A

Project Report

On

“Smart Attendance System using OPENCV based on Facial Recognition”

Submitted in partial fulfillment of the requirement of the University of Mumbai for the Degree of **Bachelor of Engineering**

(Computer Engineering)

By

Akash Pramod Raut

Under the guidance of

Prof. Mahesh Thakur

Department of Computer Engineering

Wilfred's Education Society
Chhatrapati Shivaji Maharaj Institute of Technology
Shedung, Panvel Dist: Raigad-410206

University of Mumbai

Academic Year 2021-22

Chhatrapati Shivaji Maharaj Institute of Technology

Department of Computer Engineering

Academic Year 2021-22

CERTIFICATE

This is to certify that

Mr. Akash Pramod Raut Sem.VIII, BE Computer, Roll No: 39 *has satisfactorily completed the requirements of the Project* entitled

"Smart Attendance System using OPENCV based on Facial Recognition"

As prescribed by the University of Mumbai Under the guidance of

Prof. Mahesh Thakur

Guide	**HOD**	**Principal**
(Prof. Mahesh Thakur)		**(Dr. Dharmendra Dubey)**

Internal Examiner

Prof._____

External Examiner

Prof._____

Project report Approval for B.E

The B.E Project Report Entitled

"Smart Attendance System using OPENCV based on Facial Recognition"

Submitted by

Mr. Akash Pramod Raut (39)

Is approved for the degree of Bachelor of Engineering in Computer Engineering.

Examiners:

1. _____
2. _____

Date: 25/04/2022

Place: Panvel

Declaration

I declare that this written submission represents my idea entitled “Smart Attendance System using OPENCV based on Facial Recognition” in my own words and where other ideas or words have been included. I have adequately quoted and referred to the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not mispresented or fabricated or falsified any idea/Data/Facts/Sources in my submission. I understand that any violation of the above will be caused by disciplinary action by the Institute and can also evoke penal action for the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Name and Signature

Mr. Akash Pramod Raut

_____

Date: 25/04/2022

Abstract

Face is the crucial part of the human body that uniquely identifies a person. Using the face characteristics as biometric, the face recognition system can be implemented. The most demanding task in any organization is attendance marking. In traditional attendance system, the students are called out by the teachers and their presence or absence is marked accordingly. However, these traditional techniques are time consuming and tedious. In this project, the Open CV based face recognition approach has been proposed. This model integrates a camera that captures an input image, an algorithm for detecting face from an input image, encoding and identifying the face, marking the attendance in a spreadsheet and converting it into PDF file. The training database is created by training the system with the faces of the authorized students. The cropped images are then stored as a database with respective labels. The features are extracted using LBPH algorithm.

Acknowledgment

This document is a review report on the research conducted and the project made in the field of computer engineering to develop a platform for attendance tracking automatically with the help of facial recognition and OPENCV based technology.

For the success of any project there needs hard work and dedication by every member of that group. But it largely depends on the support and encouragement given to the team members. We take this opportunity to express our gratitude to the people who have been leading and guiding us in the completion of this project.

We are greatly thankful to our project guide Lecturer **Prof. Mahesh Thakur** for their kind support and guidance involved in the successful completion of this project. We have highly benefited from his guidance and have found his suggestions helpful in various phases of this project.

We are highly grateful to Principal, **Mr. Dharmendra Dubey**, H.O.D of Computer Dept. Chhatrapati Shivaji Maharaj Institute of Technology, Panvel for providing all the necessary facilities and encouraging us during work.

We would also think the entire Teaching and Non-Teaching staff of the Computer Department for their constant assistance and co-operation.

Akash Pramod Raut

Table of Content

Table of Figure

CHAPTER 1

INTRODUCTION

1.1 Background

Attendance maintenance is a significant function in all the institutions to monitor the performance of the students. Every institute does this in its own way. Some of these institutes use the old paper or file-based systems and some have adopted strategies of automatic attendance using some biometric techniques. A facial recognition system is a computerized biometric software which is suited for determining or validating a person by performing comparison on patterns based on their facial appearances. Face recognition systems have upgraded appreciably in their management over the recent years and this technology is now vastly used for various objectives like security and in commercial operations. Face recognition is a powerful field of research which is a computer based digital technology. Face recognition for the intent of marking attendance is a resourceful application of attendance system. It is widely used in security systems and it can be compared with other biometrics such as fingerprint or eye iris recognition systems. As the number of students in an educational institute or employees at an organization increases, the needs for lecturers or to the organization also increase the complication of attendance control. This project may be helpful for the explanation of these types of problems.

The technology aims in imparting a tremendous knowledge oriented technical innovation these days. Deep Learning is one among the interesting domain that enables the machine to train itself by providing some datasets as input and provides an appropriate output during testing by applying different learning algorithms. Nowadays Attendance is considered as an important factor for both the student as well as the teacher of an educational organization. With the advancement of the deep learning technology the machine automatically detects the attendance performance of the students and maintains a record of those collected data.

1.2 Relevance

With advances in computing and telecommunications technologies, digital images are playing key roles in the present information era. Human face is an important biometric object in image databases of surveillance systems. Detecting and locating human faces and facial features in an image or image sequence are important tasks in dynamic environments, such as videos, where noise conditions, illuminations, locations of subjects and pose can vary significantly from frame to frame. An automated system for human face recognition in real time background for a college to mark the attendance of their employees and students. So Smart Attendance using Real Time Face Recognition is a real-world solution which comes with day to day activities of handling employees. Here multiple user faces are detected and recognised with the data base trained multiple texture-based features.

Face recognition being a biometric technique implies determination if the image of the face of any particular person matches any of the face images that are stored in a database. This difficulty is tough to resolve automatically because of the changes that several factors, like facial expression, aging and even lighting can affect the image. Facial recognition among the various biometric techniques may not be the most authentic but it has various advantages over the others. Face recognition is natural, feasible and does not require assistance. The expected system engages the face recognition approach for the automating the attendance procedure of students or employees without their involvement. A web cam is used for capturing the images of students or employees. The faces in the captured images are detected and compared with the images in database and the attendance is marked.

1.3 Organization of Project Report.

Chapter-1 is about introduction which gives an idea about of our project domain i.e. Attendance tracking using Facial recognition.

Chapter 2 is about literature survey concerning this project. Here a brief look into all previous methods and existing models are examined.

Chapter 3 contains information about system requirements.

Chapter 4 contains methodology. In this project, we used different algorithms to enhance the facial recognition using OPENCV and many technologies.

Chapter 5 consists experimental analysis and results in this sample code, testing results, system configurations such as software and hardware requirements, input and output images are displayed.

Chapter 6 contains information about implementation of the system, its system architecture various working states and models.

Chapter 7 contains information about result analysis

Chapter-8 explains conclusion and future work about our project. Various ways of extending this project is explained here.

CHAPTER 2

LITERATURE SURVEY

The 'Literature Review' is the part of the dissertation where there is extensive reference to related research and theory in the field. Overall, the function of a literature review is to show how related work in the field has shaped and influenced your research. You should aim to use the literature selectively and creatively to provide a stimulus for your work. A literature survey or a literature review in a project report is that section that shows the various analyses and research made in the field of your interest and the results already published, taking into account the various parameters of the project and the extent of the project.

It is the most important part of your report as it gives you a direction in the area of your research. It helps you set a goal for your analysis - thus giving you your problem statement. A literature review is a text of a scholarly paper, which includes the current knowledge including substantive findings, as well as theoretical and methodological contributions to a particular topic. Literature reviews are secondary sources and do not report new or original experimental work.

Most often associated with academic-oriented literature, such reviews are found in academic journals and are not to be confused with book reviews that may also appear in the same publication. Literature reviews are a basis for research in nearly every academic field. [unreliable source] A narrow-scope literature review may be included as part of a peer-reviewed journal article presenting new research, serving to situate the current study within the body of the relevant literature and to provide context for the reader. In such a case, the review usually precedes the methodology and results in sections of the work. Producing a literature review may also be part of graduate and post-graduate student work, including in the preparation of a thesis, dissertation, or journal article.

Literature reviews are also common in a research proposal or prospectus (the document that is approved before a student formally begins a dissertation or thesis).

Automatic face recognition (AFR) technologies have made many improvements in the changing world. Smart Attendance using Real-Time Face Recognition is a real-world solution which comes with day to day activities of handling student attendance system. Face recognition-based attendance system is a process of recognizing the students face for taking attendance by using face biometrics based on high - definition monitor video and other information technology. In my face recognition project, a computer system will be able to find and recognize human faces fast and precisely in images or videos that are being captured through a surveillance camera. Numerous algorithms and techniques have been developed for improving the performance of face recognition but the concept to be implemented here is Deep Learning. It helps in conversion of the frames of the video into images so that the face of the student can be easily recognized for their attendance so that the attendance database can be easily reflected automatically.

The main purpose of this project is to build a face recognition-based attendance monitoring system for educational institution to enhance and upgrade the current attendance system into more efficient and effective as compared to before. The current old system has a lot of ambiguity that caused inaccurate and inefficient of attendance taking. Many problems arise when the authority is unable to enforce the regulation that exist in the old system. The technology working behind will be the face recognition system. The human face is one of the natural traits that can uniquely identify an individual. Therefore, it is used to trace identity as the possibilities for a face to deviate or being duplicated is low. In this project, face databases will be created to pump data into the recognizer algorithm. Then, during the attendance taking session, faces will be compared against the database to seek for identity. When an individual is identified, its attendance will be taken down automatically saving necessary information into a excel sheet. At the end of the day, the excel sheet containing attendance information regarding all individuals are mailed to the respective faculty.

2.1 Existing System

1. Fingerprint Based recognition system: In the Fingerprint based existing attendance system, a portable fingerprint device needs to be configured with the students fingerprint earlier. Later either during the lecture hours or before, the student needs to record the fingerprint on the configured device to ensure their attendance for the day. The problem with this approach is that during the lecture time it may distract the attention of the students. Fingerprint recognition refers to the automated method of identifying or confirming the identity of an individual based on the comparison of two fingerprints. Fingerprint recognition is one of the most well-known biometrics, and it is by far the most used biometric solution for authentication on computerized systems. The reasons for fingerprint recognition being so popular are the ease of acquisition, established use and acceptance when compared to other biometrics, and the fact that there are numerous (ten) sources of this biometric on everyone.

2. RFID (Radio Frequency Identification) Based recognition system: Radio Frequency Identification (RFID) refers to a wireless system comprised of two components: tags and readers. The reader is a device that has one or more antennas that emit radio waves and receive signals back from the RFID tag. Tags, which use radio waves to communicate their identity and other information to nearby readers, can be passive or active. Passive RFID tags are powered by the reader and do not have a battery. Active RFID tags are powered by batteries. In the RFID based existing system, the student needs to carry a Radio Frequency Identity Card with them and place the ID on the card reader to record their presence for the day. The system is capable of to connect to RS232 and record the attendance to the saved database. There are possibilities for the fraudulent access may occur. Some are students may make use of other student's ID to ensure their presence when the particular student is absent or they even try to misuse it sometimes. RFID tags can store a range of information from one serial number to several pages of data. Readers can be mobile so that they can be carried by hand, or they can be mounted on a post or overhead. Reader systems can also be built into the architecture of a cabinet, room, or building.

3. Iris Based Recognition System: Iris recognition or iris scanning is the process of using visible and near-infrared light to take a high-contrast photograph of a person's iris. It is a form of biometric technology in the same category as face recognition and fingerprinting. Advocates of iris scanning technology claim it allows law enforcement officers to compare iris images of suspects with an existing database of images to determine or confirm the subject's identity.

They also state that iris scans are quicker and more reliable than fingerprint scans since it is easier for an individual to obscure or alter their fingers than it is to alter their eyes. Iris scanning raises significant civil liberties and privacy concerns. It may be possible to scan irises from a distance or even on the move, which means that data could be collected surreptitiously, without individuals' knowledge, let alone consent. There are security concerns as well: if a database of biometric information is stolen or compromised, it is not possible to get a new set of eyes like one would get a reissued credit card number. And iris biometrics are often collected and stored by third-party vendors, which greatly expands this security problem. In the Iris based student attendance system, the student needs to stand in front of a camera, so that the camera will scan the Iris of the student. This reduces the paper and pen workload of the faculty member of the institute. This also reduces the chances of proxies in the class, and helps in maintaining the student records safe. It is a wireless biometric technique that solves the problem of spurious attendance and the trouble of laying the corresponding network.

2.2 Problem Statement

According to the previous attendance management system, the accuracy of the data collected is the biggest issue. This is because the attendance might not be recorded personally by the original person, in another word, the attendance of a particular person can be taken by a third party without the realization of the institution which violates the accuracy of the data. For example, student A is lazy to attend a particular class, so student B helped him/her to sign for the attendance which in fact student A didn't attend the class, but the system overlooked this matter due to no enforcement practiced. Supposing the institution establish an enforcement, it might need to waste a lot of human resource and time which in turn will not be practical at all. Thus, all the recorded attendance in the previous system is not reliable for analysis usage. The second problem of the previous system is where it is too time consuming. Assuming the time taken for a student to sign his/her attendance on a 3-4 paged name list is approximately 1 minute. In 1 hour, only approximately 60 students can sign their attendance which is obviously inefficient and time consuming. The third issue is with the accessibility of those information by the legitimate concerned party. For an example, most of the parents are very concerned to track their child's actual whereabouts to ensure their kid really attend the classes in college/school. However in the previous system, there are no ways for the parents to access such information. Therefore, evolution is needed to be done to the previous system to improve efficiency, data accuracy and provides accessibility to the information for those legitimate party.

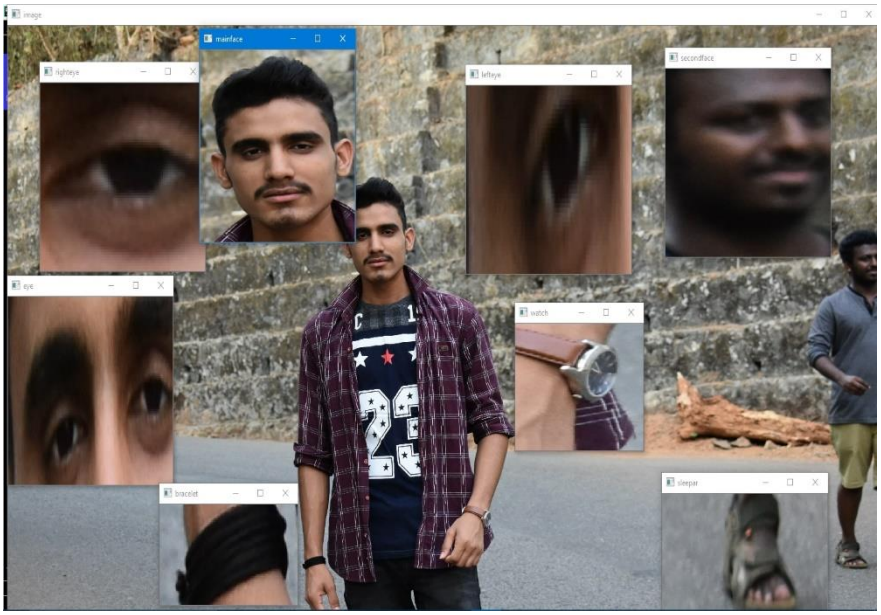
CHAPTER 3

REQUIREMENT GATHERING

3.1 Software and Hardware Requirements

OpenCV: OpenCV is the huge open-source library for the computer vision, machine learning, and image processing and now it plays a major role in real-time operation which is very important in today's systems. By using it, one can process images and videos to identify objects, faces, or even handwriting of a human. When it integrated with various libraries, such as NumPy, python is capable of processing the OpenCV array structure for analysis. To Identify image pattern and its various features we use vector space and perform mathematical operations on these features.

The first OpenCV version was 1.0. OpenCV is released under a BSD license and hence it's free for both academic and commercial use. It has C++, C, Python and Java interfaces and supports Windows, Linux, Mac OS, iOS and Android. When OpenCV was designed the main focus was real-time applications for computational efficiency. All things are written in optimized C/C++ to take advantage of multi-core processing.



From the above original image, lots of pieces of information that are present in the original image can be obtained. Like in the above image there are two faces available and the person(I) in the images wearing a bracelet, watch, etc so by the help of OpenCV we can get all these types of information from the original image.

It's the basic introduction to OpenCV we can continue the Applications and all the things in our upcoming articles.

Applications of OpenCV: There are lots of applications which are solved using OpenCV, some of them are listed below

- face recognition
- Automated inspection and surveillance
- number of people – count (foot traffic in a mall, etc)
- Vehicle counting on highways along with their speeds
- Interactive art installations
- Anamoly (defect) detection in the manufacturing process (the odd defective products)
- Street view image stitching
- Video/image search and retrieval
- Robot and driver-less car navigation and control

- object recognition
- Medical image analysis
- Movies – 3D structure from motion
- TV Channels advertisement recognition

OpenCV Functionality:-

- Image/video I/O, processing, display (core, imgproc, highgui)
- Object/feature detection (objdetect, features2d, nonfree)
- Geometry-based monocular or stereo computer vision (calib3d, stitching, videostab)
- Computational photography (photo, video, superres)
- Machine learning & clustering (ml, flann)
- CUDA acceleration (gpu)

NumPy:-NumPy, which stands for Numerical Python, is a library consisting of multidimensional array objects and a collection of routines for processing those arrays. NumPy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays. It is the fundamental package for scientific computing with Python. It contains various features including these important ones:

- A powerful N-dimensional array object
- Sophisticated (broadcasting) functions
- Tools for integrating C/C++ and Fortran code
- Useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined using Numpy which allows NumPy to seamlessly and speedily integrate with a wide variety of databases. Using NumPy, mathematical and logical operations on arrays can be performed. This tutorial explains the basics of NumPy such as its architecture and environment. It also discusses the various array functions, types of indexing, etc. An introduction to Matplotlib is also provided. All this is explained with the help of examples for better understanding.

SQLite:-

SQLite is an in-process library that implements a self-contained, serverless, zero-configuration, transactional SQL database engine. It is a database, which is zero-configured, which means like other databases you do not need to configure it in your system.

SQLite engine is not a standalone process like other databases, you can link it statically or dynamically as per your requirement with your application. SQLite accesses its storage files directly.

Below are the features listed for SQLite

- SQLite does not require a separate server process or system to operate (serverless).
- SQLite comes with zero-configuration, which means no setup or administration needed.
- A complete SQLite database is stored in a single cross-platform disk file.
- SQLite is very small and light weight, less than 400KiB fully configured or less than 250KiB with optional features omitted.
- SQLite is self-contained, which means no external dependencies.
- SQLite transactions are fully ACID-compliant, allowing safe access from multiple processes or threads.
- SQLite supports most of the query language features found in SQL92 (SQL2) standard.
- SQLite is written in ANSI-C and provides simple and easy-to-use API.
- SQLite is available on UNIX (Linux, Mac OS-X, Android, iOS) and Windows (Win32, WinCE, WinRT).

PYQT5:-

PyQt5 is a set of Python bindings for Qt5 application framework from Digia. Qt library is one of the most powerful GUI libraries. The official home site for PyQt5 is www.riverbankcomputing.co.uk/news. PyQt5 is developed by Riverbank Computing. PyQt5 is implemented as a set of Python modules. It has over 620 classes and 6000 functions and methods. It is a multiplatform toolkit which runs on all major operating systems, including Unix, Windows, and Mac OS. PyQt5 is dual licensed. Developers can choose between a GPL and a commercial license. PyQt5 is cross-platform GUI toolkit, a set of python bindings for Qt v5. One can develop an interactive desktop application with so much ease because of the tools and simplicity provided by this library. A GUI application consists of Front-end and Back-end. PyQt5 has provided a tool called 'QtDesigner' to design the front-end by drag and drop method so that development can become faster and one can give more time on back-end stuff.

PyQt brings together the Qt C++ cross-platform application framework and the cross-platform interpreted language Python. Qt is more than a GUI toolkit. It includes abstractions of network sockets, threads, Unicode, regular expressions, SQL databases, SVG, OpenGL, XML, a fully functional web browser, a help system, a multimedia framework, as well as a rich collection of GUI widgets.

Qt classes employ a signal/slot mechanism for communicating between objects that is type safe but loosely coupled making it easy to create re-usable software components. Qt also includes Qt Designer, a graphical user interface designer. PyQt is able to generate Python code from Qt Designer. It is also possible to add new GUI controls written in Python to Qt Designer. Python is a simple but powerful object-orientated language. Its simplicity makes it easy to learn, but its power means that large and complex applications can be created. Its interpreted nature means that Python programmers are very productive because there is no edit/compile/link/run development cycle. Much of Python's power comes from its comprehensive set of extension modules providing a wide variety of functions including HTTP servers, XML parsers, database access, data compression tools and, of course, graphical user interfaces. Extension modules are usually implemented in either Python, C or C++. Using tools such as SIP it is relatively straight forward to create an extension module that encapsulates an existing C or C++ library. Used in this way, Python can then become the glue to create new applications from established libraries.

CHAPTER 4

PLAN OF THE PROJECT

4.1 Description of the Project

Attendance using face recognition system is developed for marking the attendance by detecting the face of the user from live camera where the user will capture the photo. We will be using OpenCV for detection of face, in OpenCV we will be using “haar cascade frontal face” xml file.

Haar Cascade is an Object Detection Algorithm used to identify faces in an image or a real time video. The algorithm uses edge or line detection features proposed by Viola and Jones in their research paper “Rapid Object Detection using a Boosted Cascade of Simple Features” published in 2001. The algorithm is given a lot of positive images consisting of faces, and a lot of negative images not consisting of any face to train on them.

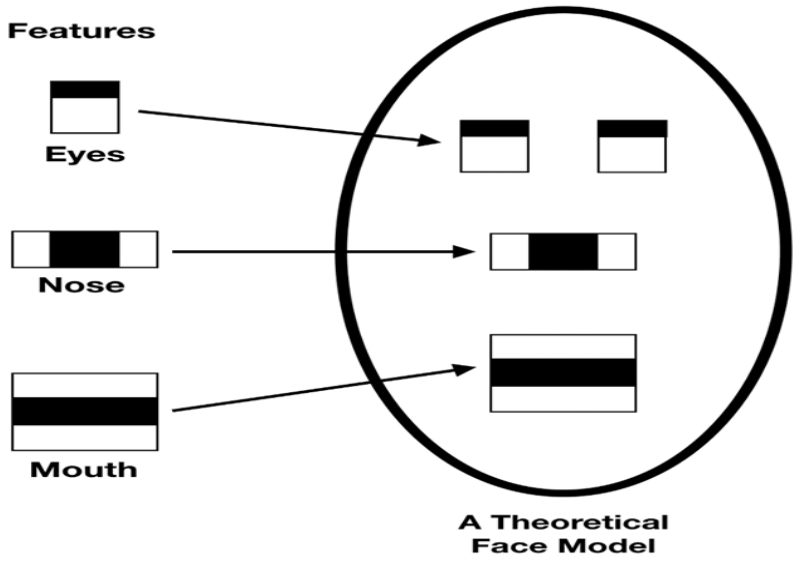


Fig. 4.1: Haar Cascade

LBPH algorithm will be used for recognizing the face. Firstly, we will train our model with data which will have 40-50 different image of same user. After the training the model will be stored into the YML file. The YML file will contain all the data for all the training set. The YML stands for "YAML Markup Language" that stores the face pixel values. We will be matching our image data which is captured from camera with the model which we have trained.

We can only store the two type of data for particular user i.e. The user roll number and Face features which is extracted while training the model. The database which we are using is the SQLite3. SQLite is a C library that provides a lightweight disk-based database that doesn't require a separate server process and allows accessing the database using a nonstandard variant of the SQL query language. Some applications can use SQLite for internal data storage. It's also possible to prototype an application using SQLite and then port the code to a larger database such as PostgreSQL or Oracle.

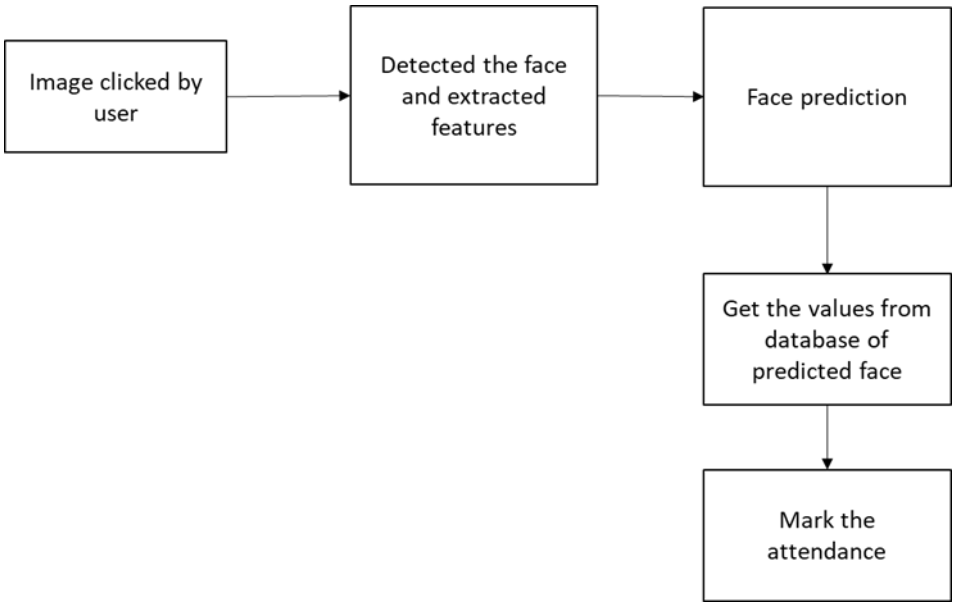


Fig. 4.2: Diagram showing the steps of process

For showing the result we are using the PYQT5 library for creating the UI of application which will capture the image and mark the attendance of the user. PyQt is a Python library for creating GUI applications using the Qt toolkit. Created by Riverbank Computing, PyQt is free software (GPL licensed) and has been in development since 1999. PyQt5 was released in 2016 and last updated in October 2021. PyQt5 is the Qt5-based edition of the Python GUI library PyQt from Riverbank Computing.

4.2 Proposed System Architecture

The task of the proposed system is to capture the face of user and to store it in the database for their attendance. The face of the user needs to be captured in such a manner that all the feature of the user's face needs to be detected. There is no need for the user to use the old biometric or any other system to mark the attendance. This system is developed using python OpenCV.

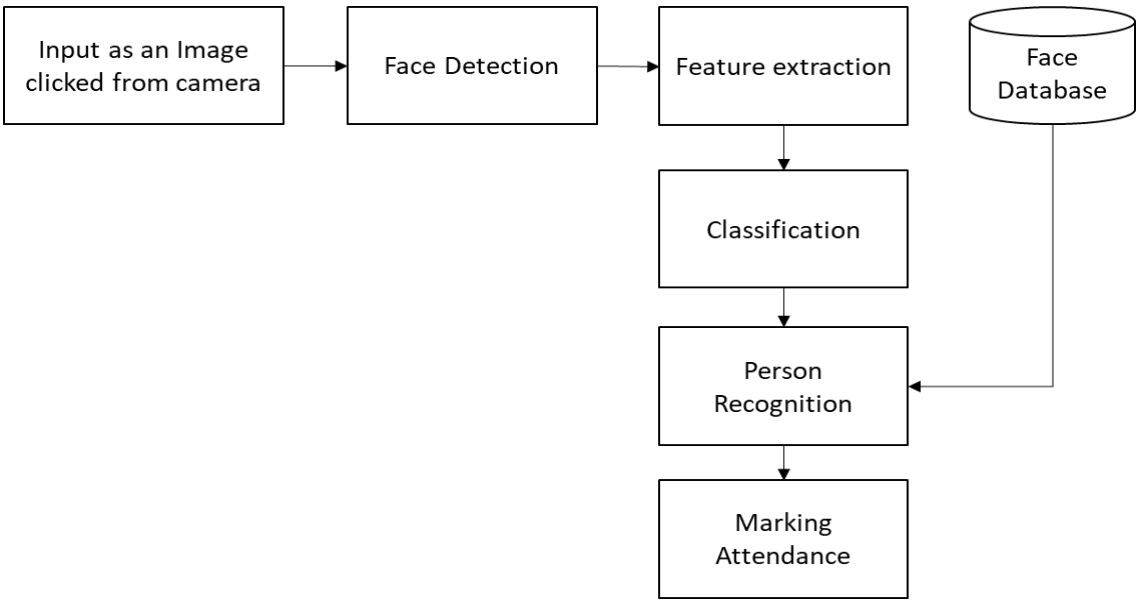


Fig. 4.3: System Architecture

The face recognition user attendance system emphasizes its simplicity by removing classical attendance marking technique such as calling the user name or checking their respective ID Cards. Thus, attendance system through facial recognition is proposed in order to replace the manual marking of user's attendance. The main task of our proposed system is to detect and recognize the image of the user and mark the attendance accordingly in the database. Also can capture new entries if needed.

The proposed framework uses OpenCV library. It is an Open-Source Computer Vision Library that is free for both scholastic and business use. It has python, and PyQt interfaces and supports various platforms like Windows, Linux, MacOS. It has a strong focus on real time application. The library has more than 2500 improved algorithms, these algorithms can be utilized to detect and recognize faces, objects, and so forth OpenCV has a FaceRecognizer class library for face recognition. This recognizes and controls faces from Python or from the command line. It is a

basic library constructed using dlib's cutting edge face recognition built with deep learning. The Dlib is a cross-stage open-source software library that is executed on various computing platform. The model has a precision of 99.38%. This provides a basic face recognition tool that allows you to perform face recognition on folder of pictures from the command line.

The task of the proposed system is to capture the face of each student and to store it in the database for their attendance. The face of the student needs to be captured in such a manner that all the feature of the students' face needs to be detected. There is no need for the teacher to manually take attendance in the class because the system records a video and through further processing steps the face is being recognized and the attendance database is updated.

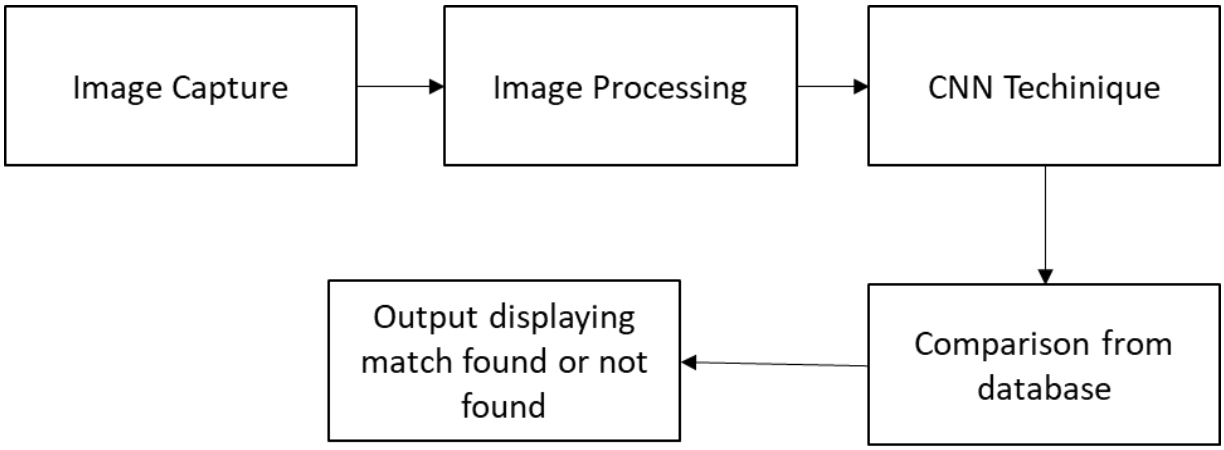


Fig. 4.4: Flow chart of proposed system

Image Capture: We need some good camera in order to get results.

Image Processing: Digital image processing is the use of a digital computer to process digital images through an algorithm. As a subcategory or field of digital signal processing, digital image processing has many advantages over analog image processing. It allows a much wider range of algorithms to be applied to the input data and can avoid problems such as the build-up of noise and distortion during processing. Since images are defined over two dimensions (perhaps more) digital image processing may be modeled in the form of multidimensional systems. The generation and development of digital image processing are mainly affected by three factors: first, the development of computers; second, the development of mathematics (especially the creation and improvement of discrete mathematics theory); third, the demand for a wide range of applications in environment, agriculture, military, industry and medical science has increased.

Convolution Neural Network: In deep learning, a convolutional neural network (CNN, or ConvNet) is a class of deep neural networks, most commonly applied to analyzing visual imagery. They are also known as shift invariant or space invariant artificial neural networks (SIANN), based on their shared-weights architecture and translation invariance characteristics. They have applications in image and video recognition, recommender systems, image classification, medical image analysis, natural language processing, and financial time series. CNNs are regularized versions of multilayer perceptron's. Multilayer perceptron's usually mean fully connected networks, that is, each neuron in one layer is connected to all neurons in the next layer. The "fully-connectedness" of these networks makes them prone to overfitting data. Typical ways of regularization include adding some form of magnitude measurement of weights to the loss function. CNNs take a different approach towards regularization: they take advantage of the hierarchical pattern in data and assemble more complex patterns using smaller and simpler patterns. Therefore, on the scale of connectedness and complexity, CNNs are on the lower extreme. Convolutional networks were inspired by biological processes in that the connectivity pattern between neurons resembles the organization of the animal visual cortex.

4.3 Flow Diagram

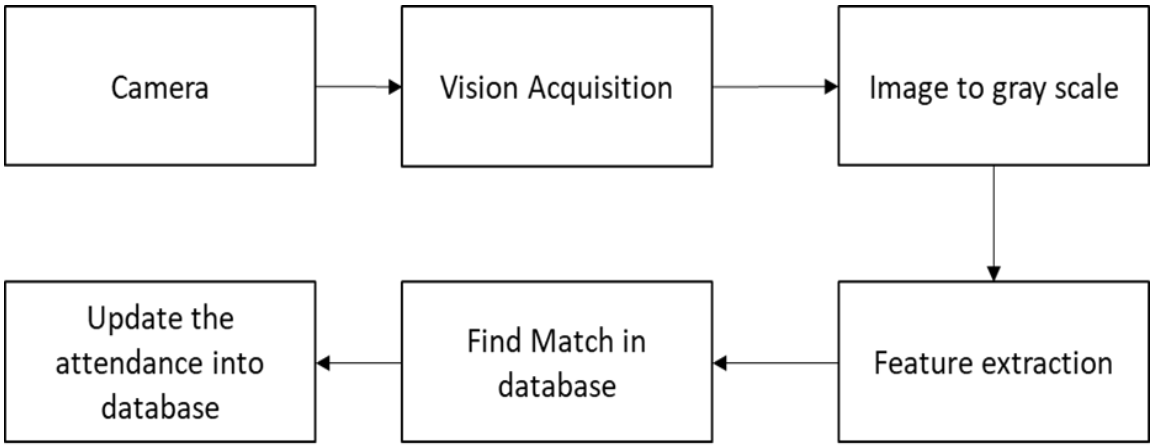


Fig. 4.5: Flow Diagram

The subsystem description is as follows:

Camera: The camera is the only hardware component required to capture live video feed of class.

Vision Acquisition: This module allows image to be captured by camera into LabVIEW for programming. It includes IMAQ submodules such as IMAQ Create, IMAQdx Open, IMAQdx Grab. They all combine to provide Continuous Acquisition of video feed from camera module.

Image to Grayscale: This process is performed using IMAQ ExtractSingleColorPlane VI to convert a 32/16bit image to 8bit image. This is a requirement for our pattern matching algorithm to work completely.

Pattern Extraction: This is included in Vision Assistant VI which deals with our face recognition algorithm. Pattern Extraction is feature in which the image inputted features are compared using Pattern Matching Algorithm.

Feature Extraction: This feature is used to extract important features out of image. It compares them with templates, saves in database and provides a score of comparison.

Find Match in database: Our database has preserved templates or images of students which we aim to recognize and mark attendance. This database can be updated or appended according to requirement. This database is used for comparison with extracted feature of image to confirm a successful hit.

Update Attendance to DB: If match is found our algorithm updates the attendance of user corresponding to his/her name in excel file of format .xlsx.

4.4 Algorithms and Methods

There are various algorithms used for facial recognition. Some of them are as follows:

1. Eigen faces
2. Fisher faces
3. Local binary patterns histogram

1. **Eigen faces: -** This method is a statistical plan. The characteristic which influences the images is derived by this algorithm. The whole Recognition method will depend on the training database that will be provided. The images from two different classes are Not treated individually.

2. **Fisher faces: -** Fisher faces algorithm also follows a progressive approach Just like the eigen faces. This method is a alteration of eigen Faces so it uses the same principal

components analysis. The Major conversion is that the fisher faces considers the classes. As mentioned previously, the eigen faces does not differentiate between the two pictures from two differed Classes while training. The total average affects each picture. A fisher face employs linear discriminant analysis for Distinguishing between pictures from a different class.

3. **Local binary patterns histograms: -** This method needs the gray scale pictures for dealing with the Training part. This algorithm in comparison to other Algorithms is not a holistic approach.

A. Parameters:

Lbph uses the following parameters:

I. Radius:
Generally, 1 is set as a radius for the circular local binary Pattern which denotes the radius around the central pixel.

II. Neighbours:
The number of sample points surrounding the central pixel Which is generally 8.the computational cost will increase with Increase in number of sample points.

III. Grid x:
The number of cells along the horizontal direction is Represented as grid x. With the increase in number of cells the Grid becomes finer which results in increase of dimensional Feature vector.

IV. Grid y:
The number of cells along the vertical direction is represented As grid y. With the increase in number of cells the grid Becomes finer which results in increase of dimensional feature Vector.

B. ALGORITHM TRAINING:

For the training purpose of the dataset of the facial images of the people to be recognized along with the unique ID is required so that the presented approach will utilize the provided information for perceiving an input image and providing the output. Same images require same ID.

C. COMPUTATION OF THE ALGORITHM:

The intermediate image with improved facial characteristics which corresponds to the original image is created in the first step. Based on the parameters provided, sliding window theory is used in order to achieve so. Facial image is converted into gray scale. A 3x3 pixels window is taken which can also be expressed as a 3x3 matrix which contains the intensity of each pixel (0-255). After this we consider the central value of the matrix which we take as the threshold. This value defines the new values obtained from the 8 neighbours. A new binary value is set for each neighbour of the central value. For the values equal to or greater than the threshold value 1 will be the output otherwise 0 will be the output. Only binary values will be present in the matrix and the concatenation is performed at each position to get new values at each position. Then the conversion of this binary value into a decimal value is done which is made the central value of the matrix. It is a pixel of the actual image. As the process is completed, we get a new image which serves as the better characteristics of the original image.

D. EXTRACTION OF HISTOGRAM:

The image obtained in the previous step uses the Grid X and Grid Y parameters and the image is split into multiple grids. Based on the image the histogram can be extracted as below:

1. The image is in gray scale and each histogram will consist of only 256 positions (0-255) which symbolises the existences of each pixel intensity.

2. After this each histogram is created and a new and bigger histogram is done. Let us suppose that there are 8x8 grids, then there will be 16.384 positions in total in the final histogram. Ultimately the histogram signifies the features of the actual image.

E. THE FACE RECOGNITION:

The training of the algorithm is done. For finding the image which is same as the input image, the two histograms are compared and the image corresponding to the nearest histogram is returned. Different approaches are used for the calculation of distance between the two histograms. Here we use the Euclidean distance based on the formula:

$$D = \sqrt{\sum_{i=1}^{n} (hist1_i - hist2_i)^2}$$

The facial recognition process can be split into three major stages: processing which occurs before detection involving face detection and alignment and later recognition is done using feature extraction and matching steps.

Face Detection:-

Face detection here is performed using Haar-Cascade Classifier with OpenCV. Haar Cascade algorithm needs to be trained to detect human faces before it can be used for face detection. This is called feature extraction. The haar cascade training data used is an xml file- haarcascade_frontalface_default. The haar features shown in Fig.2. will be used for feature extraction.

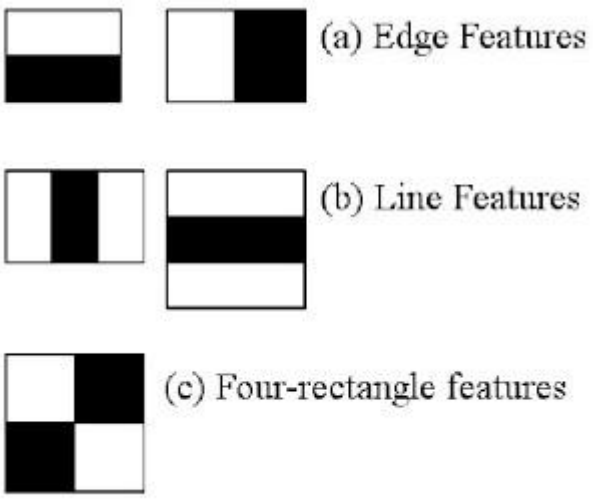


Fig. 4.6: Haar-Cascade Classifier

The primary function of this step is to conclude whether the human faces emerge in a given image, and what is the location of these faces. The expected outputs of this step are patches which contain each face in the input image. In order to get a more robust and easily designable face recognition system.

Here we are using detectMultiScale module from OpenCV. This is required to create a rectangle around the faces in an image. It has got three parameters to consider- scaleFactor, minNeighbors, minSize. minNeighbors specifies how many neighbors each candidate rectangle must have. Higher values usually detects less faces but detects high quality in image. minSize specifies the minimum object size. By default it is (30,30) [8]. The parameters used in this system is scaleFactor and minNeighbors with the values 1.3 and 5 respectively.

Feature extraction: -

Face recognition based on the geometric features of a face is probably the most intuitive approach to face recognition, marker points (position of eyes, ears, nose, ...) were used to build a feature vector (distance between the points, angle between them, ...). The recognition was performed by calculating the Euclidean distance between feature vectors of a probe and reference image. Such a method is robust against changes in illumination by its nature. A 22-dimensional feature vector was used and experiments on large datasets have shown, that geometrical features alone may not carry enough information for face recognition.

Face Recognition: -

Face recognition process can be divided into three steps- prepare training data, train face recognizer, prediction. Here training data will be the images present in the dataset. They will be assigned with a integer label of the student it belongs to. These images are then used for face recognition. Face recognizer used in this system is Local Binary Pattern Histogram. Initially, the list of local binary patterns (LBP) of entire face is obtained. At the end, one histogram will be formed for each image in the training data. Later, during recognition process histogram of the face to be recognized is calculated and then compared with the already computed histograms and returns the best matched label associated with the student. The last step after the representation of faces is to identify them. For automatic recognition we need to build a face database. Various images are taken foe each person and their features are extracted and stored in the database. Then when an input image is fed the face detection and feature extraction is performed and its feature to each face class is compared and stored in the database.

CHAPTER 5

PROJECT DESIGN & ANALYSIS

5.1 Use Case Diagram

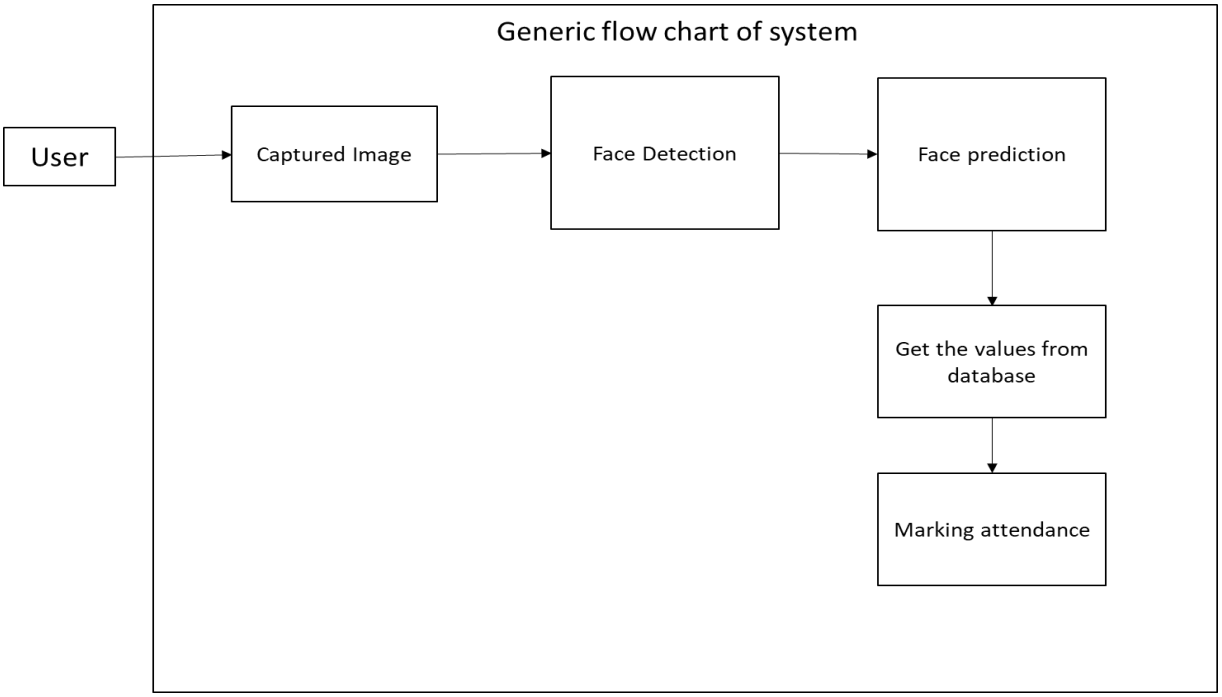


Fig. 5.1: Use case diagram of user

The interaction between the user and the software application is shown diagrammatically in Figure. The case diagram shows the set of actions that were performed by the system and how it interacts with user's of the system. It displays the different purposes of the system and also relates what the system is capable of and lays the rules of interaction for the required service. It can be referred to as the blueprint of the system.

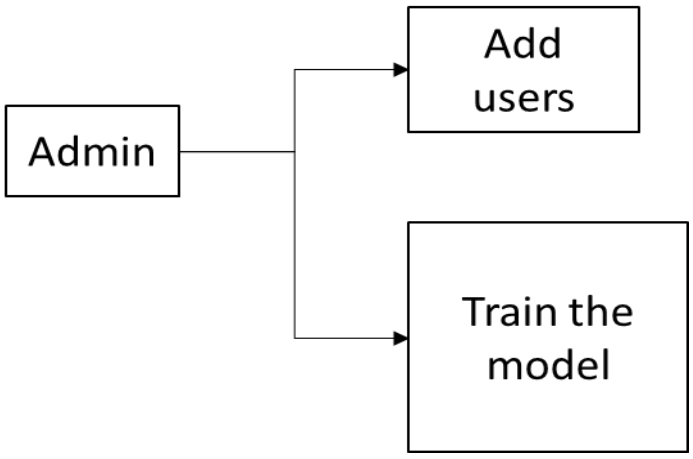


Fig. 5.2: Admin use case diagram

Above figure represent the admin actions. Admin can add the users and also admin can train the model for a particular user by providing the required information into the GUI. After filling the form Admin can submit the form and the model will get automatically trained.

5.2 Use Case Analysis – Sequence Diagrams / Activity Diagram

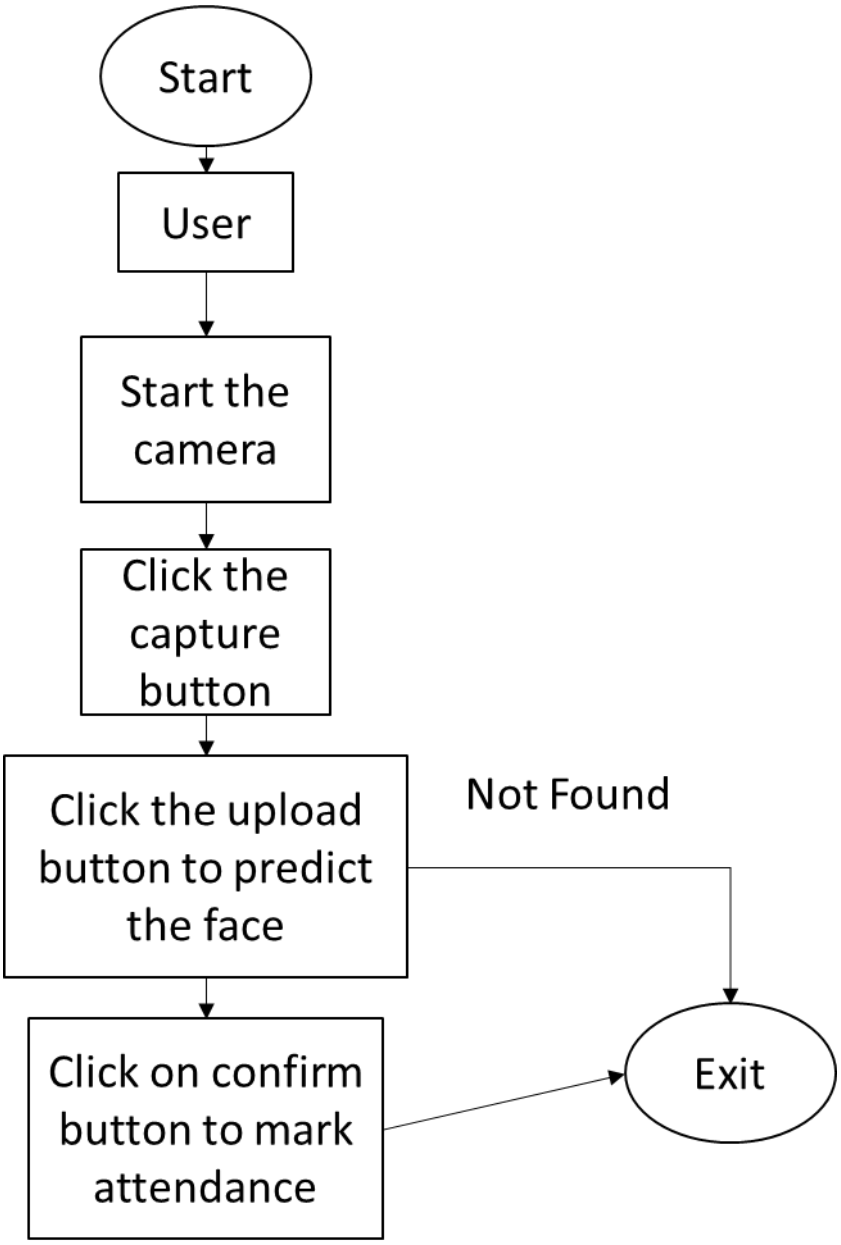


Fig. 5.3: Activity Diagram of User

The above figure shows how the user will mark the attendance, the explanation is mentioned below:

1. **Start the Camera:** The user will click on the button to start the camera. We are initializing camera using OpenCV library by default the laptop camera is used to capture the photo.

2. **Click the Capture button:** After starting the camera user must click on capture button to capture the photo of the current frame which will be present in the GUI.

3. **Click the upload button to predict the face:** After Capturing the photo user must click on the upload button which will predict the face of the user if it is present in our system or not.

4. **Click on confirm button to mark the attendance:** This is the final step to mark the attendance, if the user confirms that the predicted face is correct then he or she click on confirm button which will mark the attendance i.e it will upload the entried in to the database.

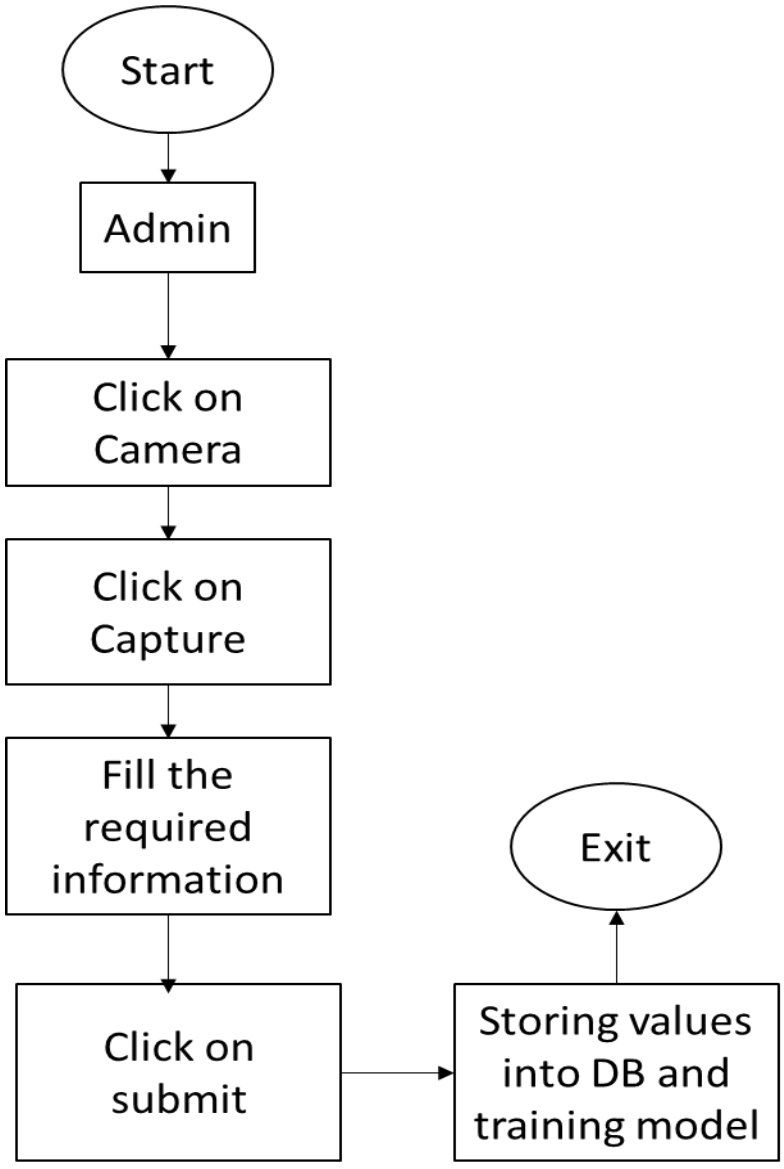


Fig. 5.4: Activity Diagram of admin adding user into the system

The above figure shows how the admin will add the user into the system, the explanation is mentioned below:

1. **Click on Camera:** Admin will click on camera to start the camera in the system. In this also we are using OpenCV library to start the camera and capture the photo.
2. **Click on capture:** In GUI there is an button the capture the photos after starting the camera, that capture button will click 200 image where user has to given certain expression while capturing the photos.
3. **Fill the required information:** After capturing the photo, the admin has to fill the information for the particular user for which the photo is clicked.
4. **Submit button:** After filling the information, Admin will click on the submit button, the submit button will validate the fields, Insert the values into the DB and train the model

5.3 Design Model – Class Diagram (Detailed Design)

Face Training.py:-

```
1. # -*- coding: utf-8 -*-

   # Form implementation generated from reading ui file 'FormUI.ui'
   #
   # Created by: PyQt5 UI code generator 5.15.4
   #
   # WARNING: Any manual changes made to this file will be lost when
   pyuic5 is
   # run again.  Do not edit this file unless you know what you are doing.
   import os
   from PyQt5 import QtCore, QtGui, QtWidgets, QtSql
   import cv2 as cv
   from PyQt5.QtGui import QImage, QPixmap
   from Camera1 import UiDialog
   from face_recog import face_recog
   import DatabaseConnection as dbClass

   CapimageStorLoc = "CapturedImage"

   class Ui_MainWindow(object):

       def setupUi(self, MainWindow):
           MainWindow.setObjectName("MainWindow")
           MainWindow.resize(764, 562)
           MainWindow.setLayoutDirection(QtCore.Qt.RightToLeft)

           self.logic = 0
           self.value = 1
           self.centralwidget = QtWidgets.QWidget(MainWindow)
           self.centralwidget.setObjectName("centralwidget")

           font = QtGui.QFont()
           font.setPointSize(16)
           self.CameraButton = QtWidgets.QPushButton(self.centralwidget)
           self.CameraButton.setGeometry(QtCore.QRect(170, 17, 431, 41))
```

```
        self.CameraButton.setObjectName("CameraButton")
        self.CameraButton.setFont(font)
        self.CameraButton.clicked.connect(self.onClicked)
        self.Photo_Box = QtWidgets.QLabel(self.centralwidget)
        self.Photo_Box.setGeometry(QtCore.QRect(20, 90, 321, 311))
        self.Photo_Box.setText("")
        self.Photo_Box.setPixmap(QtGui.QPixmap("ImageBasic/Train/Firoz
Rangrez/0_2.jpg"))
        #self.Photo_Box.setScaledContents(True)
        self.Photo_Box.setObjectName("Photo_Box")

        self.Photo_Box.setAlignment(QtCore.Qt.AlignCenter)
        self.label_2 = QtWidgets.QLabel(self.centralwidget)
        self.label_2.setGeometry(QtCore.QRect(390, 110, 71, 31))
        font = QtGui.QFont()
        font.setPointSize(13)
        self.label_2.setFont(font)
        self.label_2.setScaledContents(True)
        self.label_2.setObjectName("label_2")
        self.RollNumber_Label = QtWidgets.QLabel(self.centralwidget)
        self.RollNumber_Label.setGeometry(QtCore.QRect(390, 150, 131,
31))
        font = QtGui.QFont()
        font.setPointSize(13)
        self.RollNumber_Label.setFont(font)
        self.RollNumber_Label.setScaledContents(True)
        self.RollNumber_Label.setObjectName("RollNumber_Label")
        self.AdmissionYear_Label = QtWidgets.QLabel(self.centralwidget)
        self.AdmissionYear_Label.setGeometry(QtCore.QRect(390, 190,
151, 31))
        font = QtGui.QFont()
        font.setPointSize(13)
        self.AdmissionYear_Label.setFont(font)
        self.AdmissionYear_Label.setScaledContents(True)
        self.AdmissionYear_Label.setObjectName("AdmissionYear_Label")
        self.CurrentYear_Label = QtWidgets.QLabel(self.centralwidget)
        self.CurrentYear_Label.setGeometry(QtCore.QRect(390, 230, 151,
31))
        font = QtGui.QFont()
        font.setPointSize(13)
        self.CurrentYear_Label.setFont(font)
        self.CurrentYear_Label.setScaledContents(True)
        self.CurrentYear_Label.setObjectName("CurrentYear_Label")

        self.Image_AdmissionYear =
QtWidgets.QTextEdit(self.centralwidget)
        self.Image_AdmissionYear.setGeometry(QtCore.QRect(890, 270,
211, 31))
        self.Image_AdmissionYear.setObjectName("Image_AdmissionYear")
        self.Attendance_ConfirmButton =
QtWidgets.QPushButton(self.centralwidget)
        self.Attendance_ConfirmButton.setGeometry(QtCore.QRect(260,
490, 181, 28))

self.Attendance_ConfirmButton.setObjectName("Attendance_ConfirmButton")
        self.RollNumber_LabelVal = QtWidgets.QLabel(self.centralwidget)
        self.RollNumber_LabelVal.setGeometry(QtCore.QRect(540, 150,
121, 31))
        font = QtGui.QFont()
        font.setPointSize(14)
        self.RollNumber_LabelVal.setFont(font)
        self.RollNumber_LabelVal.setText("")
        self.RollNumber_LabelVal.setObjectName("RollNumber_LabelVal")
```

```
            self.AdmissionYear_LabelVal =
    QtWidgets.QLabel(self.centralwidget)
            self.AdmissionYear_LabelVal.setGeometry(QtCore.QRect(550, 190,
    121, 31))
            font = QtGui.QFont()
            font.setPointSize(14)
            self.AdmissionYear_LabelVal.setFont(font)
            self.AdmissionYear_LabelVal.setText("")

    self.AdmissionYear_LabelVal.setObjectName("AdmissionYear_LabelVal")
            self.CurrentYear_LabelVal =
    QtWidgets.QLabel(self.centralwidget)
            self.CurrentYear_LabelVal.setGeometry(QtCore.QRect(550, 230,
    121, 31))
            font = QtGui.QFont()
            font.setPointSize(14)
            self.CurrentYear_LabelVal.setFont(font)
            self.CurrentYear_LabelVal.setText("")
            self.CurrentYear_LabelVal.setObjectName("CurrentYear_LabelVal")
            self.Name_LabelVal = QtWidgets.QLabel(self.centralwidget)
            self.Name_LabelVal.setGeometry(QtCore.QRect(250, 450, 55, 16))
            self.Name_LabelVal.setText("")
            self.Name_LabelVal.setObjectName("Name_LabelVal")
            self.NameLabel_Val = QtWidgets.QLabel(self.centralwidget)
            self.NameLabel_Val.setGeometry(QtCore.QRect(550, 120, 151, 21))
            font = QtGui.QFont()
            font.setPointSize(14)
            self.NameLabel_Val.setFont(font)
            self.NameLabel_Val.setObjectName("NameLabel_Val")
            self.CaptureBtn = QtWidgets.QPushButton(self.centralwidget)
            self.CaptureBtn.setGeometry(QtCore.QRect(80, 430, 93, 28))
            self.CaptureBtn.setObjectName("CaptureBtn")
            self.UploadBtn = QtWidgets.QPushButton(self.centralwidget)
            self.UploadBtn.setGeometry(QtCore.QRect(200, 430, 93, 28))
            self.UploadBtn.setObjectName("UploadBtn")
            self.StatusLabel = QtWidgets.QLabel(self.centralwidget)
            self.StatusLabel.setGeometry(QtCore.QRect(10, 540, 55, 16))
            self.StatusLabel.setObjectName("StatusLabel")
            self.StatusOfExecution = QtWidgets.QLabel(self.centralwidget)
            self.StatusOfExecution.setGeometry(QtCore.QRect(80, 540, 331,
    16))
            self.StatusOfExecution.setText("")
            self.StatusOfExecution.setObjectName("StatusOfExecution")
            self.CaptureBtn.clicked.connect(self.CaptureClicked)
            self.UploadBtn.clicked.connect(self.uploadPhotoAndDetect)
            MainWindow.setCentralWidget(self.centralwidget)

            self.retranslateUi(MainWindow)
            QtCore.QMetaObject.connectSlotsByName(MainWindow)

        def CaptureClicked(self):
            self.StatusOfExecution.setText("Image Captured")
            self.logic = 2

        def on_pushButton_clicked(self):
            self.dialog = QtWidgets.QDialog()
            self.ui = UiDialog()
            self.ui.setupUi(self.dialog)
            self.dialog.show()
            self.ui.onClicked(self.dialog)

        # Starting Camera
        def onClicked(self):
```

```
            self.setvalues([])
            self.StatusOfExecution.setText("Initializing Camera....")
            cap = cv.VideoCapture(0)
            while(cap.isOpened()):
                ret,frame = cap.read()
                self.StatusOfExecution.setText("Camera Started....")
                self.Photo_Box.setText("")
                if ret == True:
                    self.displayImage(frame, 1)
                    cv.waitKey()
                    if self.logic == 2:
                        self.value = self.value + 1
                        filePath = os.path.join(CapimageStorLoc,'1.jpg')
                        cv.imwrite(filePath,frame)
                        self.logic = 1
                        break
                else:
                    print('return not found')
            cap.release()
            cv.destroyAllWindows()

        def uploadPhotoAndDetect(self):
            faceObj = face_recog()
            db = dbClass.DbConnection()
            self.StatusOfExecution.setText("Detecting Face.....")
            filePath = os.path.join(CapimageStorLoc, '1.jpg')
            filePathOfCroppedImage = faceObj.DetectFace(filePath)
            print("File path of cropped image
    ",str(filePathOfCroppedImage))
            if(filePathOfCroppedImage != None):

    self.Photo_Box.setPixmap(QtGui.QPixmap(filePathOfCroppedImage))
                rollNumber = faceObj.facePrediction(filePath)
                valuesFromDB = db.getnamefromrollno(int(rollNumber))
                self.setvalues([i for sub in valuesFromDB for i in sub])
                self.StatusOfExecution.setText("Face Detected")
            else:
                self.StatusOfExecution.setText("No face detected.....")

        def setvalues(self, arrayOfVal):
            if len(arrayOfVal) > 0:
                self.NameLabel_Val.setText(arrayOfVal[2] + "
    "+arrayOfVal[3])
                self.CurrentYear_LabelVal.setText(arrayOfVal[4])
                self.AdmissionYear_LabelVal.setText(arrayOfVal[4])
                self.RollNumber_LabelVal.setText(str(arrayOfVal[1]))
            else:
                self.NameLabel_Val.setText("")
                self.CurrentYear_LabelVal.setText("")
                self.AdmissionYear_LabelVal.setText("")
                self.RollNumber_LabelVal.setText("")

        # Displaying image of camera
        def displayImage(self, img, window=1):
            qformat = QImage.Format_Indexed8
            if(len(img.shape) == 3):
                if(img.shape[2] == 4):
                    qformat=QImage.Format_RGBA8888
                else:
                    qformat= QImage.Format_RGB888
            img = QImage(img,img.shape[1],img.shape[0],qformat)
            img = img.rgbSwapped()
            self.Photo_Box.setPixmap(QPixmap.fromImage(img))
```

```
            self.Photo_Box.setAlignment(QtCore.Qt.AlignHCenter |
    QtCore.Qt.AlignVCenter)

        def retranslateUi(self, MainWindow):
            _translate = QtCore.QCoreApplication.translate
            MainWindow.setWindowTitle(_translate("MainWindow",
    "MainWindow"))
            self.CameraButton.setText(_translate("MainWindow", "Start
    Camera"))
            self.label_2.setText(_translate("MainWindow", "Name"))
            self.RollNumber_Label.setText(_translate("MainWindow", "Roll
    Number"))
            self.AdmissionYear_Label.setText(_translate("MainWindow",
    "Admission year"))
            self.CurrentYear_Label.setText(_translate("MainWindow",
    "Current year"))
            self.Attendance_ConfirmButton.setText(_translate("MainWindow",
    "Confirm Attendance"))
            self.CaptureBtn.setText(_translate("MainWindow", "Capture"))
            self.UploadBtn.setText(_translate("MainWindow", "Upload"))
            self.StatusLabel.setText(_translate("MainWindow", "Status :"))

    if __name__ == "__main__":
        import sys
        app = QtWidgets.QApplication(sys.argv)
        MainWindow = QtWidgets.QMainWindow()
        ui = Ui_MainWindow()
        ui.setupUi(MainWindow)
        MainWindow.show()
        sys.exit(app.exec_())
```

DataBaseConnection.py: -

```
import sqlite3


class DbConnection:

    def __init__(self):
        global con
        = sqlite3.connect('mydatabase.db')

    def CreateDbAndInsertValues(self):
        cursorObj = con.cursor()
        cursorObj.execute("CREATE TABLE IF NOT EXISTS Student(id integer
PRIMARY KEY AUTOINCREMENT,RollNumber integer "
                          "UNIQUE, First_Name text,Last_Name text,
AdmissionYear text)")
        con.commit()
        """cursorObj.execute("Insert into Student(RollNumber, First_Name
,Last_Name , AdmissionYear) Values(123,'Firoz',"
                          "'Rangrez','2019')")
        cursorObj.execute("Insert into Student(RollNumber, First_Name
,Last_Name , AdmissionYear) Values(456,"
                          "'Sameer','Kodgire','2019')")
        cursorObj.execute("Insert into Student(RollNumber, First_Name
,Last_Name , AdmissionYear) Values(789,"
                          "'Anchita','Lokhande','2019')")
        con.commit()"""
```

```
        cursorObj.execute("CREATE TABLE IF NOT EXISTS Attendance(id integer
PRIMARY KEY AUTOINCREMENT,RollNumber integer "
                          "UNIQUE, First_Name text,Last_Name text, Date
text,Status text)")
con.commit()
        con.close()

    def getlistofidsfromdb(self):
        cursorObj = con.cursor()
        cursorObj.execute("Select RollNumber from Student")
        rows = cursorObj.fetchall()
        #print(rows)
        return rows

    def getnamefromrollno(self,rollNumber):
        cursorObj = con.cursor()
        cursorObj.execute(f"Select * from Student where RollNumber=
{rollNumber}")
        rows = cursorObj.fetchall()
        return rows

    def MarkAttendance(self,arrayOfVals):

        cursorObj = con.cursor()

        cursorObj.execute(f"Insert into Attendance(RollNumber ,First_Name,
Last_Name, Date, Status) Values"

f"({arrayOfVals[0]},{arrayOfVals[1]},{arrayOfVals[2]},{arrayOfVals[3]},{arra
yOfVals[4]},{arrayOfVals[5]})")
        con.commit()

    def InsertValueIntoUserTable(self,dbValues):
        print("Value inserted into DB")
        """ cursorObj = con.cursor()

        cursorObj.execute(f"Insert into User(RollNumber, First_Name
,Last_Name , AdmissionYear,Branch,UserType)
Values({dbValues['RollNumber']},'{dbValues['FirstName']}',"

f"'{dbValues['LastName']}','{dbValues['AdmissionYear']}')")
        con.commit()"""
```

face_recog.py: -

```
# pylint:disable=no-member

import numpy as np
import cv2 as cv
import DatabaseConnection as dbClass

CapimageStorLoc = "CapturedImage"
db = dbClass.DbConnection()
res = db.getlistofidsfromdb()
arr = np.array(res).ravel()
people = []
haar_cascade = cv.CascadeClassifier('haarcascade_frontalface_default.xml')
```

```
class face_recog():
    def facePrediction(self,imgpath):
        print(imgpath)
        for name in arr:
            people.append(str(name))
        # features = np.load('features.npy', allow_pickle=True)
        # labels = np.load('labels.npy')

        face_recognizer = cv.face.LBPHFaceRecognizer_create(
            radius=1,
            neighbors=6,
            grid_x=8,
            grid_y=8)
        face_recognizer.read('face_trained.yml')

        img = cv.imread(imgpath)

        # img = cv.resize(img,(300,300))
        gray = cv.cvtColor(img, cv.COLOR_BGR2GRAY)
        #cv.imshow('Person', gray)
        RollNumber = ""
        # Detect the face in the image
        faces_rect = haar_cascade.detectMultiScale(gray,  1.3, 6)
        print(len(faces_rect))
        if(len(faces_rect) > 0):
            for (x, y, w, h) in faces_rect:
                faces_roi = gray[y:y + h, x:x + w]
                print(faces_rect)
                label, confidence = face_recognizer.predict(faces_roi)
                names = db.getnamefromrollno(people[label])
                b = [i for sub in names for i in sub]
                print(b)
                print(f'Label = {" ".join(b)} with a confidence of {confidence}')
                RollNumber = str(people[label])
                '''cv.putText(img, str(" ".join(b)), (20, 20), cv.FONT_HERSHEY_COMPLEX, 1.0, (0, 255, 0),
thickness=2)
                cv.rectangle(img, (x, y), (x + w, y + h), (0, 255, 0), thickness=2)'''
        else:
            RollNumber = -1
        return RollNumber
        '''cv.imshow('Detected Face', img)

        cv.waitKey(0)'''

    def DetectFace(self,path):
        print(path)
        img = cv.imread(path)
        # convert to gray scale of each frames
        gray = cv.cvtColor(img, cv.COLOR_BGR2GRAY)

        # Detects faces of different sizes in the input image
        faces = haar_cascade.detectMultiScale(gray, 1.3, 6)
        print(faces)
        if len(faces) > 0:
            for (x, y, w, h) in faces:
```

```
            faces = img[y:y + h, x:x + w]
            grayCroppedImage = cv.cvtColor(faces, cv.COLOR_BGR2GRAY)
            fileName = rf"{CapimageStorLoc}/CroppedImage_1.jpg"
            faces = cv.resize(faces, (400, 400))
            cv.imwrite(filename=fileName, img=faces)
            return fileName

    def DetectFaceWithParentFolder(self,path,Parentfolder):
        print(path)
        img = cv.imread(path)
        # convert to gray scale of each frames
        gray = cv.cvtColor(img, cv.COLOR_BGR2GRAY)

        # Detects faces of different sizes in the input image
        faces = haar_cascade.detectMultiScale(gray, 1.3, 6)
        print(faces)
        if len(faces) > 0:
            for (x, y, w, h) in faces:
                faces = img[y:y + h, x:x + w]
                grayCroppedImage = cv.cvtColor(faces, cv.COLOR_BGR2GRAY)
                fileName = rf"{Parentfolder}/CroppedImage.jpg"
                faces = cv.resize(faces, (400, 400))
                cv.imwrite(filename=fileName, img=faces)
                return fileName
```

FORMS: -

1. Admin.py:-

```
2. # -*- coding: utf-8 -*-

    # Form implementation generated from reading ui file 'Admin.ui'
    #
    # Created by: PyQt5 UI code generator 5.15.4
    #
    # WARNING: Any manual changes made to this file will be lost when
    pyuic5 is
    # run again.  Do not edit this file unless you know what you are doing.
    import shutil

    from PyQt5 import QtCore, QtGui, QtWidgets
    import cv2 as cv
    import os
    from face_recog import face_recog

    from PyQt5.QtGui import QImage, QPixmap
    from PyQt5.QtWidgets import QDialog, QMessageBox
    from face_train import CreateTraining

    import DatabaseConnection as dbClass
    CapturedImageFromAdminPath = "ImageForTraining"
    ImageTrainingPath = r"ImageBasic\Train"


    class Ui_AdminWindow(object):
        def setupUi(self, AdminWindow):
            AdminWindow.setObjectName("AdminWindow")
            AdminWindow.resize(837, 639)
```

```
            self.logic = 1
            self.value = 1
            self.isPhotoCaptured = False
            self.centralwidget = QtWidgets.QWidget(AdminWindow)
            self.centralwidget.setObjectName("centralwidget")
            self.Camera_Label = QtWidgets.QLabel(self.centralwidget)
            self.Camera_Label.setGeometry(QtCore.QRect(40, 60, 291, 331))
            self.Camera_Label.setAlignment(QtCore.Qt.AlignCenter)
            self.Camera_Label.setObjectName("Camera_Label")
            self.StartCamera_btn =
    QtWidgets.QPushButton(self.centralwidget)
            self.StartCamera_btn.setGeometry(QtCore.QRect(40, 430, 131,
    41))
            self.StartCamera_btn.setObjectName("StartCamera_btn")
            self.Capture_btn = QtWidgets.QPushButton(self.centralwidget)
            self.Capture_btn.setGeometry(QtCore.QRect(190, 430, 131, 41))
            self.Capture_btn.setObjectName("Capture_btn")
            self.PhotoCapture_Label = QtWidgets.QLabel(self.centralwidget)
            self.PhotoCapture_Label.setGeometry(QtCore.QRect(50, 490, 61,
    16))
            self.PhotoCapture_Label.setObjectName("PhotoCapture_Label")
            self.Status_Label = QtWidgets.QLabel(self.centralwidget)
            self.Status_Label.setGeometry(QtCore.QRect(10, 610, 61, 16))
            font = QtGui.QFont()
            font.setPointSize(10)
            self.Status_Label.setFont(font)
            self.Status_Label.setObjectName("Status_Label")
            self.Status_Val = QtWidgets.QLabel(self.centralwidget)
            self.Status_Val.setGeometry(QtCore.QRect(70, 610, 281, 16))
            font = QtGui.QFont()
            font.setPointSize(10)
            self.Status_Val.setFont(font)
            self.Status_Val.setObjectName("Status_Val")
            self.Name_Val = QtWidgets.QLineEdit(self.centralwidget)
            self.Name_Val.setGeometry(QtCore.QRect(560, 120, 221, 31))
            self.Name_Val.setObjectName("Name_Val")
            self.RollNumber_Val = QtWidgets.QLineEdit(self.centralwidget)
            self.RollNumber_Val.setGeometry(QtCore.QRect(560, 160, 221,
    31))
            self.RollNumber_Val.setObjectName("RollNumber_Val")
            self.Name_Label = QtWidgets.QLabel(self.centralwidget)
            self.Name_Label.setGeometry(QtCore.QRect(440, 120, 55, 16))
            self.Name_Label.setObjectName("Name_Label")
            self.RollNumber_Label = QtWidgets.QLabel(self.centralwidget)
            self.RollNumber_Label.setGeometry(QtCore.QRect(440, 160, 55,
    16))
            self.RollNumber_Label.setObjectName("RollNumber_Label")
            self.Branch_Label = QtWidgets.QLabel(self.centralwidget)
            self.Branch_Label.setGeometry(QtCore.QRect(440, 205, 55, 16))
            self.Branch_Label.setObjectName("Branch_Label")
            self.AdmissionYear_Label = QtWidgets.QLabel(self.centralwidget)
            self.AdmissionYear_Label.setGeometry(QtCore.QRect(440, 250,
    101, 16))
            self.AdmissionYear_Label.setObjectName("AdmissionYear_Label")
            self.UserType_CmbBox = QtWidgets.QComboBox(self.centralwidget)
            self.UserType_CmbBox.setGeometry(QtCore.QRect(560, 70, 221,
    31))
            self.UserType_CmbBox.setObjectName("UserType_CmbBox")
            self.UserType_CmbBox.addItem("")
            self.UserType_CmbBox.addItem("")
            self.UserType_CmbBox.addItem("")
            self.UserType_Label = QtWidgets.QLabel(self.centralwidget)
            self.UserType_Label.setGeometry(QtCore.QRect(440, 80, 61, 16))
```

```
        self.UserType_Label.setObjectName("UserType_Label")
        self.Submit_btn = QtWidgets.QPushButton(self.centralwidget)
        self.Submit_btn.setGeometry(QtCore.QRect(330, 520, 201, 71))
        font = QtGui.QFont()
        font.setPointSize(14)
        self.Submit_btn.setFont(font)
        self.Submit_btn.setObjectName("Submit_btn")
        self.line = QtWidgets.QFrame(self.centralwidget)
        self.line.setGeometry(QtCore.QRect(-3, 500, 841, 20))
        self.line.setFrameShape(QtWidgets.QFrame.HLine)
        self.line.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.line.setObjectName("line")
        self.line_2 = QtWidgets.QFrame(self.centralwidget)
        self.line_2.setGeometry(QtCore.QRect(410, 0, 20, 511))
        self.line_2.setFrameShape(QtWidgets.QFrame.VLine)
        self.line_2.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.line_2.setObjectName("line_2")
        self.Branch_CmbBox = QtWidgets.QComboBox(self.centralwidget)
        self.Branch_CmbBox.setGeometry(QtCore.QRect(560, 200, 221, 31))
        self.Branch_CmbBox.setObjectName("Branch_CmbBox")
        self.Branch_CmbBox.addItem("")
        self.Branch_CmbBox.addItem("")
        self.Branch_CmbBox.addItem("")
        self.Branch_CmbBox.addItem("")
        self.AdmissionYear_DtBox =
QtWidgets.QDateEdit(self.centralwidget)
        self.AdmissionYear_DtBox.setGeometry(QtCore.QRect(560, 250,
151, 31))

self.AdmissionYear_DtBox.setLayoutDirection(QtCore.Qt.LeftToRight)
        self.AdmissionYear_DtBox.setAlignment(QtCore.Qt.AlignCenter)
        self.AdmissionYear_DtBox.setCalendarPopup(False)
        self.AdmissionYear_DtBox.setObjectName("AdmissionYear_DtBox")
        self.withAttachment =
QtWidgets.QRadioButton(self.centralwidget)
        self.withAttachment.setGeometry(QtCore.QRect(560, 310, 71, 20))
        self.withAttachment.setObjectName("withAttachment")
        self.withoutAttachment =
QtWidgets.QRadioButton(self.centralwidget)
        self.withoutAttachment.setGeometry(QtCore.QRect(650, 310, 61,
20))
        self.withoutAttachment.setObjectName("withoutAttachment")
        self.ImageCap_Label = QtWidgets.QLabel(self.centralwidget)
        self.ImageCap_Label.setGeometry(QtCore.QRect(440, 310, 111,
20))
        self.ImageCap_Label.setObjectName("ImageCap_Label")
        self.StartCamera_btn.clicked.connect(self.onClicked)
        self.Submit_btn.clicked.connect(self.GetTheTextFromInputFields)
        self.Capture_btn.clicked.connect(self.CaptureClicked)
        AdminWindow.setCentralWidget(self.centralwidget)

        self.retranslateUi(AdminWindow)
        QtCore.QMetaObject.connectSlotsByName(AdminWindow)

    def retranslateUi(self, AdminWindow):
        _translate = QtCore.QCoreApplication.translate
        AdminWindow.setWindowTitle(_translate("AdminWindow",
"MainWindow"))
        self.Camera_Label.setText(_translate("AdminWindow", "Camera
Place"))
        self.StartCamera_btn.setText(_translate("AdminWindow", "Start
Camera"))
        self.Capture_btn.setText(_translate("AdminWindow", "Capture"))
```

```
        self.PhotoCapture_Label.setText(_translate("AdminWindow",
"1/500"))
        self.Status_Label.setText(_translate("AdminWindow", "Status:"))
        self.Status_Val.setText(_translate("AdminWindow", "----Status
filed ---"))
        self.Name_Label.setText(_translate("AdminWindow", "Name"))
        self.RollNumber_Label.setText(_translate("AdminWindow", "Roll
No"))
        self.Branch_Label.setText(_translate("AdminWindow", "Branch"))
        self.AdmissionYear_Label.setText(_translate("AdminWindow",
"Admission Year"))
        self.UserType_CmbBox.setItemText(0, _translate("AdminWindow",
"- Select item-"))
        self.UserType_CmbBox.setItemText(1, _translate("AdminWindow",
"Student"))
        self.UserType_CmbBox.setItemText(2, _translate("AdminWindow",
"Staff"))
        self.UserType_Label.setText(_translate("AdminWindow", "User
Type"))
        self.Submit_btn.setText(_translate("AdminWindow", "Submit"))
        self.Branch_CmbBox.setItemText(0, _translate("AdminWindow", "-
Select - "))
        self.Branch_CmbBox.setItemText(1, _translate("AdminWindow",
"B.E. Mechanical"))
        self.Branch_CmbBox.setItemText(2, _translate("AdminWindow",
"B.E. Computer"))
        self.Branch_CmbBox.setItemText(3, _translate("AdminWindow",
"B.E. Civil"))

self.AdmissionYear_DtBox.setDisplayFormat(_translate("AdminWindow",
"yyyy"))
        self.withAttachment.setText(_translate("AdminWindow", "Yes"))
        self.withoutAttachment.setText(_translate("AdminWindow", "No"))
        self.ImageCap_Label.setText(_translate("AdminWindow", "Image
Captured"))

    def onClicked(self):
        self.Status_Val.setText("Initializing Camera....")
        cap = cv.VideoCapture(0)
        while (cap.isOpened()):
            ret, frame = cap.read()
            self.Status_Val.setText("Camera Started....")
            self.Camera_Label.setText("")
            if ret == True:
                self.displayImage(frame, 1)
                cv.waitKey()
                if self.logic == 2:
                    self.Status_Val.setText("Capturing Photos....")
                    if self.value == 1:

self.DeleteAllFilesFromFolder(CapturedImageFromAdminPath)
                        filePath = os.path.join(CapturedImageFromAdminPath,
f'Og_{self.value}.jpg')
                        cv.imwrite(filePath, frame)
                        fObj = face_recog()
                        facedetectedFilePath =
fObj.DetectFaceWithParentFolder(filePath,
Parentfolder=CapturedImageFromAdminPath)
                        print(self.value)
                        if facedetectedFilePath is not None:
                            if len(facedetectedFilePath) > 0:
                                os.rename(facedetectedFilePath,
os.path.join(CapturedImageFromAdminPath, f'C_{self.value}.jpg'))
```

```
                            os.remove(filePath)
                            self.value = self.value + 1

self.PhotoCapture_Label.setText(f"{self.value}/100")
                            if self.value == 200:
                                self.logic = 1
                                self.Status_Val.setText("All images has
been clicked")
                                self.Camera_Label.setText("Camera
Stopped")
                                self.isPhotoCaptured = True
                                break
                    else:
                        self.Status_Val.setText("No face detected")

            else:
                print('return not found')
        cap.release()
        cv.destroyAllWindows()

    def DeleteAllFilesFromFolder(self, folder):
        for filename in os.listdir(folder):
            file_path = os.path.join(folder, filename)
            try:
                if os.path.isfile(file_path) or
os.path.islink(file_path):
                    os.unlink(file_path)
                elif os.path.isdir(file_path):
                    shutil.rmtree(file_path)
            except Exception as e:
                print('Failed to delete %s. Reason: %s' % (file_path,
e))

    def CaptureClicked(self):
        self.logic = 2

    # Displaying image of camera
    def displayImage(self, img, window=1):
        qformat = QImage.Format_Indexed8
        if (len(img.shape) == 3):
            if (img.shape[2] == 4):
                qformat = QImage.Format_RGBA8888
            else:
                qformat = QImage.Format_RGB888
        img = QImage(img, img.shape[1], img.shape[0], qformat)
        img = img.rgbSwapped()
        self.Camera_Label.setPixmap(QPixmap.fromImage(img))
        self.Camera_Label.setAlignment(QtCore.Qt.AlignHCenter |
QtCore.Qt.AlignVCenter)

    def GetTheTextFromInputFields(self):
        userType = self.UserType_CmbBox.currentText()
        if len(userType) < 1:
            self.CallMessageBox("User type is required", "Roll Number
Error")
            return
        if userType == "- Select item-":
            self.CallMessageBox("User type is required", "Roll Number
Error")
            return

        name = self.Name_Val.text()
        if len(name) < 4:
```

```
            self.CallMessageBox("Name length should be greater than 4",
"Name Error")
            return

        rollnumber = self.RollNumber_Val.text()
        if len(rollnumber) < 1:
            self.CallMessageBox("Roll number length should be greater
than 0", "Roll Number Error")
            return
        admissYear = self.AdmissionYear_DtBox.text()
        print(admissYear)
        if len(admissYear) < 1:
            self.CallMessageBox("Admission Year is required",
"Admission Year Error")
            return
        branch = self.Branch_CmbBox.currentText()
        if len(branch) < 1:
            self.CallMessageBox("Branch is required", "Branch Error")
            return

        if branch == "- Select - ":
            self.CallMessageBox("Branch is required", "Roll Number
Error")
            return

        isAttachment_required = self.withAttachment.isChecked()
        if isAttachment_required:
            if not self.isPhotoCaptured:
                self.CallMessageBox("Please capture the photo", "Photo
Capture Error")
                return
            else:
                folderPath = os.path.join(ImageTrainingPath,
rollnumber)
                print(folderPath)

                shutil.copytree(CapturedImageFromAdminPath, folderPath)
                self.Status_Val.setText("File copied to training
folder")

        self.Status_Val.setText("Starting training")
        # Start training
        trainObj = CreateTraining()
        people = [rollnumber]
        trainObj.StartTraining(people)
        dbValues = {}
        dbValues["RollNumber"] = rollnumber
        dbValues["First Name"] = name
        dbValues["Admission Year"] = admissYear
        dbValues["User Type"] = userType
        dbValues["Branch"] = branch

        db = dbClass.DbConnection()
        db.InsertValueIntoUserTable(dbValues)

        self.Status_Val.setText("Training Completed")

    def accept(self):
        print("accept")

    def CallMessageBox(self, texttodisplay, MessageBoxTitle):
        self.msg = QMessageBox()  # create an instance of it
        self.msg.setIcon(QMessageBox.Information)
```

```
            self.msg.setText(texttodisplay)
            self.msg.setWindowTitle(MessageBoxTitle)
            self.msg.setStandardButtons(QMessageBox.Ok)
            # self.msg.accepted.connect(self.accept)
            self.msg.exec_()


    if __name__ == "__main__":
        import sys

        app = QtWidgets.QApplication(sys.argv)
        AdminWindow = QtWidgets.QMainWindow()
        ui = Ui_AdminWindow()
        ui.setupUi(AdminWindow)
        AdminWindow.show()
        sys.exit(app.exec_())
```

2. Form.py: -

```
2. # -*- coding: utf-8 -*-

    # Form implementation generated from reading ui file 'FormUI.ui'
    #
    # Created by: PyQt5 UI code generator 5.15.4
    #
    # WARNING: Any manual changes made to this file will be lost when
    pyuic5 is
    # run again.  Do not edit this file unless you know what you are doing.
    import os
    from PyQt5 import QtCore, QtGui, QtWidgets, QtSql
    import cv2 as cv
    from PyQt5.QtGui import QImage, QPixmap
    from Camera1 import UiDialog
    from face_recog import face_recog
    import DatabaseConnection as dbClass

    CapimageStorLoc = "CapturedImage"

    class Ui_MainWindow(object):

        def setupUi(self, MainWindow):
            MainWindow.setObjectName("MainWindow")
            MainWindow.resize(764, 562)
            MainWindow.setLayoutDirection(QtCore.Qt.RightToLeft)

            self.logic = 0
            self.value = 1
            self.centralwidget = QtWidgets.QWidget(MainWindow)
            self.centralwidget.setObjectName("centralwidget")

            font = QtGui.QFont()
            font.setPointSize(16)
            self.CameraButton = QtWidgets.QPushButton(self.centralwidget)
            self.CameraButton.setGeometry(QtCore.QRect(170, 17, 431, 41))
            self.CameraButton.setObjectName("CameraButton")
            self.CameraButton.setFont(font)
            self.CameraButton.clicked.connect(self.onClicked)
            self.Photo_Box = QtWidgets.QLabel(self.centralwidget)
            self.Photo_Box.setGeometry(QtCore.QRect(20, 90, 321, 311))
            self.Photo_Box.setText("")
```

```
        self.Photo_Box.setPixmap(QtGui.QPixmap("ImageBasic/Train/Firoz
Rangrez/0_2.jpg"))
        #self.Photo_Box.setScaledContents(True)
        self.Photo_Box.setObjectName("Photo_Box")

        self.Photo_Box.setAlignment(QtCore.Qt.AlignCenter)
        self.label_2 = QtWidgets.QLabel(self.centralwidget)
        self.label_2.setGeometry(QtCore.QRect(390, 110, 71, 31))
        font = QtGui.QFont()
        font.setPointSize(13)
        self.label_2.setFont(font)
        self.label_2.setScaledContents(True)
        self.label_2.setObjectName("label_2")
        self.RollNumber_Label = QtWidgets.QLabel(self.centralwidget)
        self.RollNumber_Label.setGeometry(QtCore.QRect(390, 150, 131,
31))
        font = QtGui.QFont()
        font.setPointSize(13)
        self.RollNumber_Label.setFont(font)
        self.RollNumber_Label.setScaledContents(True)
        self.RollNumber_Label.setObjectName("RollNumber_Label")
        self.AdmissionYear_Label = QtWidgets.QLabel(self.centralwidget)
        self.AdmissionYear_Label.setGeometry(QtCore.QRect(390, 190,
151, 31))
        font = QtGui.QFont()
        font.setPointSize(13)
        self.AdmissionYear_Label.setFont(font)
        self.AdmissionYear_Label.setScaledContents(True)
        self.AdmissionYear_Label.setObjectName("AdmissionYear_Label")
        self.CurrentYear_Label = QtWidgets.QLabel(self.centralwidget)
        self.CurrentYear_Label.setGeometry(QtCore.QRect(390, 230, 151,
31))
        font = QtGui.QFont()
        font.setPointSize(13)
        self.CurrentYear_Label.setFont(font)
        self.CurrentYear_Label.setScaledContents(True)
        self.CurrentYear_Label.setObjectName("CurrentYear_Label")

        self.Image_AdmissionYear =
QtWidgets.QTextEdit(self.centralwidget)
        self.Image_AdmissionYear.setGeometry(QtCore.QRect(890, 270,
211, 31))
        self.Image_AdmissionYear.setObjectName("Image_AdmissionYear")
        self.Attendance_ConfirmButton =
QtWidgets.QPushButton(self.centralwidget)
        self.Attendance_ConfirmButton.setGeometry(QtCore.QRect(260,
490, 181, 28))

self.Attendance_ConfirmButton.setObjectName("Attendance_ConfirmButton")
        self.RollNumber_LabelVal = QtWidgets.QLabel(self.centralwidget)
        self.RollNumber_LabelVal.setGeometry(QtCore.QRect(540, 150,
121, 31))
        font = QtGui.QFont()
        font.setPointSize(14)
        self.RollNumber_LabelVal.setFont(font)
        self.RollNumber_LabelVal.setText("")
        self.RollNumber_LabelVal.setObjectName("RollNumber_LabelVal")
        self.AdmissionYear_LabelVal =
QtWidgets.QLabel(self.centralwidget)
        self.AdmissionYear_LabelVal.setGeometry(QtCore.QRect(550, 190,
121, 31))
        font = QtGui.QFont()
        font.setPointSize(14)
```

```
            self.AdmissionYear_LabelVal.setFont(font)
            self.AdmissionYear_LabelVal.setText("")

    self.AdmissionYear_LabelVal.setObjectName("AdmissionYear_LabelVal")
            self.CurrentYear_LabelVal =
    QtWidgets.QLabel(self.centralwidget)
            self.CurrentYear_LabelVal.setGeometry(QtCore.QRect(550, 230,
    121, 31))
            font = QtGui.QFont()
            font.setPointSize(14)
            self.CurrentYear_LabelVal.setFont(font)
            self.CurrentYear_LabelVal.setText("")
            self.CurrentYear_LabelVal.setObjectName("CurrentYear_LabelVal")
            self.Name_LabelVal = QtWidgets.QLabel(self.centralwidget)
            self.Name_LabelVal.setGeometry(QtCore.QRect(250, 450, 55, 16))
            self.Name_LabelVal.setText("")
            self.Name_LabelVal.setObjectName("Name_LabelVal")
            self.NameLabel_Val = QtWidgets.QLabel(self.centralwidget)
            self.NameLabel_Val.setGeometry(QtCore.QRect(550, 120, 151, 21))
            font = QtGui.QFont()
            font.setPointSize(14)
            self.NameLabel_Val.setFont(font)
            self.NameLabel_Val.setObjectName("NameLabel_Val")
            self.CaptureBtn = QtWidgets.QPushButton(self.centralwidget)
            self.CaptureBtn.setGeometry(QtCore.QRect(80, 430, 93, 28))
            self.CaptureBtn.setObjectName("CaptureBtn")
            self.UploadBtn = QtWidgets.QPushButton(self.centralwidget)
            self.UploadBtn.setGeometry(QtCore.QRect(200, 430, 93, 28))
            self.UploadBtn.setObjectName("UploadBtn")
            self.StatusLabel = QtWidgets.QLabel(self.centralwidget)
            self.StatusLabel.setGeometry(QtCore.QRect(10, 540, 55, 16))
            self.StatusLabel.setObjectName("StatusLabel")
            self.StatusOfExecution = QtWidgets.QLabel(self.centralwidget)
            self.StatusOfExecution.setGeometry(QtCore.QRect(80, 540, 331,
    16))
            self.StatusOfExecution.setText("")
            self.StatusOfExecution.setObjectName("StatusOfExecution")
            self.CaptureBtn.clicked.connect(self.CaptureClicked)
            self.UploadBtn.clicked.connect(self.uploadPhotoAndDetect)
            MainWindow.setCentralWidget(self.centralwidget)

            self.retranslateUi(MainWindow)
            QtCore.QMetaObject.connectSlotsByName(MainWindow)

        def CaptureClicked(self):
            self.StatusOfExecution.setText("Image Captured")
            self.logic = 2

        def on_pushButton_clicked(self):
            self.dialog = QtWidgets.QDialog()
            self.ui = UiDialog()
            self.ui.setupUi(self.dialog)
            self.dialog.show()
            self.ui.onClicked(self.dialog)

        # Starting Camera
        def onClicked(self):
            self.setvalues([])
            self.StatusOfExecution.setText("Initializing Camera....")
            cap = cv.VideoCapture(0)
            while(cap.isOpened()):
                ret,frame = cap.read()
                self.StatusOfExecution.setText("Camera Started....")
```

```
            self.Photo_Box.setText("")
            if ret == True:
                self.displayImage(frame, 1)
                cv.waitKey()
                if self.logic == 2:
                    self.value = self.value + 1
                    filePath = os.path.join(CapimageStorLoc,'1.jpg')
                    cv.imwrite(filePath,frame)
                    self.logic = 1
                    break
            else:
                print('return not found')
        cap.release()
        cv.destroyAllWindows()

    def uploadPhotoAndDetect(self):
        faceObj = face_recog()
        db = dbClass.DbConnection()
        self.StatusOfExecution.setText("Detecting Face.....")
        filePath = os.path.join(CapimageStorLoc, '1.jpg')
        filePathOfCroppedImage = faceObj.DetectFace(filePath)
        print("File path of cropped image
",str(filePathOfCroppedImage))
        if(filePathOfCroppedImage != None):

self.Photo_Box.setPixmap(QtGui.QPixmap(filePathOfCroppedImage))
            rollNumber = faceObj.facePrediction(filePath)
            valuesFromDB = db.getnamefromrollno(int(rollNumber))
            self.setvalues([i for sub in valuesFromDB for i in sub])
            self.StatusOfExecution.setText("Face Detected")
        else:
            self.StatusOfExecution.setText("No face detected.....")

    def setvalues(self, arrayOfVal):
        if len(arrayOfVal) > 0:
            self.NameLabel_Val.setText(arrayOfVal[2] + "
"+arrayOfVal[3])
            self.CurrentYear_LabelVal.setText(arrayOfVal[4])
            self.AdmissionYear_LabelVal.setText(arrayOfVal[4])
            self.RollNumber_LabelVal.setText(str(arrayOfVal[1]))
        else:
            self.NameLabel_Val.setText("")
            self.CurrentYear_LabelVal.setText("")
            self.AdmissionYear_LabelVal.setText("")
            self.RollNumber_LabelVal.setText("")

    # Displaying image of camera
    def displayImage(self, img, window=1):
        qformat = QImage.Format_Indexed8
        if(len(img.shape) == 3):
            if(img.shape[2] == 4):
                qformat=QImage.Format_RGBA8888
            else:
                qformat= QImage.Format_RGB888
        img = QImage(img,img.shape[1],img.shape[0],qformat)
        img = img.rgbSwapped()
        self.Photo_Box.setPixmap(QPixmap.fromImage(img))
        self.Photo_Box.setAlignment(QtCore.Qt.AlignHCenter |
QtCore.Qt.AlignVCenter)

    def retranslateUi(self, MainWindow):
        _translate = QtCore.QCoreApplication.translate
        MainWindow.setWindowTitle(_translate("MainWindow",
```

```
    "MainWindow"))
            self.CameraButton.setText(_translate("MainWindow", "Start
    Camera"))
            self.label_2.setText(_translate("MainWindow", "Name"))
            self.RollNumber_Label.setText(_translate("MainWindow", "Roll
    Number"))
            self.AdmissionYear_Label.setText(_translate("MainWindow",
    "Admission year"))
            self.CurrentYear_Label.setText(_translate("MainWindow",
    "Current year"))
            self.Attendance_ConfirmButton.setText(_translate("MainWindow",
    "Confirm Attendance"))
            self.CaptureBtn.setText(_translate("MainWindow", "Capture"))
            self.UploadBtn.setText(_translate("MainWindow", "Upload"))
            self.StatusLabel.setText(_translate("MainWindow", "Status :"))

    if __name__ == "__main__":
        import sys
        app = QtWidgets.QApplication(sys.argv)
        MainWindow = QtWidgets.QMainWindow()
        ui = Ui_MainWindow()
        ui.setupUi(MainWindow)
        MainWindow.show()
        sys.exit(app.exec_())
```

CHAPTER 6

IMPLEMENTED SYSTEM

6.1 System Architecture

The whole project is divided into the two module

1. Admin Module
2. User Module

Admin Module

The Admin can add the user based on their type, he or she can provide the information for that particular user. He or she can also train the model by giving the images to the model from the UI. In the UI Admin has option to capture 200 photos for a user if he is sitting in front of camera. After capturing the photo Admin can fill the information and submit the form which will add the new entry to the database and train the model with the captured image.

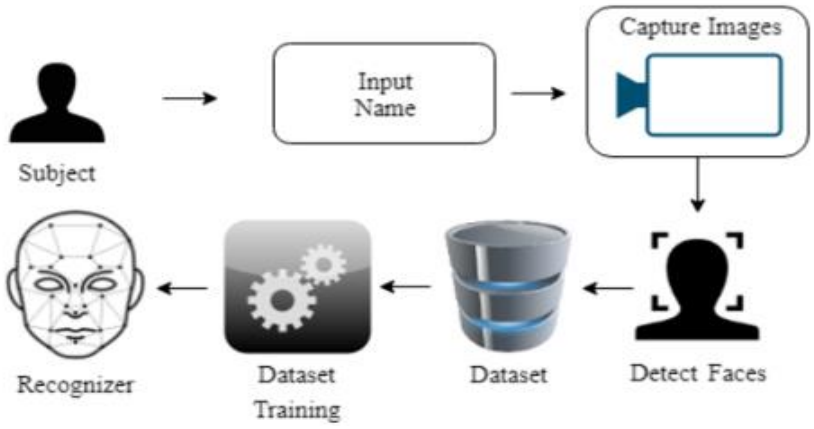


Fig.6.1: Dataset creating and training

The above diagram illustrates how the dataset will be created and the training will be done.

6.2 Dataset: -

The first step is to collect the faces of the user from the camera for creating the dataset. To detect the face from the photo, Haar Cascade Classifier is used. This classifier is based on Haar-like features which are efficient for detecting edge and lines. Moreover, Haar cascade classifier executes faster requiring less computations which is perfect for real time detection. Obviously, pictures are taken one person at a time during dataset creation. The default setting is set to take 200 images within 10 seconds from the live camera. The configuration can be changed to take more pictures for making a more accurate dataset. During data collection, different head position is suggested for creating a better dataset. In this system, if no face is detected while taking an image, data collection is halted.

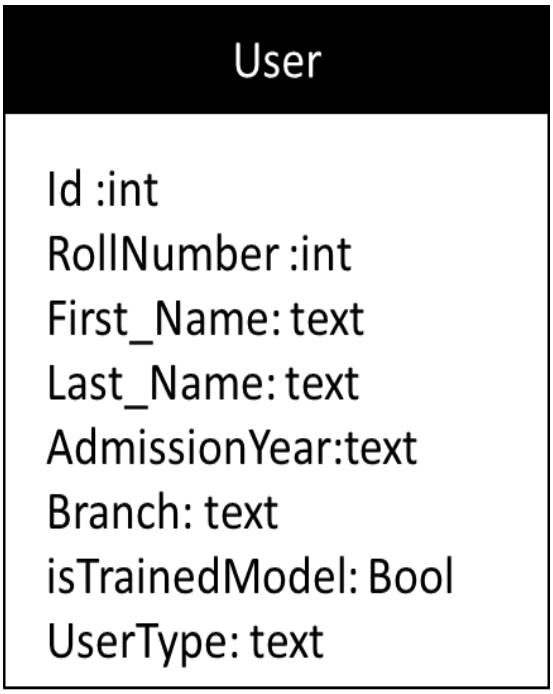
6.3 Data Training: -

In this step, LBPH is used which can recognize both front face and side face. It is an effective as well as straightforward texture operator. The pixels of a target image is labelled by thresholding every pixel's neighbourhood and then the result is taken as a binary number. Integrating Local Binary Pattern (LBP) with histograms of oriented gradients (HOG) model significantly boosts the performance of detection on a few datasets [13]. Firstly, LBPH generates an intermediate image which interprets the actual image in a more appropriate way by emphasizing the facial features. Here, LBPHFaceRecognizer.create() function of OpenCV library is used to create a trained recognizer. In this function, the path to the dataset directory is given first, then some labels and IDs are created by splitting the image file name.

6.4 User Module: -

User can mark the attendance from the GUI by capturing the photos. We have given provision to user that the user can capture the photo and upload it after that the GUI will automatically predict the user from captured image and show the result into the GUI. If the face is predicted properly then the user can click on confirm attendance to mark the attendance into the Database.

Below is the structure of database table



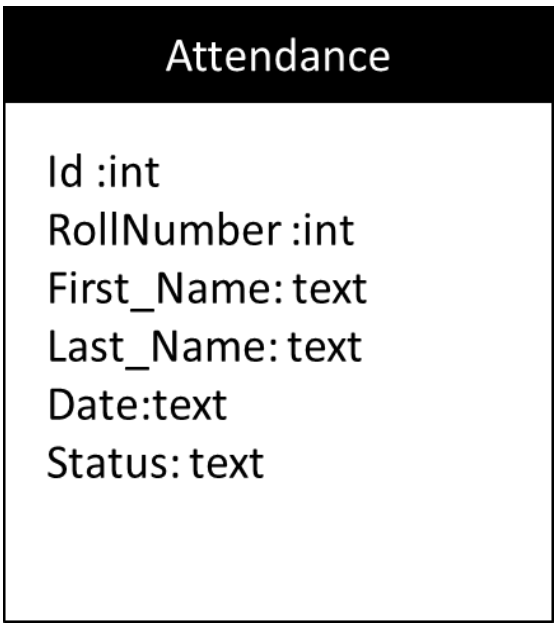


Fig. 6.2: Structure of Database Table

6.5 Results:-

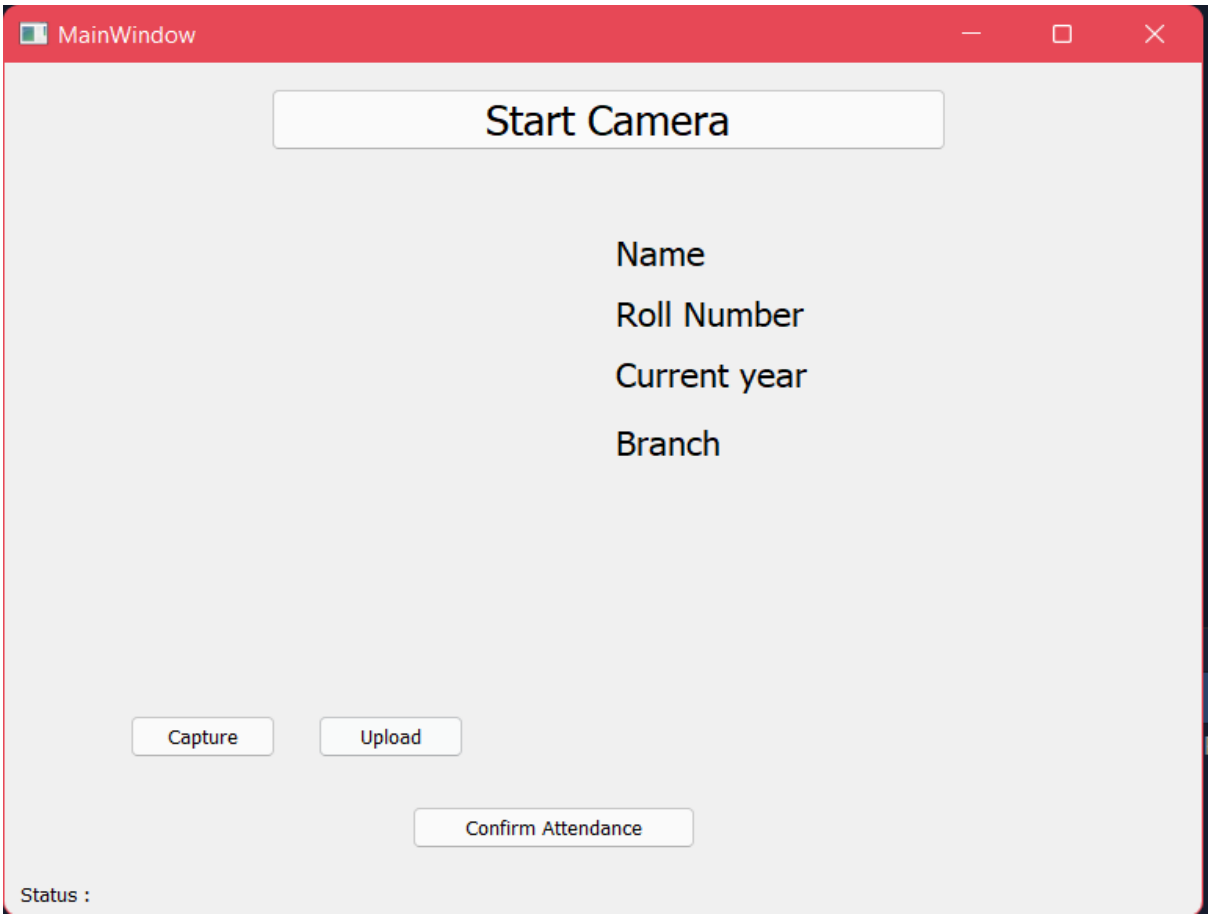


Fig.6.3: User marking attendance GUI

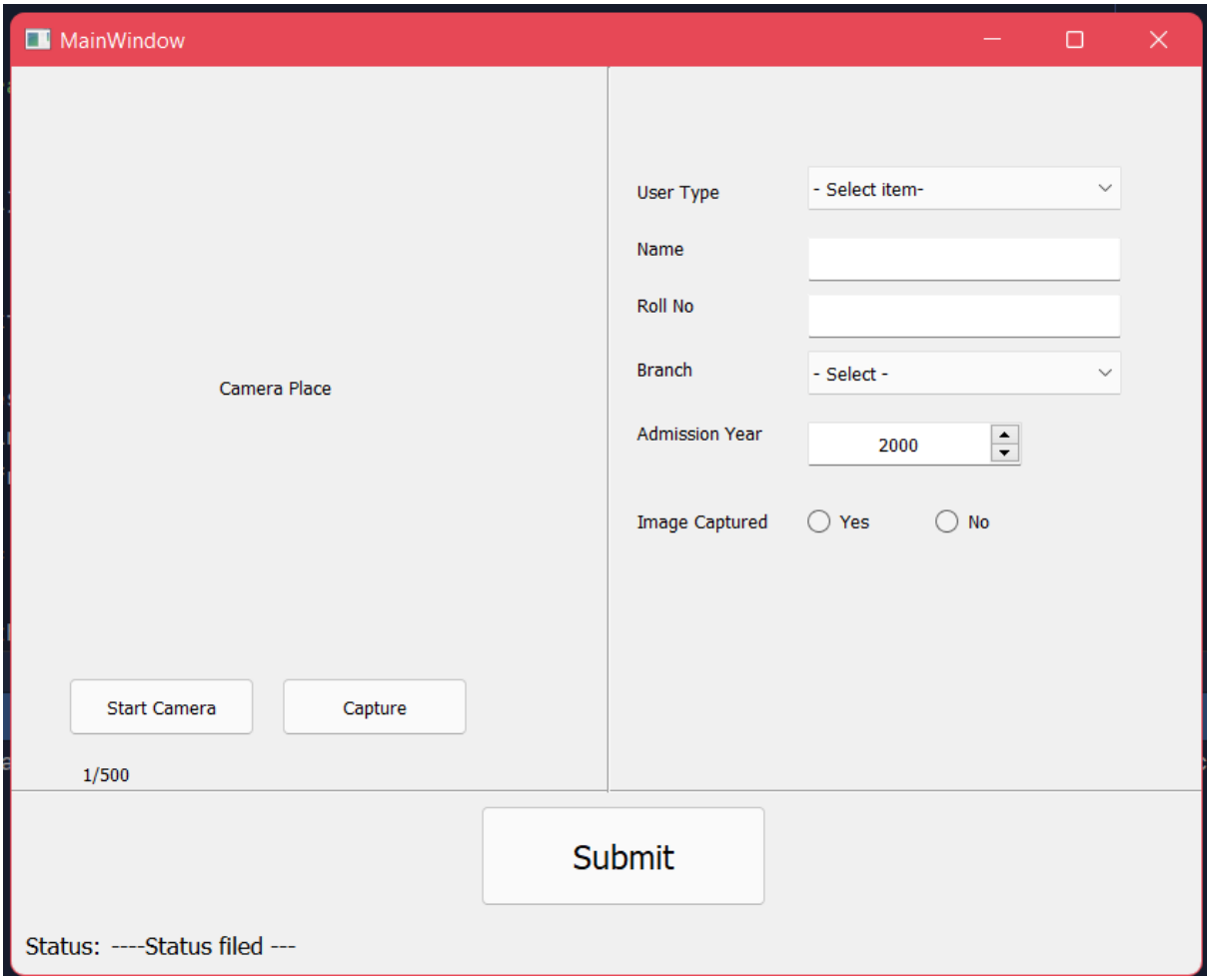


Fig. 6.4: Admin module GUI

CHAPTER 7

RESULT ANALYSIS

The main goal of this system is to help an organisation in recognizing the person for marking the attendance. So, the accuracy of face recognition of this system is the main concern. The accuracy of the developed face recognition sys-tem using LBPH algorithm is compared with Fisherface (FF) algorithm as shown in TABLE I. Fisherface algorithm is one of the popular and widely used algorithms in recognition of faces. It is the improved version of Eigenface algorithm [14]. From the table, it can be observed that the amount of images per person in the dataset affects the accuracy of facial recognition. In the first three rows, when there are 10, 20 or 50 images per person, the average accuracy of recognition is below 90%for LBPH and below 80% for Fisherface. And when the amount of images per person is increased to 100, the accuracy surpasses 90% for LBPH and 80% for Fisherface. The highest average accuracy achieved by the system is 93.33% using LBPH algorithm and 86.67% using Fisherface algorithm. So, we can say that LBPH performs better than Fisherface. One of the causes is that the performance of Fisherface algorithm greatly depends on the illumination of the sample images in the dataset. If the dataset images are in good illuminated condition, then this algorithm does not perform well for badly illuminated images. To obtain the best performance from this system, it is suggested to keep a uniform amount of images for every person in the dataset. In this study, the system has been tested using a web camera, the default camera of a computer whose resolution is low. But if a high-resolution camera is used, then this system will perform better. The training dataset was created with images of a few persons. Performance can be further analyzed using a dataset containing large group of persons.

Serial No.	Input Image Per Person	Results											
		Total Faces		*Total Recognition*		*Correct Recognition*		*Wrong Recognition*		*Accuracy (%)*		*Average Accuracy (%)*	
		LBPH	FF	LBPH	FF	LBPH	FF	LBPH	FF	LBPH	FF	LBPH	FF
1	10	1	1	5	5	4	3	1	2	80	60	66.67	53.33
2		2	2	5	5	3	3	2	2	60	60		
3		3	3	5	5	3	2	2	3	60	40		
4	20	1	1	5	5	4	4	1	1	80	80	73.33	66.67
5		2	2	5	5	4	3	1	2	80	60		
6		3	5	5	5	3	3	2	2	60	60		
7	50	1	1	5	5	5	4	0	1	100	80	86.67	73.33
8		2	2	5	5	4	4	1	1	80	80		
9		3	3	5	5	4	3	1	2	80	60		
10	100	1	1	5	5	5	5	0	0	100	100	93.33	86.67
11		2	2	5	5	5	4	0	1	100	80		
12		3	3	5	5	4	4	1	1	80	80		

Fig. 7.1: Accuracy for LBPH and FISHERFACE

CHAPTER 8

CONCLUSION AND FUTURE SCOPE

8.1 CONCLUSION

This system aims to build an effective class attendance system using face recognition techniques. The proposed system will be able to mark the attendance via face Id. It will detect faces via webcam and then recognize the faces. After recognition, it will mark the attendance of the recognized student and update the attendance record.

8.2 FUTURE SCOPE

- The face recognition model would be done more precisely so that the maximum accuracy can be achieved.
- Adding each student manually can be a tedious task, despite of this fetching data from excels would be efficient.
- Will be combining multiple algorithms to detect the face.
- Adding more functionalities to admin for managing the student view, deleted, edit options.
- Admin can download the attendance by providing some information into the GUI.

REFERENCE

[1] Face Recognition Based Attendance System using Python 2020 JETIR October 2020, Volume 7, Issue 10 https://www.jetir.org/papers/JETIR2010064.pdf

[2] Smart Attendance System using OPENCV based on Facial Recognition by Sudhir Bussa , Shruti Bharuka, Ananya Mani and Sakshi KaushikB. T. Jap, S. Lal, P. Fischer, and E. Bekiaris, "Using EEG spectral components to assess algorithms for detecting fatigue," Expert Systems with Applications, vol. 36, pp. 2352- 2359, 2009.

[3] John D. Woodward, Jr., Christopher Horn, Julius Gatune, and Aryn Thomas “Biometrics: A look at facial Recognition,” 2003. URL: https://apps.dtic.mil/sti/pdfs/ADA414520.pdfH. Seifoory, D. Taherkhani, B. Arzhang, Z. Eftekhari, and H. Memari, "An Accurate Morphological Drowsy Detection," ed: IEEE, 2011.

[4] Li Wang, Ali Akbar Siddique. "Facial recognition system using LBPH face recognizer for anti-theft and surveillance application based on drone technology" , Measurement and Control, 2020 URL: https://journals.sagepub.com/doi/10.1177/0020294020932344

[5] A brief history of facial recognition. 2020. URL: https://www.nec.co.nz/market-leadership/publications-media/abrief-history-of-facial-recognition/

[6] OpenCV Documentation [online]. URL: https://opencv.org/

[7] Dr. V Suresh“Facial Recognition Attendance System Using Python and OpenCv" Quest Journals Journal Of Software Engineering And Simulation, Vol. 05, No. 02, 2019, Pp. 18-29. https://www.questjournals.org/jses/papers/Vol5-issue-2/D05021829.pdf.