ABSTRACT

Attendance management system is a necessary tool for taking attendance in any environment where attendance is critical. However, most of the existing approaches are time consuming, intrusive and require manual work from the users. Face recognition is an important application of Image processing owing to its use in many fields. Identification of individuals in an organization for the purpose of attendance is one such application of face recognition. Maintenance and monitoring of attendance records plays a vital role in the analysis of performance of any organization. The purpose of developing attendance management systems is to computerize the traditional way of taking attendance. Automated Attendance Management System performs the daily activities of attendance marking and analysis with reduced human intervention. In this project we will be designing a single shot attendance system mobile application which will capture the group of faces of the student and check the presence of the student face in the database and mark it as present. As the attendance time closes an automatic report is generated and sent to the staff's registered mail id. Thus, this project helps in saving time of the people as well as automates the records which helps in saving manual maintenance. Implementation of this project also gives rise to many applications such as biometric scanning in bank lockers, etc.

TABLE OF CONTENTS

CHAPTER	TITLE	PAGE NO.
	ABSTRACT	**IV**
	LIST OF FIGURES	**VII**
	LIST OF SYMBOLS AND ABBREVIATION	**VIII**
1	**INTRODUCTION**	1
	1.1 GENERAL	1
	1.2 TECHNOLOGIES USED	2
	1.2.1 OpenCV	2
	1.2.2 Face detection	4
	1.2.3 Face recognition	5
	1.2.4 React native app development	7
	1.3 OBJECTIVE	8
2	**LITERATURE REVIEW**	9
3	**SYSTEM ANALYSIS**	13
	3.1 EXISTING SYSTEM	13
	3.2 DISADVANTAGE OF EXISTING SYSTEM	13
	3.3 PROPOSED SYSTEM	13
	3.4 ADVANTAGE OF PROPOSED SYSTEM	14
	3.5 APPLICATIONS	14
4	**SYSTEM DESIGN**	15
	4.1 BLOCK DIAGRAM	15

	4.2 WORKING React native bridge	16
	4.3 MODULE DESCRIPTION	16
	4.3.1 Dataset collection	17
	4.3.2 Data Augmentation	18
	4.3.3 Face Detection Module	19
	4.3.4 Face Recognition Module	20
	4.3.5 Email Integration	22
	4.3.6 Mobile app development	24
5	**SOFTWARE USED**	26
	5.1 Microsoft visual studio	26
	5.2 Python	27
	5.3 Android studio	27
6	**RESULTS**	29
7	**CONCLUSION**	38
	REFERENCE	40
	APPENDIX I – SOURCE CODE	42
	APPENDIX II - SCREENSHOTS	57

LIST OF FIGURES

FIGURE NO	NAME OF THE FIGURE	PAGE NO.
1.1	OpenCV software architecture	4
1.2	Face detection methods	5
1.3	Template matching	6
1.4	Geometric face recognition	7
1.5	Photometric stereo for face recognition	7
4.1	Architecture Diagram	15
4.2	React native architecture diagram	16
4.3	Data augmentation	19
4.4	Ultralight face detector flow diagram	20
4.5	MobileFaceNet Architecture diagram	22
4.6	Email Integration	24
4.7	React native bridge	15
6.1	Dataset Collection	29
6.2	Environment setup	29
6.3	Face detection	30
6.4	Collection of faces from the dataset	30
6.5	Saved dataset	31
6.6	Face extraction	31
6.7	Embedding file	32
6.8	Face recognition initiation	32

6.9	Face recognition	32
6.10	Final script code	33
6.11	Login screen of mobile application.	33
6.12	Home page	34
6.13	Camera capture	34
6.14	Confirmation	35
6.15	Upload image	35
6.16	Attendance report	36
6.17	E-mail	36

LIST OF ABBREVIATIONS

S. NO	ABBREVIATION	EXPANSION
1.	CNN	Convolutional Neural Network
2.	OpenCV	Open COMPUTER VISION
3.	RNN	Recurrent Neural Network

CHAPTER 1
INTRODUCTION

1.1 Problem Statement

1.1 GENERAL:

Attendance Management System is a software developed for daily student attendance in schools, colleges and institutes. If it facilitates to access the attendance information of a particular student in a particular class. The information is sorted by the operators, which will be provided by the teacher for a particular class. This system will also help in evaluating attendance eligibility criteria of a student. The purpose of developing attendance management systems is to computerize the traditional way of taking attendance. Another purpose for developing this software is to generate the report automatically at the end of the session or in the between of the session

The system being developed is economic with respect to School or Colleges point of view. It is cost effective in the sense that it eliminated the paper work completely. The system is also time effective because the calculations are automated which are made at the end of the month or as per the user requirement. The result obtained contains minimum errors and is highly accurate as the data is required. The technical requirement for the system is economic and it does not use any other additional Hardware and software. The system working is quite easy to use and learn due to its simple but attractive interface. Users require no special training for operating the system. Working of Present System In the present system all work is done on paper. The whole session attendance is stored in the register and at the end of the session the reports are generated. We are not interested in generating reports in the middle of the session or as per the requirement because it takes more time in calculation. At the end of the session the students who don't have 75% attendance get a notice.

Disadvantages of present working system

The existing system is not user friendly because the retrieval of data is very slow and data is not maintained efficiently. We require more calculations to generate the report so it is generated at the end of the session. And the students do not get a single chance to improve their attendance.

All calculations to generate reports are done manually so there is greater chance of errors. Existing system requires a lot of paperwork. Loss of even a single register/record led to difficult situations because all the papers are needed to generate the reports. Every work is done manually so we cannot generate reports in the middle of the session or as per the requirement because it is very time consuming.

Characteristic of Attendance Management System

The proposed system is user friendly because the retrieval and storing of data is fast and data is maintained efficiently. More over the graphical user interface is provided in the proposed system, which provides users to deal with the system very easily. The reports can be easily generated in the proposed system so users can generate the report as per the requirement (monthly) or in the middle of the session. Users can give the notice to the students so he/she becomes regular. The proposed system requires very less paper work. All the data is feted into the computer immediately and reports can be generated through computers. Moreover, work becomes very easy because there is no need to keep data on papers. Computer operator control will be there so no chance of errors. Moreover, storing and retrieving information is easy. So, work can be done speedily and in time.

1.2 TECHNOLOGIES USED:

1.2.1 OpenCV

OpenCV was started at Intel in 1999 by Gary Bradsky and the first release came out in 2000. Vadim Pisarevsky joined Gary Bradsky to manage Intel's Russian software OpenCV team. In 2005, OpenCV was used on Stanley, the vehicle who won 2005 DARPA Grand Challenge. Later its active development continued under the support of Willow Garage, with Gary Bradsky and Vadim Pisarevsky leading the project. Right now, OpenCV supports a lot of algorithms related to Computer Vision and Machine Learning and it is expanding day-by-day.

Currently OpenCV supports a wide variety of programming languages like C++, Python, Java etc and is available on different platforms including Windows, Linux, OS X, Android, iOS etc. Also, interfaces based on CUDA and OpenCL are also under active development for

high-speed GPU operations.OpenCV-Python is the Python API of OpenCV. It combines the best qualities of OpenCV C++ API and Python language.

OpenCV is the leading open source library for computer vision, image processing and machine learning, and now features GPU acceleration for real-time operation.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. OpenCV was designed for computational efficiency and with a strong focus on real-time applications. Written in optimized C/C++, the library can take advantage of multi-core processing. Adopted all around the world, OpenCV has more than 47 thousand people in the user community and an estimated number of downloads exceeding 6 million. Usage ranges from interactive art, to mines inspection, stitching maps on the web or through advanced robotics.

OpenCV-Python

Python is a general purpose programming language started by Guido van Rossum**,** which became very popular in short time mainly because of its simplicity and code readability. It enables the programmer to express his ideas in fewer lines of code without reducing any readability.Compared to other languages like C/C++, Python is slower. But another important feature of Python is that it can be easily extended with C/C++. This feature helps us to write computationally intensive codes in C/C++ and create a Python wrapper for it so that we can use these wrappers as Python modules. This gives us two advantages: first, our code is as fast as the original C/C++ code (since it is the actual C++ code working in background) and second, it is very easy to code in Python. This is how OpenCV-Python works, it is a Python wrapper around original C++ implementation. And the support of Numpy makes the task easier. Numpy is a highly optimized library for numerical operations. It gives a MATLAB-style syntax. All the OpenCV array structures are converted to-and-from Numpy arrays. So whatever operations you can do in Numpy, you can combine it with OpenCV, which increases the number of weapons in your arsenal. Besides that, several other libraries like SciPy, Matplotlib which supports Numpy can be used with this. So OpenCV-Python is an appropriate tool for fast prototyping of computer vision problems.

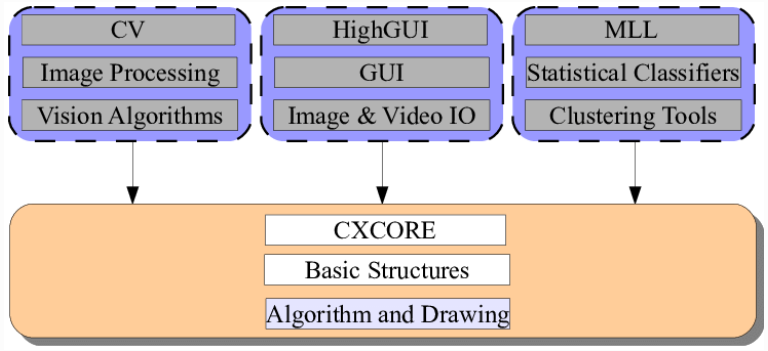


Figure 1.1 OpenCV software architecture

OpenCV Applications

OpenCV is being used for a very wide range of applications which include:

- Street view image stitching
- Automated inspection and surveillance
- Robot and driver-less car navigation and control
- Medical image analysis
- Video/image search and retrieval
- Movies - 3D structure from motion
- Interactive art installations

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*).

1.2.2 Face detection

Face detection is usually the first step towards many face-related technologies, such as face recognition or verification. However, face detection itself can have very useful applications. The most successful applications of face detection would probably be photo taking. When you take a photo of your friends, the face detection algorithm built into your digital camera detects where the faces are and adjusts the focus accordingly.

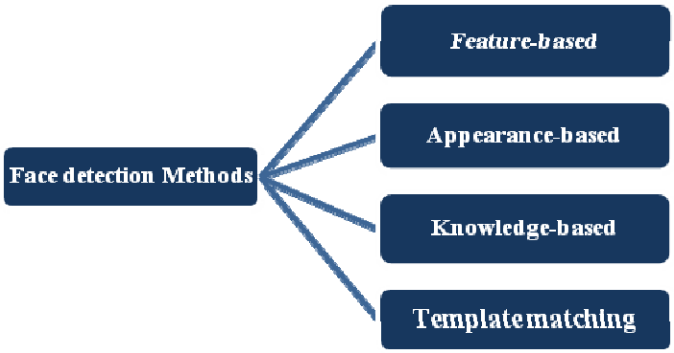


Figure 1.2 Face detection methods

Knowledge-Based

The knowledge-based method depends on the set of rules, and it is based on human knowledge to detect the faces. Ex- A face must have a nose, eyes, and mouth within certain distances and positions with each other. The big problem with these methods is the difficulty in building an appropriate set of rules. There could be many false positive if the rules were too general or too detailed. This approach alone is insufficient and unable to find many faces in multiple images.

Feature-Based

The feature-based method is to locate faces by extracting structural features of the face. It is first trained as a classifier and then used to differentiate between facial and non-facial regions. The idea is to overcome the limits of the instinctive knowledge of faces. This approach divided into several steps and even photos with many faces that reported a success rate of 94%.

Template Matching

Template Matching method uses pre-defined or parameterized face templates to locate or detect the faces by the correlation between the templates and input images. Ex- a human face can be divided into eyes, face contour, nose, and mouth. Also, a face model can be built by edges just by using edge detection methods. This approach is simple to implement, but it is inadequate for face detection. However, deformable templates have been proposed to deal with these problems.

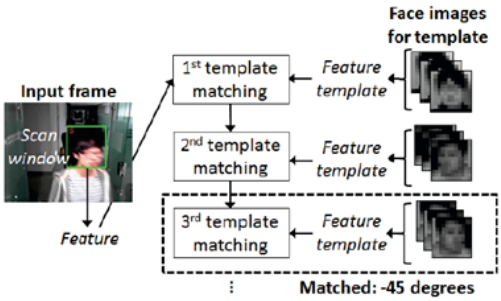


Figure 1.3:Template Matching

Appearance-Based

The appearance-based method depends on a set of delegate training face images to find out face models. The appearance-based approach is better than other ways of performance. In general, appearance-based methods rely on techniques from statistical analysis and machine learning to find the relevant characteristics of face images. This method is also used in feature extraction for face recognition.

1.2.3 Face recognition

Face recognition is the task of identifying an already detected object as a known or unknown face. Often the problem of face recognition is confused with the problem of face detection.Face Recognition on the other hand is to decide if the "face" is someone known, or unknown, using for this purpose a database of faces in order to validate this input face.

Different approaches of face recognition

There are two predominant approaches to the face recognition problem:
Geometric (feature based) and photometric (view based). As researcher interest in face recognition continued, many different algorithms were developed, three of which have been well studied in face recognition literature. Recognition algorithms can be divided into two main approaches:

1. Geometric: Is based on geometrical relationship between facial landmarks, or in other words the spatial configuration of facial features. That means that the main geometrical features of the face such as the eyes, nose and mouth are first located and then faces are classified on the basis of various geometrical distances and angles between features.

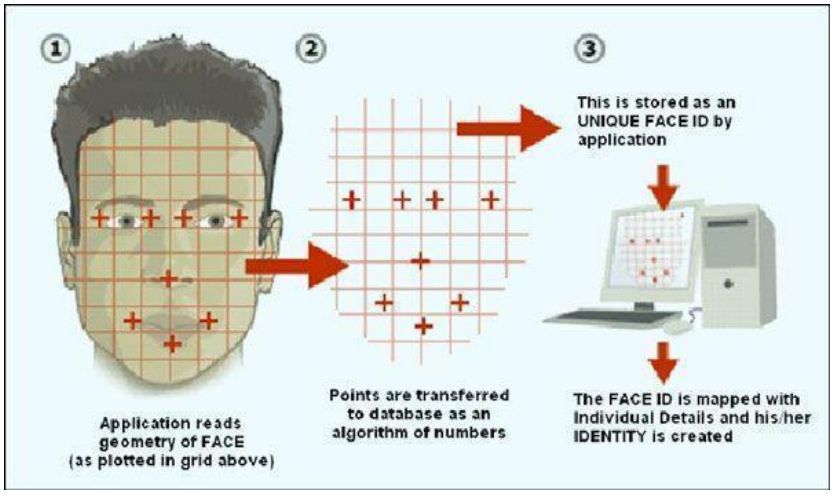


Figure 1.4 Geometric face recognition

2. Photometric stereo: Used to recover the shape of an object from a number of images taken under different lighting conditions. The shape of the recovered object is defined by a gradient map, which is made up of an array of surface normal.

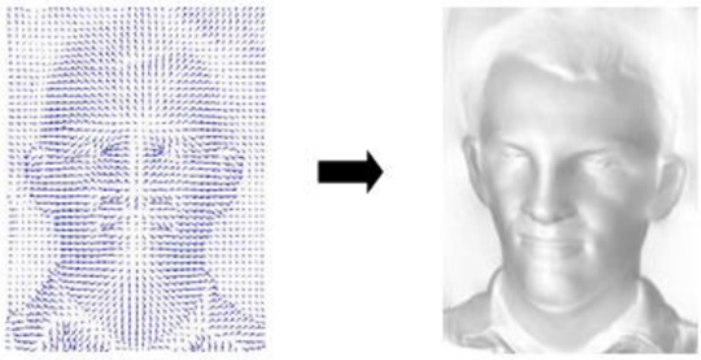


Figure 1.5 Photometric stereo for face recognition

1.2.4 React native app development:

Cross-platform development has become a great alternative to fully native mobile app development. Following the native mobile development approach, you create separate apps for Android and iOS. The cross-platform development allows you to cut expenses and save time by using the same code across both platforms. The React Native framework is a rising mobile solution and is considered the future of cross-platform mobile app development. React Native is an open-source framework that allows to build a mobile app with only JavaScript. It was introduced by Jordan Walke, a Facebook software engineer, as a new technology for simpler

development and a better user experience. The main distinctive of this framework is that React Native apps function just like native apps. They don't differ from apps built on Java, Objective-C or Swift and they use the same UI building blocks as native iOS or Android apps. But with React Native, building a mobile app is much faster and less expensive.

React Native is an open-source platform. That means all documentation related to this technology is open for everyone and is available for free to everyone in the React Native community. There's a great advantage to using a community-driven technology. For example, if you face any issue related to React Native development, anyone can get help from community experts or find information online. This is probably the most important advantage of React Native. With the convenience of this framework, developers don't need to build a separate mobile app for each platform, as almost 90 percent of React Native code can be reused between iOS and Android. React Native is all about the mobile UI. Comparing this framework to AngularJS or MeteorJS, it looks more like a JavaScript library than a framework. It's important to create a sequence of actions when building a mobile app, and React Native makes an implementing order just perfect. In addition, user interfaces designed in React Native are more responsive, decrease load time, and provide a smoother feel. The React Native framework is still advancing, so it might lack some components in the core framework. To fill this gap, React Native provides two types of third-party plugins: native modules and JavaScript modules.

1.3 OBJECTIVE

- To design a smart attendance system from a group photo.
- To send automatic report to the staff through mail depending on the number of present and absent students
- To design and develop a mobile application to receive notifications and capture a group photo.

CHAPTER 2

LITERATURE SURVEY

2.1 Jingxiao Zheng , Rajeev Ranjan, Ching-Hui Chen , Jun-Cheng Chen, Carlos D. Castillo, and Rama Chellappa," An Automatic System for Unconstrained Video-Based Face Recognition", VOL. 2, NO. 3, JULY 2020

Although deep learning approaches have achieved performance surpassing humans for still image-based face recognition, unconstrained video-based face recognition is still a
challenging task due to large volume of data to be processed and intra/inter-video variations on pose, illumination, occlusion, scene, blur, video quality, etc. In this work, we consider challenging scenarios for unconstrained video-based face recognition from multiple-shot videos and surveillance videos with low-quality frames. To handle these problems, we propose a robust and efficient system for unconstrained video-based face recognition,
which is composed of modules for face/fiducial detection, face association, and face recognition. First, we use multi-scale singleshot face detectors to efficiently localize faces in videos. The detected faces are then grouped through carefully designed face association methods, especially for multi-shot videos. Finally, the faces are recognized by the proposed face matcher based on an unsupervised subspace learning approach and a subspace-to subspace similarity metric. Extensive experiments on challenging video datasets, such as Multiple Biometric Grand Challenge (MBGC), Face and Ocular Challenge Series (FOCS), IARPA Janus Surveillance Video Benchmark (IJB-S) for low-quality surveillance videos and IARPA JANUS Benchmark B (IJB-B) for multiple-shot videos, demonstrate that the proposed system can accurately detect and associate faces from unconstrained videos and effectively learn robust and discriminative features for recognition.

Pros:

Face recognition is much faster in identification compared to other methods.

Cons:

A constant surveillance camera is required for this method to become a reality, thus making this project too expensive to run.

2.2 Xiaojuan Cheng, Jiwen Lu, Senior Member, IEEE, Bo Yuan, Member, IEEE, and Jie Zhou, Senior Member, IEEE "Face Segmentor-Enhanced Deep Feature Learning for Face Recognition" 2019 IEEE Transaction.

A Face Segmentor-Enhanced Network (FSENet) for face recognition to exploit facial localized property. Most existing methods emphasize the holistic characteristics of entire face images, which have limited discriminative ability due to large intra-class variations and inter-class fine-grain. To address this, we present a face segmentor to parse the face into local components and explore their internal correlations, which strengthens the discriminability to discern identities. Specifically, we introduce a semantic parsing module to assign each pixel with a semantic part label. This module generates a set of parsing maps, where each of them represents the pixel-wise occurrence probability of a certain facial component. We then segment facial regions masked by the parsing maps to achieve local features. We further build the structure correlation of facial part features to boost personalized attributes. We finally incorporate holistic and local information to improve the discriminative power of the face descriptor. Extensive experiments on popular public-domain datasets including labeled Face in the Wild (LFW), youtube Faces (YTF), IARPA IJB-A, IJB-B and IJB-C, and the MegaFace Challenge show that our method achieves promising performance.

Pros:

This method achieves promising performance.

Cons:

The method only focuses on the performance side and not on the attendance module alone.

2.3 Lei Zhang, Senior Member, IEEE, Ji Liu, Bob Zhang, Member, IEEE, David Zhang, Fellow, IEEE, Ce Zhu, Fellow, IEEE "Deep Cascade Model based Face Recognition: When Deep-layered Learning Meets Small Data" IEEE transactions on image processing, VOL. X, NO. X, AUG 2019.

Deep cascade model (DCM) based on SRC and NMR with hierarchical learning, nonlinear transformation and multi-layer structure for corrupted face recognition. The contributions include four aspects. First, an end-to-end deep cascade model for small-scale data without back-propagation is proposed. Second, a multi-level pyramid structure is integrated for local feature representation. Third, for introducing nonlinear transformation in layer-wise learning, soft max vector coding of the errors with class discrimination is proposed. Fourth, the existing representation methods can be easily integrated into our DCM framework. Experiments on a number of small-scale benchmark FR datasets demonstrate the superiority of the proposed model over state-of-the-art counterparts. Additionally, a perspective that deep-layered learning does not have to be convolutional neural network with back-propagation optimization is consolidated.

Pros:

Experiments on a number of small-scale benchmark FR datasets demonstrate the superiority of the proposed model over state-of-the-art counterparts.

Cons:

The project results are not reliable, since the face recognition output can easily be corrupted.

2.4 Yuqi Zhang, Yongzhen Huang, Senior Member, IEEE, Shiqi Yu, and Liang Wang, Fellow, IEEE "Cross-view Gait Recognition by Discriminative Feature Learning". Citation information: DOI 10.1109/TIP.2019.2926208, IEEE Transactions on Image Processing.

A robust, effective and gait-related loss function, called angle center loss (ACL), is proposed to learn discriminative gait features. The proposed loss function is robust to different local parts and temporal window sizes. Different from center loss which learns a center for each identity, the proposed loss function learns multiple sub-centers for each angle of the same identity. Only the largest distance between the anchor feature and the corresponding cross view sub-centers is penalized, which achieves better intra-subject compactness. We also propose to extract discriminative spatial temporal features by local feature extractors and a temporal attention model. A simplified spatial transformer network is proposed to localize the suitable horizontal parts of the human body. Local gait features for each horizontal part are extracted and then concatenated as the descriptor. We introduce long-short term memory (LSTM) units as the temporal attention model to learn the attention score for each frame, e.g., focusing more on discriminative frames and less on frames with bad quality. The temporal attention model shows better performance than the temporal average pooling or gait energy images (GEI). By combining the three aspects, we achieve the state-of-the-art results on several cross-view gait recognition benchmarks.

Pros:

The temporal attention model shows better performance than the temporal average pooling or gait energy images (GEI).

Cons:

The LSTM requires a lot of resources and **time** to get trained and become ready for real-world applications.

2.5 Christian Galea, Student Member, IEEE, and Reuben A. Farrugia, Member, IEEE "Matching Software-Generated Sketches to Face Photos with a Very Deep CNN, Morphed Faces, and Transfer Learning" IEEE transactions on information forensics and security, 2017.

Sketches obtained from eyewitness descriptions of criminals have proven to be useful in apprehending criminals, particularly when there is a lack of evidence. A very deep convolutional neural network is utilized to determine the identity of a subject in a composite sketch by comparing it to face photos, and is trained by applying transfer learning to a state of-the-art model pre-trained for face photo

recognition. 3D morphable model is used to synthesis both photos and sketches to augment the available training data, an approach that is shown to significantly aid performance, and the UoM-SGFS database is extended to contain twice the number of subjects, now having 1200 sketches of 600 subjects. An extensive evaluation of popular and state-of-the-art algorithms is also performed due to the lack of such information in literature, where it is demonstrated that the proposed approach comprehensively outperforms state-of-the-art methods on all publicly available composite sketch datasets.

Pros:

It is demonstrated that the proposed approach comprehensively outperforms state-of-the-art methods on all publicly available composite sketch datasets.

Cons:

The 3D model requires a lot of memory space for processing

2.6 Wasserstein CNN: Learning Invariant Features for NIR-VIS Face Recognition.

Extensive experiments using three challenging NIR-VIS face recognition databases demonstrate the superiority of the WCNN method over state-of-the-art methods. The novel Wasserstein convolutional neural network (WCNN) approach for learning invariant features between near-infrared (NIR) and visual (VIS) face images (i.e., NIR-VIS face recognition). The low-level layers of the WCNN are trained with widely available face images in the VIS spectrum, and the high-level layer is divided into three parts: the NIR layer, the VIS layer and the NIR-VIS shared layer. The first two layers aim at learning modality-specific features, and the NIR-VIS shared layer is designed to learn a modality-invariant feature subspace. W-CNN learning is performed to minimize the Wasserstein distance between the NIR distribution and the VIS distribution for invariant deep feature representations of heterogeneous face images. To avoid the over-fitting problem on small-scale heterogeneous face data, a correlation prior is introduced on the fully-connected WCNN layers to reduce the size of the parameter space. This prior is implemented by a low-rank constraint in an end-to-end network. The joint formulation leads to an alternating minimization for deep feature representation at the training stage and an efficient computation for heterogeneous data at the testing stage.

Pros:

It is implemented by a low-rank constraint in an end-to-end network.

Cons:

This model cannot be used for recognizing faces from infrared images.

2.7 Jiaojiao Zhao, Jungong Han, and Ling Shao, Senior Member IEEE, "Unconstrained Face Recognition Using A Set-to-Set Distance Measure on Deep Learned Features" Citation information: DOI 10.1109/TCSVT.2017.2710120, IEEE Transactions on Circuits and Systems for Video Technology IEEE transactions on circuits and systems for video technology.

A novel Set-to-Set (S2S) distance measure to calculate the similarity between two sets with the aim to improve the accuracy of face recognition in real-world situations such as extreme poses or severe illumination conditions. Our S2S distance adopts the CNN-average pooling for the similarity scores computed on all the media in two sets, making the identification far less susceptible to the poor representations (outliers) than traditional feature-average pooling and score-average pooling. This allows us to choose the appropriate metric depending on the recognition task in order to achieve the best results. To evaluate the proposed S2S distance, we conduct extensive experiments on the challenging set-based IJB-A face dataset, which demonstrate that our algorithm achieves the state-of-the-art results clearly superior to the base lines including several deep learning based face recognition algorithms.

Pros:

Algorithm achieves state-of-the-art results clearly superior to the base lines including several deep learning-based face recognition algorithms.

Cons:

The algorithm is very slow in performance.

2.8 Jiwen Lu, Gang Wang, Weihong Deng, and Jie Zhou," Simultaneous Feature and Dictionary Learning for Image Set Based Face Recognition" Citation information: DOI 10.1109/TIP.2017.2713940, IEEE Transactions on Image Processing.

A SFDL method to learn discriminative features and dictionaries simultaneously from raw face pixels so that discriminative information from facial image sets can be jointly exploited by a one-stage learning procedure. To better exploit the nonlinearity of face samples from different image sets, we propose a deep SFDL (D-SFDL) method by jointly learning hierarchical non-linear transformations and class-specific dictionaries to further improve the recognition performance. Extensive experimental results on five widely used face datasets clearly show that our SFDL and D-SFDL achieve very competitive or even better performance with the state-of-the-arts.

Pros:

Extensive experimental results on five widely used face datasets clearly show that our SFDL and D-SFDL achieve very competitive or even better performance with the state-of-the-arts.

Cons:

The SFDL (D-SFDL) method used is very expensive in terms of time.

2.9 Gaurav Goswami, Student Member, IEEE, Mayank Vatsa, Senior Member, IEEE, and Richa Singh, Senior Member, IEEE," Face Verification via Learned Representation on Feature-Rich Video Frames" Citation information: DOI 10.1109/TIFS.2017.2668221, IEEE Transactions on Information Forensics and Security.

Abundance and availability of video capture devices such as mobile phones and surveillance cameras has instigated research in video face recognition which is highly pertinent in law enforcement applications. While the current approaches have reported high accuracies at equal error rates, performance at lower false accept rates requires significant improvement. Frame selection is followed by representation learning based feature extraction where three contributions are presented: (i) deep learning architecture which is a combination of stacked denoising sparse autoencoder(SDAE)and deep Boltzmann machine (DBM), (ii) formulation for joint representation in an auto encoder, and (iii) updating the loss function of DBM by including sparse and low rank regularization. Finally, a multilayer neural network is used as a classifier to obtain the verification decision.

Pros:

Begins the face recognition using a video sequence.

Cons:

The performance at lower false accept rates requires significant improvement.

2.10 Jiwen Lu, Senior Member, IEEE, Junlin Hu, and Yap-Peng Tan, Senior Member, IEEE "Discriminative Deep Metric Learning for Face and Kinship Verification" Citation information: DOI 10.1109/TIP.2017.2717505, IEEE Transactions on Image Processing.

A new discriminative deep metric learning (DDML) method for face and kinship verification in wild conditions. DDML method to train a deep neural network to learn a set of hierarchical nonlinear transformations to project face pairs into the same latent feature space, under which the distance of each positive pair is reduced and that of each negative pair is enlarged, respectively. To better use the commonality of multiple feature descriptors to make all the features more robust for face and kinship verification, we develop a discriminative deep multi-metric learning (DDMML) method to jointly learn multiple neural networks under which the correlation of different features of each sample is maximized, and the distance of each positive pair is reduced and that of each negative pair is enlarged, respectively.

Pros:

The algorithm jointly learns multiple neural networks under which the correlation of different features of each sample is maximized.

Cons:

The deep metric learning (DDML) method cannot handle more than 2 samples at a time.

CHAPTER 3

SYSTEM ANALYSIS

3.1 EXISTING SYSTEM

Although deep learning approaches have achieved performance surpassing humans for still image-based face recognition, unconstrained video-based face recognition is still a challenging task due to large volume of data to be processed and intra/inter-video variations on pose, illumination, occlusion, scene, blur, video quality, etc. In this work, we consider challenging scenarios for unconstrained video-based face recognition from multiple-shot videos and surveillance videos with low-quality frames. To handle these problems, we propose a robust and efficient system for unconstrained video-based face recognition, which is composed of modules for face/fiducial detection, face association, and face recognition. First, we use multi-scale single-shot face detectors to efficiently localize faces in videos. The detected faces are then grouped through carefully designed face association methods, especially for multi-shot videos. Finally, the faces are recognized by the proposed face matcher based on an unsupervised subspace learning approach and a subspace-to subspace similarity metric.

3.1.1 DISADVANTAGES OF EXISTING SYSTEM:

- Only the presence of faces in image or video is recognized.
- It doesn’t recognise whose face appeared in the image or video.
- The algorithm used is not efficient, and takes longer to recognize.

3.2 PROPOSED SYSTEM:

Attendance management system is a necessary tool for taking attendance in any environment where attendance is critical. However, most of the existing approaches are time consuming, intrusive and require manual work from the users. Face recognition is an important application of Image processing owing to its use in many fields. Identification of individuals in an organization for the purpose of attendance is one such application of face recognition. Maintenance and monitoring of attendance records plays a vital role in the analysis of performance of any organization. The purpose of developing attendance management systems is to computerize the traditional way of taking attendance. Automated Attendance Management System performs the daily activities of attendance marking and analysis with reduced human intervention. In this project we will be designing a single shot attendance system mobile

application which will capture the group of faces of the student and check the presence of the student face in the database and mark it as present. As the attendance time closes an automatic report is generated and sent to the staff's registered mail id. Thus, this project helps in saving time of the people as well as automates the records which helps in saving manual maintenance. Implementation of this project also gives rise to many applications such as biometric scanning in bank lockers, etc.

3.2.1 ADVANTAGES:

- This project helps in saving time of the people as well as automates the records which helps in saving manual maintenance.
- Automatic report to the staff leading them to maintain the record of the class attendance.
- Automating the whole attendance system.
- Productive use of time in student development
- Easy to use in real time

3.2.2 APPLICATIONS:

- Used in schools and colleges.
- Used in MNC Companies.
- Used in where ever the attendance system is required

CHAPTER: 4

SYSTEM DESIGN

DETAILED DESIGN OF THE PROJECT:

This chapter describes the overall and the detailed architectural design. It also describes each module that is to be implemented along with a Data Flow diagram.

4.1 SYSTEM ARCHITECTURE

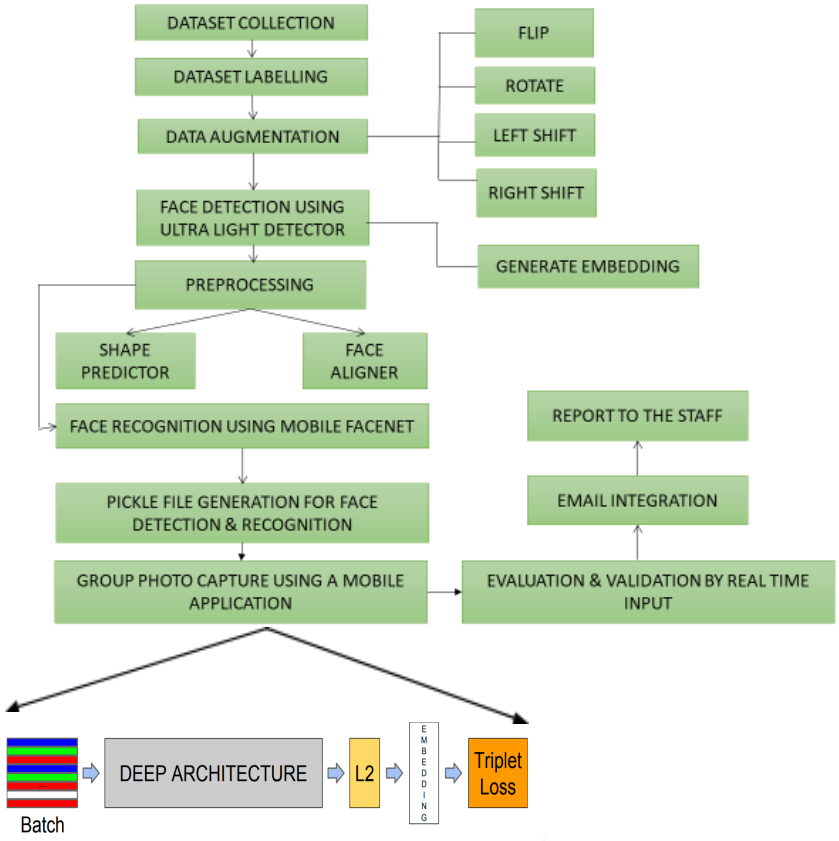


Figure 4.1 : Architecture Diagram

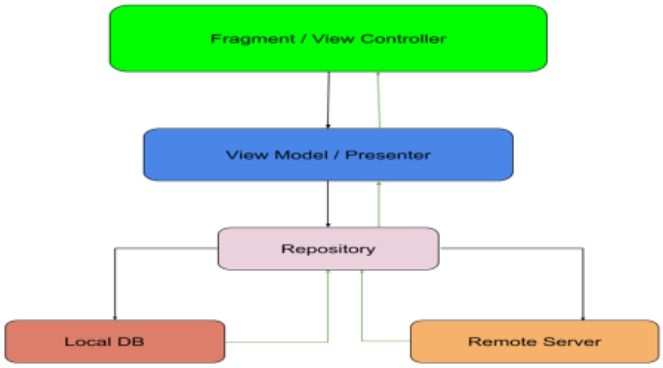


Figure 4.2 : React native Architecture Diagram

4.2 WORKING:

First the Intel Processor has been configured using the Linux OS to work on. The necessary libraries needed for the project are installed using the terminal command window which is used to run all the commands of the project as well as run the project. The datasets for the student attendance is collected by taking the images of the students. Now the datasets are trained by the Intel Processor using the deep learning algorithm to identify the correct face and mark the student as present or absent automatically. Now a group photo of the students is taken, the faces of the students in the photo are identified and extracted using a mobile application. The MobileFaceNet and Ultralight face detector algorithms are used for face recognition. As the student's face matches with any of the faces in the database, the student is marked present. And the process goes on until the attendance time closes. In absence of a student, automatically a report of the students absent are sent through mail to the class teacher. Thus, this project helps in effective implementation of automatic attendance system using the latest face recognition technology as well as saves time of the institution.

4.3 Modules Description:

- Dataset collection
- Data augmentation
- Face detection module
- Face Recognition Module
- Email Integration

- Database integration Module
- Mobile app development

4.3.1 Dataset Collection

A data set is a collection of data. Deep Learning has become the go-to method for solving many challenging real-world problems. It's definitely by far the best performing method for computer vision tasks. The image above showcases the power of deep learning for computer vision. With enough training, a deep network can segment and identify the "key points" of every person in the image. These deep learning machines that have been working so well need lots of fuel; that fuel is data. The more labelled data available, the better our model performs. The idea of more data leading to better performance has even been explored at a large-scale by Google with a dataset of 300 Million images! When deploying a Deep Learning model in a real-world application**,** data must be constantly fed to continue improving its performance. And, in the deep learning era, data is very well arguably the most valuable resource. There are three steps of collecting data

Scraping From the Web

Manually finding and downloading images takes a long time simply due to the amount of human work involved. The task probably has some kind of common objects that are to be detected. And so that becomes the keyword for web-scraping. It also becomes the class name for that object. Every single pixel in the image is required. To get those, it's best to use some really great image annotation tools that are already out there. Given a rough set of polygon points around an object, can generate the pixel labels for segmentation. Deep extreme cut is also quite similar except they use only the four extreme points around the object. This will then give some nice bounding box and segmentation labels. Another option is to use an existing image annotation GUIs.

Third-party:

Since data has become such a valuable commodity in the deep learning era, many start-ups have started to offer their own image annotation services they'll gather and label the data. Given a description of what kind of data and annotations needed. Mighty is one that has been doing self-driving car image annotation and has become pretty big in the space were

at CVPR 2018 too. Payment AI are less specialized than Mighty AI, offering image annotation for any domain.

4.3.2 Data Augmentation

The performance of deep learning neural networks often improves with the amount of data available. Data augmentation is a technique to artificially create new training data from existing training data. This is done by applying domain-specific techniques to examples from the training data that create new and different training examples. Image data augmentation is perhaps the most well-known type of data augmentation and involves creating transformed versions of images in the training dataset that belong to the same class as the original image. Transforms include a range of operations from the field of image manipulation, such as shifts, flips, zooms, and much more. The intent is to expand the training dataset with new, plausible examples. This means, variations of the training set images that are likely to be seen by the model. For example, a horizontal flip of a picture of a cat may make sense, because the photo could have been taken from the left or right. A vertical flip of the photo of a cat does not make sense and would probably not be appropriate given that the model is very unlikely to see a photo of an upside-down cat.

As such, it is clear that the choice of the specific data augmentation techniques used for a training dataset must be chosen carefully and within the context of the training dataset and knowledge of the problem domain. In addition, it can be useful to experiment with data augmentation methods in isolation and in concert to see if they result in a measurable improvement to model performance, perhaps with a small prototype dataset, model, and training run. Modern deep learning algorithms, such as the convolutional neural network, or CNN, can learn features that are invariant to their location in the image. Nevertheless, augmentation can further aid in this transform invariant approach to learning and can aid the model in learning features that are also invariant to transforms such as left-to-right to top-to-bottom ordering, light levels in photographs, and more. Image data augmentation is typically only applied to the training dataset, and not to the validation or test dataset. This is different from data preparation such as image resizing and pixel scaling; they must be performed consistently across all datasets that interact with the model.

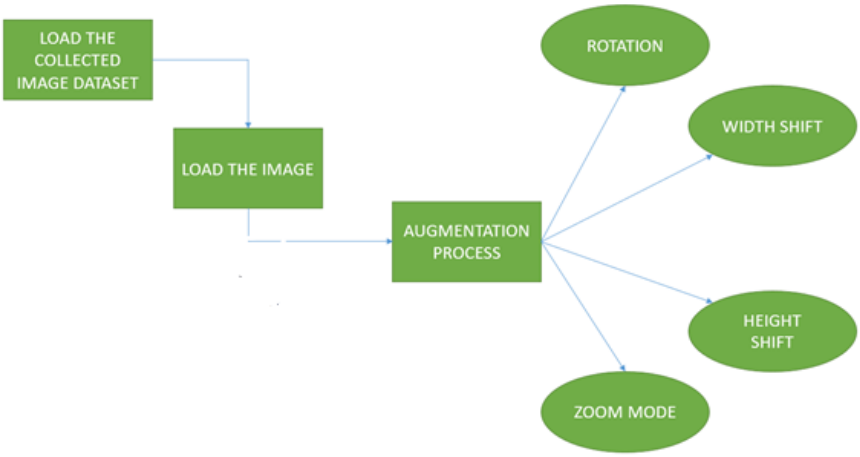


Figure 4.3 Data augmentation

4.3.3 Face Detection Module

The "Ultra-Light-Fast-Generic-Face-Detector" is designed for general-purpose face detection applications in low-power computing devices and is applicable to both Android and iOS phones as well as PCs (CPU and GPU). The model is a real-time ultra-lightweight universal face detection model designed for edge computing devices or low-power devices. It can be used in low-power computing devices such as ARM for real-time common scene faces. Facial recognition technology is widely applied in security monitoring, surveillance, human-computer interaction, entertainment, etc. Detecting human faces in digital images is the first step in facial recognition, and an ideal face detection model can be evaluated by how quickly and accurately it performs.

The Face-Detector-1MB stands out in terms of speed — the model's default FP32 precision (.pth) file size is 1.1MB, and the inference frame int8 is quantized to a size of 300KB. In terms of model calculation, the input resolution of 320×240 is only about 90 to 109 MFlops. The Face-Detector-1MB training process used a VOC dataset generated by the WIDER FACE dataset, a face detection benchmark. WIDER FACE was released in 2015 and consists of 32,203 images and 393,703 face bounding boxes with a high degree of variability in scale, pose, expression, occlusion and illumination. The 1MB lightweight model comes in a version-slim with slightly faster simplification, and a version-RFB with a modified RFB module for higher precision. The model was tested on Ubuntu16.04, Windows 10, Python3.6, Pytorch1.2, CUDA10.0, etc.

Features

- In terms of model size, the default FP32 precision (pth) file size is 1.04~1.1MB, and the inference frame int8 is about 300KB.
- In the calculation of the model, the input resolution of 320x240 is about 90~109 M-Flops.
- There are two versions of the model, version-slim (slightly faster simplification), version-RFB (with the modified RFB module, higher precision).
- Provides pre-training models using wider face training at 320x240 and 640x480 different input resolutions to better work in different application scenarios.
- Support for on xx export, easy to transplant.

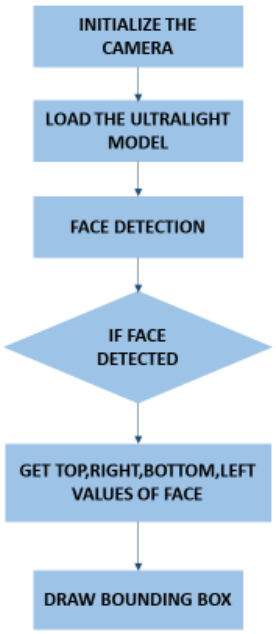


Figure 4.4 Ultralight face detector flow diagram

4.3.4 Face Recognition Module

For recognizing the face mobile facenet has been used which is more accurate in classifying faces. MobileFaceNet is a neural network and obtains accuracy up to 99.28 percent on labelled faces in the wild (LFW) dataset, and a 93.05 percent accuracy on recognizing faces in the AgeDB dataset. The network used around a million parameters taking only 24 milliseconds to run and produce results on a Qualcomm Snapdragon processor. On comparing this

performance to accuracies of 98.70 percent and 89.27 percent for ShuffleNet, which has many more parameters and takes a little longer to execute on the CPU. The researchers have made it easy to replace the global average pooling layer in the CNN with a depth wise convolution layer, which improves performance on facial recognition. This development is really important as the artificial intelligence world searches for efficient models that run on small compute powers which are available on today's mobile phones.

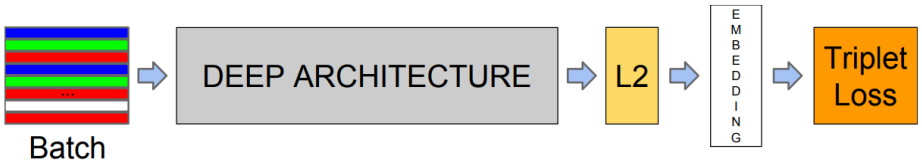
Another approach for obtaining lightweight facial verification models is by compressing pretrained networks by knowledge distillation. Such approaches have achieved 97.32 percent facial verification accuracy on LFW with 4.0 MB model size. The remarkable achievement is that MobileFaceNets achieves comparable accuracy with a very small budget. MobileFaceNet architecture is partly inspired by the MobileNetV2 architecture. The residual bottlenecks proposed in MobileNetV2 are used as our main building blocks. The researchers use PReLU as the non-linearity, which is better suited for facial verification than using ReLU. The researchers also use a fast down sampling strategy at the beginning of the network, and a linear 1×1 convolution layer following a linear global depth wise convolution layer as the feature output layer. The detailed architecture is mentioned in the table below:

Input	Operator	*t*	*c*	*n*	*s*
$112^2 \times 3$	conv3x3	-	64	1	2
$56^2 \times 64$	depthwise conv3x3	-	64	1	1
$56^2 \times 64$	bottleneck	2	64	5	2
$28^2 \times 64$	bottleneck	4	128	1	2
$14^2 \times 128$	bottleneck	2	128	6	1
$14^2 \times 128$	bottleneck	4	128	1	2
$7^2 \times 128$	bottleneck	2	128	2	1
$7^2 \times 128$	conv1x1	-	512	1	1
$7^2 \times 512$	linear GDConv7x7	-	512	1	1
$1^2 \times 512$	linear conv1x1	-	128	1	1

Table 4.2: MobileFaceNet architecture

The primary MobileFaceNet network uses 0.99 million parameters. To reduce computational cost, the researchers decided to change input resolution from 112×112 to 112×96 or 96×96. The linear 1×1 convolution layer after the linear GD Conv layer was also removed from MobileFaceNet. This gives a resulting network called MobileFaceNet M. The researchers have used MobileNetV1, ShuffleNet, and MobileNetV2 as the baseline models. All MobileFaceNet models and baseline models are trained on CASIA-Webface dataset from scratch

by ArcFace loss, for a fair performance comparison among them. The training is finished at 60K iterations. To pursue further excellent performance, MobileFaceNet, MobileFaceNet (112×96), and MobileFaceNet (96×96) are also trained on the cleaned training set of MSCeleb-1M database with 3.8 million images from 85,000 subjects. The accuracy of our primary MobileFaceNet is boosted to 99.55 percent and 96.07 percent on LFW and AgeDB-30, respectively.



4.5 MobileFaceNet Architecture diagram

4.3.5 EMAIL INTEGRATION

For integrating E-mail, we will be using the SMTP protocol which is used for sending and receiving mail as configured. The work flow of that module can be seen as follows. Intel Processor is a small sized computer mainly designed for education purpose. The company launched Intel Processor-3 model in February 2016, with inbuilt WIFI, Bluetooth and USB boot capabilities. Due to its small size and affordable price, it is quickly adopted by makers and electronics enthusiasts for projects. The Intel Processor is slower than laptop or desktop but is still a complete Linux computer. Intel Processor generally comes with installed Raspbian OS.

With the vision of IoT (internet of things) Intel Processor is the powerful tool. A number of IoT projects using Intel Processor have been developed. It can also be used with many IoT cloud platform like IFTTT, ThingSpeak, Artik Cloud, Firebase and Particle. One of the application of raspberry-pi/intel-processor is to use SMTP (Simple mail transfer protocol) for sending and receiving emails**.** In this project, an email will be sent via SMTP server using raspberry pi/Intel Processor**.** SMTP works by starting a session between the user and server, whereas MTA (Mail Transfer Agent) and MDA (Mail Delivery Agent) provide domain searching and local delivery services. SMTP is the standard protocol for providing email services on a TCP/IP network. This server provides the ability to receive and send email messages.

SMTP is an application-layer protocol that provides the delivery and transmission of email over the Internet. It is maintained by the Internet Engineering Task Force (IETF). SMTP is generally summed within an email client application and is composed of four key components:

1. Local user or client-end utility known as the mail user agent (MUA)
2. Server known as mail submission agent (MSA)
3. Mail transfer agent (MTA)
4. Mail delivery agent (MDA)

Getting email alerts or a set of data using Intel Processor python program is a very useful application. All that is needed is smtplib library in the python script. There are many versions of python but pi is more compatible with 3.2 and 2.7 versions of it. Below are the mentioned steps of sending SMTP email using pi:

Steps for Sending Email using Intel Processor

Step 1: - Setting up the Intel Processor module- connect the power cable and LAN cable to Intel Processor then create WIFI hotspot and connect with it.

Step 2: - After then open the terminal window on Pi. Then, open the putty software and paste the host name or IP address.

Step 3: - We need to update the Intel Processor. So, install the latest packages by using the below command.

Step 4: - Then use the following command – echo “hello” | mail –s “test” xyz@gmail.com

This command specifies the content, subject of our mail, as well as the mail id to which our mail will be delivered.

Step 5: - Then we need to create a new file in the python and this can be done by using the following command- nano newmailing.py

Alternate way to do the same step

Open the Python IDE 2.7 or above 3.2, create a new file and save it as newmailing.py by pressing Ctrl + x. Here, newmailing.py is the name given by the user while saving the file.

Step 6:- Allowing Gmail SMTP Access for Accounts with Standard Authentication

To allow access to Gmail’s SMTP server from your app

Step 7:- Login to your Gmail account and check the mail, if everything works correctly then a mail will be delivered to your mail id.

An overall flow of the working can be seen in the figure below

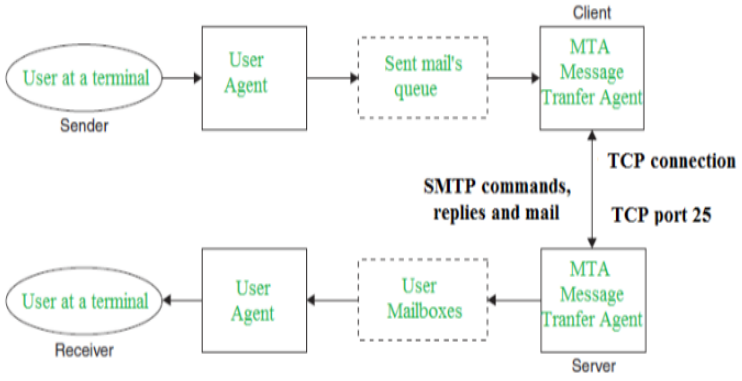


Figure 4.6: Email Integration

4.3.6 Mobile app development

In this project we have used react native for mobile application development. A native mobile app is developed as an interface using a javascript framework react native. The mobile app enables the user to look for available slots for parking in the preferred location. It also enables the user to book the parking slot of his/her location. React Native (also known as RN) is a popular JavaScript-based mobile app framework that allows you to build natively-rendered mobile apps for iOS and Android. The framework lets you create an application for various platforms by using the same codebase. React Native was first released by Facebook as an open-source project in 2015. In just a couple of years, it became one of the top solutions used for mobile development. React Native development is used to power some of the world's leading mobile apps, including Instagram, Facebook, and Skype. We discuss these and other examples of React Native-powered apps further in this post.

There are several reasons behind React Native's global success.

Firstly, by using React Native, companies can create code just once and use it to power both their iOS and Android apps. This translates to huge time and resource savings.

Secondly, React Native was built based on React – a JavaScript library, which was already hugely popular when the mobile framework was released. We discuss the differences between React and React Native in detail further in this section. Thirdly, the framework empowered frontend developers, who could previously only work with web-based technologies, to create robust, production-ready apps for mobile platforms. React Native is written with a mixture of JavaScript and JXL, a special mark-up code of XML. The framework has the ability to communicate with both realms – JavaScript-based threads and existent, native app threads.

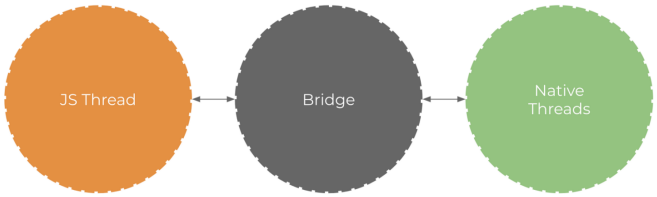


Figure 4.7: React native bridge

React Native uses a so-called “bridge”. While JavaScript and Native threads are written in completely different languages, it’s the bridge feature that makes bidirectional communication possible.

CHAPTER 5

SOFTWARE DESCRIPTION

5.1 VISUAL STUDIO

In this project the Microsoft visual studio is used as an IDE. Visual Studio Code combines the simplicity of a source code editor with powerful developer tooling, like IntelliSense code completion and debugging. First and foremost, it is an editor that gets out of your way. The delightfully frictionless edit-build-debug cycle means less time fiddling with your environment, and more time executing on your ideas. Visual Studio Code supports macOS, Linux, and Windows - so you can hit the ground running, no matter the platform.

At its heart, Visual Studio Code features a lightning fast source code editor, perfect for day-to-day use. With support for hundreds of languages, VS Code helps you be instantly productive with syntax highlighting, bracket-matching, auto-indentation, box-selection, snippets, and more. Intuitive keyboard shortcuts, easy customization and community-contributed keyboard shortcut mappings let you navigate your code with ease. For serious coding, you'll often benefit from tools with more code understanding than just blocks of text. Visual Studio Code includes built-in support for IntelliSense code completion, rich semantic code understanding and navigation, and code refactoring. And when the coding gets tough, the tough get debugging. Debugging is often the one feature that developers miss most in a leaner coding experience, so we made it happen. Visual Studio Code includes an interactive debugger, so you can step through source code, inspect variables, view call stacks, and execute commands in the console.

VS Code also integrates with build and scripting tools to perform common tasks making everyday workflows faster. VS Code has support for Git so you can work with source control without leaving the editor including viewing pending changes. Customize every feature to your liking and install any number of third-party extensions. While most scenarios work "out of the box" with no configuration, VS Code also grows with you, and we encourage you to optimize your experience to suit your unique needs. VS Code includes enriched built-in support for Node.js development with JavaScript and TypeScript, powered by the same underlying technologies that drive Visual Studio. VS Code also includes great tooling for web technologies such as JSX/React, HTML, CSS, SCSS, Less, and JSON.

Architecturally, Visual Studio Code combines the best of web, native, and language-specific technologies. Using Electron, VS Code combines web technologies such as JavaScript and Node.js with the speed and flexibility of native apps. VS Code uses a newer, faster version of the same industrial-strength HTML-based editor that has powered the "Monaco" cloud editor, Internet Explorer's F12 Tools, and other projects. Additionally, VS Code uses a tools service architecture that enables it to integrate with many of the same technologies that power Visual Studio, including Roslyn for .NET, TypeScript, the Visual Studio debugging engine, and more. Visual Studio Code includes a public extensibility model that lets developers build and use extensions, and richly customize their edit-build-debug experience.

5.2 Python:

In this project, python is used as the programming language. In technical terms, Python is an object-oriented, high-level programming language with integrated dynamic semantics primarily for web and app development. It is extremely attractive in the field of Rapid Application Development because it offers dynamic typing and dynamic binding options. Python is relatively simple, so it's easy to learn since it requires a unique syntax that focuses on readability. Developers can read and translate Python code much easier than other languages. In turn, this reduces the cost of program maintenance and development because it allows teams to work collaboratively without significant language and experience barriers. Additionally, Python supports the use of modules and packages, which means that programs can be designed in a modular style and code can be reused across a variety of projects. One of the most promising benefits of Python is that both the standard library and the interpreter are available free of charge, in both binary and source form. There is no exclusivity either, as Python and all the necessary tools are available on all major platforms. Therefore, it is an enticing option for developers who don't want to worry about paying high development costs.

5.3 Android Studio:

In this project android studio is used for mobile application development. Android Studio is the official integrated development environment (IDE) for Google's Android operating system, built on JetBrains IntelliJ IDEA software and designed specifically for Android development. It is available for download on Windows, macOS and Linux based operating systems or as a

subscription-based service in 2020. It is a replacement for the Eclipse Android Development Tools (E-ADT) as the primary IDE for native Android application development. To write some code for your Android application, some kind of environment or code editor that will make the task easy. Android Studio is an Integrated Development Environment (IDE) for Android app development. It is based on Intelligence IDEA. To enhance productivity while building Android application, Android Studio provides a number of features, They are

1. Instant run on the mobile device or any virtual device to check the working of the application
2. Extensive testing tools that can be used for better testing before launching the application on the Play Store
3. Flexible Gradle-based build system
4. Code auto-complete.

New project templates have been updated to use Material Design Components. **Integrated emulator window**: The emulator can directly be run in the IDE instead of on its own, separate window. **Dagger/Hilt code navigation**: Clicking on the new gutter actions to find out more about Dagger and Hilt types in your code. **ML Model Binding**: Studio can generate code for an imported TensorFlow Lite model which makes it easier to interact with that model from the app code.

CHAPTER 6

RESULTS

6.1 OUTPUTS OBTAINED

To begin with, by testing the trained model, we can split our project into modules of implementation. Dataset collection involves the process of collecting different video files of people. Various datasets were collected and one example among the collected dataset can be found below

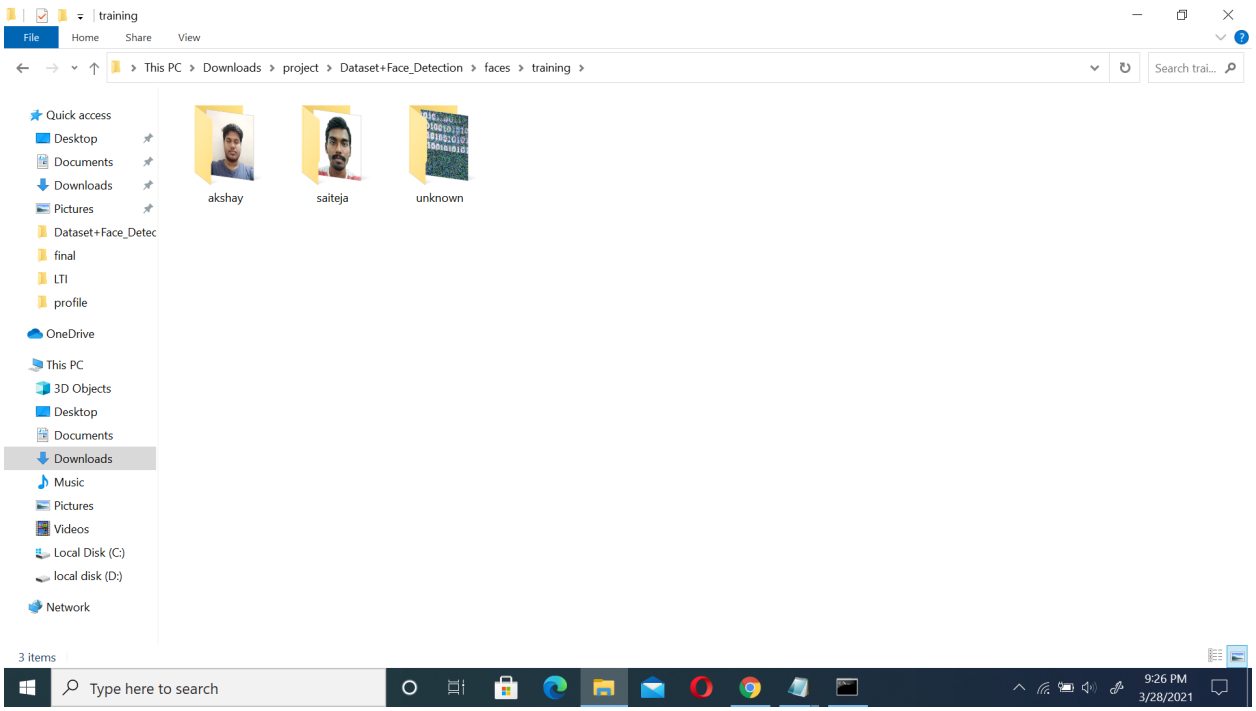


Figure 6.1 Dataset Collection

The environment and setup of python on which the scripts are being executed and the below screenshot shows the environment and setup to run the script:

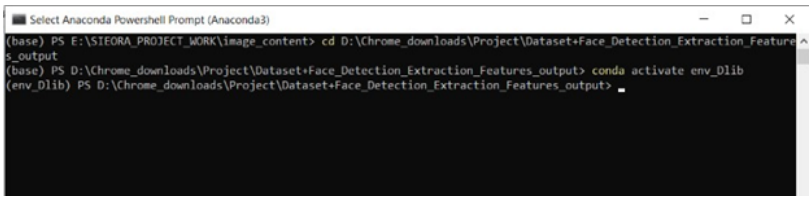


Figure 6.2 Environment setup

The next step is face detection, in this process the faces are detected from the dataset collected as video files of the students. An example of the detected face can be found below

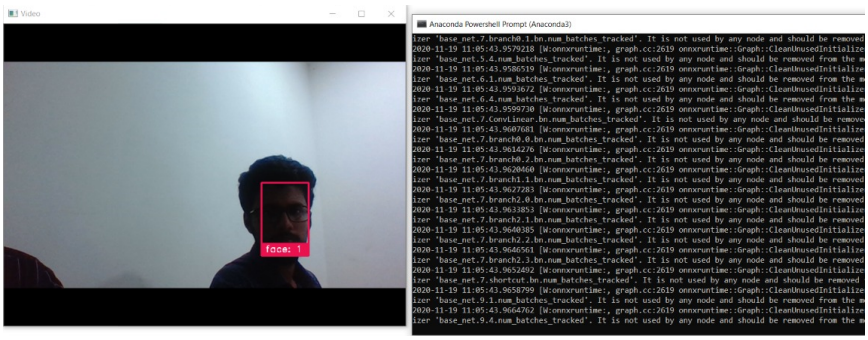


Figure 6.3 Face detection

In the next step, the faces are detected from each and every video file from the dataset that is collected. The below image shows the collection of faces from the dataset.

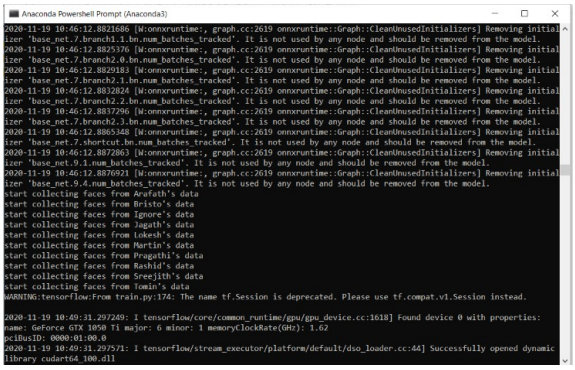


Figure 6.4 Collection of faces from the dataset

In the next step all the faces that are detected from the dataset are collected and saved as shown below

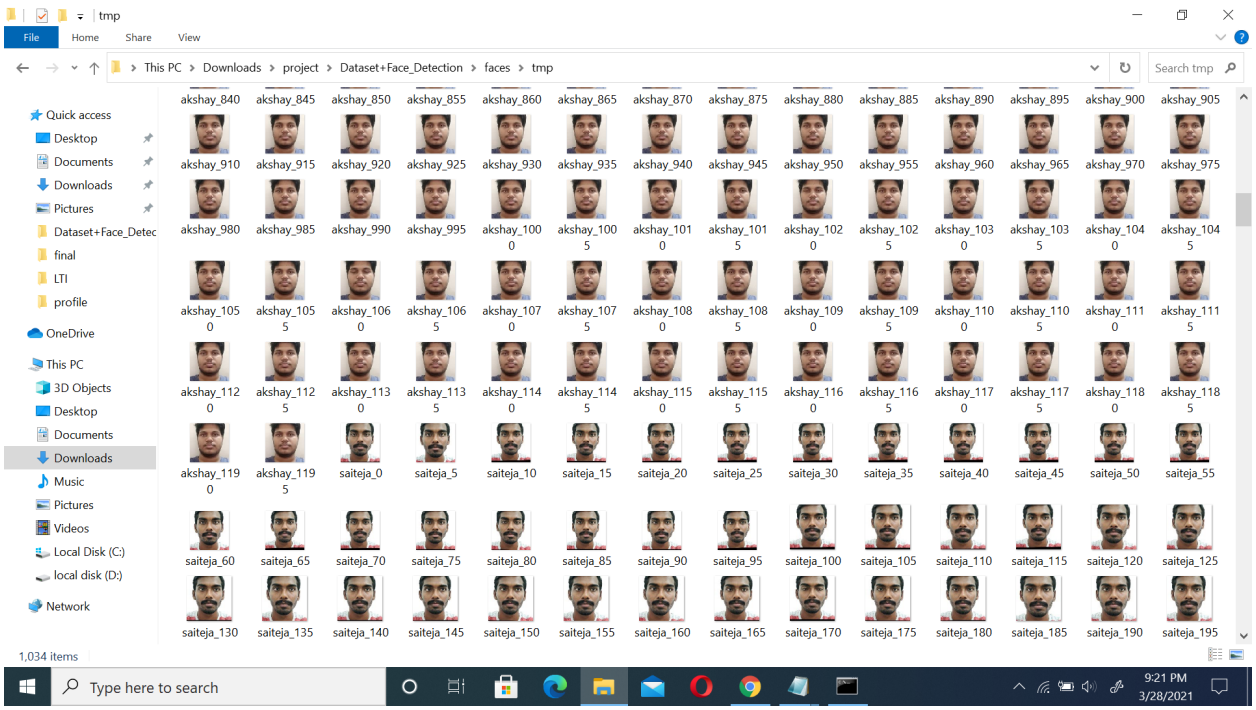


Figure 6.5: Saved dataset

In the next step, the saved faces undergo a feature extraction process. Feature extraction is the process of extracting features from the saved faces. The below image shows the initiation of face extraction.

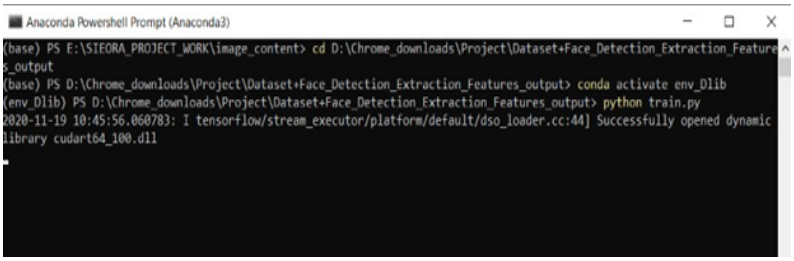


Figure 6.6: Face extraction

An embedding file is generated which consists of the features extracted from the face extraction. The below image shows the embedding file generated after feature extraction of all the faces collected.

Figure 6.7: Embedding file

The next step is face recognition, in this process the face given as input is recognized. The initiation of the face recognition module is shown below

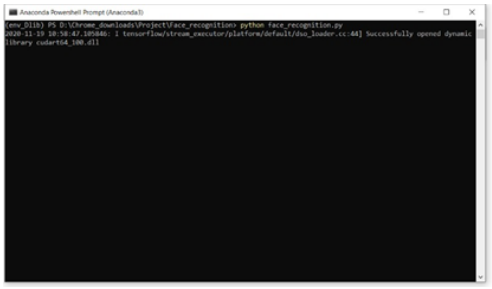


Figure 6.8: Face recognition initiation

The image shows the face recognition of a student face successfully

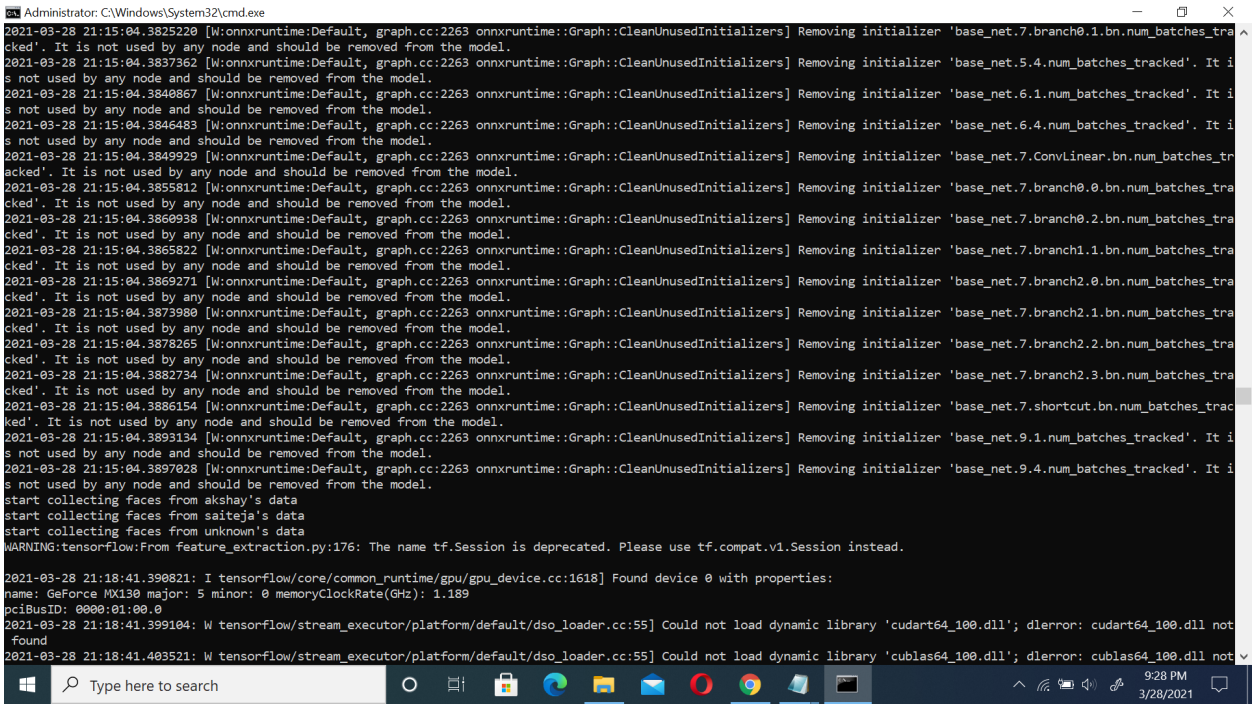


Figure 6.9:Face recognition

The script for the group attendance is initiated. The below shown script is responsible for all the functionality of the attendance system including the API s .

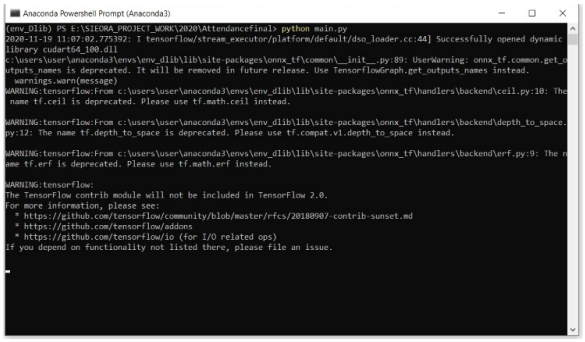


Figure 6.10: Final script code

A mobile application using react native is developed to capture the group photo and notify the user as present. The below image shows the login page of the mobile application

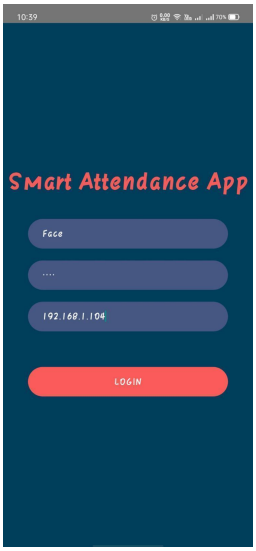


Figure 6.11 Login screen of mobile application.

On successful login into the mobile application the user views the home page of the mobile application. On clicking start the camera is initiated to capture the group photos. The below image shows the home page of the mobile application

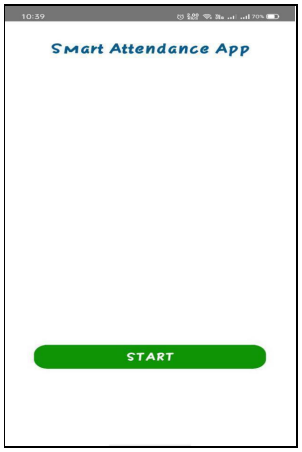


Figure 6.12 Home page

The below image shows the photo captured on the mobile application

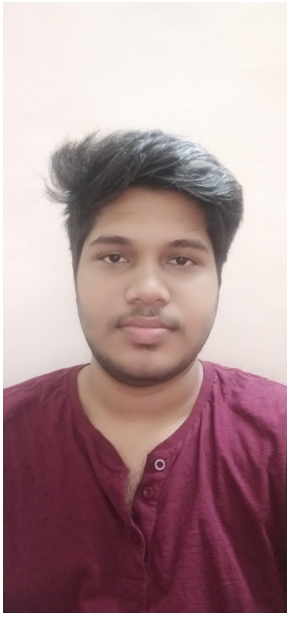


Figure 6.13 Camera capture

On clicking the submit button the image is sent to the server and the faces from the image are recognized and given present. The below image shows the confirmation request

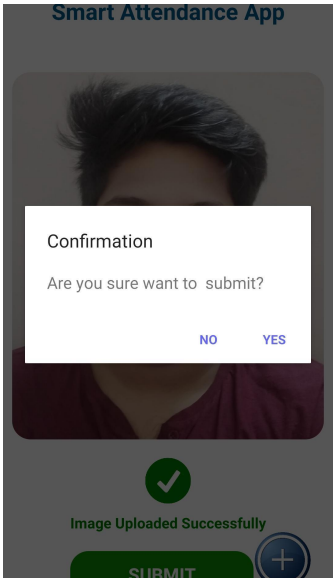


Figure 6.14 Confirmation

On successful processing of the image, the faces have been successfully detected and recognized giving them present. The below image notifies the user on successful upload.

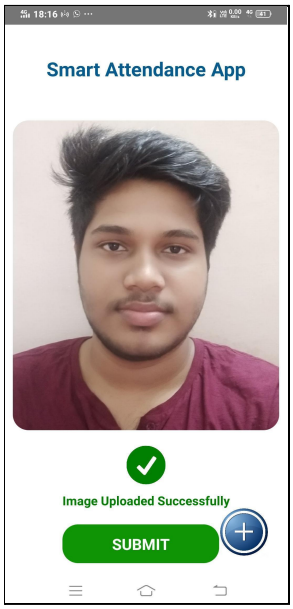


Figure 6.15 Upload image

Once the attendance is completed a report is generated mentioning the names of the students along with their status of attendance. The below image is an example of the attendance report.

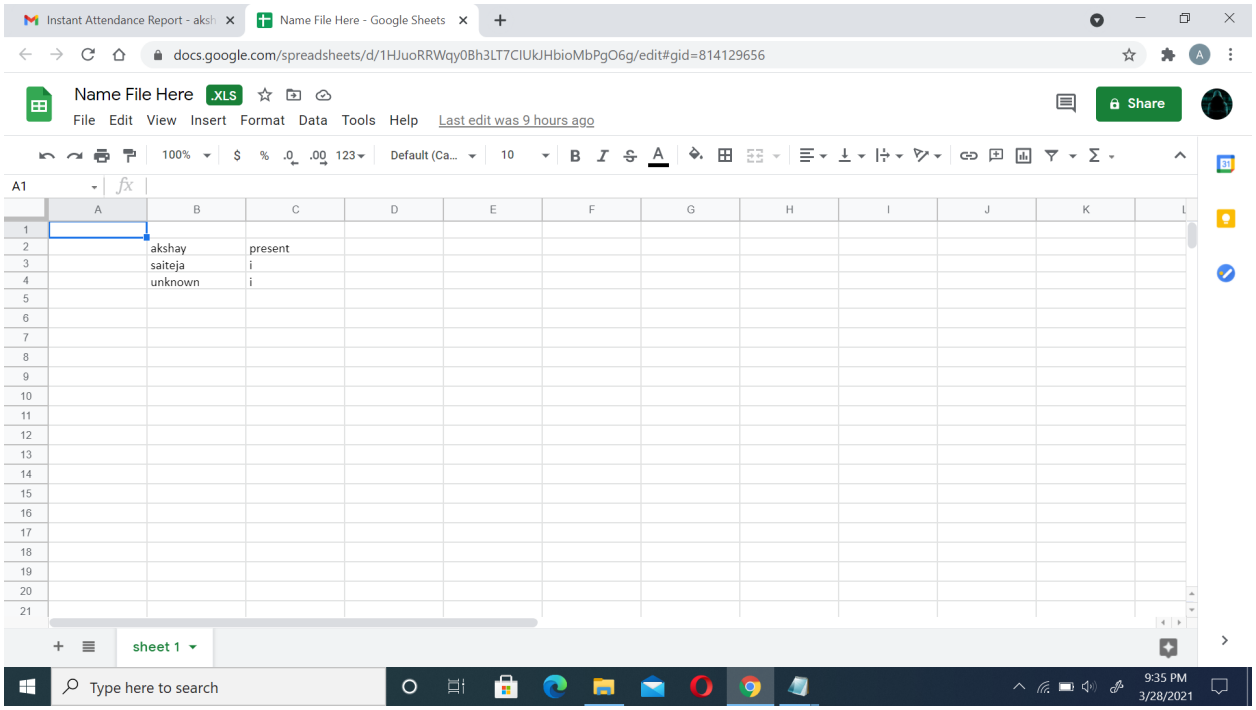


Figure 6.16 Attendance report

The teacher receives the report through Email. The below image shows the report that is sent through E-mail to the teacher.

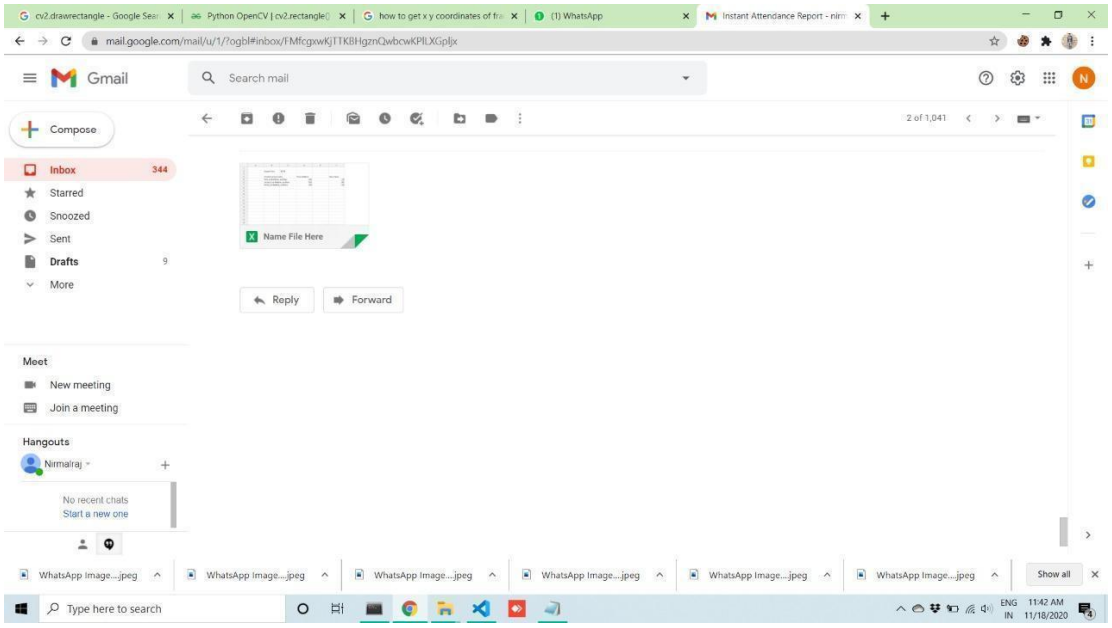


Figure 6.17 Report sent to mail

Thus, this project helps in automating the attendance system which will give rise to all the attendance management team. It will also save the staff time which he/she can be able to spend their time effectively.

6.2 COMPARISON BETWEEN EXISTING SYSTEM AND PROPOSED SYSTEM

In the existing system, the project aims to exploit the facial localized property making it very inefficient for implementation in real time applications. The system is very inherently complicated making it very unpredictable. The system does not detect or recognize a face in real time. The proposed system, overcomes all of the disadvantages in the existing system by the successful implementation of a mobile application developed using react native. The mobile application is used to capture a group image of the students to verify from the image the students present and absent. An automatic mail is sent to the parent concerned and also a report is sent to the staff of the list of students absent. The proposed system makes use of highly effective face detection and recognition algorithms.

FEATURES	**EXISTING SYSTEM -**	**PROPOSED SYSTEM –**
Number of Faces detection (Given 150 faces)	70	140
Time taken for detection	3 seconds	200 milliseconds
Model Size	20MB	1.04~1.1MB
Efficiency	Slow and efficient	Fast and efficient
User friendly	The system is not user friendly.	The system uses a mobile application as an interface.

CHAPTER 7

CONCLUSION

In this project we will be designing an attendance system which will recognize the face of the student and check the presence of the student face in the database and mark it as present. By this project we save time and avoid manual work. And it determined the absent students then an automatic message will be sent to their parents then Automatic mail sent to the staff also the report will generate automatically. Thus, this project helps in monitoring the student whether they attend class or not effectively.

FUTURE WORK

In the coming future, we review the application of the student's smart attendance system technology and it can be promoted for MNC, teachers, or other private offices with more accuracy compared to this project there are more chances to develop or convert this project in many ways. Then provide automatic mail sending to the staff, message to parent and the attendance taking time will be saved by this project.

REFERENCE

[1] André Stuhlsatz, Jens Lippel, and Thomas Zielke "Feature Extraction with Deep Neural Networks by a Generalized Discriminant Analysis" IEEE transactions on neural networks and learning systems, vol. 23, no. 4, april 2020.

[2] Christian Galea, Student Member, IEEE, and Reuben A. Farrugia, Member, IEEE "Matching Software-Generated Sketches to Face Photos with a Very Deep CNN, Morphed Faces, and Transfer Learning" IEEE transactions on information forensics and security, 2019.

[3] Gaurav Goswami, Student Member, IEEE, Mayank Vatsa, Senior Member, IEEE, and Richa Singh, Senior Member, IEEE," Face Verification via Learned Representation on Feature-Rich Video Frames" Citation information: DOI 10.1109/TIFS.2017.2668221, IEEE Transactions on Information Forensics and Security.

[4] Giovani Chiachia, Alexandre X. Falcão, Member, IEEE, Nicolas Pinto, Anderson Rocha, Member, IEEE, and David Cox, Member, IEEE "Learning Person-Specific Representations From Faces in the Wild" IEEE transactions on information forensics and security, vol. 9, no. 12, december 2014.

[5] Jingxiao Zheng , Rajeev Ranjan, Ching-Hui Chen , Jun-Cheng Chen, Carlos D. Castillo, and Rama Chellappa," An Automatic System for Unconstrained Video-Based Face Recognition" VOL. 2, NO. 3, JULY 2020

[6] Jiaojiao Zhao, Jungong Han, and Ling Shao, Senior Member IEEE, "Unconstrained Face Recognition Using A Set-to-Set Distance Measure on Deep Learned Features" Citation information: DOI 10.1109/TCSVT.2017.2710120, IEEE Transactions on Circuits and Systems for Video Technology IEEE transactions on circuits and systems for video technology.

[7] Jiwen Lu, Gang Wang, Weihong Deng, and Jie Zhou," Simultaneous Feature and Dictionary Learning for Image Set Based Face Recognition" Citation information: DOI 10.1109/TIP.2017.2713940, IEEE Transactions on Image Processing.

[8] Jiwen Lu, Senior Member, IEEE, Junlin Hu, and Yap-Peng Tan, Senior Member, IEEE "Discriminative Deep Metric Learning for Face and Kinship Verification" Citation information: DOI 10.1109/TIP.2017.2717505, IEEE Transactions on Image Processing.

[9] Lei Zhang, Senior Member, IEEE, Ji Liu, Bob Zhang, Member, IEEE, David Zhang, Fellow, IEEE, Ce Zhu, Fellow, IEEE "Deep Cascade Model based Face Recognition: When Deep-layered Learning Meets Small Data" IEEE transactions on image processing, VOL. X, NO. X, AUG 2019.

[10] Ming Shao, Member, IEEE, Yizhe Zhang, Student Member, IEEE, and Yun Fu, Senior Member, IEEE," Collaborative Random Faces-Guided Encoders for Pose-Invariant Face Representation Learning" IEEE transactions on neural networks and learning systems.

[11] Ran He, Senior Member, IEEE, Xiang Wu, Zhenan Sun∗, Member, IEEE, and Tieniu Tan, Fellow, IEEE.” Wasserstein CNN: Learning Invariant Features for NIR-VIS Face Recognition” journal of latex class files, vol. 14, no. 8, august 2017.

[12] Tong Zhang, Wenming Zheng, Member, IEEE, Zhen Cui, Yuan Zong, Jingwei Yan and Keyu Yan,” A Deep Neural Network Driven Feature Learning Method for Multi-view Facial Expression Recognition” Citation information: DOI 10.1109/TMM.2016.2598092, IEEE Transactions on Multimedia.

[13] Thibaud Senechal, Member, IEEE, Vincent Rapp, Member, IEEE, Hanan Salam, Renaud Seguier, Kevin Bailly, and Lionel Prevost “Facial Action Recognition Combining Heterogeneous Features via Multi Kernel Learning” IEEE transactions on systems, man, and cybernetics—part b: cybernetics, vol. 42, no. 4, august 2012.

[14] Xiaojuan Cheng, Jiwen Lu, Senior Member, IEEE, Bo Yuan, Member, IEEE, and Jie Zhou, Senior Member, IEEE “Face Segmentor - Enhanced Deep Feature Learning for Face Recognition” 2019 IEEE Transaction.

[15] Yuqi Zhang, Yongzhen Huang, Senior Member, IEEE, Shiqi Yu, and Liang Wang, Fellow, IEEE “Cross-view Gait Recognition by Discriminative Feature Learning”. Citation information: DOI 10.1109/TIP.2019.2926208, IEEE Transactions on Image Processing.

APPENDIX I – SOURCE CODE

```
import argparse
import imutils
import pyttsx3
import pickle
import time
import cv2
import urllib.request
import urllib.parse

def sendSMS(apikey, numbers, sender, message):
        data = urllib.parse.urlencode({'username':username,'apikey': apikey, 'numbers': numbers,
'message' : message, 'sender': sender})
        data = data.encode('utf-8')
        fr = f.read()
        return(fr)
# construct the argument parser and parse the arguments
ap = argparse.ArgumentParser()
ap.add_argument("-c", "--conf", required=True,
        help="Path to the input configuration file")
args = vars(ap.parse_args())

# load the configuration file
conf = Conf(args["conf"])

# initialize consecutive recognition count to 0
consecCount = 0
student1 = "irfan"
student2 = "tahmina"
#student3 = "3"

unknown_flag=0
student1_flag=0
```

```
student2_flag=0
#student3_flag=0
process_flag = 0

def create_msg():
        msg = MIMEMultipart()
        msg['Subject'] = 'Class1'
        #msg['From'] = 'sieoraiot@gmail.com'
        #msg['To'] = '@mail.cc'

        text = MIMEText("Class1")
        msg.attach(text)
        return msg


def create_msg2():
        msg1 = MIMEMultipart()
        msg1['Subject'] = 'Class2'
        #msg['From'] = 'sieoraiot@gmail.com'
        #msg['To'] = '@mail.cc'

        text = MIMEText("Class2")
        msg1.attach(text)
        return msg1


def attach_file(msg_cont):

        f = open(file_path)
        image = MIMEText(f.read())
        msg_cont.attach(image)
        return msg_cont
def attach_file2(msg_cont):
        f = open(file_path2)
```

```
        image = MIMEText(f.read())
        msg_cont.attach(image)

        return msg_cont

def SendMail(msg):

        s = smtplib.SMTP(Server)
        s.ehlo()
        s.starttls()
        s.ehlo()
        s.login(UserName, UserPassword)
        s.sendmail("sieoraiot@gmail.com", "yadavaprasath@gmail.com", msg.as_string())
        s.quit()

# initialize the database, student table, and attendance table
# objects
db = TinyDB(conf["db_path"])
studentTable = db.table("student")
attendanceTable = db.table("attendance")

# load the actual face recognition model along with the label encoder
recognizer = pickle.loads(open(conf["recognizer_path"], "rb").read())
le = pickle.loads(open(conf["le_path"], "rb").read())

# initialize the video stream and allow the camera sensor to warmup
print("[INFO] warming up camera...")
vs = VideoStream(src=0).start()
#vs = VideoStream(usePiCamera=True).start()
time.sleep(2.0)

# initialize previous and current person to None
file_path = "a.txt"
file_path2 = "b.txt"
```

```
prevPerson = None
curPerson = None

# initialize consecutive recognition count to 0
#consecCount = 0

# initialize the text-to-speech engine, set the speech language, and
# the speech rate



# initialize a dictionary to store the student ID and the time at
# which their attendance was taken
#studentDict = {}




# loop over the frames from the video stream
while True:
        if process_flag==0:
                # store the current time and calculate the time difference
                # between the current time and the time for the class
                currentTime = datetime.now()
                timeDiff = (currentTime - datetime.strptime(conf["timing"],
                        "%H:%M")).seconds

                # grab the next frame from the stream, resize it and flip it
                # horizontally
                frame = vs.read()
                frame = imutils.resize(frame, width=400)
                frame = cv2.flip(frame, 1)

                # if the maximum time limit to record attendance has been crossed
                # then skip the attendance taking procedure
```

```
if timeDiff > conf["max_time_limit"]:
        # check if the student dictionary is not empty
        #if len(studentDict) != 0:
                # insert the attendance into the database and reset the
                # student dictionary
                #attendanceTable.insert({str(date.today()): studentDict})
                #studentDict = {}

        # draw info such as class, class timing, and current time on
        # the frame
        cv2.putText(frame, "Class: {}".format(conf["class"]),
                (10, 10), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 255), 1)
        cv2.putText(frame, "Class timing: {}".format(conf["timing"]),
                (10, 25), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 255), 1)
        cv2.putText(frame, "Current time: {}".format(
                currentTime.strftime("%H:%M:%S")), (10, 40),
                cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 255), 1)

        # show the frame
        cv2.imshow("Attendance System", frame)
        key = cv2.waitKey(1) & 0xFF

        # if the `q` key was pressed, break from the loop
        if key == ord("q"):
                break

        # skip the remaining steps since the time to take the
        # attendance has ended
        continue

# convert the frame from RGB (OpenCV ordering) to dlib
# ordering (RGB)
rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
```

```
# detect the (x, y)-coordinates of the bounding boxes
# corresponding to each face in the input image
boxes = face_recognition.face_locations(rgb,
        model=conf["detection_method"])

# loop over the face detections
for (top, right, bottom, left) in boxes:
        # draw the face detections on the frame
        cv2.rectangle(frame, (left, top), (right, bottom),
                (0, 255, 0), 2)

# calculate the time remaining for attendance to be taken
timeRemaining = conf["max_time_limit"] - timeDiff

# draw info such as class, class timing, current time, and
# remaining attendance time on the frame
cv2.putText(frame, "Class: {}".format(conf["class"]), (10, 10),
        cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 255), 1)
cv2.putText(frame, "Class timing: {}".format(conf["timing"]),
        (10, 25), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 255), 1)
cv2.putText(frame, "Current time: {}".format(
        currentTime.strftime("%H:%M:%S")), (10, 40),
        cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 255), 1)
cv2.putText(frame, "Time remaining: {}s".format(timeRemaining),
        (10, 55), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 255), 1)

# check if atleast one face has been detected
if len(boxes) > 0:
        # compute the facial embedding for the face
        encodings = face_recognition.face_encodings(rgb, boxes)

        preds = recognizer.predict_proba(encodings)[0]
        j = np.argmax(preds)
        curPerson = le.classes_[j]
```

```
                    # if the person recognized is the same as in the previous
                    # frame then increment the consecutive count
                    if prevPerson == curPerson:
                            consecCount += 1

                    # otherwise, these are two different people so reset the
                    # consecutive count
                    else:
                            consecCount = 0

                    # set current person to previous person for the next
                    # iteration
                    prevPerson = curPerson

                    # if a particular person is recognized for a given
                    # number of consecutive frames, we have reached a
                    # positive recognition and alert/greet the person accordingly
                    if consecCount >= conf["consec_count"]:
                            # check if the student's attendance has been already
                            # taken, if not, record the student's attendance
                            #if curPerson not in studentDict.keys():
                                    #studentDict[curPerson]                                  =
datetime.now().strftime("%H:%M:%S")

                                    # get the student's name from the database and let them
                                    # know that their attendance has been taken
                            name = studentTable.search(where(
                                            curPerson))[0][curPerson][0]


                            # construct a label saying the student has their attendance
                            # taken and draw it on to the frame
                            label = "{}, you are now marked as present in {}".format(
```

```
                    name, conf["class"])
                cv2.putText(frame, label, (5, 175),
                    cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255, 0, 0), 2)

                if name == student1:
                    student1_flag=1
                    consecCount = 0
                elif name == student2:
                    student2_flag=1
                    consecCount = 0

                elif name == 'unknown':
                    if unknown_flag == 0:
                        unknown_flag=1

                        if unknown_flag ==1:

                            #resp = sendSMS(apikey, numbers,sender,
message)

                            #print(resp)
                            unknown_flag=2
            else:
                # construct a label asking the student to stand in fron
                # to the camera and draw it on to the frame
                label = "Please stand in front of the camera"
                cv2.putText(frame, label, (5, 175),
                    cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255, 0, 0), 2)

        # show the frame
        cv2.imshow("Survilance System", frame)
        key = cv2.waitKey(1) & 0xFF

        # check if the `q` key was pressed
        if key == ord("q"):
```

```
                # check if the student dictionary is not empty, and if so,
                # insert the attendance into the database
                if len(studentDict) != 0:
                        attendanceTable.insert({str(date.today()): studentDict})

                # break from the loop
                break

        if timeDiff2 == conf["max_time_limit"]:

                f = open("log.txt", "a+")
                if student1_flag == 1:
                        f.write("\n" +  student1 +" "+ "is present")
                else:
                        f.write("\n"+ student1 +" "+"is absent")
                if student2_flag == 1:
                        f.write("\n"+ student2 +" "+ "is present")
                else:
                        f.write("\n"+ student2 +" "+"is absent")

                f.close()
                msg_head = create_msg()
                attach=attach_file(msg_head)
                #SendMail(attach)

                process_flag=1
                student1_flag=0
                student2_flag=0
elif process_flag==1:

        # store the current time and calculate the time difference
        # between the current time and the time for the class
        currentTime = datetime.now()
        timeDiff2 = (currentTime - datetime.strptime(conf["timing2"],
```

```
            "%H:%M")).seconds

# grab the next frame from the stream, resize it and flip it
# horizontally
frame = vs.read()
frame = imutils.resize(frame, width=400)
frame = cv2.flip(frame, 1)

# if the maximum time limit to record attendance has been crossed
# then skip the attendance taking procedure
if timeDiff2 > conf["max_time_limit2"]:
        # check if the student dictionary is not empty
        #if len(studentDict) != 0:
                # insert the attendance into the database and reset the
                # student dictionary
                #attendanceTable.insert({str(date.today()): studentDict})
                #studentDict = {}

        # draw info such as class, class timing, and current time on
        # the frame
        cv2.putText(frame, "Class: {}".format(conf["class2"]),
                (10, 10), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 255), 1)
        cv2.putText(frame, "Class timing: {}".format(conf["timing2"]),
                (10, 25), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 255), 1)
        cv2.putText(frame, "Current time: {}".format(
                currentTime.strftime("%H:%M:%S")), (10, 40),
                cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 255), 1)

        # show the frame
        cv2.imshow("Attendance System", frame)
        key = cv2.waitKey(1) & 0xFF

        # if the `q` key was pressed, break from the loop
        if key == ord("q"):
```

```
                break

        # skip the remaining steps since the time to take the
        # attendance has ended
        continue

# convert the frame from RGB (OpenCV ordering) to dlib
# ordering (RGB)
rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)

# detect the (x, y)-coordinates of the bounding boxes
# corresponding to each face in the input image
boxes = face_recognition.face_locations(rgb,
        model=conf["detection_method"])

# loop over the face detections
for (top, right, bottom, left) in boxes:
        # draw the face detections on the frame
        cv2.rectangle(frame, (left, top), (right, bottom),
                (0, 255, 0), 2)

# calculate the time remaining for attendance to be taken
timeRemaining = conf["max_time_limit2"] - timeDiff2

# draw info such as class, class timing, current time, and
# remaining attendance time on the frame
cv2.putText(frame, "Class: {}".format(conf["class2"]), (10, 10),
        cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 255), 1)
cv2.putText(frame, "Class timing: {}".format(conf["timing2"]),
        (10, 25), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 255), 1)
cv2.putText(frame, "Current time: {}".format(
        currentTime.strftime("%H:%M:%S")), (10, 40),
        cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 255), 1)
cv2.putText(frame, "Time remaining: {}s".format(timeRemaining),
```

```
                (10, 55), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 255), 1)

        # check if atleast one face has been detected
        if len(boxes) > 0:
                # compute the facial embedding for the face
                encodings = face_recognition.face_encodings(rgb, boxes)

                preds = recognizer.predict_proba(encodings)[0]
                j = np.argmax(preds)
                curPerson = le.classes_[j]

                # if the person recognized is the same as in the previous
                # frame then increment the consecutive count
                if prevPerson == curPerson:
                        consecCount += 1

                # otherwise, these are two different people so reset the
                # consecutive count
                else:
                        consecCount = 0

                # set current person to previous person for the next
                # iteration
                prevPerson = curPerson

                # if a particular person is recognized for a given
                # number of consecutive frames, we have reached a
                # positive recognition and alert/greet the person accordingly
                if consecCount >= conf["consec_count"]:
                        # check if the student's attendance has been already
                        # taken, if not, record the student's attendance
                        #if curPerson not in studentDict.keys():
                                #studentDict[curPerson]                                  =
datetime.now().strftime("%H:%M:%S")
```

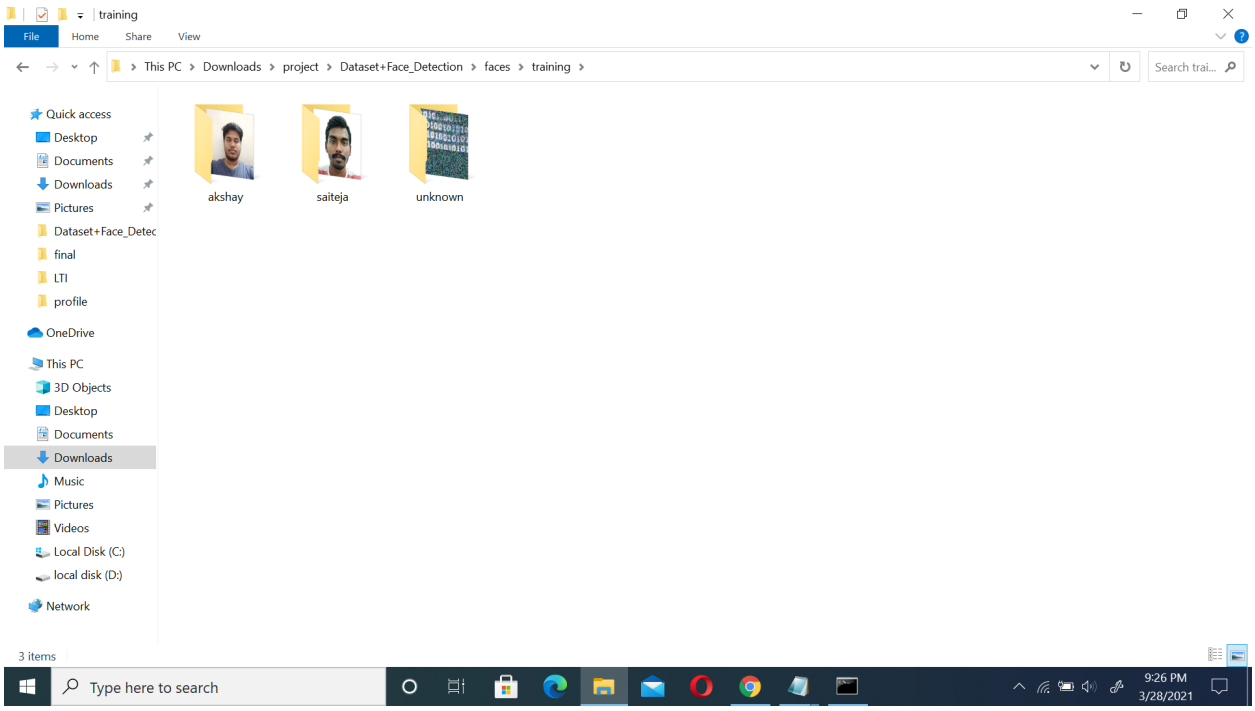
```
                    # get the student's name from the database and let them
                    # know that their attendance has been taken
                name = studentTable.search(where(
                            curPerson))[0][curPerson][0]


                # construct a label saying the student has their attendance
                # taken and draw it on to the frame
                label = "{}, you are now marked as present in {}".format(
                    name, conf["class2"])
                cv2.putText(frame, label, (5, 175),
                    cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255, 0, 0), 2)

                if name == student1:
                    student1_flag=1
                    consecCount = 0
                elif name == student2:
                    student2_flag=1
                    consecCount = 0

                elif name == 'unknown':
                    if unknown_flag == 0:
                        unknown_flag=1

                        if unknown_flag ==1:

                            #resp   =      sendSMS(apikey,   numbers,sender,
message)

                            #print(resp)
                            unknown_flag=2
            else:
                # construct a label asking the student to stand in fron
                # to the camera and draw it on to the frame
```

```
                label = "Please stand in front of the camera"
                cv2.putText(frame, label, (5, 175),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255, 0, 0), 2)

# show the frame
cv2.imshow("Survilance System", frame)
key = cv2.waitKey(1) & 0xFF

# check if the `q` key was pressed
if key == ord("q"):
        # check if the student dictionary is not empty, and if so,
        # insert the attendance into the database
        if len(studentDict) != 0:
                attendanceTable.insert({str(date.today()): studentDict})

        # break from the loop
        break

if timeDiff2 == conf["max_time_limit2"]:

        f = open("a.txt", "a+")
        if student1_flag == 1:
                f.write("\n" +  student1 +" "+ "is present")
        else:
                f.write("\n"+ student1 +" "+"is absent")
        if student2_flag == 1:
                f.write("\n"+ student2 +" "+ "is present")
        else:
                f.write("\n"+ student2 +" "+"is absent")

        f.close()
        msg_head = create_msg()
        attach=attach_file(msg_head)
        #SendMail(attach)
```

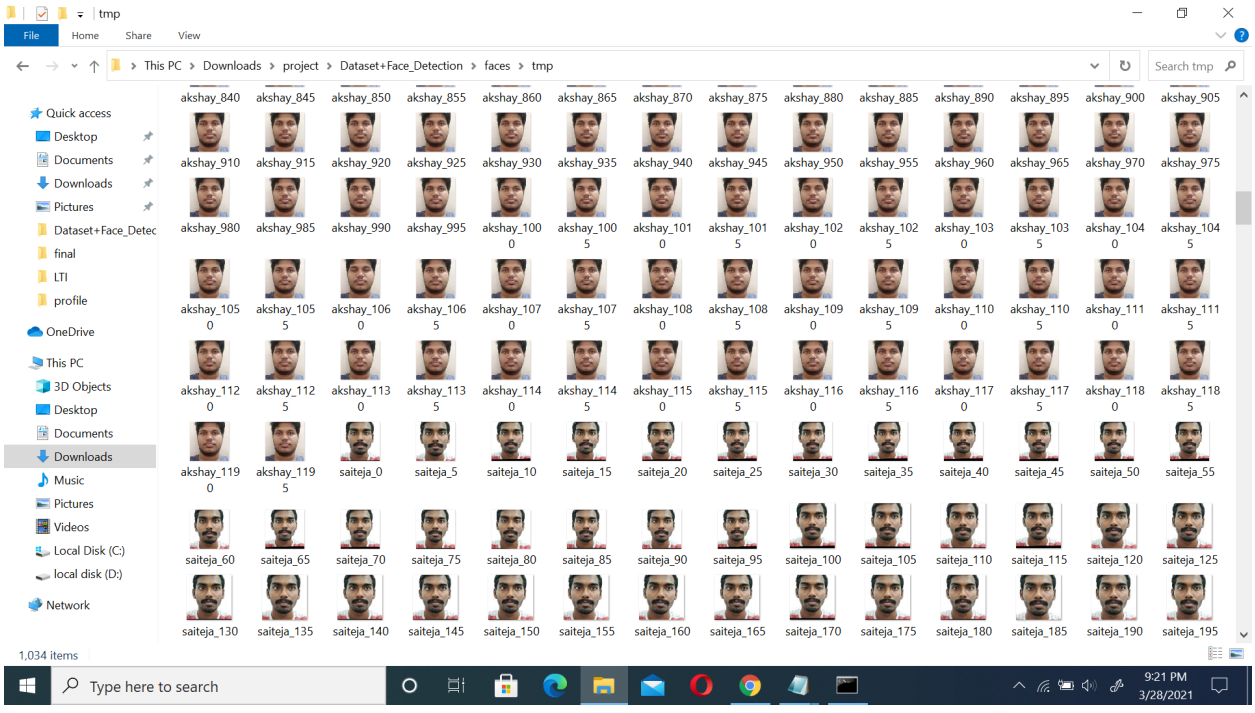
```
                    process_flag=1
                    student1_flag=0
                    student2_flag=0

print("[INFO] cleaning up...")
print('label', label)
print("student1_flag,student2_flag,student3_flag", student1_flag,student2_flag,student3_flag)
# clean up
print("[INFO] cleaning up...")
vs.stop()
db.close()
```

APPENDIX II - SCREENSHOTS

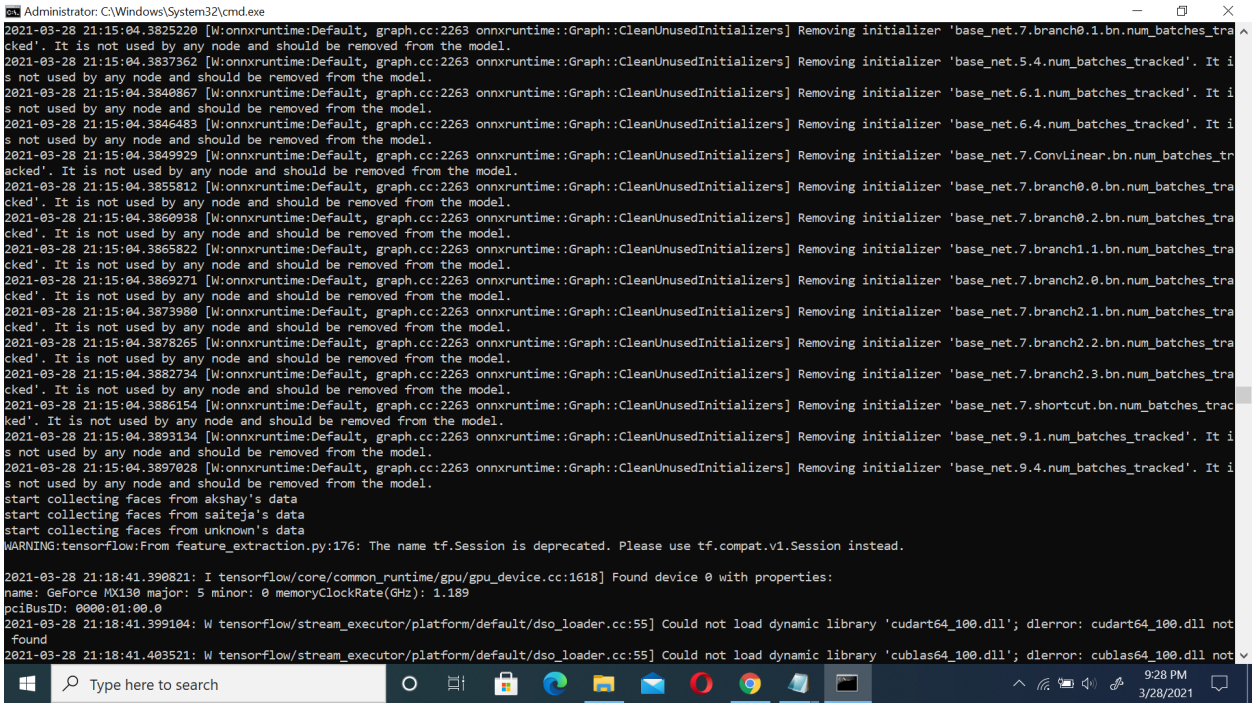
DATASET COLLECTION



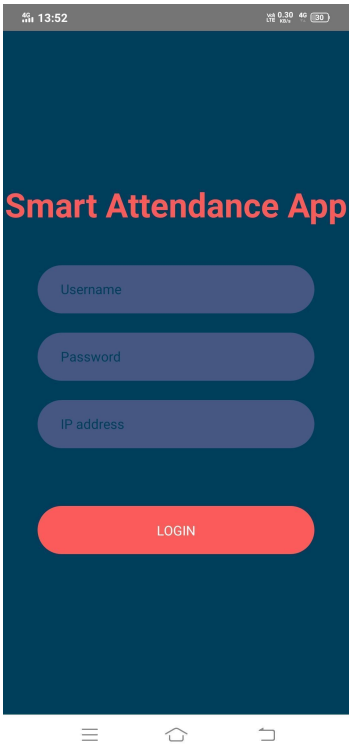
SAVED FACE DATASET



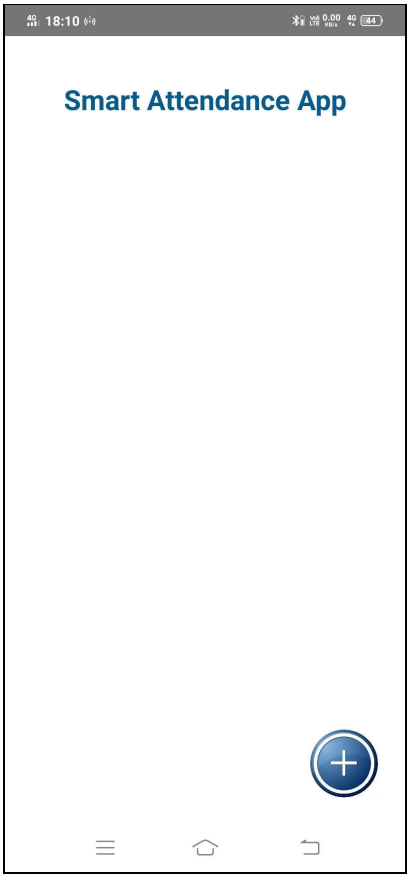
FACE RECOGNITION



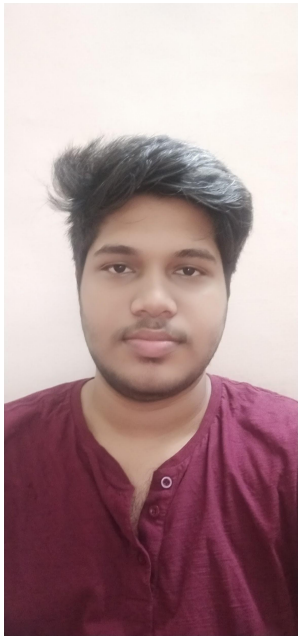
LOGIN SCREEN OF MOBILE APPLICATION



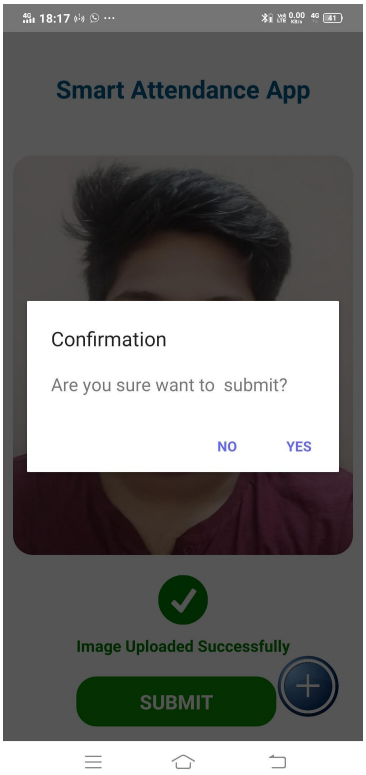
HOME PAGE



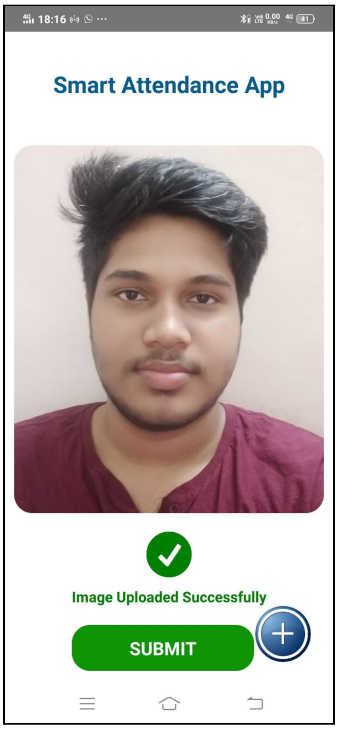
CAPTURING IMAGE FOR ATTENDANCE



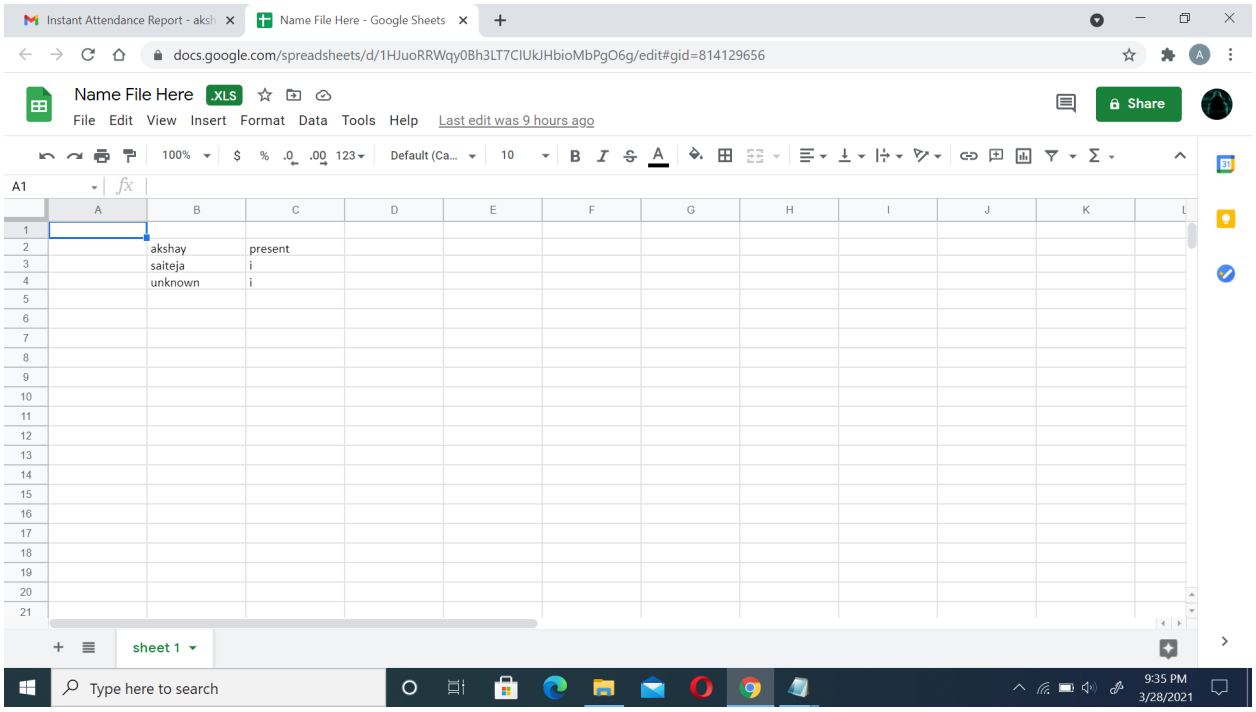
CONFIRMATION ON SUBMIT



SUCCESSFUL UPLOAD



ATTENDANCE REPORT GENERATION



REPORT SENT TO EMAIL