ATTENDANCE MARKING BASED ON FACE IDENTIFICATION

A Major Project Report Submitted
In partial fulfillment of the requirements for the award of the degree of

Bachelor of Technology
in
Computer Science and Engineering

by

Ambidi Harshavardhan	**17N31A0513**
Ambeeru Vignesh	**17N31A0512**
M. Pavan Kumar	**17N31A0508**

Under the esteemed guidance of

G. Manoj Kumar
Assistant Professor

Department of Computer Science and Engineering

Malla Reddy College of Engineering & Technology
(Autonomous Institution- UGC, Govt. of India)
(Affiliated to JNTUH, Hyderabad, Approved by AICTE, NBA &NAAC with 'A' Grade)
Maisammaguda, Kompally, Dhulapally, Secunderabad – 500100
website: www.mrcet.ac.in

2017-2021

Malla Reddy College of Engineering & Technology

(Autonomous Institution- UGC, Govt. of India)

(Affiliated to JNTUH, Hyderabad, Approved by AICTE, NBA &NAAC with 'A' Grade)

Maisammaguda, Kompally, Dhulapally, Secunderabad – 500100

website: www.mrcet.ac.in

CERTIFICATE

This is to certify that this is the bonafide record of the project entitled "Attendance Marking Based on Face Identification", submitted by Ambeeru Vignesh(17N31A0512), Ambidi Harshavardhan(17N31A0513) and M. Pavan Kumar(17N31A0508) of B.Tech in the partial fulfillment of the requirements for the degree of Bachelor of Technology in Computer Science and Engineering, Department of CSE during the year 2020-2021. The results embodied in this project report have not been submitted to any other university or institute for the award of any degree or diploma.

Internal Guide	**Head of the Department**
G. Manoj Kumar	**Dr. T. Venu Gopal**
Assistant Professor	**HOD, CSE.**

External Examiner

DECLARATION

We hereby declare that the project titled “Attendance Marking Based on Face Identification” submitted to Malla Reddy College of Engineering and Technology (UGC Autonomous), affiliated to Jawaharlal Nehru Technological University Hyderabad (JNTUH) for the award of the degree of Bachelor of Technology in Computer Science and Engineering is a result of original research carried-out in this thesis. It is further declared that the project report or any part thereof has not been previously submitted to any University or Institute for the award of degree or diploma.

Ambidi Harshavardhan - 17N31A0513

Ambeeru Vignesh- 17N31A0512

M. Pavan Kumar - 17N31A0508

ACKNOWLEDGEMENT

We feel ourselves honored and privileged to place our warm salutation to our college Malla Reddy College of Engineering and Technology (UGC-Autonomous), our Director Dr. VSK Reddy and Principal Dr. S. Srinivasa Rao who gave us the opportunity to have experience in engineering and profound technical knowledge.

We express our heartiest thanks to our Head of the Department Dr. T. Venu Gopal for encouraging us in every aspect of our project and helping us realize our full potential.

We would like to thank our internal guide and Project Coordinator Dr. S. Shanthi for his regular guidance and constant encouragement. We are extremely grateful to his valuable suggestions and unflinching co-operation throughout project work.

We would like to thank our class in charge Mr. G. Manoj Kumar who in spite of being busy with his duties took time to guide and keep us on the correct path.

We would also like to thank all the supporting staff of the Department of CSE and all other departments who have been helpful directly or indirectly in making our project a success.

We are extremely grateful to our parents for their blessings and prayers for the completion of our project that gave us strength to do our project.

With regards and gratitude,

Ambidi Harshavardhan - 17N31A0513
Ambeeru Vignesh - 17N31A0512
M. Pavan Kumar - 17N31A0508

ABSTRACT

Automatic face recognition (AFR) technologies have seen dramatic improvements in performance over the past years, and such systems are now widely used for security and commercial applications. An automated system for human face recognition in a real time background for a college to mark the attendance of their students. So, Attendance Marking Based on Face Identification is a real-world solution which comes with day to day activities of handling students. The task is very difficult as the real time background subtraction in an image is still a challenge. To detect real time human face are used and a simple fast Principal Component Analysis (PCA) has used to recognize the faces detected with a high accuracy rate. The matched face is used to mark attendance of the student. Our system maintains the attendance records of students automatically. Manual entering of attendance in logbooks becomes a difficult task and it also wastes the time. So, we designed an efficient module that comprises of face recognition to manage the attendance records of students.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF SCREENSHOTS

LIST OF TESTCASES

1. INTRODUCTION

Maintaining the attendance is very important in all the institutes for checking the performance of employees. Every institute has its own method in this regard. Some are taking attendance manually using the old paper or file-based approach and some have adopted methods of automatic attendance using some biometric techniques. But in these methods employees have to wait for long time in making a queue at time they enter the office. Many biometric systems are available but the key authentications are same is all the techniques. Every biometric system consists of enrolment process in which unique features of a person is stored in the database and then there are processes of identification and verification. These two processes compare the biometric feature of a person with previously stored template captured at the time of enrollment. Biometric templates can be of many types like Fingerprints, Eye Iris, Face, Hand Geometry, Signature, Gait and voice. Our system uses the face recognition approach for the automatic attendance of employees in the office room environment without employee's intervention. Face recognition consists of two steps, in first step faces are detected in the image and then these detected faces are compared with the database for verification. A number of methods have been proposed for face detection i.e. Ada Boost algorithm, the Float Boost algorithm, the S-Ada Boost algorithm Support Vector Machines (SVM), and the Bayes classifier. The efficiency of face recognition algorithm can be increased with the fast face detection algorithm. In all the above methods SURF is most efficient. Our system utilized this algorithm for the detection of faces in the office room image. Face recognition techniques can be divided into two types Appearance based which use texture features that is applied to whole face or some specific Regions, other is Feature based which uses geometric features like mouth, nose, eyes, eye brows, cheeks and Relation between them. Statistical tools such as Linear Discriminant Analysis (LDA), Principal Component Analysis (PCA), Kernel Methods, and Neural Networks, Eigen-faces have been used for construction of face templates. Illumination invariant algorithm is utilized for removing the lighting effect inside the office room.

This chapter gives an overview about the purpose, aim, objectives, background and operation environment of the system.

1.1 PURPOSE, AIM AND OBJECTIVE:

Face recognition-based attendance system is a process of recognizing the students face for taking attendance by using face feature matching based on high - definition monitor video and other information technology.

Every biometric system consists of enrollment process in which unique features of a person is stored in the database and then there are processes of identification and verification. These two processes compare the biometric feature of a person with previously stored template captured at the time of enrollment. Biometric templates can be of many types like Fingerprints, Eye Iris, Face, Hand Geometry, Signature, Gait and voice. Our system uses the face recognition approach for the automatic attendance of employees in the office room environment without employees' intervention. Face recognition consists of two steps, in first step faces are detected in the image and then these detected faces are compared with the database for verification.

1.2 BACKGROUND OF PROJECT:

Automatic face recognition (AFR) technologies have seen dramatic improvements in performance over the past years, and such systems are now widely used for security and commercial applications. An automated system for human face recognition in a real time background for a college to mark the attendance of their students. So, Attendance Marking Based on Face Identification is a real-world solution which comes with day to day activities of handling students. This method is secure enough, reliable and available for use.

1.3 SCOPE OF PROJECT:

Every biometric system consists of enrollment process in which unique features of a person is stored in the database and then there are processes of identification and verification. But our project provides facility for the automated attendance of students. Uses live face recognition to recognize each individual and mark their attendance automatically. Utilizes video and image processing to provide inputs to the system. Thus enhancing the security for users.

1.4 MODULES DESCRIPTION:

This project is composed of three main modules:

User module:

In this module, user will open camera and track images of every student to whom he wanted to take attendance through this application. For each tracking process it will take 60 images and then user should close training process. Once the process is done data will be stored in a folder 0, 1, 2 etc. for each user new folder is created. Also, it stores the details of user (id, name) in an excel file.

Training Process:

Once taking images process is done images from folder and for each image training process is done using OpenCV and yml file is stored in folder. This yml file is used for testing new images.

Detection Process:

In this process, when user opens camera it will track live images of user and convert user image to gray colour and check with face recognition model and then boxes are drawn on each face and features are verified with trained model and displays a box around the face with name and id.

Finally, attendance is marked to that user with correct date and time in an excel file.

2. SYSTEM ANALYSIS

In this chapter, we will discuss and analyze about the developing process of Audit Control including software requirement specification (SRS) and comparison between existing and proposed system. The functional and non-functional requirements are included in SRS part to provide complete description and overview of system requirement before the developing process is carried out. Besides that, existing vs. proposed provides a view of how the proposed system will be more efficient than the existing one.

2.1 HARDWARE AND SOFTWARE REQUIREMENTS:

Requirement Specification provides a high secure storage to the web server efficiently. Software requirements deal with software and hardware resources that need to be installed on a serve which provides optimal functioning for the application. These software and hardware requirements need to be installed before the packages are installed. These are the most common set of requirements defined by any operation system. These software and hardware requirements provide a compatible support to the operation system in developing an application.

2.1.1 HARDWARE REQUIREMENTS:

The hardware requirement specifies each interface of the software elements and the hardware elements of the system. These hardware requirements include configuration characteristics.

OS	:	Windows 7 or above
Processor	**:**	I3 or above.
Ram	**:**	2GB or above.
Hard disk	**:**	500GB or above.

2.1.2 SOFTWARE REQUIREMENTS:

The software requirements specify the use of all required software products like data management system. The required software product specifies the numbers and version. Each interface specifies the purpose of the interfacing software as related to this software product.

Technology/Language	**:**	Python.
Operating System	**:**	Windows, Linux, Mac.
IDE	**:**	Visual Studio Code/Sublime Editor.
Front End	**:**	tkinter.
Libraries	**:**	OpenCV, pandas, NumPy, datetime, PIL.

2.2 SOFTWARE REQUIREMENT SPECIFICATION:

2.2.1 What is SRS:

A **Software Requirements Specification** (SRS) is a detailed description of a software system to be developed with its functional and non-functional requirements. The SRS is developed based the agreement between customer and contractors. It may include the use cases of how user is going to interact with software system.

The software requirement specification document consistent of all necessary requirements required for project development. To develop the software system, we should have clear understanding of Software system. To achieve this, we need to continuous communication with customers to gather all requirements.

A good SRS defines the how Software System will interact with all internal modules, hardware, communication with other programs and human user interactions with wide range of real-life scenarios. Using the Software requirements specification (SRS) document on QA lead, managers create test plan. It is very important that testers must be cleared with every detail specified in this document in order to avoid faults in test cases and its expected results. It is highly recommended to review or test SRS documents before start writing test cases and making any plan for testing.

The SRS phase consists of two basic activities:

2.2.1.1 REQUIREMENTS ANALYSIS:

Requirements Analysis is the process of defining the expectations of the users for an application that is to be built or modified. It involves all the tasks that are conducted to identify the needs of different stakeholders. Therefore, requirements analysis means to analyse, document, validate and manage software or system requirements.

High-quality requirements are documented, actionable, measurable, testable, traceable, helps to identify business opportunities, and are defined to a facilitate system design.

2.2.1.2 REQUIREMENTS SPECIFICATION:

A Requirement is a statement of one thing a product must do or a quality it must have. A Requirement Specification is a collection of the set of all requirements that are to be imposed on the design and verification of the product. The specification also contains other related information necessary for the design, verification, and maintenance of the product.

2.2.2 DOCUMENT CONVENTIONS:

We have used Times New Roman (text size 12).Bold Font is used for Main Headings (text size of 16). Normal font is used for sub headings (text size of 14).
Font: Times New Roman
Main Heading: Bold Font

2.2.3 INTENDED AUDIENCE AND READING SUGGESTIONS:

This document is for better understanding for Remote desktop control. Mainly intended for Head of the Dept., Internal guide, External guide, Staff members, Users and colleagues. This detail given below guides every normal user to how to go through this document for better understanding. The sequence to follow for better understanding is here Purpose, Scope, Features, Operating requirements, Modules present in the project, Advantages, References etc.

2.2.4 SCOPE:

This document is the only one that describes the requirements of the system. It is meant for the use by the developers, and will also be the basis for validating the final delivered system. Any changes made to the requirements in the future will have to go through a formal change approval process. The developer is responsible for asking for clarifications, where necessary, and will not make any alterations without the permission of the client.

2.3 EXISTING SYSTEM:

In the existing system, every organization has its own method in this regard. Some are taking attendance manually using the old paper or file-based approach. Some have adopted methods of automatic attendance using some biometric techniques.

2.4 DRAWBACKS OF EXISTING SYSTEM:

- But in these methods, people have to wait for long time in making a queue at time they enter the organization.
- Many biometric systems are available but the key authentications are same as all the techniques.
- Every biometric system consists of enrollment process in which unique features of a person is stored in the database and then there are processes of identification and verification.

2.5 PROPOSED SYSTEM:

The system consists of a camera that captures 60 images of the person at a time using OpenCV and saves those particular images for further process. After enhancement, it trains the machine to recognize the face using the collection of images captured before. This is shown in the experimental setup in Figure. Each person will be identified with a name and a unique ID. When the machine is trained perfectly, it will detect the faces during the time of attendance and logs the attendance for that particular person with complete date and time in an excel file.

2.6 ADVANTAGES OF THE PROPOSED SYSTEM:

- In this way, a lot of time is saved and this is highly secure process no one can mark the attendance of other.

- Attendance is maintained on the excel sheet so it can be accessed for purposes like administration, monitoring etc..

2.7 FEASABILITY STUDY:

A feasibility study is an analysis of how successfully a project can be completed, accounting for factors that affect it such as economic, technological and operational. Project managers use feasibility studies to determine potential positive and negative outcomes of a project before investing a considerable amount of time and money into it.

During the stage of our feasibility study, we had to undergo the following steps as described under:

- Identify the origin of data at different levels of the system.
- Identify the expectation of end user from the finished product/system.
- Analyze the drawback(s) of the existing system.

Technical Feasibility Study: It lays out details on how a good or service will be delivered, which includes transportation, business location, technology needed, materials and labor.

Financial Feasibility Study: It is a projection of the amount of funding or startup capital needed, what sources of capital can and will be used and what kind of return can be expected on the investment.

Organizational Feasibility Study: It is a definition of the corporate and legal structure of the business; this may include information about the founders, their professional background and the skills they possess necessary to get the company off the ground and keep it operational.

2.8 OPERATING ENVIRONMENT:

This application will be operating in a Graphical User Interface (GUI) which is created using tkinter library. It is an application through which users will be interacting with the machine for attendance marking. The only requirement to use this application is that the machine should run the build file to install all the required libraries and run the application perfectly with no errors.

3. TECHNOLOGIES USED

3.1 PYTHON:

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

- **Python is Interpreted** − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
- **Python is Interactive** − You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
- **Python is Object-Oriented** − Python supports Object-Oriented style or technique of programming that encapsulates code within objects.
- **Python is a Beginner's Language** − Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

3.1.1 HISTORY OF PYTHON:

Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands.

Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, SmallTalk, and Unix shell and other scripting languages.

Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL).

Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress.

3.1.2 PYTHON FEATURES:

Python's features include −

- **Easy-to-learn** − Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
- **Easy-to-read** − Python code is more clearly defined and visible to the eyes.
- **Easy-to-maintain** − Python's source code is fairly easy-to-maintain.
- **A broad standard library** − Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.
- **Interactive Mode** − Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.
- **Portable** − Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
- **Extendable** − You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
- **Databases** − Python provides interfaces to all major commercial databases.
- **GUI Programming** − Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.
- **Scalable** − Python provides a better structure and support for large programs than shell scripting.

Apart from the above-mentioned features, Python has a big list of good features, few are listed below −

- It supports functional and structured programming methods as well as OOP.
- It can be used as a scripting language or can be compiled to byte-code for building large applications.
- It provides very high-level dynamic data types and supports dynamic type checking.
- IT supports automatic garbage collection.
- It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.

3.1.3 STANDARD DATA TYPES:

The data stored in memory can be of many types. For example, a person's age is stored as a numeric value and his or her address is stored as alphanumeric characters. Python has various standard data types that are used to define the operations possible on them and the storage method for each of them.

Python has five standard data types −

- **Python Numbers:** Number data types store numeric values. Number objects are created when you assign a value to them
- **Python Strings:** Strings in Python are identified as a contiguous set of characters represented in the quotation marks. Python allows for either pairs of single or double quotes. Subsets of strings can be taken using the slice operator ([ ] and [:]) with indexes starting at 0 in the beginning of the string and working their way from -1 at the end.
- **Python Lists:** Lists are the most versatile of Python's compound data types. A list contains items separated by commas and enclosed within square brackets ([]). To some extent, lists are similar to arrays in C. One difference between them is that all the items belonging to a list can be of different data type.

  The values stored in a list can be accessed using the slice operator ([ ] and [:]) with indexes starting at 0 in the beginning of the list and working their way to end -1. The plus (+) sign is the list concatenation operator, and the asterisk (*) is the repetition operator.
- **Python Tuples:** A tuple is another sequence data type that is similar to the list. A tuple consists of a number of values separated by commas. Unlike lists, however, tuples are enclosed within parentheses.

  The main differences between lists and tuples are: Lists are enclosed in brackets ([ ]) and their elements and size can be changed, while tuples are enclosed in parentheses (( )) and cannot be updated. Tuples can be thought of as read-only lists.
- **Python Dictionaries:** Python's dictionaries are kind of hash table type. They work like associative arrays or hashes found in Perl and consist of key-value pairs. A dictionary key can be almost any Python type, but are usually numbers or strings. Values, on the other hand, can be any arbitrary Python object.

Dictionaries are enclosed by curly braces ({ }) and values can be assigned and accessed using square braces ([]).

3.1.4 DIFFERENT MODES IN PYTHON:

Python has two basic modes: normal and interactive.

- The normal mode is the mode where the scripted and finished .py files are run in the Python interpreter.
- Interactive mode is a command line shell which gives immediate feedback for each statement, while running previously fed statements in active memory. As new lines are fed into the interpreter, the fed program is evaluated both in part and in whole

3.1.5 SOME PYTHON LIBRARIES:

1. Requests. The most famous http library written by kenneth reitz. It’s a must have for every python developer.

2. Scrapy. If you are involved in webscraping then this is a must have library for you. After using this library you won’t use any other.

3. wxPython. A gui toolkit for python. I have primarily used it in place of tkinter. You will really love it.

4. Pillow. A friendly fork of PIL (Python Imaging Library). It is more user friendly than PIL and is a must have for anyone who works with images.

5. SQLAlchemy. A database library. Many love it and many hate it. The choice is yours.

6. BeautifulSoup. I know it’s slow but this xml and html parsing library is very useful for beginners.

7. Twisted. The most important tool for any network application developer. It has a very beautiful api and is used by a lot of famous python developers.

8. NumPy. How can we leave this very important library ? It provides some advance math functionalities to python.

9. SciPy. When we talk about NumPy then we have to talk about scipy. It is a library of algorithms and mathematical tools for python and has caused many scientists to switch from ruby to python.

10. matplotlib. A numerical plotting library. It is very useful for any data scientist or any data analyzer.

11. Pygame. Which developer does not like to play games and develop them ? This library will help you achieve your goal of 2d game development.

12. Pyglet. A 3d animation and game creation engine. This is the engine in which the famous python port of minecraft was made

13. pyQT. A GUI toolkit for python. It is my second choice after wxpython for developing GUI's for my python scripts.

14. pyGtk. Another python GUI library. It is the same library in which the famous Bittorrent client is created.

15. Scapy. A packet sniffer and analyzer for python made in python.

16. pywin32. A python library which provides some useful methods and classes for interacting with windows.

17. nltk. Natural Language Toolkit – I realize most people won't be using this one, but it's generic enough. It is a very useful library if you want to manipulate strings. But it's capacity is beyond that. Do check it out.

18. nose. A testing framework for python. It is used by millions of python developers. It is a must have if you do test driven development.

19. SymPy. SymPy can do algebraic evaluation, differentiation, expansion, complex numbers, etc. It is contained in a pure Python distribution.

20. IPython. I just can't stress enough how useful this tool is. It is a python prompt on steroids. It has completion, history, shell capabilities, and a lot more. Make sure that you take a look at it.

3.1.6 PYTHON MODULES:

Python allows us to store our code in files (also called modules). This is very useful for more serious programming, where we do not want to retype a long function definition from the very beginning just to change one mistake. In doing this, we are essentially defining our own modules, just like the modules defined already in the Python library.

To support this, Python has a way to put definitions in a file and use them in a script or in an interactive instance of the interpreter. Such a file is called a module; definitions from a module can be imported into other modules or into the main module.

3.1.7 TESTING CODE:

As indicated above, code is usually developed in a file using an editor. To test the code, import it into a Python session and try to run it.

Usually there is an error, so you go back to the file, make a correction, and test again. This process is repeated until you are satisfied that the code works. The entire process is known as the development cycle.

There are two types of errors that you will encounter. Syntax errors occur when the form of some command is invalid. This happens when you make typing errors such as misspellings, or call something by the wrong name, and for many other reasons. Python will always give an error message for a syntax error.

3.1.8 FUNCTIONS IN PYTHON:

It is possible, and very useful, to define our own functions in Python. Generally speaking, if you need to do a calculation only once, then use the interpreter. But when you or others have need to perform a certain type of calculation many times, then define a function.

You use functions in programming to bundle a set of instructions that you want to use repeatedly or that, because of their complexity, are better self-contained in a sub-program and called when needed. That means that a function is a piece of code written to carry out a specified task.

To carry out that specific task, the function might or might not need multiple inputs. When the task is carried out, the function can or can not return one or more values. There are three types of functions in python:

help(), min(), print().

3.1.9 PYTHON NAMESPACE:

Generally speaking, a **namespace** (sometimes also called a context) is a naming system for making names unique to avoid ambiguity. Everybody knows a namespacing system from daily life, i.e. the naming of people in first name and family name (surname).

An example is a network: each network device (workstation, server, printer, ...) needs a unique name and address. Yet another example is the directory structure of file systems.

The same file name can be used in different directories, the files can be uniquely accessed via the pathnames. Many programming languages use namespaces or contexts for identifiers. An identifier defined in a namespace is associated with that namespace.

This way, the same identifier can be independently defined in multiple namespaces. (Like the same file names in different directories) Programming languages, which support namespaces, may have different rules that determine to which namespace an identifier belongs.

Namespaces in Python are implemented as Python dictionaries, this means it is a mapping from names (keys) to objects (values). The user doesn't have to know this to write a Python program and when using namespaces.

Some namespaces in Python:

- **global names** of a module.
- **local names** in a function or method invocation.
- **built-in names**: this namespace contains built-in functions (e.g. abs(), cmp(), ...) and built-in exception names.

3.1.10 GARBAGE COLLECTION:

Garbage Collector exposes the underlying memory management mechanism of Python, the automatic garbage collector. The module includes functions for controlling how the collector operates and to examine the objects known to the system, either pending collection or stuck in reference cycles and unable to be freed.

3.2 OpenCV Library:

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

The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms. These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, produce 3D point clouds from stereo cameras, stitch images together to produce a high resolution image of an entire scene, find similar images from an image database, remove red eyes from images taken using flash, follow eye movements, recognize scenery and establish markers to overlay it with augmented reality, etc. OpenCV has more than 47 thousand people of user community and estimated number of downloads exceeding 18 million. The library is used extensively in companies, research groups and by governmental bodies.

Along with well-established companies like Google, Yahoo, Microsoft, Intel, IBM, Sony, Honda, Toyota that employ the library, there are many startups such as Applied Minds, VideoSurf, and Zeitera, that make extensive use of OpenCV. OpenCV's deployed uses span the range from stitching street view images together, detecting intrusions in surveillance video in Israel, monitoring mine equipment in China, helping robots navigate and pick up objects at Willow Garage, detection of swimming pool drowning accidents in Europe, running interactive art in Spain and New York, checking runways for debris in Turkey, inspecting labels on products in factories around the world on to rapid face detection in Japan.

It has C++, Python, Java and MATLAB interfaces and supports Windows, Linux, Android and Mac OS. OpenCV leans mostly towards real-time vision applications and takes advantage of MMX and SSE instructions when available. A full-featured CUDA and OpenCL interfaces are being actively developed right now. There are over 500 algorithms and about 10 times as many functions that compose or support those algorithms. OpenCV is written natively in C++ and has a templated interface that works seamlessly with STL containers.

3.3 Tkinter Module:

Tkinter is a Python binding to the Tk GUI toolkit. It is the standard Python interface to the Tk GUI toolkit, and is Python's de facto standard GUI. Tkinter is included with standard Linux, Microsoft Windows and Mac OS X installs of Python.

The name Tkinter comes from Tk interface. Tkinter was written by Fredrik Lundh. Tkinter is free software released under a Python license.

As with most other modern Tk bindings, Tkinter is implemented as a Python wrapper around a complete Tcl interpreter embedded in the Python interpreter. Tkinter calls are translated into Tcl commands, which are fed to this embedded interpreter, thus making it possible to mix Python and Tcl in a single application.

There are several popular GUI library alternatives available, such as wxPython, PyQt, PySide, Pygame, Pyglet, and PyGTK.

3.4 Pandas:

Pandas is a Python package that provides fast, flexible, and expressive data structures designed to make working with structured (tabular, multidimensional, potentially heterogeneous) and time series data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, real world data analysis in Python. Additionally, it has the broader goal of becoming the most powerful and flexible open source data analysis / manipulation tool available in any language. It is already well on its way toward this goal.

Pandas is well suited for many different kinds of data:

- Tabular data with heterogeneously-typed columns, as in an SQL table or Excel spreadsheet
- Ordered and unordered (not necessarily fixed-frequency) time series data.
- Arbitrary matrix data (homogeneously typed or heterogeneous) with row and column labels
- Any other form of observational / statistical data sets. The data actually need not be labeled at all to be placed into a pandas data structure

The two primary data structures of pandas, Series (1-dimensional) and DataFrame (2-dimensional), handle the vast majority of typical use cases in finance, statistics, social science, and many areas of engineering. For R users, DataFrame provides everything that

R’s data frame provides and much more. pandas is built on top of NumPy and is intended to integrate well within a scientific computing environment with many other 3rd party libraries.

Here are just a few of the things that pandas does well:

- Easy handling of missing data (represented as NaN) in floating point as well as non-floating point data
- Size mutability: columns can be inserted and deleted from DataFrame and higher dimensional objects
- Automatic and explicit data alignment: objects can be explicitly aligned to a set of labels, or the user can simply ignore the labels and let Series, DataFrame, etc. automatically align the data for you in computations
- Powerful, flexible group by functionality to perform split-apply-combine operations on data sets, for both aggregating and transforming data
- Make it easy to convert ragged, differently-indexed data in other Python and NumPy data structures into DataFrame objects
- Intelligent label-based slicing, fancy indexing, and subsetting of large data sets
- Intuitive merging and joining data sets
- Flexible reshaping and pivoting of data sets
- Hierarchical labelling of axes (possible to have multiple labels per tick)
- Robust IO tools for loading data from flat files (CSV and delimited), Excel files, databases, and saving / loading data from the ultrafast HDF5 format
- Time series-specific functionality: date range generation and frequency conversion, moving window statistics, date shifting and lagging.

Many of these principles are here to address the shortcomings frequently experienced using other languages / scientific research environments. For data scientists, working with data is typically divided into multiple stages: munging and cleaning data, analyzing / modeling it, then organizing the results of the analysis into a form suitable for plotting or tabular display. pandas is the ideal tool for all of these tasks.

3.5 NumPy:

NumPy is the fundamental package for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more.

At the core of the NumPy package, is the and array object. This encapsulates n-dimensional arrays of homogeneous data types, with many operations being performed in compiled code for performance. There are several important differences between NumPy arrays and the standard Python sequences:

- NumPy arrays have a fixed size at creation, unlike Python lists (which can grow dynamically). Changing the size of a ndarray will create a new array and delete the original.
- The elements in a NumPy array are all required to be of the same data type, and thus will be the same size in memory. The exception: one can have arrays of (Python, including NumPy) objects, thereby allowing for arrays of different sized elements.
- NumPy arrays facilitate advanced mathematical and other types of operations on large numbers of data. Typically, such operations are executed more efficiently and with less code than is possible using Python's built-in sequences.
- A growing plethora of scientific and mathematical Python-based packages are using NumPy arrays; though these typically support Python-sequence input, they convert such input to NumPy arrays prior to processing, and they often output NumPy arrays. In other words, in order to efficiently use much (perhaps even most) of today's scientific/mathematical Python-based software, just knowing how to use Python's built-in sequence types is insufficient - one also needs to know how to use NumPy arrays.

Vectorization describes the absence of any explicit looping, indexing, etc., in the code - these things are taking place, of course, just "behind the scenes" in optimized, pre-compiled C code.

NumPy fully supports an object-oriented approach, starting, once again, with *ndarray*. For example, *ndarray* is a class, possessing numerous methods and attributes. Many of its methods are mirrored by functions in the outer-most NumPy namespace, allowing the programmer to code in whichever paradigm they prefer. This flexibility has allowed the

NumPy array dialect and NumPy *ndarray* class to become the *de-facto* language of multi-dimensional data interchange used in Python.

3.6 datetime:

In Python, date and time are not a data type of its own, but a module named datetime can be imported to work with the date as well as time. Datetime module comes built into Python, so there is no need to install it externally.
Datetime module supplies classes to work with date and time. These classes provide a number of functions to deal with dates, times and time intervals. Date and datetime are an object in Python, so when you manipulate them, you are actually manipulating objects and not string or timestamps.

The datetime classes are categorize into 6 main classes –

- date – An idealized naive date, assuming the current Gregorian calendar always was, and always will be, in effect. Its attributes are year, month and day.
- time – An idealized time, independent of any particular day, assuming that every day has exactly 24*60*60 seconds. Its attributes are hour, minute, second, microsecond, and tzinfo.
- datetime – Its a combination of date and time along with the attributes year, month, day, hour, minute, second, microsecond, and tzinfo.
- timedelta – A duration expressing the difference between two date, time, or datetime instances to microsecond resolution.
- tzinfo – It provides time zone information objects.
- timezone – A class that implements the tzinfo abstract base class as a fixed offset from the UTC (New in version 3.2).

3.7 PIL:

The Python Imaging Library adds image processing capabilities to your Python interpreter.

This library provides extensive file format support, an efficient internal representation, and fairly powerful image processing capabilities.

The core image library is designed for fast access to data stored in a few basic pixel formats. It should provide a solid foundation for a general image processing tool.

3.8 LBPH ALGORITHM:

Local Binary Pattern (LBP) is a simple yet very efficient texture operator which labels the pixels of an image by thresholding the neighborhood of each pixel and considers the result as a binary number.

It was first described in 1994 (LBP) and has since been found to be a powerful feature for texture classification. It has further been determined that when LBP is combined with histograms of oriented gradients (HOG) descriptor, it improves the detection performance considerably on some datasets.

Using the LBP combined with histograms we can represent the face images with a simple data vector.

As LBP is a visual descriptor it can also be used for face recognition tasks, as can be seen in the following step-by-step explanation.

3.8.1 STEP-BY-STEP:

Now that we know a little more about face recognition and the LBPH, let's go further and see the steps of the algorithm:

1. **Parameters**: the LBPH uses 4 parameters:
    - Radius: the radius is used to build the circular local binary pattern and represents the radius around the central pixel. It is usually set to 1.
    - Neighbors: the number of sample points to build the circular local binary pattern. Keep in mind: the more sample points you include, the higher the computational cost. It is usually set to 8.
    - Grid X: the number of cells in the horizontal direction. The more cells, the finer the grid, the higher the dimensionality of the resulting feature vector. It is usually set to 8.
    - Grid Y: the number of cells in the vertical direction. The more cells, the finer the grid, the higher the dimensionality of the resulting feature vector. It is usually set to 8.

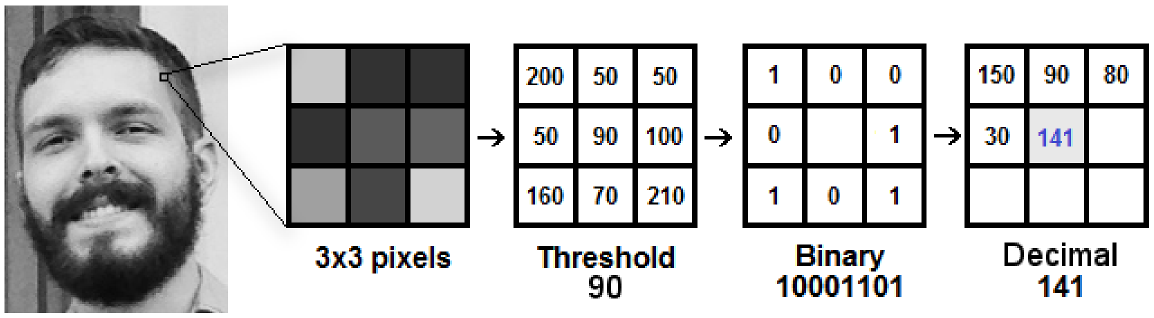
Don't worry about the parameters right now, you will understand them after reading the next steps.

2. **Training the Algorithm**: First, we need to train the algorithm. To do so, we need to use a dataset with the facial images of the people we want to recognize. We need to also set an ID (it may be a number or the name of the person) for each image, so the

algorithm will use this information to recognize an input image and give you an output. Images of the same person must have the same ID. With the training set already constructed, let's see the LBPH computational steps.

3. **Applying the LBP operation**: The first computational step of the LBPH is to create an intermediate image that describes the original image in a better way, by highlighting the facial characteristics. To do so, the algorithm uses a concept of a sliding window, based on the parameters radius and neighbors.

The image below shows this procedure:



Based on the image above, let's break it into several small steps so we can understand it easily:

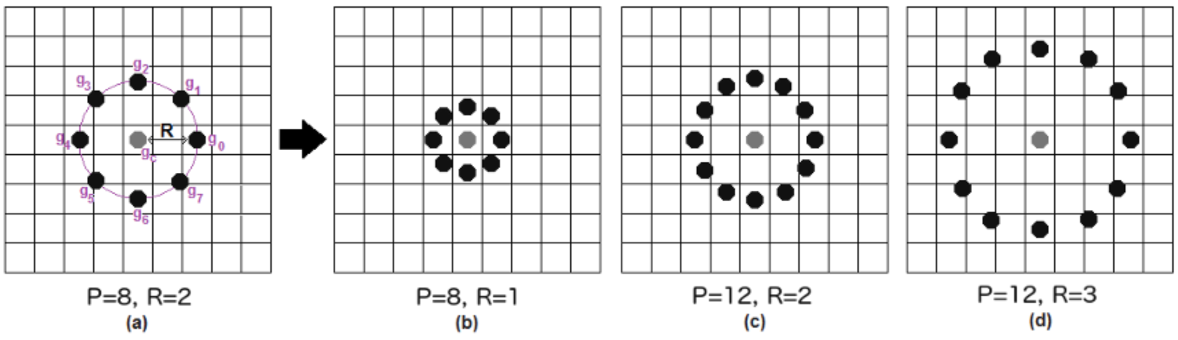
- Suppose we have a facial image in grayscale.
- We can get part of this image as a window of 3x3 pixels.
- It can also be represented as a 3x3 matrix containing the intensity of each pixel (0~255).
- Then, we need to take the central value of the matrix to be used as the threshold.
- This value will be used to define the new values from the 8 neighbors.
- For each neighbor of the central value (threshold), we set a new binary value. We set 1 for values equal or higher than the threshold and 0 for values lower than the threshold.

- Now, the matrix will contain only binary values (ignoring the central value). We need to concatenate each binary value from each position from the matrix line by line into a new binary value (e.g. 10001101). Note: some authors use other approaches to concatenate the binary values (e.g. clockwise direction), but the final result will be the same.

- Then, we convert this binary value to a decimal value and set it to the central value of the matrix, which is actually a pixel from the original image.

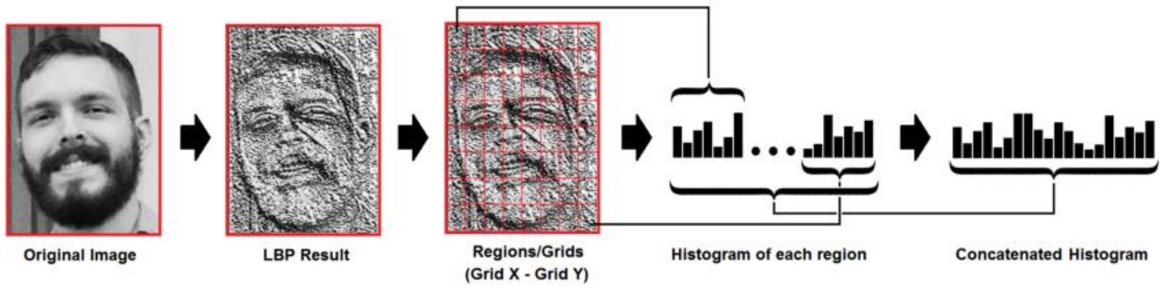
- At the end of this procedure (LBP procedure), we have a new image which represents better the characteristics of the original image.

- **Note**: The LBP procedure was expanded to use a different number of radius and neighbors, it is called Circular LBP.



It can be done by using **bilinear interpolation**. If some data point is between the pixels, it uses the values from the 4 nearest pixels (2x2) to estimate the value of the new data point.

4. Extracting the Histograms: Now, using the image generated in the last step, we can use the **Grid X** and **Grid Y** parameters to divide the image into multiple grids, as can be seen in the following image:



Based on the image above, we can extract the histogram of each region as follows:

- As we have an image in grayscale, each histogram (from each grid) will contain only 256 positions (0~255) representing the occurrences of each pixel intensity.
- Then, we need to concatenate each histogram to create a new and bigger histogram. Supposing we have 8x8 grids, we will have 8x8x256=16.384 positions in the final histogram. The final histogram represents the characteristics of the image original image.

The LBPH algorithm is pretty much it.

5. **Performing the face recognition**: In this step, the algorithm is already trained. Each histogram created is used to represent each image from the training dataset. So, given an input image, we perform the steps again for this new image and creates a histogram which represents the image.

- So to find the image that matches the input image we just need to compare two histograms and return the image with the closest histogram.
- We can use various approaches to compare the histograms (calculate the distance between two histograms), for example: euclidean distance, chi-square, absolute value, etc. In this example, we can use the Euclidean distance (which is quite known) based on the following formula:

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

- So the algorithm output is the ID from the image with the closest histogram. The algorithm should also return the calculated distance, which can be used as a '**confidence**' measurement. **Note**: don't be fooled about the 'confidence' name, as lower confidences are better because it means the distance between the two histograms is closer.
- We can then use a threshold and the 'confidence' to automatically estimate if the algorithm has correctly recognized the image. We can assume that the algorithm has successfully recognized if the confidence is lower than the threshold defined.

3.8.3 CONCLUSIONS OF LBPH ALGORITHM:

- LBPH is one of the easiest face recognition algorithms.
- It can represent local features in the images.
- It is possible to get great results (mainly in a controlled environment).
- It is robust against monotonic gray scale transformations.
- It is provided by the OpenCV library (Open Source Computer Vision Library).

4. SYSTEM DESIGN & UML DIAGRAMS

System design is transition from a user oriented document to programmers or data base personnel. The design is a solution, how to approach to the creation of a new system. This is composed of several steps. It provides the understanding and procedural details necessary for implementing the system recommended in the feasibility study. Designing goes through logical and physical stages of development, logical design reviews the present physical system, prepare input and output specification, details of implementation plan and prepare a logical design walkthrough.

4.1 SOFTWARE DESIGN:

In designing the software following principles are followed:

1) **Modularity and partitioning**: Software is designed such that, each system should consist of hierarchy of modules and serve to partition into separate function.
2) **Coupling:** Modules should have little dependence on other modules of a system.
3) **Cohesion:** Modules should carry out in a single processing function.
4) **Shared use:** Avoid duplication by allowing a single module be called by other that need the function it provides.

4.2 ARCHITECTURE:

Architecture diagram is a diagram of a system, in which the principal parts or functions are represented by blocks connected by lines that show the relationships of the blocks. The block diagram is typically used for a higher level, less detailed description aimed more at understanding the overall concepts and less at understanding the details of implementation.

The purpose of the design phase is to arrange an answer of the matter such as by the necessity document. This part is that the opening moves in moving the matter domain to the answer domain. The design phase satisfies the requirements of the system. The design of a system is probably the foremost crucial issue warm heartedness the standard of the software package. It’s a serious impact on the later part, notably testing and maintenance.

The output of this part is that the style of the document. This document is analogous to a blueprint of answer and is employed later throughout implementation, testing and maintenance. The design activity is commonly divided into 2 separate phases System Design and Detailed Design.

System Design conjointly referred to as top-ranking style aims to spot the modules that ought to be within the system, the specifications of those modules, and the way them move with one another to supply the specified results.

At the top of the system style all the main knowledge structures, file formats, output formats, and also the major modules within the system and their specifications square measure set. System design is that the method or art of process the design, components, modules, interfaces, and knowledge for a system to satisfy such as needs. Users will read it because the application of systems theory to development.

Detailed Design, the inner logic of every of the modules laid out in system design is determined. Throughout this part, the small print of the info of a module square measure sometimes laid out in a high-level style description language that is freelance of the target language within which the software package can eventually be enforced.

In system design the main target is on distinguishing the modules, whereas throughout careful style the main target is on planning the logic for every of the modules.

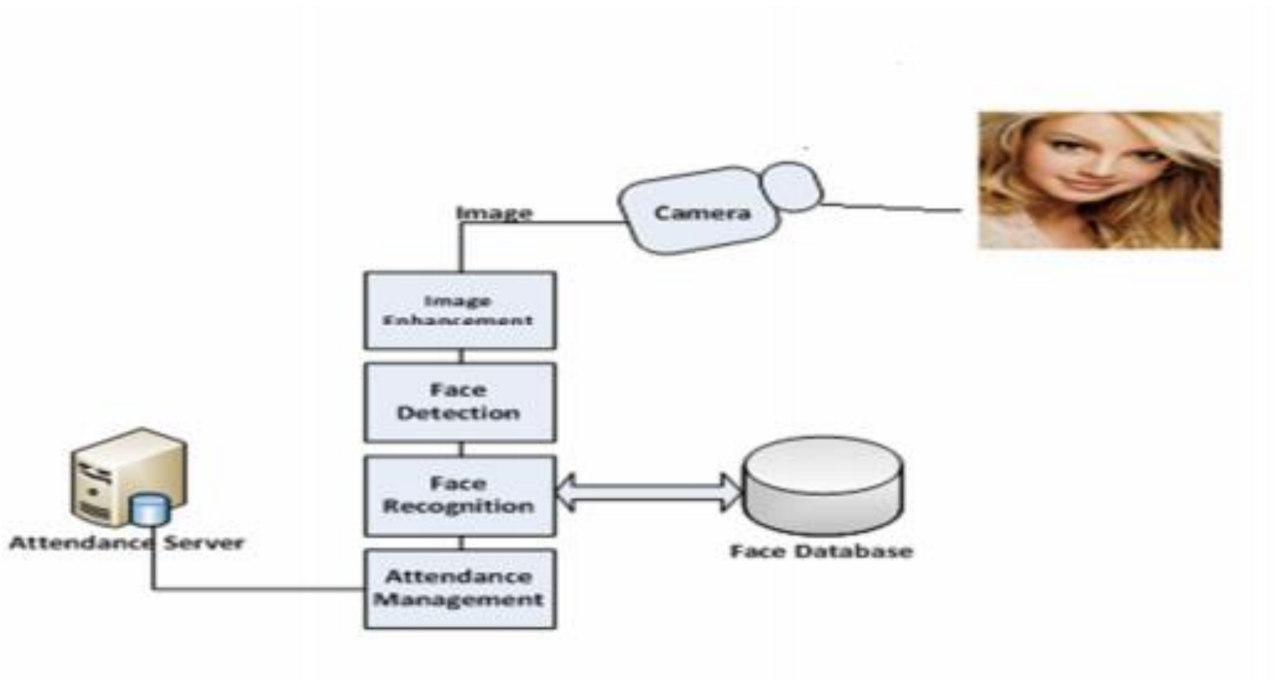


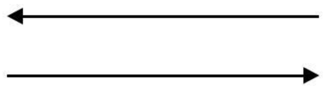
Fig-4.2 System Architecture Design

4.3 **DATA FLOW DIAGRAMS:**

Data Flow Diagram can also be termed as bubble chart. It is a pictorial or graphical form, which can be applied to represent the input data to a system and multiple functions carried out on the data and the generated output by the system.

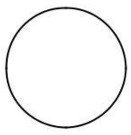
A graphical tool accustomed describe and analyze the instant of knowledge through a system manual or automatic together with the method, stores of knowledge, and delays within the system. The transformation of knowledge from input to output, through processes, is also delineate logically and severally of the physical elements related to the system. The DFD is also known as a data flow graph or a bubble chart. The Basic Notation used to create a DFD's are as follows:

- **Dataflow:**

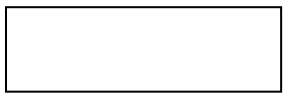


- **Process:**

.

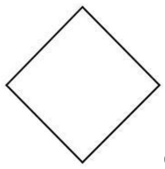


- **Source:**



- **Data Store:**

- **Rhombus**:



decision

4.4 UNIFIED MODELING LANGUAGE (UML) :

The unified modeling is a standard language for specifying, visualizing, constructing and documenting the system and its components is a graphical language which provides a vocabulary and set of semantics and rules.

The UML focuses on the conceptual and physical representation of the system. It captures the decisions and understandings about systems that must be constructed. It is used to understand, design, configure and control information about the systems.

Depending on the development culture, some of these artifacts are treated more or less formally than others. Such artifacts are not only the deliverables of a project; they are also critical in controlling, measuring, and communicating about a system during its development and after its deployment.

The UML addresses the documentation of a system's architecture and all of its details. The UML also provides a language for expressing requirements and for tests. Finally, the UML provides a language for modeling the activities of project planning and release management.

A UML system is represented using five different views that describe the system from distinctly different perspective.
Each view is defined by a set of diagrams, which is as follows.

User Model View: This view represents the system from the users perspective. The analysis representation describes a usage scenario from the end-user’s perspective.

Structural Model view: In this model the data and functionality are arrived from inside the system. This model view models the static structures.

Behavioral Model View: It represents the dynamic of behavioral as parts of the system, depicting the interactions of collection between various structural elements described in the user model and structural model view.

Implementation Model View: In this the structural and behavioral as parts of the system are represented as they are to be built.

4.5 BUILDING BLOCKS OF UML:

The vocabulary of the UML encompasses three kinds of building blocks:

- ✓ Things.

- ✓ Relationships.
- ✓ Diagrams.

4.5.1 Things in the UML:

Things are the abstractions that are first-class citizens in a model; relationships tie these things together; diagrams group interesting collections of things.

There are four kinds of things in the UML:

- ✓ Structural things.
- ✓ Behavioral things.
- ✓ Grouping things.
- ✓ Annotational things.

1. **Structural things** are the nouns of UML models. The structural things used in the project design are:
   - ✓ First, a **class** is a description of a set of objects that share the same attributes, operations, relationships and semantics.

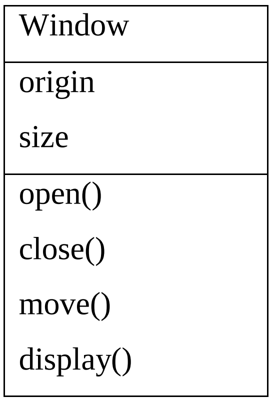


Fig: Classes

   - ✓ Second, a **use case** is a description of set of sequence of actions that a system performs that yields an observable result of value to particular actor.

Place order

Fig: Use Cases

   - ✓ Third, a node is a physical element that exists at runtime and represents a computational resource, generally having at least some memory and often processing capability.

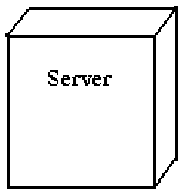


Fig: Nodes

2. **Behavioral things** are the dynamic parts of UML models. The behavioral thing used is:
    - ✓ Interaction: An interaction is a behavior that comprises a set of messages exchanged among a set of objects within a particular context to accomplish a specific purpose. An interaction involves a number of other elements, including messages, action sequences (the behavior invoked by a message, and links (the connection between objects).

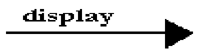


Fig: Messages

4.5.2 Relationships in the UML:

There are four kinds of relationships in the UML:

- ✓ Dependency.
- ✓ Association.
- ✓ Generalization.
- ✓ Realization.

- ✓ A **dependency** is a semantic relationship between two things in which a change to one thing may affect the semantics of the other thing (the dependent thing).

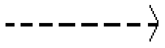


Fig: Dependencies

- ✓ An **association** is a structural relationship that describes a set links, a link being a connection among objects. Aggregation is a special kind of association, representing a structural relationship between a whole and its parts.

Fig: Association

- ✓ A **generalization** is a specialization/ generalization relationship in which objects of the specialized element (the child) are substitutable for objects of the generalized element(the parent).

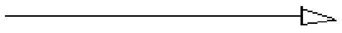


Fig: Generalization

- ✓ A **realization** is a semantic relationship between classifiers, where in one classifier specifies a contract that another classifier guarantees to carry out.

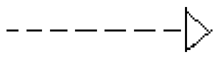


Fig: Realization

4.5.3 UML DIAGRAMS:

4.5.3.1 CLASS DIAGRAM:

A class is a representation of an object and, in many ways; it is simply a template from which objects are created. Classes form the main building blocks of an object-oriented application. Although thousands of students attend the university, you would only model one class, called Student, which would represent the represent the entire collection of students.

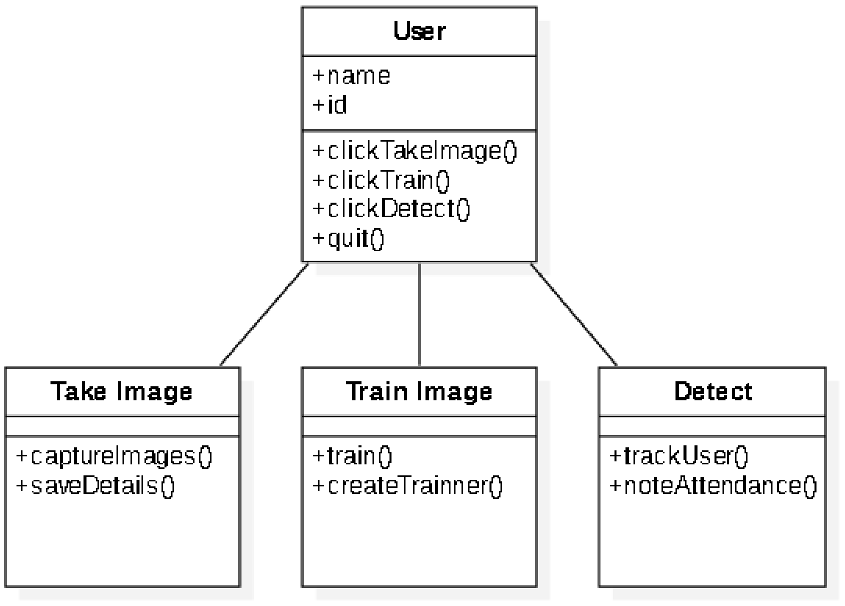


Fig-4.5.3.1 Class Diagram

4.5.3.2 USE CASE DIAGRAM:

A use case diagram is a graph of actors set of use cases enclosed by a system boundary, communication associations between the actors and users and generalization among use cases. The use case model defines the outside (actors) and inside (use case) of the system’s behavior.

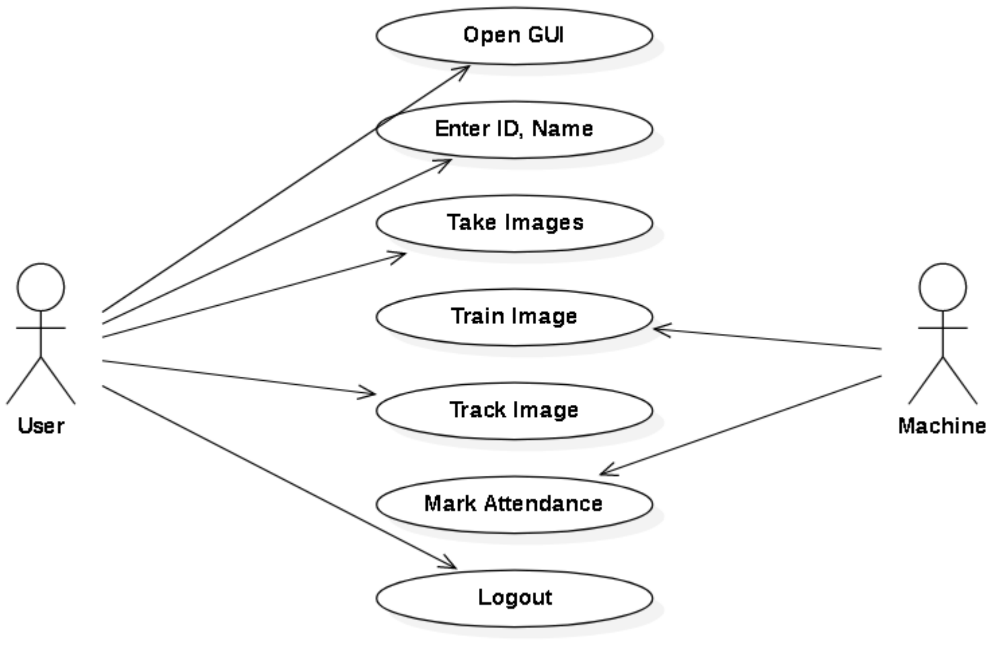


Fig-4.5.3.2 Use case Diagram

4.5.3.3 SEQUENCE DIAGRAM:

Sequence diagram are used to represent the flow of messages, events and actions between the objects or components of a system. Time is represented in the vertical direction showing the sequence of interactions of the header elements, which are displayed horizontally at the top of the diagram.

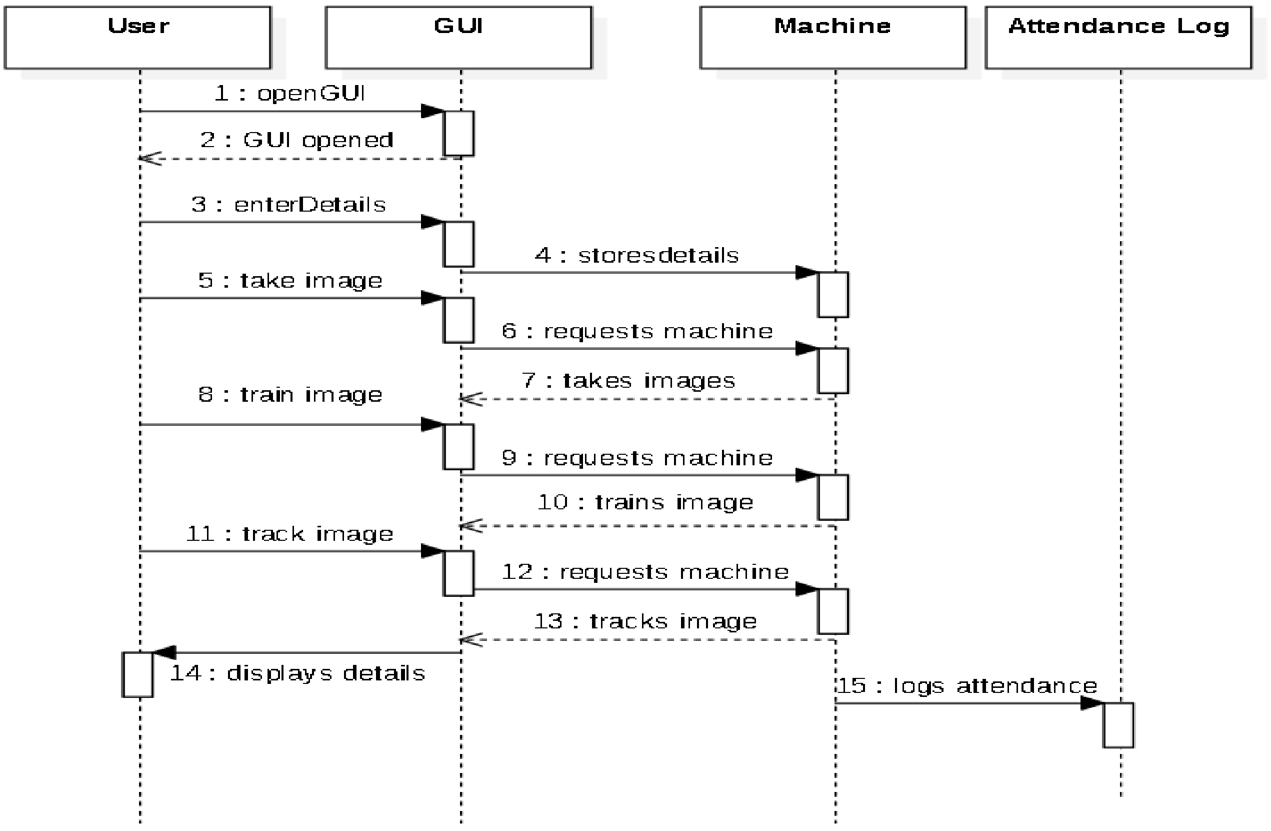


Fig-4.5.3.3 Sequence Diagram

4.5.3.4 ACTIVITY DIAGRAM:

Activity diagram represent the business and operational workflows of a system. An Activity diagram is a dynamic diagram that shows the activity and the event that causes the object to be in the particular state.

So, what is the importance of an Activity diagram, as opposed to a State diagram? A State diagram shows the different states an object is in during the lifecycle of its existence in the system, and the transitions in the states of the objects. These transitions depict the activities causing these transitions, shown by arrows.

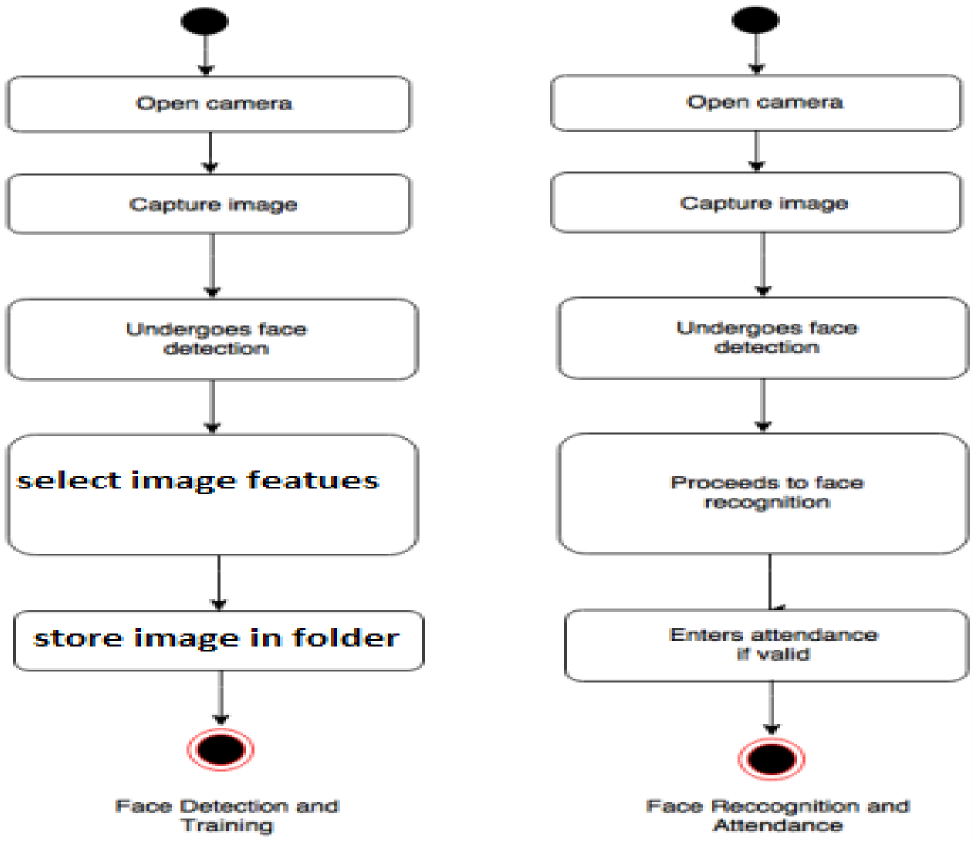


Fig-4.5.3.4 Activity Diagram

4.5.3.5 STATE CHART DIAGRAM:

State chart diagram is used to describe the states of different objects in its life cycle. So the emphasis is given on the state changes upon some internal or external events. These states of objects are important to analyse and implement them accurately.

State chart diagrams are very important for describing the states. States can be identified as the condition of objects when a particular event occurs.

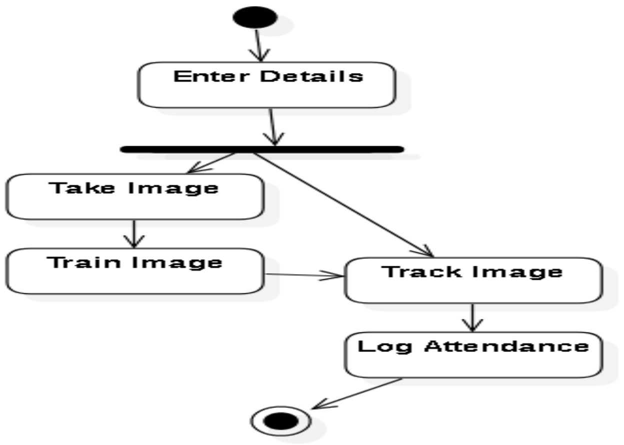


Fig-4.5.3.5 State Chart Diagram

4.5.3.6 COMPONENT DIAGRAM:

In the Unified Modeling Language, a Component diagram depicts how components are wired together to form larger components and or software systems. They are used to illustrate the structure of arbitrarily complex systems.

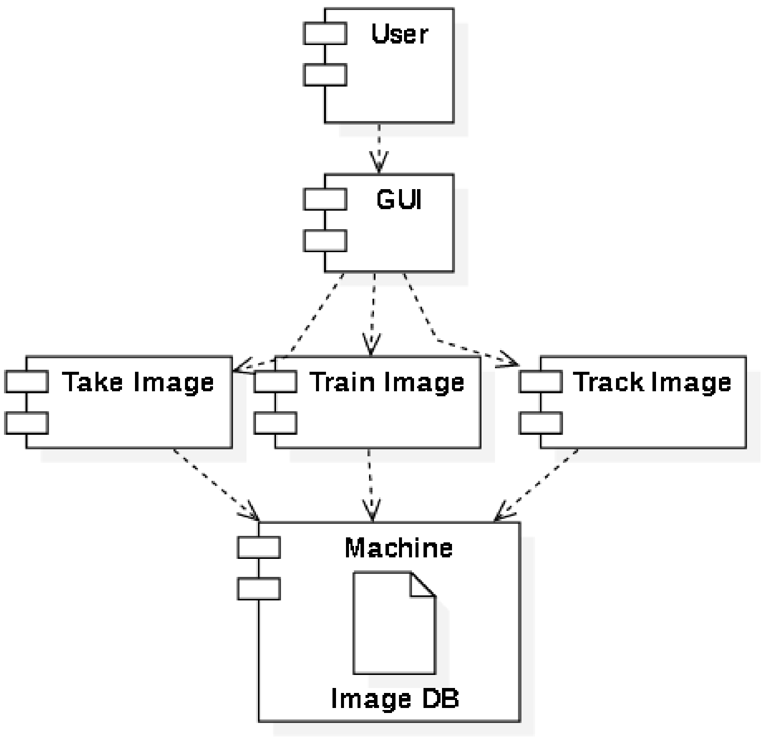


Fig-4.5.3.6 Component Diagram

4.5.3.7 DEPLOYMENT DIAGRAM:

Deployment diagrams are used to visualize the topology of the physical components of a system where the software components are deployed. So, deployment diagrams are used to describe the static deployment view of a system. Deployment diagrams consist of nodes and their relationships.

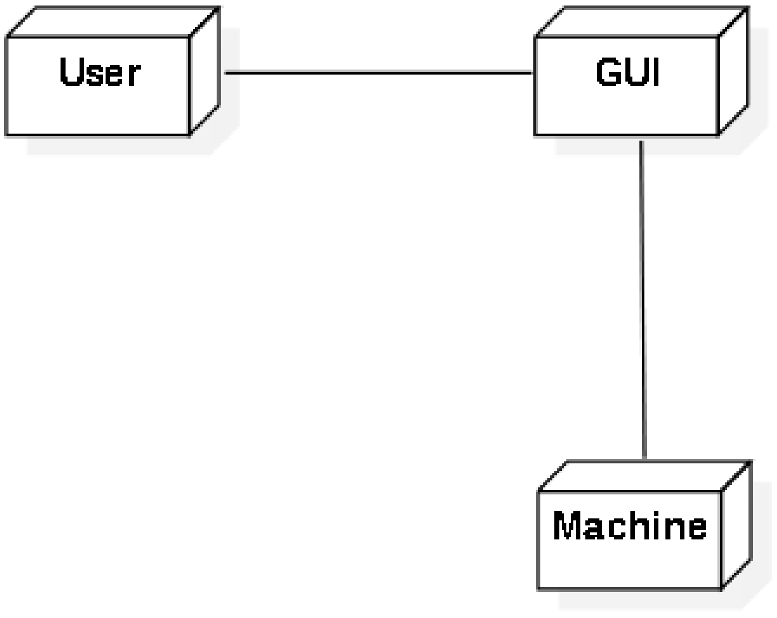


Fig-4.5.3.7 Deployment Diagram

5. INPUT/OUTPUT DESIGN

5.1 INPUT DESIGN:

Input Design plays a vital role in the life cycle of software development, it requires very careful attention of developers. The input design is to feed data to the application as accurate as possible. So, inputs are supposed to be designed effectively so that the errors occurring while feeding are minimized. According to Software Engineering Concepts, the input forms or screens are designed to provide to have a validation control over the input limit, range and other related validations. Input design is the process of converting the user created input into a computer-based format. The goal of the input design is to make the data entry logical and free from errors.

Validations are required for each data entered. Whenever a user enters an erroneous data, error message is displayed and the user can move on to the subsequent pages after completing all the entries in the current page.

5.2 OUTPUT DESIGN:

The Output from the computer is required to mainly create an efficient method of communication within the company primarily among the project leader and his team members, in other words, the administrator and the clients. The output of VPN is the system which allows the project leader to manage his clients in terms of creating new clients and assigning new projects to them, maintaining a record of the project validity and providing folder level access to each client on the user side depending on the projects allotted to him. After completion of a project, a new project may be assigned to the client. User authentication procedures are maintained at the initial stages itself.

6. IMPLEMENTATION

setup.py:

```
from cx_Freeze import setup, Executable
import sys,os
PYTHON_INSTALL_DIR = os.path.dirname(os.path.dirname(os.__file__))
os.environ['TCL_LIBRARY'] = os.path.join(PYTHON_INSTALL_DIR, 'tcl', 'tcl8.6')
os.environ['TK_LIBRARY'] = os.path.join(PYTHON_INSTALL_DIR, 'tcl', 'tk8.6')

base = None

if sys.platform == 'win32':
    base = None



executables = [Executable("train.py", base=base)]

packages = ["idna","os","sys","cx_Freeze","tkinter","cv2","setup",
            "numpy","PIL","pandas","datetime","time"]
options = {
    'build_exe': {

        'packages':packages,
    },

}

setup
(
    name = "ToolBox",
    options = options,
    version = "0.0.1",
    description = 'Vision ToolBox',
    executables = executables
)

#write python setup build
```

train.py:

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

from tkinter import *
import tkinter as tk
from tkinter import Message ,Text
import cv2,os
import shutil
import csv
import numpy as np
from PIL import Image, ImageTk
import pandas as pd
import datetime
import time
import tkinter.ttk as ttk
import tkinter.font as font

window = tk.Tk()
#helv36 = tk.Font(family='Helvetica', size=36, weight='bold')
window.title("A14 Batch Major Project")

#dialog_title = 'QUIT'
#dialog_text = 'Are you sure?'
#answer = messagebox.askquestion(dialog_title, dialog_text)

window.geometry('1280x720')
window.configure(background='gainsboro')
```

```
#img = ImageTk.PhotoImage(file='H:/Face detection based Attendance system/Face
detection based Attendance system/CODE/Face-Recognition-Based-Attendance-
System/BG.jpg')
#panel = Label(window, image=img)
#panel.place(x=0, y=0)

#window.attributes('-fullscreen', True)

window.grid_rowconfigure(0, weight=1)
window.grid_columnconfigure(0, weight=1)

#path = "profile.jpg"

#Creates a Tkinter-compatible photo image, which can be used everywhere Tkinter expects
an image object.
#img = ImageTk.PhotoImage(Image.open(path))

#The Label widget is a standard Tkinter widget used to display a text or image on the screen.
#panel = tk.Label(window, image = img)



#panel.pack(side = "left", fill = "y", expand = "no")

#cv_img = cv2.imread("img541.jpg")
#x, y, no_channels = cv_img.shape
#canvas = tk.Canvas(window, width = x, height =y)
#canvas.pack(side="left")
#photo = PIL.ImageTk.PhotoImage(image = PIL.Image.fromarray(cv_img))
# Add a PhotoImage to the Canvas
#canvas.create_image(0, 0, image=photo, anchor=tk.NW)

#msg = Message(window, text='Hello, world!')

# Font is a tuple of (font_family, size_in_points, style_modifier_string)
```

```
message = tk.Label(window, text="Attendance Marking Based on Face Identification"
,bg="lawn green"  ,fg="red"  ,width=50  ,height=3,font=('times', 30, 'bold underline'))

message.place(x=200, y=20)

lbl = tk.Label(window, text="Enter ID",width=20  ,height=2  ,fg="red"  ,bg="green yellow"
,font=('times', 15, ' bold ') )
lbl.place(x=400, y=200)

txt = tk.Entry(window,width=20  ,bg="lawn green" ,fg="black",font=('times', 15, ''))
txt.place(x=700, y=215)

lbl2 = tk.Label(window, text="Enter Name",width=20  ,fg="red"  ,bg="green yellow"
,height=2 ,font=('times', 15, ' bold '))
lbl2.place(x=400, y=300)

txt2 = tk.Entry(window,width=20  ,bg="lawn green"  ,fg="black",font=('times', 15, '')  )
txt2.place(x=700, y=315)

lbl3 = tk.Label(window, text="Status",width=20  ,fg="red"  ,bg="green yellow"  ,height=2
,font=('times', 15, ' bold underline '))
lbl3.place(x=400, y=400)

message = tk.Label(window, text="" ,bg="lawn green"  ,fg="black"  ,width=30  ,height=2,
activebackground = "yellow" ,font=('times', 15, ' bold '))
message.place(x=700, y=400)

lbl3 = tk.Label(window, text="Attendance Tracker: ",width=20  ,fg="red"  ,bg="green
yellow"  ,height=2 ,font=('times', 15, ' bold  underline'))
lbl3.place(x=400, y=650)
```

```
message2 = tk.Label(window, text="" ,fg="black"   ,bg="lawn green",activeforeground =
"green",width=30  ,height=2  ,font=('times', 15, ' bold '))
message2.place(x=700, y=650)

message3 = tk.Label(window,text="(Batch A14
Project)",fg="red",bg="gainsboro",width="20",height="3",font=('times', 16, ' bold'))
message3.place(x=1100, y=650)



def clear():
    txt.delete(0, 'end')
    res = ""
    message.configure(text= res)



def clear2():
    txt2.delete(0, 'end')
    res = ""
    message.configure(text= res)



def is_number(s):
    try:
        float(s)
        return True
    except ValueError:
        pass

    try:
        import unicodedata
        unicodedata.numeric(s)
        return True
    except (TypeError, ValueError):
        pass

    return False


def TakeImages():
    Id=(txt.get())
    name=(txt2.get())
    if(is_number(Id) and name.isalpha()):
        cam = cv2.VideoCapture(0)
        harcascadePath = "haarcascade_frontalface_default.xml"
        detector=cv2.CascadeClassifier(harcascadePath)
        sampleNum=0
        while(True):
            ret, img = cam.read()
            gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
            faces = detector.detectMultiScale(gray, 1.3, 5)
            for (x,y,w,h) in faces:
                cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,0),2)
                #incrementing sample number
                sampleNum=sampleNum+1
                #saving the captured face in the dataset folder TrainingImage
                cv2.imwrite("TrainingImage\ "+name +"."+Id +'.'+ str(sampleNum) + ".jpg",
gray[y:y+h,x:x+w])
                #display the frame
                cv2.imshow('frame',img)
            #wait for 100 miliseconds
            if cv2.waitKey(100) & 0xFF == ord('q'):
                break
            # break if the sample number is morethan 100
            elif sampleNum>60:
                break
        cam.release()
        cv2.destroyAllWindows()
        res = "Images Saved for ID : " + Id +" Name : "+ name
        row = [Id , name]
        with open('StudentDetails\StudentDetails.csv','a+') as csvFile:
            writer = csv.writer(csvFile)
            writer.writerow(row)
        csvFile.close()
        message.configure(text= res)
    else:
        if(is_number(Id)):
            res = "Enter Alphabetical Name"
            message.configure(text= res)
        if(name.isalpha()):
            res = "Enter Numeric Id"
            message.configure(text= res)


def TrainImages():
    recognizer = cv2.face_LBPHFaceRecognizer.create()#recognizer =
cv2.face.LBPHFaceRecognizer_create()#$cv2.createLBPHFaceRecognizer()
    harcascadePath = "haarcascade_frontalface_default.xml"
    detector =cv2.CascadeClassifier(harcascadePath)
    faces,Id = getImagesAndLabels("TrainingImage")
    recognizer.train(faces, np.array(Id))
    recognizer.save("TrainingImage\Trainner.yml")
    res = "Image Trained"#+",".join(str(f) for f in Id)
    message.configure(text= res)


def getImagesAndLabels(path):
    #get the path of all the files in the folder
    imagePaths=[os.path.join(path,f) for f in os.listdir(path)]
    #print(imagePaths)

    #create empth face list
    faces=[]
    #create empty ID list
    Ids=[]
    #now looping through all the image paths and loading the Ids and the images
    for imagePath in imagePaths:
        #loading the image and converting it to gray scale
        pilImage=Image.open(imagePath).convert('L')
        #Now we are converting the PIL image into numpy array
        imageNp=np.array(pilImage,'uint8')
        #getting the Id from the image
        Id=int(os.path.split(imagePath)[-1].split(".")[1])
        # extract the face from the training image sample
        faces.append(imageNp)
        Ids.append(Id)
    return faces,Ids


def TrackImages():
    recognizer = cv2.face.LBPHFaceRecognizer_create()#cv2.createLBPHFaceRecognizer()
    recognizer.read("TrainingImage\Trainner.yml")
    harcascadePath = "haarcascade_frontalface_default.xml"
    faceCascade = cv2.CascadeClassifier(harcascadePath);
    df=pd.read_csv("StudentDetails\StudentDetails.csv")
    cam = cv2.VideoCapture(0)
    font = cv2.FONT_HERSHEY_SIMPLEX
    col_names =  ['Id','Name','Date','Time']
    attendance = pd.DataFrame(columns = col_names)
    while True:
        ret, im =cam.read()
        gray=cv2.cvtColor(im,cv2.COLOR_BGR2GRAY)
        faces=faceCascade.detectMultiScale(gray, 1.2,5)
        for(x,y,w,h) in faces:
            cv2.rectangle(im,(x,y),(x+w,y+h),(225,0,0),2)
            Id, conf = recognizer.predict(gray[y:y+h,x:x+w])
            if(conf < 50):
                ts = time.time()
                date = datetime.datetime.fromtimestamp(ts).strftime('%Y-%m-%d')
                timeStamp = datetime.datetime.fromtimestamp(ts).strftime('%H:%M:%S')
                aa=df.loc[df['Id'] == Id]['Name'].values
                tt=str(Id)+"-"+aa
                attendance.loc[len(attendance)] = [Id,aa,date,timeStamp]

            else:
                Id='Unknown'
                tt=str(Id)
            if(conf > 75):
                noOfFile=len(os.listdir("ImagesUnknown"))+1
                cv2.imwrite("ImagesUnknown\Image"+str(noOfFile) + ".jpg", im[y:y+h,x:x+w])
            cv2.putText(im,str(tt),(x,y+h), font, 1,(255,255,255),2)
        attendance=attendance.drop_duplicates(subset=['Id'],keep='first')
        cv2.imshow('im',im)
        if (cv2.waitKey(1)==ord('q')):
            break
    ts = time.time()
    date = datetime.datetime.fromtimestamp(ts).strftime('%Y-%m-%d')
    timeStamp = datetime.datetime.fromtimestamp(ts).strftime('%H:%M:%S')
    Hour,Minute,Second=timeStamp.split(":")
    fileName="Attendance\Attendance_"+date+"_"+Hour+"-"+Minute+"-"+Second+".csv"
    attendance.to_csv(fileName,index=False)
    cam.release()
    cv2.destroyAllWindows()
    #print(attendance)
    res=attendance
    message2.configure(text= res)


clearButton = tk.Button(window, text="Clear", command=clear  ,fg="red"  ,bg="green yellow"  ,width=20  ,height=2 ,activebackground = "Red" ,font=('times', 15, ' bold '))
```

```
clearButton.place(x=950, y=200)

clearButton2 = tk.Button(window, text="Clear", command=clear2  ,fg="red"  ,bg="green yellow"  ,width=20  ,height=2, activebackground = "Red" ,font=('times', 15, ' bold '))

clearButton2.place(x=950, y=300)

takeImg = tk.Button(window, text="Take Pictures", command=TakeImages  ,fg="red"  ,bg="green yellow"  ,width=20  ,height=3, activebackground = "Red" ,font=('times', 15, ' bold '))

takeImg.place(x=200, y=500)

trainImg = tk.Button(window, text="Train", command=TrainImages  ,fg="red"  ,bg="green yellow"  ,width=20  ,height=3, activebackground = "Red" ,font=('times', 15, ' bold '))

trainImg.place(x=500, y=500)

trackImg = tk.Button(window, text="Detect", command=TrackImages  ,fg="red"  ,bg="green yellow"  ,width=20  ,height=3, activebackground = "Red" ,font=('times', 15, ' bold '))

trackImg.place(x=800, y=500)

quitWindow = tk.Button(window, text="Quit", command=window.destroy  ,fg="red"  ,bg="green yellow"  ,width=20  ,height=3, activebackground = "Red" ,font=('times', 15, ' bold '))

quitWindow.place(x=1100, y=500)


window.mainloop()
```

7. TESTING

Software testing is a critical element of software quality assurance and represents the ultimate review of specification, design and code generation.

7.1 TESTING OBJECTIVES:

- To ensure that during operation the system will perform as per specification.
- To make sure that system meets the user requirements during operation.
- To make sure that during the operation, incorrect input, processing and output will be detected.
- To see that when correct inputs are fed to the system the outputs are correct.
- To verify that the controls incorporated in the same system as intended.
- Testing is a process of executing a program with the intent of finding an error.
- A good test case is one that has a high probability of finding an as yet undiscovered error.

The software developed has been tested successfully using the following testing strategies and any errors that are encountered are corrected and again the part of the program or the procedure or function is put to testing until all the errors are removed. A successful test is one that uncovers an as yet undiscovered error.

Note that the result of the system testing will prove that the system is working correctly. It will give confidence to system designer, users of the system, prevent frustration during implementation process etc.

7.2 TESTING METHODOLOGIES:

Different kinds of testing methodologies are:

- ✓ White box testing.
- ✓ Black box testing.
- ✓ Unit testing.
- ✓ Integration testing.

- ✓ User acceptance testing.
- ✓ Output testing.
- ✓ Validation testing.
- ✓ System testing.

1) White Box Testing:

White box testing is a testing case design method that uses the control structure of the procedure design to derive test cases. All independents path in a module are exercised at least once, all logical decisions are exercised at once, execute all loops at boundaries and within their operational bounds exercise internal data structure to ensure their validity. Here the customer is given three chances to enter a valid choice out of the given menu. After which the control exits the current menu.

2) Black Box Testing:

Black Box Testing attempts to find errors in following areas or categories, incorrect or missing functions, interface error, errors in data structures, performance error and initialization and termination error. Here all the input data must match the data type to become a valid entry.

3) Unit Testing:

Unit testing focuses verification effort on the smallest unit of Software design that is the module. Unit testing exercises specific paths in a module’s control structure to ensure complete coverage and maximum error detection. This test focuses on each module individually, ensuring that it functions properly as a unit. Hence, the naming is Unit Testing.

4) Integration Testing:

Integration testing addresses the issues associated with the dual problems of verification and program construction. After the software has been integrated a set of high order tests are conducted. The main objective in this testing process is to take unit tested modules and builds a program structure that has been dictated by design.

The following are the types of Integration Testing:

- ✓ **Top Down Integration:**

This method is an incremental approach to the construction of program structure. Modules are integrated by moving downward through the control hierarchy, beginning with the main program module.

- ✓ **Bottom Up Integration:**

This method begins the construction and testing with the modules at the lowest level in the program structure. Since the modules are integrated from the bottom up, processing required for modules subordinate to a given level is always available and the need for stubs is eliminated.

5) User acceptance Testing:

User Acceptance of a system is the key factor for the success of any system. The system under consideration is tested for user acceptance by constantly keeping in touch with the prospective system users at the time of developing and making changes wherever required. The system developed provides a friendly user interface that can easily be understood even by a person who is new to the system.

6) Output Testing:

After performing the validation testing, the next step is output testing of the proposed system, since no system could be useful if it does not produce the required output in the specified format. Asking the users about the format required by them tests the outputs generated or displayed by the system under consideration. Hence the output format is considered in 2 ways – one is on screen and another in printed format.

7) Validation Testing:

Validation testing is generally performed on the following fields:

- ✓ **Text Field:**

The text field can contain only the number of characters lesser than or equal to its size. The text fields are alphanumeric in some tables and alphabetic in other tables. Incorrect entry always flashes and error message.

- ✓ **Numeric Field:**

The numeric field can contain only numbers from 0 to 9. An entry of any character flashes an error message. The individual modules are checked for accuracy and what it has to perform.

- ✓ **Preparation of Test Data:**

Taking various kinds of test data does the above testing. Preparation of test data plays a vital role in the system testing. After preparing the test data the system under study is tested using that test data. While testing the system by using test data errors are again uncovered and corrected by using above testing steps and corrections are also noted for future use.

- ✓ **Using Live Test Data:**

Live test data are those that are actually extracted from organization files. After a system is partially constructed, programmers or analysts often ask users to key in a set of data from their normal activities. Then, the systems person uses this data as a way to partially test the system. In other instances, programmers or analysts extract a set of live data from the files and have them entered themselves.

- ✓ **Using Artificial Test Data:**

Artificial test data are created solely for test purposes, since they can be generated to test all combinations of formats and values. In other words, the artificial data, which can quickly be prepared by a data generating utility program in the information systems department, make possible the testing of all login and control paths through the program.

The most effective test programs use artificial test data generated by persons other than those who wrote the programs. Often, an independent team of testers formulates a testing plan, using the systems specifications.

7.3 USER TRAINING:

Whenever a new system is developed, user training is required to educate them about the working of the system so that it can be put to efficient use by those for whom the system has been primarily designed. For this purpose the normal working of the project was demonstrated to the prospective users. Its working is easily understandable and since the expected users are people who have good knowledge of computers, the use of this system is very easy.

7.4 MAINTAINENCE:

This covers a wide range of activities including correcting code and design errors. To reduce the need for maintenance in the long run, we have more accurately defined the user’s requirements during the process of system development. Depending on the requirements, this system has been developed to satisfy the needs to the largest possible extent. With development in technology, it may be possible to add many more features based on the requirements in future. The coding and designing is simple and easy to understand which will make maintenance easier.

7.5 TESTING STRATEGY:

A strategy for system testing integrates system test cases and design techniques into a well-planned series of steps that results in the successful construction of software. The testing strategy must co-operate test planning, test case design, test execution, and the resultant data collection and evaluation. A strategy for software testing must accommodate low-level tests that are necessary to verify that a small source code segment has been correctly implemented as well as high level tests that validate major system functions against user requirements.

Software testing is a critical element of software quality assurance and represents the ultimate review of specification design and coding.

7.5.1 SYSTEM TESTING:

Software once validated must be combined with other system elements (e.g. Hardware, people, database). System testing verifies that all the elements are proper and that overall system function performance is achieved. It also tests to find discrepancies between the system and its original objective, current specifications and system documentation.

7.5.2 UNIT TESTING:

In unit testing different are modules are tested against the specifications produced during the design for the modules. Unit testing is essential for verification of the code produced during the coding phase, and hence the goals to test the internal logic of the modules. Using the detailed design description as a guide, important Conrail paths are tested to uncover errors within the boundary of the modules. This testing is carried out during the programming stage itself. In this type of testing step, each module was found to be working satisfactorily as regards to the expected output from the module. In Due Course, latest technology advancements will be taken into consideration. As part of technical build-up many components of the networking system will be generic in nature so that future projects can either use or interact with this.

7.6 TEST CASES:

Test Case 1: Run the setup.py program	**Priority (H, L):** High
Test Objective: Successfully execute the program and install the required libraries.	
Test Description: Getting the system ready to run the application.	
Requirements Verified: Yes	
Test Environment: Program is to be executed in Anaconda Prompt.	
Test Setup/Pre-Conditions:	

Actions	Expected Results
The user runs the setup file	Successfully Executed
Pass: Yes **Conditions pass:** Yes **Fail**: No	
Problems / Issues: NIL	
Notes: Successfully Executed	

Test Case - 1

Test Case 2: Run the train.py program	**Priority (H, L):** High
Test Objective: Successfully execute the program and open the GUI.	
Test Description: Opening the Graphical User Interface so that user can communicate.	
Requirements Verified: Yes	
Test Environment: Program is to be executed in Anaconda Prompt.	
Test Setup/Pre-Conditions:	
Actions	Expected Results
The user runs the train file	Successfully Executed
Pass: Yes **Conditions pass:** Yes **Fail**: No	
Problems / Issues: NIL	
Notes: Successfully Executed	

Test Case – 2

Test Case 3: User enters alphabets instead of numbers in ID field.	**Priority (H, L):** Medium

Test Objective: Display TypeError and ValueError	
Test Description: Making sure that user enters only numbers in the ID field.	
Requirements Verified: Yes	
Test Environment: Values are to be entered in the UI fields created using tkinter module.	
Test Setup/Pre-Conditions:	
Actions	Expected Results
The user enters alphabets in the ID field.	TypeError and ValueError
Pass: Yes **Conditions pass:** Yes **Fail**: No	
Problems / Issues: NIL	
Notes: Successfully Executed	

Test Case – 3

Test Case 4: User clicks on Take Pictures.	**Priority (H, L):** High
Test Objective: Machine has to capture images and store in a folder.	
Test Description: Making sure that 60 images are captured and stored.	
Requirements Verified: Yes	
Test Environment: Button is to be clicked in the UI created using tkinter module.	
Test Setup/Pre-Conditions:	
Actions	Expected Results
The user clicks the button.	Capture images and store in folder
Pass: Yes **Conditions pass:** Yes **Fail**: No	

Problems / Issues: NIL
Notes: Successfully Executed

Test Case – 4

Test Case 5: User clicks on Train.	**Priority (H, L):** High
Test Objective: Machine trains the image and creates the trainner.yml file.	
Test Description: Making sure that image gets trained.	
Requirements Verified: Yes	
Test Environment: Button is to be clicked in the UI created using tkinter module.	
Test Setup/Pre-Conditions:	
Actions	Expected Results
The user clicks the button.	Image Trained.
Pass: Yes **Conditions pass:** Yes **Fail**: No	
Problems / Issues: NIL	
Notes: Successfully Executed	

Test Case – 5

Test Case 6: User clicks on Detect.	**Priority (H, L):** High
Test Objective: Machine tracks the user and displays details.	
Test Description: Making sure that machine detects the user.	
Requirements Verified: Yes	

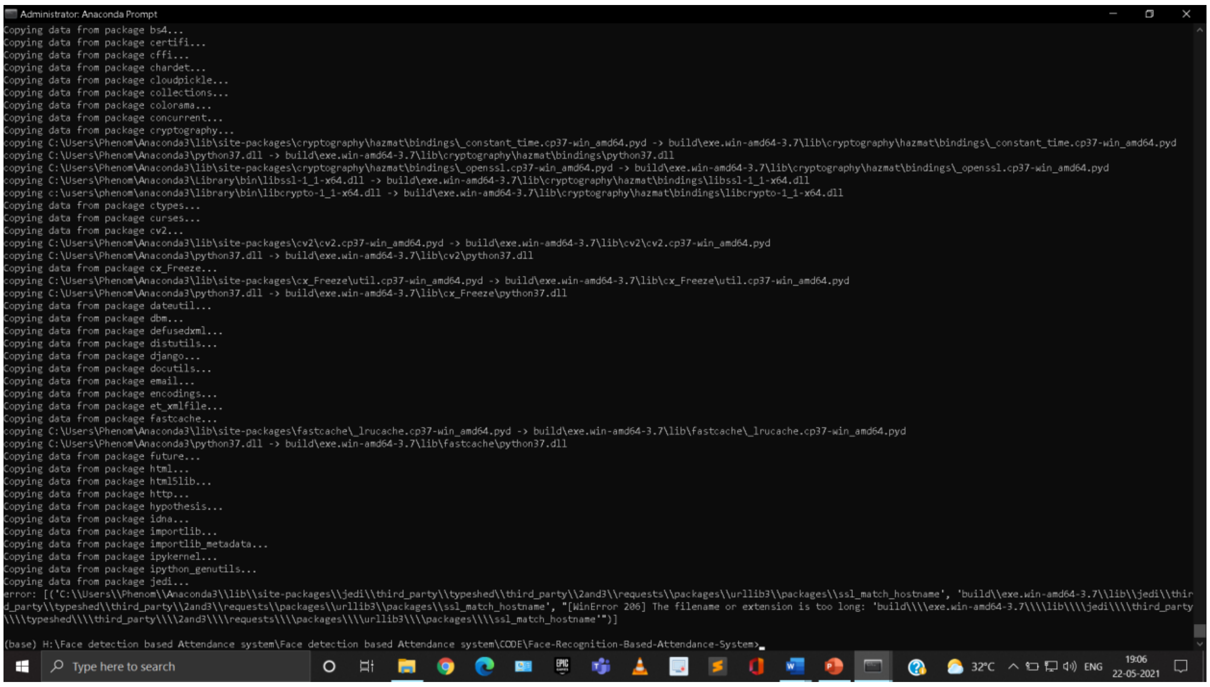
Test Environment: Button is to be clicked in the UI created using tkinter module.	
Test Setup/Pre-Conditions:	
Actions	Expected Results
The user clicks the button.	Tracks the user and logs the attendance.
Pass: Yes **Conditions pass:** Yes **Fail**: No	
Problems / Issues: NIL	
Notes: Successfully Executed	

Test Case – 6

Test Case 7: User clicks on Detect.	**Priority (H, L):** High
Test Objective: Machine tracks the user and displays unknown user.	
Test Description: Making sure that machine detects the unknown user.	
Requirements Verified: Yes	
Test Environment: Button is to be clicked in the UI created using tkinter module.	
Test Setup/Pre-Conditions:	
Actions	Expected Results
The user clicks the button.	Tracks the unknown user image and stores in the folder.
Pass: Yes **Conditions pass:** Yes **Fail**: No	
Problems / Issues: NIL	
Notes: Successfully Executed	

Test Case – 7

8. SCREEN SHOTS



```
Administrator: Anaconda Prompt
Copying data from package bs4...
Copying data from package certifi...
Copying data from package cffi...
Copying data from package chardet...
Copying data from package cloudpickle...
Copying data from package collections...
Copying data from package colorama...
Copying data from package concurrent...
Copying data from package cryptography...
copying C:\Users\Phenom\Anaconda3\lib\site-packages\cryptography\hazmat\bindings\_constant_time.cp37-win_amd64.pyd -> build\exe.win-amd64-3.7\lib\cryptography\hazmat\bindings\_constant_time.cp37-win_amd64.pyd
copying C:\Users\Phenom\Anaconda3\python37.dll -> build\exe.win-amd64-3.7\lib\cryptography\hazmat\bindings\python37.dll
copying C:\Users\Phenom\Anaconda3\lib\site-packages\cryptography\hazmat\bindings\_openssl.cp37-win_amd64.pyd -> build\exe.win-amd64-3.7\lib\cryptography\hazmat\bindings\_openssl.cp37-win_amd64.pyd
copying C:\Users\Phenom\Anaconda3\Library\bin\libssl-1_1-x64.dll -> build\exe.win-amd64-3.7\lib\cryptography\hazmat\bindings\libssl-1_1-x64.dll
copying c:\users\phenom\anaconda3\library\bin\libcrypto-1_1-x64.dll -> build\exe.win-amd64-3.7\lib\cryptography\hazmat\bindings\libcrypto-1_1-x64.dll
Copying data from package ctypes...
Copying data from package curses...
Copying data from package cv2...
copying C:\Users\Phenom\Anaconda3\lib\site-packages\cv2\cv2.cp37-win_amd64.pyd -> build\exe.win-amd64-3.7\lib\cv2\cv2.cp37-win_amd64.pyd
copying C:\Users\Phenom\Anaconda3\python37.dll -> build\exe.win-amd64-3.7\lib\cv2\python37.dll
Copying data from package cx_Freeze...
copying C:\Users\Phenom\Anaconda3\lib\site-packages\cx_Freeze\util.cp37-win_amd64.pyd -> build\exe.win-amd64-3.7\lib\cx_Freeze\util.cp37-win_amd64.pyd
copying C:\Users\Phenom\Anaconda3\python37.dll -> build\exe.win-amd64-3.7\lib\cx_Freeze\python37.dll
Copying data from package dateutil...
Copying data from package dbm...
Copying data from package defusedxml...
Copying data from package distutils...
Copying data from package django...
Copying data from package docutils...
Copying data from package email...
Copying data from package encodings...
Copying data from package et_xmlfile...
Copying data from package fastcache...
copying C:\Users\Phenom\Anaconda3\lib\site-packages\fastcache\_lrucache.cp37-win_amd64.pyd -> build\exe.win-amd64-3.7\lib\fastcache\_lrucache.cp37-win_amd64.pyd
copying C:\Users\Phenom\Anaconda3\python37.dll -> build\exe.win-amd64-3.7\lib\fastcache\python37.dll
Copying data from package future...
Copying data from package html...
Copying data from package html5lib...
Copying data from package http...
Copying data from package hypothesis...
Copying data from package idna...
Copying data from package importlib...
Copying data from package importlib_metadata...
Copying data from package ipykernel...
Copying data from package ipython_genutils...
Copying data from package jedi...
error: [('C:\\Users\\Phenom\\Anaconda3\\lib\\site-packages\\jedi\\third_party\\typeshed\\third_party\\2and3\\requests\\packages\\urllib3\\packages\\ssl_match_hostname', 'build\\exe.win-amd64-3.7\\lib\\jedi\\thir
d_party\\typeshed\\third_party\\2and3\\requests\\packages\\urllib3\\packages\\ssl_match_hostname', "[WinError 206] The filename or extension is too long: 'build\\\\exe.win-amd64-3.7\\\\lib\\\\jedi\\\\third_party
\\\\typeshed\\\\third_party\\\\2and3\\\\requests\\\\packages\\\\urllib3\\\\packages\\\\ssl_match_hostname'")]

(base) H:\Face detection based Attendance system\Face detection based Attendance system\CODE\Face-Recognition-Based-Attendance-System>_
```

Fig-1: After running setup.py program, it builds our machine and installs all the required packages.

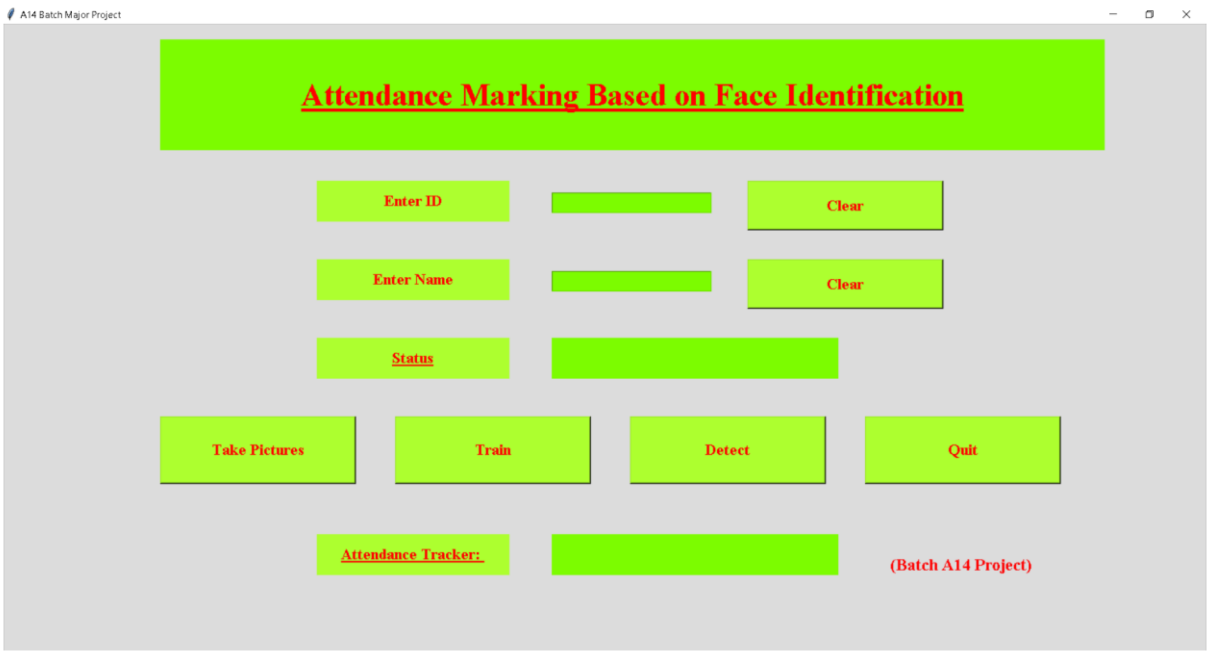
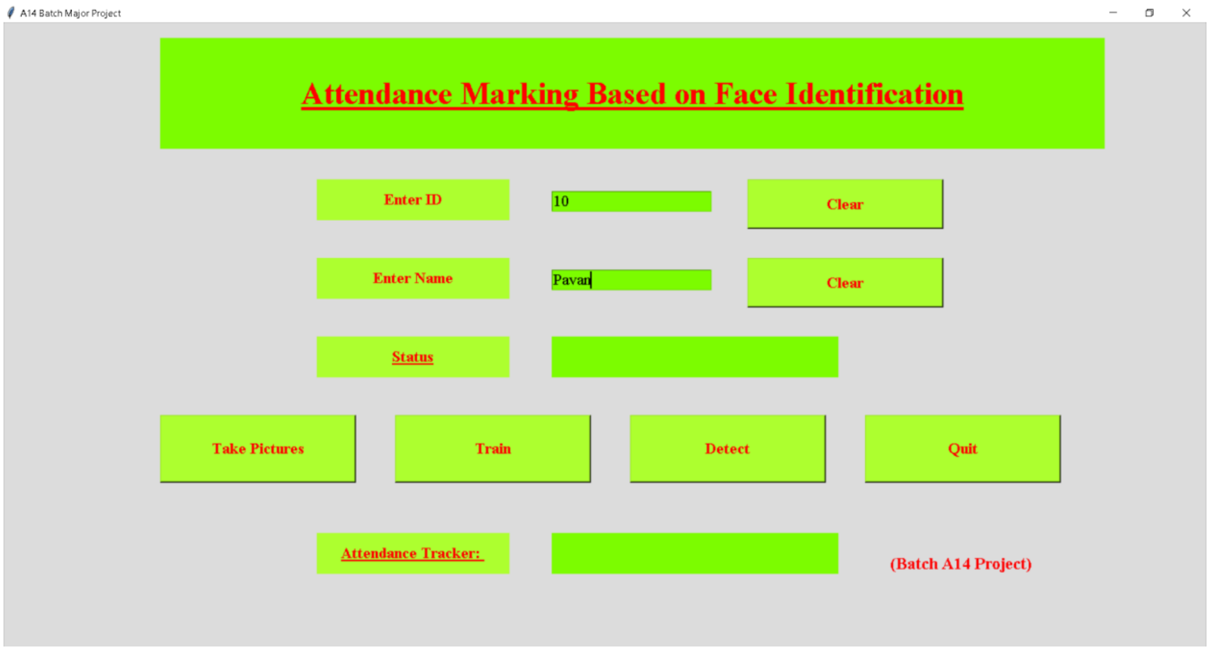


Fig-2: The following GUI is opened after successful execution of the program.


A14 Batch Major Project

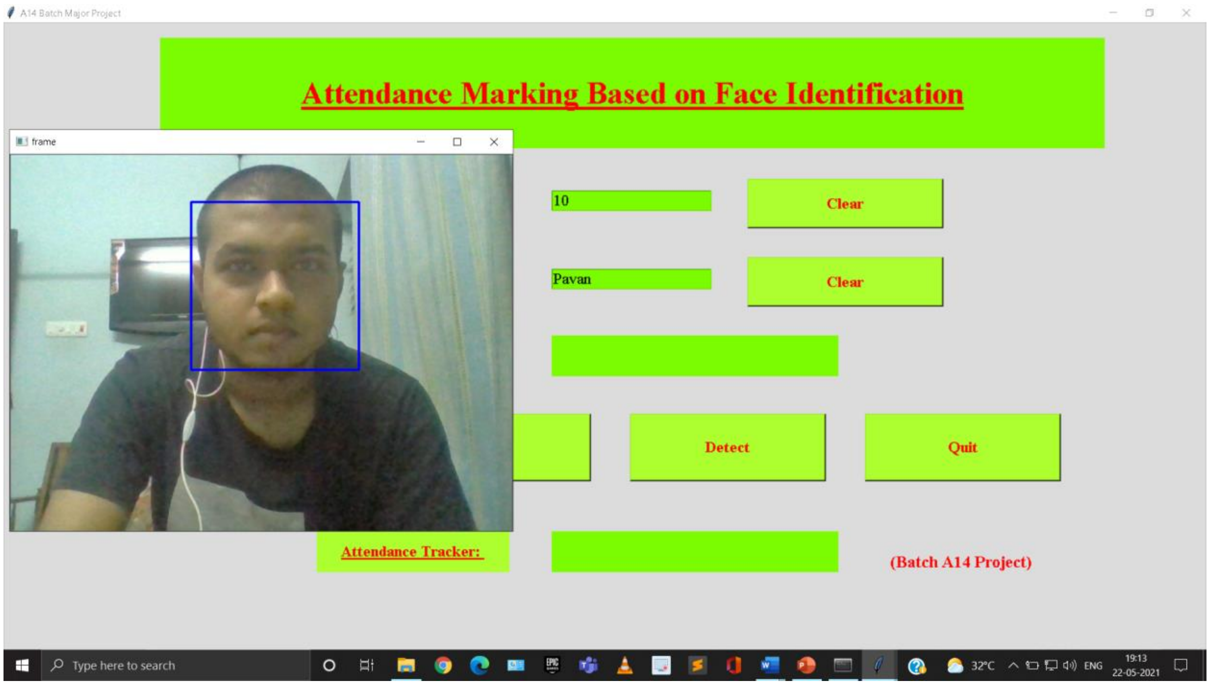
Attendance Marking Based on Face Identification

Enter ID	10	Clear
Enter Name	Pavan	Clear
Status		

Attendance Tracker:

(Batch A14 Project)

Fig-3: User enters details in the fields(ID, Name).


A14 Batch Major Project

Attendance Marking Based on Face Identification

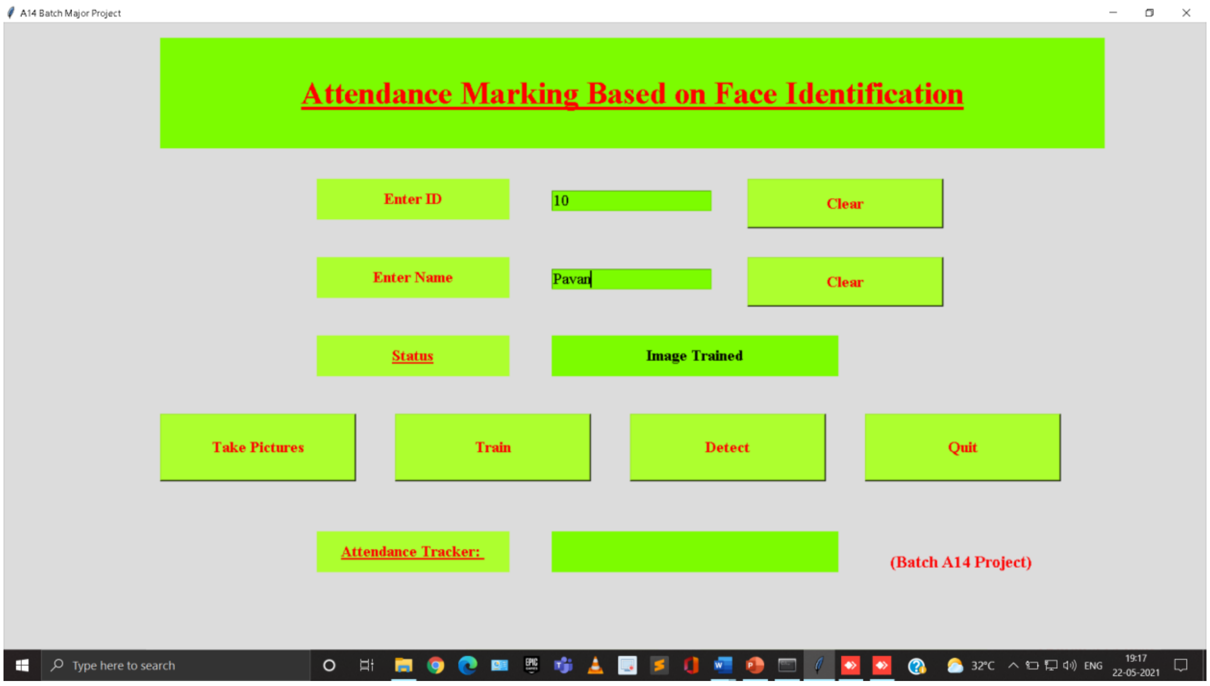
frame

10	Clear
Pavan	Clear

Attendance Tracker:

(Batch A14 Project)

Fig-4: User clicks Take Pictures button and camera captures 60 images of the person.


A14 Batch Major Project

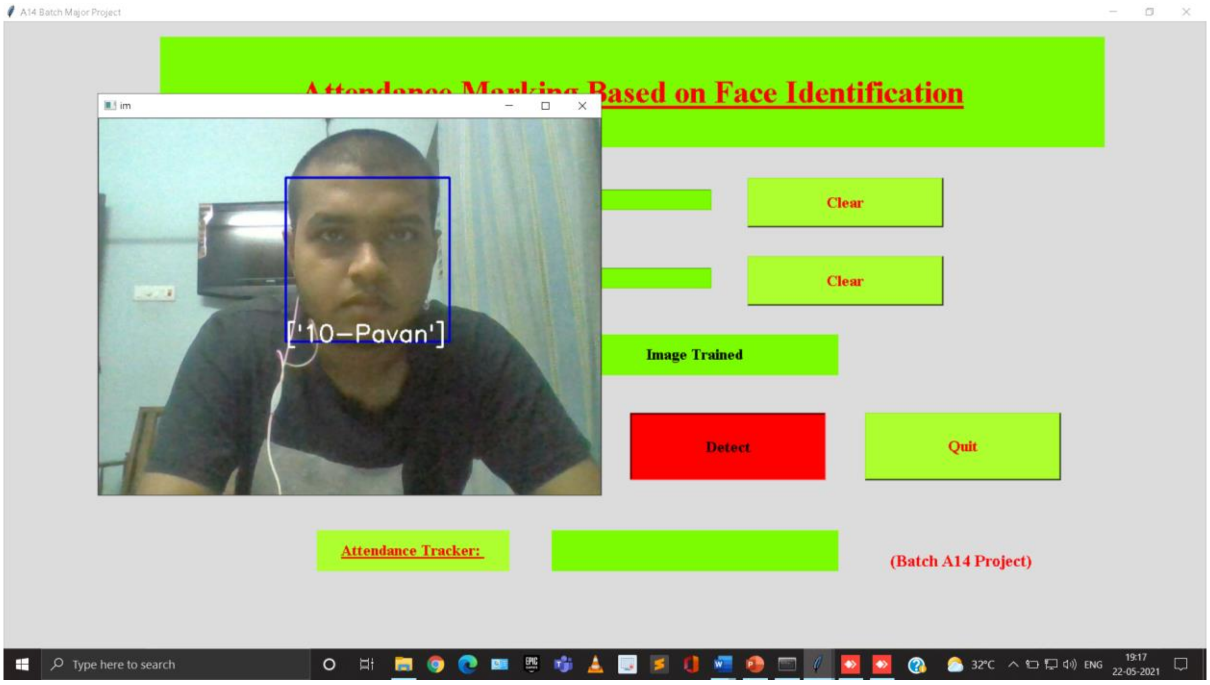
Attendance Marking Based on Face Identification

Enter ID: 10

Enter Name: Pavan

Status: Image Trained

Fig-5: User clicks Train button and image gets trained.

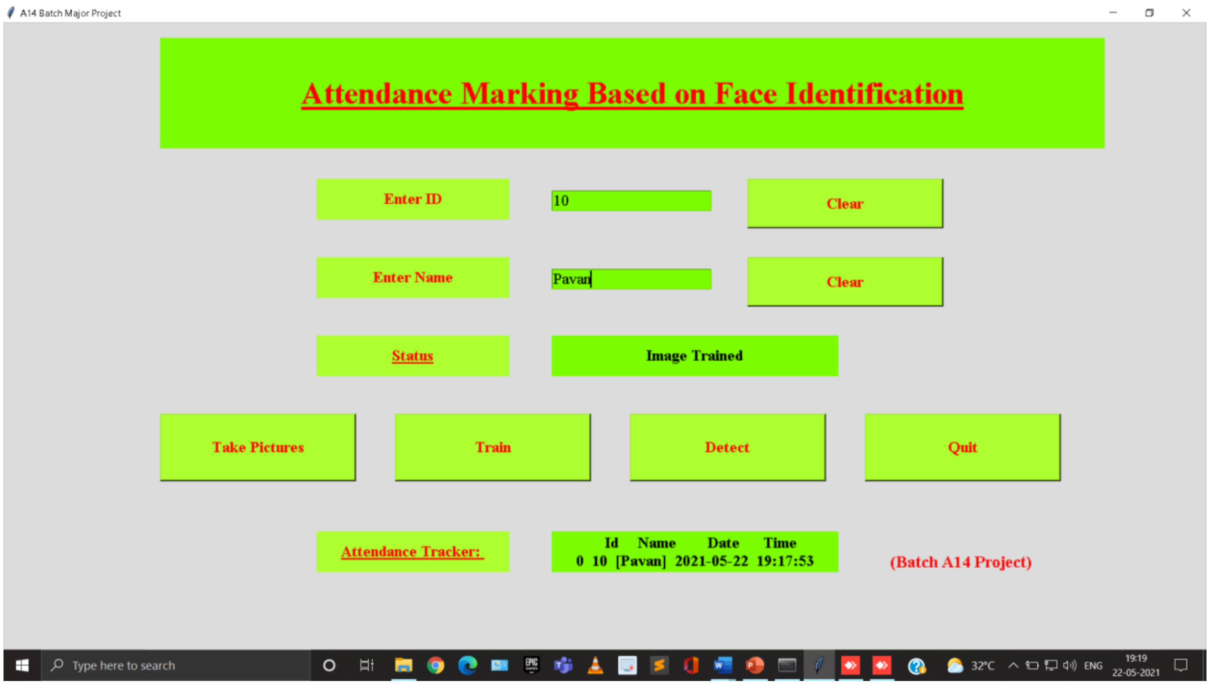

A14 Batch Major Project

Attendance Marking Based on Face Identification

Image Trained

['10–Pavan']

Fig-6: User clicks Detect button and camera tracks the user and displays the details.


A14 Batch Major Project

Attendance Marking Based on Face Identification

Enter ID	10	Clear
Enter Name	Pavan	Clear
Status	Image Trained	

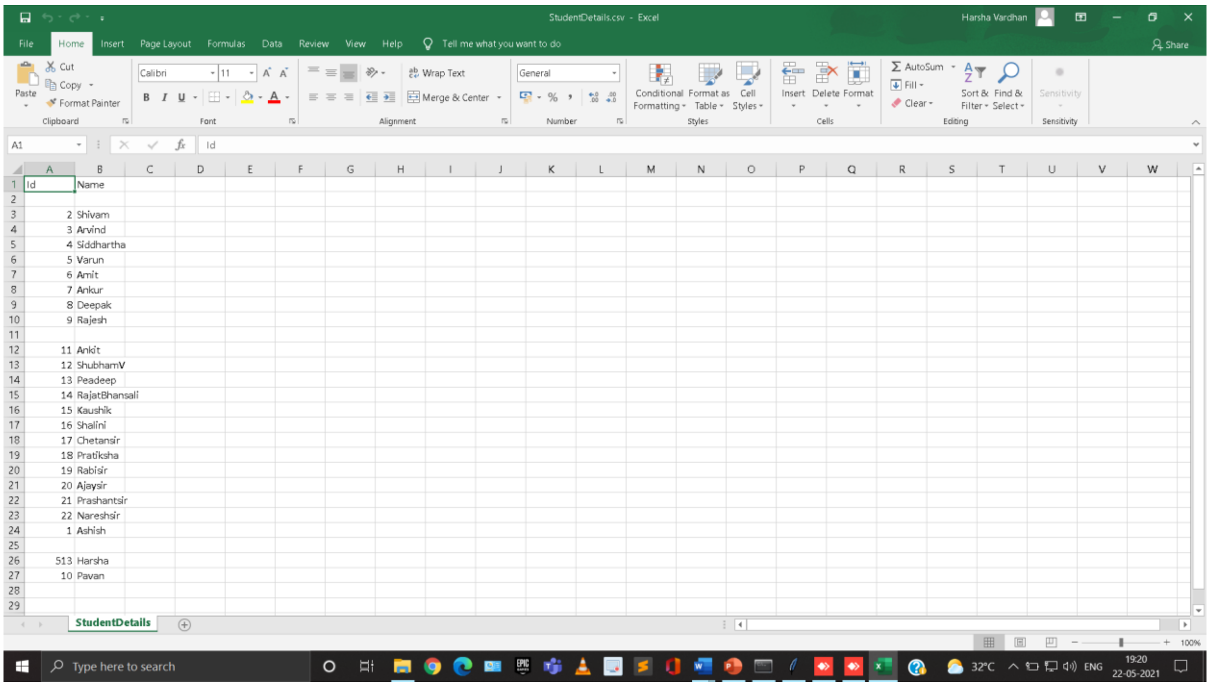
Take Pictures | Train | Detect | Quit

Attendance Tracker:

```
   Id   Name      Date      Time
0  10  [Pavan]  2021-05-22  19:17:53
```

(Batch A14 Project)

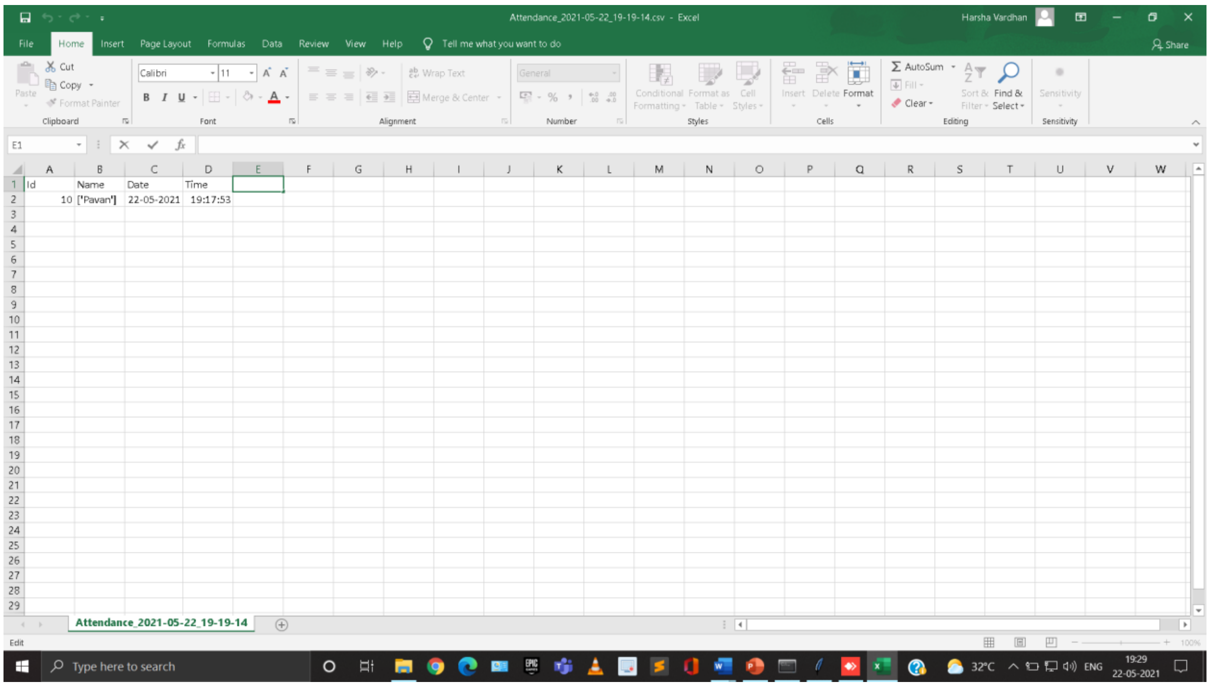
Fig-7: User presses ‘q’ button and his attendance gets recorded


StudentDetails.csv - Excel

Id	Name
2	Shivam
3	Arvind
4	Siddhartha
5	Varun
6	Amit
7	Ankur
8	Deepak
9	Rajesh
11	Ankit
12	ShubhamV
13	Peadeep
14	RajatBhansali
15	Kaushik
16	Shalini
17	Chetansir
18	Pratiksha
19	Rabisir
20	Ajaysir
21	Prashantsir
22	Nareshsir
1	Ashish
513	Harsha
10	Pavan

StudentDetails

Fig-8: Details of user gets appended as a new record in StudentDetails.csv file.


Attendance_2021-05-22_19-19-14.csv - Excel

Id	Name	Date	Time
10	['Pavan']	22-05-2021	19:17:53

Fig-9: Attendance of user gets inserted as a new record in Attendance.csv file along with date and time.

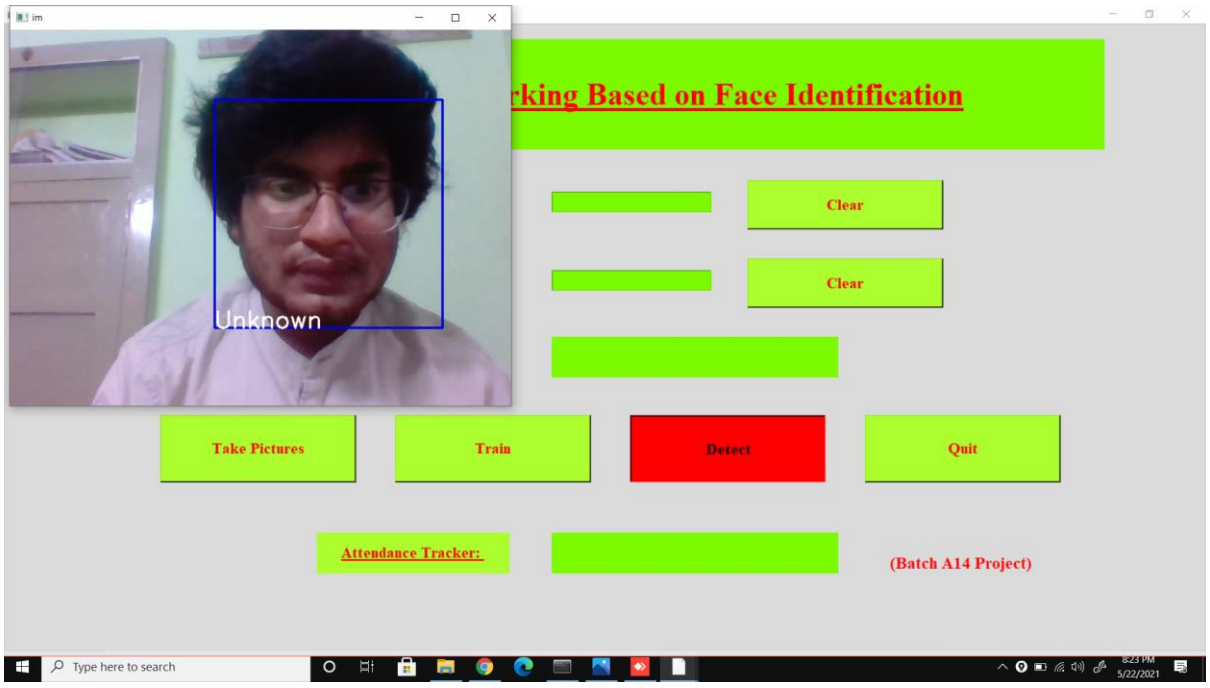


Fig-10: Machine tracks unknown user and stores in the ImagesUnknown folder.

9. CONCLUSION

9.1 CONCLUSION:

Automated Attendance System has been envisioned for the purpose of reducing the errors that occur in the traditional (manual) attendance taking system. The aim is to automate and make a system that is useful to the organization such as an institute. The efficient and accurate method of attendance in the office environment that can replace the old manual methods. This method is secure enough, reliable and available for use. No need for specialized hardware for installing the system in the office. It can be constructed using a camera and computer.

10. REFERENCES

- W. Zhao, R. Chellappa, P. J. Phillips, and A. Rosenfeld,“Face recognition: A literature survey,” ACM Computing Surveys, 2003, vol. 35, no. 4, pp. 399-458.
- Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. Computer Vision and Image Understanding (CVIU), 110(3):346–359.
- H.K.Ekenel and R.Stiefelhagen,Analysis of local appearance based face recognition: Effe cts of feature selection and feature normalization. In CVPR Biometrics Workshop, New York, USA, 2006.
- IJCSI International Journal of Computer Science Issues, Vol. 9, Issue 4, No 1, July 2012 ISSN (Online): 1694-0814.
- Javier Ruiz Del Solar, Rodrigo Verschae, and Mauricio Correa. Face recognition in unconstrained environments: A comparative study. In ECCV Workshop on Faces in RealLife Images: Detection, Alignment, and Recognition, Marseille, France, October 2008.
- Kyungnam Kim “Face Recognition using Principle Component Analysis”, Department of Computer Science, University of Maryland, College Park, MD 20742, USA.
- Osuna, E., Freund, R. and Girosit, F. (1997). "Training support vector machines: an application to face detection." 130-136
- Ahonen, Timo, Abdenour Hadid, and Matti Pietikainen. “**Face description with local binary patterns: Application to face recognition**.” *IEEE transactions on pattern analysis and machine intelligence* 28.12 (2006): 2037–2041.
- Ojala, Timo, Matti Pietikainen, and Topi Maenpaa. “**Multiresolution gray-scale and rotation invariant texture classification with local binary patterns**.” *IEEE Transactions on pattern analysis and machine intelligence* 24.7 (2002): 971–987.
- Ahonen, Timo, Abdenour Hadid, and Matti Pietikäinen. “**Face recognition with local binary patterns**.” *Computer vision-eccv 2004* (2004): 469–481.
- LBPH
  OpenCV: https://docs.opencv.org/2.4/modules/contrib/doc/facerec/facerec_tutorial.html#local-binary-patterns-histograms
- Local Binary Patterns: http://www.scholarpedia.org/article/Local_Binary_Patterns