A PROJECT REPORT ON “Automated Face Recognition Based Monitoring System”

Submitted in partial fulfillment of the

Requirement for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

INFORMATION AND TECHNOLOGY

Submitted By

Sumit Kumar Dubey (U42000000871)

Ankit Sharma (U42000000854)

Under the Guidance of

Prof. Jigar Pandya

Rai University
EVOLVING THINKING MINDS
A LEADING RESEARCH DRIVEN UNIVERSITY

May - 2020

DEPARTMENT OF COMPUTER SCIENCE
& INFORMATION TECHNOLOGY
SCHOOL OF ENGINEERING & APPLIED SCIENCE
RAI UNIVERSITY, SARODA, DHOLKA-382225
DIST: AHMEDABAD, GUJARAT

SCHOOL OF ENGINEERING & APPLIED SCIENCE
RAI UNIVERSITY

CERTIFICATE

This is to certify that **Mr. Sumit Kumar Dubey** of B.Tech. Semester VIII (Information Technology) has completed his final year project work titled "**Automated Face Recognition based Monitoring System**" satisfactorily in partial fulfillment of requirement of **Bachelor of Technology** degree of **Information Technology**, **Rai University, Saroda, Ahmedabad** in the year 2020.

Internal Guide	**HOD**
Prof. Jigar Pandya	Prof. Rajdipsinh Vaghela
(Assistant professor)	
CSE/IT Department	**CSE/IT Department**
Date:	**Date:**

SCHOOL OF ENGINEERING & APPLIED SCIENCE
RAI UNIVERSITY

CERTIFICATE

This is to certify that **Mr. Ankit Sharma** of B.Tech. Semester VIII (Computer Science Engineering) has completed his final year project work titled "**Automated Face Recognition based Monitoring System**" satisfactorily in partial fulfillment of requirement of **Bachelor of Technology** degree of **Computer Science**, **Rai University, Saroda, Ahmedabad** in the year 2020.

Internal Guide	**HOD**
Prof. Jigar Pandya	Prof. Rajdipsinh Vaghela
(Assistant professor)	
CSE/IT Department	**CSE/IT Department**
Date:	**Date:**

ACKNOWLEDGEMENT

Apart from the efforts of ours, the success of any project depends largely on the encouragement and guidelines of many others. We take this opportunity to pen down and express our gratitude to the people who have been instrumental in the successful completion of this project.

During this Project, we gained a good experience of real work, which is completely different from the academics. Moreover, this Project can be considered as the final step towards achieving the bachelor’s degree. We express our sincere thanks to many people who have helped us during our training.

We are grateful to our college, **School of Engineering & Applied Science, Rai University** for giving us this wonderful opportunity to undertake this training.

We convey our sincere thanks to **Mr. Rajdipsinh Vaghela (HOD CSE/IT Department, SEAS, Rai University)**, **Ms. Jigar Pandya (Our Internal Project Guide)** for providing us her time, constructive criticism and valuable guidance. The guidance and support received from all the other faculty members who contributed to this project, was vital for the success of the project. Besides this, we would like to thank all those people who helped us in one or the other way.

CANDIDATE DECLARATION

We hereby declare that, the project report titled **“Automated Face Recognition Based Monitoring System”** submitted towards the completion of Project in 8th semester, of the Bachelor of Technology in Information Technology Engineering , in School of Engineering & Applied Sciences, Rai University, Ahmedabad, is an authentic record of our work carried out.

We further declare that the work reported in this project has not been submitted and will not be submitted, either in part or in full for the award of any other degree in this institute or any other institute.

Candidate’s Name	: **Sumit Kumar Dubey**
Branch	: **B. Tech IT**
Enrollment No	: **U42000000871**

Candidate’s Signature

This is to certify that the above statement made by the candidates is correct to the best of my knowledge.

Signature of Internal Guide
(Prof. Jigar Pandya)

CANDIDATE DECLARATION

We hereby declare that, the project report titled **"Automated Face Recognition Based Monitoring System"** submitted towards the completion of Project in 8th semester, of the Bachelor of Technology in Computer Science Engineering , in School of Engineering & Applied Sciences, Rai University, Ahmedabad, is an authentic record of our work carried out.

We further declare that the work reported in this project has not been submitted and will not be submitted, either in part or in full for the award of any other degree in this institute or any other institute.

Candidate's Name	: **Ankit Sharma**
Branch	: **B. Tech CSE**
Enrollment No	: **U42000000854**

Candidate's Signature

This is to certify that the above statement made by the candidates is correct to the best of my knowledge.

Signature of Internal Guide
(Prof. Jigar Pandya)

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Security is one of the biggest concerns of this technology driven era. With increase in the available resources one has to keep track, for that we use multiple channels to be up to date with those logs. Channels like CCTV which are widely used till now to monitor the track. This is one of the most accurate techniques which is widely used. But one of the exhausting necessities of this channel is manpower and separate hardware resources.

This system aims towards automatic monitoring of the room and activities of a person in the room. It monitors faces with a camera and creates a log file which contains the record of people and their time spent in the room. It helps in keeping track of specific persons and their activity in real time and provide customized data as per requirement as and when needed.

The project **"Automated Face Recognition based Monitoring System"** is a Python based application aimed towards the modern way of monitoring or surveillance system. This system work on the basis of face recognition and face detection. After the successful recognition it will generate a log file that will show the daily record of that person in the premises.

CHAPTER 1: INTRODUCTION

1.1 PROJECT SUMMARY

In this fast pacing, never stopping world security has become a major concern of our day to day lives. We are concerned at who is sneaking in our privacy or property to steal, exploit or destroy our belongings or other things. Keeping that in mind we always focus on monitoring our premises it could be of any type in which the most popular is the Closed-Circuit Television (CCTV) camera as a means of monitoring. But it takes time and man to analyze the daily activities. So, keeping that in mind we are going to introduce a system named **"Automated Face Recognition based Monitoring System"** which is able to generate a daily record on the basis of recognition of face.

1.2 PURPOSE

"Automated Face Recognition based Monitoring System" helps in maintaining a record of persons entered in the premises by identifying the face and recording the time of that person spent in the premises. Which is then further used to verify any condition just by simply entering the date and name of the person. And by implementing this we don't have to put up physical person to monitor the area. The system can automatically keep track.

1.3 SCOPE

In today's world we are equipped with great technologies and resources that can be accessed very easily from anywhere. This system is able to polish this available resource and perform the manual work in an automated way. Our system can help in maintaining a log information of user which contain the daily track of their people. This will help the authorities to keep track of activities in question by simply looking up to log file. Instead of looking for the whole story which becomes an efficient way to look up.

1.4 TOOLS AND TECHNOLOGIES USED

1.4.1 Python

Python is an interpreted, high-level, general-purpose programming language. Created by Guido van Rossum and first released in 1991, Python's design philosophy emphasizes code readability with its notable use of significant whitespace. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects.

1.4.2 PyCharm IDE

PyCharm is an integrated development environment (IDE) used in computer programming, specifically for the Python language. It is developed by the Czech company JetBrains. It provides code analysis, a graphical debugger, an integrated unit tester, integration with version control systems (VCSes), and supports web development with Django as well as Data Science with Anaconda.

[Fig. 1.4.5 PyCharm]

CHAPTER 2: PROJECT MANAGEMENT

Project management is the discipline of initiating, planning, executing, controlling, and closing the work of a team to achieve specific goals and meet specific success criteria. A project is a temporary endeavor designed to produce a unique product, service or result with a defined beginning and end undertaken to meet unique goals and objectives, typically to bring about beneficial change or benefit. The primary challenge of project management is to achieve all the project goals within the given constraints. This information is usually described in a user or project manual, which is created at the beginning of the development process. The primary constraints are scope, time, quality and budget. The secondary and more ambitious challenge is to optimize the allocation of necessary inputs and integrate them to meet pre-defined objectives. In project management, a schedule consists of a list of a project's terminal elements with intended start and finish dates.

2.1 PROJECT PLANNING

Project planning is part of project management, which relates to the use of schedules to plan and subsequently report progress within the project environment. Initially, the project scope is defined and the appropriate methods for completing the project are determined. Following this step, the durations for the various tasks necessary to complete the work are listed and grouped into a work breakdown structure.

At this stage, the project plan may be optimized to achieve the appropriate balance between resource usage and project duration to comply with the project objectives. Once established and agreed, the plan becomes what is known as the baseline. Progress will be measured against the baseline throughout the life of the project. Analyzing progress compared to the baseline is known as earned value management.

2.1.1 Project development approach

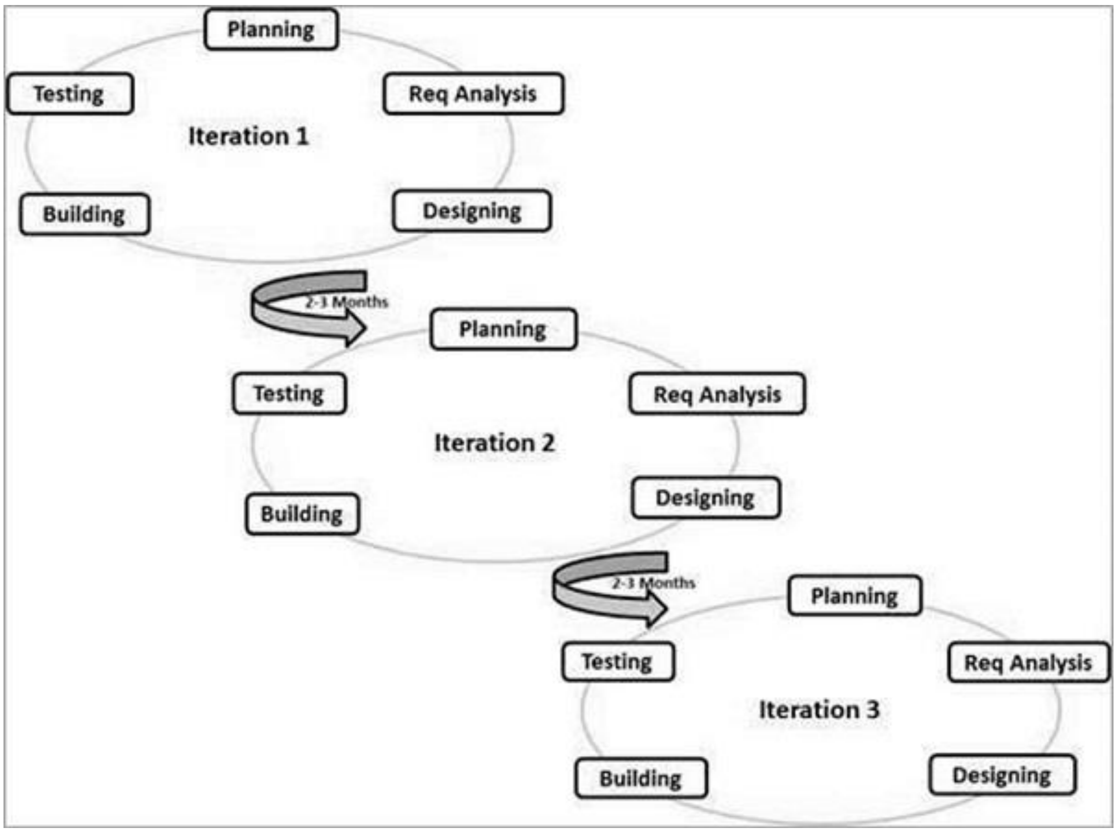
Project development approach or software development models are the various processes or methodologies that are being selected for the development of the project depending on the project's aims and goals. Many development life cycle models have been developed in order to achieve different required objectives. The models specify the various stages of the process and the order in which they are carried out. The selection of model has very high impact on the testing. It will define the what, where and when of our planned testing, influence regression testing and largely determines which test techniques to use.

Agile model

Our project follows the Agile model for software development. Agile SDLC model is a combination of iterative and incremental process models with focus on process adaptability and customer satisfaction by rapid delivery of working software product. Agile Methods break the product into small incremental builds. These builds are provided in iterations. Each iteration typically lasts from about one to three weeks. Every iteration involves cross functional teams working simultaneously on various areas like –

- Planning
- Requirements Analysis
- Design
- Coding
- Unit Testing and
- Acceptance Testing

Agile model believes that every project needs to be handled differently and the existing methods need to be tailored to best suit the project requirements. In Agile, the tasks are divided to time boxes (small time frames) to deliver specific features for a release. Iterative approach is taken and working software build is delivered after each iteration. Each build is incremental in terms of features; the final build holds all the features required by the customer.



[Fig. 2.1.1 Agile SDLC Model]

Advantages of Agile model

- It's a very realistic approach to software development.
- Promotes teamwork and cross training.
- Functionality can be developed rapidly and demonstrated.
- Resource requirements are minimum.
- Suitable for fixed or changing requirements
- Delivers early partial working solutions.
- Good model for environments that change steadily.
- Minimal rules, documentation easily employed.
- Enables concurrent development and delivery within an overall planned context.
- Easy to manage.
- Gives flexibility to developers.

Disadvantages of Agile model

- Not suitable for handling complex dependencies.
- More risk of sustainability, maintainability and extensibility.
- An overall plan, an agile leader and agile PM practice is a must without which it will not work.
- Strict delivery management dictates the scope, functionality to be delivered, and adjustments to meet the deadlines.
- Depends heavily on customer interaction, so if customer is not clear, team can be driven in the wrong direction.
- There is a very high individual dependency, since there is minimum documentation generated.
- Transfer of technology to new team members may be quite challenging due to lack of documentation.

2.1.2 Justification

Agile Model allows dynamic changes to the project at any point in the development cycle. Not much documentation is required, and new features can be added when necessary. No predefined path must be followed, and the method of implementation could be changed at any time. Therefore, we chose Agile Model as an approach to develop our project.

2.2 PROJECT SCHEDULING

Scheduling in project management is the listing of activities, deliverables, and milestones within a project. A schedule also usually includes the planned start and finish date, duration, and resources assigned to each activity. Effective project scheduling is a critical component of successful time management.

For scheduling a project, it is necessary to -

- Break down the project tasks into smaller, manageable form
- Find out various tasks and correlate them
- Estimate time frame required for each task
- Divide time into work-units
- Assign adequate number of work-units for each task
- Calculate total time required for the project from start to finish

CHAPTER 3: SYSTEM REQUIREMENT STUDY

To be used efficiently, all software needs certain hardware components or other software resources to be present on a device on which the software runs. These prerequisites are known as system requirements and are often used as a guideline as opposed to an absolute rule. Most software defines two sets of system requirements: minimum and recommended. With increasing demand for higher processing power and resources in newer versions of software, system requirements tend to increase over time. Industry analysts suggest that this trend plays a bigger part in driving upgrades to existing computer systems than technological advancements.

3.1 USER CHARACTERISTICS

Different Users can access and utilize the application. No predefined login or registration details is needed to use the system. A User can freely access all functionalities of the application without any restrictions. Users are able to process as many videos as they want to without any usage limitations.

3.2 HARDWARE AND SOFTWARE REQUIREMENTS

- **Software Requirements**

Software	**Recommended System Requirements**
Front End	Python Tkinter
Back End	Python 3
OS	Windows 10 and Other

[3.2.1 Software Requirements]

- **Hardware Requirements**

Hardware	**Recommended System Requirements**
Processor	Intel Core i5 CPU & above, 2.50 GHz
Memory	8 GB
Hard Disk	50 GB
GPU	GTX or RTX series of NVidia to perform efficiently

[3.2.2 Hardware Requirements]

CHAPTER 4: SYSTEM REQUIREMENT STUDY

4.1 STUDY OF CURRENT SYSTEM

In this fast pacing world, every sector is increasing in a fast and smooth way. But concern related to monitoring and security is always a major concern of all this infrastructure. Nowadays for security and monitoring of people we majorly rely on Closed-Circuit Television (CCTV) for monitoring. But the only problem is that it requires the man labour to investigate the activities which are in question. This is a very time-consuming process as one has to watch the whole story to generate a track and its very time consuming. And technologies like face recognition is now very efficient in monitoring and tracking the daily activities.

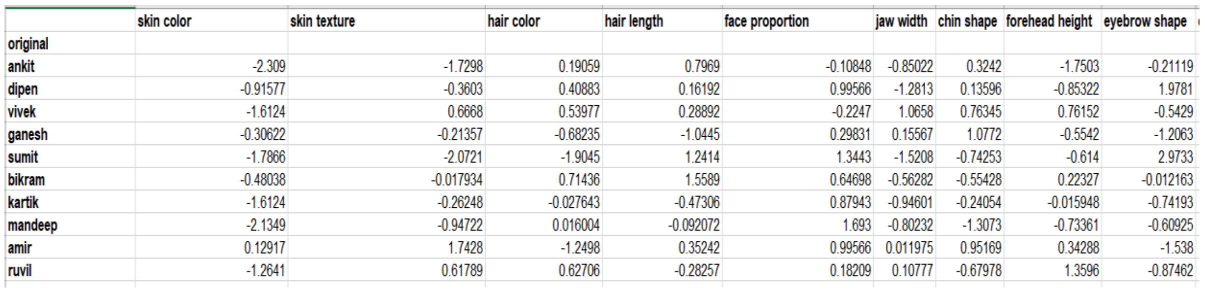
4.2 FEATURES OF NEW SYSTEM

Our systems primary moto is the complete automation of this activity tracking on the basis of recognition of faces by several models which can be easily accessed to keep a track of people in the premises:

- Monitor using a camera and physical machine without any man labour or effort only you have to start the monitoring process.

- It uses hog model to recognize which uses less computational power to give more results.

- Its trainer is an efficient one which generate encodings on the basis many criteria like eyes noses faces etc.



	skin color	skin texture	hair color	hair length	face proportion	jaw width	chin shape	forehead height	eyebrow shape
original									
ankit	-2.309	-1.7298	0.19059	0.7969	-0.10848	-0.85022	0.3242	-1.7503	-0.21119
dipen	-0.91577	-0.3603	0.40883	0.16192	0.99566	-1.2813	0.13596	-0.85322	1.9781
vivek	-1.6124	0.6668	0.53977	0.28892	-0.2247	1.0658	0.76345	0.76152	-0.5429
ganesh	-0.30622	-0.21357	-0.68235	-1.0445	0.29831	0.15567	1.0772	-0.5542	-1.2063
sumit	-1.7866	-2.0721	-1.9045	1.2414	1.3443	-1.5208	-0.74253	-0.614	2.9733
bikram	-0.48038	-0.017934	0.71436	1.5589	0.64698	-0.56282	-0.55428	0.22327	-0.012163
kartik	-1.6124	-0.26248	-0.027643	-0.47306	0.87943	-0.94601	-0.24054	-0.015948	-0.74193
mandeep	-2.1349	-0.94722	0.016004	-0.092072	1.693	-0.80232	-1.3073	-0.73361	-0.60925
amir	0.12917	1.7428	-1.2498	0.35242	0.99566	0.011975	0.95169	0.34288	-1.538
ruvil	-1.2641	0.61789	0.62706	-0.28257	0.18209	0.10777	-0.67978	1.3596	-0.87462

[Fig. 4.2.1 Face key Feature Charts (example)]

- Its GUI is user friendly which is stable enough to perform all tack like recognition encoding generation, recognition, activity tracking.

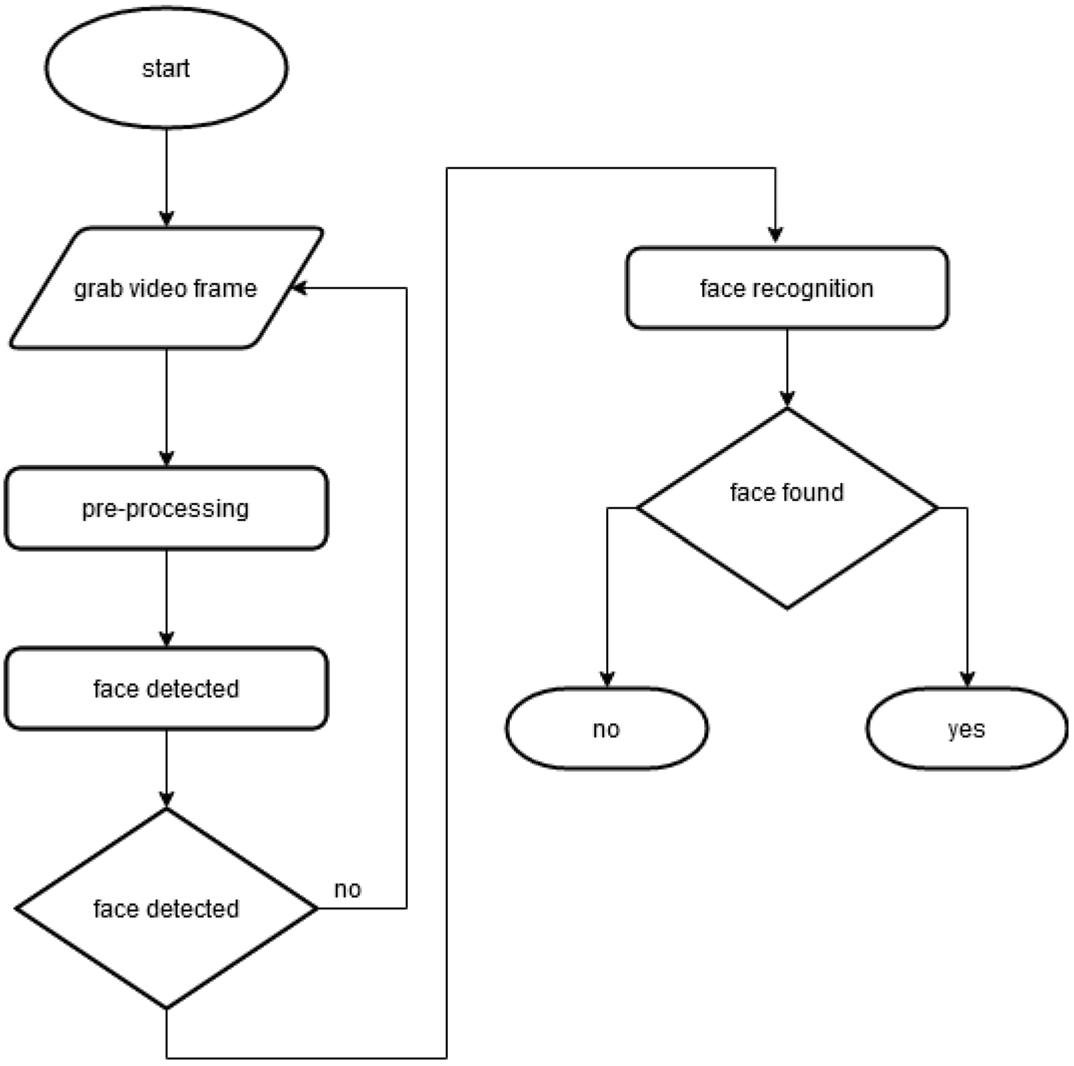
- The size of file which is generated is very less and contains information like date, Time of entrance, time of exit and duration of time spent in premises.

- An inbuilt face register module is also introduced which is able to register the faces via GUI.

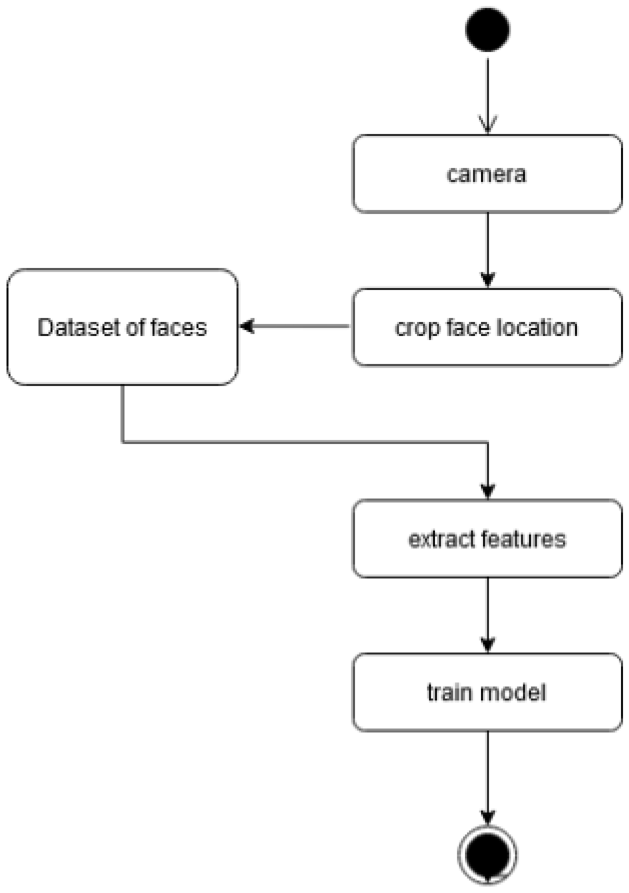
- In this all data is saved permanently in single log file which is very small in size.

4.3 DIAGRAMS

4.3.1 Flow diagram

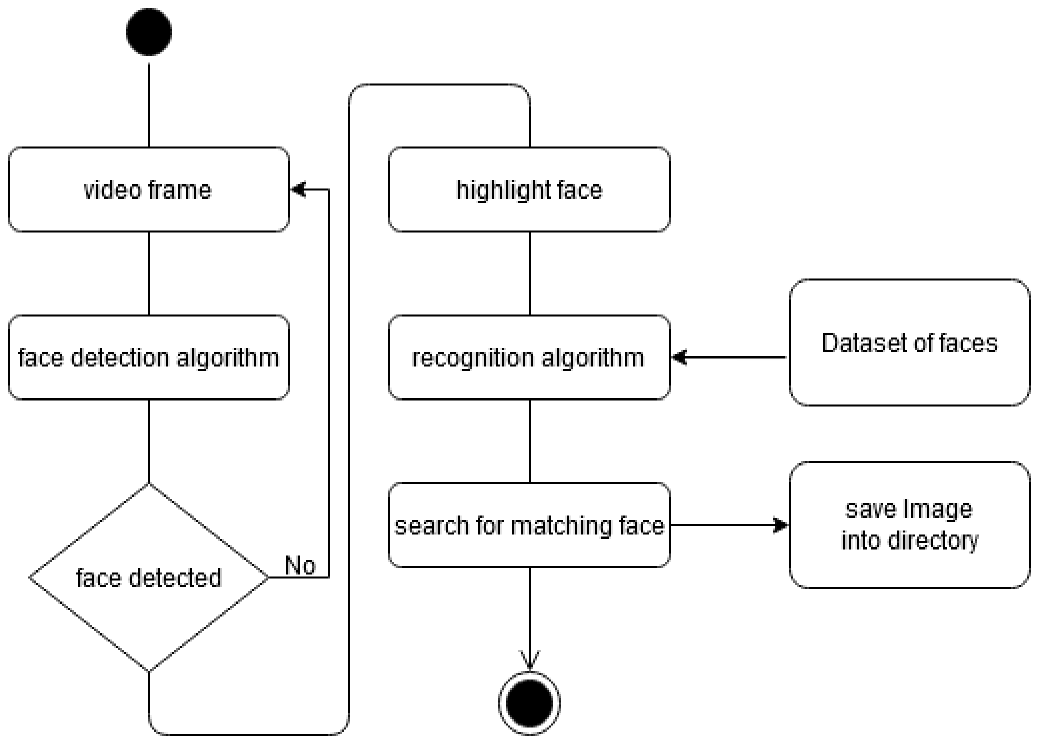


[Fig. 4.3.1(a) Flow Diagram of Recognition]

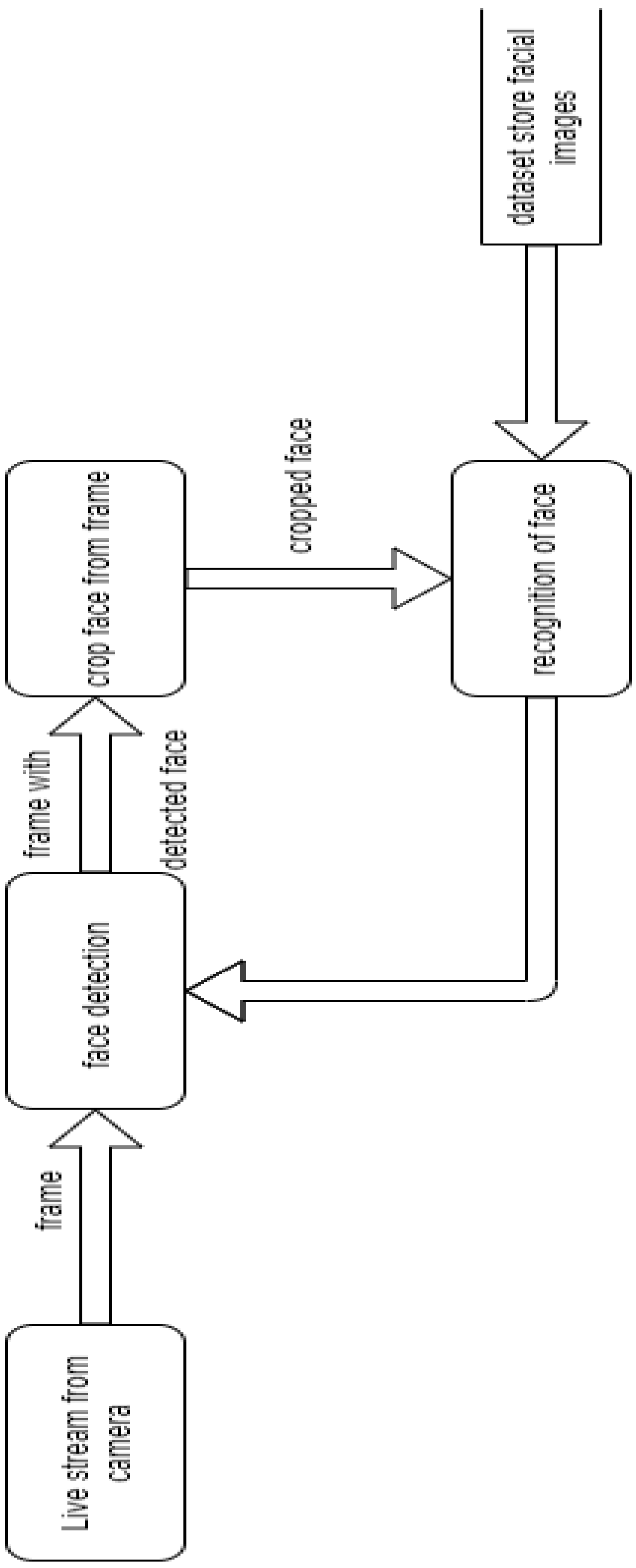


[Fig. 4.3.1(b) Flow Diagram of Recognition]

4.3.2 Activity diagram

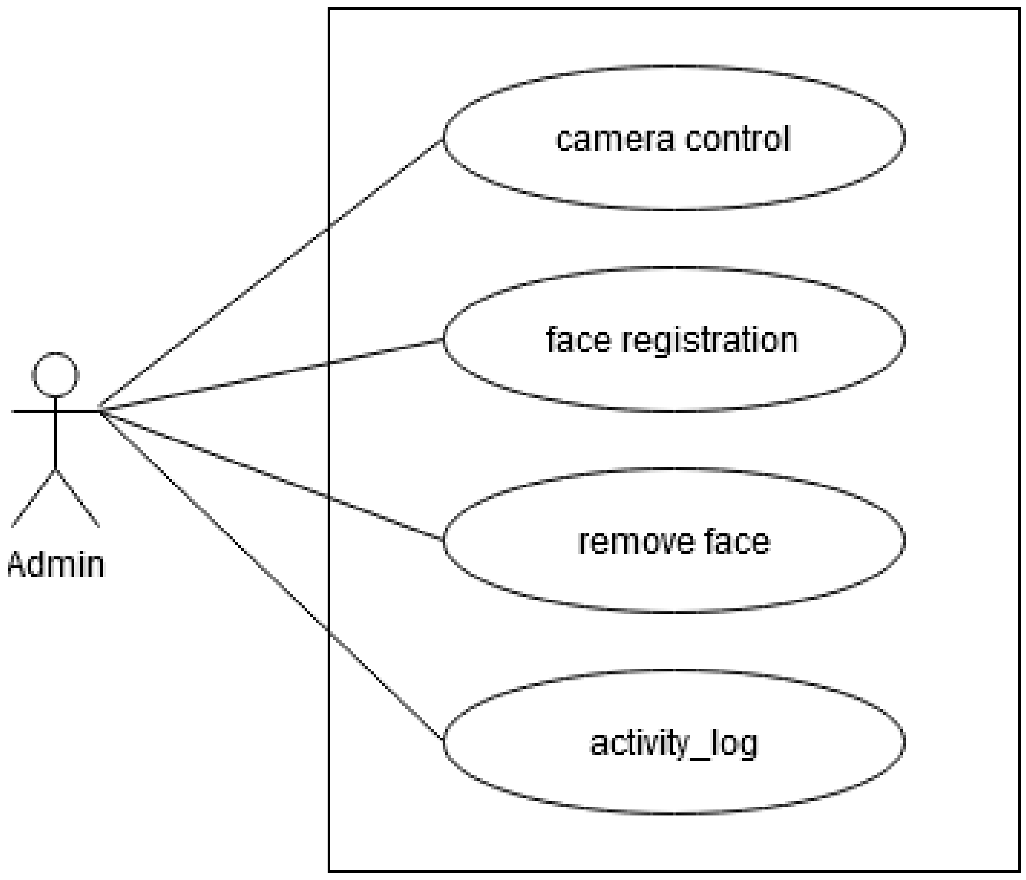


[Fig. 4.3.2(a) Face recognition Activity diagram]



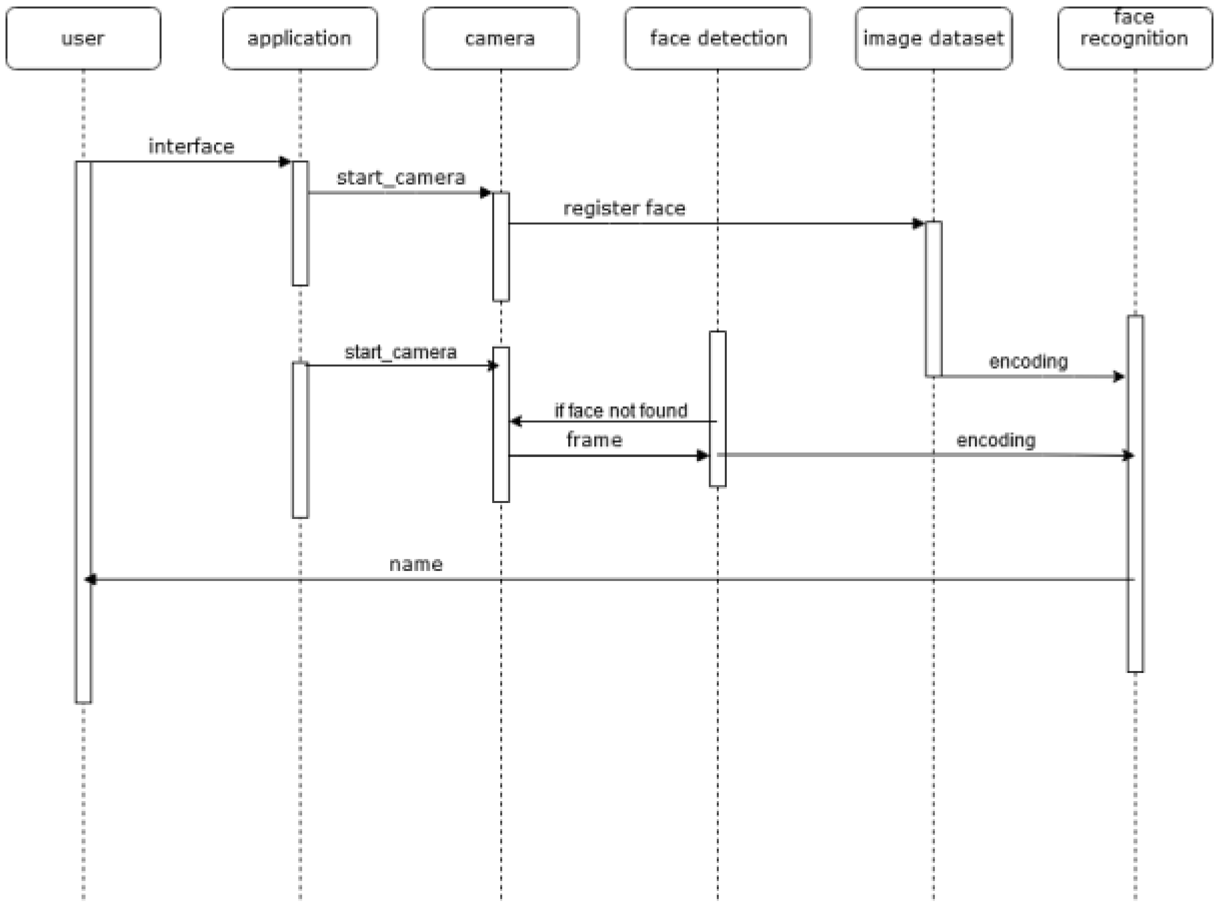
[Fig. 4.3.2(b) Activity diagram

4.3.3 Use case diagram



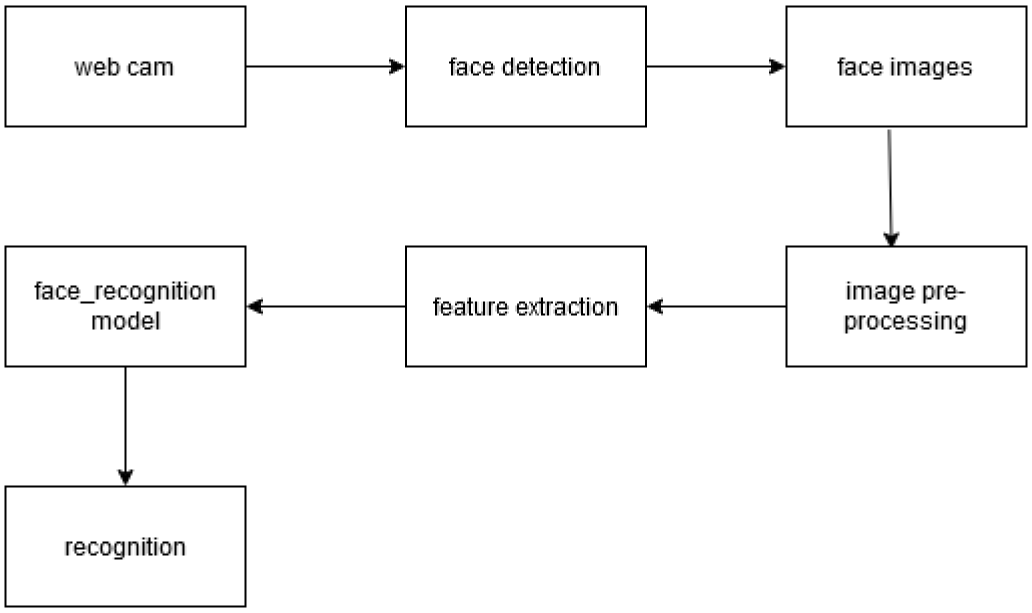
[Fig. 4.3.3 User Use case diagram]

4.3.4 Sequence diagram



[Fig. 4.3.4 Sequence diagram]

4.3.5 Block Diagram



[Fig. 4.3.5 Block diagram]

CHAPTER 5: SYSTEM SOURCE CODE

5.1 FRONT END

GUI.py

```
import shutil
from functools import partial
from tkinter import *
from tkcalendar import *
from PIL import ImageTk, Image
from face_register import *
from Csvgenrator import *
import threading
from recognize_faces_video import *
import os
from imutils import paths
import tkinter as tk
from tkinter import font  as tkfont, ttk
from encode_faces import  *


# View records
BASE_DIR = os.path.dirname(os.path.abspath(__file__))
image_dir = os.path.join(BASE_DIR, "dataset")
logDIR = os.path.join(BASE_DIR, "logJson")


def restart_program():
    """Restarts the current program.
    Note: this function does not return. Any cleanup action (like
    saving data) must be done before calling this function."""
    python = sys.executable
    os.execl(python, python, * sys.argv)


class GUI(tk.Tk):

    def __init__(self, *args, **kwargs):
        tk.Tk.__init__(self, *args, **kwargs)

        self.title_font = tkfont.Font(family='Helvetica', size=18, weight="bold", slant="italic")
        self.title("Automated Face Monitoring")
        self.minsize(600, 400)
        self.geometry("900x700")
        self.tk_setPalette(background="#000000")
        self.tk.call('wm', 'iconphoto', self._w,
        tk.PhotoImage(file='F:\\automatedfacerecognition\\gui_logo.png'))
        self.resizable(False,False)


        container = tk.Frame(self)
        container.pack(side="top", fill="both", expand=True)
        container.grid_rowconfigure(0, weight=1)
        container.grid_columnconfigure(0, weight=1)
        self.frames = {}
        for F in (mainScreen, allLog, reg_face, viewFsces, faces_reg):
            page_name = F.__name__
            frame = F(parent=container, controller=self)
            self.frames[page_name] = frame

            # put all of the pages in the same location;
            # the one on the top of the stacking order
            # will be the one that is visible.
            frame.grid(row=0, column=0, sticky="nsew")

        self.show_frame("mainScreen")

    def show_frame(self, page_name):
        '''Show a frame for the given page name'''
        frame = self.frames[page_name]

        frame.tkraise()



class mainScreen(tk.Frame):

    def __init__(self, parent, controller):
        tk.Frame.__init__(self, parent)
        self.controller = controller


        background_image = tk.PhotoImage(file="F:\\automatedfacerecognition\\mainscreen.png")
        background_label = tk.Label(self, image=background_image)
        background_label.place(x=0, y=0, relwidth=1, relheight=1)
        background_label.image = background_image


        tk.Label(self,text="Automated Face Monitoring", anchor=CENTER, bg="#626a6a",fg="#c7d2e6",
                 font=("Helvetica", 26)).grid(row=1,
                 column=2columnspan=4,sticky="w",padx=40,ipadx=200,pady=20, ipady=40)
        _separator = ttk.Separator(self, orient="horizontal")
        _separator.grid(row=2, columnspan=6, sticky="ew")

        reco=tk.Label(self,width=50,height=1,bg="#c7d2e6",fg="#050505",text=" Starts the video streaming.",
                     font=("Calibri", 12))
        reco.grid(row=3,column=1,columnspan=2)

        starRecognition_Button = Button(self, text="Start
        Recognition",fg="black",bg="#a6776d",width=20,height=2,
                                 font=("Calibri", 13),command=self.start_thread)
        starRecognition_Button.grid(row=3, column=2,columnspan=2, padx=400, pady=20)

        reco1 = tk.Label(self, width=52,bg="#c7d2e6",fg="#050505",text="It will look for all th Users
            DataSet.",
                      height=1,font=("Calibri", 12))
        reco1.grid(row=5, column=1, columnspan=2)


        register_button = Button(self, text="Users",
         fg="black",bg="#a6776d",width=20,height=2,font=("Calibri", 13)
                           ,command=lambda: controller.show_frame("reg_face"))
        register_button.grid(row=5, column=2,columnspan=2, pady=20, padx=20)
        reco = tk.Label(self, bg="#c7d2e6",fg="#050505",text="Display the daily track.", width=52,
                    height=1,font=("Calibri", 12))
        reco.grid(row=4, column=1, columnspan=2)


        showResult_button = Button(self, text="Daily Record",fg="black",bg="#a6776d",width=20,
                          height=2,font=("Calibri", 13)
                          , command=lambda: controller.show_frame("allLog"))
        showResult_button.grid(row=4, column=2,columnspan=2, pady=20, padx=20)
        reco = tk.Label(self ,bg="#c7d2e6",fg="#050505",text="To Update all the properties of system.",
        width=52, height=1,font=("Calibri", 12))
        reco.grid(row=6, column=1, columnspan=2)


        Restart_button = Button(self, text="Refresh",fg="black",bg="#a6776d",
         width=20,height=2,font=("Calibri", 13   , command=restart_program)
        Restart_button.grid(row=6, column=2,columnspan=2, pady=20, padx=20)
        _separator = ttk.Separator(self, orient="horizontal")
        _separator.grid(row=7, columnspan=6, sticky="ew")



    def start_thread(self):
        self.no_thread = threading.Thread(target=startRecognition)
        self.no_thread.daemon = True
        self.no_thread.start()
        self.after(1, self.check_thread)

    def check_thread(self):
        if self.no_thread.is_alive():
            self.after(1, self.check_thread)


class viewFsces(tk.Frame):

    def __init__(self, parent, controller):
        tk.Frame.__init__(self, parent)
        self.controller = controller

        background_image = tk.PhotoImage(file="F:\\automatedfacerecognition\\mainscreen.png")
        background_label = tk.Label(self, image=background_image)
        background_label.place(x=0, y=0, relwidth=1, relheight=1)
        background_label.image = background_image
        tk.Label(self, text="Automated Face Monitoring", anchor=CENTER,bg="#626a6a",fg="#c7d2e6",
              font=("Helvetica", 26)).grid(row=1,
                             column=2, columnspan=4, sticky="w", padx=40,
                             ipadx=200, pady=20, ipady=40)
        _separator = ttk.Separator(self, orient="horizontal")
        _separator.grid(row=2, columnspan=6, sticky="ew")


        reco = tk.Label(self, text="List of all Users.",bg="#c7d2e6",fg="#050505",width=52,
                    height=1,font=("Calibri", 12))
        reco.grid(row=3, column=1, columnspan=2)

        allFace_list = Listbox(self,font=("Calibri", 12),bg="#c7d2e6",fg="#050505")
        allFace_list.grid(row=3, column=2,columnspan=2, padx=400, pady=20)
        imagePaths = os.listdir(image_dir)

        for file in imagePaths:
            allFace_list.insert(END, file)


        def remove_face():
            curent_Selection = allFace_list.get(allFace_list.curselection())


            allFace_list.delete(ANCHOR)

            removedir = os.path.join(image_dir, str(curent_Selection))

            shutil.rmtree(removedir)


        reco = tk.Label(self,text="Selected user are removed.",bg="#c7d2e6",fg="#050505", width=52,
                    height=1,font=("Calibri", 12))
        reco.grid(row=4, column=1, columnspan=2)

        removeFace_Button = Button(self, text="Remove User",fg="black",bg="#a6776d", width=20,
                           height=2,font=("Calibri", 11),
                           ommand=remove_face)
        removeFace_Button.grid(row=4, column=2,columnspan=2, pady=20, padx=20)


        _separator = ttk.Separator(self, orient="horizontal")
        _separator.grid(row=5, columnspan=6, sticky="ew")
        backButton = Button(self,text="back",fg="black",bg="#a4801c", width=20,height=2,font=("Calibri", 10),
                      command=lambda: controller.show_frame("reg_face"))
        backButton.grid(row=6, column =2,columnspan=2, pady=20, padx=20)


#registration  of faces page

class reg_face(tk.Frame):

    def __init__(self, parent, controller):
        tk.Frame.__init__(self, parent)
        self.controller = controller

        background_image = tk.PhotoImage(file="F:\\automatedfacerecognition\\mainscreen.png")
        background_label = tk.Label(self, image=background_image)
        background_label.place(x=0, y=0, relwidth=1, relheight=1)
        background_label.image = background_image

        tk.Label(self, text="Automated Face Monitoring", anchor=CENTER, bg="#626a6a",fg="#c7d2e6",
             font=("Helvetica", 26)).grid(row=1, column=2, columnspan=4, sticky="w", padx=40, ipadx=200,
        pady=20,ipady=40)

        _separator = ttk.Separator(self, orient="horizontal")
        _separator.grid(row=2, columnspan=6, sticky="ew")


        reco = tk.Label(self,text="Users are added and faces are Registered.",bg="#c7d2e6",fg="#050505",
            width=52,  height=1,font=("Calibri", 12))
        reco.grid(row=3, column=1, columnspan=2)


        captureFace_Button = Button(self, text="Capture faces",fg="black",bg="#a6776d",width=20,height=2,
                           font=("Calibri", 13),command = lambda: controller.show_frame("faces_reg"))
        captureFace_Button.grid(row=3, column=2,padx=400, pady=20,columnspan=2)
        reco = tk.Label(self,text="Users are displayed and modified.", bg="#c7d2e6",fg="#050505",width=52,
                height=1,font=("Calibri", 12))
        reco.grid(row=4, column=1, columnspan=2)


        viewRecord_Button = Button(self,
        text="View",fg="black",bg="#a6776d",width=20,height=2,font=("Calibri", 13),
                         command=lambda: controller.show_frame("viewFsces"))
        viewRecord_Button.grid(row=4, column=2, columnspan=2, pady=20, padx=20)

        reco = tk.Label(self,text="All the features of user's faces are extracted \nand stored here.
         ",bg="#c7d2e6",   fg="#050505", width=52, height=2,font=("Calibri", 12))
        reco.grid(row=5, column=1, columnspan=2)

        encode_Button = Button(self,
        text="Encode",fg="black",bg="#a6776d",width=20,height=2,font=("Calibri", 13),
                       command=self.start_thread)
        encode_Button.grid(row=5, column=2,columnspan=2, pady=20, padx=20)


        self.processVar = tk.StringVar(value="")
        self.processsLabel = tk.Label(self, textvar=self.processVar,fg="black",bg="white")
        self.processsLabel.grid(row=6,column=2,columnspan=2, pady=20, padx=20)

        _separator = ttk.Separator(self, orient="horizontal")
        _separator.grid(row=7, columnspan=6, sticky="ew")

        backButton = Button(self, text="back",fg="black",bg="#a4801c",width=20,height=2,font=("Calibri", 10),
                     command=lambda: controller.show_frame("mainScreen"))
        backButton.grid(row=8,column=2,columnspan=2,  pady=20, padx=20)

    def start_thread(self):
        self.no_thread = threading.Thread(target=encodings, args=[self.processVar])
        self.no_thread.daemon = True
        self.no_thread.start()
        self.after(1, self.check_thread)

    def check_thread(self):
        if self.no_thread.is_alive():
            self.after(1, self.check_thread)


#capture faces

class faces_reg(tk.Frame):

    def __init__(self, parent, controller):
        tk.Frame.__init__(self, parent)
        self.controller = controller

        background_image = tk.PhotoImage(file="F:\\automatedfacerecognition\\mainscreen.png")
        background_label = tk.Label(self, image=background_image)
        background_label.place(x=0, y=0, relwidth=1, relheight=1)
        background_label.image = background_image

        tk.Label(self, text="Automated Face Monitoring", anchor=CENTER, bg="#626a6a",fg="#c7d2e6",
             font=("Helvetica", 26)).grid(row=1, column=2, columnspan=4, sticky="w", padx=40, ipadx=200,
         pady=20,ipady=40)
        _separator = ttk.Separator(self, orient="horizontal")
        _separator.grid(row=2, columnspan=6, sticky="ew")


        reco = tk.Label(self,text="Enter the name of user.", bg="#c7d2e6",fg="#050505",width=52, height=1,
                font=("Calibri", 12))
        reco.grid(row=3, column=1, columnspan=2)




        inputUser=Entry(self,bg="white",fg="black",width="20",font=("Calibri", 15))
        inputUser.grid(row=3, column=2,columnspan=2, pady=20, padx=20)


        reco = tk.Label(self,text="Creation of user.", bg="#c7d2e6",fg="#050505",width=52, height=1,
                font=("Calibri", 12))
        reco.grid(row=4, column=1, columnspan=2)

        creatUser_Button= Button(self, text="Create user",fg="black",bg="#a6776d",width=20,height=2,
                    font=("Calibri", 11),command=lambda: createUser(str(inputUser.get())))
        creatUser_Button.grid(row=4, column=2, columnspan=2, pady=20, padx=400)


        reco = tk.Label(self,text="Registration of faces.", bg="#c7d2e6",fg="#050505",width=52, height=1,
                font=("Calibri", 12))
        reco.grid(row=5, column=1, columnspan=2)

        capture_Button= Button(self, text="Capture Face",fg="black",bg="#a6776d",width=20,height=2,
                    font=("Calibri", 11),command=lambda: capture(str(inputUser.get())))
        capture_Button.grid(row=5, column=2, columnspan=2, pady=20, padx=20)


        _separator = ttk.Separator(self, orient="horizontal")
        _separator.grid(row=6, columnspan=6, sticky="ew")

        backButton = Button(self, text="back", fg="black",bg="#a4801c",width=20,height=2,
                   font=("Calibri", 10),command=lambda: controller.show_frame("reg_face"))
        backButton.grid(row=7, column=2, columnspan=2, pady=20, padx=20)


# all log files

class allLog(tk.Frame):

    def __init__(self, parent, controller):
        tk.Frame.__init__(self, parent)
        self.controller = controller


        background_image = tk.PhotoImage(file="F:\\automatedfacerecognition\\mainscreen.png")
        background_label = tk.Label(self, image=background_image)
        background_label.place(x=0, y=0, relwidth=1, relheight=1)
        background_label.image = background_image
        tk.Label(self, text="Automated Face Monitoring", anchor=CENTER,bg="#626a6a",fg="#c7d2e6",
             font=("Helvetica", 26)).grid(row=1, column=2, columnspan=4, sticky="w", padx=40, ipadx=200,
             pady=20,  ipady=40)

        _separator = ttk.Separator(self, orient="horizontal")
        _separator.grid(row=2, columnspan=6, sticky="ew")
        selectLabel = Label(self, text="Select Date:-", bg="#c7d2e6",fg="#050505",width=20,height=2,
                    font=("Calibri", 12)).grid(row=3, column=2,columnspan=2 )
        entrydate = DateEntry(self,font=("Calibri", 12), width=15, background="blue", foregronud="red",
        borderwidth=3,date_pattern='dd-mm-yy')
        entrydate.grid(row=3, column=3)
        global dateValue
        dateValue = entrydate.get()


        nameLabel = Label(self, text="Enter name:-", bg="#c7d2e6",fg="#050505",
                   width=20,height=2,font=("Calibri", 12)).grid(row=4, column=2,columnspan=2 )
        inputName = Entry(self,font=("Calibri", 12), bg="#c7d2e6",fg="#050505")
        inputName.grid(row=4, column=3)

        global nameValue
        nameValue = inputName.get()


        getCsv = Button(self, text="show Record",fg="black",bg="#a6776d",width=20,height=2,
                  font=("Calibri", 11), command=lambda: parser_pr(str(inputName.get()), str(entrydate.get())))
        getCsv.grid(row=5, column=2 ,columnspan=2, pady=20, padx=400)

        tk.Label(self,text="All Log Files", bg="#c7d2e6",fg="#050505",font=("Calibri",
          12)).grid(row=6,column=2)
        tk.Label(self, text="All Users", bg="#c7d2e6",fg="#050505",font=("Calibri", 12)).grid(row=6,
         column=3)
        def viewalllog():
            alllog_list = Listbox(self,font=("Calibri", 12), bg="#c7d2e6",fg="#050505",)
            alllog_list.grid(row=7, column=2)
            logPaths = os.listdir(logDIR)


            for file in logPaths:
                if file.endswith("json"):
                    alllog_list.insert(0, file)

        allFace_list = Listbox(self,font=("Calibri", 12), bg="#c7d2e6",fg="#050505",)
        allFace_list.grid(row=7, column=3)
        imagePaths = os.listdir(image_dir)


        for file in imagePaths:
            allFace_list.insert(0, file)
        viewalllog()

        _separator = ttk.Separator(self, orient="horizontal")
        _separator.grid(row=8, columnspan=6, sticky="ew")

        backButton = Button(self, text="back", fg="black",bg="#a4801c",width=20,height=2,
                    font=("Calibri", 10),command=lambda: controller.show_frame("mainScreen"))
        backButton.grid(row=9  , column=2,columnspan=2, pady=20, padx=400)



if __name__ == "__main__":
    app = GUI()
   app.mainloop()
```

5.2BACK END

recognize_faces_video.py

```
import sys
import face_recognition
from imutils.video import VideoStream, FPS
import argparse
import imutils
import pickle
import time
import cv2
import os
import json
import datetime


def startRecognition():
  ap = argparse.ArgumentParser()

  ap.add_argument("-o", "--output", type=str,
          help="path to output video")
  ap.add_argument("-y", "--display", type=int, default=1,
          help="whether or not to display output frame to screen")
  ap.add_argument("-d", "--detection-method", type=str, default="hog",
          help="face detection model to use: either `hog` or `hog`")
  args = vars(ap.parse_args())

  store = '['
  # folder creation
  if not os.path.exists('known'):
    print("New directory created")
    os.makedirs('known')
  if not os.path.exists('unknown'):
    os.makedirs('unknown')
  if not os.path.exists('logJson'):
    os.makedirs('logJson')

  # load the known faces and embeddings
  print("[INFO] loading encodings...")
  data = pickle.loads(open("encodings.pickle", "rb").read())

  BASE_DIR = os.path.dirname(os.path.abspath(__file__))
  log_dir = os.path.join(BASE_DIR, "logJson")

  # initialize the video stream
  print("[INFO] starting video stream...")
  vs = VideoStream(src=0).start()
  writer = None
  time.sleep(2.0)
  fps = FPS().start()
  i = 1
  # loop over frames
  ktemp = []
  utemp = []
  while True:

    frame = vs.read()
```

```
rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
rgb = imutils.resize(frame, width=750)
r = frame.shape[1] / float(rgb.shape[1])

boxes = face_recognition.face_locations(rgb,
                    model=args["detection_method"])
encodings = face_recognition.face_encodings(rgb, boxes)
names = []


for encoding in encodings:
  # attempt to match each face in the input image to our known
  # encodings
  matches = face_recognition.compare_faces(data["encodings"],
                    encoding)
  name = "Unknown"

  if True in matches:

    matchedIdxs = [i for (i, b) in enumerate(matches) if b]
    counts = {}

    for i in matchedIdxs:
      name = data["names"][i]
      counts[name] = counts.get(name, 0) + 1


    name = max(counts, key=counts.get)


  names.append(name)

for ((top, right, bottom, left), name) in zip(boxes, names):
  # rescale the face coordinates
  top = int(top * r)
  right = int(right * r)
  bottom = int(bottom * r)
  left = int(left * r)

  # draw the predicted face name on the image
  cv2.rectangle(frame, (left, top), (right, bottom),
        (0, 255, 0), 2)
  y = top - 15 if top - 15 > 15 else top + 15
  cv2.putText(frame, name, (left, y), cv2.FONT_HERSHEY_SIMPLEX,
        0.75, (0, 255, 0), 2)

  if name != "Unknown":
    if name not in ktemp:
      ktemp.append(name)
      file_path = 'known/' + str(name) + '.jpg'
      cv2.imwrite(file_path, frame)
    store += '{"name" : "' + name + '", "datetime" : "' + str(time.time()) + '"},'
  # fp.write("name:" + name + ':' + time.strftime('%Y-%m-%d %H:%M:%S', time.localtime()) + '\n')
  else:
    name = 'unknown' + str(i)
    if name not in utemp:
      utemp.append(name)
      file_path = 'unknown/' + name + '.jpg'
      cv2.imwrite(file_path, frame)
      i = i + 1
```

```
            store += '{"name" : "' + name + str(i) + '", "datetime" : "' + str(time.time()) + '"},'


        if args["display"] > 0:
            cv2.imshow("Frame", frame)
            key = cv2.waitKey(1) & 0xFF

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

    store += ']'
    fps.stop()

    RAW_FILE = datetime.date.today().strftime("%d-%m-%y")

    if not os.path.exists('logJson/'+RAW_FILE + '.json'):
        # print("New directory created")
        fp = open('logJson/'+RAW_FILE + '.json', 'w+')
        fp.write(store.split(',]')[0] + ']')
        fp.close()
    else:
        with open('logJson/'+RAW_FILE + '.json', 'a') as fp:
            fp.write(store.split(',]')[0] + ']')
        fp.close()

        with open('logJson/'+RAW_FILE + '.json', 'r') as fp:
            line = fp.read()
        # print(line)

        with open('logJson/'+RAW_FILE + '.json', 'w') as f:
            f.write(line.replace('][', ','))

        f.close()
        fp.close()

    print("Elasped time: {:.2f}".format(fps.elapsed()))
    print("Approx. FPS: {:.2f}".format(fps.fps()))
    # do a bit of cleanup
    cv2.destroyAllWindows()
    vs.stop()

    # check to see if the video writer point needs to be released
    if writer is not None:
        writer.release()

    python = sys.executable
    os.execl(python, python, *sys.argv)
```

Csvgenrator.py

```
import json
from datetime import datetime
import time
import csv
import time
import os
import tkinter as tk
```

```
from subprocess import Popen


#parse the details and generate csv file

def parser_pr(UID, Date_user):

    temp = []
    broken_time = []
    total_t = 0

    with open('logJson/'+Date_user + ".json", "r", encoding='utf-8') as f:
        c = json.load(f, encoding='utf-8')
        for key in c:
            if key['name'] == UID:
                temp.append(float(key['datetime']))

        if len(temp) > 1:
            print('..................')
            OUTPUT_FILE = "raiuniverity-" + UID + '-' + Date_user + ".csv"

            with open(OUTPUT_FILE, "w+", newline="") as cv:
                details_writer = csv.writer(cv, delimiter=",", quotechar='"', quoting=csv.QUOTE_MINIMAL)
                details_writer.writerow(["Camera_id", "ID", "Start_time", "End_time", "Total_time"])

                s_time = temp[0]
                for i in range(0, len(temp) - 1):
                    e_time = temp[i]
                    e1_time = temp[i + 1]
                    second = time_calulator(e_time, e1_time)
                    # print(second)
                    if time_partition(second):
                        total_t
                        broken_time.append(s_time)
                        broken_time.append(e_time)

                        details_writer.writerow(['1001', UID, datetime.fromtimestamp(s_time).strftime('%H:%M:%S'),
                                                 datetime.fromtimestamp(e_time).strftime('%H:%M:%S'), total_t])

                        s_time = e1_time
                        total_t = 0
                    else:
                        total_t = second + total_t


            cv.close()
            f.close()
            if (total_t < 16):
                os.remove("raiuniverity-" + UID + '-' + Date_user + ".csv")
                alert()
                return False
            csv_open("raiuniverity-" + UID + '-' + Date_user + ".csv")
            return True
        else:
            f.close()
            alert()
            return False


#check time dilation between each user
```

```
def time_calulator(t1, t2):
    elip = datetime.fromtimestamp(t2) - datetime.fromtimestamp(t1)
    return elip.seconds + (elip.microseconds / 1000000)

#divide time on basis ka time
def time_partition(second):
    if second > 15:
        return True
    else:
        return False

#open csv in excel
def csv_open(rasta):
    Popen(rasta, shell=True)

#alert if user not  found
def alert():
    popup = tk.Tk()
    popup.wm_title("!")
    label = tk.Label(popup, text="User not Found")
    label.pack(side="top", fill="x", pady=10)
    B1 = tk.Button(popup, text="Okay", command = popup.destroy)
    B1.pack()
    popup.mainloop()
```

encode_faces.py

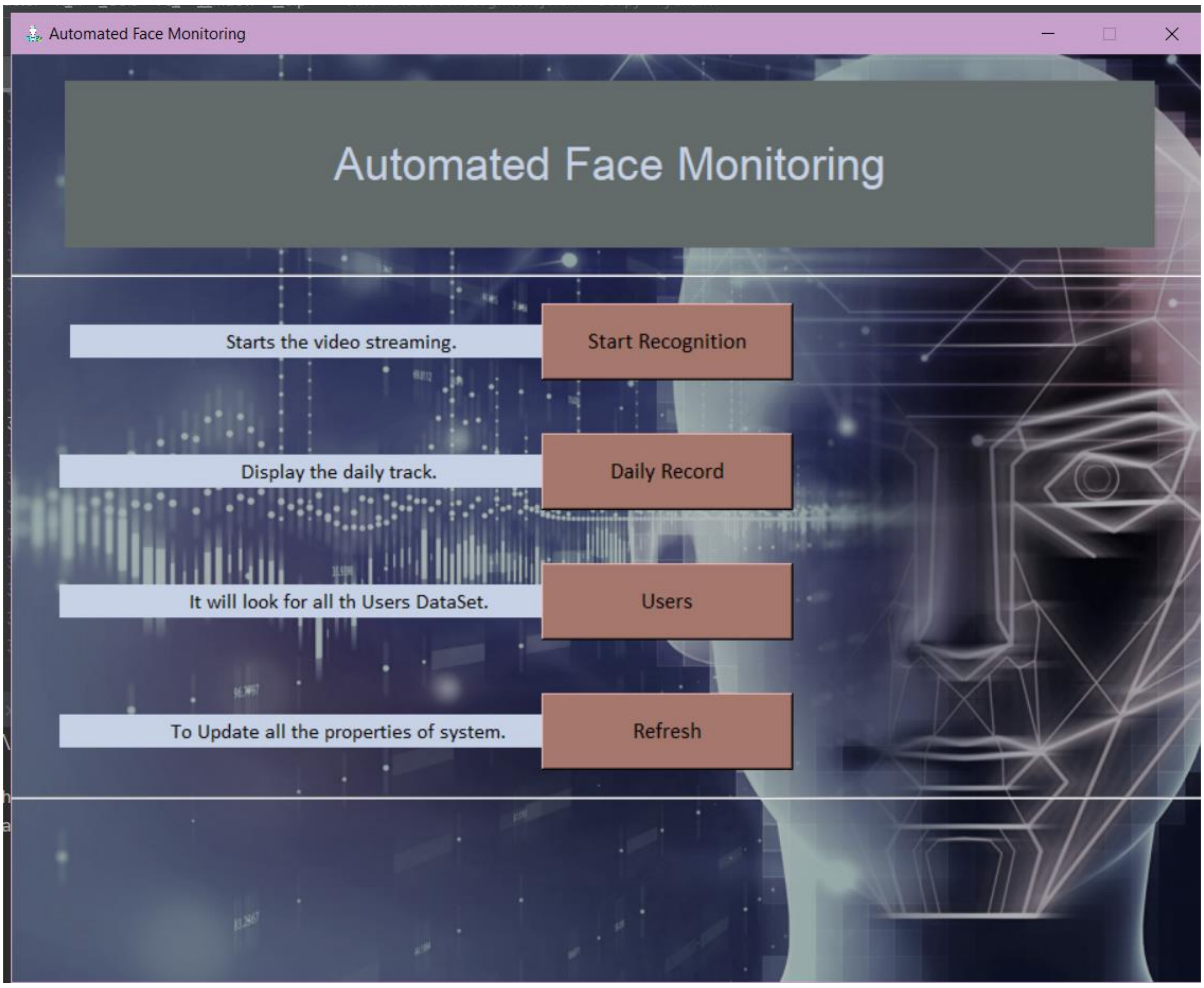
```
from imutils import paths
import face_recognition
import argparse
import pickle
import cv2
import os


def encodings(processvar1):

   process = processvar1
   process.set("Changed")
   ap = argparse.ArgumentParser()
   ap.add_argument("-d", "--detection-method", type=str, default="hog",
            help="face detection model to use: either `hog` or `cnn`")
   args = vars(ap.parse_args())

   # grab the paths to the input images in our dataset
   process.set("[INFO] quantifying faces...")
   imagePaths = list(paths.list_images("dataset"))

   # initialize the list of known encodings and known names
   knownEncodings = []
   knownNames = []


   for (i, imagePath) in enumerate(imagePaths):
     # extract the person name from the image path
     process.set("[INFO] processing image {}/{}".format(i + 1,
                                len(imagePaths)))
     name = imagePath.split(os.path.sep)[-2]
```

```
        image = cv2.imread(imagePath)
        rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

        boxes = face_recognition.face_locations(rgb,
                            model=args["detection_method"])


        encodings = face_recognition.face_encodings(rgb, boxes)

        for encoding in encodings:

            knownEncodings.append(encoding)
            knownNames.append(name)

    process.set("[INFO] serializing encodings...")
    process.set("Co")
    data = {"encodings": knownEncodings, "names": knownNames}
    f = open("encodings.pickle", "wb")
    f.write(pickle.dumps(data))
    f.close()
    process.set("Completed Successfully")
```

face_register.py

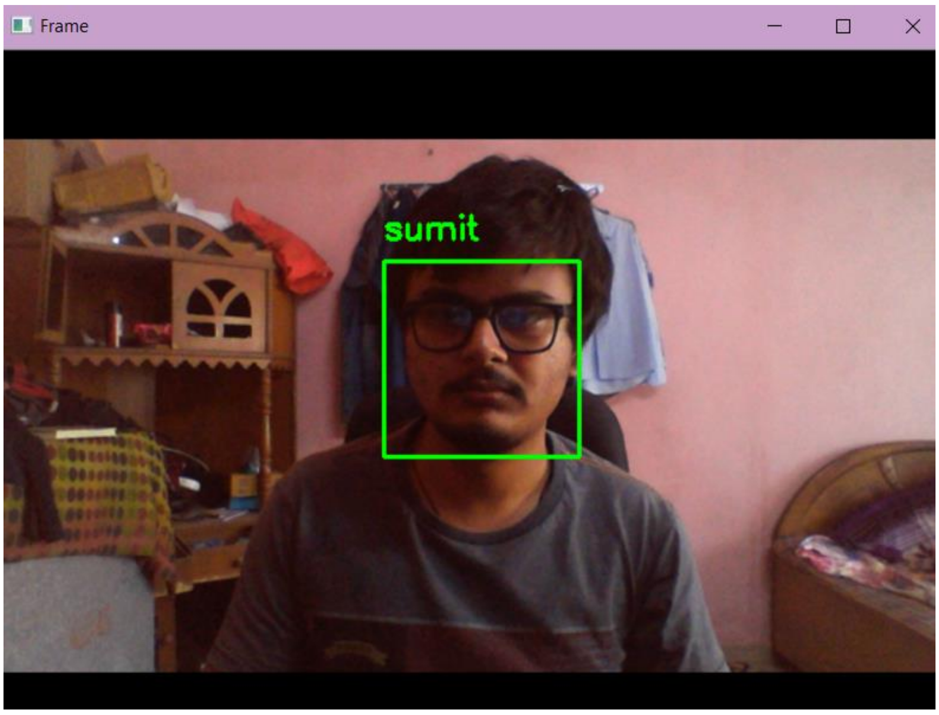
```
import sys
import cv2
import time
import os
import numpy as np
import urllib.request


url = 'http://192.168.0.102:8080/shot.jpg'


def createUser(name):
    BASE_DIR = os.path.dirname(os.path.abspath(__file__))
    image_dir = os.path.join(BASE_DIR, "dataset")

    image_folder = os.path.join(image_dir, name)
    folder_create = os.mkdir(image_folder)

    save_folder = os.path.join(image_folder, name)

   # python = sys.executable
    #os.execl(python, python, *sys.argv)


def capture(name1):
    name = name1
    face_count=0

    while True:


        #ret, frame =cap.read()
        imgResp = urllib.request.urlopen(url)
        imgNp = np.array(bytearray(imgResp.read()), dtype=np.uint8)
        img = cv2.imdecode(imgNp, -1)

        # put the image on screen
        #cv2.imshow('IPWebcam', img)
        gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
        if img is not None:
            face_count+=1
            gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)


            file_path = 'F:/automatedfacerecognition/dataset/'+name+'/user'+str(face_count)+'.jpg'
            cv2.imwrite(file_path,img)

            cv2.putText(img, str(face_count), (50, 50), cv2.FONT_HERSHEY_COMPLEX, 1, (0, 255, 0), 2)
            cv2.imshow('press q to quit Window', img)
            if cv2.waitKey(20)  & 0xFF == ord('q'):
                break

    #cap.release()
    cv2.destroyAllWindows()
```

CHAPTER 6: PROJECT SCREENSHOTS

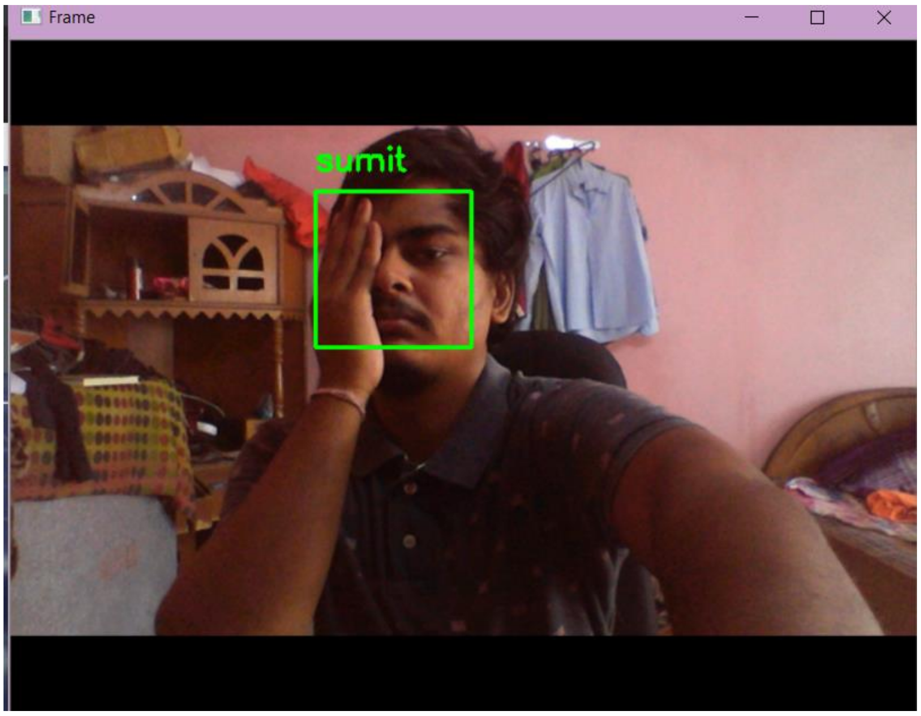


Automated Face Monitoring
Automated Face Monitoring
Starts the video streaming.
Start Recognition
Display the daily track.
Daily Record
It will look for all th Users DataSet.
Users
To Update all the properties of system.
Refresh

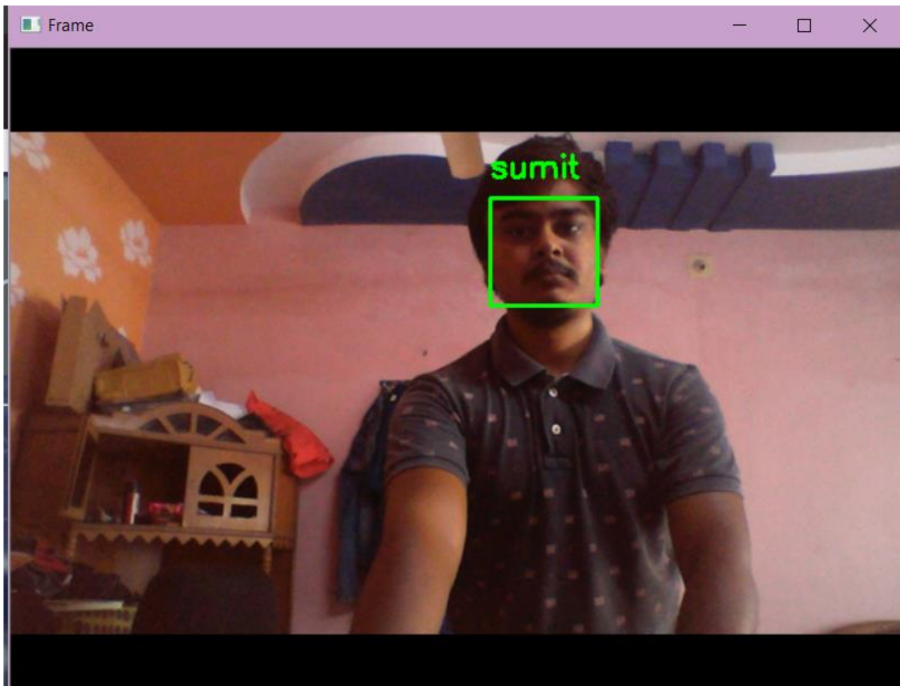
[Fig. 6.1 Main Menu Screen]



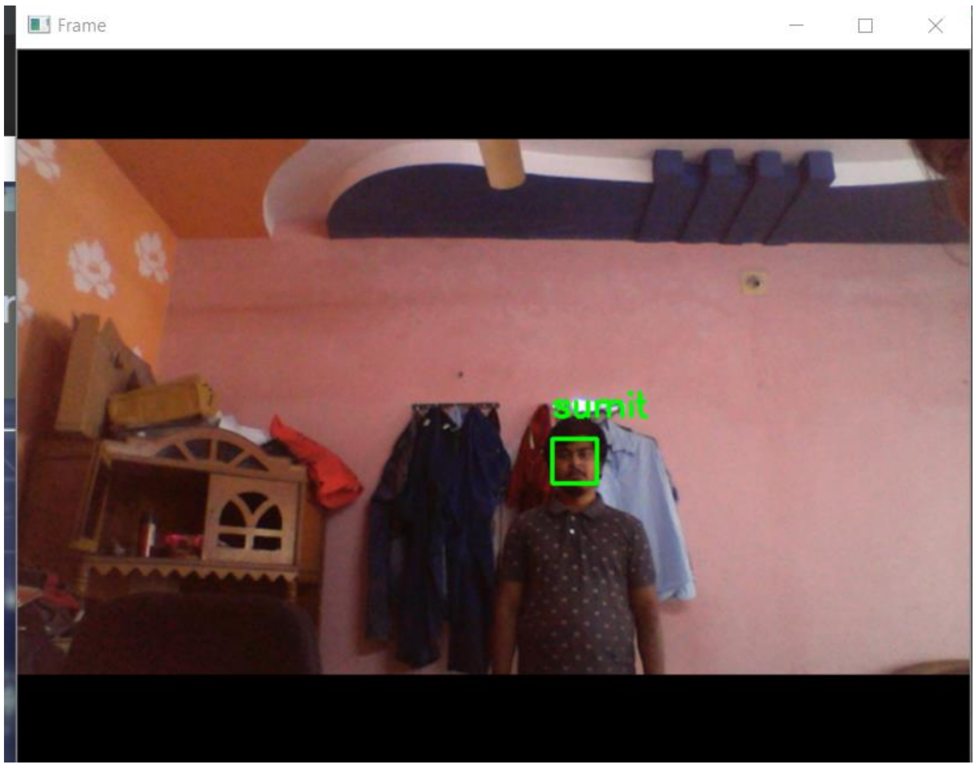
[Fig. 6.2.1 Recognition Screen Normal]



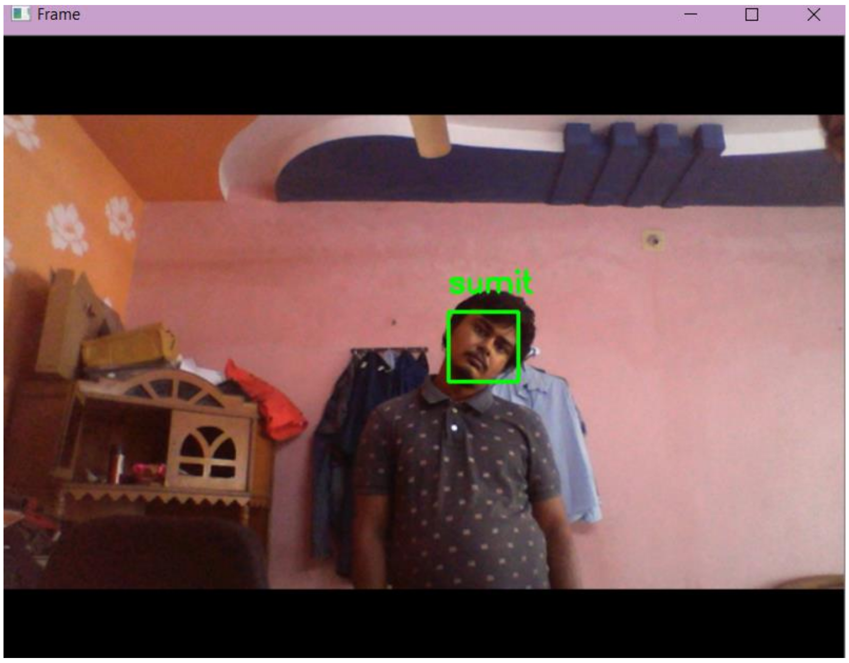
[Fig. 6.2.2 Recognition Screen Obstructed Face]



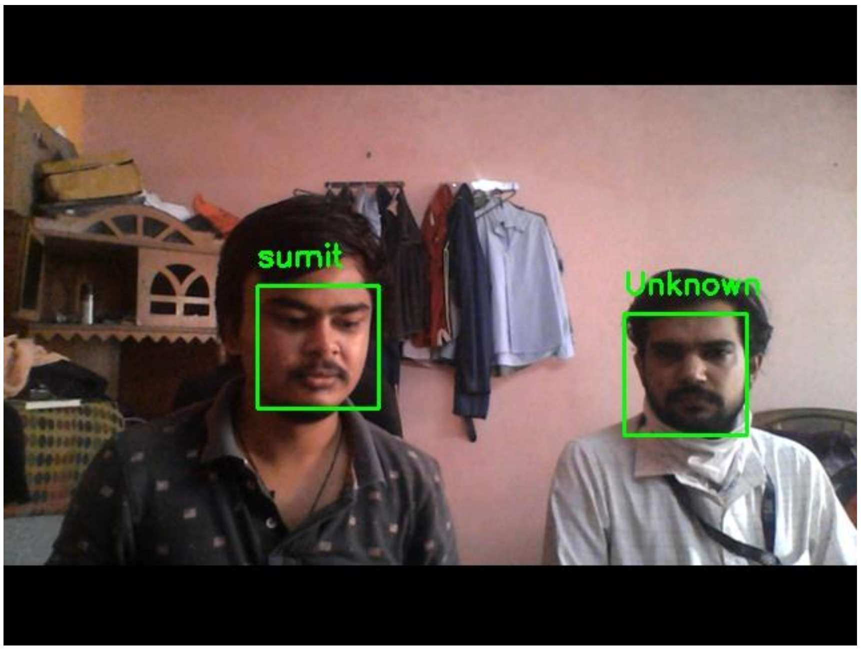
[Fig. 6.2.3 Recognition Screen Standing Position]



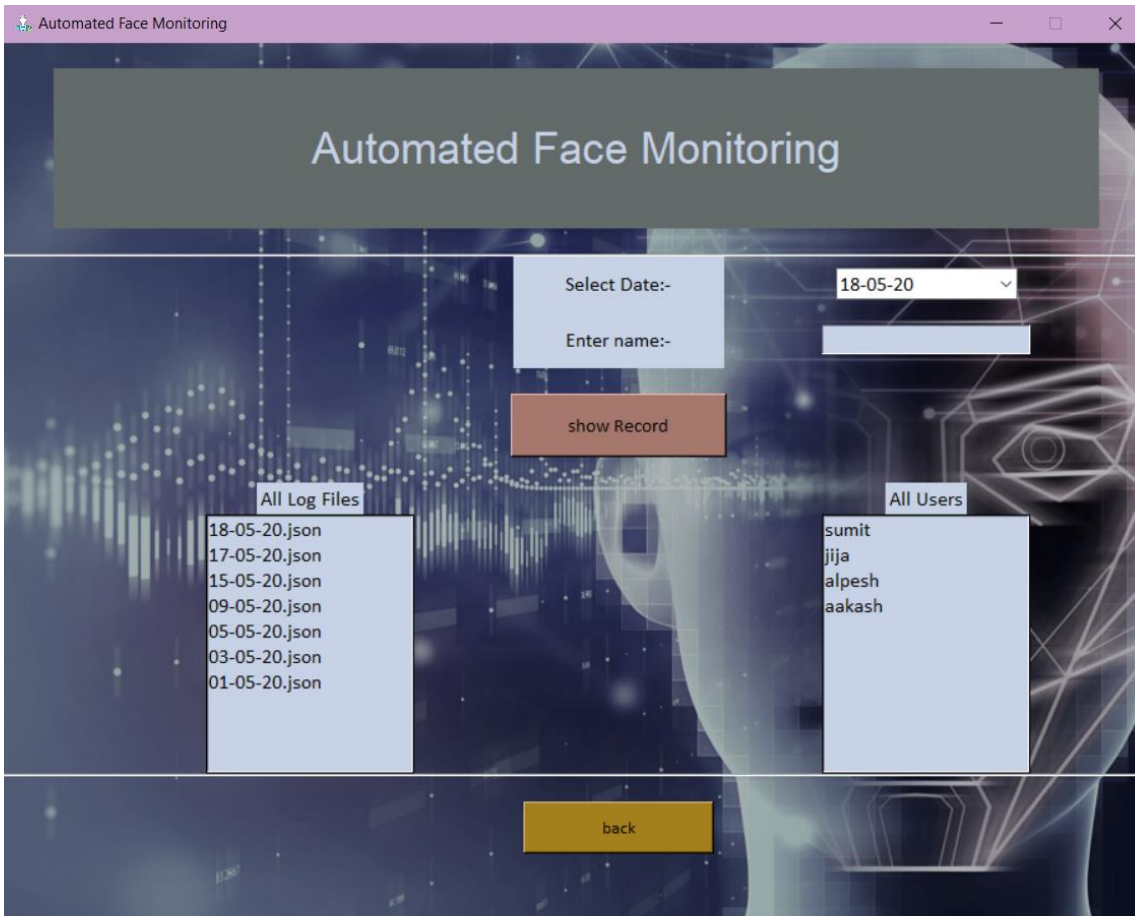
[Fig. 6.2.4 Recognition Screen Distant Position]



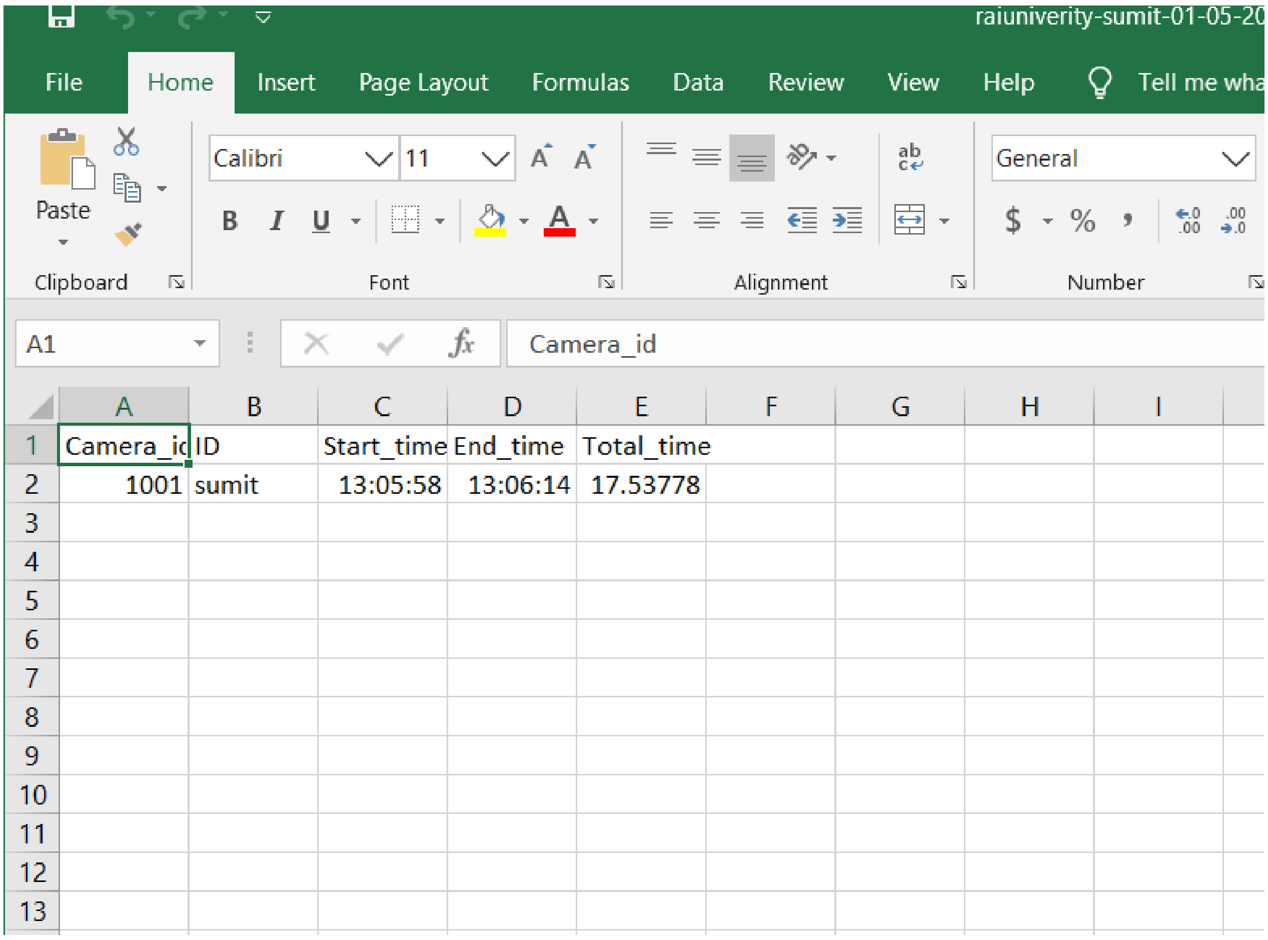
[Fig. 6.2.5 Recognition Screen Angled Position]



[Fig. 6.2.6 Recognition Screen Multiple Faces]

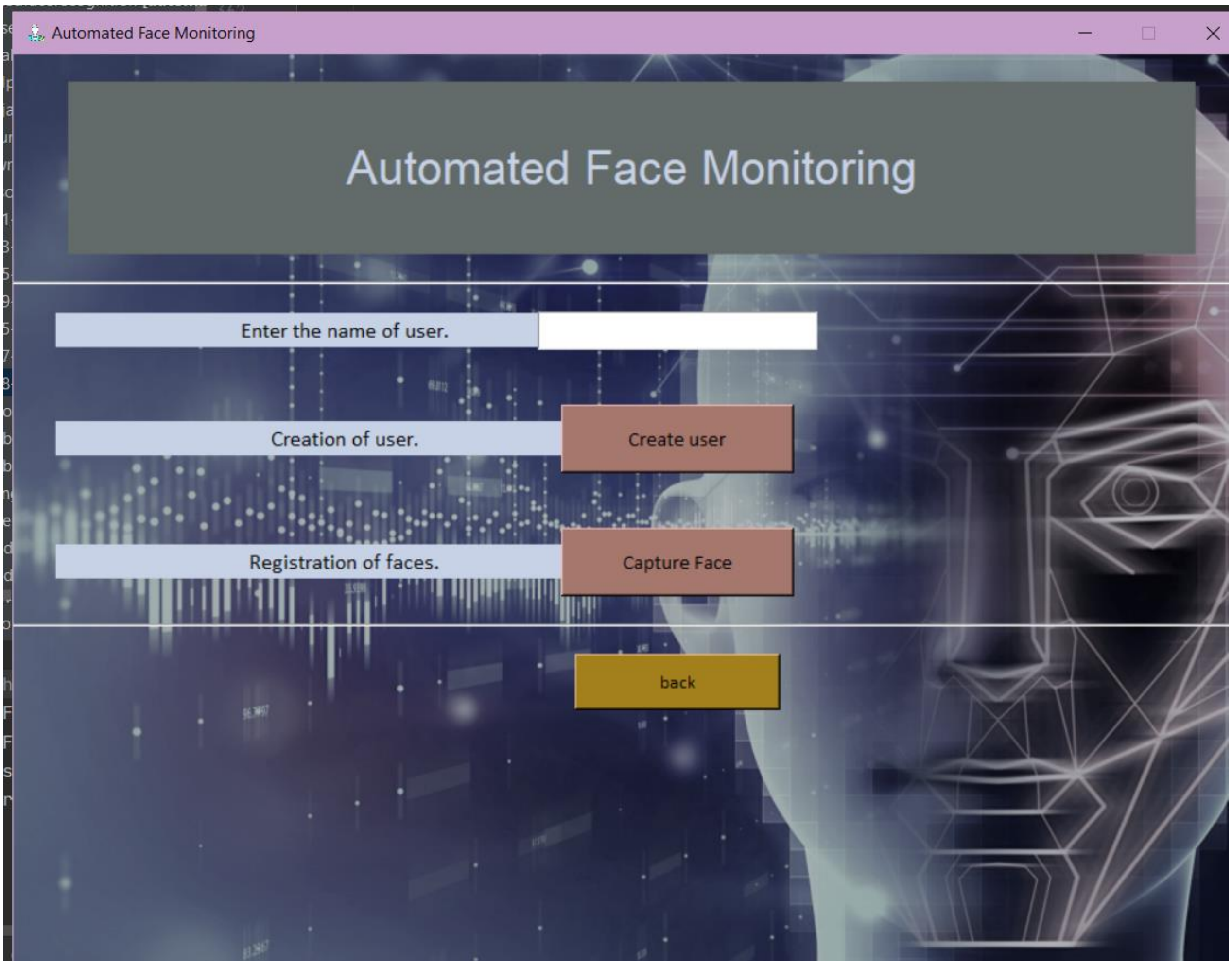


[Fig. 6.3Daily Record Screen]

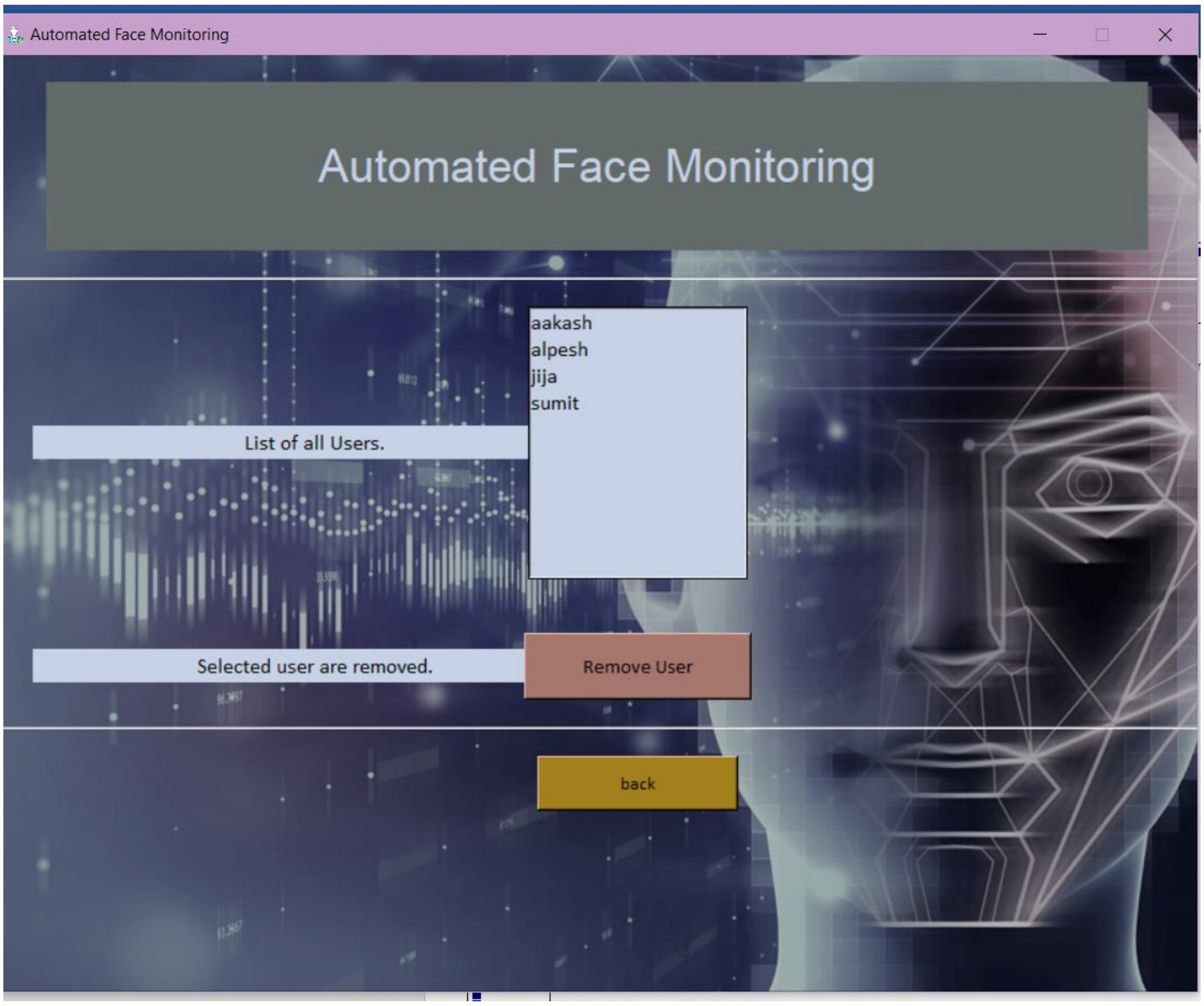


	A	B	C	D	E
1	Camera_id	ID	Start_time	End_time	Total_time
2	1001	sumit	13:05:58	13:06:14	17.53778

[Fig. 6.4 Daily Track]



[Fig. 6.5 Faces Registration Screen]


Automated Face Monitoring
Automated Face Monitoring
List of all Users.
aakash
alpesh
jija
sumit
Selected user are removed.
Remove User
back

[Fig. 6.6 Users Details]

CHAPTER 7: CONCLUSION AND DISCUSSION

The main aim of this project was to put what we learnt in our Computer engineering theory into practice. The automated face recognition system designed by our team allowed us to fully exercise the various techniques used for the development and implementation of computer programs. We were able to learn a new programming language, Python and also understood the working different face recognition models, exercise performed in today’s world. We also learnt the importance of teamwork and how to divide our work in order to efficiently develop our software.

7.1 SUMMARY OF PROJECT WORK

Using our Automated monitoring system, **“Automated Face Recognition based Monitoring System”** we were successfully able to monitor daily activities of people with the help of a camera. After multiple rounds of testing and modifications, our system can finally produce results with accuracies that can be deemed applicable in the practical world. Users can now easily track their premises activities without man effort. With a very small size of log file we can avail a large amount of information in an efficient way.

7.2 FUTURE ENHANCEMENTS

There is always room for improvement, and the software we created can also be further enhanced. This is especially because we had to create it within a limited amount of time.

With more time, the software can be improved to a much extent in the way of recognizing and remove the lankness of the system. We can work on the features that is to be extracted from face for recognition to increase the accuracy. We can also work on some specific activities that can be trained like stand, sit to get some more track from the real time monitoring.

REFERENCES

1. https://en.wikipedia.org/wiki/Python_(programming_language)
2. https://en.wikipedia.org/wiki/PyCharm
3. https://opentextbc.ca/projectmanagement/chapter/chapter-10-project-schedule-planning-project-management/
4. https://www.tutorialspoint.com/sdlc/sdlc_agile_model.htm