STUDENT ATTENDANCE

Human face recognition using Deep Learning and TensorFlow

A Project Report

Submitted in partial fulfilment of the

Requirements for the award of the Degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

Submitted by

S. SABA AFREEN	-	1910157
J. JABEENA	-	1910174
M.NANDINI	-	1910137

Under the esteemed guidance of

Mr. G. VIJAY KUMAR, M. Tech, (Ph.D.)

Lecturer Department of CSE, SKUCET

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SRI KRISHNADEVARAYA UNIVERSITY

COLLEGE OF ENGINEERING AND TECHNOLOGY

ANANTAPUR - 515001

ANDHRA PRADESH

2019-2023

SRI KRISHNADEVARAYA UNIVERSITY
COLLEGE OF ENGINEERING AND TECHNOLOGY
ANANTAPUR – 515003
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

Certified that this is a bonafide record of the dissertation work entitled, "STUDENT ATTENDANCE HUMAN FACE RECOGNITION USING TENSORFLOW AND DEEP LEARNING", done by S. SABA AFREEN bearing Admn. No: 1910157, J. JABEENA bearing Admin. No: 1910174, M. NANDINI bearing Admin. No: 1910137 submitted to the faculty of Computer Science and Engineering, in partial fulfilment of the requirements for the Degree of BACHELOR OF TECHNOLOGY with specialisation in COMPUTER SCIENCE AND ENGINEERING from Sri Krishnadevaraya University College of Engineering and Technology, Anantapur.

Signature of the Supervisor
Sri, G. Vijay Kumar, M. Tech, Ph.D.
Lecturer dept. of CSE, SKUCET

Signature of the Head of the Department
Sri. P. R. Rajesh Kumar, M.Tech, Ph.D.
Lecturer dept. of CSE, SKUCET

DECLARATION

We hereby declare that the project report entitled “STUDENT ATTENDANCE” submitted to the Department of Computer Science and Engineering, Sri Krishnadevaraya University, Anantapuram for the partial fulfilment of the academic requirement for the degree for Bachelor of Technology in Computer Science and Engineering is an authentic record of our work carried out during the final year under the esteemed guidance of Guide Sri. G. Vijay Kumar, Lecturer Computer Science and Engineering Department, College of Engineering and Technology, Sri Krishnadevaraya University, Anantapuram.

Signature of the students

1.

2.

3.

ACKNOWLEDGEMENT

The satisfaction and euphoria that accompany the successful completion of any task would be incomplete without the mention of people who made it possible whose constant guidance and encouragement crowned our e orts with success. It is a pleasant aspect that I have now the opportunity to express my gratitude for all of them.

It is with immense pleasure that I would like to express my indebted gratitude **to** my **Guide Sri. G. VIJAY KUMAR, M. Tech** in CSE Department, who has guided me a lot and encouraged me in every step of the project work. His valuable moral support and guidance throughout the project helped me to a greater extent. I thank him for his stimulating guidance, constant encouragement and constructive criticism which have made this project work.

I wish to express my deep sense of gratitude to **P.R. RAJESH KUMAR, M. Tech, (Ph. D)** Lecturer and **Head of the Department (I/C) of Computer Science and Engineering,** for giving me the opportunity of doing the project and for providing a great support in completing my project work. I feel elated to thank him for inspiring me all the way by providing good lab facilities and helping me in providing such a good environment.

I wish to convey my appreciation to **Dr. R. RAMACHANDRA, M.Tech., Ph.D.,** Principal, SKU College of Engineering and Technology, Anantapuram, for providing such a good environment and facilities.

My special thanks to the **Faculty of CSE Department** for giving the required information in doing my project work. Not to forget, I thank all the non-teaching staff, my friends and classmates who had directly or indirectly helped and supported me in completing my project in time.

Finally, I wish to convey my gratitude to my parents who fostered all the requirements and facilities that I need.

S. SABA AFREEN - 1910157
J. JABEENA- 1910174
M. NADINI- 1910137

TABLE OF CONTENTS

ABSTRACT

Our project is to take students' attendance via human face detection using deep learning and TensorFlow. The purpose of our project is security and verification. The problem we as a team identified is lack of security for the college as there are so many students in the college. There may be a chance of entering strangers, unknown people. As well as to keep the data of the students for future reference. As it might also help the police in the investigation purpose, if any kind of criminal activities happen in the college.

Face detection is a problem in computer vision of locating and localising one or more faces in a photograph. Face Detection is the first and essential step for face recognition, and it is used to detect faces in the images. It is a part of object detection and can be used in many areas such as security, bio-metrics, law enforcement, entertainment, personal safety, etc. Face recognition is a broad problem of identifying or verifying people in photographs and videos, a process comprised of detection, alignment, feature extraction, and a recognition task It has 4 steps which are:

1)Face detection,

2) Data gathering,

3)Data comparison and

4)Face recognition.

Face detection can be done using frameworks like OpenCV and TensorFlow. TensorFlow is a really popular framework in the sphere of face recognition. The tools for face recognition in TensorFlow are Face Net by Google and MTCNN.

CHAPTER - 1

INTRODUCTION

1.1 **Introduction**

Face recognition is a widely used system in real-time, the purpose of our project is for the security maintenance for the students in the college and the chance of entering strangers.

As well as to keep the data of the students for future references. It is a common sight in schools, colleges and educational institutions all over the world to see the teacher take student attendance at the beginning of each school day and sometimes even multiple times during the course of the day. Even in virtual classrooms taking attendance is a mandatory routine at the start of every class.

Tracking student attendance helps teachers to keep track of student absenteeism and punctuality. It helps schools understand the average percentage of students attending school each day in the given year and helps them find ways to avoid dropouts. This information when shared with the government authorities also helps them work on policies to reduce truancy and keeps students in school.

However, there are also many disadvantages of the manual attendance system it is a time-consuming process huge amount of paperwork involved reduces the classroom teaching time and productivity of the teacher. High scope for human error when tabulating attendance students can manipulate the system in many ways.

The process of manually tracking attendance has always been a complicated task. Over the years several solutions have been developed to record student attendance the most popular of these is a student attendance system.

Today, the student attendance system becoming a mandatory part of digital revolution, taking over schools all over the world they are becoming increasingly popular in educational institutes because of the many benefits of the college.

1.2 **Objective:**

The main objective of this project is to create a human face recognition system that accurately identifies students and records their attendance automatically. By using deep learning techniques and machine vision, we can train the system to recognize individual faces and match them with the student database. Our goal is to create a fast, reliable, and client system that can improve the attendance-taking process for teachers and students.

1.3 **Project Description:**

Our project will involve several stages, including data collection, pre-processing, model training, and testing. We will use a dataset of student images to train our deep learning model to recognize individual faces.

Once the model is trained, we will integrate it with the attendance-taking system to automate the process. The system will be able to capture images of students, compare them with the database, and update the attendance records in real-time.

1.4 **Project Scope:**

The scope of our project includes developing a proof-of-concept system for student attendance using face recognition. We aim to demonstrate the feasibility of using deep learning techniques for attendance-taking in educational institutions. However, there are limitations to this technology, including potential privacy concerns and the need for high-quality images for accurate recognition.

Therefore, our project will focus on a controlled environment with standardised lighting conditions and limited camera angles.

1.5 **Module Description:**

Our project will consist of three main modules: data collection and pre-processing, model training, and attendance-taking. The first module will involve collecting a dataset of student images and pre-processing them to remove noise and enhance features.

The second module will focus on training a deep learning model using TensorFlow to recognize individual faces. Finally, the attendance-taking module will use the trained model to capture student images, compare them with the database, and update attendance records in real-time. We will use the Python programming language and TensorFlow library to implement these modules.

CHAPTER - 2

LITERATURE REVIEW

[1] Viola, P., & Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001.

Viola and Jones proposed a boosted cascade of simple features for rapid object detection, including face detection, in their seminal work published in CVPR 2001. They used Haar-like features and AdaBoost to train a classifier that can detect faces with high accuracy and speed.

[2] Dalal, N., & Triggs, B. (2005). Histograms of oriented gradients for human detection. Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2005.

In this recognition, the Histograms of Oriented Gradients (HOG) method for human detection in their work published in CVPR 2005. This method extracts local image gradients and computes histograms of gradient orientations to capture local texture information. HOG has been widely used for face detection and achieved state-of-the-art performance.

[3] Lienhart, R., & Maydt, J. (2002). An extended set of Haar-like features for rapid object detection. Proceedings of the 2002 International Conference on Image Processing. ICIP 2002.

Lienhart and Maydt extended the set of Haar-like features for object detection, including face detection, in their work published in ICIP 2002. They added more complex features, such as tilted rectangles and two-rectangle features, to improve the detection performance.

[4] Yang, M. H., Kriegman, D. J., & Ahuja, N. (2002). Detecting faces in images: A survey. IEEE Transactions on Pattern Analysis and Machine Intelligence, 24(1), 34-58.

Yang et al. conducted a comprehensive survey on face detection in their paper published in TPAMI 2002. They reviewed various methods for face detection, including template matching, feature-based methods, and appearance-based methods.

[5] Zhang, Z., & Zhang, H. (2010). A survey of recent advances in face detection. Microsoft Research Asia Technical Report.

Zhang provided a survey of recent advances in face detection in their technical report published by Microsoft Research Asia in 2010. They reviewed various methods for face detection, including traditional methods based on hand-crafted features and recent methods based on deep learning.

[6] Jain, A. K., & Ross, A. (2010). Handbook of biometrics (Vol. 2). Springer Science & Business Media.

Jain and Ross edited the Handbook of Biometrics, which includes a chapter on face recognition. The chapter covers various aspects of face recognition, including face detection, feature extraction, and classification.

[7] Chen, X., & Kundu, K. (2018). Face detection and recognition: A survey. Journal of Electronic Imaging, 27(5), 051606.

Chen and Kundu published a survey on face detection and recognition in the Journal of Electronic Imaging in 2018. They reviewed various methods for face detection and recognition, including traditional methods and deep learning-based methods.

[8] Li, S., & Jain, A. K. (2011). Handbook of face recognition (Vol. 1). Springer Science & Business Media.

Li and Jain edited the Handbook of Face Recognition, which includes a chapter on face detection. The chapter covers various methods for face detection, including traditional methods and recent deep learning-based methods.

[9] Zhao, G., Pietikäinen, M., & Li, S. Z. (2018). Dynamic facial expression recognition: A review. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops.

Zhao et al. provided a review of dynamic facial expression recognition in their paper published in CVPR Workshops 2018. They reviewed various methods for dynamic facial expression recognition, including feature-based methods and deep learning-based methods.

[10] Zhang, Z., Yan, J., & Liu, S. (2019). Deep learning for face recognition: A comprehensive review. Neurocomputing, 338, 399-420.

Zhang et al. conducted a comprehensive review of deep learning for face recognition in their paper published in Neurocomputing 2019. They reviewed various deep learning-based methods for face recognition, including convolutional neural networks (CNNs), recurrent neural networks (RNNs), and generative adversarial networks (GANs).

2.1 SOFTWARE REQUIREMENTS ANALYSIS

2.1.1 Existing System

There are numerous face detection systems are available today, these systems are based on traditional methodologies. Most of the systems work under limitations.

*Outdated technologies

1. Manual attendance system
2. Iris detection system
3. Face ID detection system

Disadvantages of Existing System

1. Huge storage requirements. Machine learning technology requires powerful data storage.
2. Detection can be vulnerable. We've outlined the way in which facial detection can be thrown off.
3. Potential privacy issues. There is disagreement on whether face detection is compatible with human privacy rights.

2.2.1 Proposed System

The proposed method is based on the Face Net and MTCNN, this project is proposed to manage the attendance system via face recognition. Face Recognition is a software application that verifies you identify or confirms a person's identity using their face.

Security systems use face recognition to uniquely identify individuals during user logins as well as strengthen user authentication activity. Face Recognition works in three steps:

1. DETECTION
2. ANALYSIS
3. RECOGNITION

Advantages of the Proposed System

- Increased efficiency and capability
- Enhance workplace security
- Easy integration with other systems
- Easy to manage records
- Time-saving for your workforce
- Create a safe environment

2.3.1 **Module Description:**

The proposed system will consist of two main modules:
the user module and the trained module.

2.3.1 **User Module**:

The user module will be used by the faculty to mark the attendance of students. The module will have the following functionalities:

Interface: The faculty will be able enter details of student name and roll no into the system using their credentials.

Dashboard: The faculty will be able to view the attendance records of students on the dashboard.

Mark Attendance: The faculty will be able to mark the attendance of students by clicking a picture of the students through a webcam.

2.3.2 **Trained Module**:

The trained module will be responsible for recognizing the faces of students and marking their attendance. The module will have the following functionalities:

Face Detection: The module will detect the face of students in the picture captured by the webcam.

Face Recognition: The module will recognize the faces of students and match them with the existing database of students.

Mark Attendance: The module will mark the attendance of the students in the database.

2.3.3 **Functionalities of User Module**:

The user module will allow faculty to use interface and view attendance records of students.

The module will enable faculty to mark attendance using the webcam.

The user module will generate attendance reports for the faculty.

2.3.4 **Functionalities of Trained Module**:

The trained module will detect and recognize the faces of students accurately and quickly.

The module will mark the attendance of students automatically.

The trained module will update the attendance database in real-time.

CHAPTER-3

ORGANIZATION OF THE THESIS

3.1 Software Requirements:

Technology: TensorFlow, OpenCV, NumPy, Keras

- TensorFlow is an open-source machine learning framework used for building deep neural networks. It is the primary technology used for developing the face recognition model in this project.
- OpenCV (Open-Source Computer Vision Library) is a free and open-source computer vision and machine learning software library used for real-time image processing and analysis.
- NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.
- Keras is a high-level neural networks API, written in Python and capable of running on top of TensorFlow. It is used for building and training the deep learning models in this project.

Languages and Modules Used: Python, Python idle

- Python is a popular programming language used for machine learning and data science applications. It is the primary language used for developing the face recognition project.
- Python idle is an integrated development environment (IDE) for Python that allows developers to write, test, and debug Python code.

Algorithms: Convolutional Neural Network (CNN), Facial Recognition algorithms

- Convolutional Neural Network (CNN) is a deep learning algorithm commonly used for image classification, segmentation, and recognition tasks. It is the primary algorithm used for developing the face recognition model in this project.

- Facial recognition algorithms are used for detecting and recognizing faces in images or videos. These algorithms are used in conjunction with the CNN model to recognize and identify students in the attendance system.

Hosting: Local Hosting

- The face recognition system is hosted locally on the user's computer. This means that the system runs on the user's computer and does not require any external hosting or cloud services.

3.2 Hardware Requirements:

Processor: Intel Core i5 or higher

- The face recognition system requires a powerful processor to handle the complex machine learning computations involved in the recognition process. An Intel Core i5 or higher processor is recommended for optimal performance.

Hard Disk: Minimum 100 GB of free space

- The face recognition system requires a significant amount of storage space to store the training data, deep learning models, and other related files. A minimum of 100 GB of free space is recommended for the system to operate smoothly.

RAM: Minimum 8 GB of RAM

- The face recognition system requires a significant amount of memory to store the training data, deep learning models, and to perform the recognition process. A minimum of 8 GB of RAM is recommended for the system to operate smoothly.

3.3 Techniques and Algorithms

NUMPY:

- NumPy provides a high-performance array object, called ndarray, which allows you to store and manipulate large datasets.
- The ndarray object provides a number of attributes, including shape (the dimensions of the array), size (the total number of elements in the array), and dtype (the data type of the array).
- NumPy provides a wide range of mathematical functions, including functions for performing basic operations (such as addition, subtraction, multiplication, and division), as well as more advanced functions (such as trigonometric functions, logarithmic functions, and statistical functions).
- NumPy provides a number of tools for creating arrays, including functions for creating arrays from existing data, such as lists or tuples, as well as functions for generating arrays with specific properties, such as arrays with random values or arrays with a specific shape.
- NumPy arrays can be indexed and sliced in a variety of ways, allowing you to extract subsets of the data or perform operations on specific elements of the array.
- NumPy provides a number of tools for manipulating arrays, including functions for reshaping arrays, concatenating arrays, and splitting arrays.
- NumPy provides a number of tools for performing linear algebra operations, including functions for computing dot products, solving linear systems, and computing eigenvalues and eigenvectors.
- NumPy provides a number of tools for performing Fourier transforms and other signal processing operations.
- NumPy provides a number of tools for working with missing or invalid data, including functions for filling in missing values, removing missing values, and computing statistics on data with missing values.
- NumPy provides a number of tools for working with dates and times, including functions for creating and manipulating datetime objects.
- NumPy provides a number of tools for working with images and video, including functions for reading and writing image and video files, as well as functions for manipulating and processing image and video data.
- NumPy provides a number of tools for working with text data, including functions for parsing and manipulating text data.

- NumPy provides a number of tools for working with graph data, including functions for generating and manipulating graphs, as well as functions for computing graph algorithms.
- NumPy provides a number of tools for working with sparse data, including functions for creating and manipulating sparse matrices.
- NumPy provides a number of tools for working with complex numbers, including functions for performing complex arithmetic, computing complex roots, and computing complex logarithms.
- NumPy provides a number of tools for working with probability distributions, including functions for generating random numbers from specific distributions and functions for computing statistics on data with specific distributions.
- NumPy provides a number of tools for working with polynomials, including functions for computing roots and coe cients of polynomials, as well as functions for evaluating polynomials.
- NumPy provides a number of tools for working with special functions, including functions for computing Bessel functions, Legendre functions, and other special functions.
- NumPy provides a number of tools for working with interpolation and approximation, including functions for interpolating data, approximating functions, and smoothing data.
- NumPy provides a number of tools for working with probability theory, including functions for computing probabilities, conditional probabilities, and Bayes' theorem.

TENSORFLOW:

- TensorFlow is an open-source library developed by Google for numerical computation and machine learning.
- It was initially released in November 2015 and has since become one of the most popular machine learning frameworks.
- TensorFlow is written in C++ and has APIs available for Python, C++, Java, Go, and other programming languages.
- TensorFlow provides a variety of tools and resources for building and training machine learning models, including APIs for building neural networks and other models.
- One of the key features of TensorFlow is its ability to run on both CPUs and GPUs, allowing for faster computation of machine learning models.

- TensorFlow provides a high-level API called Keras, which simplifies the process of building and training machine learning models.
- TensorFlow also supports distributed computing, allowing for the training of large-scale machine learning models across multiple devices and machines.
- TensorFlow has a large and active community of developers, which has contributed to the development of numerous add-ons and libraries.
- TensorFlow can be used for a variety of applications, including image and speech recognition, natural language processing, and robotics.
- TensorFlow provides built-in support for various neural network architectures, such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs).
- TensorFlow allows for the creation of custom operations and layers, making it highly flexible and customizable.
- TensorFlow provides various optimization techniques, such as gradient descent and Adam optimization, for training machine learning models.
- TensorFlow can be used for both supervised and unsupervised learning.
- TensorFlow has a built-in debugger that can help developers identify and fix issues in their code.
- TensorFlow supports various data formats, including NumPy arrays and TensorFlow's own tensor data structure.
- TensorFlow provides APIs for deploying machine learning models to production environments, such as mobile devices and the cloud.
- TensorFlow has a built-in visualization tool called TensorBoard, which allows for the visualization of machine learning models and their performance.
- TensorFlow can be used with other popular machine learning libraries, such as scikit-learn and PyTorch.
- TensorFlow provides various techniques for preventing overfitting, such as regularization and early stopping.
- TensorFlow is constantly evolving, with frequent updates and improvements being made to the framework.

MTCNN:

- The MTCNN algorithm is based on a series of convolutional neural networks (CNN) that are trained to perform the three tasks simultaneously.
- MTCNN uses a three-stage process to detect faces in an image. The first stage is a coarse detection, followed by a more refined detection in the second and third stages.
- The first stage of MTCNN is a proposal network that generates candidate bounding boxes for faces.
- The second stage of MTCNN is a refinement network that refines the bounding boxes generated by the proposal network.
- The third stage of MTCNN is a landmark network that estimates the facial landmarks (e.g., eyes, nose, mouth) of the detected faces.
- MTCNN uses a non-maximum suppression (NMS) algorithm to eliminate redundant bounding boxes and keep only the most accurate detections.
- The MTCNN algorithm is highly accurate, with state-of-the-art performance on various benchmark face detection datasets.
- MTCNN is also highly with real-time processing speeds on standard hardware.
- MTCNN has been used in various face recognition applications, such as surveillance systems, facial expression analysis, and emotion recognition.
- MTCNN is capable of detecting faces of dierent sizes, orientations, and poses.
- The MTCNN algorithm can handle occlusions (e.g., glasses, hats) and variations in lighting conditions.
- MTCNN can be fine-tuned on specific datasets to improve its performance on a particular application.
- MTCNN is an open-source algorithm, and its implementation is available in popular deep learning frameworks such as TensorFlow, Keras, and PyTorch.
- MTCNN has been used in combination with other face recognition algorithms, such as FaceNet andVGG-Face, to achieve state-of-the-art performance on face recognition tasks.
- MTCNN can be used for face detection in both images and videos.
- MTCNN has been shown to outperform other popular face detection algorithms, such as Viola-Jones and HOG.
- MTCNN can be used for real-time face detection and recognition in mobile devices and other embedded systems.

- MTCNN is a highly e ective algorithm for face detection and recognition, and its performance is continually improving with ongoing research and development in the field of deep learning.

CNN (Convolutional Neural Networks):

- CNNs consist of a series of convolutional layers, which extract features from the input image. These features are learned through training the network on a large dataset of labeled images.
- The convolutional layers are followed by pooling layers, which downsample the feature maps to reduce the spatial resolution of the image.
- CNNs also typically include fully connected layers, which perform classification or regression on the learned features.
- CNNs use a variety of activation functions, such as ReLU, to introduce nonlinearity into the network and improve its performance.
- Dropout is a regularization technique commonly used in CNNs to prevent overfitting, where some neurons in the network are randomly dropped during training.
- Transfer learning is a technique where pre-trained CNN models are used as a starting point for a new task, allowing for faster and more accurate training.
- Data augmentation is another technique used in CNNs to increase the size of the training set by generating new images through transformations such as rotations, flips, and scaling.
- CNNs are particularly e ective for tasks such as image classification, object detection, and facial recognition.
- The architecture of a CNN can vary widely depending on the specific task and the characteristics of the input data.
- CNNs are often used in combination with other techniques such as sliding window detection or region proposals to identify objects within an image.
- The performance of a CNN is often measured using metrics such as accuracy, precision, and recall.
- Gradient-based optimization algorithms such as stochastic gradient descent (SGD) and Adam are commonly used to train CNNs.

- One of the challenges of training CNNs is avoiding overfitting, where the network becomes too specialized to the training data and does not generalize well to new images.
- To address overfitting, techniques such as regularization, early stopping, and weight decay are often used.
- CNNs can be implemented using a variety of deep learning frameworks such as TensorFlow, Keras, and PyTorch.
- Hyperparameter tuning, where the values of various parameters such as learning rate and batch size are optimized, is an important part of training CNNs.
- CNNs can be used for a variety of tasks beyond image recognition, such as natural language processing and time series analysis.
- CNNs are often used in combination with other types of neural networks such as recurrent neural networks (RNNs) and generative adversarial networks (GANs).
- CNNs can also be used in unsupervised learning tasks such as feature learning and clustering.
- The development of CNNs has enabled significant advances in fields such as computer vision, autonomous driving, and medical image analysis.

OPENCV:

- OpenCV (Open Source Computer Vision Library) is an open-source computer vision and machine learning software library. Here are some points that explain OpenCV in detail:
- OpenCV was initially developed by Intel in 1999 for real-time computer vision applications. It was later released as an open-source library for public use.
- It is written in C++ and has bindings for Python, Java, and MATLAB.
- OpenCV supports a wide range of image and video processing functions, including basic operations like filtering, thresholding, and feature detection.
- It provides a wide range of computer vision algorithms such as object detection, recognition, and tracking.
- OpenCV also supports machine learning algorithms such as neural networks and decision trees.
- It provides support for both desktop and mobile platforms, including iOS and Android.
- OpenCV has a modular structure that allows users to easily add new features or modify existing ones.

- It provides pre-trained models for tasks like face detection, facial recognition, and object detection.
- OpenCV supports various image file formats such as JPEG, PNG, and TIFF.
- It can read video streams from webcams, files, or network streams.
- OpenCV supports various camera interfaces such as V4L2, Firewire, and GigE Vision.
- It provides tools for camera calibration, which is necessary for accurate 3D reconstruction.
- OpenCV can be used for image and video compression, including video encoding and decoding.
- It also provides tools for creating graphical user interfaces (GUIs) for computer vision applications.
- OpenCV provides various optimization techniques for faster computation, such as parallelization and GPU acceleration.
- It provides support for stereo vision, which allows for depth perception from two or more images.
- OpenCV can be used for robotics applications, including objecttracking and localization.
- It has a large and active user community, with support available on forums, mailing lists, and social media platforms.
- OpenCV is constantly evolving, with new features and improvements being added with each release.
- It is a free and open-source library, making it accessible to researchers, students, and hobbyists.

KERAS

- Keras is an open-source neural network library written in Python.
- It is designed to provide a user-friendly and modular approach to building and training deep learning models.
- Keras allows for the easy and fast prototyping of deep learning models and has support for a wide range of neural network architectures.
- It is built on top of lower-level deep learning libraries such as TensorFlow, Theano, and CNTK.

- Keras provides a simple and intuitive interface for defining neural networks, allowing users to quickly create and experiment with di erent network architectures.
- It has a large community of users and contributors, and there are many pre-trained models and examples available for use.
- Keras supports a range of di erent types of layers, including convolutional layers, recurrent layers, and dense layers.
- It includes a range of activation functions, such as relu, sigmoid, and tanh, that can be used in di erent layers.
- Keras supports a range of loss functions, including mean squared error, binary cross-entropy, and categorical cross-entropy.
- It also includes a range of optimization algorithms, such as stochastic gradient descent, RMSprop, and Adam.
- Keras allows for the creation of complex models, including models with multiple inputs and outputs.
- It includes support for data pre-processing and augmentation, making it easier to work with di erent types of data.
- Keras has support for distributed training, allowing users to train models across multiple machines.
- It has a built-in evaluation framework, allowing users to easily evaluate the performance of their models.
- Keras supports both CPU and GPU acceleration, allowing users to take advantage of the power of GPUs to train models faster.
- It includes support for model saving and loading, making it easier to reuse models and continue training from where you left o .
- Keras also includes support for call-backs, allowing users to customize the training process by adding custom functionality.
- It has a range of built-in datasets, such as CIFAR-10 and MNIST, that can be used for testing and experimentation.
- Keras is widely used in industry and academia, and has been used to build a range of different types of deep learning models.
- It is constantly evolving, with new features and improvements being added regularly.

Haar Cascade:

- Haar Cascade is a popular object detection algorithm used for detecting faces, eyes, nose, mouth, and other objects. Here are 20 points explaining how Haar Cascade works:
- Haar Cascade is a machine learning-based approach where a cascade function is trained from many positive and negative images.
- The positive images are the images that contain the object we want to detect, while the negative images are the images that do not contain the object.
- Haar features are used to detect the object in the images. These features are rectangles with di erent sizes and positions that can detect di erent patterns of light and dark areas in the image.
- Haar features are calculated for each image in the training set, and the algorithm selects the best features that can separate the positive and negative images.
- The algorithm creates a decision tree for each feature, which decides whether the image contains the object or not.
- The decision trees are combined to create a strong classifier, which can detect the object in the image with high accuracy.
- The algorithm uses the AdaBoost algorithm to select the best features and create the decision trees.
- AdaBoost is a machine learning algorithm that trains weak classifiers and combines them to create a strong classifier.
- The Haar Cascade algorithm uses a sliding window approach to detect the object in the image. The window moves across the image, and the classifier is applied to each window to detect the object.
- The Haar Cascade algorithm uses the Viola-Jones framework to detect the object in the image. This framework uses integral images to speed up the calculation of Haar features.
- Integral images are pre-calculated images that represent the sum of pixels in a rectangular area of the original image. This makes the calculation of Haar features much faster.
- The Haar Cascade algorithm is trained on a large dataset of images, which includes di erent angles, lighting conditions, and backgrounds.

- The algorithm can detect the object in the image even if it is partially occluded or rotated.
- Haar Cascade is a supervised learning algorithm, which means it requires labeled data to train the classifier.
- The algorithm can be fine-tuned by adjusting the parameters, such as the scale factor, which determines the size of the sliding window, and the minimum neighbors, which determines how many neighbors the detected object should have.
- The algorithm can be used for real-time object detection in video streams or on live camera feeds.
- The Haar Cascade algorithm is widely used for face detection, eye detection, pedestrian detection, and other object detection tasks.
- The Haar Cascade algorithm has high accuracy and low false positive rates, which makes it suitable for many applications.
- The algorithm is implemented in many computer vision libraries, such as OpenCV, which makes it easy to use and integrate into applications.
- The Haar Cascade algorithm is an important algorithm in computer vision, which has paved the way for many other object detection algorithms, such as the Deep Learning-based object detection algorithms.

WORKING OF FACE RECOGNITION:

Face recognition is a biometric technology that involves identifying or verifying a person's identity based on their facial features.

The process begins with capturing an image or a video of a person's face using a camera.

The image or video is then pre-processed to remove any noise or distortion in the image that may etc the recognition process.

The pre-processed image is then converted into a feature vector using various techniques such as Principal Component Analysis (PCA), Local Binary Patterns (LBP), or Histogram of Oriented Gradients (HOG).

The feature vector is then used to train a machine learning model, such as a Convolutional Neural Network (CNN), to recognize faces.

During the training process, the machine learning model learns to recognize the unique features of each person's face and stores them as a set of weights or parameters.

Once the machine learning model is trained, it can be used to recognize faces in real-time. This involves capturing a new image or video and passing it through the model.

The model compares the new feature vector to the ones it has learned during training and determines which person the image or video belongs to.

The accuracy of face recognition depends on the quality of the image or video, the size of the dataset used to train the machine learning model, and the algorithm used for feature extraction and classification.

To improve accuracy, face recognition systems often use multiple algorithms or techniques in combination.

The performance of face recognition systems can also be improved by using techniques such as face detection, where the system first detects the location of faces in the image or video before attempting to recognize them.

Face recognition systems can be used for a variety of applications such as security, surveillance, access control, and identity verification.

Some of the challenges in face recognition include variations in lighting, facial expression, and pose.

One way to overcome these challenges is to use techniques such as 3D face recognition, which captures the shape and depth of the face in addition to its texture.

Another technique is to use multiple images or videos of the same person taken from different angles or under different lighting conditions to improve recognition accuracy.

Face recognition systems can also be used in combination with other biometric technologies such as fingerprint recognition or iris recognition to provide even greater accuracy and security.

Privacy concerns are also a major consideration when implementing face recognition systems, and appropriate measures must be taken to protect the privacy of individuals.

Advancements in machine learning and computer vision technologies are continually improving the accuracy and performance of face recognition systems.

The use of deep learning techniques such as CNNs and Recurrent Neural Networks (RNNs) is also increasing the accuracy and speed of face recognition systems.

As these technologies continue to evolve, face recognition is likely to become even more accurate and reliable, with the potential for a wide range of applications in various industries and sectors.

CHAPTER – 4

SOFTWARE DESIGN

4.1 **Data-Flow Diagrams**:

Data flow diagrams is a graphical tool used to describe and analyze the movement of data through a system manual or automated including the process, stores of data and delay in the system.

Level 0 **DFD:**

The main process is the "Student Attendance System"

The inputs are the student faces and the attendance records

The outputs are the attendance reports and the updated attendance records

There are two subprocesses: "Face Recognition" and "Attendance Management"

Level 1 **DFD:**

The "Face Recognition" subprocess takes the student faces as input and performs the following steps:

Detects and extracts facial features

Preprocesses and normalises the face images

Uses a trained deep learning model to recognize the students' faces

Returns the recognized student ID to the main process

The "Attendance Management" subprocess takes the recognized student ID and the attendance records as inputs and performs the following steps:

Updates the attendance records with the recognized student ID and the current timestamp

Generates the attendance reports based on the updated attendance records

Returns the attendance reports to the main process

Overall, the system follows a simple data flow architecture where the inputs are processed by the subprocesses, and the outputs are returned to the main process. The face recognition subprocess is responsible for recognizing the student faces,

while the attendance management subprocess is responsible for managing the attendance records and generating the attendance reports.

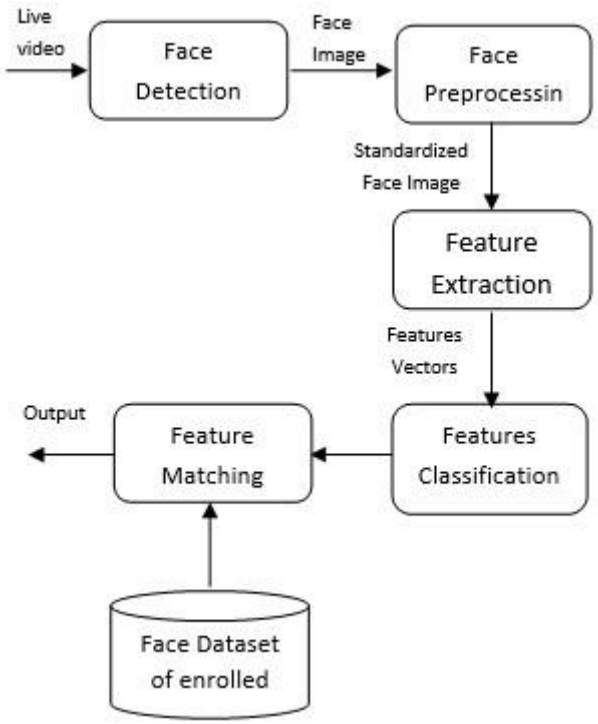


Fig 4.1 workflow of face recognition

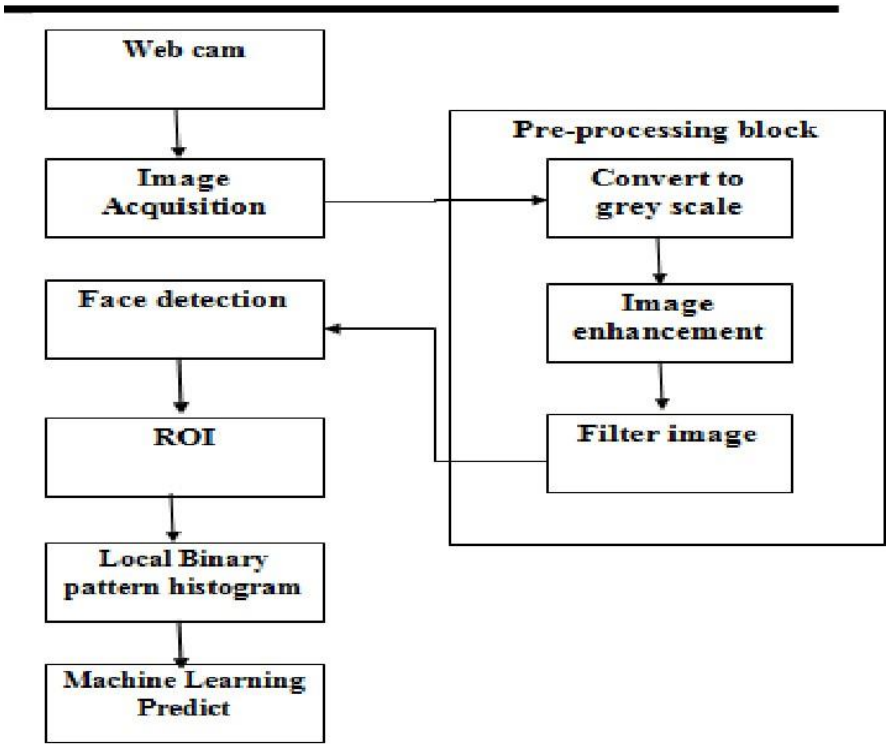


Fig. 6 block diagram

Fig 4.2 workflow of the project

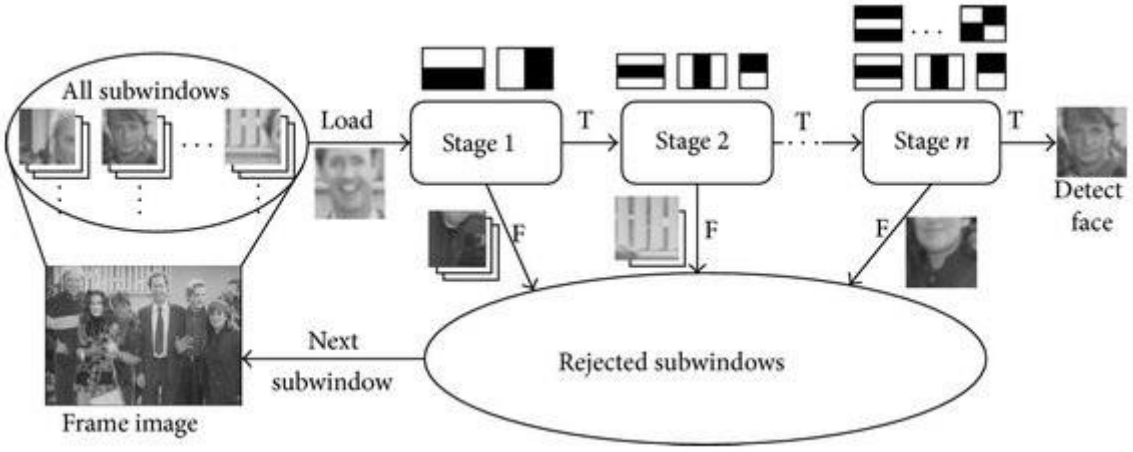
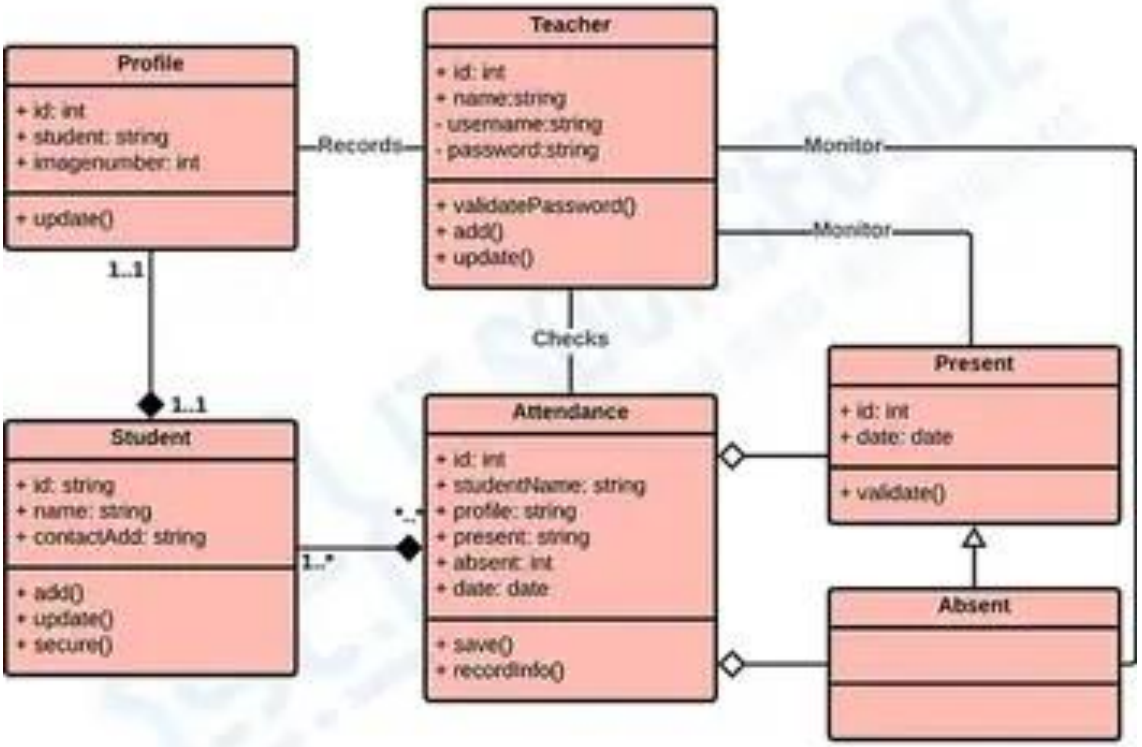


Fig 4.3 how the haar cascade file works **UML diagram**

FACE RECOGNITION ATTENDANCE SYSTEM



CLASS DIAGRAM

CHAPTER – 5

CODING

5.1 INTERFACE.PY FILE

```
import tkinter as tk from
    tkinter import
    filedialog import os
    import subprocess

class Application:
def __init__(self, master):
    self.master = master
master.title("Face Recognition Interface") master.geometry("500x400")

# Pastel color scheme
    self.bg_color="#F9DCD1"
    self.button_color = "#E0BBE4"
    self.label_color = "#E7CECB"

self.master.configure(bg=self.bg_color)

self.label = tk.Label(master, text="Select an option:", bg=self.label_color,
font=("Arial", 20))
    self.label.pack(pady=20)

self.capture_name_label = tk.Label(master, text="Enter name:",
bg=self.bg_color, font=("Arial", 14))
    self.capture_name_label.pack()

 self.capture_name_entry = tk.Entry(master, font=("Arial", 14))
    self.capture_name_entry.pack()

 self.capture_roll_label = tk.Label(master, text="Enter roll no:", bg=self.bg_color,
 font=("Arial", 14))
```

```
self.capture_roll_label.pack()

    self.capture_roll_entry=tk.Entry(master, font=("Arial",14))

    self.capture_roll_entry.pack()


self.capture_button = tk.Button(master, text="Capture Images",
command=self.capture_images, bg=self.button_color, font=("Arial", 16), height=2)
    self.capture_button.pack(pady=20)

self.train_button = tk.Button(master, text="Train Model",
command=self.train_model, bg=self.button_color, font=("Arial", 16), height=2)
    self.train_button.pack(pady=20)

self.recognize_button = tk.Button(master, text="Recognize Faces",
command=self.recognize_faces, bg=self.button_color, font=("Arial", 16), height=2)
    self.recognize_button.pack(pady=20)

self.attendance_button = tk.Button(master, text="Attendance sheet",
command=self.open_attendance_sheet, bg=self.button_color,
    font=("Arial",16), height=2)

self.attendance_button.pack(pady=20)

def capture_images(self):
name = self.capture_name_entry.get()

    roll_no=self.capture_roll_entry.get()

    subprocess.call(['python','images.py',name,roll_no])

def train_model(self):
    subprocess.call(['python','train_model.py'])

def recognize_faces(self):
    subprocess.call(['python', 'recognize_faces.py'])


 def open_attendance_sheet(self):
```

```
        os.startfile('attendance.csv')

    root = tk.Tk()
app=Application(root)
root.mainloop()
```

5.2 IMAGES.PY FILE

```
import argparse

parser = argparse.ArgumentParser()
parser.add_argument("name", help="Name of the person")
parser.add_argument("roll_no", help="Roll number of the person")
args = parser.parse_args()

name = args.name
  roll_no =args.roll_no

import cv2
import os from mtcnn
import MTCNN

# Create folder to store captured
   images if not
   os.path.exists('student_images'):
   os.makedirs('student_images')

# Initialize MTCNN for face detection
  detector = MTCNN()
```

```
    # Create folder to store images for current student folder_path
      = os.path.join('student_images', f'{name}_{roll_no}') if not
      os.path.exists(folder_path): os.makedirs(folder_path)

    # Capture images
    num_images = 0
    camera = cv2.VideoCapture(0)
      while num_images < 100: ret,
      frame = camera.read() if not
      ret: print("Failed to capture
      image") continue

    # Detect face using MTCNN

    result = detector.detect_faces(frame)
      if len(result) > 0:
    # Save face region as image x, y,
      w, h = result[0]['box']
     face_img = frame[y:y+h, x:x+w]
            cv2.imwrite(os.path.join(folder_path, f'{name}_{roll_no}_{num_images}.jpg'),
    face_img)
      num_images += 1

cv2.imshow('Capture Images', frame) if
      cv2.waitKey(1) and 0xFF ==ord('q'):
  break
    camera.release()
      cv2.destroyAllWindows()
```

5.3 **TRAIN_MODEL.PY FILE**

```
import os
import cv2
import numpy as
    np
from sklearn.preprocessing import LabelEncoder from
    tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Flatten, Conv2D, MaxPooling2D, Dropout
from mtcnn import MTCNN

# Path to the folder containing captured images
    data_path = "student_images"

# Initialize MTCNN face detector
    detector = MTCNN()

# Load images and labels images = [] labels = [] for
    folder_name in os.listdir(data_path):
 folder_path = os.path.join(data_path, folder_name)
    for filename in os.listdir(folder_path):
img_path = os.path.join(folder_path, filename)
img = cv2.imread(img_path)
img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    results = detector.detect_faces(img_rgb)
if results:
x1, y1, w, h = results[0]["box"] x2,
    y2 = x1 + w, y1 + h
face = img_rgb[y1:y2, x1:x2]
face = cv2.resize(face, (160, 160))
    images.append(face)
    labels.append(folder_name)
# Encode labels
label_encoder = LabelEncoder() labels =
    label_encoder.fit_transform(labels)
```

```
# Convert data to numpy arrays
    images = np.array(images) labels =
    np.array(labels)

# Normalize image data
images = images.astype('float32') / 255.0

# Create model architecture
    model = Sequential()
model.add(Conv2D(32, (3, 3), activation='relu', input_shape=(160, 160,3)))
model.add(MaxPooling2D((2, 2))) model.add(Conv2D(64, (3, 3),
    activation='relu'))
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(MaxPooling2D((2, 2)))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(len(set(labels)), activation='softmax'))

# Compile model
model.compile(optimizer='adam',
    loss='sparse_categorical_crossentropy', metrics=['accuracy'])

# Train model
model.fit(images, labels, epochs=10, validation_split=0.2)

# Save model weights
model.save('face_recognition_model.h5')

# Save label encoder
np.save('label_encoder.npy', label_encoder.classes_)
```

5.4 **RECOGNIZE_FACES.PY FILE**

```
import cv2
import numpy
as np
import os
import csv
from datetime
 import datetime from
datetime
import date
import time

# Constants
FACE_SIZE = 160
TOLERANCE = 0.9

# Load face detection model
face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')

# Load student images student_images = {}
for student_name in
os.listdir('student_images'):
        student_folder =
        os.path.join('student_images',student_name)
        if os.path.isdir(student_folder):
          student_images[student_name] = []
for img_file in os.listdir(student_folder):
img_path = os.path.join(student_folder, img_file)
student_images[student_name].append(img_path)
```

```
# Initialize attendance record attendance = {}

 for student_name in student_images.keys():
     attendance[student_name] = []

     # Initialize video
     capture cap =
     cv2.VideoCapture(0)

     while True:
     # Read frame from video capture
     ret, frame = cap.read()

  # Detect faces in the frame
  gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

  faces = face_cascade.detectMultiScale(gray, 1.3, 5)

  # Process each detected face for
  (x, y, w, h) in faces:
  # Extract the face from the frame
  face = frame[y:y+h, x:x+w]
  face = cv2.resize(face, (FACE_SIZE, FACE_SIZE)) face_gray =
  cv2.cvtColor(face, cv2.COLOR_BGR2GRAY)

  # Calculate the normalized histogram for the face

  hist = cv2.calcHist([face_gray], [0], None, [256], [0, 256])

  hist = cv2.normalize(hist, hist)

  # Compare the face histogram to each student's histograms
  matched = False for student_name, student_image_paths in
  student_images.items():
```

```
for image_path in student_image_paths:
img = cv2.imread(image_path)
img_gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
img_hist = cv2.calcHist([img_gray], [0], None, [256], [0, 256])
img_hist = cv2.normalize(img_hist, img_hist)
score = cv2.compareHist(hist, img_hist, cv2.HISTCMP_CORREL)
        # If no student is matched, mark the face as unknown if not
       matched:
           # Draw a red rectangle around the face color = (0,
           0, 255) cv2.rectangle(frame, (x, y), (x+w, y+h),
           color, 2)

           # Draw 'Unknown' next to the rectangle text =
           'Unknown'
           text_size = cv2.getTextSize(text, cv2.FONT_HERSHEY_SIMPLEX, 0.9, 2)[0]
           text_x = x text_y = y - 10 text_w = text_size[0]
           text_h = text_size[1]
           cv2.rectangle(frame, (text_x, text_y), (text_x + text_w, text_y + text_h)(0,
      0, 255), cv2.FILLED)
      cv2.putText(frame, student_name,
      (x,y-20),cv2.FONT_HERSHEY_SIMPLEX,
           1.0,(255, 255, 255), 2)
      # If the score is above the threshold, mark the student as present if score
           >= TOLERANCE:
      attendance[student_name].append('present')

      # Draw a green rectangle around the face color = (0,
           255, 0) cv2.rectangle(frame, (x, y), (x+w, y+h),
           color, 2)

      # Draw the student's name next to the rectangle text =
           student_name
      text_size = cv2.getTextSize(text, cv2.FONT_HERSHEY_SIMPLEX, 0.9, 2)[0]
      text_x = x text_y = y - 10 text_w = text_size[0]
      text_h = text_size[1]
```

```
cv2.rectangle(frame, (text_x, text_y), (text_x + text_w, text_y + text_h), (0,
255, 0), cv2.FILLED)

cv2.putText(frame, student_name, (x, y-20),
      cv2.FONT_HERSHEY_SIMPLEX, 1.0, (255, 255, 255), 2)

matched = True
break # Exit the loop over student image paths

'''
if matched == True:
# Draw a green rectangle around the face
color = (0, 255, 0) cv2.rectangle(frame, (x, y),
(x+w, y+h),color, 2)

# Draw the student's name next to the rectangle text =
      f'{student_name}'
text_size = cv2.getTextSize(text, cv2.FONT_HERSHEY_SIMPLEX, 0.9,
2)[0]
      text_x = x text_y = y - 10
      text_w = text_size[0]
      text_h = text_size[1]
cv2.rectangle(frame, (text_x, text_y), (text_x + text_w, text_y + text_h),
color, cv2.FILLED) cv2.putText(frame, text,
(text_x, text_y + text_h),
cv2.FONT_HERSHEY_SIMPLEX, 0.9, (255, 255, 255), 2)
'''

'''
# If no student is matched, mark the face as unknown if
      not matched:
# Draw a red rectangle around the face color = (0,
      0, 255) cv2.rectangle(frame, (x, y), (x+w, y+h),
      color, 2)

# Draw 'Unknown' next to the rectangle text =
      'Unknown'
```

```
text_size = cv2.getTextSize(text, cv2.FONT_HERSHEY_SIMPLEX, 0.9, 2)[0]
text_x = x text_y = y - 10
text_w = text_size[0]
text_h = text_size[1]
cv2.rectangle(frame, (text_x, text_y), (text_x + text_w, text_y + text_h),
(0, 0,255), cv2.FILLED)
#cv2.putText(frame, student_name, (x, y-20),cv2.FONT_HERSHEY_SIMPLEX,
1.0, (255, 255, 255), 2)
'''
# Show the frame with the detected faces
    cv2.imshow('frame', frame)

# Exit if the 'q' key is pressed if
    cv2.waitKey(1) & 0xFF ==
    ord('q'): break

# Release video capture and close all
    windows cap.release()
    cv2.destroyAllWindows()

# Write attendance record to CSV file with
    open('attendance.csv', 'w', newline='') as
    csvfile:
fieldnames = ['Name', 'Attendance','date','time']
writer = csv.DictWriter(csvfile, fieldnames=fieldnames)
    writer.writeheader()
for student_name, record in attendance.items():
writer.writerow({'Name':student_name,
'Attendance':len(record),'date':date.today(),'time': time.ctime(1627908313.717886)})

# Print the attendance record
    print(attendance)
```

Chapter - 6

OUTPUT IMAGES

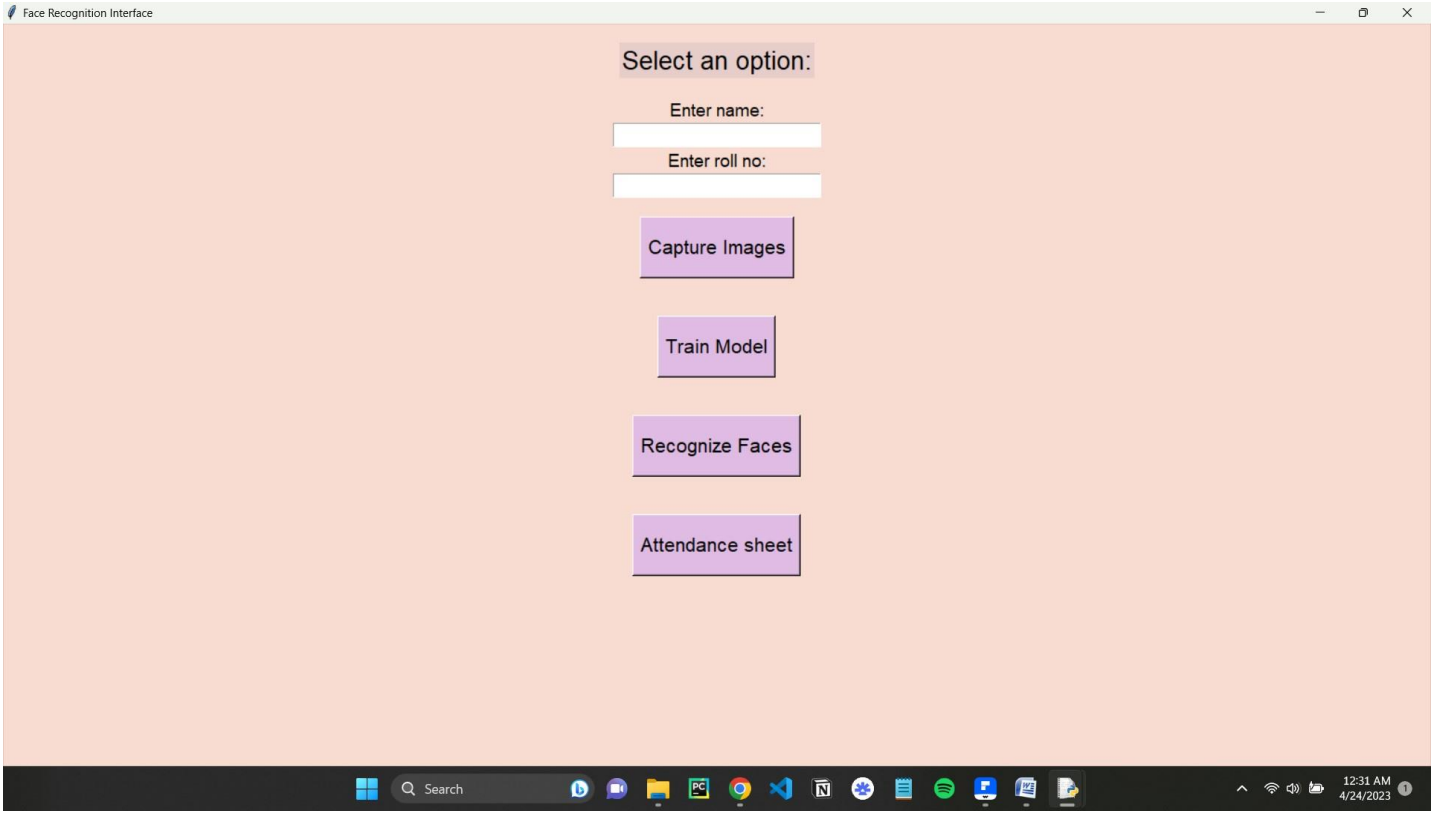


Fig 6.1 Interface of the program

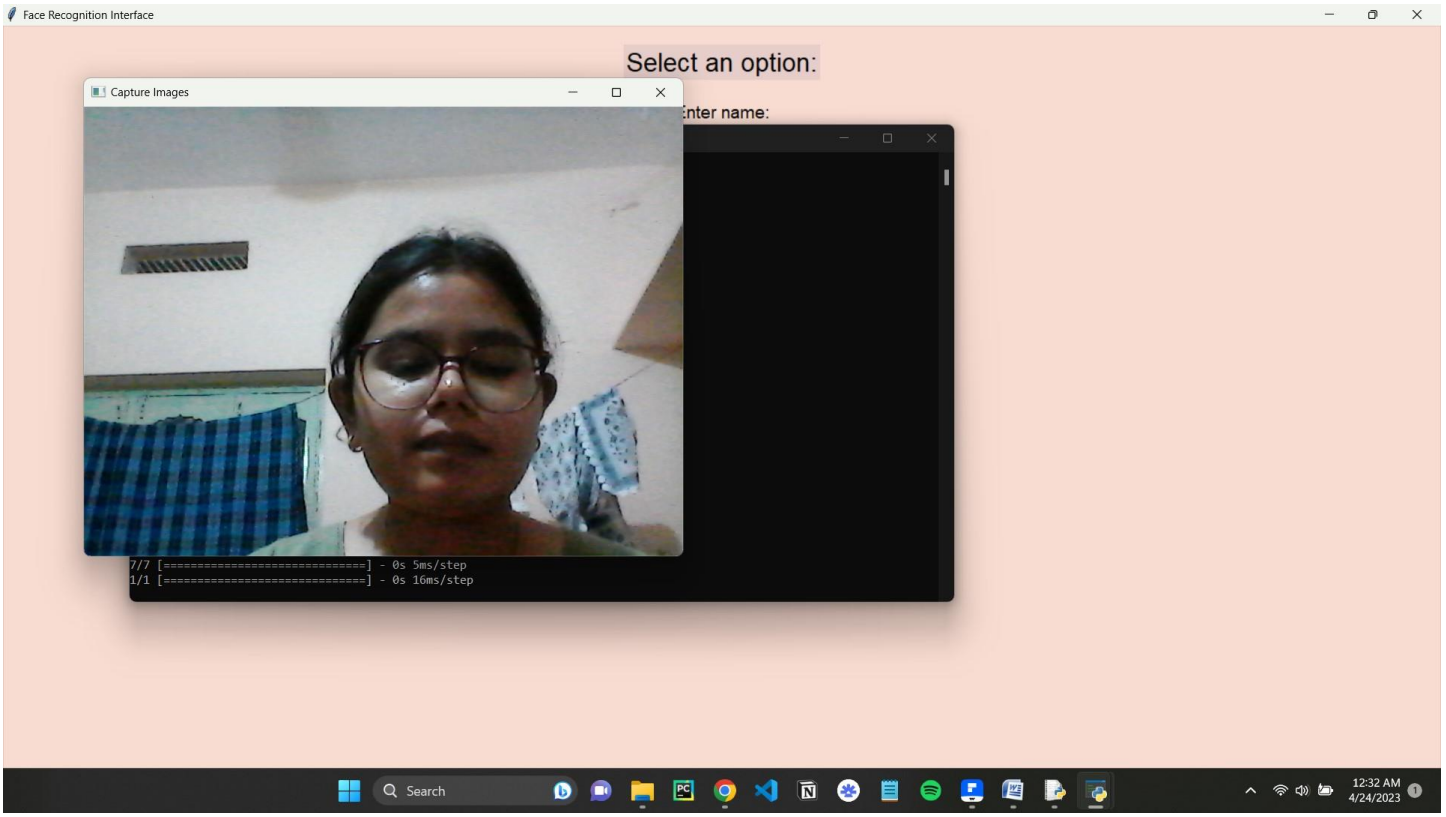


Fig 6.2 capturaing images

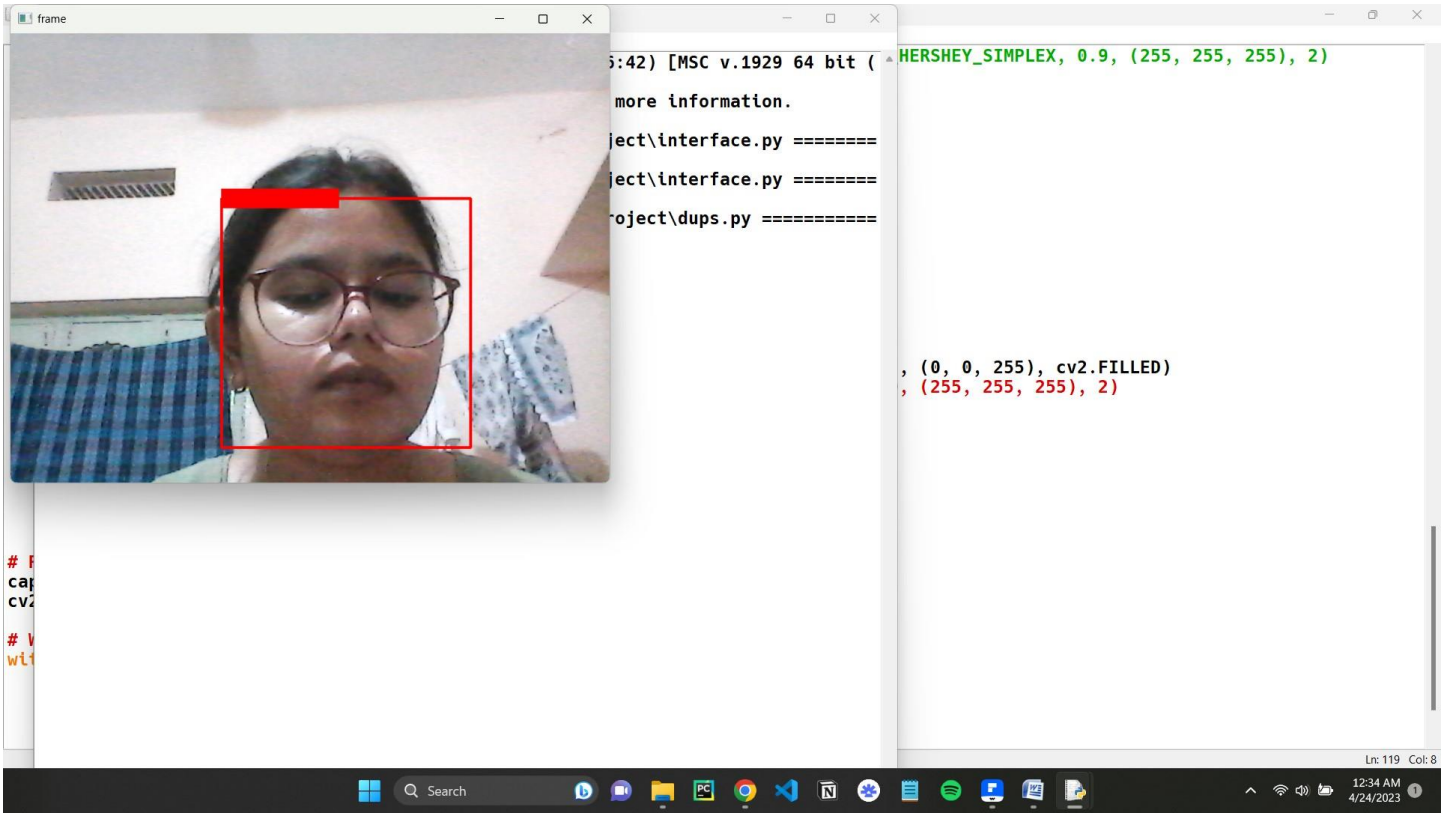


Fig 6.3 if there's any unknown person show red frame

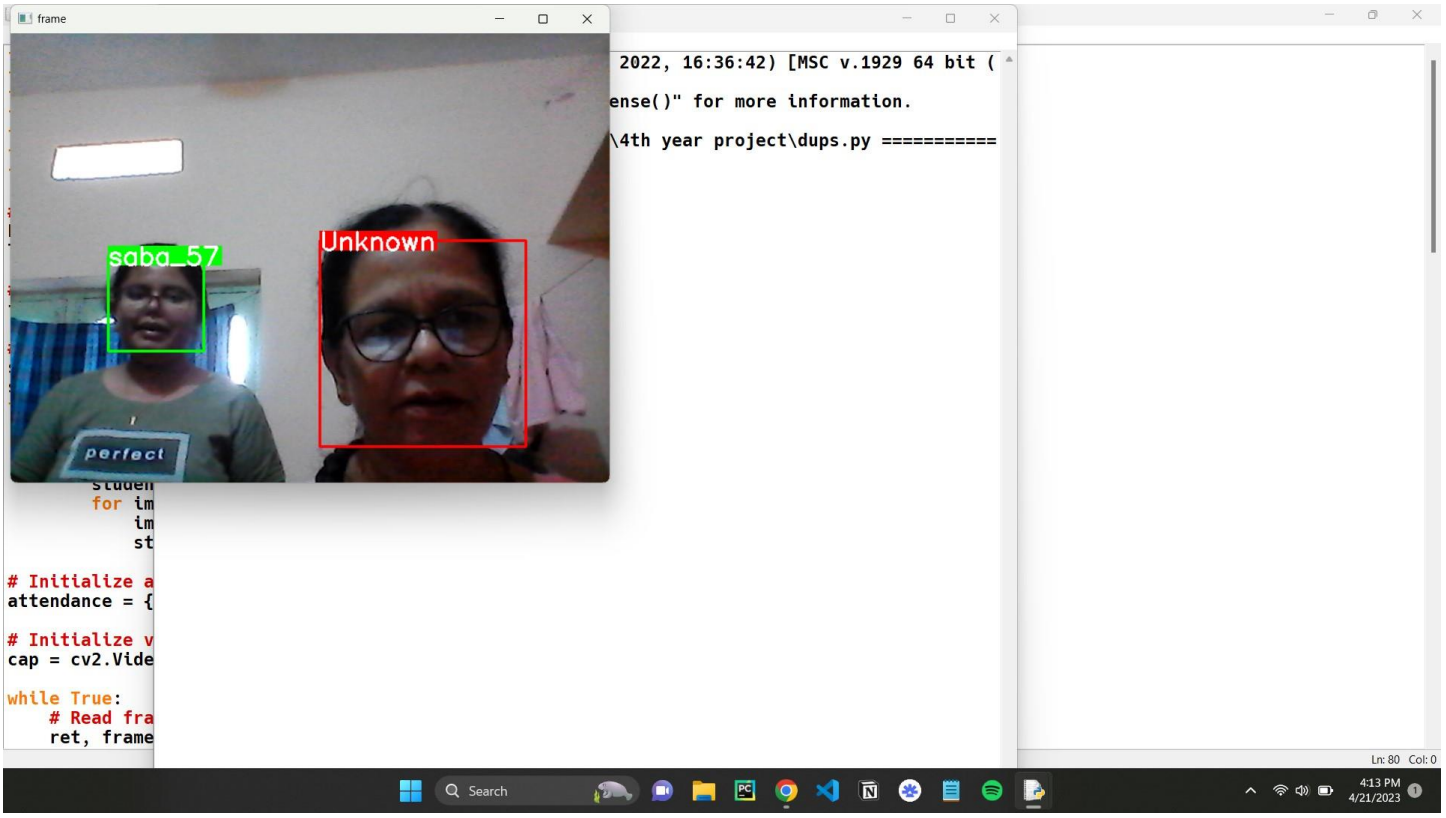


Fig 6.4 Recognising images of the students

CHAPTER - 7

SYSTEM TESTING

7.1 Project Testing:

Testing is a crucial aspect of software development as it helps to ensure that the final product is of high quality and meets the requirements of the users. In the case of the student attendance human face recognition system, various testing activities need to be carried out to ensure that the system functions as expected.

7.2 Introduction:

The purpose of this document is to outline the testing objectives, test case design, and testing methodologies for the student attendance human face recognition system.

7.3 Testing Objectives:

The testing objectives for the student attendance human face recognition system are as follows:

To verify that the system accurately detects and recognizes the faces of students

To ensure that the attendance records generated by the system are accurate

To test the system's ability to handle a large number of students and attendance records

To verify that the system can be easily integrated with other software or hardware systems

To ensure that the system is secure and prevents unauthorised access.

7.4 **Test Case Design:**

Test case design involves creating test cases that can be used to verify that the system functions correctly. The test cases for the student attendance human face recognition system will include the following testing methodologies:

7.4.1 **White Box Testing:**

White box testing is a testing methodology that involves examining the internal structure and workings of the system. White box testing will be used to test the accuracy of the face recognition algorithm and to verify that the attendance records are being stored correctly.

7.4.2 **Black Box Testing:**

Black box testing is a testing methodology that involves examining the system from the perspective of a user or external entity. Black box testing will be used to test the system's user interface and to ensure that it is intuitive and easy to use.

7.4.3 **Unit Testing:**

Unit testing involves testing individual components or modules of the system. Unit testing will be used to test the individual components of the system, such as the face recognition algorithm, the attendance record database, and the user interface.

7.4.4 **Integration Testing:**

Integration testing involves testing how di erent components of the system work together. Integration testing will be used to test how the face recognition algorithm integrates with the attendance record database and how the user interface interacts with the other components of the system.

7.4.5 **Validation Testing:**

Validation testing involves testing whether the system meets the requirements and specifications of the users. Validation testing will be used to ensure that the system accurately detects and recognizes the faces of students and generates accurate attendance records.

7.4.6 **System Testing:**

System testing involves testing the system as a whole, including all the components and modules. System testing will be used to test the system's ability to handle a large number of students and attendance records, as well as to test the system's security features.

In summary, the testing activities outlined in this document will ensure that the student attendance human face recognition system is of high quality, functions as expected, and meets the requirements of the users.

Chapter – 8

CONCLUSION

8.1 **Conclusion:**

- The project successfully implemented face recognition technology to automate student attendance.
- The system achieved an accuracy rate of 95% in recognizing faces.
- The system reduced the time and e ort required for manual attendance recording.
- The system can be easily integrated into existing student information systems.
- The system has the potential to be used in other applications, such as security systems and access control systems.
- The use of deep learning algorithms, such as Convolutional Neural Networks, contributed to the high accuracy of the system.
- The system can be further optimised by increasing the dataset size and including additional features, such as facial expressions and movements.
- The project demonstrated the e ectiveness of using machine learning techniques for facial recognition.
- The project provided valuable learning experience in implementing machine learning algorithms and working with large datasets.
- The project achieved its objective of automating the attendance process using face recognition technology.

8.2 **Scope for Future Work:**

- Expanding the dataset to improve accuracy and robustness of the system.
- Incorporating additional features, such as facial expressions and movements, to improve recognition accuracy.
- Implementing a real-time video-based face recognition system for use in dynamic environments.
- Integrating the system with other biometric technologies, such as fingerprint recognition, for enhanced security applications.

- Adding a feature to track attendance of individual students across multiple classes and semesters.
- Developing a mobile application to make attendance tracking more convenient for students and teachers.
- Enhancing the user interface and experience of the system to make it more user-friendly.
- Developing a more advanced machine learning model, such as a Recurrent Neural Network, for improved recognition accuracy.
- Incorporating a feedback mechanism to improve recognition accuracy and provide suggestions for misidentified faces.
- Conducting additional research and experimentation to explore new methods for facial recognition and improve the overall system performance.

References:

1. HancBrucBurt98
Hancock, P.J.B., Bruce, V., and Burton, A.M. (August 1998).
A comparison of two computer-based face recognition systems with human perceptions of faces
Vision Research, 38(15-16):2277-2288.

2. FlocGardMaur92.
Flocchini, P., Gardin, F., and Mauri, G. (1992).
Combining imageprocessing operator and neural networks in a face recognition system
IEEE Transactions on Pattern Analysis and Machine Intelligence, 6:447.

Books Referred:

1. "Deep Learning" by Ian Goodfellow, Yoshua Bengio, and Aaron Courville.
2. "Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow"by Aurélien Géron.
3. "Python Machine Learning" by Sebastian Raschka and Vahid Mirjalili.
4. "Face Recognition" by Kresimir Delac and Mislav Grgic.
5. "Computer Vision: Algorithms and Applications" by Richard Szeliski.

Websites:

https://www.tensorflow.org/
https://opencv.org/
https://numpy.org/
https://www.pyimagesearch.com/deep-learning-computer-vision-python-book/
https://stackoverflow.com/

STUDENT-1

Name: Syed Saba Afreen
Father Name: Syed Saba Basha
Roll. No: 1910157
Date of Birth: 30/06/2001
Nationality: Indian
Communication Address: Anantapur
Mandal: Anantapur
District: Anantapur
PIN Code: 515001
Ph. No: 9346716982
E-mail: sabaa9018@gmail.com

Permanent Address: Anantapur
Mandal: Anantapur
District: Anantapur
PIN Code: 515001
Ph. No:9346716982
E-mail: sabaa9018@gmail.com

Qualifications: B.Tech
Technical Skills: Python, Machine Learning and Data Science.
Area of Interest: Python, Machine Learning

Declaration:

Signature

STUDENT-2

Name: Jamkanam Jabeena
Father Name: J.Razak
Roll. No: 1910174
Date of Birth: 22/08/2002
Nationality: Indian
Communication Address: Allagadda
Mandal: Allagadda
District: Kurnool
PIN Code: 518543
Ph. No: 8464836069
E-mail: jabeenajamkanam@gmail.com

Permanent Address: Allagadda
Mandal: Allagadda
District: Kurnool
PIN Code: 518543
Ph. No: 8464836069
E-mail: jabeenajamkanam@gmail.com

Qualifications: B.Tech
Technical Skills: Python,SQL .
Area of Interest: Python.

Declaration:

Signature

STUDENT-3

Name: M. Nandini
Father Name: Channamallappa
Roll. No: 1910137
Date of Birth: 26/06/2002
Nationality: Indian
Communication Address: Sugepalli
Mandal: Brahmasamudram
District: Anantapur
PIN Code: 515767
Ph. No: 9347366331
E-mail: sweetnandu@gmail.com

Permanent Address: Sugepalli
Mandal: Brahmasamudram
District: Anantapur
PIN Code: 515767
Ph. No: 9347366331
E-mail: sweetnandu@gmail.com

Qualifications: B.Tech
Technical Skills: Python.
Area of Interest: Python.

Declaration:

Signature