NEW HORIZON
COLLEGE OF ENGINEERING
Autonomous College Permanently Affiliated to VTU, Approved by AICTE & UGC
Accredited by NAAC with 'A' Grade, Accredited by NBA

A MINI PROJECT REPORT

for

Mini Project using Python (20CSE59)

on

SMART LOCK USING FACE RECOGNITION

Submitted by

R. MANOJ KUMAR
USN: 1NH19CS138, Sem-Sec: 5-c

In partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

ACADEMIC YEAR :2021-22

NEW HORIZON
COLLEGE OF ENGINEERING
Autonomous College Permanently Affiliated to VTU, Approved by AICTE & UGC
Accredited by NAAC with 'A' Grade, Accredited by NBA

CERTIFICATE

This is to certify that the mini project work titled

SMART LOCK USING FACE RECOGNITION

submitted in partial fulfillment of the degree of Bachelor of Engineering in Computer Science and Engineering by

USN:1NH19CS138

DURING

ODD SEMESTER 2021-2022

for

Course: Mini Project using Python-20CSE59

Signature of Reviewer

Signature of HOD

SEMESTER END EXAMINATION

Name of the Examiner *Signature with date*

1._____ _____

2._____ _____

1nh19cs138

ORIGINALITY REPORT

8%	%	8%	%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	R. Jagadheeswaran, R. Arjunan, N. Balamurugan, Barath Kumar D., E. Ramya. "Luggage Theft Identification And Smart Lock Using Face Recognition", 2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS), 2020 Publication	1%
2	Amritha Nag, J N Nikhilendra, Mrutyunjay Kalmath. "IOT Based Door Access Control Using Face Recognition", 2018 3rd International Conference for Convergence in Technology (I2CT), 2018 Publication	1%
3	Firuz Khafizov, Aleksejs Jurenoks, Tahmina Saidova. "Overview of Big Data and Application in Aviation Sector", 2021 62nd International Scientific Conference on Information Technology and Management Science of Riga Technical University (ITMS), 2021 Publication	1%

ABSTRACT

—In recent years, it is important to own a reliable security system that can secure our assets as well as to protect our privacy. The traditional security system needs an individual to use a key, identification (ID) card or password to access an area such as home and workplace. However, the present security system has many weaknesses wherever it is simply cast and taken. Most doors are controlled by persons with the employment of keys, security cards, countersign or pattern to open the door.

The aim of this paper is to assist users for improvement of the door security of sensitive locations by using face detection and recognition. The proposed system mainly consists of subsystems namely image capture, face detection and recognition, email notification and automatic door access management. Face Recognition supported OpenCV is brought up because it uses Eigen faces and reduces the scale of face images without losing vital features, facial images for many persons can be stored in the database.

The door lock can also be accessed remotely from any part of the world by using Telegram android application. The captured image from pi camera will be sent to the authorized person through email for safety purposes.

ACKNOWLEDGEMENT

The satisfaction and euphoria that accompany the successful completion of any task would be impossible without the mention of the people who made it possible, whose constant guidance and encouragement crowned our efforts with success.

I have great pleasure in expressing gratitude to **Dr. Mohan Mangnani**, Chairman, New Horizon Educational Institutions, for providing necessary infrastructure and creating good environment.

I take this opportunity to express my profound gratitude to **Dr. Manjunatha,** Principal, New Horizon College of Engineering, for his constant support and encouragement.

I would like to thank **Dr. Amarjeet Singh**, Professor and Dean-Academics, NHCE, for his valuable guidance.

I would also like to thank **Dr. B. Rajalakshmi**, Professor and HOD, Department of Computer Science and Engineering, for her constant support.

I also express my gratitude to **Dr. Pamela Vinitha Eric**, Professor, Department of Computer Science and Engineering, my mini project reviewer, for constantly monitoring the development of the project and setting up precise deadlines. Her / His valuable suggestions were the motivating factors in completing the work.

R. MANOJKUMAR

1NH19CS138

CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 PROBLEM DEFINITION

Building face recognition door lock using raspberry Pi. The door must open when the authorized user scans his/her face else it must be locked. Face recognition is the task of identifying an already object as a known or unknown face. Often the problem of face recognition is confused with the problem of face detection. Face recognition on the other hand is to decide if the face is someone known, or unknown, using for this purpose a database of faces in order to validate this input face.

1.2 OBJECTIVES

We utilized all the calculations to recognize faces and Eigenfaces calculation to perceive individuals. Appearances were extricated out of pictures and the machine was prepared for certain sure and negative pictures. We at that point effectively tried the framework with various designs and diverse known and obscure individuals. The test outcomes were recorded and we accomplished more than 65% exactness in acknowledgment in fluorescent lighting conditions. Finally, zones of additional advancement were examined. As there is no limit of flawlessness, with more assets and coordination's uphold in addition to sharp perception of its exhibition on various areas, the extent of progress of the framework is yet on table.

1.3 METHODOLOGY

The proposed system was built using a high-performance processor i.e., Raspberry Pi model which runs on a Debian based Linux Operating system called Raspbian. Initially, the PIR sensor senses the presence of human at the door. As and when a human is detected, the Pi camera captures the image of the person and sends the image to the remote user through mail. Haar

face recognition algorithm is run on the captured image using OpenCV in the Raspbian on the basis of the images saved in the system.

1.4 Requirement Specifications

Hardware

1. Raspberry Pi 3
2. Raspberry Pi camera module
3. Solenoid lock

Software

1. OpenCV

CHAPTER 2

FUNDAMENTALS OF PYTHON

2.1 INTRODUCTION TO PYTHON

Python is a commonly and extensively used general-purpose, high-level programming language. Guido van Rossum in 1991 was the founder of Python and was later developed by Python Software Foundation. It was primarily designed to emphasize on code readability, and its syntax allows programmers to express ideas in few lines of code. Python can be used for things like:

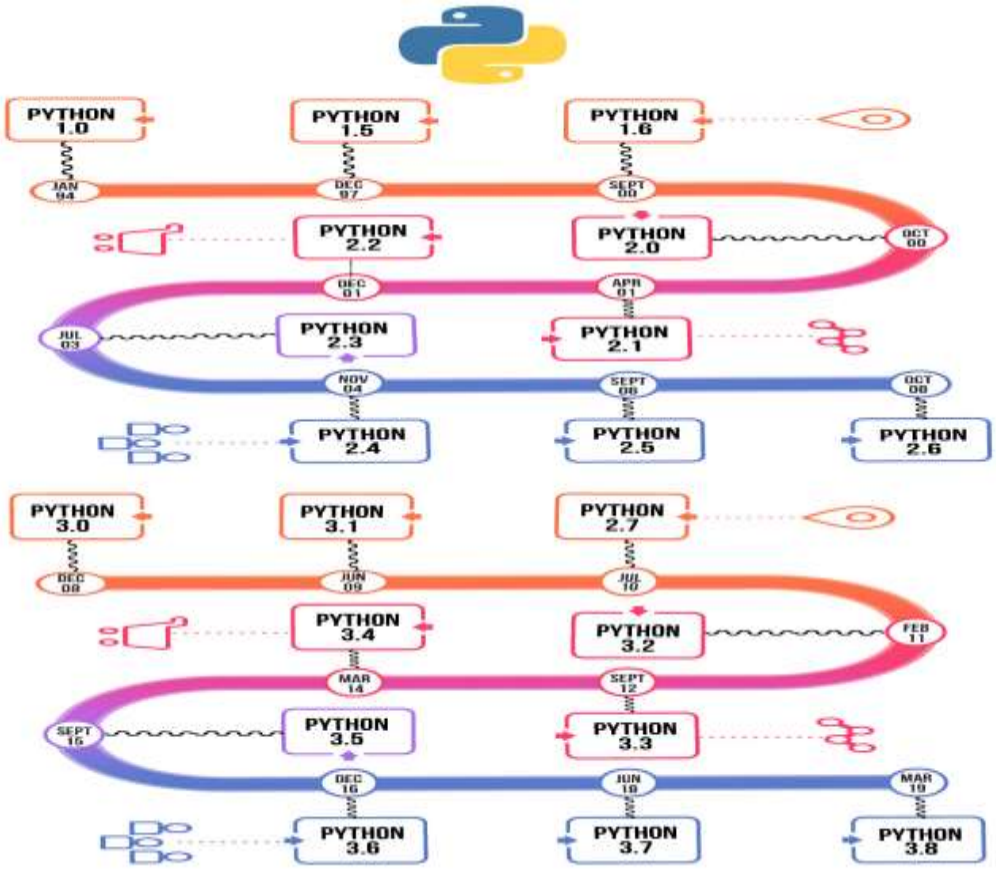

Figure 2.1: Different versions of Python over the years

2.1.1 ENTITY-RELATIONSHIP (ER) MODEL

The Entity-Relationship (ER) Model is an attractive high level conceptual data model. It has an entity which may be an object with a physical existence like a particular car, house, person or employee or it may be an object with a coneptual existence like an organization, a profession, or a university course. Each enity has attribtes—the defnite properies that characterize it. For exmple, a student entty my be described by the student's name, age, address, USN etc.

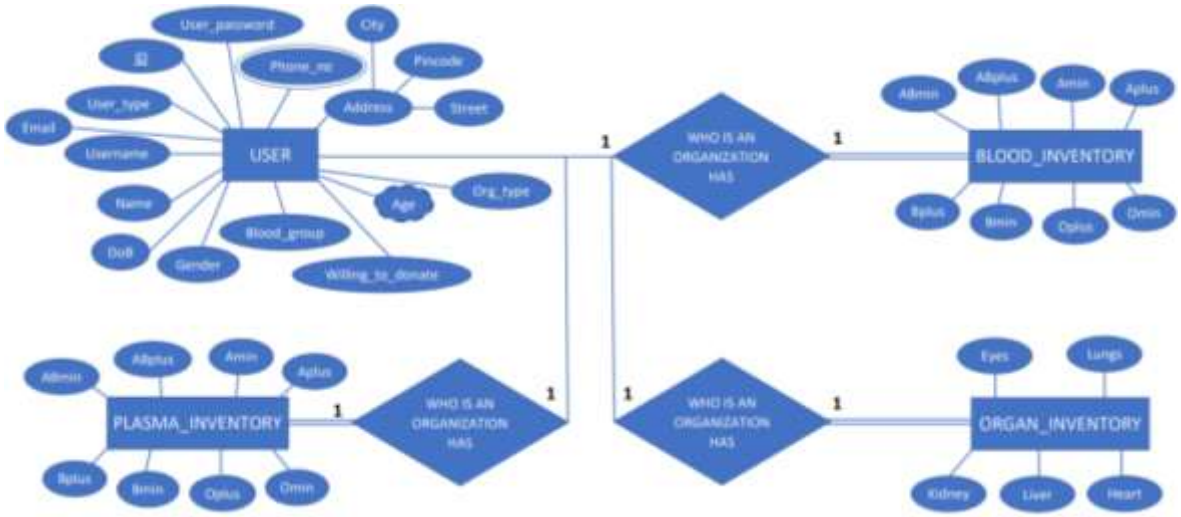

Figure 2.1.1: DBMS Component Modules

2.1.2 ENTITY-RELATIONSHIP (ER) MODEL

The Entity-Relationship (ER) Model is an attractive high level conceptual data model. It has an entity which may be an object with a physical existence like a particular car, house, person or employee or it may be an object with a coneptual existence like an organization, a profession, or a university course. Each enity has attribtes—the defnite properies that characterize it. For exmple, a student entty my be described by the student's name, age, address, USN etc.

2.2 ADVANTAGES OF PYTHON

Advantages include, firstly, the presence of third-party modules, Extensive support libraries (NumPy for numerical computation, Pandas for data analytics, and so on) are available. development-based and open source. Versatile Simple to read, learn, and compose, Data structures that are user-friendly high-level linguistic proficiency level, dynamically coded language (there is no need to specify data type dependent on the value given; it accepts data type) object-oriented programming language. Mobile and interactive, ideal for prototypes since it allows for greater functionality with less code. Python is extremely efficient (Python's clean, object-oriented architecture improves procedures, and the language offers excellent text processing and integration tools and its own unit testing framework, creating a more efficient program).

Opportunities for the Internet of Things (IoT), Language Analysis, Inter interoperability

2.3 DATA TYPES

Data Types Supported Datasets are a fundamental notion in programming.Variables can hold several sorts of data, and various types can perform various functions. Python contains the following data types by default in these categories:

Text Type: str

Numeric Types: int, float, complex

Sequence Types: list, tuple, range

Mapping Type: dict

Set Types: set, frozenset

Boolean Type: bool

Binary Types: bytes, bytearray, memoryview

2.4 PYTHON NUMBERS

The data types like int, float and complex come under the category of Python numbers. Integers in the Python programming language do not have any definite size constraint as seen in other programming languages. They are only limited by the amount of memory available.

Floating point numbers have a precision up to 15 decimal places. A key point to note is that while the number 1 is an integer, 1.0 is treated as a floating number in python. Complex numbers can also be represented in Python. They are given in the format a + bj where a is the real number part and b is the imaginary part.

2.5 PYTHON STRINGS

In Python, a string is a progression or sequence of characters, mostly Unicode characters. Unicode was developed to have every character in every language and bring similarity in encoding.

Characters enclosed between single quote or double quotes create a string. Triple quotes can also be used in Python but usually it is used to signify multiple line strings and function docstrings.

We can access each character in a string using the Python feature called indexing and a range of characters using the Python feature called slicing. Indexing starts from 0. If we try to access a character out of the range of indices, then Python will raise an IndexError. The index must always be an integer. Floats or any other type of data type cannot be used for indexing as this will result into TypeError.

Python also provides negative indexing for its sequences. An index of -1 refers to the last element, -2 to the second last element and so on. One can access a range of elements in a string by using the slicing operator :(colon).

Strings are immutable in the Python programming language. This means that content of a string cannot be changed after it has been assigned. Reassigning can be done by assigning different strings to the same name. The del keyword can be used to delete the entire string from memory but individual characters of the string cannot be deleted or removed.

Concatenation is the process of joining of two or more strings into a single one. The addition (+) operator is used for concatenation in Python. Just writing two string variables together also concatenates them. A string can be repeated multiple time by using the multiply (*) operator along with the string.

2.6 PYTHON LISTS

The list in Python is the closest data type that resembles the array data type seen in other programming languages. It is an ordered sequence of elements. It is one of the widely used data types in Python due to its flexible nature.

A list can have elements of various data types and therefore is both homogenous and heterogenous in nature. It is also mutable, i.e., its content can be changed later. A list is denoted by using the square brackets [ ] with the elements in between the square brackets separated by a comma.

Lists can also be sliced. Extracting a specific sub list from list is called slicing. This is carried out using a colon (:) called the slicing operator.

Lists can also be indexed in two ways:

- Forward Indexing
- Backward Indexing

  Forward indexing is the usual way we index by starting from 0 for the first element. In backward indexing, -1 refers to the last element, -2 to the second last element and so on. We can add an item to a list which we create by using the method append() or add several items using extend() method by providing a list as an argument.

2.7 PYTHON TUPLES

The tuple in Python is very similar to the list data type discussed earlier. The only difference between a list and a tuple is that, while a list is mutable, a tuple is immutable, i.e., its content cannot be changed once initialized.

A tuple is denoted by using the parentheses ( ) with the elements in between the parentheses separated by a comma. However, to create just a tuple with a single element is a bit difficult. Enclosing just the single element in parentheses will not create a tuple. A trailing comma is required after the element to indicate that it is, in fact, a tuple.

The indexing followed in tuples is similar to that followed in lists. Tuples can also be sliced using the colon (:) as seen in lists.

As discussed earlier, tuples are immutable unlike lists. That is, once a tuple is assigned, the elements of a tuple cannot be changed. But, if the element inside the tuple is a mutable Python data type like list, then the nested elements can be changed or are mutable. Reassignment i.e., assigning a tuple to different values is permitted.

2.8 PYTHON SETS

An unordered collection of items is called a set in the python programming language. Every element of the set is unique, that is there are no duplicates and it is immutable. The point to be noted is that a set itself is mutable. We may remove or add elements from it. Set operations like union, intersection, symmetric difference can be performed using sets.

A set is defined by placing all the elements inside curly braces { }, each separated by a comma, or by using Python's built-in set() method. It can have any number of elements and they may be of different types or in other words heterogenous. Importantly, a set cannot have mutable data types like lists, sets or dictionaries as its elements.

2.9 PYTHON DICTIONARIES

An unordered collection of data values, used to store data values like a map is called a dictionary. Unlike other Data Types in the Python programming language that hold only a single value as an element, a dictionary in Python holds key: value pairs as its elements. Key value is provided in the dictionary to make it more optimized.

A dictionary is denoted by using the curly braces { } with the elements in between the parentheses separated by a comma. Each element is a key: value pair.

A dictionary uses keys whereas indexing is used in other data types to access elements or values. Keys can be used to retrieve their corresponding values either inside square brackets [] or with the get() method. A KeyError is raised in case a key is not found when we use the square brackets [] in the dictionary. The get() method returns None if the key is not found unlike the square bracket method.

Dictionaries are mutable, that is its values can be changed even after assignment. Using an assignment operator, we can add new items or change the value of existing items. The existing value gets updated, if the key is already present. A new (key: value) pair is added to the dictionary in case the key is not present.

2.10 FUNCTIONS IN PYTHON

Functions are an easy to use and convenient way or method to divide the Python code into blocks of code thus making the entire code more ordered which enhances or increases the readability and shortens the overall length of the program. Using functions also helps us in reusing the code thus reducing the typing load and saving time.

A function is defined by using the keyword def followed by the function name and a pair of parentheses which encloses the arguments if any. This is followed by the function body which starts after an indentation (usually a tab) from the margin. The function body may also have a return statement that returns an object to where the function was invoked / called.

The portion of a program where the variable is recognized is called the scope of a variable. Variables and parameters defined inside a function are not visible or accessible from outside the function or in other words their scope is limited to within the function. A variable's lifetime is the time period throughout which the variable exists in the memory. Variables inside a function have a lifetime that is as long as the function executes. Once a function is returned, the variables inside them are destroyed. Hence, the value of a variable from a function's previous calls is not remembered by a function.

In python there are 2 types of function as seen in other programming languages:

- Built-in functions: Those functions whose functionality is pre-defined in Python are called as built-in functions. The python interpreter has many predefined functions that are always present for a programmer to use. There are several built-in functions in Python. A few are listed below:
    - abs( ): It takes a single argument and returns the absolute value of the number
    - bin( ): It takes a single argument and returns the binary equivalent of the number
    - sum( ): It can take a list as argument and returns the sum of all elements in the list
    - max( ): It can take a list as argument and returns the maximum element among all elements in the list
    - min( ): It can take a list as argument and returns the minimum element among all elements in the list
    - sorted( ): It sorts the elements of a sequence like a string or a list and returns a collection in the sorted order
    - input( ): This function is used to take in an input from the user. It takes the input as a string.
- User defined functions: These are functions written by a programmer to do a specific task and help reduce the typing load and length of code.

CHAPTER 3

FUNDAMENTALS OF DBMS

This mini project has ensured that the user has an interactive and explorable environment. The interface is user friendly, simple to understand and has tried to ensure that there are no bugs.

Table 3.1: Various widgets available in Tkinter

WIDGETS	DESCRIPTION
Label	This widget is used to display text or image on the window/frame
Button	This widget is used to add buttons to the user interface
Canvas	This widget allows one to draw pictures and different types of layouts like texts, graphics etc.
Entry	This widget is used to take as input, a single line text entry from user
Frame	This widget is used as box or container. It holds and organizes the widgets in an orderly fashion
SpinBox	This widget allows users to select from a given number of values
ComboBox	This widget contains a down arrow to select from a list of options
CheckButton	This widget displays a number toggle buttons which represent various options from which user can select any number of options.
RadioButton	This widget is similar to the CheckButton but allows only one option to be selected
Scale	This widget is used to provide a slider which allows the user to select any value from the scale

Various triggers were also used to ensure referential integrity and data integrity:

- Age_Calc: This trigger calculates the age from the attribute 'date_of_birth' when a new tuple is inserted into the table.

```
CREATE TRIGGER age_calc

AFTER INSERT ON users

FOR EACH ROW

BEGIN

UPDATE users

SET age = CAST (strftime('%Y.%m%d', 'now') - strftime('%Y.%m%d',
new.dob) AS INT)

WHERE ID = new.ID AND new.usertype = "Individual";

END;
```

A collection of logically related data is generally defined as a database. In other words, it is a collection of large volumes of facts and figures in an organized and orderly manner. It has few inherent properties:

- It personifies some facet of the actual world, called the miniworld or universe of discourse.
- It is a collection of data with some inherent meaning where the data is logically coherent.
- A database is designed, developed, and filled with data for a peculiar purpose. It has a targeted group of users. It can be of any size and of differing complexity.

A database management system (DBMS) is a collection of applications that enable users to create and maintain a database. It can be described as an all-purpose software application that provides the platform for defining, constructing, and manipulating databases for various uses. MySQL, Microsoft Access, Sybase, Oracle and IBM DB2 etc. are some examples of popular DBMS software.

3.2 CHARACTERISTICS OF A DBMS

A database management system generally has the following features:

1. Controlling Redundancy: A good DBMS has to ensure minimal or nil redundancy. Redundancy refers to the storing of the exact same data many times. This leads to many issues like duplication of effort, wastage of storage space and inconsistency. Storing each logical data item at only one place in database is generally a feature of an ideal DBMS software.
2. Data Consistency: By managing redundancy of data, the consistency of data is achieved. If a data item is seen only once, then any update to its value has to be done only once and the new value of item is immediately available to all users accessing the database.
3. Data Integration: Data in the database is stored as tables or relations. A single database contains several tables and relationships can be established between tables. This makes it easy to query, i.e., retrieve and update data.
4. Enforcing Integrity Constraints: Consistency rules or Integrity constraints can be enforced to a database so that the actual data can be entered into database. The constraints may be enforced to data within a single record or they may be enforced to relationships between different records. Examples: The examples of integrity constraints are:
    a. 'Marks' in a Grading system cannot be greater than the ‘Maximum Marks’
    b. The age of a person cannot be less than 0
5. Restricting Unauthorized Access: When multiple users jointly use a database, it is likely that some users will not have the permission to access all data in the database. A DBMS has a security and authorization subsystem, which the Database Administrator uses to make accounts and to specify restrictions on the account. The DBMS should then apply these restrictions accordingly.
6. Providing Storage Structures for Efficient Query Processing: Indexes (typically in the form of trees and/or hash tables) are maintained by DBMS softwares that are used to enhance the execution time of queries and updates. The choice of which indexes to make and maintain is a part of the responsibility of the DBA along with the actual database design and tuning. The query processing and optimization module is the one

that is responsible for selecting an optimized query execution process for each query given to the system.

7. Providing Backup and Recovery: A DBMS must have the provision for recovering from hardware or software crashes. The backup and recovery subsystem of the DBMS is the main subsystem that is responsible for recovery of data.
8. Providing Multiple User Interfaces: Because many different types of users with differing levels of knowledge on how to use a database, a DBMS provides a wide variety of interfaces for the user. These include query languages, formstyle interfaces, menu-driven interfaces, programming language interfaces and natural language interfaces.
9. Sharing of Data: Many users can have the permission to access the same part of information at the same time. The remote users can also share the exact data. Similarly, the data of the same database can be shared between various applications and programs.

3.3 DATA MODEL

One basic characteristic of the approach that a database takes is that it provides a certain level of data abstraction by hiding implementation details of storage of data that are not required by most database users. A Data Model is a group of concepts which is used to describe the Database's logical structure i.e., the relationships, data types and constraints that should be applied on the data.

Categories of Data Model:

- High level or conceptual data models: They provide approaches on how users perceive the data.
- Eg. the ER model, which uses concepts such as
- Low level or physical data models: These type of data models provides concepts that characterize the details of how data is stored in the computer's memory. e.g. record orderings, record formats, access path, index etc.
- Representational data models: They provide the characteristics which is in between the above two extremes.

3.4 THREE - SCHEMA ARCHITECTURE

The aim of the 3-schema architecture is to differentiate the user applications and the physical database. In this architecture, schemas can be determined at the following three levels:

Internal level: which has an internal schema

Conceptual level: which has a conceptual schema

External level: which includes a number of user views or external schemas.

The system catalog stores information about all three schemas.

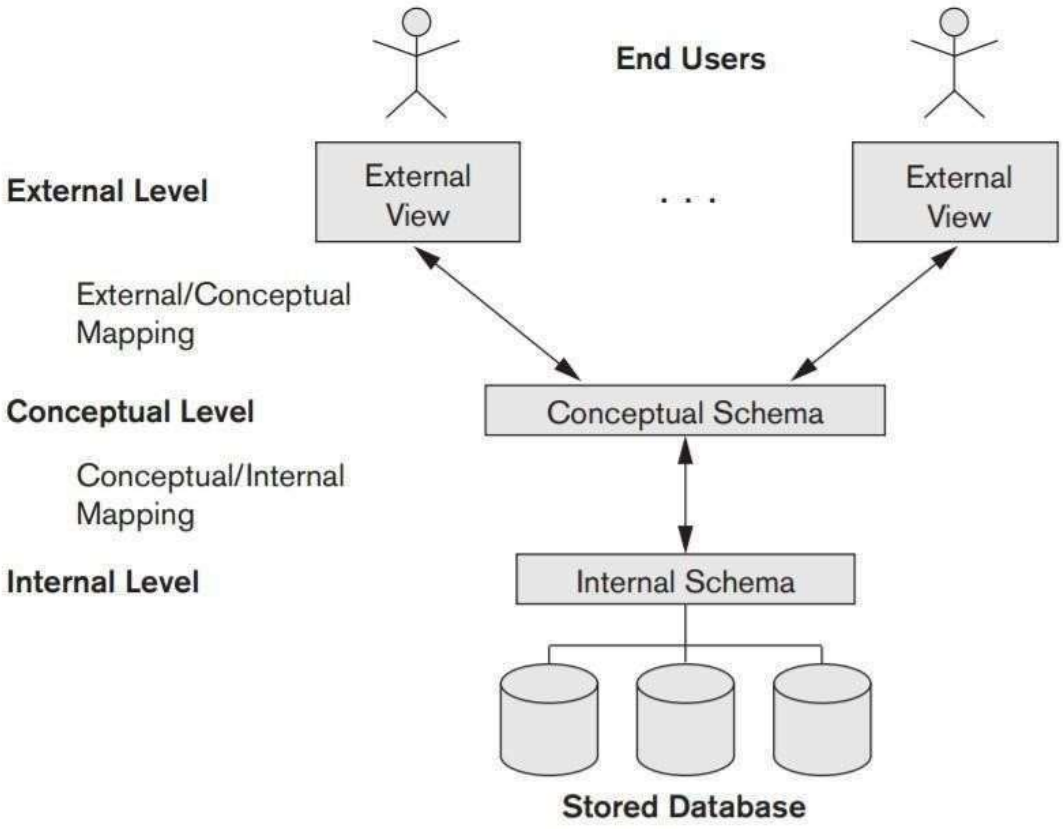

Fig 3.4 : Three-Schema Architecture

3.5 DBMS COMPONENT MODULES

Figure 4 illustrates, in a very simple form, the typical components of a DBMS. The figure is divided into two halves. The top portion of the figure refers to the various users of the database and their respective interfaces. The lower portion shows the various internal modules of the DBMS which are responsible for the processing of transactions and storage of data.

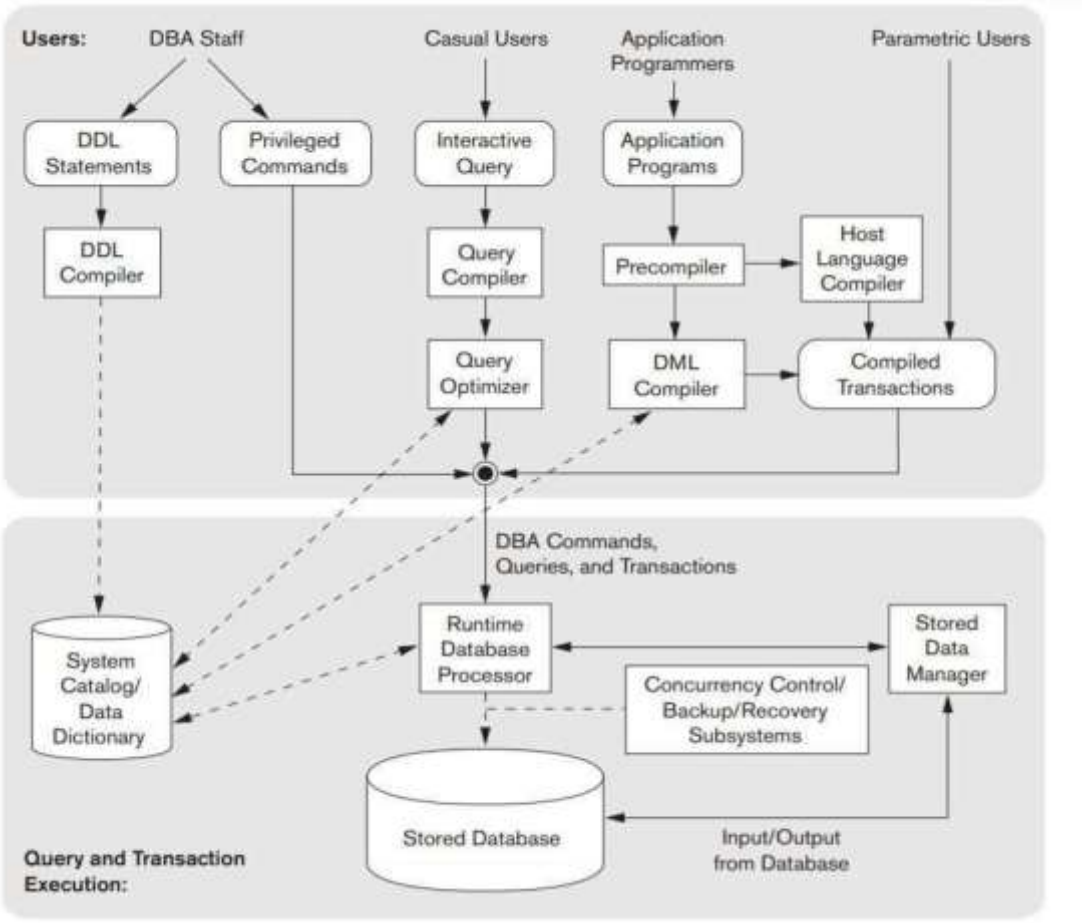

Fig 3.5: DBMS Component Modules

The DBMS catalog and the database are saved and stored on the disk. A module of DBMS named stored data manager manages access to DBMS data that is stored on disk whether it is part of the catalog or database. Various users such as

- Casual users & DBA staff work with interactive interfaces to create queries.

- Application programmers who write programs via a host language.
- Parametric users who make enter data into the database by providing parameters to previously defined transactions.

The DB Administrator staff's job is to define the database and tune it using DDL and other privileged commands.

DDL compiler processes schema definitions which are specified in the Data Definition Language, and stores the description of the database schema in the catalog. Casual users are users who occasionally use the database using interactive query interface. These queries and transactions are parsed and analysed for correctness of the query operations along with the names of data elements by the query compiler that compiles. Then the internal query is optimized by the query optimizer.

The query optimizer performs reordering, rearrangements of operations, deletion of redundancies and usage of correct indexes & algorithms. It looks up the system catalog for information about the storage and for generating executable codes. These executable codes in turn make calls to runtime processor.

Application programmer code programs in host language like COBOL, C and Java that are then submitted to the precompiler. The precompiler extracts the Data Manipulation Language commands from an application program and sends it to the Data Manipulation Language compiler for compilation. The remaining part of the program is sent to the host language compiler. The object codes for the Data Manipulation Language commands and rest of the program are linked, thus creating a canned transaction whose executable code contains calls to the runtime database processor. Parametric users supply the parameters to these canned transactions so that they can run these transactions many times.

Runtime database processor executes

• Privileged commands

• Executable query plans

• Canned transactions with runtime parameters.

• It works along with the system dictionary, stored data manager which in turn uses primitive OS services for carrying out low level input and output operations between the disk and memory.

• It also manages some aspects of control of buffers in the main memory. Some DBMS have their own buffer management. Concurrency control, backup and recovery manager are integrated into the working of the runtime database processor for purposes of transaction management.

3.6 ENTITY-RELATIONSHIP (ER) MODEL

The Entity-Relationship (ER) Model is an attractive high level conceptual data model. It has an entity which may be an object with a physical existence like a particular car, house, person or employee or it may be an object with a con ceptual existence like an organization, a profession, or a university course. Each en tity has attrib utes—the definite properties that characterize it. For ex ample, a student ent ity m ay be described by the student's name, age, address, USN etc.

A relationship is an association among two or more entities. Whenever an attrib ute of one entity. type refers to the atribute of another ent ity type, some relat ionship exists.

For example, in Figure 5, the W . .O.R.K.S._F.O.R relationship type, in which E.M.P.L.O.Y.E.E .s and D.E.P.A.R.T.M.E.N.T.s entities participate. The M.A.N.A.G.E.S relationship type in which E.M .P.L.O.Y.E.E.s and D.E.P.A.R.TM.E.NTs .entities participate.

One or more relationship types can exist with the same participating entity types. For example, MANAGES and .W.ORKS._FOR are distinct relationships between EM. .P.LO.Y.EE and D.E. .PA.RTME.NT, but with different meanings and different relationship instances.

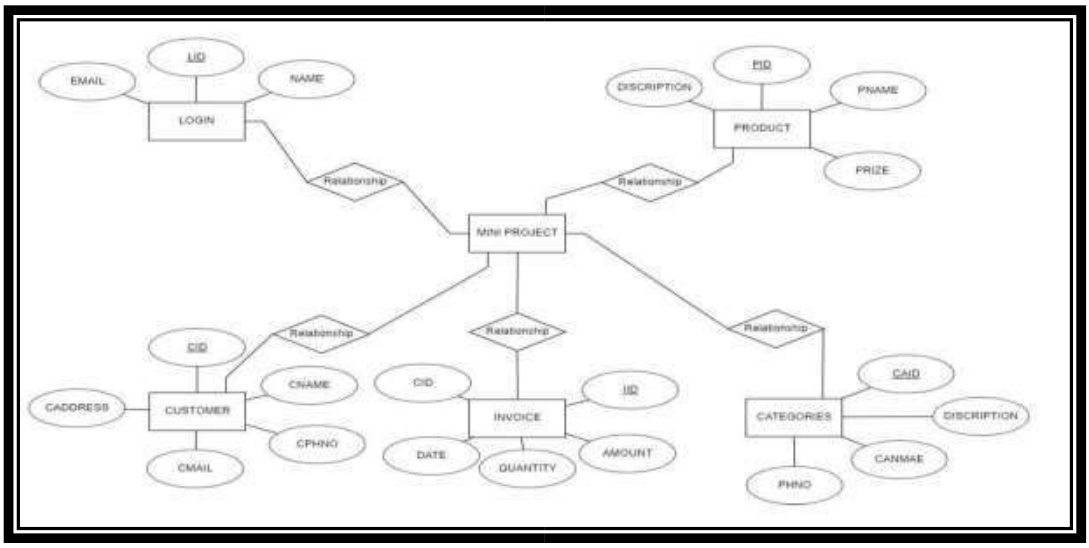

Fig 3.6: Entity-Relationship (ER) Model

3.7 RELATIONAL SCHEMA

A relation schema R, where R is the n.name of the re. lation is denoted by R(A1, A. 2, A3. . . .,An-1, An), where A1, A2, A3. . . .,An-1, An is a list of its attributes. A rel . ation sch ema is . use . d to describe a relation R .

• Where, each Ai is the name of each unique attribute • Do. main of Ai is D and is denoted by dom(Ai). • The number of attributes n in R is the degree of the relation R An example of a relation schema for a relation of degree 5, which describes university students, is the following: STUDENT(Stud_Name, USN, Mobile_Num, Department, DOB) For this relation schema, STUDENT is the na. me of the rel. ation, which has 5 att . ributes.

CHAPTER 4

FUNDAMENTALS OF SQL

4.1 INTRODUCTION

SQL is Structured Query Language, which is a computer language for storing, manipulating and retrieving data stored in a relational database. SQL is the standard language for Relational Data base System. All the Relational Database Management Systems (RDMS) like MySQL, MS Access, Oracle, Sybase, Informix, Postgres and SQL Server use SQL as their standard database language.

SQL is widely popular because it offers the following advantages:

- Allows users to access data in the relational database management systems.
- Allows users to describe the data.
- Allows users to define the data in a database and manipulate that data.
- Allows to be embedded within other languages using SQL modules, libraries & pre-compilers.
- Allows users to create and drop databases and tables.
- Allows users to create view, stored procedure, functions in a database.
- Allows users to set permissions on tables, procedures and views.

4.2 SQL COMMANDS

There are five types of SQL commands:

- DDL: Data Definition Language
- DML: Data Manipulation Language
- DCL: Data Control Language
- TCL: Transaction Control Language

4.3 DATA DEFINITION LANGUAGE

The DDL commands are used for making changes to the structure of a table. DDL commands are by default auto-committed which means that that all changes made in the database using DDL commands are permanent. The DDL commands include:

• ALTER: To alter the structure of the database by adding/deleting a column etc.

• TRUNCATE: To delete all existing content/records while preserving structure

• DROP: To delete both the structure and contents of the table

• CREATE: To create new tables and databases

4.4 DATA MANIPULATION LANGUAGE

The DML commands are used for making changes to the content/records of a table. DML commands are by not default auto-committed which means that that all changes made in the database using DML commands are not permanent and hence can be roll backed. The DML commands include:

• INSERT: To insert a row or value into the table or a specific column of a table

• UPDATE: To update a value of a specific column of a table

• DELETE: To delete rows from the table

4.5 DATA CONTROL LANGUAGE

The DCL commands are used for granting or take back authority from any database user. The DCL commands include:

• GRANT: To grant access privileges to a database user

• REVOKE: To revoke access privileges from a database user

4.6 TRANSACTION CONTROL LANGUAGE

The TCL commands are used only along with commands belonging to the DML category like INSERT, UPDATE and DELETE. TCL commands include:

• COMMIT: To ensure that all transactions processed are saved to the database

• ROLLBACK: To undo all transactions that have processed but not saved onto the database

• SAVEPOINT: To roll back the transactions to a certain point but not entirely like the ROLLBACK command does.

CHAPTER 5
5.1 DESIGN GOALS

The proposed system was built using a high-performance processor i.e., Raspberry Pi model which runs on a Debian based Linux Operating system called Raspbian. Initially, the PIR sensor senses the presence of human at the door.

As and when a human is detected, the Pi camera captures the image of the person and sends the image to the remote user through mail. Haar face recognition algorithm is run on the captured image using OpenCV in the Raspbian on the basis of the images saved in the system.

If the face is recognized it implies that an authorized person is trying for the door access and hence, the door lock is opened.

If the face is not recognized, then the remote user can check the mail for the image of the person trying to access the door and allow or deny the access of the door through an android application- Telegram.

If the user sends" allow" from the Telegram app, the door is opened and if the user sends" deny" from the Telegram app, the person trying to access the door is denied from accessing it.

CHAPTER 6

IMPLEMENTATION

```
import cv2
import os

# Available at https://github.com/opencv/opencv/blob/master/data/haarcascades/haarcascade_frontalface_default.xml
face_detector = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
# For each person, enter one numeric face id
user_name = input('\n enter user name and press <return> ==>  ')
# Initialize individual sampling face count
count = 0
# Check if the dataset folder exists in current working directory. If not create it.
if not os.path.exists('dataset'):
    os.makedirs('dataset')
# Get the length of the folder
length = len(os.listdir('dataset'))
id = length
# create a folder with length of the dataset folder
os.makedirs('dataset/' + str(id))
print("\n [INFO] Initializing face capture. Look the camera and wait ...")
# Initialize the camera
cam = cv2.VideoCapture(0)

while(True):
    ret, img = cam.read()
    if not ret:
        continue
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    faces = face_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)
        count += 1
        # Save the captured image into the datasets folder with the name of the user id
        cv2.imwrite("dataset/" + str(id)+"/" + str(user_name) + '.' +
                    str(count) + ".jpg", gray[y:y+h, x:x+w])
        cv2.imshow('image', img)
    k = cv2.waitKey(100) & 0xff  # Press 'ESC' for exiting video
    if k == 27:
        break
    elif count >= 100:
        break
# Do a bit of cleanup
print("\n [INFO] Exiting Program and cleanup stuff")
userIdLen = len(os.listdir('dataset/' + str(id)))
if userIdLen < 50:
    os.rmdir('dataset/' + str(id))
    print("\n [INFO] User not registered. Try again.")
cam.release()
cv2.destroyAllWindows()
```

Figure 6.1: Creating a Data Base

```
import cv2
import numpy as np
from PIL import Image
import os
# Path for face image database
path = 'dataset'
recognizer = cv2.face.LBPHFaceRecognizer_create()
detector = cv2.CascadeClassifier("haarcascade_frontalface_default.xml")


# function to get the images and label data
def getImagesAndLabels(path):
    userIdPaths = [os.path.join(path, f) for f in os.listdir(path)]
    faceSamples = []
    ids = []
    for idPath in userIdPaths:
        imagePaths = [os.path.join(idPath, f) for f in os.listdir(idPath)]
        for imagePath in imagePaths:
            PIL_img = Image.open(imagePath).convert('L')
            img_numpy = np.array(PIL_img, 'uint8')
            # 'dataset\\0' ⇒ ['dataset', '0']
            id = int(idPath.split("\\")[1])
            faces = detector.detectMultiScale(img_numpy)
            for (x, y, w, h) in faces:
                faceSamples.append(img_numpy[y:y + h, x:x + w])
                ids.append(id)
    return faceSamples, ids


print("\n [INFO] Training faces. It will take a few seconds. Wait ...")
faces, ids = getImagesAndLabels(path)
recognizer.train(faces, np.array(ids))
# Save the model into trainer.yml
recognizer.write('trainer.yml')
# Print the numer of faces trained and end program
print("\n [INFO] {0} faces trained. Exiting Program".format(
    len(np.unique(ids))))
```

Figure 6.2: Connecting the Data Base to the Application

```python
import cv2
# OS import is used to access any os instance.(accessing files, directories, etc.)
import os
# GPIO is used to control the door lock.
# Emulation is used to emulate the door lock.
from GPIOEmulator.EmulatorGUI import GPIO
# This is the class that will be used to recognize the user's face.
import time

# Relay is used for the door lock relay in Emulator.
relay = 23
# GPIO configurations
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)
GPIO.setup(relay, GPIO.OUT)
GPIO.output(relay, 1)
# Loading the face cascade classifier.
recognizer = cv2.face.LBPHFaceRecognizer_create()
# Reading the trained data from the file.
recognizer.read('trainer.yml')
cascadePath = "haarcascade_frontalface_default.xml"
# Loading the cascade classifier.
faceCascade = cv2.CascadeClassifier(cascadePath)
font = cv2.FONT_HERSHEY_SIMPLEX

# Initialize and start realtime video capture
cam = cv2.VideoCapture(0)
names = []
# get the length of the names list
length = len(os.listdir('dataset'))
# Loop through the names list and append the names to the names list
for i in range(length):
    name = os.listdir('dataset/' + str(i))[0].split('.')[0]
    names.append(name)

# Loop till the user presses the Esc key.
while True:
    # Read the video frame
    ret, img = cam.read()
    if not ret:
        continue
    # Convert the captured frame into grayscale
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    # Get all face from the video frame
    faces = faceCascade.detectMultiScale(gray, 1.3, 5)
    # For each face in faces
    for(x, y, w, h) in faces:
```

Figure 6.3: Recognizer Set

```
        # Create rectangle around the face
        cv2.rectangle(img, (x, y), (x+w, y+h), (0, 255, 0), 2)
        # Predict the image using our face recognizer
        id, confidence = recognizer.predict(gray[y:y+h, x:x+w])
        # Check if confidence is less them 100 ==> "0" is perfect match
        if (confidence < 100):
            # Check if the name is in the names list
            id = names[id]
            # Put text describe who is in the picture with percentage
            confRound = round(100-confidence)
            if confRound > 30:
                confidence = "==> {0}%".format(confRound)
                # Open the door if the confidence is greater than 30.(Just for testing)
                # If we want to increase the confidence threshold, we can do it here.
                # But we need to increase the threshold in the trainer.py file.
                # That means we need to increase data set size.
                GPIO.output(relay, 0)
                print(str(confidence) + " sure user ID is " + str(id))
                cv2.putText(img, str(id), (x+5, y-5),
                            font, 1, (255, 255, 255), 2)
                cv2.putText(img, str(confRound), (x+5, y+h-5),
                            font, 1, (255, 255, 0), 1)
                # Hold the door open for 2 seconds.
                time.sleep(2)
            GPIO.output(relay, 1)
        # Handle if the confidence is less than 30
        else:
            id = "unknown"
            confidence = "  {0}%".format(round(100 - confidence))
            GPIO.output(relay, 1)
            cv2.putText(img, str(id), (x+5, y-5), font, 1, (255, 255, 255), 2)
            cv2.putText(img, str(confidence), (x+5, y+h-5),
                        font, 1, (255, 255, 0), 1)
    # Display an image in the specified window
    cv2.imshow('camera', img)
    # Wait for the user to press some key
    k = cv2.waitKey(10) & 0xff
    # Press 'ESC' for exiting video
    if k == 27:
        break
# Do a bit of cleanup
print("\n [INFO] Exiting Program and cleanup stuff")
cam.release()
cv2.destroyAllWindows()
```

Figure 6.3: Recognizer Set

CHAPTER 7

RESULTS

7.1 Capturing and Saving into Dataset

```
PS D:\programs\Python\smart-door-lock> py .\recognizer.py^C
PS D:\programs\Python\smart-door-lock> ./start.bat

 enter user name and press <return> ==>  anju

 [INFO] Initializing face capture. Look the camera and wait ...
```

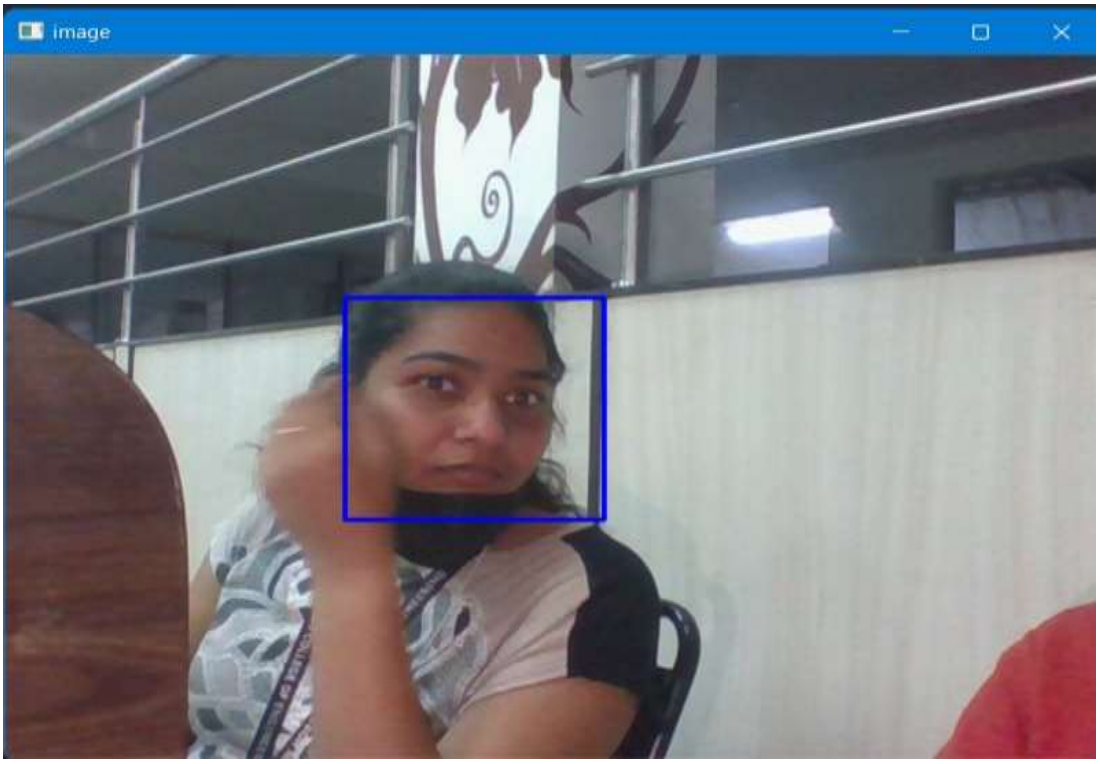

Figure 7.1: Registering a New User

7.2 Training Faces

```
PS D:\programs\Python\smart-door-lock> py .\trainer.py

 [INFO] Training faces. It will take a few seconds. Wait ...
```

Figure 7.1: Training faces of New User

7.3 Recognition

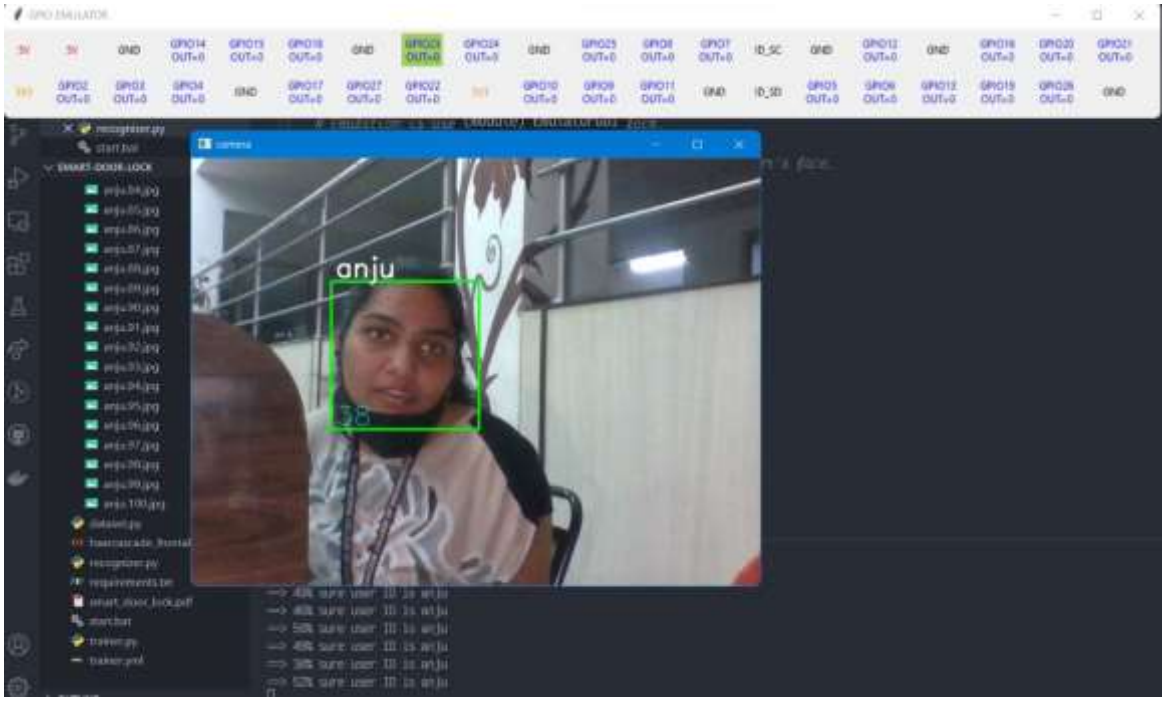

Figure 7.3: Recognition of User(trained faces)

CHAPTER 8

CONCLUSION

The project is good example of Raspberry pi and pi camera with Open CV. A face recognition system using Raspberry Pi was developed. The system was programmed by Python programming language. Both Real times face recognition from specific images, i.e. Stored images. The efficiency of the system was analyzed in terms of face recognition rate. The analysis revealed that the present system shows excellent performance efficiency. Often the problem of face recognition is confused with the problem of face detection. Face recognition on the other hand is to decide if the face is someone known, or unknown, using for this purpose a database of faces in order to validate this input face.

REFERENCES

[1] Hteik Htar Lwin, Aung Soe Khaing, Hla Myo Tun, Automatic Door Access System Using Face Recognition International Journal Of Scientific Technology Research,Issue 06, Volume 4,June 2015.

[2] Prathamesh Timse, Pranav Aggarwal, Prakhar Sinha, Neel Vora, Face Recognition Based Door Lock System Using Opencv and C with Remote Access and Security Features, Int. Journal of Engineering Research and Applications, Vol. 4, Issue 4(Version 6), April 2014, ISSN : 2248-9622.

[3] Faisal, F., & Hossain, S. A. (2019). Smart Security System Using Face Recognition on Raspberry Pi. 2019 13th International Conference on Software, Knowledge, Information Management and Applications (SKIMA). doi:10.1109/skima47702.2019.8982466 .