



Department Information Science & Engineering

2021-2022

A Project Report on

**“Secured Authentication Process for Physically  
Challenged People Using Morse Code and Machine  
Learning Algorithm”**

Submitted in partial fulfilment for the award of the degree of

**BACHELOR OF TECHNOLOGY  
IN  
INFORMATION SCIENCE AND ENGINEERING**

Submitted by

**ABHISHEK N  
18BTRIS002**

**PRANAV M  
18BTRIS029**

**SATHVIK K C  
18BTRIS042**

**MANSI D  
18BTRIS023**

Under the guidance of

**Prof. Soumya K N**

Assistant Professor

Department of Information Science & Engineering

Department of Information Science &  
Engineering

Jain Global campus  
Kanakapura Taluk –  
562112 Ramanagara  
District  
Karnataka, India

## CERTIFICATE

This is to certify that the project work titled “**Secured Authentication Process for Physically Challenged People Using Morse Code and Machine Learning Algorithm**” is carried out by **Abhishek N(18BTRIS002),Pranav M(18BTRIS029),Sathvik KC (18BTRIS042),Mansi D (18BTRIS023)**, a bonafide students of Bachelor of Technology at the Faculty of Engineering & Technology, Jain University, Bangalore in partial fulfillment for the award of degree in Bachelor of Technology in Information Science & Engineering, during the year **2021-2022**.

**Prof. Soumya K N**

Incharge, Assistant Professor  
Dept. of ISE,  
Faculty of Engineering &  
Technology,  
Jain University  
Date:

**Dr. Arun N**

Head of the  
Department,  
Dept. of ISE,  
Faculty of Engineering &  
Technology,  
Jain University  
Date:

**Dr. Hariprasad S A**

Director,  
Faculty of Engineering  
& Technology,  
Jain University  
Date:

Name of the Examiner

Signature of Examiner

1.

2.

# DECLARATION

We, **Abhishek N(18BTRIS002), Pranav M(18BTRIS029), Sathvik KC (18BTRIS042),Mansi D (18BTRIS023)**, are students of eighth semester Boteh in **Information Science & Engineering**, at Faculty of Engineering & Technology, **Jain University**, hereby declare that the project titled “**Secured Authentication Process for Physically Challenged People Using Morse Code and Machine Learning Algorithm**” has been carried out by us and submitted in partial fulfilment for the award of degree in **Bachelor of Technology in Information Science & Engineering** during the academic year **2018-2022**. Further, the matter presented in the project has not been submitted previously by anybody for the award of any degree or any diploma to any other University, to the best of our knowledge and faith.

Signature

Name1: Abhishek N  
USN: 18BTRIS002  
Name2: Pranav M  
USN: 18BTRIS029  
Name3: Sathvik K C  
USN: 18BTRIS042  
Name4: Mansi D  
USN: 18BTRIS023

Place: Bangalore  
Date:

## ACKNOWLEDGEMENT

*It is a great pleasure for us to acknowledge the assistance and support of a large number of individuals who have been responsible for the successful completion of this project work.*

*First, we take this opportunity to express our sincere gratitude to Faculty of Engineering & Technology, Jain University for providing us with a great opportunity to pursue our Bachelor's Degree in this institution.*

*In particular we would like to thank **Dr. Hariprasad S A, Director, Faculty of Engineering & Technology, Jain University** for his constant encouragement and expert advice.*

*It is a matter of immense pleasure to express our sincere thanks to **Dr. Arun N, Head of the department, Information Science & Engineering, Jain University**, for providing right academic guidance that made our task possible.*

*We would like to thank our guide **Prof. Soumya K N, Assistant Professor, Dept. of Information Science & Engineering, Jain University**, for sparing his/her valuable time to extend help in every step of our project work, which paved the way for smooth progress and fruitful culmination of the project.*

*We would like to thank our Project Coordinator **Prof. Soumya K N** and all the staff members of Information Science & Engineering for their support.*

*We are also grateful to our family and friends who provided us with every requirement throughout the course.*

*We would like to thank one and all who directly or indirectly helped us in completing the Project work successfully.*

*Signature of Students*

# **ABSTRACT**

Data science is a collaborative study of data inference, technology and algorithm development for the purpose of solving analytical complex problems. Data science are being used in numerous industries for the purpose of analyzing and decision taking. Data science is field that makes use of Artificial Intelligence for the purpose of prediction. Image processing, Speech recognition and audio processing are some of the practical applications of Artificial Intelligence. Security and Authentication is one of the fields of Artificial Intelligence and Data Science facing problems. So here in this project we are providing a Unique solution for the security of mobile applications and web applications. Here real time eye tracking is been used for the purpose of password authentication using Morse code instead of Numbers. Real time eye tracking refers to finding the eye location on the image frames of eyes and tracking center of eyes over time. Password will be done and set in the form of Morse code. Morse codes will be tracked by our eyes with respect to time. Morse code are the numbers that are represented by the symbol dots and dash. A person or a hacker cannot guess or identify the password which is in the form of morse code. The pattern of morse code set to each number can also be altered based on our needs which again makes it hard for anyone to crack to password. A physically challenged individual will also be able use the application security since we design It in a way where the password is entered only through eyes.

# TABLE OF CONTENTS

List of Figures

Nomenclature used

## **Chapter 1**

### **1. INTRODUCTION**

#### **1.1 Limitations**

#### **1.2 Problem Definition**

#### **1.3 Objectives**

#### **1.4 Methodology**

## **Chapter 2**

### **2. LITERATURE SURVEY**

## **Chapter 3**

### **3.TOOL DESCRIPTION**

#### **3.1 Python 3.8**

#### **3.2 OpenCV**

#### **3.3 Tkinter**

#### **3.4 Pandas**

#### **3.5 PyOT5**

#### **3.6 Dlib**

## **Chapter 4**

### **4.DESIGN OVERVIEW AND IMPLEMENTATION**

#### **4.1 Design Overview**

## **4.2 Software Algorithm**

## **4.3 Use Case Diagram**

## **4.4 Data Flow Diagram**

# **Chapter 5**

## **5.Results and Conclusion**

### **5.1 Results and Conclusion**

### **5.2 Conclusion and Future Scope**

## **REFERENCES**

## **APPENDIX – I**

## **INFORMATION REGARDING STUDENTS**

## **BATCH PHOTOGRAPH ALONG WITH GUIDE**

## LIST OF FIGURES

Fig. No.	Description of the figure	Page No.
1.4. a	Workflow Of Face Recognition	13
1.4. b	Facial Points	13
1.4.c	EAR With 6 Points	14
1.4. d	EAR Formula	14
1.4. e	Virtual Keyboard	15
1.4. f	Workflow of Morse code Detection	15
2.1. a	Yen Hough Transform	21
3.1. a	Python Logo	24
3.2. a	OpenCV Logo	25
3.3. a	Tkinter Logo	26
3.4. a	Pandas Logo	27
3.5. a	PyQT5 Logo	28
3.6. a	Dlib Logo	29
4.2. a	Haar Features	31
4.2. b	Haar Pixels	31
4.2.c	7*7 Cells on Image	32
4.2. d	The original face image followed by the weighting scheme for the 7*7 cells.	33
4.3. a	Use Case Diagram	34
4.4. a	Data Flow Diagram	35
5.1. a	Starting GUI	36
5.1. b	Registration	36
5.1.c	Filling Details	37
5.1. d	Training	37
5.1. e	Registration Successful	38
5.1. f	Login Window	38
5.1. g	Webcam and Virtual Keyboard	39
5.1.h	Detection and entering	39
5.1. i	Final output	40



## NOMENCLATURE USED

EAR	Eye Aspect Ratio
LBPH	Local Binary Pattern Histogram
GUI	Graphical User Interface
MAR	Mouth Aspect Ratio
CSV	Comma Separated Values

# Chapter 1

## 1. INTRODUCTION

Artificial intelligence is a class of algorithms that attempt to replicate human intelligence or the ability to solve problems by computing. There are many different approaches with artificial intelligence, each using a different type of computer model. An example of Artificial Intelligence is an algorithm controlling a self-driving car. Artificial intelligence often refers to "machine learning" because most complex AI systems are based on data mining and pattern recognition algorithms, rather than programming. Machine learning is a subfield of computer science that generally deals with using statistical and other mathematical analysis to improve the efficiency, effectiveness, and speed of automated software. Artificial intelligence is commonly addressed in the context of machine learning. Machine learning can also be seen as an extension of computational statistics and information theory in which traditional approaches have been augmented by methods based on pattern recognition, data fitting or approximation to model a real-world phenomenon through hypothesizing and testing until finding a suitable solution.

Machine learning is the study of computer algorithms that automatically improve with time and experience.

It addresses the question - "How can computers become intelligent?"

In order to achieve this intelligence, machine learning uses algorithms (specialized programs) to explore data and make predictions from it. These predictions are based on previous outcomes, which can help to explain how our system will behave in future situations. As time progresses, machine learning improves its ability to solve complex problems by processing more data and receiving feedback from the environment without being explicitly programmed in advance.

Morse code is the most common form of communication from one point to another via Morse code. Morse code is a variant of the International Morse Code that uses long and short pulses, called dots and dashes respectively, to spell out letters and numbers. It is still in use by some amateur radio operators, although it has been largely replaced by teleprinter codes for commercial use; see below for an explanation of how it became mostly obsolete for commercial purposes.

In its original form, Morse code was based on hand-held electromagnetic signalers similar to those developed by Samuel Morse in 1832; it was first used by amateurs in 1880. It was rapidly adopted by others, and came to be the international standard language for telegraphy. However, commercial use of the systems declined following the introduction of voice radio in the 1930s because the

expense of maintaining a transmitter and receiver in a commercial installation made Morse impractical for business users.

Software Security is a field which is facing challenges day by day, new kind hackings keep on happening so providing security for all the software so that users can use without any interruption is a challenge for leading security companies. Nowadays passwords are easily leaked and most passwords are being hacked without the knowledge of user. This is one of the major issues in present world where everything is about data, it is an even bigger challenge for physically disabled people to face the security issues since people who are physically disabled are always being dependent on others for entering any kind of credentials like username and password. So, our project helps physically disabled to enter password on their own by using blinks of eye. Passwords are usually set using numbers i.e., 0-9. A standard password will have 4 or 6 digits so the sequences of 4 and 6 digits can be found out easily, for that particular reason in this process the numbers 0-9 are assigned with morse codes ['. ' & '- '].

## 1.1 Limitations

**Problem Formulation:** Security using Morse code is new research where morse code is used as password, this is solving problem of today's security where credentials are alphabets and numbers.

**Time consumption:** International Morse code is having two symbols (.'. ' & '- '). Since we are using numbers from 0-9 there should be different combination or sequence of dots and dash which resulted in using five symbols for each number. So, if the password length is 4 then we have to blink for 20 times which will consume lot of time.

**Eye Visibility:** After several tests it was determined that the eyes are detected only when the illumination of the light is more than 60%. If a user is wearing sunglasses or some kind of eye protection glasses which aren't transparent then the eyes won't be detected.

**Person with eye issue :** A person having eye issues cannot go through this security because the model which we have trained will detect both left and right eyes .

## **1.2. Problem Definition**

To offer a platform where a handicapped person can efficiently go through security process, this will tackle the problem of motor disabled people who cannot access devices on their own. In today's innovative world most of the applications and websites are using numbers and alphabets, parallelly the thefts are also being innovative so we used more code, where a person should first find the password and then understand the sequence of the morse symbols we have assigned to each number which is indirectly two factor authentication in a single step.

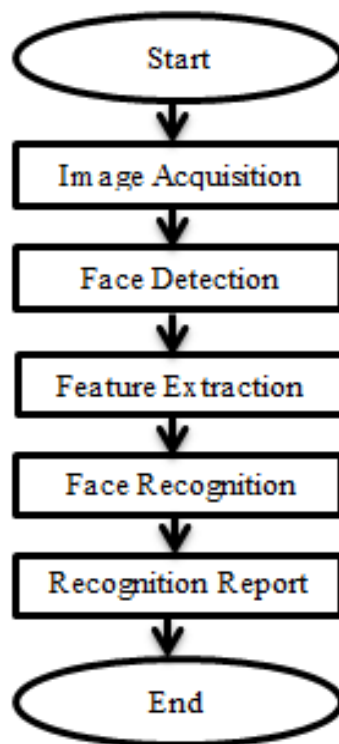
## **1.3. Objectives**

- To create the security system where a user can enter password without any physical contact with the device.
- To create an authentication process which can also be used by a person who is not completely blind.
- To make sure the required parts of the face and eyes are detected accurately for the purpose of entering the password.
- To create a system where shoulder surfing and thermal impersonating can be avoided

## **1.4 Methodology**

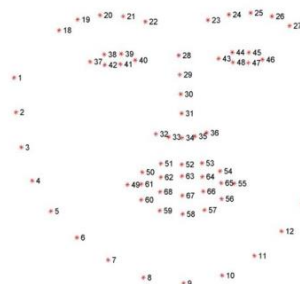
Secured Authentication Process for Physically Challenged People Using Morse Code and Machine Learning Algorithm uses one algorithm Haar Cascade the reason of using this algorithm is explained in detail by considering technical aspects.

The purpose of using Haar Cascade is because it is the best and widely used algorithm when it comes to object detection. Object detection is the initial phase of our project where the face and eyes are the objects; this algorithm not only gave us some good results but also increased the accuracy while detecting. Developed version of Object detection will give various kinds of data like EAR and Number of Faces.



**Figure 1.4.a Workflow Of Face Recognition**

Dlib library is being used for face detection, this library has a frontal face detector function which will verify the facial point, A trained model is used which has facial points the dlib library will take the frames captured and will detect the facial points when all the facial points are detected then it concludes that the face is present.



**Figure 1.4.b Facial Points**

Parallelly the eyes are detected. EAR eye aspect ratio is determined to check if the eyes are closed or not. While implementing and finding the face facial landmarks are used which includes points that are on eyes, each eye comprises of 6 points as shown below.



**Figure 1.4.c EAR With 6 Points**

A vertical line from top of the eye to bottom of eye is drawn that is achieved by finding the midpoint of upper two point and midpoint of lower two points, followed by a horizontal line from left to right is drawn. When the eyes are closed the vertical line distance will be lower than the distance of vertical line when opened. This determines whether eyes are closed or not.

$$EAR = \frac{||p_2 - p_6|| + ||p_3 - p_5||}{2||p_1 - p_4||}$$

**Figure 1.4.d EAR Formula**

After detecting the eyes, we created a virtual keyboard which has only three keys i.e, '.', '-' & '<'.



Figure 1.4.e Virtual Keyboard

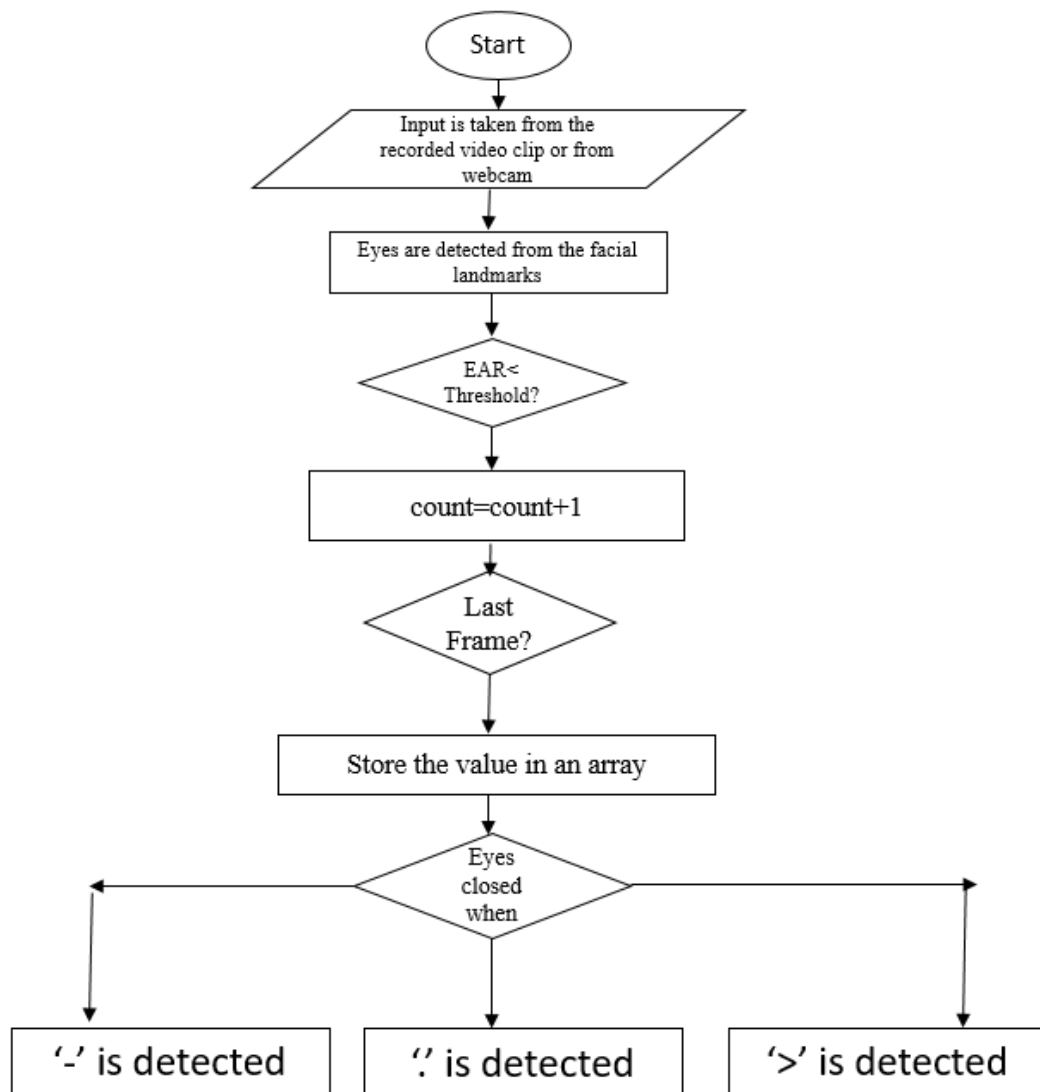


Figure 1.4.f Workflow of Morse code Detection

The virtual keyboard is set with timer where the cursor will point to each symbol for a second.

When the eye is blinked then the cursor is at a particular symbol then that symbol will be taken as the morse code, after entering five of the morse code it will be converted into a number this number will be verified with the password entered at the registration time.



## Chapter 2

### 2. Literature Survey

**[1]Morse Code Detector and Decoder using Eye Blinks** written by S. M, N. Kolkar, S. G. S and K. Kulkarni. This paper is mainly based on detecting morse code through eye blinks, The methods they have used are Eye localization, EAR –Eye Aspect Ratio, Detecting Morse code from Blinks One of the most difficult problems for people with disabilities is communicating with the outside world or their caregivers. The authors offer a real-time technique for detecting morse code from a series of eye blinks captured by a live camera in this paper. The proposed approach uses eye aspect ratio to detect eye landmarks and evaluate the level of eye opening (EAR). The low-cost software precisely decodes morse code based on the length of the eyes closed or opened, and then converts it to English. As a result, this system is designed to provide an alternative mode of communication for people with disabilities, as well as to transmit confidential communications.

The project's first stage is to identify ocular landmarks. They do this by capturing frames from a live video feed or a previously recorded video clip. `ret` is a Boolean variable that returns true if the frame is available, whereas `frame` is an image array vector recorded based on the default frames per second defined directly or implicitly. To access camera more efficiently and quickly, we'll use the `im-utils` library. The video stream will play in an indefinite loop until the video is finished. If the video should be paused in the middle, then the `q` key on our keyboard is used. The video frames are shrunk and transformed to greyscale images.

EAR, a fundamental component of this technique, can be utilized to assess whether or not a blink has occurred. Six two-dimensional points depict each of the eyes.

It is observed that the blink pattern is turned into morse code after collecting it. The dots usually symbolize a quick and swift signal. Dashes, on the other hand, are longer and indicate a slow signal.

A '.' (dot) is recognized when the user closes his eyes for 1 second, and a '-' (dash) is identified when the user closes his eyes for 2 seconds, based on the number of frames.

A dot represents one unit, while a dash represents three times the dot. A short quiet equivalent to one dot is used to divide the dots and dashes. Quite equal to one dash separates characters in a word, and silence equal to seven dots separates words.

**[2] Real Time Human Face Detection and Recognition Based on Haar Features** written by

authors Asif Mohammed Arf, Debasish Bal, Mohammad Anisul Hasan discusses on detecting and recognizing face through Haar features, The methods used are image capturing, generating data set, feature extraction, face detection process, face recognition

One of the main advantages is the security will be strong when we use this process as it captures real time image which is cultivated by reconnaissance camera

Drawback is that ,this technology has various limitations, such as when the face is rotated or shifted, or when the face is obstructed, the system does not work effectively.

A method based on the approach to measure the effectiveness of different feature kinds has been developed in this study. Haar has eight distinct characteristics. Only six of them have been considered in this study. Using a hand-picked database, the implemented system was trained. This study includes photographs of people in various positions and lighting conditions. The detectors that were employed in this study were trained through a variety of classification stages, including weak and strong classifiers. The face will be recognized if the image passes through all phases of the detector. It will compare the locally stored database after detecting it. If it matches the database, the face will be automatically detected and the real-time output displayed.

To begin, a color image was captured for processing, and progress was made utilizing two-dimensional color space. The image has to be transformed to greyscale as a result. The abundance of types Haar feature has been converted to greyscale. Extracted concurrently over the clipped 24\*24 image The face was then discovered by sampling the extracted data image. Following that, the detected face was compared to the computer's stored face database. Finally, if a match is found with a database image, the algorithm will label the recognized face.

**[3] Drowsiness Detection System using Eye Aspect Ratio Technique** written by S. Sathasivam, A. K. Mahamad, S. Saon, A. Sidek, M. M. Som and H. A. Ameen is based on detecting if a person is in drowsiness state or not , the methods used here are Eyes Aspect Ratio (EAR), Python Programming for Proposed method.

This system detects the drowsiness which can help to avoid accidents.

The main disadvantage is that Detecting the yawning is not included in this system

Transportation is frequently used to allow users to travel easily from one location to another, whether for personal or official reasons. Traveling at rush hour or on a holiday exposes the driver to lengthy traffic jams, making the driver-tired owing to increased concentration and lack of sleep. This circumstance adds to an increase in the percentage of car accidents, with car driver weariness being the leading cause. Using the Eye Aspect Ratio (EAR) technique, an image detection drowsiness system is proposed in this study to detect the state of the car driver. A built system with a Pi camera, Raspberry Pi 4 and GPS module is used to detect and analyze the status of eye closure in real time. With the initial, wearing spectacles, dim light, and microsleep condition experimentally done effectively, this system is able to determine whether the driver is tired or not, giving 90 percent accuracy. This condition may cause drivers to become much more cautious. The majority of drowsiness detection systems rely on the detection of eye blink movement. Calculating EAR utilizing OPENCV platform and Dlib prediction, also known as Dlib pre trained Neural network-based prediction, can be used to detect recorded blinks using computer webcam. Using the EAR formula, the EAR may be determined using the eye coordinates returned from OPENCV. The EAR is constant initially and then rapidly drops near zero but increases again afterwards. This indicates that a blink has taken place.

**[4] REAL-TIME EYE BLINK DETECTION BASED ON PYTHON** written by authors Pengji RAN<sup>1,2</sup>, Haobo WANG<sup>1,2</sup> is based on detecting eye blink , the methods used here are Detect or locate the face , Detection and determination algorithm ,system implementation and testing.The author has described the following in this paper:

Face recognition and detection technology has become widely used in various fields as a result of continual invention and development. In this article, Python, OpenCV, Dlib, and other third-party libraries are used to collect a video of the driver's face from a mobile phone camera and detect changes in eye aspect ratio (EAR) and mouth aspect ratio (MAR) to determine the driver's condition. Finally, for this test, it generates a graph of the association between EAR and MAR over time. Many accidents are caused by traffic's rapid development. Statistics show that fatigued driving, intoxicated driving, and speeding are the leading causes of traffic accidents. According to statistics, accidents caused by fatigue driving account for between 7% and 40% of all traffic accidents. As a result, researching and implementing driver tiredness monitoring technologies is critical. Subjective detection and objective detection are the two most used strain detection methods currently. Because the subjective detection approach is based primarily on the driver's

---

subjective interpretation, its reliability is low, and it is difficult to establish and apply

People's faces change when they're fatigued. Blinking, yawning, and nodding are the most prevalent states. Because of the computational difficulty, this method only extracts the first two behaviors for determining driver fatigue level and ignores head attitude changes.

This study uses Dlib's "shape predictor 68 face landmarks. Dat" model to extract 68 important points of faces from the video stream.

The EAR varies up and down a particular value when the human eye is open; in theory, it should be close to zero. When  $EAR = EAR \text{ THRESHOLD}$ , the eye is considered closed and blinking (for example, the empirical value 0.3, which is best computed according to the individual data collection).

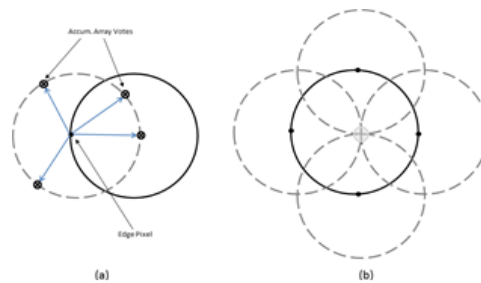
**[5] Driver's real-time Drowsiness Detection using Adaptable Eye Aspect Ratio and Smart Alarm System** written by authors Janki Chandiwala, Shrushti Agarwal discusses about drowsiness detection, the methods used here are Pre-processing, Data gathering, Detection system, Alarm system.

The most important problem to address is that the driver should be urged to take precautionary measures well in advance. This research proposes a radical solution by giving a technique for detecting the problem Sleepiness in the driver can be identified and a timely warning provided. The study proposes a non-intrusive method that takes into factors which considers the Eye Aspect Ratio (EAR) and the Mouth Aspect Ratio (MAR).

The face detector in dlib is used to discover and locate the face in the picture after the image has been processed such that the face area is totally visible. The facial landmark that has been pre-trained. The dlib library's detector is used to estimate the location of an object. 68 (x, y)-coordinates that correspond to face features, facial landmarks are located using structures on the face. The system then follows the coordinates' location. To focus, creates a bounding box around the face correctly. The coordinates of both eyes and mouth are extracted, then utilize a convex hull to enclose the concentrated region into a polygon. Following that, they drew outlines around the eyes by connecting the locations along the river's edge and mouth region visual form analysis boundary. As soon as the system is turned on, that is, when the car is started and a face is recognized, they determined the driver's initial Eye Aspect Ratio (EAR). This is what we do. Because a single EAR cutoff number might be confusing for those with particular genetic mutations, this can lead to erroneous forecasts. persons of various nations or diseases

**[6] Smart Eye-Tracking system** written by authors Aniwat Juhong, T. Treebupachatsakul and C. Pintavirooj is based tracking eye by using Circular Hough transform, Eye blink detection, Eye position Control,

The Yen Hough Transform (CHT) is an algorithm for detecting circles. First, the Huff gradient method is used to detect the edges of the circle, and then a separate circle is created for each pixel on the edge of this circle, so the position where the most pixels are accumulated in this circle generation is, Will be the center of the required circle. Detect as shown in Figure, because the eyeball is circular, this technique can be used to identify the eyeball.



**Figure 2.1.a Yen Hough Transform**

**[7] Review and Comparison of Face Detection Algorithms** written by authors Kirti Dang, Shanu Sharma is a comparison of different face detection algorithm. Algorithms that are discussed here are Viola Jones Face detection Algorithm, Successive Mean Quantization Transform (SMQT) Features and Sparse Network of Winnows (SNOW) Classifier Method, Neural Network-Based Face Detection, Support Vector Machines.

With the huge increase in videos and images Emergency database requires automatic understanding and manually review data with intelligent systems out of reach. Reduce it by one domain, one of the most specific objects that can be drawn in the images are people and faces. Facial recognition becomes a challenged by the increasing use of applications. That is First step towards face recognition, face analysis and other person detection facial features. In this article, different facial recognition algorithms, discussion and analysis like Viola Jones, features of SMQT & SNOW classifier, face recognition based on neural network and Support for face recognition based on Vector Machine. All these faces detection methods are compared according to their accuracy and call back value is calculated by software DetEval processing Exact values of

bounding boxes around faces will be provided accurate results.

The above experiment is done, it can be estimated by depending on accuracy and salvage value, higher value is by Viola Jones, followed by SMQT and SWEET Features. Then the classification method is Neural Network for face detection and finally the support vector machine. So best of all these algorithms are Viola Jones for face recognition. Because future work, many other object algorithms can be compared using this procedure.

**[8] Model-based Separation, Detection, and Classification of Eye Movements** written by Federico Wadehn, Thilo Weber, David J. Mack, Thomas Heldt, and Hans-Andrea Loeliger discusses about eye movements, the methods used here are Oculomotor Plant for Horizontal Eye Movements, Saccade Controller, SPEM Controller, FEM Generator, Blink Artifact Model, Complete Model.

They present a physiologically driven eye movement analysis framework for model-based separation, detection, and classification (MBSDC). Eye movements by estimating controller kinematics and neural signals for glide, tracking and immobilization Eye movement in a mechanical eye movement pattern. A system capable of separating and analyzing these eyes independent movement. Extend established eye muscle model to horizontal eye movement by neural control signals and vibration artifact patterns. Because kinetic inference (position, velocity, acceleration, force) Neural controller signals from eye position data, Using Kalman smoothing and estimating sparse entries Technology. Estimated signals used to detect start and end points are categorized and categorized Register for recoil, smooth tracking, fixation, impulsive movement and flash.

Simulation process: Data that is a speed configuration recovery error, approximately half of the error values obtained by commonly used numerical identification and filtering methods. Accurate signal separation was observed in experiments using smooth tracking data from human subjects. In addition, in neural records from nonhuman primates, the estimated signal from the neural controller matches the actual recording very well.

**[9] An Eye-Tracker-Based 3D Point-of-Gaze Estimation Method Using Head Movement**

written by authors wudthipong pichitwong and kosin chamnongthai talks about tracking eye based on Gaze, the methods and requirements used here are eye tracker calibration, position approximation, conversion from eye tracker coordinates to model coordinates, finding the model fitting parameters

This paper proposes a method to obtain a 3D point sensor (POG) estimated using head movement and 3D pupil position 2D and POG data on virtual screen acquired by eye Tracking Devices. Since current eye trackers can detect 3D pupil center data and 2D position data for POG on virtual screen, can draw lines central line of sight between two virtual students POG 2D filtered. If the eyes and head change positions, another line of vision is formed. The proposed method assumes that a person will generally move the head and eyes the least a short distance looking at a fixed point in 3D, or consciously or unconsciously, this automatically leads to in the formation of many passing glances 2D POG on virtual screen and intersect with 3D POG. Therefore, the 3D POG can be found from the intersection of several lines of sight formed by the motion of head and eyes. If the person's gaze moves in another direction 3D POG, the point of intersection of the lines of sight also new movements and 3D POG can be easily detected from new line of sight. In tests using device with five male participants and five female participants, the results show that the proposed method can measure 3D POG with average distance error of 13.36cm, 7.58cm, 5.72 cm, 3.97 cm and 3.52 cm for head travel distance 1 cm, 2 cm, 3 cm, 4 cm and 5 cm, respectively.

**[10] Eye movements during change detection: the role of depth of field** written by authors Tingting Zhang<sup>1</sup>, Ling Xia<sup>1</sup>, Xiaofeng Liu<sup>1</sup>, Xiaoli Wu discussed in depth about eye movements the methods and requirements used here are Participants, Apparatus, Stimuli, Design, Procedure. This study examines eye movements when detecting changes in scenes with different depth of field. A non-subjective experiment was performed using a flicker pattern to induce variable blindness. This experience studied two main factors: depth of field and location. The Tobii X120 optometrist was used to record the participant's gaze motion when looking for changes in the flickering scene. It is concluded that shallow depth of field can be effectively directed the viewer's attention in the net area. The size of the depth of field cannot affect the change detection time while uniform opacity can make it easier to detect changes. All participants have normal or corrected visual acuity (measured on the Friborg scale vision test). Their color vision is also normal. The experiment was carried out in a normal working room. Participants were seated in front of a 21-inch LCD monitor, with remote desktop eye tracker, Tobii X120 in front of them. The screen resolution of the monitor has been fixed at  $1280 \times 800$  and the frame eye tracking frequency is 120Hz.

## Chapter 3

### 3. Tool Description

#### 3.1. Python 3.8



**Figure 3.1.a Python Logo**

Python is a simple language which syntax is similar language a c. It is easy to learn, effective programming language. It has efficient high-level information structures and easy and effective technique to object-oriented programming concepts. Python's fashionable syntax and dynamic typing, collectively with its interpreted nature, make it a high-level language for scripting and rapid application building across various platforms.

Features of python 3.8

- Assignment Expressions: `=`. It is also known as walrus operator.
- Positional-only parameters (`/`) If you want to call a function in Python that accepts parameters then you can pass the arguments by position or by keyword.
- New date and datetime constructors are added for “fromisocalendar”
- Improved support for type hinting.
- Higher precision timing functions.



### **3.2. OPENCV**



**Figure 3.2.a OpenCV Logo**

OpenCV is a Python open-source library used for computer vision such as artificial intelligence, machine learning, and facial recognition.

In OpenCV, CV is an abbreviation for Computer Vision, defined as a research area that helps computers understand the content of digital images such as photos and videos.

The purpose of computer vision is to understand the content of an image. Extract the description from the image. This includes objects, text descriptions, 3D models, and so on. For example, a car can assist with computer vision that can recognize and act on various objects on the road, such as traffic lights, pedestrians, and road signs.

Computer vision allows computers to perform the same type of tasks as humans with the same efficiency.

There are two main tasks defined below:

- **Object Classifier:** In object classification, we train a model on a dataset of specific objects, and the model classifies new objects in one or more of your training categories.
- **Object recognition:** In object recognition, our model identifies a particular instance of an object

### 3.3. TKinter



**Figure 3.3.a Python TKinter Logo**

The Tkinter Tutorial provides basic and advanced Tkinter Python concepts. Our Tkinter guide is designed for beginners and experts alike. Python provides the Tkinter standard library for creating graphical user interfaces for desktop applications. Developing desktop applications with python Tkinter is not a complicated task. An empty top-level Tkinter window can be created by doing the following.

- Import the Tkinter module.
- Create the main application window.
- Add widgets like labels, buttons, frames, etc. at the window.
- Call the main event loop so that actions can take place on the user's desktop.

The Tkinter geometry specifies the method by using which, the widgets are represented on display. The python Tkinter provides the following geometry methods.

- The pack () method
- The grid () method
- The place () method

### 3.4. Pandas



**Figure 3.4.a Pandas Logo**

Python Data Analysis Library popularly known as pandas is used for analyzing data. It is an open source and developed by Wes McKinney.

Pandas take various inputs files like CSV or TSV or an SQL database and convert them to python object with rows and columns known as data frames similar to EXCEL files.

It can be used for converting csv dataset to python object (data frames) so then we can easily perform operations on each data.

It has an easy operation for converting sentences to words, removing stop words, deletion of a particular row or column, easy modification of data, etc.

Pandas is a BSD-licensed, open-source Python library that provides easy-to-use, high-performance data structures and data analysis tools for the Python programming language.

Python with Pandas is used in many fields including academic and business fields including finance, economics, statistics, analytics, etc.

### 3.5. PyQT5



**Figure 3.5.a PyQt5 Logo**

Qt is a collection of cross-platform C++ libraries that implement high-level APIs to access many aspects of modern desktop and mobile systems.

These include location and location services, multimedia, NFC and Bluetooth connectivity, Chromium-based web browsers, as well as traditional user interface development. PyQt5 is a complete set of Python bindings for Qt v5.

It is implemented as over 35 extension modules and allows Python to be used as an alternative application development language to C++ on all supported platforms including iOS and Android.

PyQt5 can also be integrated into C++-based applications to allow users of these applications to configure or improve the functionality of these applications.

This is a Python frontend for Qt, one of the most popular and powerful cross platform GUI libraries. PyQt5 is a combination of the Python programming language and the Qt library. This introductory tutorial will help you create graphical applications using PyQt5.

### 3.6. Dlib



**Figure 3.6.a Flask Logo**

Dlib is a modern C++ toolkit containing machine learning algorithms and tools for building complex C++ software to solve real-world problems. It is used in both industry and academia in many areas including robotics, embedded devices, mobile phones, and large high-performance computing environments. Dlib's open-source license allows you to use it in any application.

- **Machine Learning Algorithms that can be implemented using Dlib**Deep Learning
  - Conventional SMO based Support Vector Machines for classification and regression
  - Reduced-rank methods for large-scale classification and regression
  - Relevance vector machines for classification and regression
  - General purpose multiclass classification tools
  - A Multiclass SVM
  - A tool for solving the optimization problem associated with structural support vector machines.
  - Structural SVM tools for sequence labeling
  - Structural SVM tools for solving assignment problems
  - Structural SVM tools for object detection in images as well as more powerful (but slower) deep learning tools for object detection.
  - Structural SVM tools for labeling nodes in graphs
  - A large-scale SVM-Rank implementation
  - An online kernel RLS regression algorithm
  - An online SVM classification algorithm
  - Semidefinite Metric Learning
  - An online kernelized centroid estimator/novelty detector and offline support vector one-class classification
  - Clustering algorithms: linear or kernel k-means, Chinese Whispers, and Newman clustering.

## **Chapter 4**

### **4. Design Overview and Implementation**

#### **4.1 Design Overview**

The process of defining the architecture, components, modules, interfaces, and data for a system in order to meet given requirements is known as system design. It may also be defined as the process of creating a new business system or updating an old one by defining its components or modules to meet the user's specific requirements. It concentrates on how to achieve the system's goal. It covers the decomposition and arrangement of software into components, as well as the interactions between those components.

Architectural design (describes the system's structure, behavior, and viewpoints), logical design (abstract representation of the system's information flows, inputs, and outputs), and physical design are all part of system design (describes how data is input, processed and displayed during a system).

One of the most crucial parts of the software development process is system design. The issue statement, requirements determination plan, existing situation analysis, and proposed system requirements are all inputs to system design, which produces a knowledge schema, function hierarchy diagram, and a prototype for the proposed system. It's a crucial component of system development without which the suggested system would be impossible to create.

#### **4.2 Software algorithm**

This project is implemented by two algorithm Haar Cascade Algorithm and LBPH  
Let's discuss these algorithms in detail:

- **Haar Cascade:**

It's an Object Detection Algorithm that detects faces in images and real-time videos. Viola and Jones proposed edge or line detection features in their research paper "Rapid Object Detection using a Boosted Cascade of Simple Features," published in 2001. To train on, the

algorithm is given a large number of positive photos with faces and a large number of negative images with no faces. The model developed as a result of this training can be found on the OpenCV GitHub repository.

The models are saved in XML files in the repository and may be accessed using OpenCV techniques. Face detection, eye detection, upper and lower body detection, license plate detection, and other models are among them. Some of the notions offered by Viola and Jones in their research are shown here.

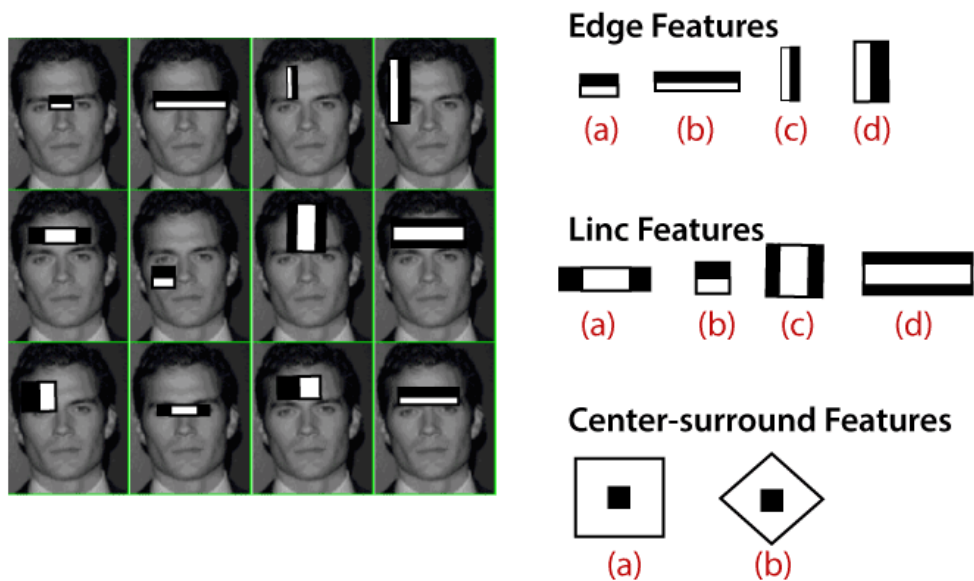


Figure 4.2.a Haar features

The introduction of the haar traits shown above was the initial contribution to the research. These elements on the image make it simple to locate the image's boundaries or lines, as well as locations where the brightness of the pixels abruptly shift.

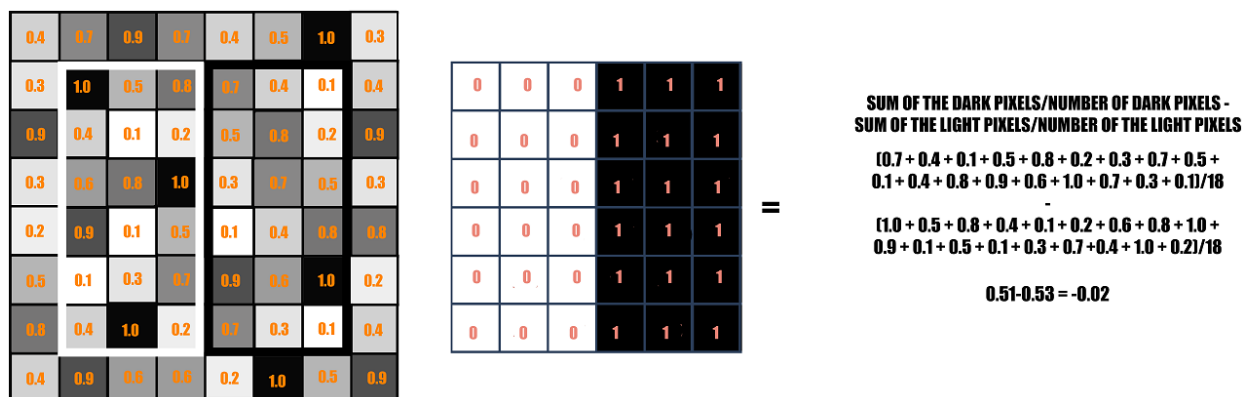


Figure 4.2.b Haar Pixels

The rectangle on the left is an example of an image with pixel values ranging from 0.0 to 1.0. A haar kernel is a rectangle in the middle with all the bright pixels on the left and all the dark

pixels on the right. The difference between the average of the pixel values in the darker zone and the average of the pixel values in the lighter region is used to calculate the haar. If the difference is close to one, the haar feature has detected an edge.

A sample calculation of Haar value from a rectangular image section has been shown here. The darker areas in the haar feature are pixels with values 1, and the lighter areas are pixels with values 0. Each of these is responsible for finding out one particular feature in the image. Such as an edge, a line or any structure in the image where there is a sudden change of intensities. For ex. in the image above, the haar feature can detect a vertical edge with darker pixels at its right and lighter pixels at its left.

The goal is to calculate the sum of all image pixels in the darker part of the haar feature, as well as the sum of all image pixels in the lighter portion of the haar feature. Then figure out how they differ. The haar value will be closer to 1 if the image has an edge dividing dark pixels on the right from light pixels on the left. That is, if the haar value is closer to 1, we claim there is an edge detected. There is no edge in the example above because the haar value is distant from 1.

- LBPH:

This is a face recognition algorithm that works on finding the details in the image.

The initial phase of the algorithm, given a face in a dataset, is to divide the face into 77 equal-sized cells.

Then we generate a Local Binary Pattern histogram for each of these cells.

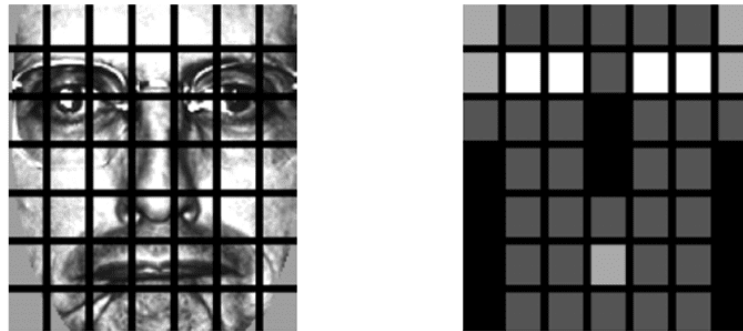


**Figure 4.2.c 7\*7 cells (Source: Google Images)**



A histogram, by definition, discards any spatial information about how the patterns are arranged next to each other. However, by computing a histogram for each of the cells, we are able to encode a level of spatial information that we would not have otherwise, such as the eyes, nose, mouth, and so on.

This spatial encoding also allows us to weight the histograms produced by each of the cells differently, giving more discriminative strength to more distinctive facial features:



**Figure 4.2.d The original face image followed by the weighting scheme for the 7\*7 cells.**

we can see the original face image divided into  $7 \times 7$  cells (*left*). Then, on the *right*, we can see the weighting scheme for each of the cells:

- LBP histograms for the white cells (such as the eyes) are weighed 4x more than the other cells. This simply means that we take the LBP histograms from the white cell regions and multiply them by 4 (taking into account any scaling/normalization of the histograms).
- Light grey cells (mouth and ears) contribute 2x more.
- Dark grey cells (inner cheek and forehead) only contribute 1x.
- Finally, the black cells, such as the nose and outer cheek, are totally disregarded and weighed 0x.

These weighting values were discovered by using hyperparameter tuning techniques to their training, validation, and testing data splits.

Finally, the final feature vector is created by concatenating the weighted 77% LBP histograms.

The  $\chi^2$  distance and a nearest neighbour classifier are used to do facial recognition: LBPs are extracted, weighted, and concatenated in the same way as training data when a face is shown to the system.

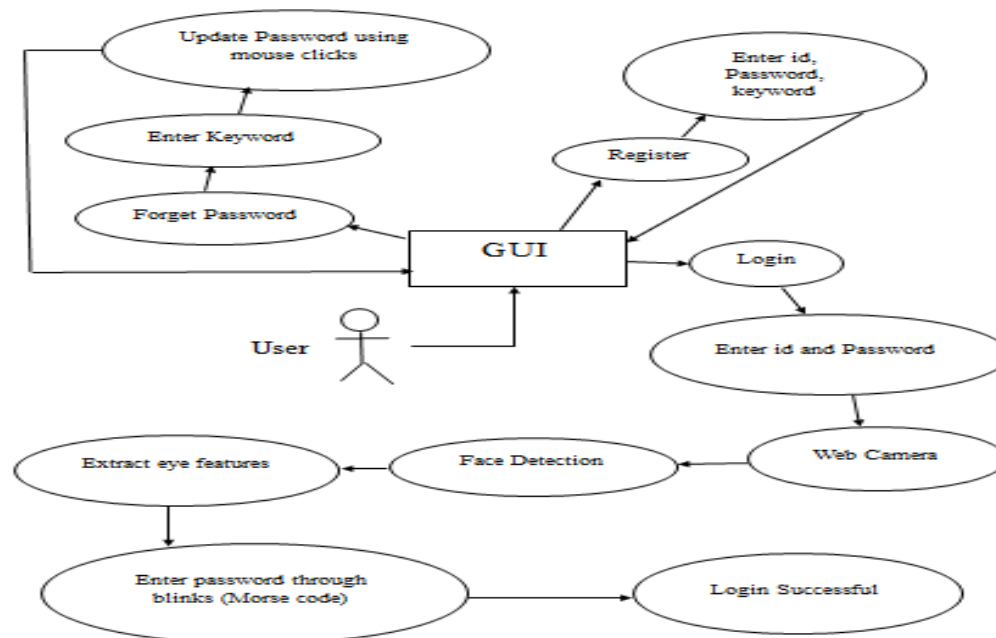
The  $\chi^2$  distance is used with k-NN (with  $k=1$ ) to discover the nearest face in the training data.

As the final categorization, the name of the person connected with the face with the shortest  $\chi^2$  distance is picked.

### 4.3 Use Case Diagram

A Use case diagram is a type of behavioral diagram specified by and derived from a Use case analysis in the Unified Modelling Language (UML). Its goal is to offer a graphical representation of a system's functionality in terms of actors, goals (expressed as use cases), and any dependencies between those use cases. The main goal of an usage diagram is to show which system functions an actor performs. The roles of the system's actors are frequently depicted.

The use case diagram for our project is depicted in the diagram below. The user will login or register as a user when interacting with the GUI (Graphical User Interface). When registration, the user must provide a user id, password, and keyword.

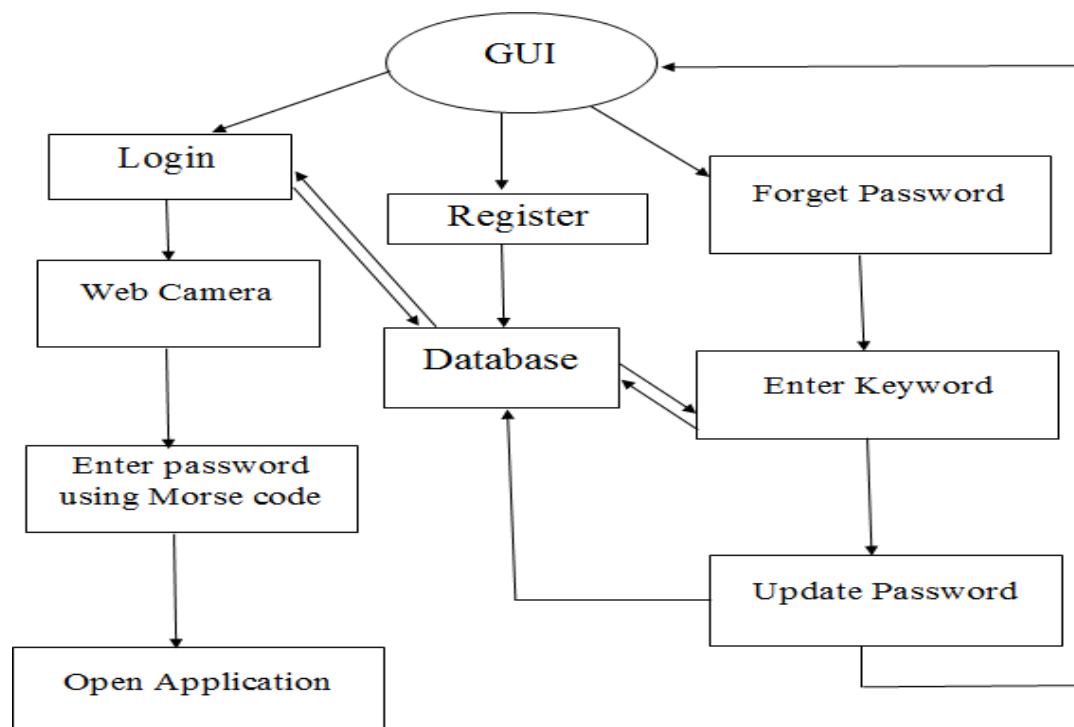


**Figure 4.3.a Use Case Diagram**

When a person has to log in to their account, they must use their user id and password. The internet camera is turned on once they've been identified as legitimate users. The webcam is used to detect the user's face and begin extracting the attention features in real time. During this time, the user must blink their eyelids to enter their password in Morse code. The user's login is successful if they are able to properly enter the password. If a user is unable to remember their password or want to change their password, they must answer the security question using the keyword that they provided when registering. The password is frequently updated when the term matches.

#### 4.4 Data-flow Diagram

A dataflow diagram (DFD) depicts the flow of data and information through a process or system. They use predefined symbols such as rectangles, circles, arrows, and text labels to represent the system's data inputs, outputs, storage sites, and routes. DFDs can range from simple hand-drawn overviews to detailed descriptive diagrams that delve deeper into data flow.



**Figure 4.4.a Data Flow Diagram**

## Chapter 5

### 5.1 RESULTS AND DISCUSSION

- This is the simple user interface for starting the process

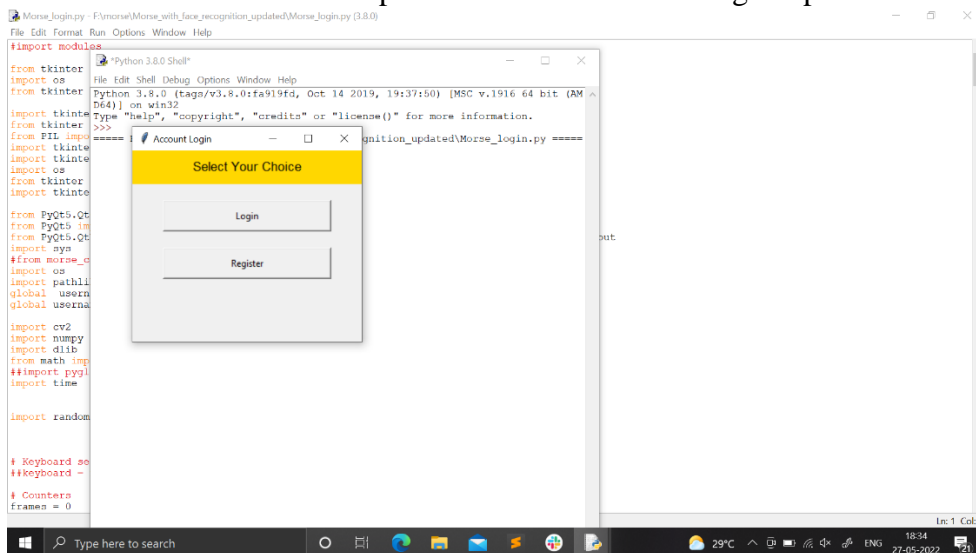


Figure 5.1.a Staring GUI

#### Registration:

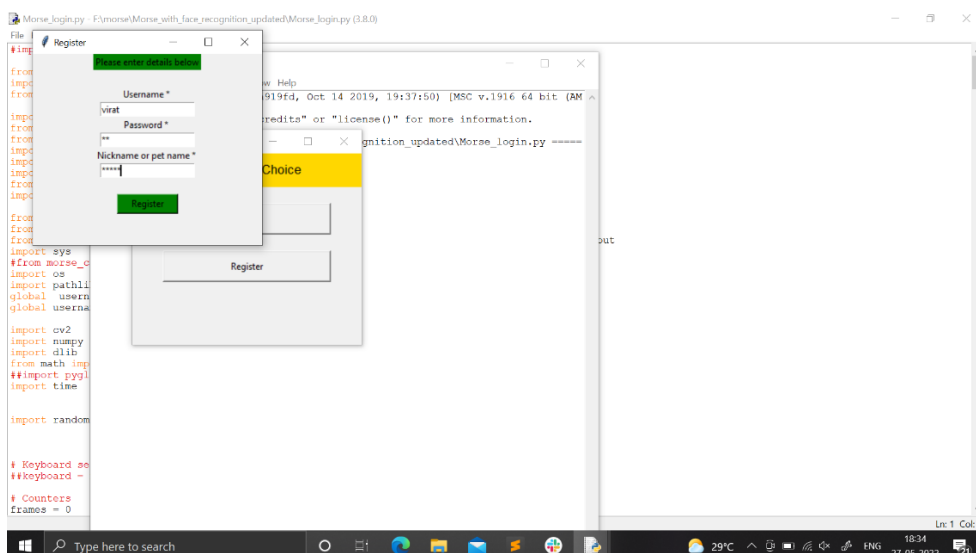
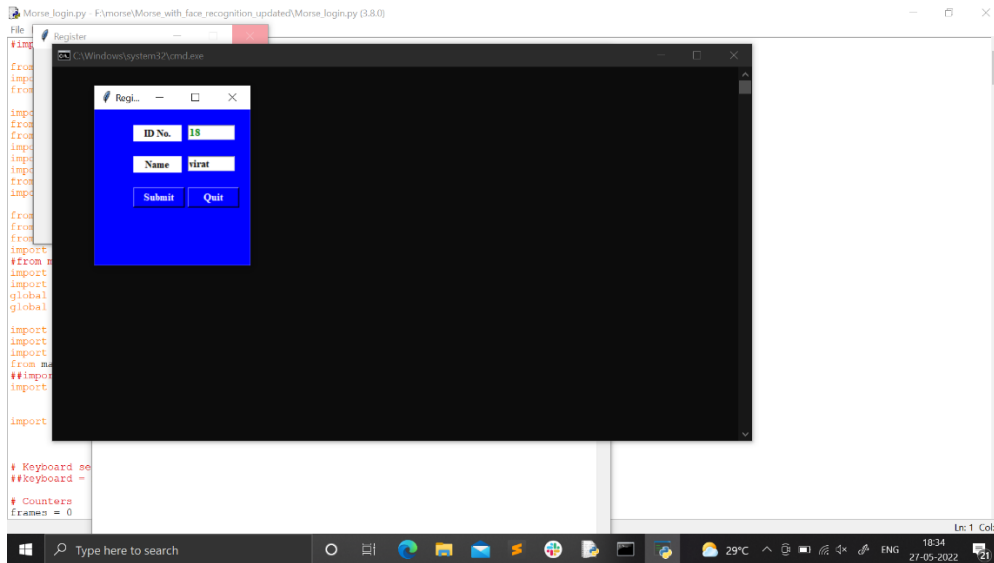


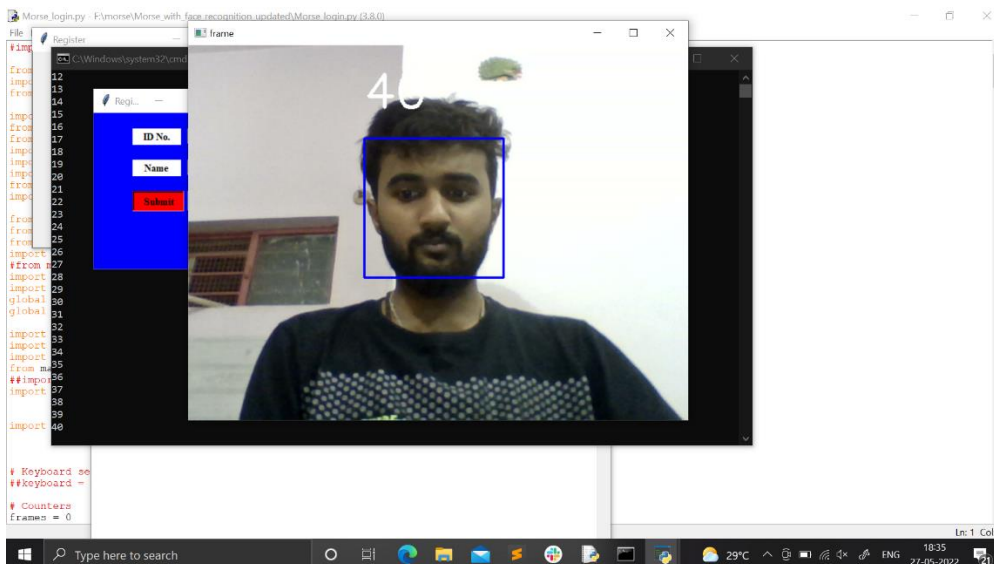
Figure 5.1.b Registration

- Details are entered while registration



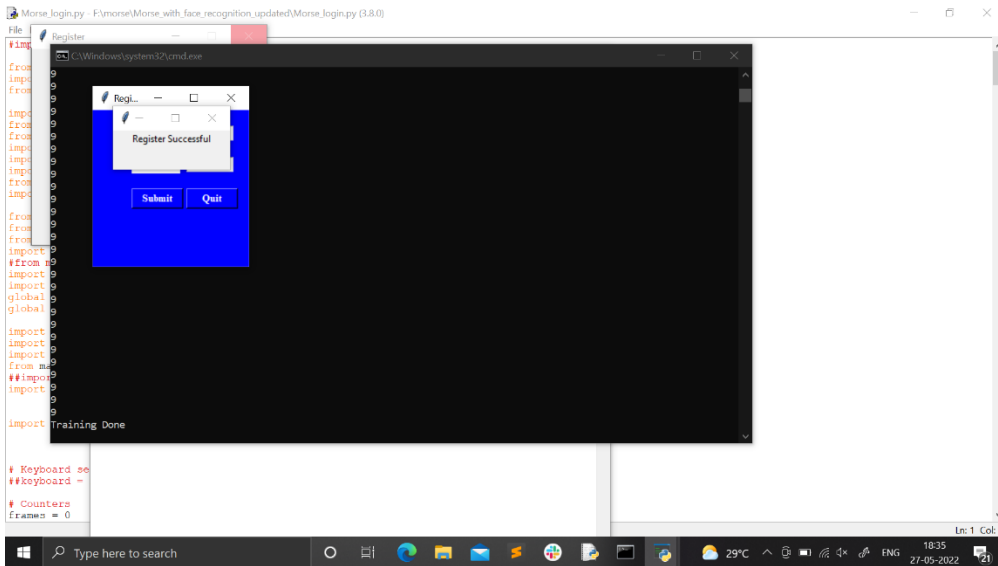
**Figure 5.1.c Filling Details**

- Live video is captured where not more than 120 frames are recorded, these frames are used to train the model for recognizing the face of person who is registering



**Figure 5.1.d Training**

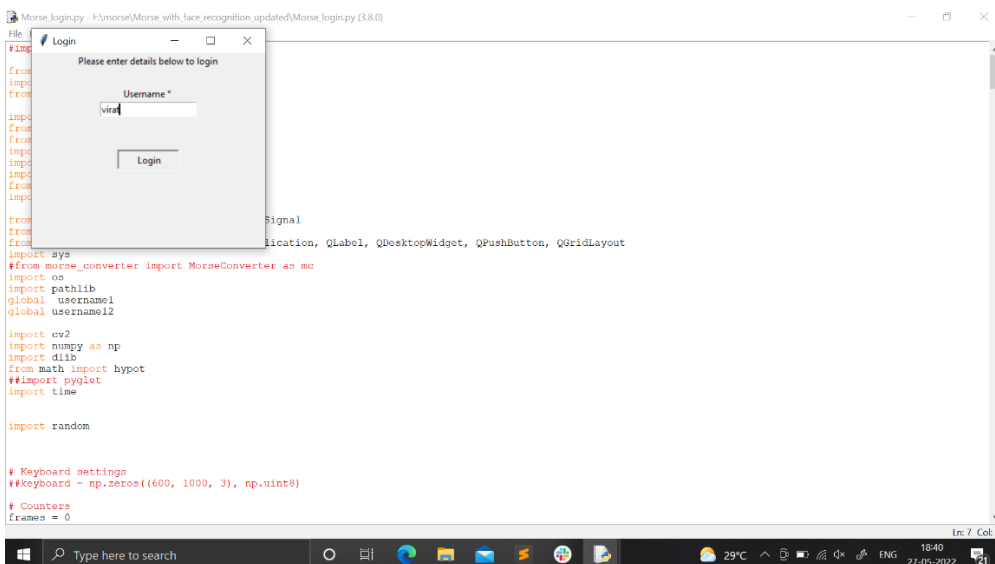
- We get a pop-up that shows registration is successful



### Figure 5.1.e Registration Successful

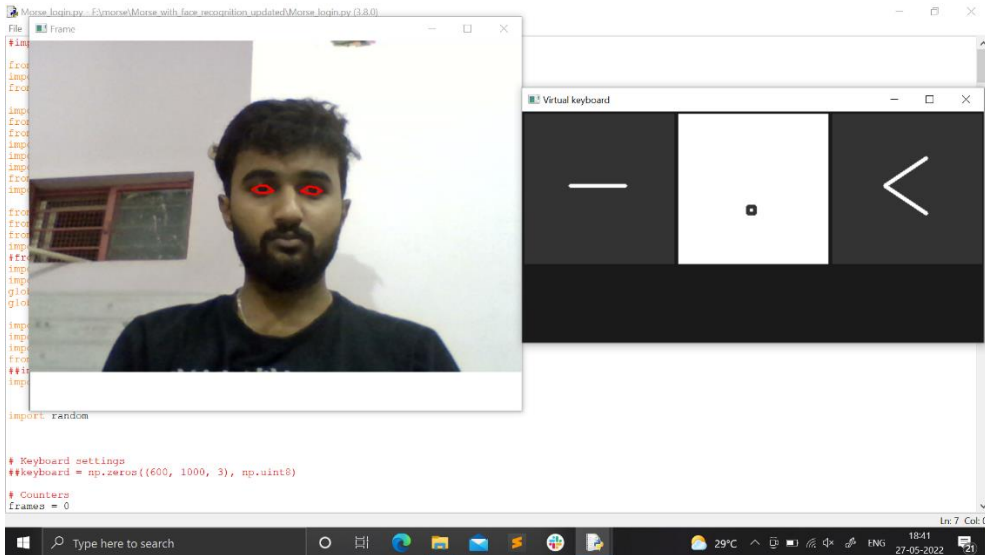
**Login:**

- Once registration is completed that person can login any time with the credentials (Username and password)



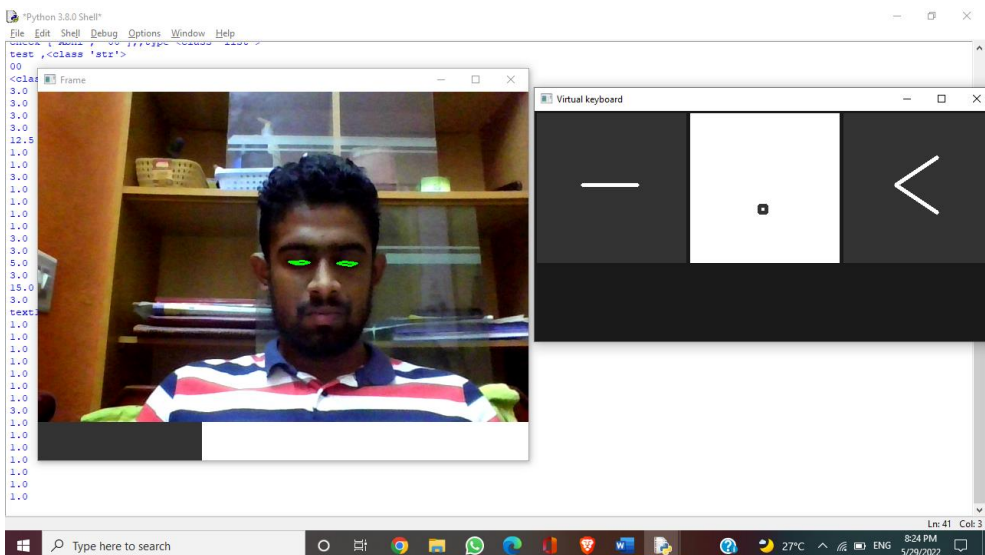
**Figure 5.1.f Login Window**

- Username will be verified and the next window will be opened that is a frame with live video and a virtual keyboard.



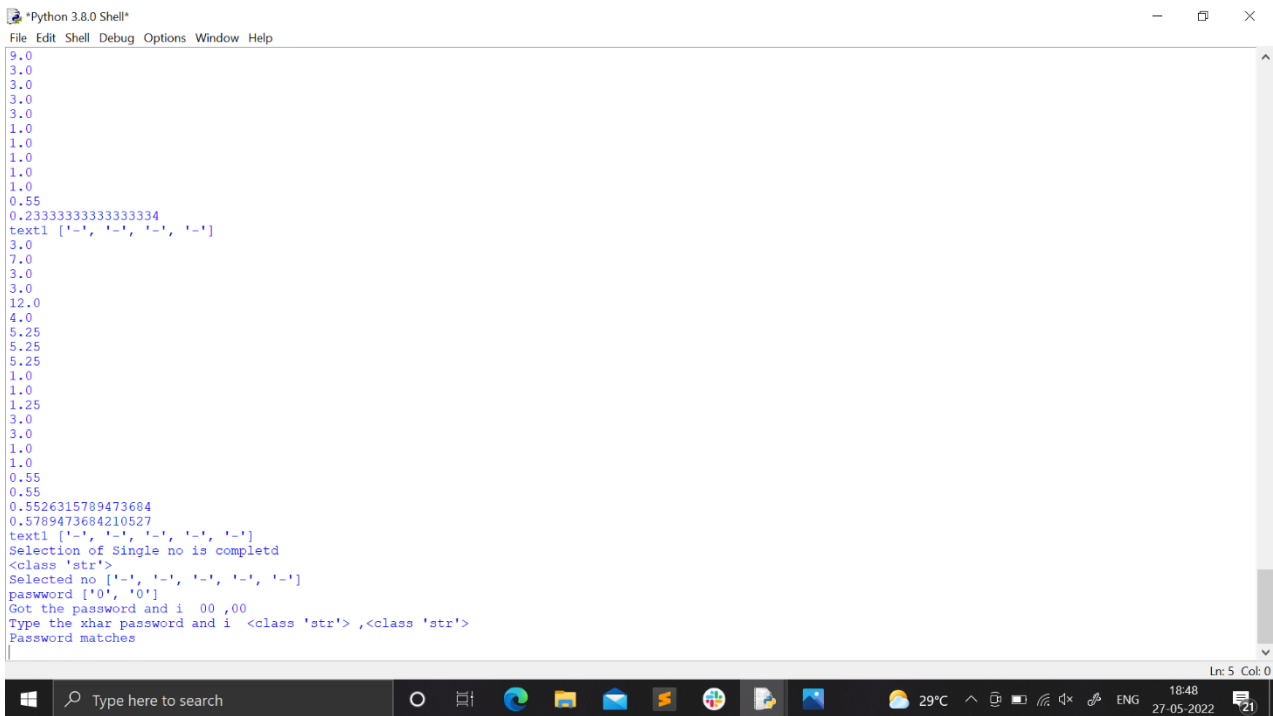
**Figure 5.1.g Webcam and Virtual Keyboard**

- When the eye blink(closed) is detected the corresponding symbol in the virtual keyboard is entered



**Figure 5.1.h Detection and Entering**

- When the sequence of symbols is entered as shown in previous step, the morse code is converted into numbers (0-9) and is verified with the password that was entered during registration



```
Python 3.8.0 Shell
File Edit Shell Debug Options Window Help
9.0
3.0
3.0
3.0
3.0
1.0
1.0
1.0
1.0
1.0
0.55
0.23333333333333334
text1 ['-', '-', '-', '-']
3.0
7.0
3.0
3.0
12.0
4.0
5.25
5.25
5.25
1.0
1.0
1.25
3.0
3.0
1.0
1.0
0.55
0.5526315789473684
0.5789473684210527
text1 ['-', '-', '-', '-']
Selection of Single no is completd
<class 'str'>
Selected no ['-', '-', '-', '-', '-']
password ['0', '0']
Got the password and i 00 ,00
Type the xhar password and i <class 'str'> ,<class 'str'>
Password matches
```

**Figure 5.1.i Final Output**

### **Scenario of wrong password:**

- When a wrong password is entered a new tab open which asks the user to enter the Nickname which was used while registering when the Nickname is verified the password can be changed.



## **5.2 CONCLUSIONS AND FUTURE SCOPE**

### **Conclusions**

- In essence, our project enables two-factor authentication. Two-factor authentication protects an account or system by adding a second layer of security.
- Here, we're using gaze-based authentication and clicking so that integers or ASCII are converted, so boosting security.
- This project is beneficial to disabled individuals in terms of authentication. This model can be used by anyone with a rudimentary understanding of morse code, from children to the elderly.
- There are keyboards with braille dots on each button for blind people. In terms of long-term enhancements, we are attempting to develop face recognition for each user, which will eliminate the need to enter a password within the shortest time possible.

### **Future Scope**

- One of the important drawbacks here is the time consumption this can be solved through adding more symbols which provides more sequences
- As of now it captures the frames when the room light is more than 60 % this can be solved by using night vision feature in the web cam.
- As the algorithm only detects the face and eyes it can be improved in such a way that it detects even when a person is wearing shades or sunglasses.
- Username can be entered through voice detection which requires accuracy at its most.

## REFERENCES

- [1] S. M, N. Kolkar, S. G. S and K. Kulkarni, "Morse Code Detector and Decoder using Eye Blinks," 2021 Third International Conference on Inventive Research in Computing Applications (ICIRCA), 2021, pp. 1-6, doi: 10.1109/ICIRCA51532.2021.9545039.
- [2] A. M. Arfi, D. Bal, M. A. Hasan, N. Islam and Y. Arafat, "Real Time Human Face Detection and Recognition Based on Haar Features," 2020 IEEE Region 10 Symposium (TENSYP), 2020, pp. 517-521, doi: 10.1109/TENSYP50017.2020.9230857.
- [3] A. Juhong, T. Treebupachatsakul and C. Pintavirooj, "Smart eye-tracking system," 2018 International Workshop on Advanced Image Technology (IWAIT), 2018, pp. 1-4, doi: 10.1109/IWAIT.2018.8369701.
- [4] P. Ran and H. Wang, "REAL-TIME EYE BLINK DETECTION BASED ON PYTHON," The 8th International Symposium on Test Automation & Instrumentation (ISTAI 2020), 2020, pp. 98-100, doi: 10.1049/icp.2021.1312.
- [5] K. Dang and S. Sharma, "Review and comparison of face detection algorithms," 2017 7th International Conference on Cloud Computing, Data Science & Engineering - Confluence, 2017, pp. 629-633, doi: 10.1109/CONFLUENCE.2017.7943228.
- [6] S. Sathasivam, A. K. Mahamad, S. Saon, A. Sidek, M. M. Som and H. A. Ameen, "Drowsiness Detection System using Eye Aspect Ratio Technique," 2020 IEEE Student Conference on Research and Development (SCOREd), 2020, pp. 448-452, doi: 10.1109/SCOREd50371.2020.9251035.
- [7] Wadehn F, Weber T, Mack DJ, Heldt T, Loeliger HA. Model-Based Separation, Detection, and Classification of Eye Movements. IEEE Trans Biomed Eng. 2020 Feb;67(2):588-600. doi: 10.1109/TBME.2019.2918986. Epub 2019 May 30. PMID: 31150326.
- [8] W. Pichitwong and K. Chamnongthai, "An EyeTracker-Based 3D Point-of-Gaze Estimation Method Using Head Movement," in IEEE Access, vol. 7, pp. 99086-99098, 2019, doi: 10.1109/ACCESS.2019.2929195.
- [9] Tingting Zhang , Ling Xia, Xiaofeng Liu, Xiaoli Wu : Eye movements during change detection: the role of depth of field doi : 10.1049/css.2019.0003
- [10] J. Chandiwalla and S. Agarwal, "Driver's real-time Drowsiness Detection using Adaptable Eye Aspect Ratio and Smart Alarm System," 2021 7th International Conference on Advanced Computing and Communication Systems (ICACCS), 2021, pp. 1350- 1355, doi: 10.1109/ICACCS51430.2021.9441756

## **APPENDIX**

### **Main.py**

```
#import modules

from tkinter import *
import os
from tkinter import ttk

import tkinter as tk
from tkinter import Message ,Text
from PIL import Image, ImageTk
import tkinter.ttk as ttk
import tkinter.font as font
import os
from tkinter import *
import tkinter as ttk

from PyQt5.QtCore import Qt, QTimer, pyqtSignal
from PyQt5 import QtGui
from PyQt5.QtWidgets import QWidget, QApplication, QLabel, QDesktopWidget, QPushButton, QGridLayout
import sys
#from morse_converter import MorseConverter as mc
import os
import pathlib
global username1
global username12

import cv2
import numpy as np
import dlib
from math import hypot
##import pygame
import time

import random

# Keyboard settings
##keyboard = np.zeros((600, 1000, 3), np.uint8)

# Counters
frames = 0
letter_index = 0
blinking_frames = 0
frames_to_blink = 6
frames_active_letter = 9

# Text and keyboard settings
text = ""
text1=[]
keyboard_selected = "left"
last_keyboard_selected = "left"
select_keyboard_menu = True
keyboard_selection_frames = 0
count=0
```

```
pf=[]

scanned=0

#####Mousse clcking #####

from PyQt5.QtCore import Qt, QTimer, pyqtSignal
from PyQt5 import QtGui
from PyQt5.QtWidgets import QWidget, QApplication, QLabel, QDesktopWidget, QPushButton, QGridLayout
import sys
from morse_converter import MorseConverter as mc

class MouseClicksMorse(QWidget):
    global username1
    global username12

    def __init__(self):
        super().__init__()
        self.inputArea = InputArea()
        self.initUI()

    def initUI(self):
        self.center()
        self.resize(700, 500)
        self.setWindowTitle('Mouse Clicks - Morse Code Conversion')

        self.inputArea.update_labels.connect(self.updateLabels)
        self.inputArea.clear_labels.connect(self.clearLabels)

        inst = QLabel()
        inst.setText('Instructions:\n Dot (.)\t\t: Left Click\n Dash (-)\t\t: Double Left Click\n Next Letter\t: Right Click\n Next Word\t: Double Right Click')
        font = QtGui.QFont("MoolBoran", 18)
        font.setStyleHint(QtGui.QFont.TypeWriter)
        inst.setFont(font)

        grid = QGridLayout()
        grid.setSpacing(5)
        grid.addWidget(inst, 1, 3)
        grid.addWidget(self.inputArea, 1, 0, 5, 1)
        grid.addWidget(self.inputArea.outputMorse, 3, 3)
        grid.addWidget(self.inputArea.outputConverted, 4, 3)
        grid.addWidget(self.inputArea.clearButton, 5, 3)

        self.setLayout(grid)

        self.setGeometry(300, 300, 550, 300)
        self.setWindowTitle('Mouse Clicks - Morse Code Conversion')

        self.show()

    def keyPressEvent(self, event):
        if event.key() == Qt.Key_Escape:
            self.close()

    def center(self):
```

```
qr = self.frameGeometry()
cp = QDesktopWidget().availableGeometry().center()
qr.moveCenter(cp)
self.move(qr.topLeft())

def updateLabels(self):
    self.inputArea.outputMorse.setText('Morse Code: <b>' + self.inputArea.message.replace('*', ' ').replace('.', ' '))
    if self.inputArea.message[-1] == '*':
        self.inputArea.outputConverted.setText('Conv. Text: <b>' + mc._morseToText(self.inputArea.message))
##outputConvclearLabelserted

def clearLabels(self):
    self.inputArea.outputMorse.setText('Morse Code: ')
    self.inputArea.outputConverted.setText('Conv. Text: ')
    test1=mc._morseToText(self.inputArea.message).strip('\n').strip('\t').strip(' ')
    print('dffdfddddd')
    print('username ,type { },{ }'.format(username1,type(username1)))
    username12=username1 + '.txt'
    print((username12))

    print(file2)
    file2.write(username1+ ',' + str(test1))

    file2.close()
    self.inputArea.message = ""
    exit()

class InputArea(QWidget):

    update_labels = pyqtSignal()
    clear_labels = pyqtSignal()

    def __init__(self):
        super().__init__()
        self.timer = QTimer()
        self.timer.setInterval(250)
        self.timer.setSingleShot(True)
        self.timer.timeout.connect(self.timeout)
        self.click_count = 0
        self.message = ""
        self.temp = ""

        self.setAutoFillBackground(True)
        p = self.palette()
        p.setColor(self.backgroundRole(), Qt.lightGray)
        self.setPalette(p)

        self.outputMorse = QLabel()
        self.outputMorse.setText('Morse Code: ')
        self.outputConverted = QLabel()
        self.outputConverted.setText('Conv. Text: ')
        font = QtGui.QFont("Consolas", 10)
        font.setStyleHint(QtGui.QFont.TypeWriter)
        self.outputMorse.setFont(font)
        self.outputConverted.setFont(font)

        self.clearButton = QPushButton('Clear All')
        self.clearButton.clicked.connect(self.sendClearSignal)
```

```
def mousePressEvent(self, event):
    self.click_count += 1
    if event.button() == Qt.LeftButton:
        self.temp = '.'
    if event.button() == Qt.RightButton:
        self.temp = '*'
    if not self.timer.isActive():
        self.timer.start()

def timeout(self):
    if self.click_count > 1:
        if self.temp == '*':
            self.message += '**'
        else:
            self.message += '-'
    else:
        self.message += self.temp
    self.click_count = 0
    self.update_labels.emit()

def sendClearSignal(self):
    self.clear_labels.emit()

def getMessage(self):
    return self.message

def printMessage(self):
    print(self.message)

#####mouse clicking#####
# Designing window for registration

def register():
    global register_screen
    register_screen = Toplevel(main_screen)
    register_screen.title("Register")
    register_screen.geometry("300x250")
    global username
    global password
    global username_entry
    global password_entry
    global question
    username = StringVar()
    password = StringVar()

    question = StringVar()

    Label(register_screen, text="Please enter details below", bg="Green").pack()
    Label(register_screen, text="").pack()
    username_label = Label(register_screen, text="Username * ")
    username_label.pack()
    username_entry = Entry(register_screen, textvariable=username)
    username_entry.pack()
    password_label = Label(register_screen, text="Password * ")
    password_label.pack()
    password_entry = Entry(register_screen, textvariable=password, show='*')
    password_entry.pack()
    question = Label(register_screen, text="Nickname or pet name * ")
```

```
question.pack()
question = Entry(register_screen, textvariable=question, show= '*')
question.pack()

Label(register_screen, text="").pack()
Button(register_screen, text="Register", width=10, height=1, bg="Green", command = register_user).pack()
```

# Designing window for login

def login():

```
global login_screen
login_screen = Toplevel(main_screen)
login_screen.title("Login")
login_screen.geometry("300x250")
Label(login_screen, text="Please enter details below to login").pack()
Label(login_screen, text="").pack()
```

```
global username_verify
global password_verify
```

```
username_verify = StringVar()
password_verify = StringVar()
```

```
## q_verify = StringVar()
```

```
global username_login_entry
global password_login_entry
os.system('python login.py')#*****
```

```
Label(login_screen, text="Username * ").pack()
username_login_entry = Entry(login_screen, textvariable=username_verify)
username_login_entry.pack()
Label(login_screen, text="").pack()
```

```
## Label(login_screen, text="Password * ").pack()
## password_login_entry = Entry(login_screen, textvariable=password_verify, show= '*')
## password_login_entry.pack()
```

```
## Label(login_screen, text="Question * ").pack()
## password_login_entry = Entry(login_screen, textvariable=q_verify, show= '*')
## password_login_entry.pack()
##
```

```
Label(login_screen, text="").pack()
Button(login_screen, text="Login", width=10, height=1, command = login_verify).pack()
```

# Implementing event on register button

def register\_user():

```
username_info = username.get()
password_info = password.get()
question_info = question.get()
username_info1=username_info+'.txt'
file = open(username_info1, "w")
#file.write(username_info + "\n")
file.write(username_info + ",")
file.write(password_info)
file.close()
```

```
username_info_p=username_info+'_p' + '.txt'
file = open(username_info_p, "w")
file.write(question_info)
file.close()

username_entry.delete(0, END)
#password_entry.delete(0, END)
os.system('python
Register.py')#####
#####
print("Registration Success")

Label(register_screen, text="Registration Success", fg="green", font=("calibri", 11)).pack()

# Implementing event on login button
def listToString(s):

    # initialize an empty string
    str1 = ""

    # return string
    return (str1.join(s))

def listToString1(s):

    # initialize an empty string
    str2 = ""

    # return string
    return (" ".join(str(e) for e in list))

def Convert(string):
    li = list(string.split(""))
    return li
def login_verify():

    global username1
    username1 = username_verify.get()
    username12=username1+'.txt'
    password1 = password_verify.get()
    username_login_entry.delete(0, END)
    #password_login_entry.delete(0, END)
    global inputpassword
    global username_info1_p
    list_of_files = os.listdir()

    if username12 in list_of_files:
        file1 = open(username12, "r")
        #verify = file1.read().splitlines()
        verify = file1.read().split(',')
        check=list(verify)
        print('check { },,type { }'.format(check,type(check)))
        if True:#password1 in verify:
            print('test { },{ }'.format(password1,type(verify[1])))
            inputpassword = listToString(verify[1])
            print(inputpassword)
            print(type(inputpassword))
```



```
## login_sucess()
cap = cv2.VideoCapture(0)
board = np.zeros((300, 700), np.uint8)
board[:] = 255

detector = dlib.get_frontal_face_detector()
predictor = dlib.shape_predictor("shape_predictor_68_face_landmarks.dat")
# gaze()
global frames
global letter_index
global blinking_frames
global frames_to_blink
global frames_active_letter

global text
global text1
global keyboard_selected
global last_keyboard_selected
global select_keyboard_menu
global keyboard_selection_frames
global count
pf=[]
#####
keyboard = np.zeros((300, 600, 3), np.uint8)

keys_set_1 = {0: "-", 1: ".",
2: "<", 3: 'D'}

def draw_letters(letter_index, text, letter_light):
# Keys
    if letter_index == 0:
        x = 0
        y = 0
    elif letter_index == 1:
        x = 200
        y = 0
    elif letter_index == 2:
        x = 400
        y = 0
    elif letter_index == 3:
        x = 600
        y = 0

    width = 200
    height = 200
    th = 3 # thickness
##login_sucess
    # Text settings
    font_letter = cv2.FONT_HERSHEY_PLAIN
    font_scale = 9
    font_th = 4
    text_size = cv2.getTextSize(text, font_letter, font_scale, font_th)[0]
    width_text, height_text = text_size[0], text_size[1]
    text_x = int((width - width_text) / 2) + x
    text_y = int((height + height_text) / 2) + y

    if letter_light is True:
        cv2.rectangle(keyboard, (x + th, y + th), (x + width - th, y + height - th), (255, 255, 255), -1)
        cv2.putText(keyboard, text, (text_x, text_y), font_letter, font_scale, (51, 51, 51), font_th)
```

```
else:
    cv2.rectangle(keyboard, (x + th, y + th), (x + width - th, y + height - th), (51, 51, 51), -1)
    cv2.putText(keyboard, text, (text_x, text_y), font_letter, font_scale, (255, 255, 255), font_th)

def draw_menu():
    rows, cols, _ = keyboard.shape
    th_lines = 4 # thickness lines

def midpoint(p1 ,p2):
    return int((p1.x + p2.x)/2), int((p1.y + p2.y)/2)

font = cv2.FONT_HERSHEY_PLAIN

def get_blinking_ratio(eye_points, facial_landmarks):
    left_point = (facial_landmarks.part(eye_points[0]).x, facial_landmarks.part(eye_points[0]).y)
    right_point = (facial_landmarks.part(eye_points[3]).x, facial_landmarks.part(eye_points[3]).y)
    center_top = midpoint(facial_landmarks.part(eye_points[1]), facial_landmarks.part(eye_points[2]))
    center_bottom = midpoint(facial_landmarks.part(eye_points[5]), facial_landmarks.part(eye_points[4]))

    hor_line_lenght = hypot((left_point[0] - right_point[0]), (left_point[1] - right_point[1]))
    ver_line_lenght = hypot((center_top[0] - center_bottom[0]), (center_top[1] - center_bottom[1]))

    ratio = hor_line_lenght / ver_line_lenght
    return ratio

def eyes_contour_points(facial_landmarks):
    left_eye = []
    right_eye = []
    for n in range(36, 42):
        x = facial_landmarks.part(n).x
        y = facial_landmarks.part(n).y
        left_eye.append([x, y])
    for n in range(42, 48):
        x = facial_landmarks.part(n).x
        y = facial_landmarks.part(n).y
        right_eye.append([x, y])
    left_eye = np.array(left_eye, np.int32)
    right_eye = np.array(right_eye, np.int32)
    return left_eye, right_eye

def get_gaze_ratio(eye_points, facial_landmarks):
    left_eye_region = np.array([(facial_landmarks.part(eye_points[0]).x,
    facial_landmarks.part(eye_points[0]).y),
                                (facial_landmarks.part(eye_points[1]).x, facial_landmarks.part(eye_points[1]).y),
                                (facial_landmarks.part(eye_points[2]).x, facial_landmarks.part(eye_points[2]).y),
                                (facial_landmarks.part(eye_points[3]).x, facial_landmarks.part(eye_points[3]).y),
                                (facial_landmarks.part(eye_points[4]).x, facial_landmarks.part(eye_points[4]).y),
                                (facial_landmarks.part(eye_points[5]).x, facial_landmarks.part(eye_points[5]).y)],
                                np.int32)

    # cv2.polylines(frame, [left_eye_region], True, (0, 0, 255), 2)

    height, width, _ = frame.shape
    mask = np.zeros((height, width), np.uint8)
    cv2.polylines(mask, [left_eye_region], True, 255, 2)
    cv2.fillPoly(mask, [left_eye_region], 255)
    eye = cv2.bitwise_and(gray, gray, mask=mask)

    min_x = np.min(left_eye_region[:, 0])
```

```
max_x = np.max(left_eye_region[:, 0])
min_y = np.min(left_eye_region[:, 1])

max_y = np.max(left_eye_region[:, 1])

gray_eye = eye[min_y: max_y, min_x: max_x]
_, threshold_eye = cv2.threshold(gray_eye, 70, 255, cv2.THRESH_BINARY)
height, width = threshold_eye.shape
left_side_threshold = threshold_eye[0: height, 0: int(width / 2)]
left_side_white = cv2.countNonZero(left_side_threshold)

right_side_threshold = threshold_eye[0: height, int(width / 2): width]
right_side_white = cv2.countNonZero(right_side_threshold)

if left_side_white == 0:
    gaze_ratio = 1
elif right_side_white == 0:
    gaze_ratio = 5
else:
    gaze_ratio = left_side_white / right_side_white
return gaze_ratio
```

```
#####
scanned=0
once=1
scanned =1
# pf =['1','5']
one=['.', '-', '-', '-', '-']
two=['.', '.', '-', '-', '-']
thrid=['.', '.', '.', '-', '-']
four=['.', '.', '.', '.', '-']

five=['.', '.', '.', '.', '.']
six=['-', '.', '.', '.', '.']
seven=['-', '-', '.', '.', '.']
eight=['-', '-', '-', '.', '.']

nine=['-', '-', '-', '-', '.']
zero=['-', '-', '-', '-', '-']

paswword =[0,0,0]
char=[]
while True:

    ret, frame = cap.read()
    rows, cols, _ = frame.shape
    keyboard[:] = (26, 26, 26)
    frames += 1
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    # Draw a white space for loading bar
    frame[rows - 50: rows, 0: cols] = (255, 255, 255)

    if select_keyboard_menu is True:
        draw_menu()

    # Keyboard selected
    if keyboard_selected == "left":
        keys_set = keys_set_1
```

```
active_letter = keys_set[letter_index]

# Face detection
faces = detector(gray)
for face in faces:
    landmarks = predictor(gray, face)

    left_eye, right_eye = eyes_contour_points(landmarks)

# Detect blinking
left_eye_ratio = get_blinking_ratio([36, 37, 38, 39, 40, 41], landmarks)
right_eye_ratio = get_blinking_ratio([42, 43, 44, 45, 46, 47], landmarks)
blinking_ratio = (left_eye_ratio + right_eye_ratio) / 2

# Eyes color
cv2.polylines(frame, [left_eye], True, (0, 0, 255), 2)
cv2.polylines(frame, [right_eye], True, (0, 0, 255), 2)

if select_keyboard_menu is True:
    # Detecting gaze to select Left or Right keybaord
    gaze_ratio_left_eye = get_gaze_ratio([36, 37, 38, 39, 40, 41], landmarks)
    gaze_ratio_right_eye = get_gaze_ratio([42, 43, 44, 45, 46, 47], landmarks)
    gaze_ratio = (gaze_ratio_right_eye + gaze_ratio_left_eye) / 2
    print(gaze_ratio)

    if gaze_ratio <= 5:
        keyboard_selected = "right"
        keyboard_selection_frames += 1
        # If Kept gaze on one side more than 15 frames, move to keyboard
        if keyboard_selection_frames == 15:
            select_keyboard_menu = False
            ##            right_sound.play()
            # Set frames count to 0 when keyboard selected
            frames = 0
            keyboard_selection_frames = 0
        if keyboard_selected != last_keyboard_selected:
            last_keyboard_selected = keyboard_selected
            keyboard_selection_frames = 0

    else:
        # Detect the blinking to select the key that is lighting up
        if blinking_ratio > 5:
            # cv2.putText(frame, "BLINKING", (50, 150), font, 4, (255, 0, 0), thickness=3)
            blinking_frames += 1
            frames -= 1

        # Show green eyes when closed
        cv2.polylines(frame, [left_eye], True, (0, 255, 0), 2)
        cv2.polylines(frame, [right_eye], True, (0, 255, 0), 2)

        # Typing letter
        if blinking_frames == frames_to_blink:
            if active_letter != "E" and active_letter != "C":
                count=count+1
                text += active_letter
                text1.append(active_letter)
                print('text1 {}'.format(text1))
            if active_letter == "<" or active_letter == "D" :
```

```
text += " "  
count=count-1  
text1.pop(count)  
count=count-1  
text1.pop(count)  
print('del {}'.format(text1))  
text += " "  
#print("text {}".format(text))  
  
if len(text1) == 5 :  
    print('Selection of Single no is complete')  
    print(type(str(text1)))  
  
if text1 == one:  
    print('Selected no {}'.format(one))  
    text1=[]  
    count=0  
    once =1  
    char.append('1')  
  
if text1 == two:  
    print('Selected no {}'.format(two))  
    text1=[]  
    count=0  
    once =1  
    char.append('2')  
  
if text1 == thrid:  
    print('Selected no {}'.format(thrid))  
    text1=[]  
    count=0  
    once =1  
    char.append('3')  
  
if text1 == four:  
    print('Selected no {}'.format(four))  
    text1=[]  
    count=0  
    once =1  
    char.append('4')  
if text1 == five:  
    print('Selected no {}'.format(five))  
    text1=[]  
    count=0  
    once =1  
    char.append('5')  
if text1 == six:  
    print('Selected no {}'.format(six))  
    text1=[]  
    count=0  
    once =1  
    char.append('6')  
  
if text1 == seven:  
    print('Selected no {}'.format(seven))  
    text1=[]  
    count=0
```

```
        once =1
        char.append('7')

    if text1 == eight:
        print('Selected no {}'.format(eight))
        text1=[]
        count=0
        once =1
        char.append('8')

    if text1 == nine:
        print('Selected no {}'.format(nine))
        text1=[]
        count=0
        once =1
        char.append('9')
    if text1 == zero:
        print('Selected no {}'.format(zero))
        text1=[]
        count=0
        once =1
        char.append('0')

    else:
        #print('not ,matched ')
        text1=[]
        count=0
        #print('Not MAtched char {}'.format(str(char)))
        print('paswword {}'.format(str(char)))
        select_keyboard_menu = True
        # time.sleep(1)

    else:
        blinking_frames = 0

# Display letters on the keyboard
if select_keyboard_menu is False:
    if frames == frames_active_letter:
        letter_index += 1
        frames = 0
    if letter_index == 3:
        letter_index = 0
    for i in range(3):
        if i == letter_index:
            light = True
        else:
            light = False
        draw_letters(i, keys_set[i], light)

# Show the text we're writing on the board
cv2.putText(board, str(text1), (80, 100), font, 9, 0, 3)

# Blinking loading bar
percentage_blinking = blinking_frames / frames_to_blink
loading_x = int(cols * percentage_blinking)
cv2.rectangle(frame, (0, rows - 50), (loading_x, rows), (51, 51, 51), -1)
pwd="
"
```

```
if len(char) > 1 :
    pwd=listToString(char)

print ('Got the password and i {} ,{}'.format(pwd,inputpassword))
print ('Type the xhar password and i {} ,{}'.format(type(pwd),type(inputpassword)))
lis=[]

start=0
##    break

    if str(pwd) == str(inputpassword) : #password
        print('Password matches ')
        lis=[]
        break
    else:
        print('Failed')
        char=[]
cv2.imshow("Frame", frame)
cv2.imshow("Virtual keyboard", keyboard)

    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
cap.release()
cv2.destroyAllWindows()

else:
    #password_not_recognised()
    file1.close()

    print('password Not Recognised')
    print('Enter the Security Answer')
    print('Pet name or nick name')
    check = input()
    check=str(check)
    print(check)
    username_info1_p=username1+'_p' + '.txt'
    file = open(username_info1_p, "r")
    print(file)

    if check in file:
        os.remove(username12)
        print('Fount the security question in database{ }'.format(check))
        app = QApplication(sys.argv)
        ex = MouseClicksMorse()
        # print(ex.message)
        sys.exit(app.exec_())

    else:
        print('Not fount in databse ')

else:
    user_not_found()
```

# Designing popup for login success

```
def login_sucess():  
    global login_success_screen  
    login_success_screen = Toplevel(login_screen)  
    login_success_screen.title("Success")  
    login_success_screen.geometry("150x100")  
    Label(login_success_screen, text="Login Success").pack()  
    Button(login_success_screen, text="OK", command=delete_login_success).pack()
```

```
#Button(login_success_screen, text="Start", command=programs).pack()
```

# Designing popup for login invalid password

```
def password_not_recognised():  
    global password_not_recog_screen  
    password_not_recog_screen = Toplevel(login_screen)  
    password_not_recog_screen.title("Success")  
    password_not_recog_screen.geometry("150x100")  
    Label(password_not_recog_screen, text="Invalid Password ").pack()  
    Button(password_not_recog_screen, text="OK", command=delete_password_not_recognised).pack()
```

# Designing popup for user not found

```
def user_not_found():  
    global user_not_found_screen  
    user_not_found_screen = Toplevel(login_screen)  
    user_not_found_screen.title("Success")  
    user_not_found_screen.geometry("150x100")  
    Label(user_not_found_screen, text="User Not Found").pack()  
    Button(user_not_found_screen, text="OK", command=delete_user_not_found_screen).pack()
```

# Deleting popups

```
def delete_login_success():  
    login_success_screen.destroy()
```

```
def delete_password_not_recognised():  
    password_not_recog_screen.destroy()
```

```
def delete_user_not_found_screen():  
    user_not_found_screen.destroy()
```



```
# Designing Main(first) window




def main_account_screen():
    global main_screen
    main_screen = Tk()
    main_screen.geometry("300x250")
    main_screen.title("Account Login")

    Label(text="Select Your Choice", bg="gold", width="300", height="2", font=("Arial", 13)).pack()
    Label(text="").pack()
    Button(text="Login", height="2", width="30", command = login).pack()
    Label(text="").pack()
    Button(text="Register", height="2", width="30", command=register).pack()


    main_screen.mainloop()

main_account_screen()
```

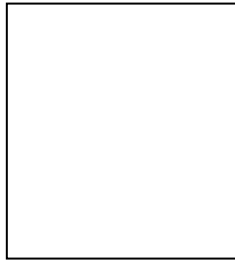
## INFORMATION REGARDING STUDENT

Student name	Usn	Email id	Phone Number	Photograph
Abhishek N	18BTRIS002	abhisinchu6@gmail.com	+91 87222 07551	
Pranav M	18BTRIS029	mpranav38@gmail.com	+91 9591555045	
Sathvik K C	18BTRIS042	sathvik366@gmail.com	+91 98446 84469	

*Secured Authentication Process for Physically Challenged  
People Using Morse Code and Machine Learning Algorithm*

Mansi D	18BTRIS023	mansi.dinesh05@gmail.co m	+91 7337831900	
---------	------------	------------------------------	----------------	---

## BATCH PHOTOGRAPH ALONG WITH GUIDE



Mrs. Sowmya KN, In  
charge ,Assistant  
Professor



ABHISHEK N  
USN:18BTRIS002



PRANAV M  
USN: 18BTRIS029



SATHVIK K C  
USN: 18BTRIS042



MANSI D  
USN: 18BTRIS023