the design of the UML are:

• Provide users with a ready-to-use, expressive visual modeling language so they can develop and exchange meaningful models.

• Provide extensibility and specialization mechanisms to extend the core concepts.

• Being independent of particular programming languages and development processes.

• Provide a formal basis for understanding the modeling language.

• Encourage the growth of the OO tools market.

• Support higher-level development concepts such as collaborations, frameworks, patterns and components.

**4.2 Types of UML diagrams**

There are several types of UML diagrams and each one of them serves a different purpose regardless of whether it is being designed before the implementation or after (as part of documentation).

The two most broad categories that encompass all other types are Behavioral UML diagram and Structural UML diagram. As the name suggests, some UML diagrams try to analyze and depict the structure of a system or process, whereas other describe the behavior of the system, its actors, and its building components.

**4.2.1 Class Diagram**

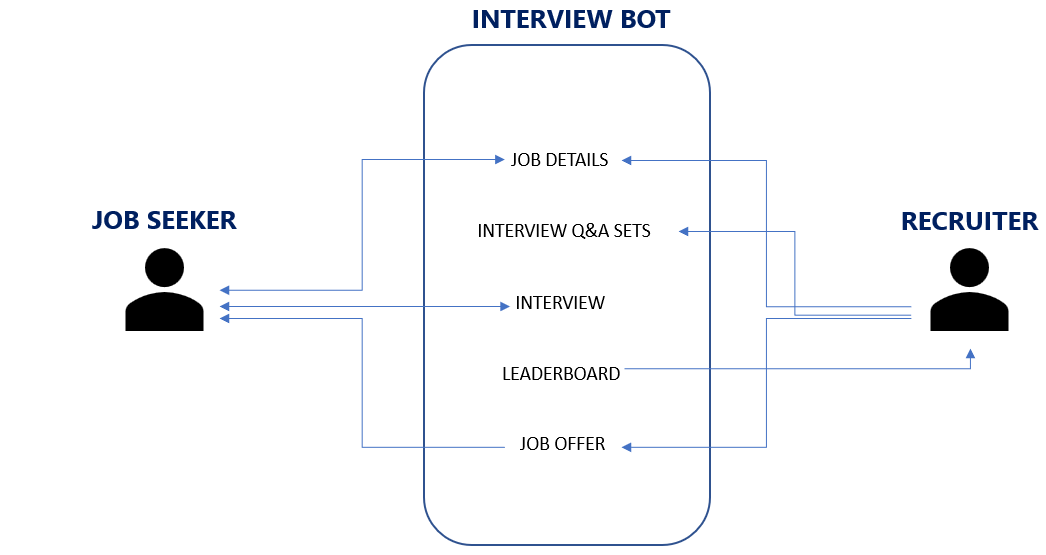
In software engineering, a class diagram in the unified modeling language (uml) is a type of static structure diagram that describes the structure of a system by showing the system’s classes, their attributes, operations (or methods), and the relationships among the classes. It explains the switch class contains information.



**Fig 4.2:** Class Diagram of AI interview proctoring system

**4.2.2 Use Case Diagram**

A use case diagram in the unified modeling language (uml) is a type of behavioral diagram defined by and created from a use case analysis.Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases.

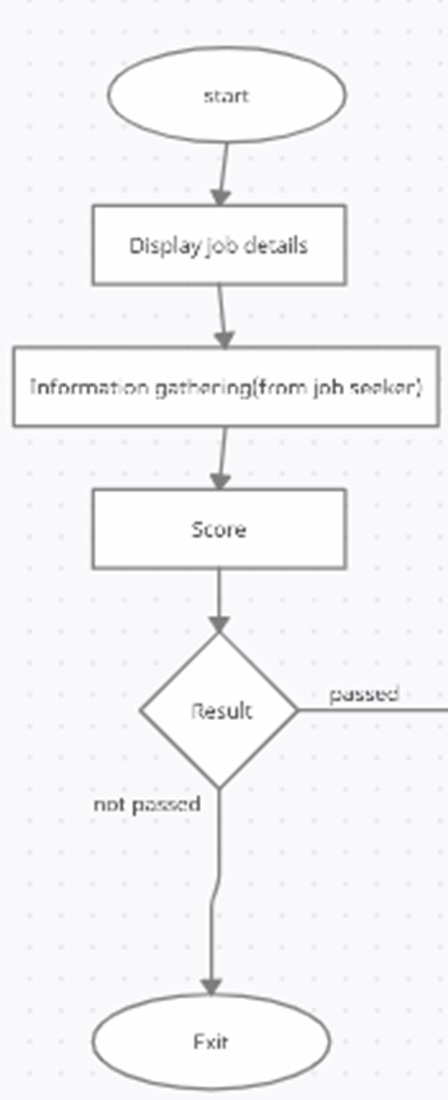


**Fig 4.3:** Use Case Diagram

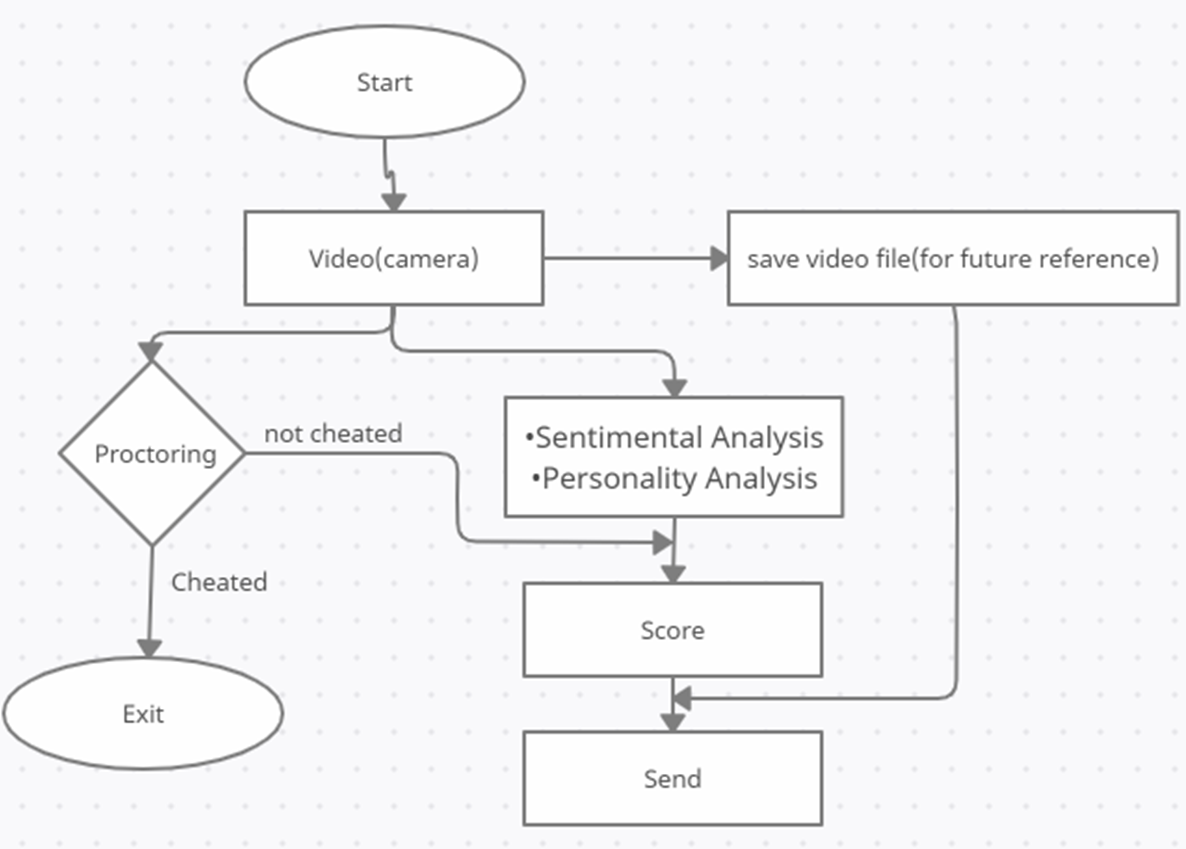
**4.2.3 Data Flow Diagram**

Data Flow diagrams are for graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the unified modeling language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

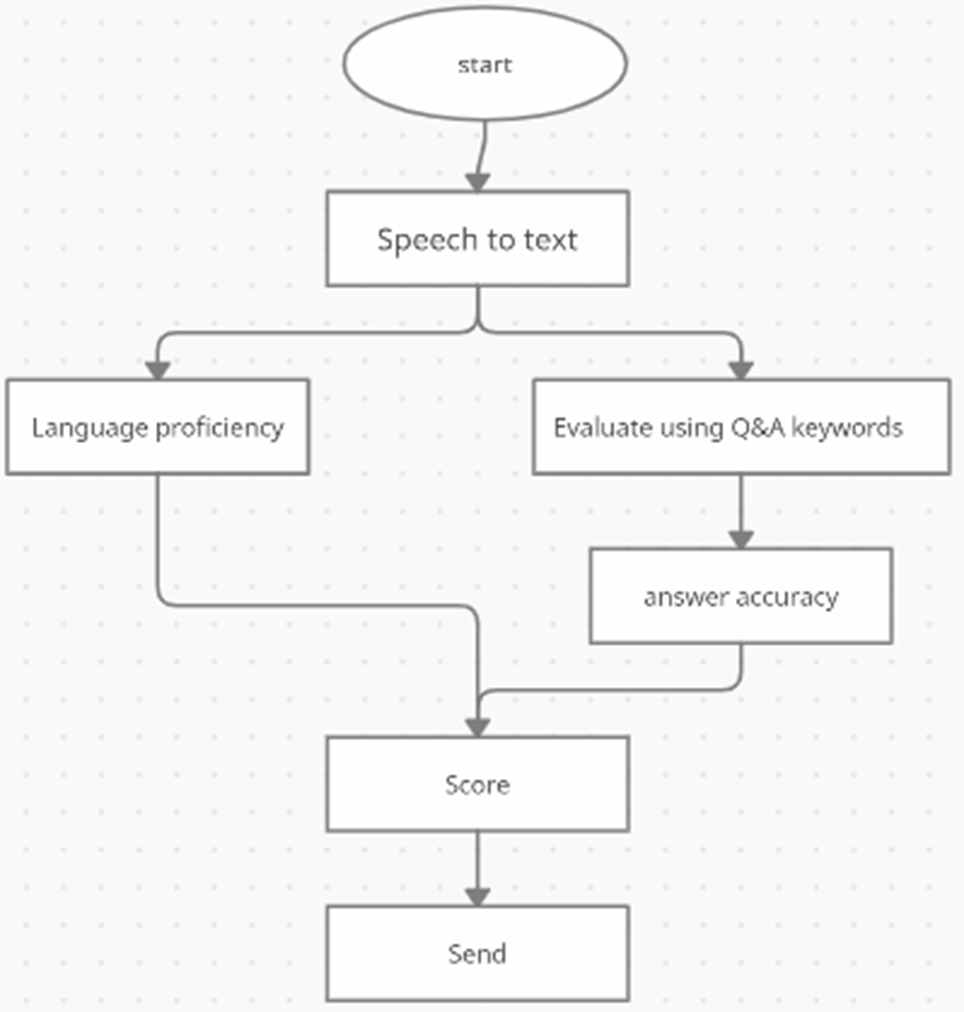
  
**FIg 4.4:** Data Flow diagram of interview automation system



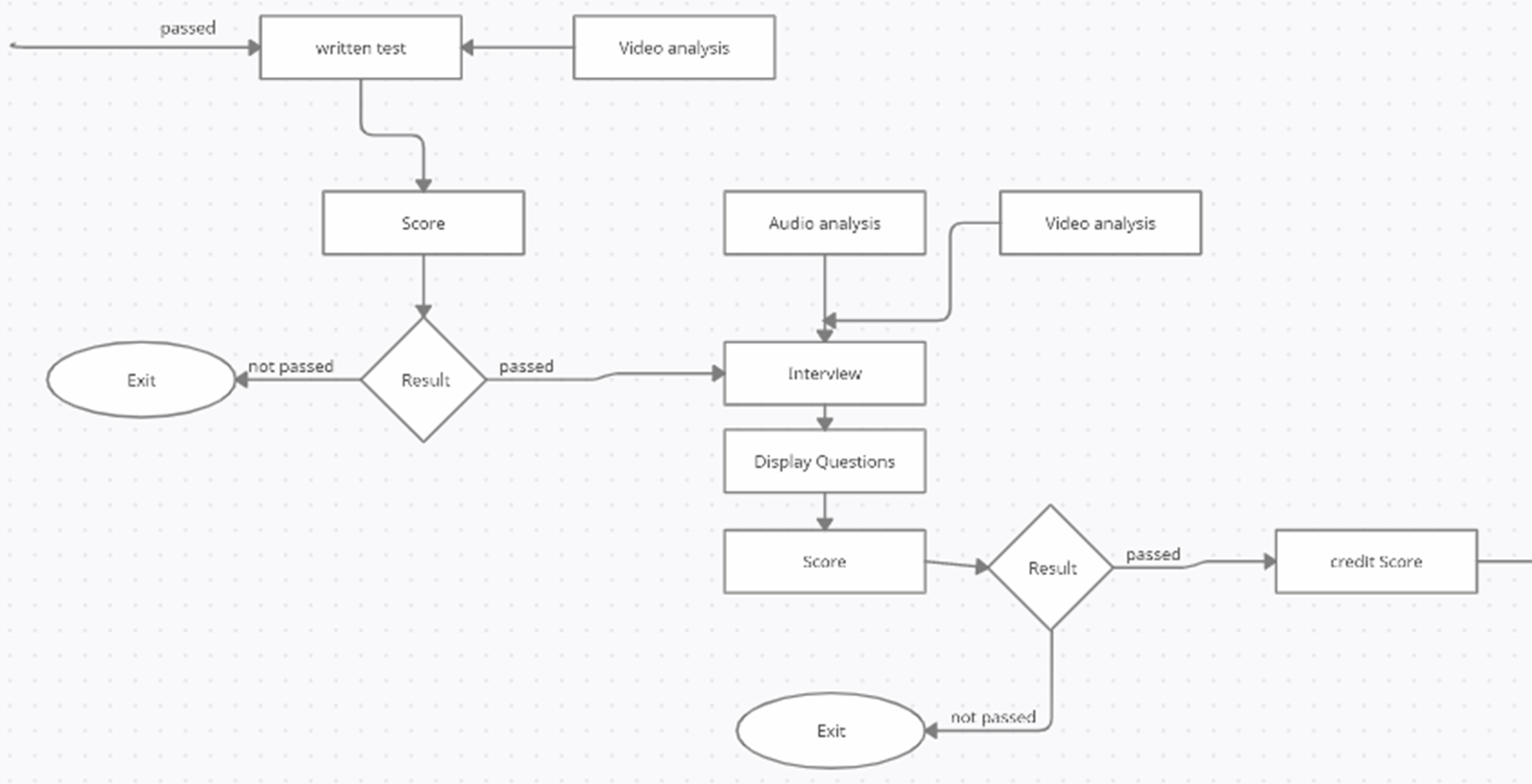
**Fig 4.5:** Data flow diagram of Information gathering



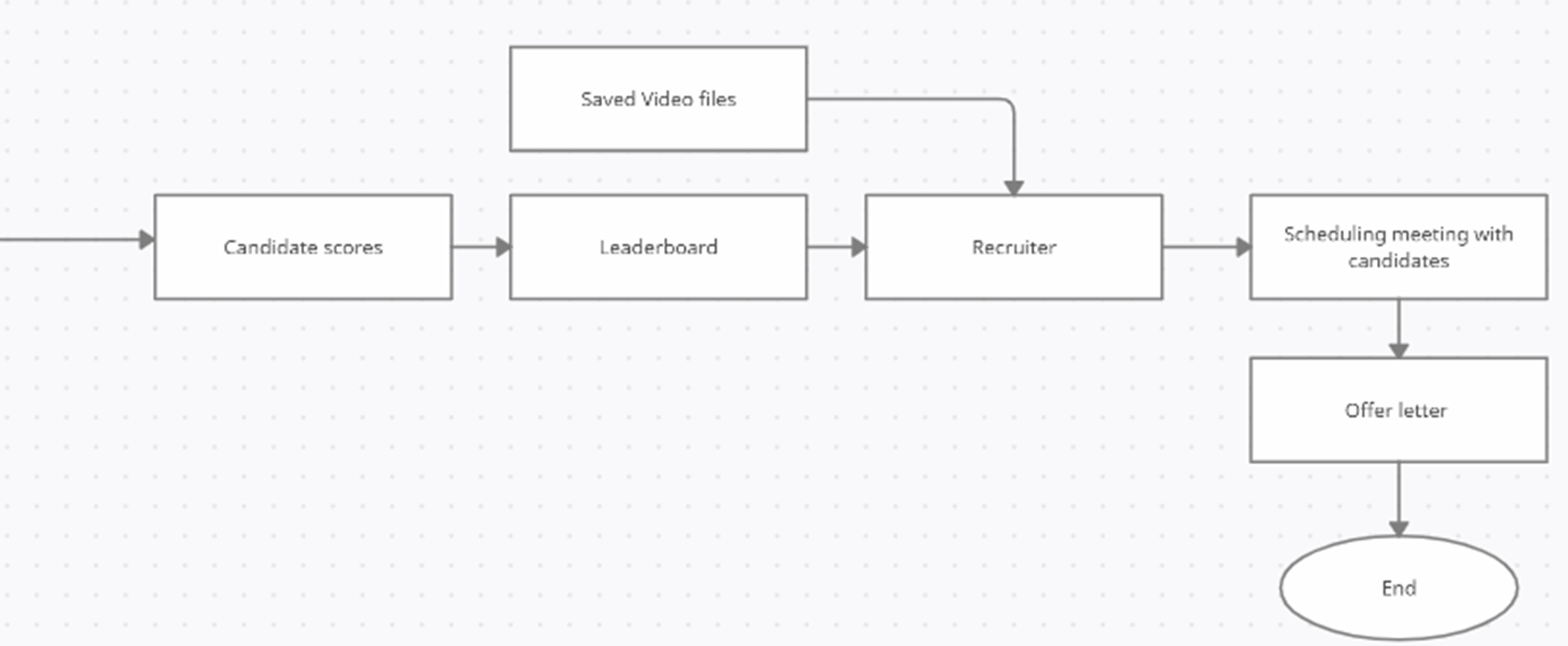
**Fig 4.6:**Data flow diagram of Video Analysis



**Fig 4.7:**Data flow diagram of Audio Analysis



**Fig 4.8:**Data flow diagram of Interview Process



**Fig 4.9:** Data flow of Hiring Process

**4.3 Conclusion**

We have demonstrated various designs and provided you with a thorough explanation of the structure and operation of our Artificial Intelligence based interviewing and proctoring system.

**CHAPTER 5**

**IMPLEMENTATION**

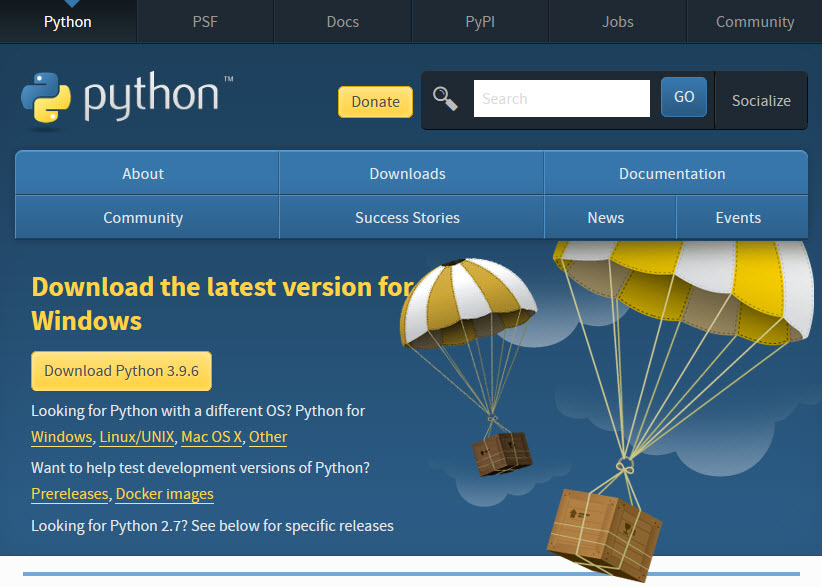
**5.1 Introduction**

This chapter discusses the implementation of the system. To implement this project you need python and a few packages to be installed in your system. Section 5.2 provides the steps to install python in your system.

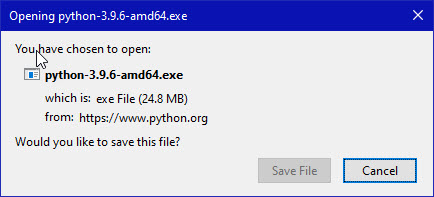
**5.2 Installations**

**5.2.1 Python Installation**

1. Visit official python website <https://www.python.org/downloads/>

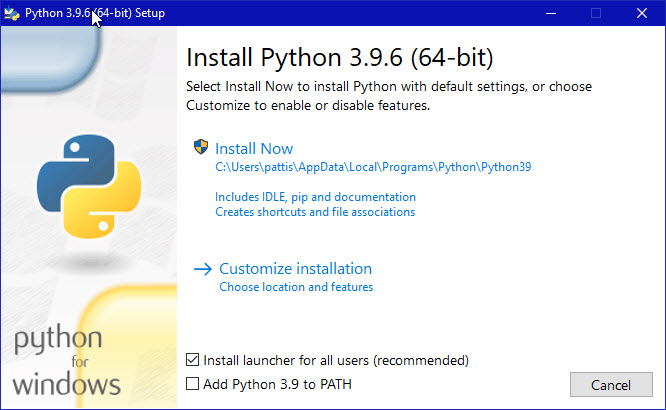
  
**Fig 5.1:** Python website

1. Click the **Download Python 3.9.6 (or the latest version)**  button. The following pop-up window titled **Opening python-3.96-amd64.exe** will appear.

  
**Fig 5.2:** Download python

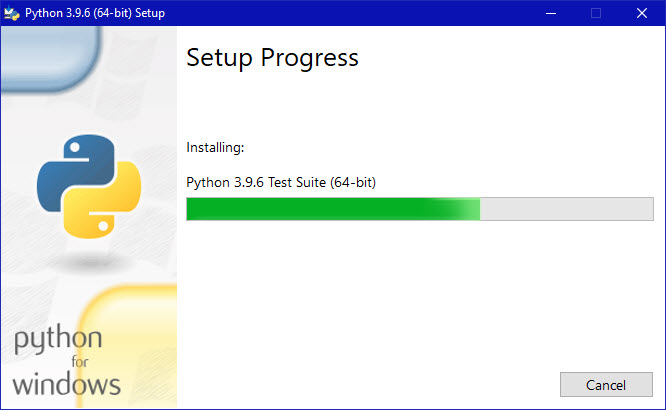
Click the **Save File** button. The file named **python-3.9.6-amd64.exe** should start downloading into your standard download folder. This file is about 25 Mb so it might take a while to download fully if you are on a slow internet connection (it took me about 10 seconds over a cable modem).

1. Double-click the icon labeling the file **python-3.9.6-amd64.exe**.  
   A **Python 3.9.6 (64-bit) Setup** pop-up window will appear.

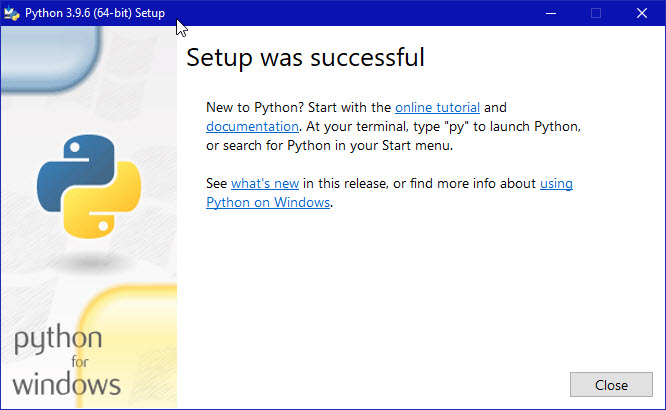
  
**Fig 5.3** Installing python

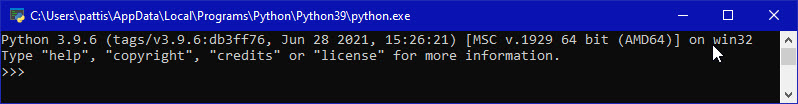
Ensure that **both** the **Install launcher for all users (recommended)** and the **Add Python 3.9 to PATH** checkboxes at the bottom are checked: typically only first is checked by default.  
If the Python Installer finds an earlier version of Python installed on your computer, the **Install Now** message may instead appear as **Upgrade Now** (and the checkboxes will not appear). Highlight the **Install Now** (or **Upgrade Now**) message, and then click it. When run, a **User Account Control** pop-up window may appear on your screen. I could not capture its image, but it asks, **Do you want to allow this app to make changes to your device?'’**

1. Click the **Yes** button.  
   A new **Python 3.9.6 (64-bit) Setup** pop-up window will appear with a **Setup Progress** message and a progress bar.

  
**Fig 5.4:** Python is installing

During installation, it will show the various components it is installing and move the progress bar towards completion. Soon, a new **Python 3.9.6 (64-bit) setup** pop-up window will appear with a **Setup was successful** message.

  
**Fig 5.5:** Successful python installation

1. Click the **Close** button.
2. To try to verify installation, Navigate to the directory **C:\Users\Pattis\AppData\Local\Programs\Python\Python39** (or to whatever directory Python was installed: see the pop-up window for Installing in Fig 5.3 Double-click the icon/file **python.exe**.  
   The following pop-up window will appear.  
     
     
     
    **Fig 5.6:** Running Python  
     
   A pop-up window with the title **C:\Users\Pattis\AppData\Local\Programs\Python\Python39\python.exe** appears, and inside the window; on the first line is the text **Python 3.9.6 ...** (notice that it should also say 64 bit). Inside the window, at the bottom left, is the prompt **>>>**: type **exit()** to this prompt and press **enter** to terminate Python.

**5.2.2 Scikit Learn**

Scikit-learn (Sklearn) is the most useful and robust library for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction via a consistence interface in Python. This library, which is largely written in Python, is built upon NumPy, SciPy and Matplotlib. We have built the project on scikit-learn version 0.24.1. To install scikit-learn run the command

| pip install -U scikit-learn==0.24.1 |
| --- |

**5.2.3 Installing required packages**

We have shared all the required packages to be installed in file requirements.txt. To install all the packages mentioned in the requirements.txt file run the below command

| pip install -r requirements.txt |
| --- |

**5.3 Implementing models**

**5.3.1 Face Detection**

Dlib's frontal face HOG detector was used to find faces. However, it did not give very good results. In face\_detection different face detection models are compared and OpenCV's DNN module provides the best result.

It is implemented in face\_detector.py and is used for tracking eyes, mouth opening detection, head pose estimation, and face spoofing.

An additional quantized model is also added for the face detector. This can be used by setting the parameter quantized as True when calling the get\_face\_detector(). On quick testing of the face detector on my laptop the normal version gave ~17.5 FPS while the quantized version gave ~19.5 FPS. This would be especially useful when deploying on edge devices due to it being uint8 quantized.

**Facial Landmarks**

Dlib's facial landmarks model was used but it did not give good results when the face was at an angle. A comparison between them and the reason for choosing the new Tensorflow based model.

It is implemented in face\_landmark.py and is used for tracking eyes, mouth opening detection, and head pose estimation.

**5.3.2 Eye Tracking**

1. Detect the face in the input image or video frame using a face detection algorithm. OpenCV provides various face detection algorithms such as Haar Cascade classifier and deep learning-based face detection networks.
2. Once the face is detected, extract the region of interest (ROI) corresponding to the eyes. This can be done using the coordinates of the facial landmarks such as the eyes, nose, and mouth, which can be detected using OpenCV's built-in facial landmark detection algorithms.
3. Apply image processing techniques such as thresholding, edge detection, and morphological operations to enhance the eye region and extract relevant features such as the pupil, iris, and eyelids.
4. Finally, use the predicted gaze direction to control the user interface or any other application. For example, if the user is looking at a button on the screen, the gaze direction can be used to simulate a mouse click on the button.

The first thing to do is to find eyes before we can move on to image processing and to find the eyes we need to find a face. The facial keypoint detector takes a rectangular object of the dlib module as input which is simply the coordinates of a face. To find faces we can use the inbuilt frontal face detector of dlib. You can use any classifier for this task. If you want high accuracy and speed is not an issue for you then I would suggest you use a CNN.

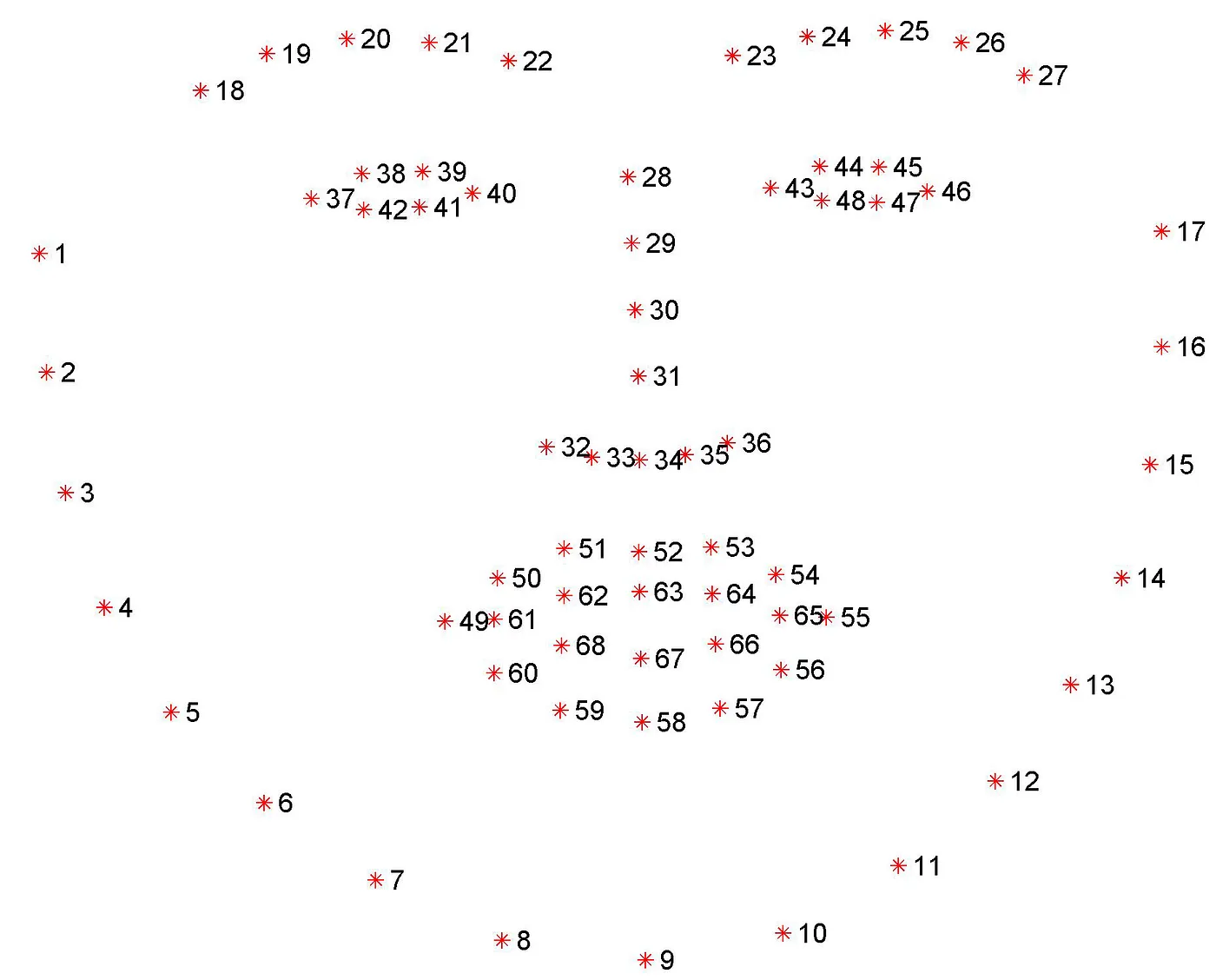


Fig 5.7 Facial Landmarks

Thresholding is used to create a binary mask. So our task is to find an optimal threshold value against which we can segment out the eyeballs from the rest of the eye and then we need to find its center. But the threshold value will be different for different lighting conditions so we can make an adjustable trackbar for controlling the threshold value. The thresholding processing steps namely erosion, dilation, and median blur are also taken from him but his final results were not convincing so I made this solution.

**5.3.3 Mouth opening detection**

mouth\_opening\_detector.py is used to check if the candidate opens his/her mouth during the exam after recording it initially. Its explanation can be found in the main article, however, it is using dlib which can be easily changed to the new models.

1. Load the input video or image and convert it to grayscale.
2. Use a face detection algorithm to detect the face in the input image or video frame. OpenCV provides various face detection algorithms such as Haar Cascade classifier and deep learning-based face detection networks.
3. Once the face is detected, extract the region of interest (ROI) corresponding to the mouth. This can be done using the coordinates of the facial landmarks such as the mouth corners and the bottom lip, which can be detected using OpenCV's built-in facial landmark detection algorithms.
4. Apply image processing techniques such as thresholding, edge detection, and morphological operations to enhance the mouth region and extract relevant features such as the mouth opening.
5. Use a suitable machine learning algorithm, such as SVM or Random Forest, to classify the mouth opening based on the extracted features. These algorithms can be trained on labeled datasets to predict the mouth opening.
6. Finally, use the predicted mouth opening to trigger an action. For example, if the user opens their mouth wider than a certain threshold, the action can be used to control a device or an application.

**5.3.4 Head Pose Estimation**

Head pose detection on images with a lot of Maths about converting the points to 3D space and using cv2.solvePnP to find rotational and translational vectors.

cv2.solvePnP is a function in OpenCV's Python bindings that can be used to estimate the pose of an object in a 3D space from its 2D image projection. It is commonly used in computer vision and robotics applications to determine the position and orientation of a camera relative to a known object or scene.

Use the coordinates of the facial landmarks to estimate the orientation of the head in 3D space. One common approach is to use the SolvePnP algorithm, which is available in OpenCV's Python bindings as the cv2.solvePnP function. This function takes as input the 3D coordinates of the facial landmarks and their 2D projections in the image plane, as well as the camera matrix and distortion coefficients, and returns the rotation and translation vectors that describe the pose of the head in 3D space

We need six points of the face i.e. is nose tip, chin, extreme left and right points of lips, and the left corner of the left eye and right corner of the right eye. We take standard 3D coordinates of these facial landmarks and try to estimate the rational and translational vectors at the nose tip. Now, for an accurate estimate, we need intrinsic parameters of the camera like focal length, optical center, and radial distortion parameters. We can estimate the former two and assume the last one is not present to make our work easier. After obtaining the required vectors we can project those 3D points on a 2D surface that is our image.

**5.3.5 Audio**

* Audio from the microphone is recorded and converted to text using Google's speech recognition API. A different thread is used to call the API such that the recording portion is not disturbed a lot, which processes the last one, appends its data to a text file and deletes it.
* NLTK we remove the stop words from that file. The question paper (in txt format) is taken whose stopwords are also removed and their contents are compared. Finally, the common words along with its number are presented to the proctor.

**Eye\_tracker.py**

| import cv2  import numpy as np  from face\_detector import get\_face\_detector, find\_faces  from face\_landmarks import get\_landmark\_model, detect\_marks  def eye\_on\_mask(mask, side, shape):    points = [shape[i] for i in side]  points = np.array(points, dtype=np.int32)  mask = cv2.fillConvexPoly(mask, points, 255)  l = points[0][0]  t = (points[1][1]+points[2][1])//2  r = points[3][0]  b = (points[4][1]+points[5][1])//2  return mask, [l, t, r, b]  def find\_eyeball\_position(end\_points, cx, cy):  """Find and return the eyeball positions, i.e. left or right or top or normal"""  x\_ratio = (end\_points[0] - cx)/(cx - end\_points[2])  y\_ratio = (cy - end\_points[1])/(end\_points[3] - cy)  if x\_ratio > 3:  return 1  elif x\_ratio < 0.33:  return 2  elif y\_ratio < 0.33:  return 3  else:  return 0    def contouring(thresh, mid, img, end\_points, right=False):    cnts, \_ = cv2.findContours(thresh, cv2.RETR\_EXTERNAL,cv2.CHAIN\_APPROX\_NONE)  try:  cnt = max(cnts, key = cv2.contourArea)  M = cv2.moments(cnt)  cx = int(M['m10']/M['m00'])  cy = int(M['m01']/M['m00'])  if right:  cx += mid  cv2.circle(img, (cx, cy), 4, (0, 0, 255), 2)  pos = find\_eyeball\_position(end\_points, cx, cy)  return pos  except:  pass    def process\_thresh(thresh):    thresh = cv2.erode(thresh, None, iterations=2)  thresh = cv2.dilate(thresh, None, iterations=4)  thresh = cv2.medianBlur(thresh, 3)  thresh = cv2.bitwise\_not(thresh)  return thresh  def print\_eye\_pos(img, left, right):    if left == right and left != 0:  text = ''  if left == 1:  print('Looking left')  text = 'Looking left'  elif left == 2:  print('Looking right')  text = 'Looking right'  elif left == 3:  print('Looking up')  text = 'Looking up'  font = cv2.FONT\_HERSHEY\_SIMPLEX  cv2.putText(img, text, (30, 30), font,  1, (0, 255, 255), 2, cv2.LINE\_AA)  face\_model = get\_face\_detector()  landmark\_model = get\_landmark\_model()  left = [36, 37, 38, 39, 40, 41]  right = [42, 43, 44, 45, 46, 47]  cap = cv2.VideoCapture(0)  ret, img = cap.read()  thresh = img.copy()  cv2.namedWindow('image')  kernel = np.ones((9, 9), np.uint8)  def nothing(x):  pass  cv2.createTrackbar('threshold', 'image', 75, 255, nothing)  while(True):  ret, img = cap.read()  rects = find\_faces(img, face\_model)    for rect in rects:  shape = detect\_marks(img, landmark\_model, rect)  mask = np.zeros(img.shape[:2], dtype=np.uint8)  mask, end\_points\_left = eye\_on\_mask(mask, left, shape)  mask, end\_points\_right = eye\_on\_mask(mask, right, shape)  mask = cv2.dilate(mask, kernel, 5)    eyes = cv2.bitwise\_and(img, img, mask=mask)  mask = (eyes == [0, 0, 0]).all(axis=2)  eyes[mask] = [255, 255, 255]  mid = int((shape[42][0] + shape[39][0]) // 2)  eyes\_gray = cv2.cvtColor(eyes, cv2.COLOR\_BGR2GRAY)  threshold = cv2.getTrackbarPos('threshold', 'image')  \_, thresh = cv2.threshold(eyes\_gray, threshold, 255, cv2.THRESH\_BINARY)  thresh = process\_thresh(thresh)    eyeball\_pos\_left = contouring(thresh[:, 0:mid], mid, img, end\_points\_left)  eyeball\_pos\_right = contouring(thresh[:, mid:], mid, img, end\_points\_right, True)  print\_eye\_pos(img, eyeball\_pos\_left, eyeball\_pos\_right)  # for (x, y) in shape[36:48]:  # cv2.circle(img, (x, y), 2, (255, 0, 0), -1)    cv2.imshow('eyes', img)  cv2.imshow("image", thresh)  if cv2.waitKey(1) & 0xFF == ord('q'):  break    cap.release()  cv2.destroyAllWindows() |
| --- |

**mouth\_opening\_detector.py**

| import cv2  from face\_detector import get\_face\_detector, find\_faces  from face\_landmarks import get\_landmark\_model, detect\_marks, draw\_marks  face\_model = get\_face\_detector()  landmark\_model = get\_landmark\_model()  outer\_points = [[49, 59], [50, 58], [51, 57], [52, 56], [53, 55]]  d\_outer = [0]\*5  inner\_points = [[61, 67], [62, 66], [63, 65]]  d\_inner = [0]\*3  font = cv2.FONT\_HERSHEY\_SIMPLEX  cap = cv2.VideoCapture(0)  while(True):  ret, img = cap.read()  rects = find\_faces(img, face\_model)  for rect in rects:  shape = detect\_marks(img, landmark\_model, rect)  draw\_marks(img, shape)  cv2.putText(img, 'Press r to record Mouth distances', (30, 30), font,  1, (0, 255, 255), 2)  cv2.imshow("Output", img)  if cv2.waitKey(1) & 0xFF == ord('r'):  for i in range(100):  for i, (p1, p2) in enumerate(outer\_points):  d\_outer[i] += shape[p2][1] - shape[p1][1]  for i, (p1, p2) in enumerate(inner\_points):  d\_inner[i] += shape[p2][1] - shape[p1][1]  break  cv2.destroyAllWindows()  d\_outer[:] = [x / 100 for x in d\_outer]  d\_inner[:] = [x / 100 for x in d\_inner]  while(True):  ret, img = cap.read()  rects = find\_faces(img, face\_model)  for rect in rects:  shape = detect\_marks(img, landmark\_model, rect)  cnt\_outer = 0  cnt\_inner = 0  draw\_marks(img, shape[48:])  for i, (p1, p2) in enumerate(outer\_points):  if d\_outer[i] + 3 < shape[p2][1] - shape[p1][1]:  cnt\_outer += 1  for i, (p1, p2) in enumerate(inner\_points):  if d\_inner[i] + 2 < shape[p2][1] - shape[p1][1]:  cnt\_inner += 1  if cnt\_outer > 3 and cnt\_inner > 2:  print('Mouth open')  cv2.putText(img, 'Mouth open', (30, 30), font,  1, (0, 255, 255), 2)  # show the output image with the face detections + facial landmarks  cv2.imshow("Output", img)  if cv2.waitKey(1) & 0xFF == ord('q'):  break    cap.release()  cv2.destroyAllWindows() |
| --- |

**head\_pose\_estimation.py**

| import cv2  import numpy as np  import math  from face\_detector import get\_face\_detector, find\_faces  from face\_landmarks import get\_landmark\_model, detect\_marks  def get\_2d\_points(img, rotation\_vector, translation\_vector, camera\_matrix, val):    point\_3d = []  dist\_coeffs = np.zeros((4,1))  rear\_size = val[0]  rear\_depth = val[1]  point\_3d.append((-rear\_size, -rear\_size, rear\_depth))  point\_3d.append((-rear\_size, rear\_size, rear\_depth))  point\_3d.append((rear\_size, rear\_size, rear\_depth))  point\_3d.append((rear\_size, -rear\_size, rear\_depth))  point\_3d.append((-rear\_size, -rear\_size, rear\_depth))    front\_size = val[2]  front\_depth = val[3]  point\_3d.append((-front\_size, -front\_size, front\_depth))  point\_3d.append((-front\_size, front\_size, front\_depth))  point\_3d.append((front\_size, front\_size, front\_depth))  point\_3d.append((front\_size, -front\_size, front\_depth))  point\_3d.append((-front\_size, -front\_size, front\_depth))  point\_3d = np.array(point\_3d, dtype=np.float).reshape(-1, 3)    # Map to 2d img points  (point\_2d, \_) = cv2.projectPoints(point\_3d,  rotation\_vector,  translation\_vector,  camera\_matrix,  dist\_coeffs)  point\_2d = np.int32(point\_2d.reshape(-1, 2))  return point\_2d  def draw\_annotation\_box(img, rotation\_vector, translation\_vector, camera\_matrix,  rear\_size=300, rear\_depth=0, front\_size=500, front\_depth=400,  color=(255, 255, 0), line\_width=2):    rear\_size = 1  rear\_depth = 0  front\_size = img.shape[1]  front\_depth = front\_size\*2  val = [rear\_size, rear\_depth, front\_size, front\_depth]  point\_2d = get\_2d\_points(img, rotation\_vector, translation\_vector, camera\_matrix, val)  # # Draw all the lines  cv2.polylines(img, [point\_2d], True, color, line\_width, cv2.LINE\_AA)  cv2.line(img, tuple(point\_2d[1]), tuple(  point\_2d[6]), color, line\_width, cv2.LINE\_AA)  cv2.line(img, tuple(point\_2d[2]), tuple(  point\_2d[7]), color, line\_width, cv2.LINE\_AA)  cv2.line(img, tuple(point\_2d[3]), tuple(  point\_2d[8]), color, line\_width, cv2.LINE\_AA)      def head\_pose\_points(img, rotation\_vector, translation\_vector, camera\_matrix):    rear\_size = 1  rear\_depth = 0  front\_size = img.shape[1]  front\_depth = front\_size\*2  val = [rear\_size, rear\_depth, front\_size, front\_depth]  point\_2d = get\_2d\_points(img, rotation\_vector, translation\_vector, camera\_matrix, val)  y = (point\_2d[5] + point\_2d[8])//2  x = point\_2d[2]    return (x, y)    face\_model = get\_face\_detector()  landmark\_model = get\_landmark\_model()  cap = cv2.VideoCapture(0)  ret, img = cap.read()  size = img.shape  font = cv2.FONT\_HERSHEY\_SIMPLEX  # 3D model points.  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  ])  # Camera internals  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"  )  while True:  ret, img = cap.read()  if ret == True:  faces = find\_faces(img, face\_model)  for face in faces:  marks = detect\_marks(img, landmark\_model, face)  # mark\_detector.draw\_marks(img, marks, color=(0, 255, 0))  image\_points = np.array([  marks[30], # Nose tip  marks[8], # Chin  marks[36], # Left eye left corner  marks[45], # Right eye right corne  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)      # Project a 3D point (0, 0, 1000.0) onto the image plane.  # We use this to draw a line sticking out of the nose    (nose\_end\_point2D, jacobian) = cv2.projectPoints(np.array([(0.0, 0.0, 1000.0)]), rotation\_vector, translation\_vector, camera\_matrix, dist\_coeffs)    for p in image\_points:  cv2.circle(img, (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][0]), int(nose\_end\_point2D[0][0][1]))  x1, x2 = head\_pose\_points(img, rotation\_vector, translation\_vector, camera\_matrix)  cv2.line(img, p1, p2, (0, 255, 255), 2)  cv2.line(img, tuple(x1), tuple(x2), (255, 255, 0), 2)  # for (x, y) in marks:  # cv2.circle(img, (x, y), 4, (255, 255, 0), -1)  # cv2.putText(img, str(p1), p1, font, 1, (0, 255, 255), 1)  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    # print('div by zero error')  if ang1 >= 48:  print('Head down')  cv2.putText(img, 'Head down', (30, 30), font, 2, (255, 255, 128), 3)  elif ang1 <= -48:  print('Head up')  cv2.putText(img, 'Head up', (30, 30), font, 2, (255, 255, 128), 3)    if ang2 >= 48:  print('Head right')  cv2.putText(img, 'Head right', (90, 30), font, 2, (255, 255, 128), 3)  elif ang2 <= -48:  print('Head left')  cv2.putText(img, 'Head left', (90, 30), font, 2, (255, 255, 128), 3)    cv2.putText(img, str(ang1), tuple(p1), font, 2, (128, 255, 255), 3)  cv2.putText(img, str(ang2), tuple(x1), font, 2, (255, 255, 128), 3)  cv2.imshow('img', img)  if cv2.waitKey(1) & 0xFF == ord('q'):  break  else:  break  cv2.destroyAllWindows()  cap.release() |
| --- |

**audio.py**

| import speech\_recognition as sr  import pyaudio  import wave  import time  import threading  import os  def read\_audio(stream, filename):  chunk = 1024 # Record in chunks of 1024 samples  sample\_format = pyaudio.paInt16 # 16 bits per sample  channels = 2  fs = 44100 # Record at 44100 samples per second  seconds = 10 # Number of seconds to record at once  filename = filename  frames = [] # Initialize array to store frames    for i in range(0, int(fs / chunk \* seconds)):  data = stream.read(chunk)  frames.append(data)    # Save the recorded data as a WAV file  wf = wave.open(filename, 'wb')  wf.setnchannels(channels)  wf.setsampwidth(p.get\_sample\_size(sample\_format))  wf.setframerate(fs)  wf.writeframes(b''.join(frames))  wf.close()  # Stop and close the stream  stream.stop\_stream()  stream.close()  def convert(i):  if i >= 0:  sound = 'record' + str(i) +'.wav'  r = sr.Recognizer()    with sr.AudioFile(sound) as source:  r.adjust\_for\_ambient\_noise(source)  print("Converting Audio To Text and saving to file..... ")  audio = r.listen(source)  try:  value = r.recognize\_google(audio) ##### API call to google for speech recognition  os.remove(sound)  if str is bytes:  result = u"{}".format(value).encode("utf-8")  else:  result = "{}".format(value)    with open("test.txt","a") as f:  f.write(result)  f.write(" ")  f.close()    except sr.UnknownValueError:  print("")  except sr.RequestError as e:  print("{0}".format(e))  except KeyboardInterrupt:  pass  p = pyaudio.PyAudio() # Create an interface to PortAudio  chunk = 1024 # Record in chunks of 1024 samples  sample\_format = pyaudio.paInt16 # 16 bits per sample  channels = 2  fs = 44100  def save\_audios(i):  stream = p.open(format=sample\_format,channels=channels,rate=fs,  frames\_per\_buffer=chunk,input=True)  filename = 'record'+str(i)+'.wav'  read\_audio(stream, filename)  for i in range(30//10): # Number of total seconds to record/ Number of seconds per recording  t1 = threading.Thread(target=save\_audios, args=[i])  x = i-1  t2 = threading.Thread(target=convert, args=[x]) # send one earlier than being recorded  t1.start()  t2.start()  t1.join()  t2.join()  if i==2:  flag = True  if flag:  convert(i)  p.terminate()  from nltk.corpus import stopwords  from nltk.tokenize import word\_tokenize  import nltk  nltk.download('stopwords')  file = open("test.txt") ## Student speech file  data = file.read()  file.close()  stop\_words = set(stopwords.words('english'))  word\_tokens = word\_tokenize(data) ######### tokenizing sentence  filtered\_sentence = [w for w in word\_tokens if not w in stop\_words]  filtered\_sentence = []    for w in word\_tokens: ####### Removing stop words  if w not in stop\_words:  filtered\_sentence.append(w)  ####### creating a final file  f=open('final.txt','w')  for ele in filtered\_sentence:  f.write(ele+' ')  f.close()    ##### checking whether proctor needs to be alerted or not  file = open("paper.txt") ## Question file  data = file.read()  file.close()  stop\_words = set(stopwords.words('english'))  word\_tokens = word\_tokenize(data) ######### tokenizing sentence  filtered\_questions = [w for w in word\_tokens if not w in stop\_words]  filtered\_questions = []    for w in word\_tokens: ####### Removing stop words  if w not in stop\_words:  filtered\_questions.append(w)    def common\_member(a, b):  a\_set = set(a)  b\_set = set(b)    # check length  if len(a\_set.intersection(b\_set)) > 0:  return(a\_set.intersection(b\_set))  else:  return([])  comm = common\_member(filtered\_questions, filtered\_sentence)  print('Number of common elements:', len(comm))  print(comm) |
| --- |

**5.3.6 Face Recognition**

In addition to the above feature we added one more feature like while signup we take a few images of a user who is and while those images are trained by our model and while he/she was writing exam we were able to detect their names and easily identify if someone else is writing their exam .

**5.3.7 Speech to text and text to speech:**

Whenever a bot is conducting an HR round based on the recruiter's choice then the recruiter can give a set of questions to the bot to ask in a text format and the bot will convert text to speech. In the same way the answers of the interview will be answering those questions then speech is converted into text.

| Text to speech  from gtts import gTTS  #from google.colab import files  import IPython  d = {  "English":'en',  "Spanish":'es',  "Indian":'hi',  "Arabic":'ar',  "French":'fr',  "Chinese":'zh',  "Polish":'pl',  "Ukrainian":'uk',  "Afrikaans":'af'  }  text = """Tell me something about yourself please include your strengths,interests,goals,hobbies,projects and experience  """#@param {type:"string"}  Accents = 'English' #@param ["English", "Spanish","Afrikaans", "Indian", "Arabic", "Chinese", "Polish", "French", "Ukrainian"]  pytts = gTTS(text, lang=d[Accents])  pytts.save('audio\_generated.mp3')  IPython.display.Audio("audio\_generated.mp3")  # files.download("audio\_generated.mp3")  speech to text  import speech\_recognition as sr  r = sr.Recognizer()  mic = sr.Microphone()  print('start')  with mic as source:  audio = r.listen(source)  print('end')  print(r.recognize\_google(audio)) |
| --- |

**CHAPTER 6**

**RESULTS**

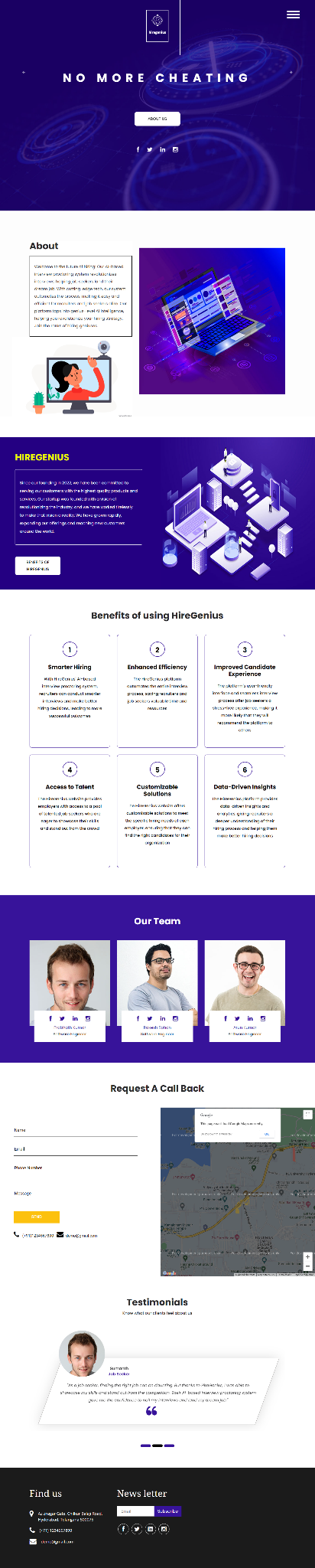
**6.1 Introduction**

We have created a web application as a prototype along with AI based models to create a strong AI based Proctoring web application.

**6.2 Results**

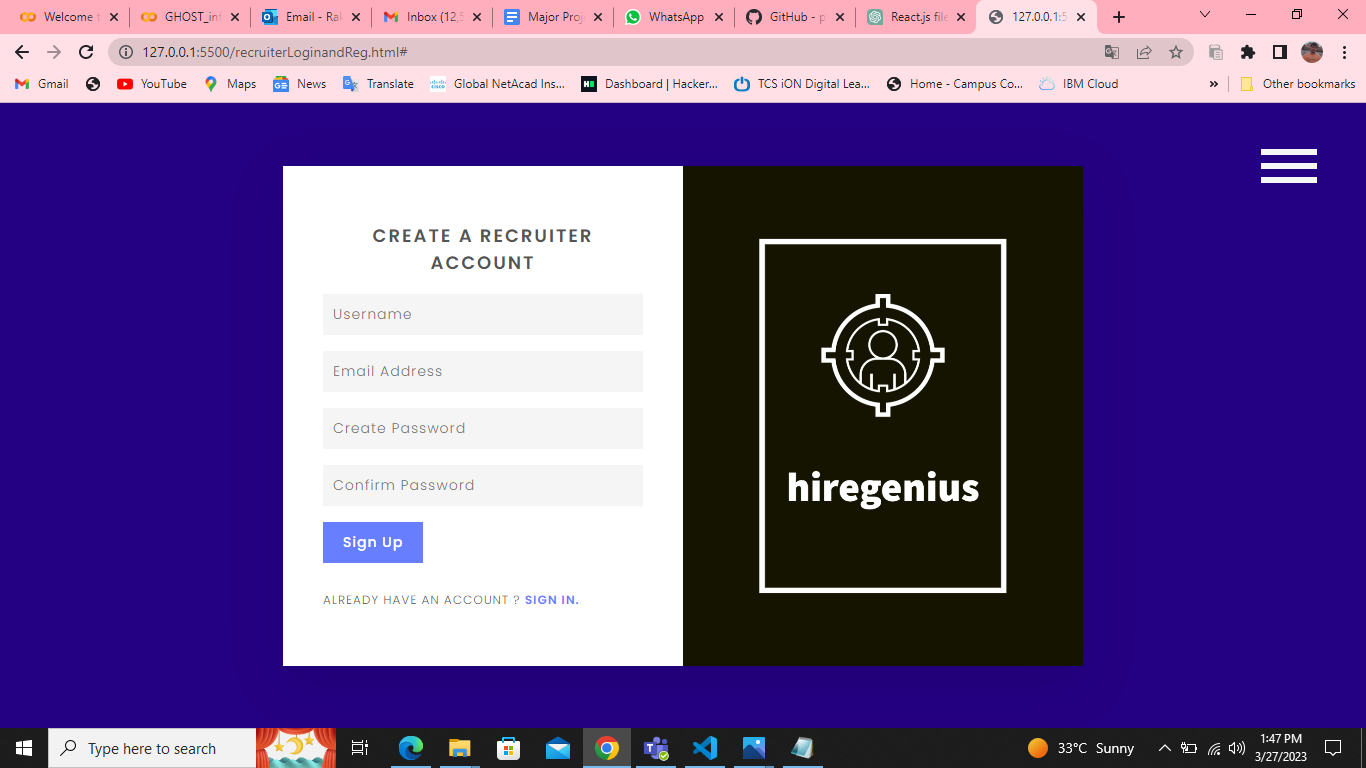
**6.2.1 Home Page**

When you visit the website, the home page is shown first. In the home page we show About Project,testimonials,newsletter,our benefits of hiregenius,login and our Team.



**Fig 6.1:** Home page

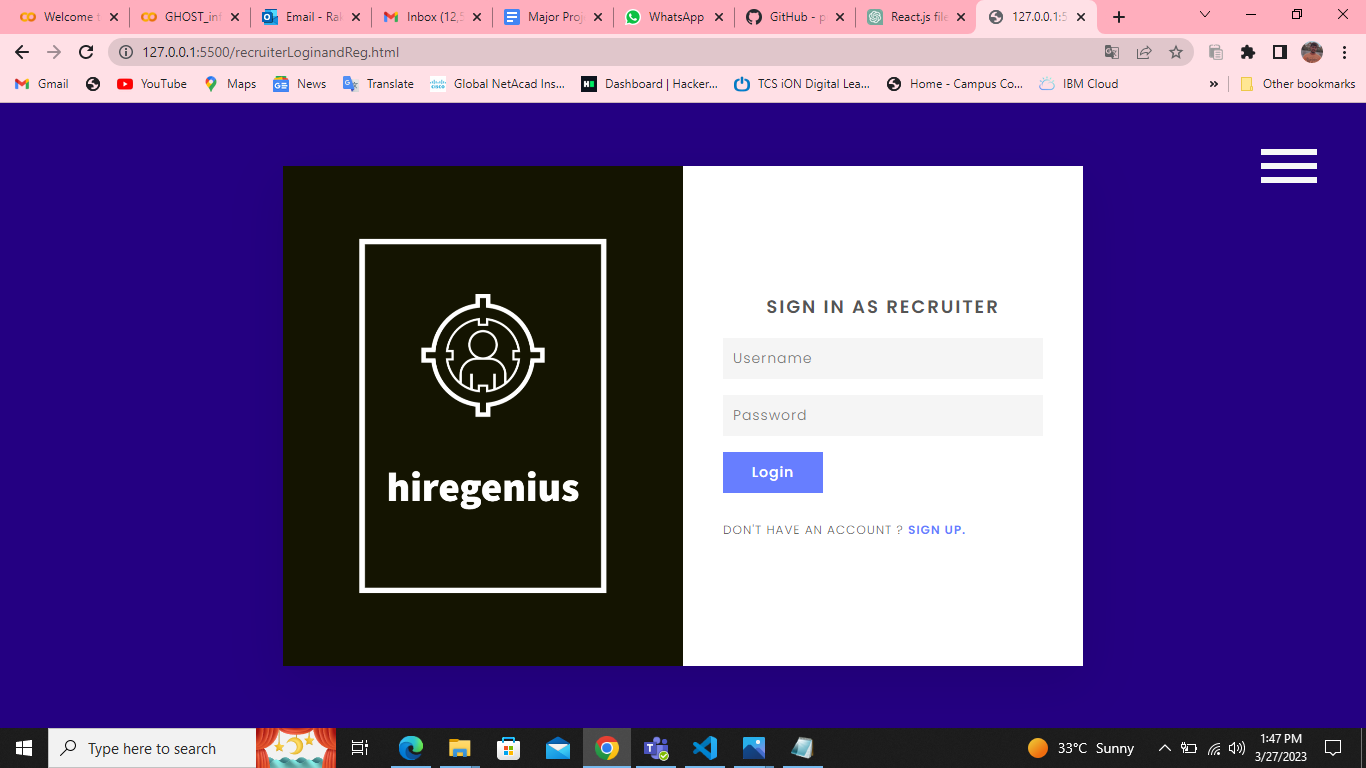
**6.2.2 Sign Up Recruiter page**

This page is used to take registration process for recruiter.

**Fig 6.2:** Sign up recruiter page

**6.2.3 Login Recruiter Page**

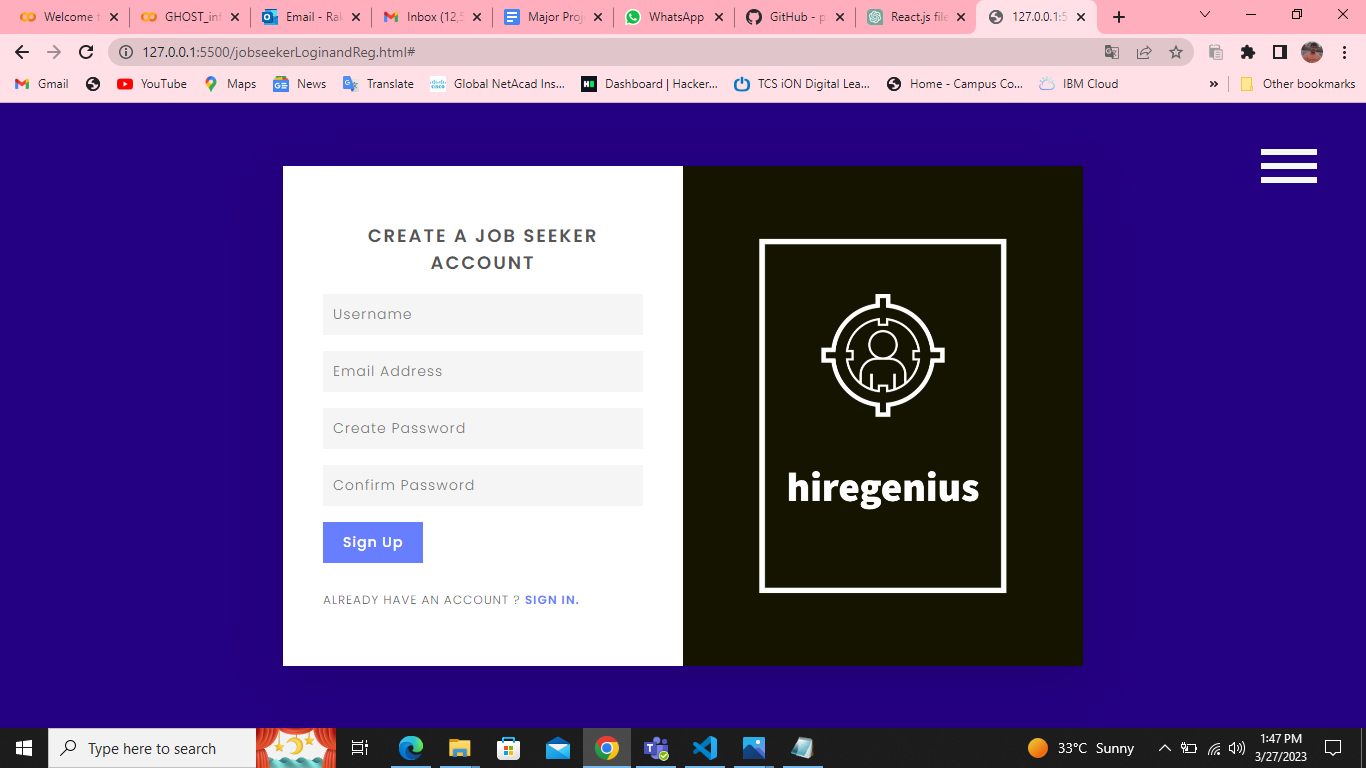
This page is used to authenticate the recruiter.If they provide correct credential then login will be successful.



**Fig 6.3:** Login Recruiter page

**6.2.4 Sign Up Job Seeker Page**

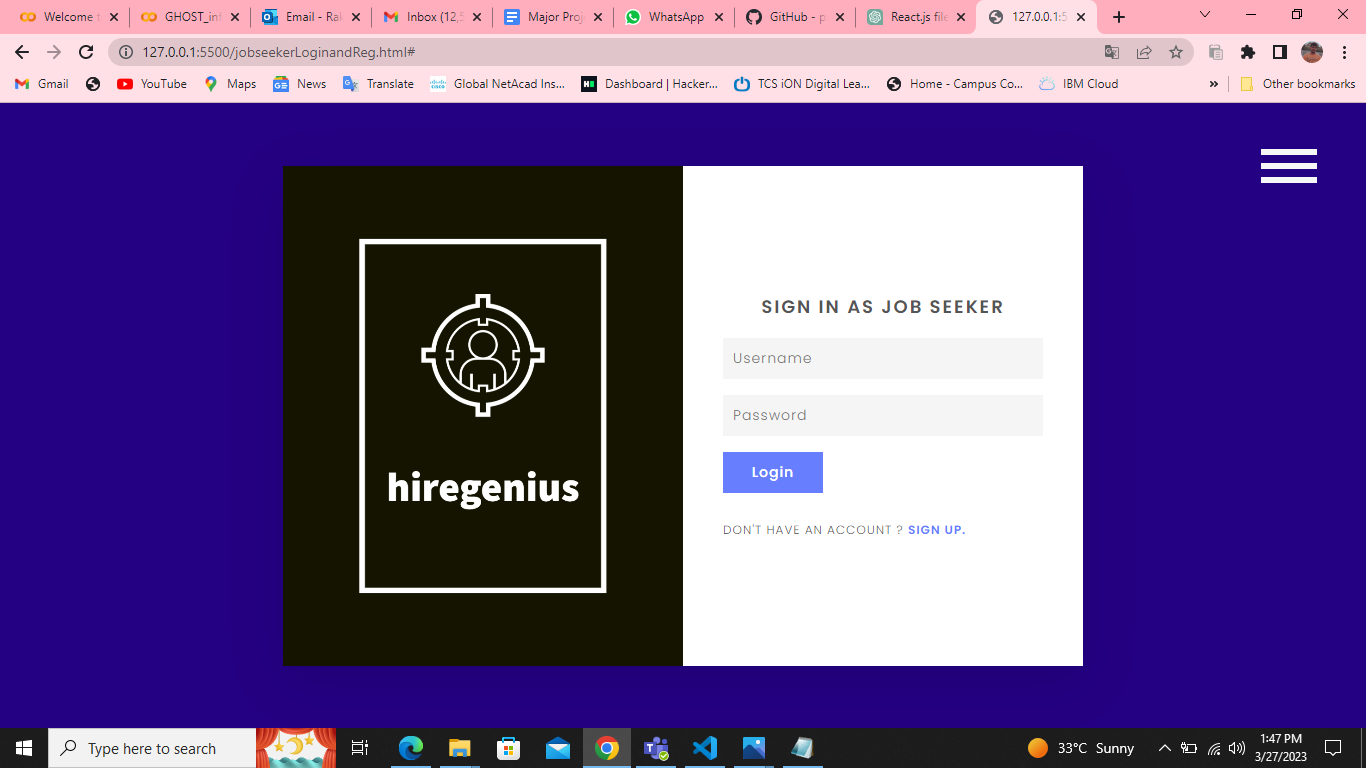
This page is used for registration process for job seeker.

****

**Fig 6.4:** Sign Up Job Seeker page

**6.2.5 Login Job Seeker Page**

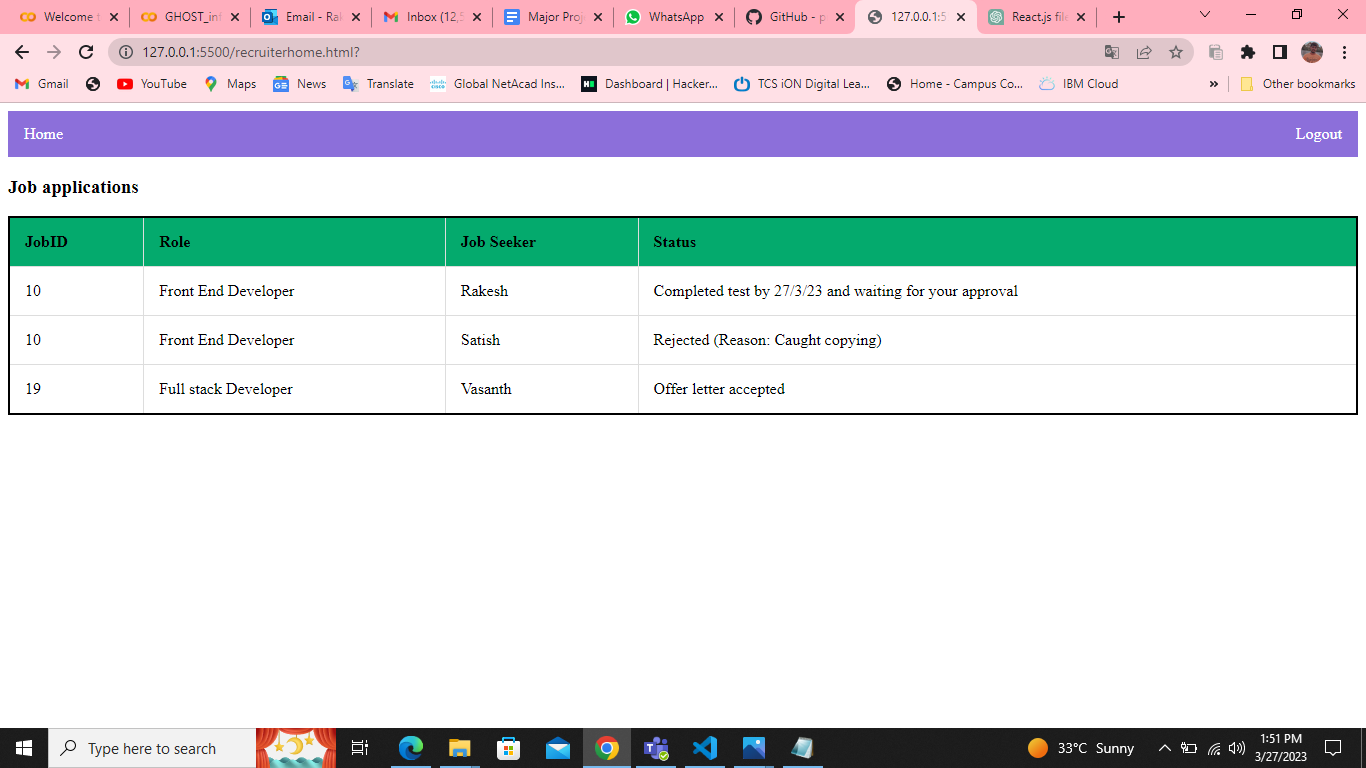
This page is used to login for job seekers to our hire genius platform.



**Fig 6.5:** Login Job Seeker page

**6.2.6 After Login Page Recruiter Side**

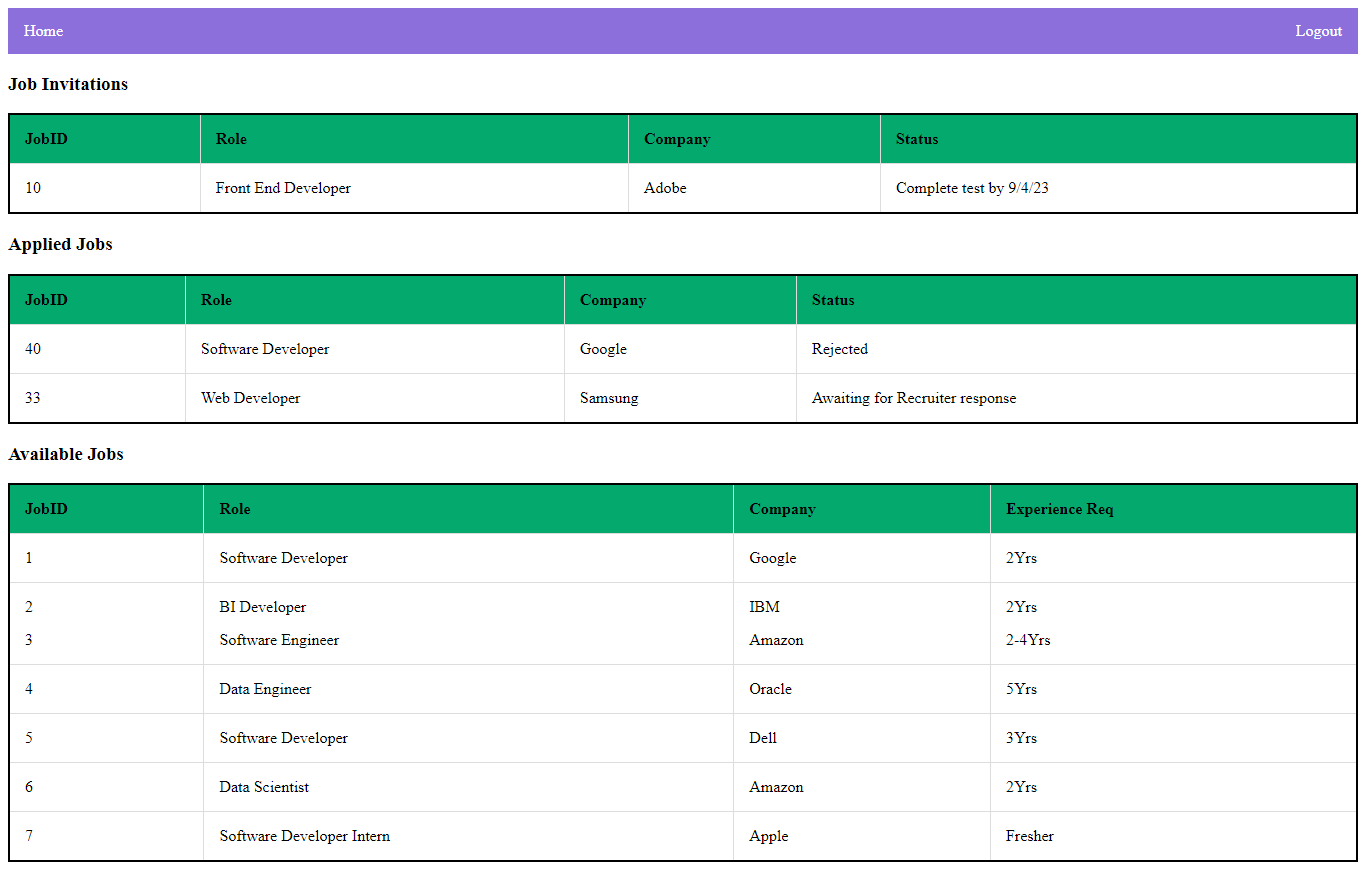
After the recruiter login successfully then they can post job applications and also the logout option will be available on the top right corner.



**Fig 6.6:**After Login Page Recruiter Side

**6.2.7 After Login Page JobSeeker Side**

This page appears after the jobseeker login sucessfully .This page shows job invitations,Applied Jobs and Available jobs.They can even logout from the page which is available on the top right corner.



**Fig 6.7:** After Login Page Job Seeker Side

**6.2.8 Exam Time Quiz**

Quiz can start by clicking start button then camera opens along with restricting multiple tabs and once quiz completes then recruiter can see the results at their side dynamically now need to integrate this models with frontend and dynamically adopt these AI features and generate report of a particular user at recruiter side where recruiter can know the all details like both non verbal and verbal behaviors will be monitored with our strong AI based models.

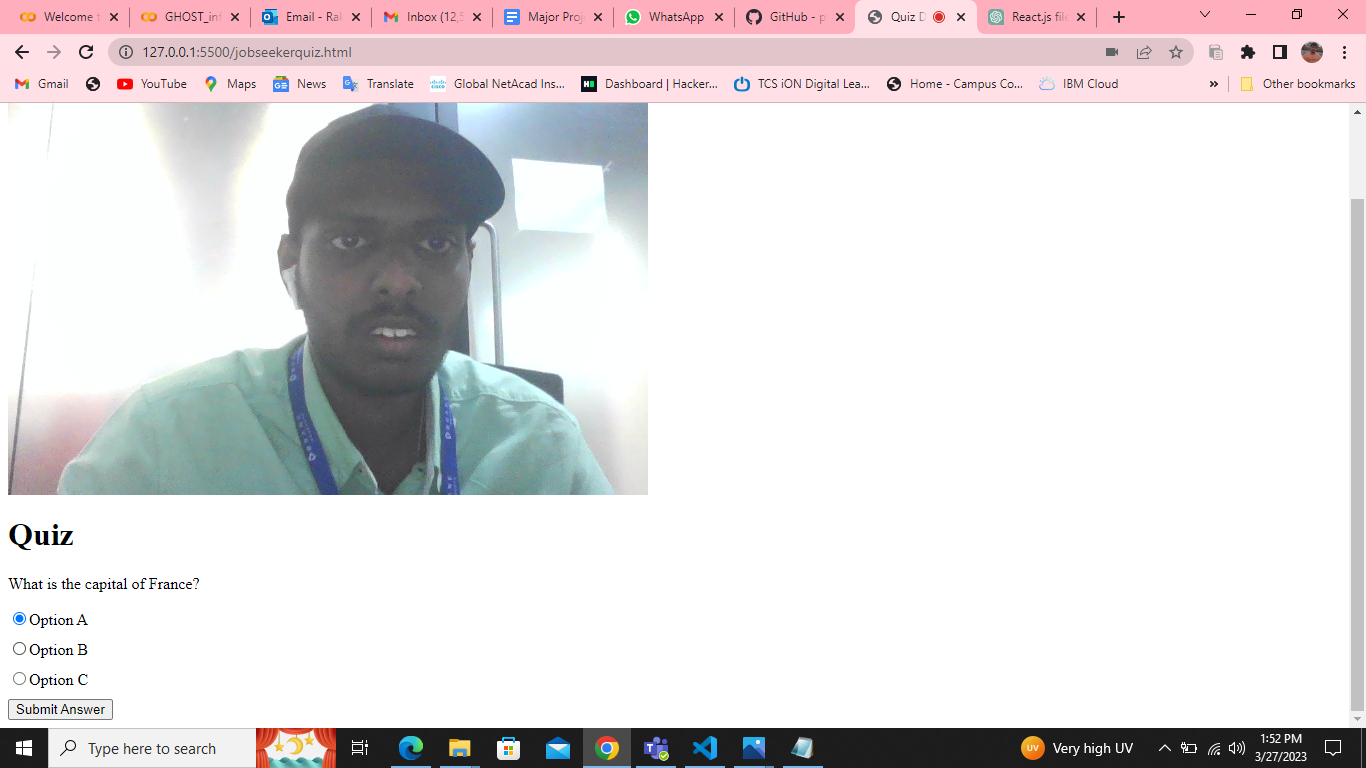
* 

Fig 6.8 Exam demo 1

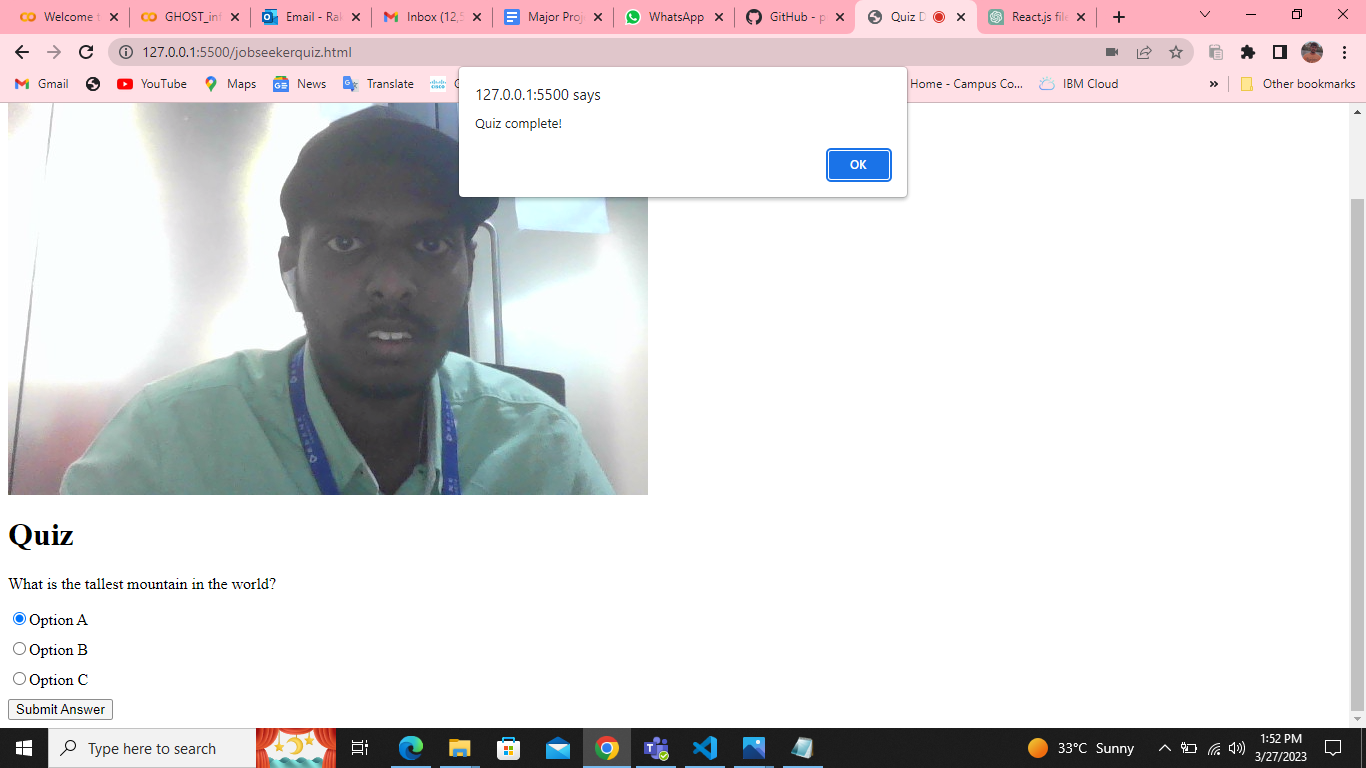


Fig 6.9 Exam demo2

**6.3 Conclusion**

This chapter includes screenshots of our whole web application as well as the overall workflow of our project at both views recruiters side and job seekers side.

**CHAPTER 7**

**CONCLUSIONS**

**7.1 Conclusion:**

Finally we conclude that we tried to build a strong AI proctoring system and build opencv models which monitor non verbal communication of users.

**7.2 Future Scope:**

AI in Proctoring Systems has a lot of future scope like everyone is trying to cheat in online exams so we tried to bring Strong AI based Proctoring and minimize the chance of cheating.

In terms of our project scope or expansion we can add databases and build more complex and accurate openCV models with the help of ensemble learning,deep learning techniques and increase the overall performance of web applications like automating the whole recruitment process without human intervention with help of Strong AI techniques.

**CHAPTER 8**

**REFERENCES**

**8.1 References**

We took the reference of a few research papers published by IEEE.

<https://link.springer.com/article/10.1007/s10639-021-10597-x>

<https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3866446>

<https://ieeexplore.ieee.org/document/9725547>

<https://ijcrt.org/papers/IJCRT2201560.pdf>

<https://www.kscst.org.in/spp/45_series/SPP45S/01_Seminar_Projects/094_45S_BE_4635.pdf>

<https://doi.org/10.1109/TMM.2017.2656064>

<https://doi.org/10.1016/j.chbr.2020.100033>

<https://doi.org/10.1007/978-3-319-97807-9_8>