Dr. Babasaheb Ambedkar Marathwada University,

Chh.Sambhajinagar

Project Report

On

"Face Recognition Attendance System"

Presented by

Mr. Vedant Davhare
Mr. Shivshankar Barbade
Mr. Santosh Khillari

Guided by

Asst. Prof. Pradeep Ubale

Submitted to

Department of Computer Science

Marathwada Institute of Technology,

Chh.Sambhajinagar

2023-2024

Department of Computer Science

Marathwada Institute of Technology

Marathwada Institute of Technology

This is to certify that **Vedant Davhare, Shivshankar Barbade, Santosh Khillari** of **Bachelor of Computer Science** have successfully presented the project entitled "**Face Recognition Attendance System**" as per the requirement of Dr. Babasaheb Ambedkar Marathwada University, Chh.Sambhajinagar as per the partial fulfilment for the award of bachelor's degree in computer science in the Academic year 2023 - 2024.

Examiner

Guided by	**Head of the Dept**	**Principal**
Asst. Prof. Pradeep Ubale	Asst. Prof. S. W. Quadri	

ACKNOWLEDGEMENT

At every outset we express our gratitude to almighty lord for showering his grace and blessings upon us to complete this project.

Although our name appears on the cover of this project, many people have contributed in some or the form to this project development. We could not have done this project without the assistance or support of each of the following. We thank all of you.

We wish to place on our record our deep sense of gratitude to our project guide **Asst. Prof. Pradeep Ubale** and course co-ordinator and Head of the department **Asst. Prof. S. W. Quadri** and **Vice Principal. Dr. Mahendra Kondekar** for the constant motivation and valuable help through the project. We also extend our thanks to other faculties for their co-operation.

Thanking you,

Vedant Davhare.

Shivshankar Barbade.

Santosh Khillari.

Class-BCS- III Year

INDEX

1. SYNOPSIS:

1.1 Introduction to project :

In today's digital age, automating attendance tracking processes is becoming increasingly important for organizations seeking efficiency and accuracy. Traditional methods such as manual attendance registers are prone to errors and time-consuming.

1.2 Purpose of application :

The purpose of our Face Recognition Attendance System is to streamline attendance tracking by offering a modern, accurate, and efficient solution. By leveraging facial recognition technology, we aim to eliminate errors associated with traditional methods, save time for administrators and attendees, and provide real-time monitoring of attendance data.

1.3 Scope :

The scope of our Face Recognition Attendance System encompasses deployment across various sectors, including education, corporate, and government institutions. It offers customizable features such as attendance policies, reporting formats, and integration capabilities with existing infrastructure.

1.4 Technology Used:

1.4.1 Language: Python

1.4.2 Backend: My SQL

1.5 Minimum System Requirement:

1.5.1 Minimum RAM: - 4GB

1.5.2 Hard Disk: - 40GB

1.5.3 Processor: - Intel/AMD

1.5.4 Operating System: - Windows7 +

1.6 Overview:

The system provides real-time monitoring of attendance data, generates reports, and offers customization options to meet the specific needs of different organizations. With its user-friendly interface and emphasis on accuracy and efficiency, the system aims to revolutionize traditional attendance tracking methods across various sectors.

I. Admin has following rights:

a. Admin can store student details and face records.

b. Admin can train data after adding new records.

c. Admin can update student details.

2.SYSTEM ANALYSIS:

2.1 Introduction:

In today's digital age, automating attendance tracking processes is becoming increasingly important for organizations seeking efficiency and accuracy. Traditional methods such as manual attendance registers are prone to errors and time-consuming. To address these challenges, we introduce our Face Recognition Attendance System, a modern solution leveraging the power of facial recognition technology to streamline the attendance management process. Our Face Recognition Attendance System offers a sophisticated yet user-friendly approach to attendance tracking. By utilizing state-of-the-art facial recognition algorithms, the system accurately identifies individuals and records their attendance without the need for manual input. This eliminates the possibility of errors associated with traditional methods and enhances overall efficiency.

2.2 Analysis Model:

SDLC Methodologies:

- In the Waterfall model, software development progresses through a linear sequence of phases, with each phase building upon the outputs of the previous one. The model is often depicted as a cascading waterfall, symbolizing the flow of activities from one phase to the next.

- In the first phase of the Waterfall model, requirements are gathered and documented in detail. This phase sets the foundation for the entire project by clearly defining what the software should accomplish and how it should function.

- The second phase involves system design, where the overall architecture and structure of the software system are planned. This includes defining the software components, interfaces, and data flow.

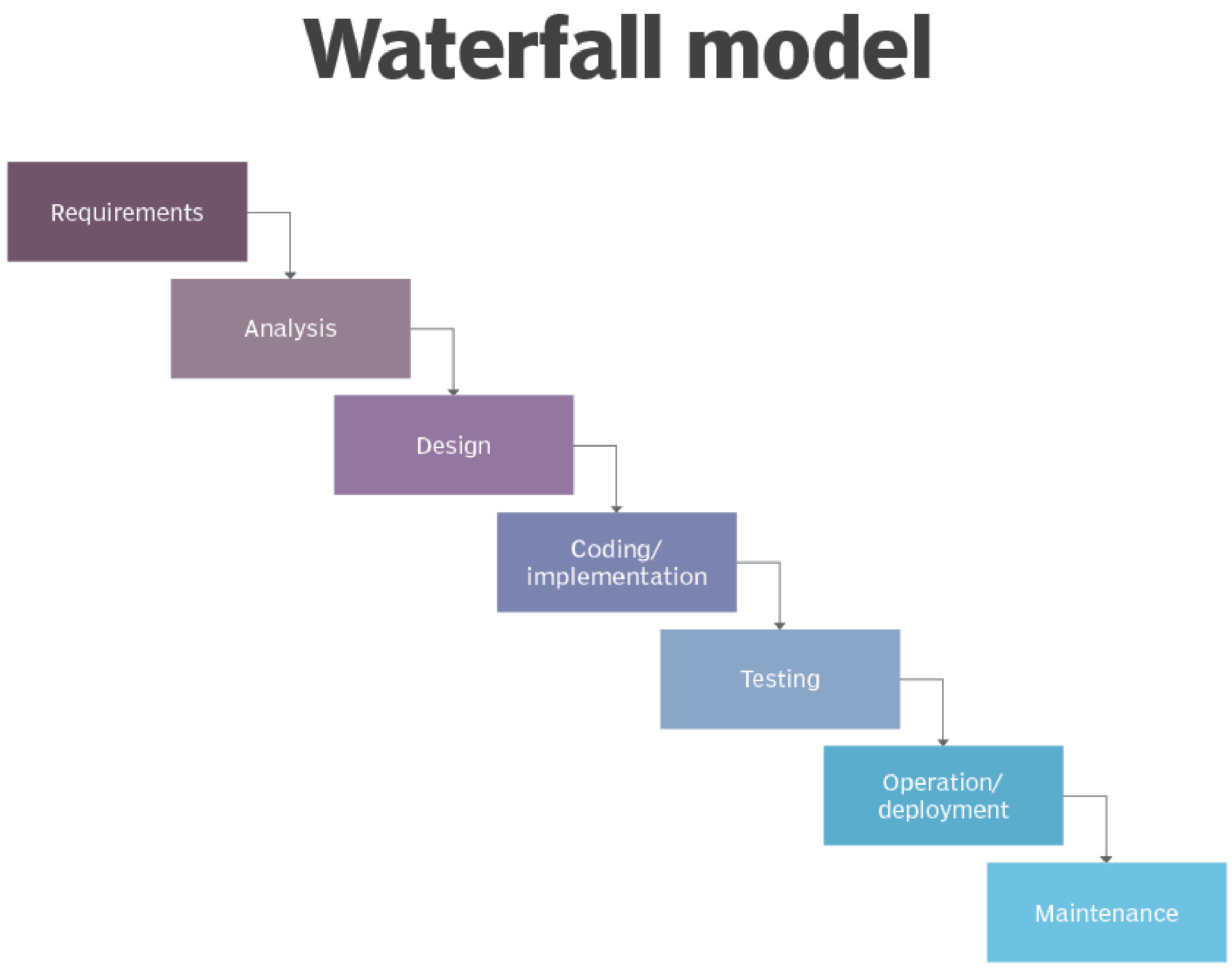
- Once the system design is complete, the development phase begins. During this phase, the actual code for the software is written based on the specifications laid out in the previous phases. This is where the design comes to life as functional software.

- Following development, the software enters the testing phase. Here, the software is rigorously tested to ensure that it meets the specified requirements and functions correctly. Any defects or bugs found during testing are documented and addressed.

- Once testing is complete and the software is deemed stable and ready for release, it moves into the deployment phase. In this phase, the software is deployed to the end-users or customers, and any necessary training or support is provided.

- After deployment, the software enters the maintenance phase, where it is regularly updated and maintained to address any issues that arise and to keep it aligned with changing user needs and technology advancements.

- Unlike the iterative nature of the Spiral model, the Waterfall model follows a sequential approach, with each phase completed before moving on to the next. This model provides a structured framework for software development, making it easier to manage and track progress. However, it can be less flexible than iterative models, as changes to requirements or design are difficult to accommodate once development has begun.



2.3 Study of the System:

The study of our Face Recognition Attendance System involves a comprehensive analysis of its components, functionalities, and performance. It encompasses examining the following aspects:

2.3.1 System Architecture:

Understanding the overall structure and organization of the system, including its software components, databases, and external interfaces.

2.3.2 Facial Recognition Algorithm:

Investigating the underlying facial recognition algorithm used by the system, its accuracy, robustness, and computational efficiency.

2.3.3 Database Management:

Analysing the database schema, data storage mechanisms, and data retrieval processes to ensure optimal performance and scalability.

2.3.4 User Interface:

The user interface design for its intuitiveness, accessibility, and responsiveness, aiming to provide a seamless user experience.

2.3.5 Performance Evaluation:

Conducting tests and benchmarks to assess the system's performance in terms of speed, accuracy, and resource utilization under varying conditions.

2.3.6Security Considerations:

Identifying potential security vulnerabilities and implementing measures to safeguard sensitive data and prevent unauthorized access.

2.4 System Requirement Specification:

Face Recognition Attendance System has two main parts:

- Software
- Hardware

(1) Software Requirement:

- Python
- PyCharm
- Windows 7/10
- MySQL

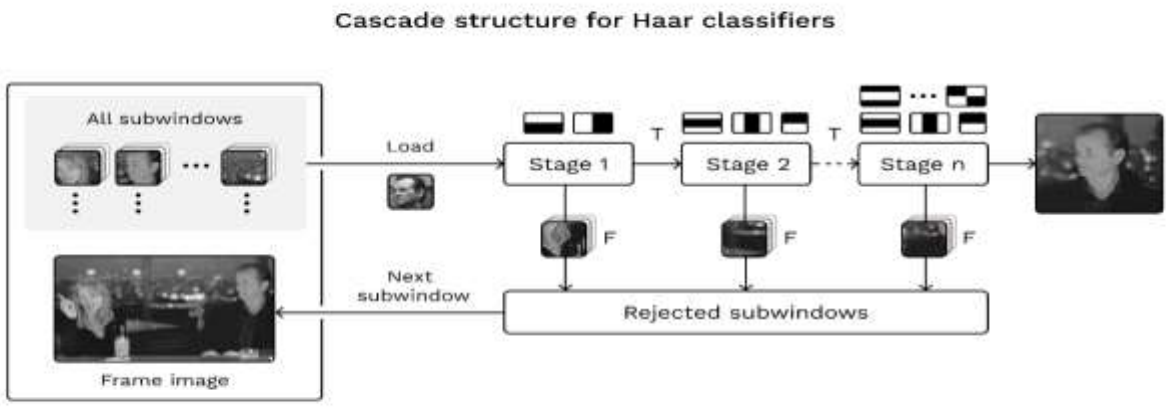
(2) Minimum Hardware Requirements:

All physical requirements i.e. input devices, processor, output devices and inter- connecting processor of the computer is called Hardware.

- Minimum RAM: - 4GB
- Hard Disk: - 40GB
- Camera
- Processor: - Intel/AMD
- Operating System: - Windows 7+

2.5 Algorithm Used:

Haar Cascade Algorithm: It is an object detection system that can recognize faces in still photos or moving videos. In order to recognize a face, Haar cascade classifiers are trained on a large number of pictures, both with and without faces. This means that instead of choosing non-facial areas, Haar classifiers scan the window for faces. According to research, image processing and pattern recognition experts have found it difficult to reliably identify human faces. An object detection system called the Haar Cascade Algorithm may be used to find faces, pedestrians, objects, and facial emotions. Haar Cascade Classifier is one of the best detectors for face detection from a picture in terms of speed and accuracy. The most important component of the face detection Haar Cascade Classifier is the Haar characteristics. While the Haar Cascade Classifier features can detect any item, these features are utilized to determine if a feature is present in a particular picture. They work and focuses on precise eye central localization using a webcam. Five distinct procedures were employed in the investigation. The study of Viola and Jones is depicted in Fig. below. A cascade function is developed using the various quantities of pictures, both positive and negative, in this machine learning-based technique. It is then utilized to find things in other images in that situation. To increase classification performance in this study, the system requires many positive pictures (photos with faces) and negative images (photos without faces). Then I must take the features out of it. According to Soo, "A Haar-like feature checks nearby rectangular areas at a certain location in a detection window, adds the pixel intensities in each region, and computes the difference between these sums."

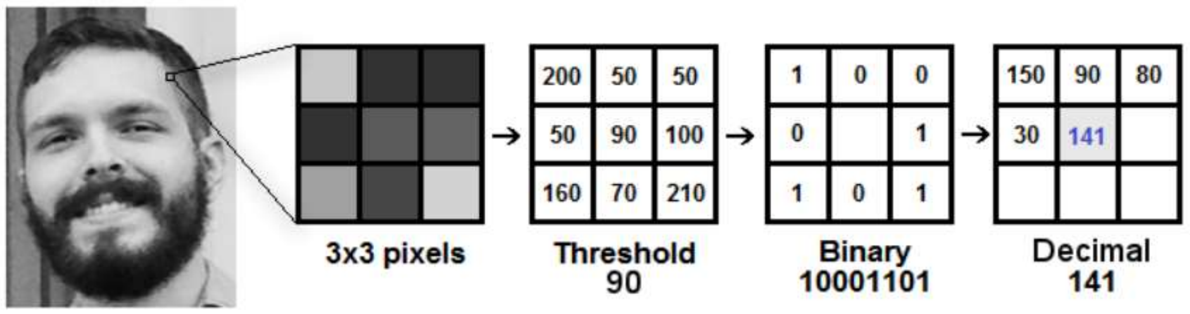


Local Binary Pattern Histogram (LBPH): It was first described in 1994 (LBP) and has since been found to be a strong feature for texture classification. Using the LBP combined with histograms we'll represent the face images with a straightforward data vector. As LBP is a visual descriptor it can even be used for face recognition tasks, as is commonly seen within the subsequent step-by-step explanation.

1.Parameters: the LBPH uses 4 parameters: 1. Radius: is the circular binary pattern that represents the radius around the central pixel. that will be 0 or 1. 2. Neighbours: it is the full number of sample points that are employed to form the circular binary pattern. If the quantity of sample points increases, the computational cost also increases so it's usually set to eight. 3.Grid X: It is the entire number of cells in an exceedingly very horizontal direction. If the number of cells increases the grid is better, and also the dimensionality of the resulting vector also will increase it is also usually set to eight. 4.Grid Y: It is the overall number of cells in an exceedingly vertical direction. If the number of cells increases the grid is better, and also the dimensionality of the resulting vector will increase it is also usually set to eight.

2.Training the Algorithm: First, we need to train the algorithm. To do so, we need to use a dataset with the facial images of the people we want to recognize. We need to also set an ID (it may be a number or the name of the person) for each image, so the algorithm will use this information to recognize an input image and give you an output. Images of the same person must have the same ID. With the training set already constructed, let's see the LBPH computational steps.

3. Applying the LBP operation: The first computational step of the LBPH is to create an intermediate image that describes the original image in a better way, by highlighting the facial characteristics. To do so, the algorithm uses a concept of a sliding window, based on the parameter's radius and neighbour. The image below shows this procedure:



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

- Suppose we have a facial image in gray scale.

- We can get part of this image as a window of 3x3 pixels.

- It can also be represented as a 3x3 matrix containing the intensity of each pixel (0~255).

- Then, we need to take the central value of the matrix to be used as the threshold.

- This value will be used to define the new values from the 8 neighbors.

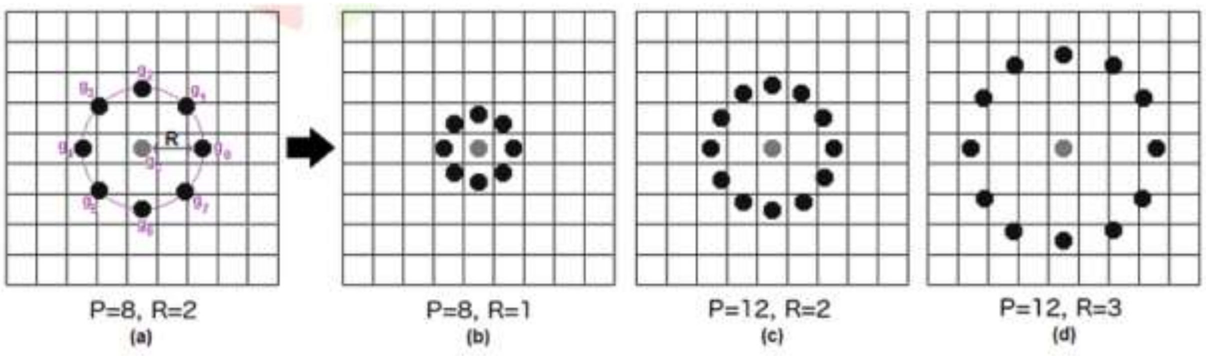
- For each neighbor of the central value (threshold), we set a new binary value. We set 1 for values equal or higher than the threshold and 0 for values lower than the threshold.

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

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

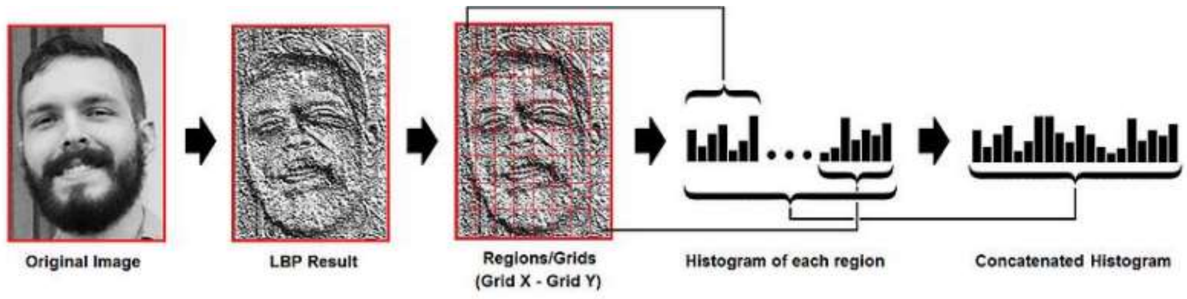
- At the end of this procedure (LBP procedure), we have a new image which represents better the characteristics of the original image.

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



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

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



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

• As we have an image in gray scale, each histogram (from each grid) will contain only 256 positions (0~255) representing the occurrences of each pixel intensity.

• Then, we need to concatenate each histogram to create a new and bigger histogram. Supposing we have 8x8 grids, we will have 8x8x256=16.384 positions in the final histogram. The final histogram represents the characteristics of the image original image.

The LBPH algorithm is pretty much it.

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

• So, to find the image that matches the input image we just need to compare two histograms and return the image with the closest histogram.

• We can use various approaches to compare the histograms (calculate the distance between two histograms), for example: Euclidean distance, chi-square, absolute value, etc. In this example, we can use the Euclidean distance (which is quite known) based on the following formula:

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

- So, the algorithm output is the ID from the image with the closest histogram. The algorithm should also return the calculated distance, which can be used as a '**confidence**' measurement. **Note**: don't be fooled about the 'confidence' name, as lower confidences are better because it means the distance between the two histograms is closer.

- We can then use a threshold and the 'confidence' to automatically estimate if the algorithm has correctly recognized the image. We can assume that the algorithm has successfully recognized if the confidence is lower than the threshold defined.

3. FEASIBILITY REPORT:

The feasibility report assesses the viability of implementing the Face Recognition Attendance System. It examines technical, economic, and operational aspects to determine the project's feasibility.

3.1 Economical feasibility:

Cost-Benefit Analysis: The initial investment in software development, hardware procurement, and training may be significant. However, the long-term benefits of automation, accuracy, and efficiency outweigh the costs.

Return on Investment (ROI): By reducing manual labour, minimizing errors, and enhancing productivity, the system promises a favourable ROI over time.

Scalability: The ability to scale the system according to organizational needs ensures its economic feasibility for both small and large enterprises.

3.2 Technology feasibility:

Facial Recognition Technology: The availability of advanced facial recognition algorithms and libraries such as OpenCV ensures the technical feasibility of the system.

Hardware Requirements: Standard hardware components like cameras and computers are readily available and compatible with the system.

Integration: Compatibility with existing software and databases enhances the technical feasibility of the project.

3.3 Operational feasibility:

User Acceptance: User-friendly interfaces and intuitive workflows facilitate user adoption and acceptance.

Training Requirements: Minimal training is required for administrators and users to operate the system effectively.

Integration with Workflow: Seamless integration with existing attendance management processes ensures operational feasibility without disrupting daily operations.

3.4 Behavioural Feasibility:

User Acceptance: Evaluate the attitudes and opinions of potential users, including administrators, teachers, employees, and students, towards the implementation of facial recognition technology for attendance tracking.

Training Requirements: Determine the level of training required for users to effectively use the system.

Organizational Culture: Consider the existing organizational culture and how receptive it is to innovation and technological advancements.

4. WORKING OF PRESENT SYSTEM:

Manual attendance systems suffer from several drawbacks that hinder their effectiveness in accurately tracking attendance. Firstly, they are prone to errors, whether due to illegible handwriting, miscounting, or data entry mistakes, leading to unreliable attendance records. This inaccuracy can result in financial losses or regulatory compliance issues for organizations. Additionally, manual attendance processes are highly time-consuming, consuming valuable resources from both administrators and attendees. The manual recording of attendance also opens the door to fraudulent practices such as buddy punching or proxy attendance, where individuals falsely record attendance for themselves or others, further compromising the integrity of the data. Furthermore, manual systems lack real-time monitoring capabilities, making it challenging for administrators to promptly identify and address attendance discrepancies or patterns.

5.CHARACTERSTICS OF THE PROPOSED SYSTEM:

5.1 Facial Recognition Technology: The proposed system leverages advanced facial recognition technology to accurately identify individuals based on unique facial features, ensuring precise attendance tracking without the need for manual input.

5.2 Automation: Unlike manual attendance systems, the proposed system automates the attendance tracking process, saving time and effort for both administrators and attendees. Attendees simply need to be present in front of a camera, and their attendance is automatically recorded.

5.3 Real-Time Monitoring: The system provides real-time monitoring of attendance data, enabling administrators to access up-to-date information on attendance status instantly. This allows for timely interventions in case of discrepancies or issues.

5.4 Customization: The proposed system offers flexibility in configuration, allowing administrators to tailor settings according to specific requirements, such as attendance policies, reporting formats, and integration with existing infrastructure.

5.5 Scalability: Designed to accommodate organizations of various sizes and industries, the system is scalable and can be easily deployed and expanded to meet the growing needs of the organization.

6. INTRODUCTION TO SELECTED SOFTWARE:

- **Software Used OpenCV python software:** OpenCV (Open Source Computer Vision Library) is a Python library that allows you to perform image processing and computer vision tasks. It provides a wide range of features, including object detection, face recognition, and tracking. So , it is very important to install OpenCV.

- **VS code:** Visual Studio Code, also commonly referred to as VS Code ,is a source-code editor made by Microsoft for Windows , Linux and macOS. Features include support for debugging , syntax highlighting , intelligent code completion , snippets , code refactoring , and embedded Git .

- **Tkinter :** Tkinter is the standard GUI library for Python. Python when combined with Tkinter provides a fast and easy way to create GUI applications. Tkinter provides a powerful object-oriented interface to the Tk GUI toolkit. Creating a GUI application using Tkinter is an easy task .

- **MySQL Database:** The most important step is to create a database. A database is required where all the data of each and every student in a particular class is placed. The basic process in creating a database is to form a system that take images. For that we will be using a camera module and pictures of each student registered in a particular class is taken. These pictures are taken and basic image processing technique is used to get data understandable by the computer and store it in .csv file

6.1 Introduction to Python:

Python is a high-level, interpreted programming language renowned for its simplicity, readability, and versatility. Created by Guido van Rossum and first released in 1991, Python has since gained immense popularity and widespread adoption across various domains, including web development, data science, artificial intelligence, automation, and more.

One of Python's defining features is its clear and concise syntax, which emphasizes readability and reduces the time and effort required for development. This makes Python an ideal choice for both beginners and experienced developers alike. Python's minimalist philosophy, often summarized as "There should be one-- and preferably only one --obvious way to do it," encourages code that is easy to understand and maintain.

Python boasts a comprehensive standard library, offering a wide range of modules and functions for tasks such as file I/O, networking, database access, and more. Additionally, Python's extensive ecosystem of third-party libraries and frameworks, such as Django, Flask, NumPy, and TensorFlow, further extends its capabilities and enables developers to tackle diverse challenges efficiently.

The language's dynamic typing and automatic memory management contribute to its flexibility and ease of use, allowing developers to focus on solving problems rather than managing low-level details. Python supports multiple programming paradigms, including procedural, object-oriented, and functional programming, giving developers the freedom to choose the most suitable approach for their projects.

Python's cross-platform compatibility ensures that code written in Python can run seamlessly on various operating systems, including Windows, macOS, and Linux. Furthermore, Python's open-source nature fosters a vibrant community of developers who contribute to its ongoing development, share knowledge, and provide support through forums, tutorials, and online resources.

Python versions:

Version	Release Date	End of Support	Significant Features
Python 1.0	January 1994	December 31, 2000	Initial release
Python 1.5	December 1997	June 15, 2001	Support for modules and garbage collection improvements
Python 1.6	September 2000	October 5, 2002	Improved Unicode support, better built-in types
Python 2.0	October 2000	April 30, 2010	List comprehensions, garbage collection improvements
Python 2.1	April 2001	April 30, 2010	Improved Unicode support, nested scopes
Python 2.2	December 2001	April 30, 2010	Generator expressions, new-style classes
Python 2.3	July 2003	July 29, 2006	Generator enhancements, new string methods
Python 2.4	November 2004	October 18, 2008	Decorators, generator expressions, new built-in functions

Python 2.5	September 2006	May 26, 2011	Conditional expressions, with statement, absolute import
Python 2.6	October 2008	October 29, 2013	Abstract base classes, optimized dictionary
Python 2.7	July 2010	January 1, 2020	Final release of Python 2.x series
Python 3.0	December 2008	January 1, 2020	Major overhaul, including print function, Unicode support
Python 3.1	June 2009	July 3, 2011	New syntax features, library improvements
Python 3.2	February 2011	February 20, 2020	New multiprocessing module, improved stdlib
Python 3.3	September 2012	September 29, 2017	Exception chaining, yield from expression
Python 3.4	March 2014	March 18, 2019	async/await syntax, statistics module
Python 3.5	September 2015	September 13, 2020	Matrix multiplication operator (@), async/await improvements
Python 3.6	December 2016	December 23, 2021	f-strings, pathlib module
Python 3.7	June 2018	June 27, 2023	Data classes, improved asyncio support
Python 3.8	October 2019	October 14, 2024	Walrus operator, positional-only arguments
Python 3.9	October 2020	October 5, 2025	Dictionary merge and update operators, Type hinting enhancements
Python 3.10	October 2021	October 2026 (Planned)	Pattern matching, improved error messages

6.2 Introduction to SQL:

MySQL is an open-source relational database management system (RDBMS) renowned for its reliability, scalability, and performance. Initially developed by MySQL AB and later acquired by Oracle Corporation, MySQL has become one of the most widely used database systems, powering a vast array of applications ranging from small-scale websites to large-scale enterprise systems.

As a relational database, MySQL organizes data into tables consisting of rows and columns, following the principles of the structured query language (SQL). SQL allows users to interact with the database by performing various operations such as querying, inserting, updating, and deleting data, as well as defining database schema, constraints, and relationships.

MySQL offers a range of features that contribute to its popularity and versatility. These include:
 Ease of Use: MySQL is known for its user-friendly interface and straightforward setup process, making it accessible to developers of all skill levels. Its comprehensive documentation and extensive community support further facilitate learning and troubleshooting.

Performance: MySQL is optimized for high performance, with efficient storage engines, indexing mechanisms, and caching strategies that ensure rapid data retrieval and processing.

Scalability: MySQL's architecture supports horizontal and vertical scalability, allowing organizations to scale their database infrastructure to accommodate growing data volumes and user demands.

Security: MySQL provides robust security features to protect sensitive data from unauthorized access, including user authentication, access control, encryption, and auditing capabilities. This ensures compliance with data privacy regulations and safeguards against security breaches.

Compatibility: MySQL is cross-platform compatible, meaning it can run on various operating systems including Windows, macOS, Linux, and Unix. Additionally, MySQL supports multiple programming languages and frameworks, enabling seamless integration with different application stacks.

Version of My SQL:

Version	Release Date	End of Support	Significant Features
MySQL 3.23	January 2001	December 31, 2006	Introduction of InnoDB storage engine, B-tree indexing
MySQL 4.0	March 2003	December 31, 2008	Unicode support, subqueries, stored procedures

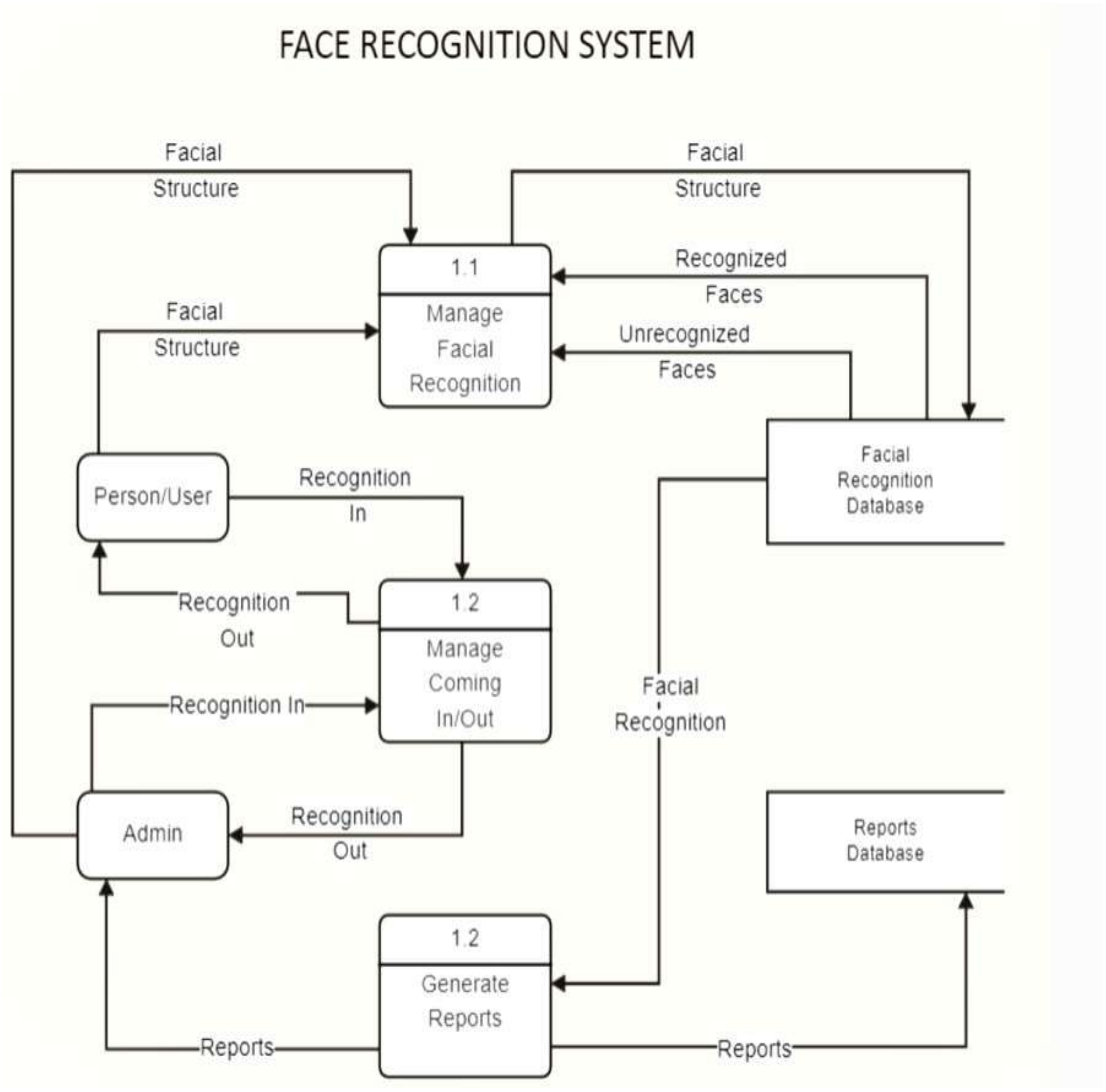
MySQL 4.1	October 2004	December 31, 2009	Views, triggers, SSL support, query caching
MySQL 5.0	October 2005	December 31, 2013	Stored procedures, triggers, XA transactions, subqueries
MySQL 5.1	November 2008	December 31, 2013	Partitioning, row-based replication, plugin API
MySQL 5.5	December 2010	December 31, 2018	InnoDB as default storage engine, semi-synchronous replication
MySQL 5.6	February 2013	February 5, 2021	Online schema changes, NoSQL interfaces, improved replication
MySQL 5.7	October 2015	October 21, 2023	JSON support, improved security features, optimizer enhancements
MySQL 8.0	April 2018	April 2026 (Planned)	Window functions, Common Table Expressions (CTEs), improved JSON support

7. SYSTEM DESIGN:

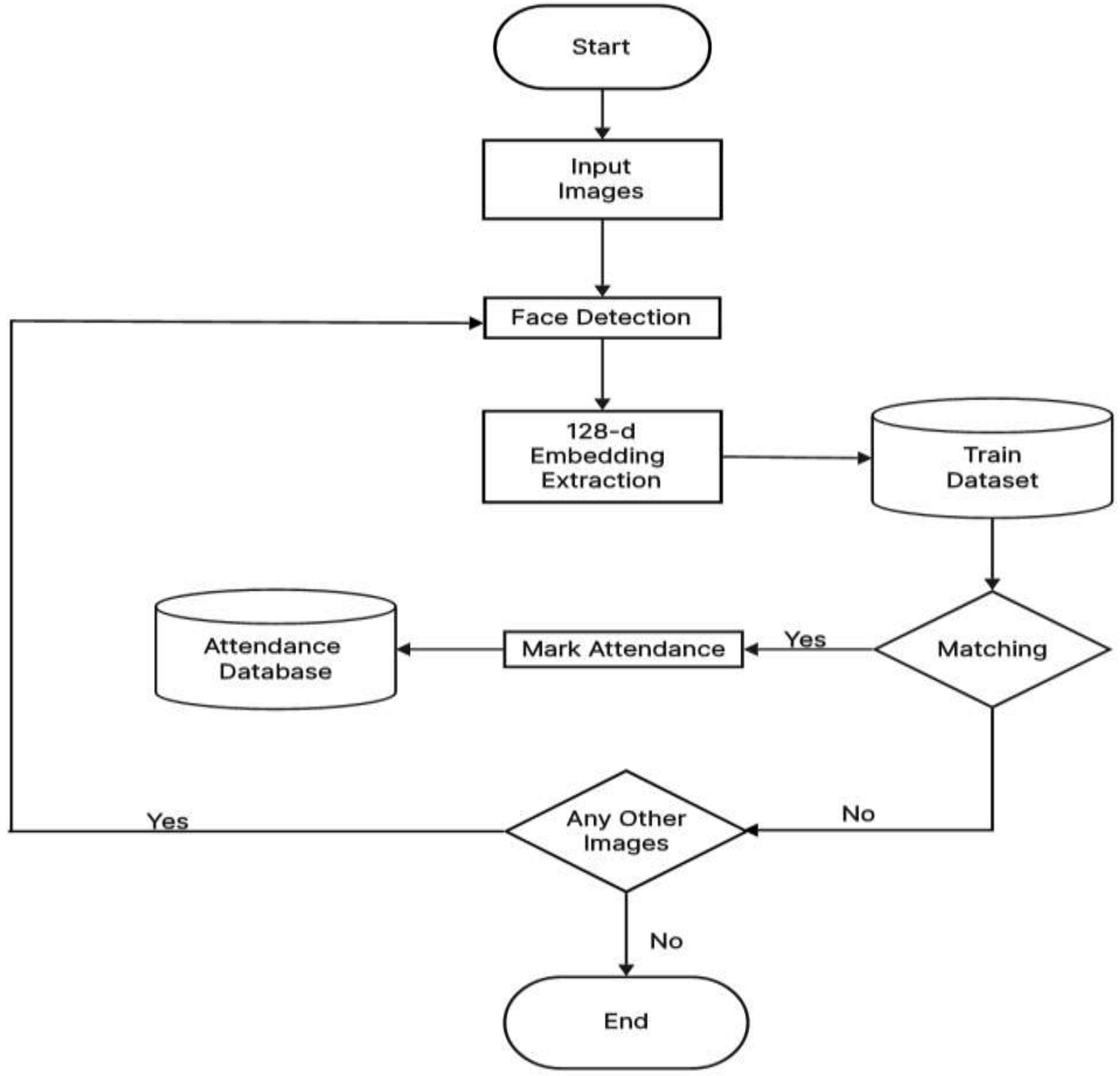
7.1 Introduction:

The main rule of the project is that the video captured by the camera is converted into an image for viewing. In addition, a known code image is also provided, otherwise the system will mark the site as non-existent. 1. Take a Video: The camera is positioned at a selected distance within the classroom to capture pre-video videos of perfect students of the class. 2. Divide as Frames in Video: The captured video must be converted to self-contained every second so that it can be easily accessed and seen by the students' faces in order to present the audience. 3. Face Detection: Face detection is a process by which an image, provided as input (image) is searched to investigate any face, after finding the face processing image cleans the face image so that the face can easily be seen. 4. Facial Recognition: After completing the face detection and analysis, it is compared with the existing face on the student website to review the Individual presence.

7.2 ER-DIAGRAM:



7.3 WORKFLOW DIAGRAM:



8.CODING:

8.1 Registration Frame:

```
from tkinter import *
from tkinter import ttk
from PIL import ImageTk
from tkinter import messagebox
import mysql.connector


class Register:
    def __init__(self, root):
        self.root = root
        self.root.title("Register")
        self.root.geometry("1300x700+0+0")
        # ================Variables============
        self.var_faname = StringVar()
        self.var_iname = StringVar()
        self.var_contact = StringVar()
        self.var_email = StringVar()
        self.var_securityQ = StringVar()
        self.var_securityA = StringVar()
        self.var_password = StringVar()
        self.var_pcf = StringVar()
        self.var_check = IntVar()

        self.bg = ImageTk.PhotoImage(file=r"college_images/bg-reg.jpg")
        bg_lb1 = Label(self.root, image=self.bg)
        bg_lb1.place(x=0, y=0, relwidth=1, relheight=1)

        #################################################################
        frame = Frame(self.root, bg="white")
        frame.place(x=300, y=100, width=800, height=501)

        register_lb1 = Label(frame, text="REGISTER HERE", font=("times new roman", 20, "bold"),
fg="green", bg="white")
        register_lb1.place(x=20, y=20)
        ##########################################
        fname = Label(frame, text="First Name", font=("times new roman", 15, "bold"), bg="white")
        fname.place(x=50, y=70)

        fname_entry = ttk.Entry(frame, textvariable=self.var_faname, font=("times new roman", 15, "bold"))
        fname_entry.place(x=50, y=100, width=250)
        lname = Label(frame, text="Last Name", font=("times new roman", 15, "bold"), bg="white")
        lname.place(x=400, y=70)

        self.lname_entry = ttk.Entry(frame, textvariable=self.var_iname, font=("times new roman", 15, "bold"))
        self.lname_entry.place(x=400, y=100, width=250)
        # ------------------------------------------------------

        contact = Label(frame, text="Contact No", font=("times new roman", 15, "bold"), bg="white")
        contact.place(x=50, y=145)

        self.txt_contact = ttk.Entry(frame, textvariable=self.var_contact, font=("times new roman", 15, "bold"))
        self.txt_contact.place(x=50, y=175, width=250)

        email = Label(frame, text="Email", font=("times new roman", 15, "bold"), bg="white")
        email.place(x=400, y=145, width=250)
        self.txt_email = ttk.Entry(frame, textvariable=self.var_email, font=("times new roman", 15, "bold"))
        self.txt_email.place(x=400, y=175, width=250)
        # --------------------------------------------------

        security_Q = Label(frame, text="Select Security Questions", font=("times new roman", 15, "bold"),
bg="white",
                           fg="black")
        security_Q.place(x=50, y=220)

        self.combo_security_Q = ttk.Combobox(frame, textvariable=self.var_securityQ, font=("times new
roman", 15),
                                             state="read only")
        self.combo_security_Q["values"] = ("Select", "Birth Place", "Dob", "Pet Name", "friend Name")
        self.combo_security_Q.place(x=50, y=260, width="250")
        self.combo_security_Q.current(0)

        security_A = Label(frame, text="Select Security Answer", font=("times new roman", 15, "bold"),
bg="white",
                           fg="black")
        security_A.place(x=400, y=220)
        self.txt_security_A = ttk.Entry(frame, textvariable=self.var_securityA, font=("times new roman", 15,
"bold"))
        self.txt_security_A.place(x=400, y=260, width=250)

        # -------------------------------------------------------

        passw = Label(frame, text="Password", font=("times new roman", 15, "bold"), bg="white",
fg="black")
        passw.place(x=50, y=300)
        self.txt_passw = ttk.Entry(frame, textvariable=self.var_password, font=("times new roman", 15,
"bold"))
        self.txt_passw.place(x=50, y=340, width=250)

        pcf = Label(frame, text="Confirm Password", font=("times new roman", 15, "bold"), bg="white",
fg="black")
        pcf.place(x=400, y=300)
        self.txt_pcf = ttk.Entry(frame, textvariable=self.var_pcf, font=("times new roman", 15, "bold"))
        self.txt_pcf.place(x=400, y=340, width=250)

        # ---------------------

        chebtn = Checkbutton(frame, variable=self.var_check, text="I Agree Terms & Condition",
                             font=("times new roman", 12, "bold"),
                             onvalue=1, offvalue=0)
        chebtn.place(x=50, y=390)
        # ---------------------------------------------#
        btn1 = Button(frame, text="Register Now", font=20, bg="cyan", command=self.data)
        btn1.place(x=400, y=390)

        btn2 = Button(frame, text="Login", font=20, bg="cyan", command=self.return_login)
        btn2.place(x=600, y=390)

    # -----------------------------------------------

    def data(self):
        if self.var_faname.get() == "" or self.var_email.get() == "" or self.var_securityQ.get() == "select":
            messagebox.showerror("Error", "All field are required")
        elif self.var_password.get() != self.var_pcf.get():
            messagebox.showerror("Error", "Password must be same")
        elif self.var_check.get() == 0:
            messagebox.showerror("Error", "Please agree our term and condition")
        else:
            con = mysql.connector.connect(host="localhost", username="root", password="vedant@9765",
                                          database="face_recognizer")
            cur = con.cursor()
            query = ("select * from register where email = %s")
            value = (self.var_email.get())
            cur.execute(query, (value,))
            data = cur.fetchone()
            if data != None:
                messagebox.showerror("Error", "user already exist, please try another email")
            else:
                cur.execute("insert into register values(%s,%s,%s,%s,%s,%s,%s)", (self.var_faname.get(),
                                                                                self.var_iname.get(),
                                                                                self.var_contact.get(),
                                                   self.var_email.get(),
                                                   self.var_securityQ.get(),
                                                   self.var_securityA.get(),
                                                   self.var_password.get()
                                                   ))
            con.commit()
            con.close()
            messagebox.showinfo("Success", "Registration Successful")

    def return_login(self):
        self.root.destroy()


if __name__ == "__main__":
    root = Tk()
    obj = Register(root)
    root.mainloop()
```

8.2 Login Frame:

```
from tkinter import *
from tkinter import ttk
from tkinter import messagebox
from PIL import Image, ImageTk
import mysql.connector
from main import Face_Recognition_System
from regester import Register


def main():
    win = Tk()
    app = LoginWindow(win)
    win.mainloop()


class LoginWindow:
    def __init__(self, root):
        self.root = root
        self.root.geometry("600x400")
        self.root.title("Creative Login Page")

        # ================Variables============
        self.var_email = StringVar()
        self.var_password = StringVar()
        # Background Image (Subtle Gradient)
        gradient_img = Image.new("RGB", (600, 400), "#009688")  # Green color
        self.gradient_photo = ImageTk.PhotoImage(gradient_img)
        gradient_label = Label(self.root, image=self.gradient_photo)
        gradient_label.place(x=0, y=0, relwidth=1, relheight=1)

        # Title
        title_label = Label(self.root, text="Welcome to Our System", font=("Helvetica", 20, "bold"),
fg="white",
                            bg="#009688")
        title_label.pack(pady=20)

        # Frame
        frame = Frame(self.root, bg="white", bd=10)
        frame.place(relx=0.5, rely=0.5, anchor=CENTER)

        # Username Label and Entry
        username_lbl = Label(frame, text="Username", font=("Arial", 12), fg="#333", bg="white")
        username_lbl.grid(row=0, column=0, padx=10, pady=5, sticky="w")
        self.txtuser = ttk.Entry(frame, font=("Arial", 12))
        self.txtuser.grid(row=0, column=1, padx=10, pady=5, sticky="w")

        # Password Label and Entry
        password_lbl = Label(frame, text="Password", font=("Arial", 12), fg="#333", bg="white")
        password_lbl.grid(row=1, column=0, padx=10, pady=5, sticky="w")
        self.txtpass = ttk.Entry(frame, show="*", font=("Arial", 12))
        self.txtpass.grid(row=1, column=1, padx=10, pady=5, sticky="w")

        # Login Button
        loginbtn = Button(frame, text="Login", font=("Arial", 12, "bold"),
                          bd=3, relief=RIDGE, fg="white", bg="#009688",
                          activeforeground="white", activebackground="#004D40", command=self.login)
        loginbtn.grid(row=2, column=0, columnspan=2, pady=20)

        # Register Button
        registerbtn = Button(frame, text="New User Register", font=("Arial", 10, "bold"),
                             bd=3, relief=RIDGE, fg="white", bg="#004D40",
                             activeforeground="white", activebackground="#009688", command=self.rigister_window)
        registerbtn.grid(row=3, column=0, columnspan=2, pady=10)

        # Forget Password Button
        forget_pass_btn = Button(frame, text="Forget Password", font=("Arial", 10, "bold"),
                                 bd=3, relief=RIDGE, borderwidth=0, fg="white", bg="#004D40",
                                 activeforeground="white", activebackground="#009688",
command=self.forget_password)
```

```
    forget_pass_btn.grid(row=4, column=0, columnspan=2, pady=10)

def login(self):
    if self.txtuser.get() == "" or self.txtpass.get() == "":
        messagebox.showerror("Error", "All fields are required")
    elif self.txtuser.get() == "user" and self.txtpass.get() == "password":
        messagebox.showinfo("Success", "Welcome to Our System")
    else:
        con = mysql.connector.connect(host="localhost", username="root", password="vedant@9765",
                                      database="face_recognizer")
        cur = con.cursor()
        cur.execute("select * from register where email=%s and password=%s",
                    (
                        self.txtuser.get(),
                        self.txtpass.get()
                    ))
        row = cur.fetchone()

        if row == None:
            messagebox.showerror("Error", "Invalid username and password")
        else:
            open_main = messagebox.askyesno("YesNo", "Access only admin")
            if open_main > 0:
                self.new_window = Toplevel(self.root)
                self.app = Face_Recognition_System(self.new_window)
            else:
                if not open_main:
                    return
        con.commit()
        con.close()

# ============Reset password==============
def reset_pass(self):
    if self.combo_security_Q.get() == "Select":
        messagebox.showerror("Error", "Select the security Question", parent=self.root2)
    elif self.txt_security_A.get() == "":
        messagebox.showerror("Error", "Please Enter Answer", parent=self.root2)
    elif self.txt_new_pass.get() == "":
        messagebox.showerror("Error", "Please Enter new password", parent=self.root2)
    else:
        con = mysql.connector.connect(host="localhost", username="root", password="vedant@9765",
                                      database="face_recognizer")
        cur = con.cursor()
        query = ("select * from register where email=%s and securityQ=%s and securityA=%s ")
        value = (self.txtuser.get(), self.combo_security_Q.get(), self.txt_security_A.get())
```

```
            cur.execute(query, value)
            row = cur.fetchone()
            if row == None:
                messagebox.showerror("Error", "Please enter correct Answer", parent=self.root2)

            else:
                query = ("update register set password=%s where email=%s")
                value = (self.txt_new_pass.get(), self.txtuser.get())
                cur.execute(query, value)
                con.commit()
                con.close()
                messagebox.showinfo("Info", "Your password has been reset ,please login new password",
                                    parent=self.root2)
                self.root2.destroy()

    # ============Forget password==============
    def forget_password(self):
        if self.txtuser.get() == "":
            messagebox.showerror("Error", "Please Enter the Email address to reset password")
        else:
            con = mysql.connector.connect(host="localhost", username="root", password="vedant@9765",
                                          database="face_recognizer")
            cur = con.cursor()
            query = ("select * from register where email=%s")
            value = (self.txtuser.get())
            print(value)
            cur.execute(query, (value,))
            row = cur.fetchone()
            con.commit()
            con.close()
            # print(row)

            if row == None:
                messagebox.showerror("Error", "Please enter valid user name")
            else:
                con.close()
                self.root2 = Toplevel()
                self.root2.title("Forget Password")
                self.root2.geometry("340x450+610+170")
                l = Label(self.root2, text="Forget Password", font=("Helvetica", 20, "bold"), fg="white",
                          bg="#009688")
                l.place(x=0, y=0, relwidth=1)

                security_Q = Label(self.root2, text="Select Security Questions", font=("times new roman", 15,
"bold"),
                            bg="white",
                            fg="black")
                security_Q.place(x=50, y=80)

                self.combo_security_Q = ttk.Combobox(self.root2,
                                           font=("times new roman", 15),
                                           state="read only")
                self.combo_security_Q["values"] = ("Select", "Birth Place", "Dob", "Pet Name", "friend
Name")
                self.combo_security_Q.place(x=50, y=110, width=250)
                self.combo_security_Q.current(0)

                security_A = Label(self.root2, text="Select Security Answer", font=("times new roman", 15,
"bold"),
                            bg="white",
                            fg="black")
                security_A.place(x=50, y=150)
                self.txt_security_A = ttk.Entry(self.root2,
                                         font=("times new roman", 15, "bold"))
                self.txt_security_A.place(x=50, y=180, width=250)

                # =======New Pass=====

                new_pass = Label(self.root2, text="Enter New Password", font=("times new roman", 15,
"bold"),
                   bg="white",
                   fg="black")
                new_pass.place(x=50, y=220)
                self.txt_new_pass = ttk.Entry(self.root2,
                                        font=("times new roman", 15, "bold"))
                self.txt_new_pass.place(x=50, y=250, width=250)

                btn = Button(self.root2, text="Reset", font=("times new roman", 15, "bold"), fg="red",
bg="white",
                  command=self.reset_pass)
                btn.place(x=150, y=290)

    def rigister_window(self):
        self.new_window = Toplevel(self.root)
        self.app = Register(self.new_window)


if __name__ == "__main__":
    main()
```

8.3 Main Frame:

```
import os
import tkinter
import tkinter.messagebox
from tkinter import *
from PIL import Image, ImageTk
from time import strftime
from student import Student
from train import Train
from face_recgnition import Face_Recgnition
from attendance import Attendance
from developer import Developer
from help import Help


class Face_Recognition_System:
    def __init__(self, root):
        self.root = root
        self.root.geometry("1280x790+0+0")
        self.root.title("Face Recognition System")

        #
        img = Image.open(r"college_images\MIT-logo-blue-background-1.png")
        img = img.resize((1280, 170), Image.LANCZOS)
        self.photoimg = ImageTk.PhotoImage(img)

        f_lbl = Label(self.root, image=self.photoimg)
        f_lbl.place(x=0, y=0, width=1280, height=170)

        # bg image
        img3 = Image.open(r"college_images\bg1.jpg")
        img3 = img3.resize((1280, 710), Image.LANCZOS)
        self.photoimg3 = ImageTk.PhotoImage(img3)

        bg_img = Label(self.root, image=self.photoimg3)
        bg_img.place(x=0, y=170, width=1280, height=710)

        title_lb1 = Label(bg_img, text="FACE RECOGNITION ATTENDANCE SYSTEM SOFTWARE",
                          font=("times new roman", 27, "bold"), bg="white", fg="red")
        title_lb1.place(x=0, y=0, width=1280, height=45)

        # ==============Time================
        def time():
            string = strftime('%H:%M:%S %p')
            lb1.config(text=string)
            lb1.after(1000, time)

        lb1 = Label(title_lb1, font=("times new roman", 15), bg="white", fg="blue")
        lb1.place(x=5, y=0, width=115, height=45)
        time()

        # student button
        img4 = Image.open(r"college_images\student_info.png")
        img4 = img4.resize((150, 150), Image.LANCZOS)
        self.photoimg4 = ImageTk.PhotoImage(img4)

        b1 = Button(bg_img, image=self.photoimg4, cursor="hand2", command=self.student_details)
        b1.place(x=100, y=90, width=150, height=150)

        b1 = Button(bg_img, text="Student Details", cursor="hand2", font=("times new roman", 10, "bold"),
    bg="darkblue",
                    fg="white", command=self.student_details)
        b1.place(x=100, y=240, width=150, height=30)

        # Detect Face button
        img5 = Image.open(r"college_images\face_detection.png")
        img5 = img5.resize((150, 150), Image.LANCZOS)
        self.photoimg5 = ImageTk.PhotoImage(img5)

        b1 = Button(bg_img, image=self.photoimg5, cursor="hand2", command=self.face_data)
        b1.place(x=400, y=90, width=150, height=150)

        b1 = Button(bg_img, text="Face Detector", cursor="hand2", font=("times new roman", 10, "bold"),
    bg="darkblue",
                    fg="white", command=self.face_data)
        b1.place(x=400, y=240, width=150, height=30)

        # Attendance face button
        img6 = Image.open(r"college_images\attendance_logo.png")
        img6 = img6.resize((150, 150), Image.LANCZOS)
        self.photoimg6 = ImageTk.PhotoImage(img6)

        b1 = Button(bg_img, image=self.photoimg6, cursor="hand2", command=self.stud_attendance)
        b1.place(x=700, y=90, width=150, height=150)

        b1 = Button(bg_img, text="Attendance", cursor="hand2", font=("times new roman", 10, "bold"),
    bg="darkblue",
                    fg="white", command=self.stud_attendance)
        b1.place(x=700, y=240, width=150, height=30)

        # Help button
        img7 = Image.open(r"C:college_images\help.png")
        img7 = img7.resize((150, 150), Image.LANCZOS)
        self.photoimg7 = ImageTk.PhotoImage(img7)

        b1 = Button(bg_img, image=self.photoimg7, cursor="hand2", command=self.std_help)
        b1.place(x=1000, y=90, width=150, height=150)

        b1 = Button(bg_img, text="Help Desk", cursor="hand2", font=("times new roman", 10, "bold"),
bg="darkblue",
                    fg="white", command=self.std_help)
        b1.place(x=1000, y=240, width=150, height=30)

        # Train face button
        img8 = Image.open(r"college_images\traindata.png")
        img8 = img8.resize((150, 150), Image.LANCZOS)
        self.photoimg8 = ImageTk.PhotoImage(img8)

        b1 = Button(bg_img, image=self.photoimg8, cursor="hand2", command=self.train_data)
        b1.place(x=100, y=290, width=150, height=150)

        b1 = Button(bg_img, text="Train Data", cursor="hand2", font=("times new roman", 10, "bold"),
bg="darkblue",
                    fg="white", command=self.train_data)
        b1.place(x=100, y=440, width=150, height=30)

        # Photo Face Button
        img9 = Image.open(r"college_images\photo.jpg")
        img9 = img9.resize((150, 150), Image.LANCZOS)
        self.photoimg9 = ImageTk.PhotoImage(img9)

        b1 = Button(bg_img, image=self.photoimg9, cursor="hand2", command=self.open_img)
        b1.place(x=400, y=290, width=150, height=150)

        b1 = Button(bg_img, text="Photo", cursor="hand2", font=("times new roman", 10, "bold"),
bg="darkblue",
                    fg="white", command=self.open_img)
        b1.place(x=400, y=440, width=150, height=30)

        # Developer Button
        img10 = Image.open(r"college_images\developer.jpg")
        img10 = img10.resize((150, 150), Image.LANCZOS)
        self.photoimg10 = ImageTk.PhotoImage(img10)
        b1 = Button(bg_img, image=self.photoimg10, cursor="hand2", command=self.std_dev)
        b1.place(x=700, y=290, width=150, height=150)

        b1 = Button(bg_img, text="Developer", cursor="hand2", font=("times new roman", 10, "bold"),
bg="darkblue",
                    fg="white", command=self.std_dev)
        b1.place(x=700, y=440, width=150, height=30)

        # Exit Button
        img11 = Image.open(r"college_images\exit1.jpg")
        img11 = img11.resize((150, 150), Image.LANCZOS)
        self.photoimg11 = ImageTk.PhotoImage(img11)

        b1 = Button(bg_img, image=self.photoimg11, cursor="hand2", command=self.iExit)
        b1.place(x=1000, y=290, width=150, height=150)

        b1 = Button(bg_img, text="Exit", cursor="hand2", font=("times new roman", 10, "bold"),
bg="darkblue",
                    fg="white", command=self.iExit)
        b1.place(x=1000, y=440, width=150, height=30)

    def open_img(self):
        os.startfile("data")

    # =====================================Functions
buttos=====================+

    def student_details(self):
        self.new_window = Toplevel(self.root)
        self.app = Student(self.new_window)

    def train_data(self):
        self.new_window = Toplevel(self.root)
        self.app = Train(self.new_window)

    def face_data(self):
        self.new_window = Toplevel(self.root)
        self.app = Face_Recgnition(self.new_window)

    def stud_attendance(self):
        self.new_window = Toplevel(self.root)
        self.app = Attendance(self.new_window)

    def std_dev(self):
        self.new_window = Toplevel(self.root)
        self.app = Developer(self.new_window)

    def std_help(self):
        self.new_window = Toplevel(self.root)
        self.app = Help(self.new_window)

    def iExit(self):
        self.exit = tkinter.messagebox.askyesno("Face Recognition", "Sure do you want to exit",
parent=self.root)
        if self.exit == True:
            self.root.destroy()
        else:
            return False


if __name__ == "__main__":
    root = Tk()
    obj = Face_Recognition_System(root)
    root.mainloop()
```

8.4 Student Details Frame:

```
from tkinter import *
from tkinter import ttk
from PIL import Image, ImageTk
from tkinter import messagebox
import mysql.connector
import cv2


class Student:
    def __init__(self, root):
        self.root = root
        self.root.geometry("1280x790+0+0")
        self.root.title("Face Recognition System")

        # ===============variables=============
        self.var_dep = StringVar()
        self.var_course = StringVar()
        self.var_year = StringVar()
        self.var_semester = StringVar()
        self.var_std_id = StringVar()
        self.var_std_name = StringVar()
        self.var_div = StringVar()
        self.var_roll = StringVar()
        self.var_gender = StringVar()
        self.var_dob = StringVar()
        self.var_email = StringVar()
        self.var_phone = StringVar()
        self.var_address = StringVar()
        self.var_teacher = StringVar()

        #
        img = Image.open(r"college_images\s1.jpg")
        img = img.resize((440, 90), Image.LANCZOS)
        self.photoimg = ImageTk.PhotoImage(img)

        f_lbl = Label(self.root, image=self.photoimg)
        f_lbl.place(x=0, y=0, width=440, height=90)

        # 1
        img1 = Image.open(r"college_images\s2.jpg")
        img1 = img1.resize((440, 90), Image.LANCZOS)
        self.photoimg1 = ImageTk.PhotoImage(img1)

        f_lbl = Label(self.root, image=self.photoimg1)
        f_lbl.place(x=440, y=0, width=440, height=90)

        # 2
        img2 = Image.open(r"college_images\s3.jpg")
        img2 = img2.resize((440, 90), Image.LANCZOS)
        self.photoimg2 = ImageTk.PhotoImage(img2)

        f_lbl = Label(self.root, image=self.photoimg2)
        f_lbl.place(x=880, y=0, width=440, height=90)

        # bg image
        img3 = Image.open(r"college_images\bg.jpg")
        img3 = img3.resize((1280, 790), Image.LANCZOS)
        self.photoimg3 = ImageTk.PhotoImage(img3)

        bg_img = Label(self.root, image=self.photoimg3)
        bg_img.place(x=0, y=90, width=1280, height=710)

        title_lb1 = Label(bg_img, text="STUDENT MANAGEMENT SYSTEM",
                          font=("times new roman", 20, "bold"), bg="violet", fg="lavender")
        title_lb1.place(x=0, y=0, width=1280, height=35)
```

```
        main_frame = Frame(bg_img, bd=2, bg="white")
        main_frame.place(x=10, y=35, width=1250, height=525)

        # left label frame
        Left_frame = LabelFrame(main_frame, bd=2, relief=RIDGE, text="Student Details",
                                font=("times new roman", 12, "bold"))
        Left_frame.place(x=10, y=10, width=625, height=500)

        img_left = Image.open(r"college_images\sl.jpg")
        img_left = img_left.resize((600, 90), Image.LANCZOS)
        self.photoimg_left \
            = ImageTk.PhotoImage(img_left)

        f_lbl = Label(Left_frame, image=self.photoimg_left)
        f_lbl.place(x=5, y=0, width=600, height=90)

        # current Course
        current_course_frame = LabelFrame(Left_frame, bd=2, relief=RIDGE, text="Current Course
Informatin",
                                          font=("times new roman", 11, "bold"))
        current_course_frame.place(x=5, y=100, width=600, height=100)

        # Department
        dep_label = Label(current_course_frame, text="Department",
                          font=("times new roman", 11, "bold"), bg="white")
        dep_label.grid(row=0, column=0, padx=10, sticky=W)

        dep_combo = ttk.Combobox(current_course_frame, textvariable=self.var_dep, font=("times new
roman", 11, "bold"),
                                 width=17,
                                 state="readonly", )
        dep_combo["values"] = ("Select Department", "C.S", "I.T", "B.C.A", "MECHANICAL",
"CIVIL")
        dep_combo.current(0)
        dep_combo.grid(row=0, column=1, padx=2, pady=10, sticky=W)

        # Course
        course_label = Label(current_course_frame, text="Course", font=("times new roman", 11, "bold"),
bg="white")
        course_label.grid(row=0, column=2, padx=10, sticky=W)

        course_combo = ttk.Combobox(current_course_frame, textvariable=self.var_course,
                                    font=("times new roman", 11, "bold"), width=17,
                                    state="readonly", )
        course_combo["values"] = ("Select Course", "B.E", "B.Tech", "B.S.C", "M.Tech", "M.E",
"M.C.A", "M.S.C")
        course_combo.current(0)
        course_combo.grid(row=0, column=3, padx=2, pady=10, sticky=W)

        # Year
        year_label = Label(current_course_frame, text="Year", font=("times new roman", 11, "bold"),
bg="white")
        year_label.grid(row=1, column=0, padx=10, sticky=W)

        year_combo = ttk.Combobox(current_course_frame, textvariable=self.var_year,
                                  font=("times new roman", 11, "bold"), width=17,
                                  state="readonly", )
        year_combo["values"] = ("Select Year", "2020-21", "2021-22", "2022-23", "2023-24", "2024-25")
        year_combo.current(0)
        year_combo.grid(row=1, column=1, pady=10, sticky=W)

        # Semester
        semester_label = Label(current_course_frame, text="Semester", font=("times new roman", 11,
"bold"), bg="white")
        semester_label.grid(row=1, column=2, padx=10, sticky=W)

        semester_combo = ttk.Combobox(current_course_frame, textvariable=self.var_semester,
                                      font=("times new roman", 11, "bold"), width=17,
                                      state="readonly", )
        semester_combo["values"] = (
            "Select Semester", "Semester-1", "Semester-2", "Semester-3", "Semester-4", "Semester-5",
"Semester-6")
        semester_combo.current(0)
        semester_combo.grid(row=1, column=3, padx=2, pady=10, sticky=W)

        # Class Student Information
        class_student_frame = LabelFrame(Left_frame, bd=2, relief=RIDGE, text="Class Student Information",
                                         font=("times new roman", 11, "bold"))
        class_student_frame.place(x=5, y=210, width=600, height=260)

        # Student id
        studentId_label = Label(class_student_frame, text="StudentID:", font=("times new roman", 10,
"bold"),
                                bg="white")
        studentId_label.grid(row=0, column=0, padx=10, pady=5, sticky=W)

        studentId_entry = ttk.Entry(class_student_frame, textvariable=self.var_std_id, width=20,
                                    font=("times new roman", 10, "bold"))
        studentId_entry.grid(row=0, column=1, padx=10, pady=5, sticky=W)
        # Student nam1
        studentName_label = Label(class_student_frame, text="Student Name:", font=("times new roman", 10,
"bold"),
                                  bg="white")
        studentName_label.grid(row=0, column=2, padx=10, pady=5, sticky=W)

        studentName_entry = ttk.Entry(class_student_frame, textvariable=self.var_std_name, width=20,
                                      font=("times new roman", 10, "bold"))
        studentName_entry.grid(row=0, column=3, padx=10, pady=5, sticky=W)

        # Student division
        student_div_label = Label(class_student_frame, text="Division:", font=("times new roman", 10,
"bold"),
                                  bg="white")
        student_div_label.grid(row=1, column=0, padx=10, pady=5, sticky=W)

        gender_combo = ttk.Combobox(class_student_frame, textvariable=self.var_div,
                                    font=("times new roman", 10, "bold"), width=17,
                                    state="readonly", )
        gender_combo["values"] = ("Select Division", "B1", "B2", "B3", "B4")
        gender_combo.current(0)
        gender_combo.grid(row=1, column=1, padx=10, pady=5, sticky=W)

        # Student Roll no
        roll_no_label = Label(class_student_frame, text="Roll No:", font=("times new roman", 10, "bold"),
                              bg="white")
        roll_no_label.grid(row=1, column=2, padx=10, pady=5, sticky=W)

        roll_no_entry = ttk.Entry(class_student_frame, textvariable=self.var_roll, width=20,
                                  font=("times new roman", 10, "bold"))
        roll_no_entry.grid(row=1, column=3, padx=10, pady=5,
                           sticky=W)

        # Gender
        gender_label = Label(class_student_frame, text="Gender:", font=("times new roman", 10, "bold"),
                             bg="white")
        gender_label.grid(row=2, column=0, padx=10, pady=5, sticky=W)

        gender_combo = ttk.Combobox(class_student_frame, textvariable=self.var_gender,
                                    font=("times new roman", 10, "bold"), width=17,
                                    state="readonly", )
        gender_combo["values"] = ("Select Gender", "Male", "Female", "Other")
        gender_combo.current(0)
        gender_combo.grid(row=2, column=1, padx=10, pady=5, sticky=W)
        # Dob
        dob_label = Label(class_student_frame, text="DOB:", font=("times new roman", 10, "bold"),
                          bg="white")
        dob_label.grid(row=2, column=2, padx=10, pady=5, sticky=W)

        dob_entry = ttk.Entry(class_student_frame, textvariable=self.var_dob, width=20,
                              font=("times new roman", 10, "bold"))
        dob_entry.grid(row=2, column=3, padx=10, pady=5,
                       sticky=W)

        # Email
        email_label = Label(class_student_frame, text="Email ID:", font=("times new roman", 10, "bold"),
                            bg="white")
        email_label.grid(row=3, column=0, padx=10, pady=5, sticky=W)

        email_entry = ttk.Entry(class_student_frame, textvariable=self.var_email, width=20,
                                font=("times new roman", 10, "bold"))
        email_entry.grid(row=3, column=1, padx=10, pady=5,
                         sticky=W)

        # phone no
        ph_no_label = Label(class_student_frame, text="Phone No:", font=("times new roman", 10, "bold"),
                            bg="white")
        ph_no_label.grid(row=3, column=2, padx=10, pady=5, sticky=W)

        ph_no_entry = ttk.Entry(class_student_frame, textvariable=self.var_phone, width=20,
                                font=("times new roman", 10, "bold"))
        ph_no_entry.grid(row=3, column=3, padx=10, pady=5,
                         sticky=W)

        # Address
        address_label = Label(class_student_frame, text="Address:", font=("times new roman", 10, "bold"),
                              bg="white")
        address_label.grid(row=4, column=0, padx=10, pady=5, sticky=W)

        address_entry = ttk.Entry(class_student_frame, textvariable=self.var_address, width=20,
                                  font=("times new roman", 10, "bold"))
        address_entry.grid(row=4, column=1, padx=10, pady=5,
                           sticky=W)

        # Teacher name
        teacher_label = Label(class_student_frame, text="Teacher Name:", font=("times new roman", 10,
"bold"),
                              bg="white")
        teacher_label.grid(row=4, column=2, padx=10, pady=5, sticky=W)
        teacher_entry = ttk.Entry(class_student_frame, textvariable=self.var_teacher, width=20,
                              font=("times new roman", 10, "bold"))
        teacher_entry.grid(row=4, column=3, padx=10, pady=5,
                          sticky=W)

        # radio Button
        self.var_radio1 = StringVar()
        radiobtn1 = ttk.Radiobutton(class_student_frame, variable=self.var_radio1, text="Take Photo Sample",
                                   value="Yes")
        radiobtn1.grid(row=6, column=0)

        radiobtn2 = ttk.Radiobutton(class_student_frame, variable=self.var_radio1, text="No Photo Sample",
                                   value="No")
        radiobtn2.grid(row=6, column=1)

        # buttons frame
        btn_frame = Frame(class_student_frame, bd=2, relief=RIDGE)
        btn_frame.place(x=0, y=176, width=595, height=30)

        # save
        save_btn = Button(btn_frame, text="Save", width=20, font=("times new roman", 10, "bold"),
bg="blue", fg="white",
                         command=self.add_data)
        save_btn.grid(row=0, column=0)

        # update
        update_btn = Button(btn_frame, text="Update", width=20, font=("times new roman", 10, "bold"),
bg="blue",
                           fg="white", command=self.update_data)
        update_btn.grid(row=0, column=1)

        # delete
        delete_btn = Button(btn_frame, text="Delete", width=20, font=("times new roman", 10, "bold"),
bg="blue",
                           fg="white", command=self.delete_data)
        delete_btn.grid(row=0, column=2)

        # reset
        reset_btn = Button(btn_frame, text="Reset", width=20, font=("times new roman", 10, "bold"),
bg="blue",
                          fg="white", command=self.reset_data)
        reset_btn.grid(row=0, column=3)

        btn_frame1 = Frame(class_student_frame, bd=2, relief=RIDGE)
        btn_frame1.place(x=0, y=200, width=595, height=30)
        # Take photo sample
        take_photo_btn = Button(btn_frame1, text="Take Photo Sample", width=42, font=("times new
roman", 10, "bold"),
                                bg="blue",
                                fg="white", command=self.generte_dataset)
        take_photo_btn.grid(row=1, column=0)

        # Update photo sample
        update_photo_btn = Button(btn_frame1, text="Update Photo Sample", width=42,
                                  font=("times new roman", 10, "bold"),
                                  bg="blue",
                                  fg="white")
        update_photo_btn.grid(row=1, column=1)

        # Right label frame
        Right_frame = LabelFrame(main_frame, bd=2, relief=RIDGE, text="Student Details",
                                 font=("times new roman", 12, "bold"))
        Right_frame.place(x=630, y=10, width=620, height=500)

        img_right = Image.open(r"college_images\sr.jpg")
        img_right = img_right.resize((600, 90), Image.LANCZOS)
        self.photoimg_right \
            = ImageTk.PhotoImage(img_right)

        f_lbl = Label(Right_frame, image=self.photoimg_right)
        f_lbl.place(x=5, y=0, width=600, height=90)

        # Search System
        search_frame = LabelFrame(Right_frame, bd=2, relief=RIDGE, text="Search System",
                                  font=("times new roman", 11, "bold"))
        search_frame.place(x=5, y=100, width=600, height=50)

        search_label = Label(search_frame, text="Search By:", font=("times new roman", 13, "bold"),
bg="red",
                             fg="white")
        search_label.grid(row=0, column=0, padx=10
                          , pady=0, sticky=W)
        search_combo = ttk.Combobox(search_frame, font=("times new roman", 11, "bold"), width=12,
                                    state="readonly", )
        search_combo["values"] = (
            "Select", "Roll_No", "Phone_No")
        search_combo.current(0)
        search_combo.grid(row=0, column=1, padx=10, pady=0, sticky=W)
        search_btn = Button(search_frame, text="Search", width=20, font=("times new roman", 10, "bold"),
bg="red",
                        fg="white")
        search_btn.grid(row=0, column=2, padx=10, pady=0, sticky=W)

        showAll_btn = Button(search_frame, text="Show All", width=20, font=("times new roman", 10,
"bold"), bg="red",
                         fg="white")
        showAll_btn.grid(row=0, column=3, padx=10, pady=0, sticky=W)

        # ===============Table Frame====================
        table_frame = LabelFrame(Right_frame, bd=2, relief=RIDGE,
                                 font=("times new roman", 11, "bold"))
        table_frame.place(x=5, y=160, width=600, height=310)

        scroll_x = ttk.Scrollbar(table_frame, orient=HORIZONTAL)
        scroll_y = ttk.Scrollbar(table_frame, orient=VERTICAL)
        self.student_table = ttk.Treeview(table_frame, column=(
            "id", "dep", "course", "sem", "name", "div", "roll", "gender", "dob", "email", "phone",
"address",
            "teacher",
            "photo", "year"), xscrollcommand=scroll_x.set, yscrollcommand=scroll_y.set)
        scroll_x.pack(side=BOTTOM, fill=X)
        scroll_y.pack(side=RIGHT, fill=Y)
        scroll_x.config(command=self.student_table.xview)
        scroll_y.config(command=self.student_table.yview)

        self.student_table.heading("id", text="StudentID")
        self.student_table.heading("dep", text="Department")
        self.student_table.heading("course", text="Course")
        self.student_table.heading("sem", text="Semester")
        self.student_table.heading("name", text="Name")
        self.student_table.heading("div", text="Division")
        self.student_table.heading("roll", text="Roll No")
        self.student_table.heading("gender", text="Gender")
        self.student_table.heading("dob", text="DOB")
        self.student_table.heading("email", text="Email")
        self.student_table.heading("phone", text="Phone")
        self.student_table.heading("address", text="Address")
        self.student_table.heading("teacher", text="Teacher")
        self.student_table.heading("year", text="Year")
        self.student_table.heading("photo", text="PhotoSampleStatus")
        self.student_table["show"] = "headings"

        # self.student_table.column("dep", width=100)
        # self.student_table.column("course", width=100)
        # self.student_table.column("year", width=100)
        # self.student_table.column("sem", width=100)
        # self.student_table.column("id", width=100)
        # self.student_table.column("name", width=100)
        # self.student_table.column("div", width=100)
        # self.student_table.column("dob", width=100)
        # self.student_table.column("email", width=100)
        # self.student_table.column("phone", width=100)
        # self.student_table.column("address", width=100)
        # self.student_table.column("teacher", width=100)
        # self.student_table.column("gender", width=100)
        # self.student_table.column("photo", width=150)

        self.student_table.pack(fill=BOTH, expand=1)
        self.student_table.bind("<ButtonRelease>", self.get_cursor)
        self.fetch_data()

    # ================function declaration=================

    def add_data(self):
        if self.var_dep.get() == "Select Department" or self.var_std_name.get() == "" or self.var_std_id.get()
== "":
            messagebox.showerror("Error", "All Field are required", parent=self.root)
        else:
            try:
                conn = mysql.connector.connect(host="localhost", username="root", password="vedant@9765",
                                               database="face_recognizer")
                my_cursor = conn.cursor()
                my_cursor.execute("insert into student
values(%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s)", (
                    self.var_std_id.get(),
                    self.var_dep.get(),
                    self.var_course.get(),
                    self.var_semester.get(),
                    self.var_std_name.get(),
                    self.var_div.get(),
                    self.var_roll.get(),
                    self.var_gender.get(),
                    self.var_dob.get(),
                    self.var_email.get(),
                    self.var_phone.get(),
                    self.var_address.get(),
                    self.var_teacher.get(),
                    self.var_radio1.get(),
                    self.var_year.get()

                ))
                conn.commit()
                self.fetch_data()
                conn.close()
                messagebox.showinfo("Success", "Student detail has been added", parent=self.root)
            except Exception as es:
                messagebox.showerror("Error", f"Due to:{str(es)}", parent=self.root)

    # ====================Fetch Data======================================
    def fetch_data(self):
        conn = mysql.connector.connect(host="localhost", username="root", password="vedant@9765",
                                database="face_recognizer")
        my_cursor = conn.cursor()
        my_cursor.execute("select * from student")
        data = my_cursor.fetchall()
        if len(data) != 0:
            self.student_table.delete(*self.student_table.get_children())
            for i in data:
                self.student_table.insert("", END, values=i)
            conn.commit()
            conn.close()

    # ================================Get
Cursor=========================
    def get_cursor(self, event=""):
        cursor_focus = self.student_table.focus()
        content = self.student_table.item(cursor_focus)
        data = content["values"]

        self.var_std_id.set(data[0])
        self.var_dep.set(data[1])
        self.var_course.set(data[2])
        self.var_semester.set(data[3])
        self.var_std_name.set(data[4])
        self.var_div.set(data[5])
        self.var_roll.set(data[6])
        self.var_gender.set(data[7])
        self.var_dob.set(data[8])
        self.var_email.set(data[9])
        self.var_phone.set(data[10])
        self.var_address.set(data[11])
        self.var_teacher.set(data[12])
        self.var_radio1.set(data[13])
        self.var_year.set(data[14])

    # ===============Update Function====================
    def update_data(self):

        if self.var_dep.get() == "Select Department" or self.var_std_name.get() == "" or self.var_std_id.get()
== "":
            messagebox.showerror("Error", "All Field are required", parent=self.root)
        else:
            try:
                update = messagebox.askyesno("Update", "Do you want to update this student details",
parent=self.root)
                if update > 0:
                    conn = mysql.connector.connect(host="localhost", username="root",
password="vedant@9765",
                                                   database="face_recognizer")
                    my_cursor = conn.cursor()
                    my_cursor.execute(
                        "update student set
Dep=%s,Course=%s,Semester=%s,Name=%s,Division=%s,Roll=%s,Gender=%s,Dob=%s,Email=%s,P
hone=%s,Address=%s,Teacher=%s,PhotoSample=%s,Year=%s where Student_id=%s",
                        (
                            self.var_dep.get(),
                            self.var_course.get(),
                            self.var_semester.get(),
                            self.var_std_name.get(),
                            self.var_div.get(),
                            self.var_roll.get(),
                            self.var_gender.get(),
                            self.var_dob.get(),
                            self.var_email.get(),
                            self.var_phone.get(),
                            self.var_address.get(),
                            self.var_teacher.get(),
                            self.var_radio1.get(),
                            self.var_year.get(),
                            self.var_std_id.get()
                        ))
                else:
                    if not update:
                        return
                messagebox.showinfo("Success", "Student detail successfully update complete", parent=self.root)
                conn.commit()
                self.fetch_data()
                conn.close()
        except Exception as es:
            messagebox.showerror("Error", f"Due To:{str(es)}", parent=self.root)

    # ======================Delete function===============================
    def delete_data(self):
        if self.var_std_id.get() == "":
            messagebox.showerror("Error", "Student id must be required", parent=self.root)
        else:
            try:
                delete = messagebox.askyesno("Student Delete Page", "Do you want to delete this student",
                                             parent=self.root)
                if delete > 0:
                    conn = mysql.connector.connect(host="localhost", username="root",
password="vedant@9765",
                                                   database="face_recognizer")
                    my_cursor = conn.cursor()
                    sql = "delete from student where Student_id=%s"
                    val = (self.var_std_id.get(),)
                    my_cursor.execute(sql, val)
                else:
                    if not delete:
                        return
                conn.commit()
                self.fetch_data()
                conn.close()
                messagebox.showinfo("Delete", f"Successfully deleted student details", parent=self.root)
            except Exception as es:
                messagebox.showerror("Error", f"Due To:{str(es)}", parent=self.root)

    # ===========================Reset Data========================
    def reset_data(self):
        self.var_dep.set("Select Department")
        self.var_course.set("Select Course")
        self.var_year.set("Select Year")
        self.var_semester.set("Select Semester")
        self.var_std_id.set("")
        self.var_std_name.set("")
        self.var_div.set("Select Division")
        self.var_roll.set("")
        self.var_gender.set("Select Gender")
        self.var_dob.set("")
        self.var_email.set("")
        self.var_phone.set("")
        self.var_address.set("")
        self.var_teacher.set("")
        self.var_radio1.set("")

    # ========================= Generate data set or Take photo
Sample========================

    def generte_dataset(self):
        if self.var_std_id.get() == "":
            messagebox.showerror("Error", "Student id must be required", parent=self.root)
        else:
            try:
                conn = mysql.connector.connect(host="localhost", username="root", password="vedant@9765",
                                               database="face_recognizer")
                my_cursor = conn.cursor()
                my_cursor.execute("select * from student")
                myresult = my_cursor.fetchall()
                id = 0
                for x in myresult:
                    id += 1
                my_cursor.execute(
                    "update student set
Dep=%s,Course=%s,Semester=%s,Name=%s,Division=%s,Roll=%s,Gender=%s,Dob=%s,Email=%s,P
hone=%s,Address=%s,Teacher=%s,PhotoSample=%s,Year=%s where Student_id=%s",
                    (
                        self.var_dep.get(),
                        self.var_course.get(),
                        self.var_semester.get(),
                        self.var_std_name.get(),
                        self.var_div.get(),
                        self.var_roll.get(),
                        self.var_gender.get(),
                        self.var_dob.get(),
                        self.var_email.get(),
                        self.var_phone.get(),
                        self.var_address.get(),
                        self.var_teacher.get(),
                        self.var_radio1.get(),
                        self.var_year.get(),
                        self.var_std_id.get() == id + 1
                    ))
                conn.commit()
                self.fetch_data()
                self.reset_data()
                conn.close()
                # ====== Load predifind data on face frontal from open cv=======
                face_classifier = cv2.CascadeClassifier("haarcascade_frontalface_default.xml")
                def face_cropped(img):
                    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
                    faces = face_classifier.detectMultiScale(gray, 1.3, 5)
                    # scaling factor=1.3
                    # Minimum Neighbor=5
                    for (x, y, w, h) in faces:
                        face_cropped = img[y:y + h, x:x + w]
                        return face_cropped

                # cap = cv2.VideoCapture(0)
                img_id = 0

                # mobile phone
                cap = cv2.VideoCapture(0)
                # change this ip address
                address = "http://100.74.129.206:8080//video"
                cap.open(address)
                while True:
                    ret, my_frame = cap.read()
                    if face_cropped(my_frame) is not None:
                        img_id += 1
                        face = cv2.resize(face_cropped(my_frame), (450, 450))
                        face = cv2.cvtColor(face, cv2.COLOR_BGR2GRAY)
                        file_name_path = "data/user." + str(id) + "." + str(img_id) + ".jpg"
                        cv2.imwrite(file_name_path, face)
                        cv2.putText(face, str(img_id), (50, 50), cv2.FONT_HERSHEY_COMPLEX, 2, (0, 255, 0), 2)
                        cv2.imshow("Cropped Face", face)

                    if cv2.waitKey(1) == 13 or int(img_id) == 100:
                        break
                cap.release()
                cv2.destroyAllWindows()
                messagebox.showinfo("Result", "Generating data set completed...")
            except Exception as es:
                messagebox.showerror("Error", f"Due To:{str(es)}", parent=self.root)


if __name__ == "__main__":
    root = Tk()
    obj = Student(root)
    root.mainloop()
```

8.5 Face Detector Frame:

```
from tkinter import *
from PIL import Image, ImageTk
from datetime import datetime
import mysql.connector
import cv2


class Face_Recgnition:
    def __init__(self, root):
        self.root = root
        self.root.geometry("1280x790+0+0")
        self.root.title("Face Recognition System")

        title_lb1 = Label(self.root, text="FACE RECOGNITION",
                          font=("times new roman", 25, "bold"), bg="green", fg="white")
        title_lb1.place(x=0, y=0, width=1280, height=35)

        # 1 image
        img_top = Image.open(r"college_images\fl.jpg")
        img_top = img_top.resize((640, 750), Image.LANCZOS)
        self.photoimg_top = ImageTk.PhotoImage(img_top)

        f_lbl = Label(self.root, image=self.photoimg_top)
        f_lbl.place(x=0, y=0, width=640, height=750)

        img_bottom = Image.open(r"college_images\fr.jpg")
        img_bottom = img_bottom.resize((640, 750), Image.LANCZOS)
        self.photoimg_bottom = ImageTk.PhotoImage(img_bottom)

        f_lbl = Label(self.root, image=self.photoimg_bottom)
        f_lbl.place(x=640, y=0, width=640, height=750)

        b1 = Button(self.root, text="Face Recognition", cursor="hand2", font=("times new roman", 10,
"bold"),
                    bg="red", fg="white", command=self.face_recog)
        b1.place(x=300, y=450, width=150, height=40)

        exit_button = Button(self.root, text="Exit", cursor="hand2", font=("times new roman", 10, "bold"),
                             bg="red", fg="white", command=self.exit_program)
        exit_button.place(x=920, y=450, width=150, height=40)

    #
==============================Attendance==============================
    def mark_attendance(self, i, r, n, d):

        # now = datetime.now()
        # fd = now.strftime("%d/%m/%Y")
        # with open(fd+".csv", "a", newline="\n") as f:

        with open("ved.csv", "r+", newline="\n") as f:
            myDataList = f.readlines()
            name_list = [entry.split(",")[0] for entry in myDataList]

            if (i not in name_list) and (r not in name_list) and (n not in name_list) and (d not in name_list):
                now = datetime.now()
                d1 = now.strftime("%d/%m/%Y")
                dtString = now.strftime("%H:%M:%S")
                f.writelines(f"\n{i},{r},{n},{d},{dtString},{d1},Present")

    # ===========Face Recognition=======================
    def face_recog(self):
        def draw_boundary(img, classifier, scaleFactor, minNeighbors, color, text, clf):
            gray_image = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
            features = classifier.detectMultiScale(gray_image, scaleFactor, minNeighbors)

            coord = []

            for (x, y, w, h) in features:
                cv2.rectangle(img, (x, y), (x + w, y + h), (0, 255, 0), 3)
                id, predict = clf.predict(gray_image[y:y + h, x:x + w])
                confidence = int((100 * (1 - predict / 300)))

                conn = mysql.connector.connect(host="localhost", username="root", password="vedant@9765",
                                               database="face_recognizer")
                my_cursor = conn.cursor()

                my_cursor.execute("SELECT Name, Roll, Dep FROM student WHERE Student_id=%s", (id,))

                result = my_cursor.fetchone()

                if result is not None:  # Check if result is not None before unpacking
                    name, roll, dep = result
                    cv2.putText(img, f"ID:{id}", (x, y - 75), cv2.FONT_HERSHEY_COMPLEX, 0.8, (255, 255,
255), 3)
                    cv2.putText(img, f"Roll:{roll}", (x, y - 55), cv2.FONT_HERSHEY_COMPLEX, 0.8, (255, 255,
255), 3)
                    cv2.putText(img, f"Name:{name}", (x, y - 30), cv2.FONT_HERSHEY_COMPLEX, 0.8, (255,
255, 255), 3)
```

```
                cv2.putText(img, f"Department:{dep}", (x, y - 5), cv2.FONT_HERSHEY_COMPLEX, 0.8, (255,
255, 255), 3)
                self.mark_attendance(id, roll, name, dep)
            else:
                cv2.rectangle(img, (x, y), (x + w, y + h), (0, 0, 255), 3)
                cv2.putText(img, "Unknown Face", (x, y - 5), cv2.FONT_HERSHEY_COMPLEX, 0.8, (255,
255, 255), 3)

            coord = [x, y, w, y]

        return coord

    def recognize(img, clf, face_cascade):

        cord = draw_boundary(img, face_cascade, 1.1, 10, (255, 255, 255), "Face", clf)
        return img

    face_cascade = cv2.CascadeClassifier("haarcascade_frontalface_default.xml")
    clf = cv2.face.LBPHFaceRecognizer.create()
    clf.read("classifier.xml")

    video_cap = cv2.VideoCapture(0)
    # change this ip address
    address = "http://100.74.129.206:8080//video"
    video_cap.open(address)
    while True:
        ret, img = video_cap.read()
        img = recognize(img, clf, face_cascade)
        cv2.imshow("Welcome To Face Recognition", img)

        # Break the loop on 'Enter' key (key code 13)
        if cv2.waitKey(1) == 13:
            break

    video_cap.release()
    cv2.destroyAllWindows()

  def exit_program(self):
    self.root.destroy()


if __name__ == "__main__":
  root = Tk()
  obj = Face_Recgnition(root)
  root.mainloop()
```

8.6 Attendance Frame:

```
from tkinter import *
from tkinter import ttk
from PIL import Image, ImageTk
from tkinter import messagebox
import mysql.connector
import cv2
import os
import csv
from tkinter import filedialog

mydata = []


class Attendance:
    def __init__(self, root):
        self.root = root
        self.root.geometry("1280x790+0+0")
        self.root.title("Face Recognition System")

        # ==========================varaibles===============
        self.var_atten_id = StringVar()
        self.var_atten_roll = StringVar()
        self.var_atten_name = StringVar()
        self.var_atten_dep = StringVar()
        self.var_atten_time = StringVar()
        self.var_atten_date = StringVar()
        self.var_atten_attendance = StringVar()

        #
        img = Image.open(r"college_images\sa.jpg")
        img = img.resize((1280, 150), Image.LANCZOS)
        self.photoimg = ImageTk.PhotoImage(img)

        f_lbl = Label(self.root, image=self.photoimg)
        f_lbl.place(x=0, y=0, width=1280, height=150)


        title_lb1 = Label(self.root, text="Student Attendance Details",
                          font=("times new roman", 30, "bold"), bg="lightcyan", fg="saddlebrown")
        title_lb1.place(x=0, y=150, width=1280, height=40)

        img3 = Image.open(r"college_images\bg.jpg")
        img3 = img3.resize((1280, 710), Image.LANCZOS)
        self.photoimg3 = ImageTk.PhotoImage(img3)

        bg_img = Label(self.root, image=self.photoimg3)
        bg_img.place(x=0, y=190, width=1280, height=710)

        main_frame = Frame(bg_img, bd=2, bg="white")
        main_frame.place(x=10, y=15, width=1250, height=500)

        # Left Side Frame
        Left_frame = LabelFrame(main_frame, bd=2, relief=RIDGE, text="Studet Information",
                                font=("times new roman", 12, "bold"))
        Left_frame.place(x=10, y=10, width=625, height=500)

        img_left = Image.open(r"college_images\ati.jpg")
        img_left = img_left.resize((600, 150), Image.LANCZOS)
        self.photoimg_left \
            = ImageTk.PhotoImage(img_left)

        f_lbl = Label(Left_frame, image=self.photoimg_left)
        f_lbl.place(x=5, y=0, width=600, height=150)

        left_inside_frame = Frame(Left_frame, bd=2, relief=RIDGE, bg="white")
        left_inside_frame.place(x=5, y=160, width=600, height=300)

        # Lable entry
        # attendance_id
        attendance_label = Label(left_inside_frame, text="AttendanceId:", font=("times new roman", 10,
"bold"),
                                 bg="white")
        attendance_label.grid(row=0, column=0, padx=10, pady=5, sticky=W)

        attendance_entry = ttk.Entry(left_inside_frame, width=20,
                                     font=("times new roman", 10, "bold"), textvariable=self.var_atten_id)
        attendance_entry.grid(row=0, column=1, padx=10, pady=5, sticky=W)

        # Roll No
        roll_label = Label(left_inside_frame, text="Roll No:", font=("times new roman", 10, "bold"),
                           bg="white", )
        roll_label.grid(row=0, column=2, padx=10, pady=5, sticky=W)

        roll_entry = ttk.Entry(left_inside_frame, width=20,
                               font=("times new roman", 10, "bold"), textvariable=self.var_atten_roll)
        roll_entry.grid(row=0, column=3, padx=10, pady=5, sticky=W)

        # Name
```

```
name_label = Label(left_inside_frame, text="Name:", font=("times new roman", 10, "bold"),
                bg="white")
name_label.grid(row=1, column=0, padx=10, pady=5, sticky=W)

name_entry = ttk.Entry(left_inside_frame, width=20,
                   font=("times new roman", 10, "bold"), textvariable=self.var_atten_name)
name_entry.grid(row=1, column=1, padx=10, pady=5, sticky=W)

# Department
dep_label = Label(left_inside_frame, text="Department:",
               font=("times new roman", 10, "bold"), bg="white")
dep_label.grid(row=1, column=2, padx=10,pady=5, sticky=W)

dep_entry = ttk.Entry(left_inside_frame, width=20,
                  font=("times new roman", 10, "bold"), textvariable=self.var_atten_dep)
dep_entry.grid(row=1, column=3, padx=2, pady=10, sticky=W)

# Time
time_label = Label(left_inside_frame, text="Time:", font=("times new roman", 10, "bold"),
                bg="white")
time_label.grid(row=2, column=0, padx=10, pady=5, sticky=W)

time_entry = ttk.Entry(left_inside_frame, width=20,
                   font=("times new roman", 10, "bold"), textvariable=self.var_atten_time)
time_entry.grid(row=2, column=1, padx=10, pady=5, sticky=W)

# Date
date_label = Label(left_inside_frame, text="Date:", font=("times new roman", 10, "bold"),
                bg="white")
date_label.grid(row=2, column=2, padx=10, pady=5, sticky=W)

date_entry = ttk.Entry(left_inside_frame, width=20,
                   font=("times new roman", 10, "bold"), textvariable=self.var_atten_date)
date_entry.grid(row=2, column=3, padx=10, pady=5, sticky=W)

# Attendance Status
atten_label = Label(left_inside_frame, text="Attendance Status:",
                 font=("times new roman", 11, "bold"), bg="white")
atten_label.grid(row=3, column=0, padx=10, sticky=W)

atten_entry = ttk.Entry(left_inside_frame, width=20,
                    font=("times new roman", 10, "bold"), textvariable=self.var_atten_attendance)
atten_entry.grid(row=3, column=1, padx=2, pady=10, sticky=W)

# buttons frame
```

```
        btn_frame = Frame(left_inside_frame, bd=2, relief=RIDGE)
        btn_frame.place(x=0, y=170, width=595, height=30)

        # save
        import_btn = Button(btn_frame, text="Import csv", width=20, font=("times new roman", 10, "bold"),
bg="blue",
                        fg="white", command=self.importCsv
                        )
        import_btn.grid(row=0, column=0)

        # update
        export_btn = Button(btn_frame, text="Export csv", width=20, font=("times new roman", 10, "bold"),
bg="blue",
                        fg="white", command=self.exportCsv)
        export_btn.grid(row=0, column=1)

        # delete
        update_btn = Button(btn_frame, text="Update csv", width=20, font=("times new roman", 10, "bold"),
bg="blue",
                        fg="white")
        update_btn.grid(row=0, column=2)

        # reset
        reset_btn = Button(btn_frame, text="Reset", width=20, font=("times new roman", 10, "bold"),
bg="blue",
                       fg="white", command=self.reset_data)
        reset_btn.grid(row=0, column=3)

        # Right Side Frame
        Right_frame = LabelFrame(main_frame, bd=2, relief=RIDGE, text="Student Details",
                              font=("times new roman", 12, "bold"))
        Right_frame.place(x=630, y=10, width=620, height=500)

        table_frame = Frame(Right_frame, bd=2, relief=RIDGE)
        table_frame.place(x=5, y=5, width=595, height=400)

        # ===================Scroll table=====================
        scroll_x = ttk.Scrollbar(table_frame, orient=HORIZONTAL)
        scroll_y = ttk.Scrollbar(table_frame, orient=VERTICAL)

        self.AttendanceReportTable = ttk.Treeview(table_frame, columns=(
        "id", "roll", "name", "department", "time", "date", "attendance"), xscrollcommand=scroll_x.set,
                                          yscrollcommand=scroll_y.set)

        scroll_x.pack(side=BOTTOM, fill=X)
```

```
    scroll_y.pack(side=RIGHT, fill=Y)

    scroll_x.config(command=self.AttendanceReportTable.xview)
    scroll_y.config(command=self.AttendanceReportTable.yview)

    self.AttendanceReportTable.heading("id", text="Attendance ID")
    self.AttendanceReportTable.heading("roll", text="Roll")
    self.AttendanceReportTable.heading("name", text="Name")
    self.AttendanceReportTable.heading("department", text="Department")
    self.AttendanceReportTable.heading("time", text="Time")
    self.AttendanceReportTable.heading("date", text="Date")
    self.AttendanceReportTable.heading("attendance", text="Attendance")

    self.AttendanceReportTable["show"] = "headings"

    self.AttendanceReportTable.pack(fill=BOTH, expand=1)

# ====================fetch data=====================

def fetchData(self, rows):
    self.AttendanceReportTable.delete(*self.AttendanceReportTable.get_children())
    for i in rows:
        self.AttendanceReportTable.insert("", END, values=i)

        self.AttendanceReportTable.bind("<ButtonRelease>", self.get_cursor)

# import CSV
def importCsv(self):
    global mydata
    mydata.clear()
    fln = filedialog.askopenfilename(initialdir=os.getcwd(), title="Open csv",
                                     filetypes=(("CSV File", "*.csv"), ("All File", "*.*")), parent=self.root)
    with open(fln) as myfile:
        csvread = csv.reader(myfile, delimiter=",")
        for i in csvread:
            mydata.append(i)
        self.fetchData(mydata)

# export CSV

def exportCsv(self):
    try:
        if len(mydata) < 1:
            messagebox.showerror("No Data", "NO DATA FOUND TO EXPORT", parent=self.root)
            return False
            fln = filedialog.asksaveasfilename(initialdir=os.getcwd(), title="Open csv",
                                               filetypes=(("Open CSV", "*.csv"), ("All File", "*.*")), parent=self.root)
            with open(fln, mode="w", newline="") as myfile:
                exp_write = csv.writer(myfile, delimiter=",")
                for i in mydata:
                    exp_write.writerow(i)
                messagebox.showinfo("Data Export", "Your data exported to" + os.path.basename(fln) +
"Successfully")
        except Exception as es:
            messagebox.showerror("Error", f"Due To:{str(es)}", parent=self.root)

    def get_cursor(self, event=""):
        cursor_row = self.AttendanceReportTable.focus()
        content = self.AttendanceReportTable.item(cursor_row)
        rows = content['values']
        self.var_atten_id.set(rows[0])
        self.var_atten_roll.set(rows[1])
        self.var_atten_name.set(rows[2])
        self.var_atten_dep.set(rows[3])
        self.var_atten_time.set(rows[4])
        self.var_atten_date.set(rows[5])
        self.var_atten_attendance.set(rows[6])

    def reset_data(self):
        self.var_atten_id.set("")
        self.var_atten_roll.set("")
        self.var_atten_name.set("")
        self.var_atten_dep.set("")
        self.var_atten_time.set("")
        self.var_atten_date.set("")
        self.var_atten_attendance.set("")


if __name__ == "__main__":
    root = Tk()
    obj = Attendance(root)
    root.mainloop()
```

8.7 Help Desk Frame:

```
from tkinter import *
from PIL import Image, ImageTk


class Help:
    def __init__(self, root):
        self.root = root
        self.root.geometry("1280x790+0+0")
        self.root.title("face Recognition System")

        title_lb1 = Label(self.root, text="Help",
                          font=("times new roman", 25, "bold"), bg="Black", fg="red")
        title_lb1.place(x=0, y=0, width=1280, height=35)

        img_top = Image.open(r"college_images\help (2).png")
        img_top = img_top.resize((1280, 625), Image.LANCZOS)
        self.photoimg_top \
            = ImageTk.PhotoImage(img_top)

        f_lbl = Label(self.root, image=self.photoimg_top)
        f_lbl.place(x=0, y=35, width=1280, height=625)


if __name__ == "__main__":
    root = Tk()
    obj = Help(root)
    root.mainloop()
```

8.8 Train Data Frame:

```
from tkinter import *
from tkinter import ttk
from PIL import Image, ImageTk
from tkinter import messagebox
import numpy as np
import mysql.connector
import cv2
import os


class Train:
    def __init__(self, root):
        self.root = root
        self.root.geometry("1280x790+0+0")
        self.root.title("face Recognition System")

        title_lb1 = Label(self.root, text="TRAIN DATA SET",
                          font=("times new roman", 30, "bold"), bg="coral", fg="mediumpurple")
        title_lb1.place(x=0, y=0, width=1280, height=60)
        img_top = Image.open(r"college_images\tu.jpg")
        img_top = img_top.resize((1280, 310), Image.LANCZOS)
        self.photoimg_top \
            = ImageTk.PhotoImage(img_top)

        f_lbl = Label(self.root, image=self.photoimg_top)
        f_lbl.place(x=0, y=60, width=1280, height=310)

        b1 = Button(self.root, text="TRAIN DATA SET", cursor="hand2", font=("times new roman", 20,
"bold"),
                    bg="yellowgreen",
                    fg="midnightblue", command=self.train_classifier)
        b1.place(x=0, y=320, width=1280, height=50)

        img_bottom = Image.open(r"college_images\td.jpg")
        img_bottom = img_bottom.resize((1280, 290), Image.LANCZOS)
        self.photoimg_bottom \
            = ImageTk.PhotoImage(img_bottom)

        f_lbl = Label(self.root, image=self.photoimg_bottom)
        f_lbl.place(x=0, y=370, width=1280, height=290)

    def train_classifier(self):
        data_dir = ("data")
        path = [os.path.join(data_dir, file) for file in os.listdir(data_dir)]

        faces = []
        ids = []

        for image in path:
            img = Image.open(image).convert('L')  # Gray scale image
            imageNp = np.array(img, "uint8")
            id = int(os.path.split(image)[1].split(".")[1])

            faces.append((imageNp))
            ids.append(id)
            cv2.imshow("Training", imageNp)
            if  cv2.waitKey(1) == 13:
                break

        ids = np.array(ids)

        # ===================Train the classifier and save=====================
        clf = cv2.face.LBPHFaceRecognizer.create()
        clf.train(faces, ids)
        clf.write("classifier.xml")
        cv2.destroyAllWindows()
        messagebox.showinfo("Result", "Training datasets completed...")


if __name__ == "__main__":
    root = Tk()
    obj = Train(root)
    root.mainloop()
```

8.9 Developer Frame:

```
from tkinter import *
from PIL import Image, ImageTk
import webbrowser


class Developer:
    def __init__(self, root):
        self.root = root
        self.root.geometry("1280x790+0+0")
        self.root.title("face Recognition System")

        title_lb1 = Label(self.root, text="Developer",
                          font=("times new roman", 25, "bold"), bg="Black", fg="white")
        title_lb1.place(x=0, y=0, width=1280, height=35)

        img_bg = Image.open(r"college_images\About.png")
        img_bg = img_bg.resize((1280, 625), Image.LANCZOS)
        self.photoimg = ImageTk.PhotoImage(img_bg)
        bg_img = Label(self.root, image=self.photoimg)
        bg_img.place(x=0, y=35, width=1280, height=625)

        # Twitter

        imgtwitter = Image.open(r"college_images\twitter.png")
        imgtwitter = imgtwitter.resize((50, 45), Image.LANCZOS)
        self.photoimgtwitter = ImageTk.PhotoImage(imgtwitter)

        b1 = Button(bg_img, image=self.photoimgtwitter, cursor="hand2", command=self.twitter)
        b1.place(x=1030, y=23, width=50, height=45)

        # Github

        imggithub = Image.open(r"college_images\github.jpg")
        imggithub = imggithub.resize((50, 45), Image.LANCZOS)
        self.photoimggithub = ImageTk.PhotoImage(imggithub)

        b1 = Button(bg_img, image=self.photoimggithub, cursor="hand2", command=self.github)
        b1.place(x=1103, y=23, width=50, height=45)

        # Linkedin

        imglinkedin = Image.open(r"college_images\linkedin.png")
        imglinkedin = imglinkedin.resize((50, 45), Image.LANCZOS)
        self.photoimglinkedin = ImageTk.PhotoImage(imglinkedin)

        b1 = Button(bg_img, image=self.photoimglinkedin, cursor="hand2", command=self.linkedin)
        b1.place(x=1168, y=23, width=50, height=45)

        # ==================Buttons==================

    def twitter(self):
        url = "https://x.com/DavhareVedant?t=opx8gQ8K_4YdtXYvN2H_5Q&s=08"
        webbrowser.open(url)

    def github(self):
        url = "https://github.com/Vedant-Davhare"
        webbrowser.open(url)

    def linkedin(self):
        url = "https://www.linkedin.com/in/vedant-davhare-
b61b64227?utm_source=share&utm_campaign=share_via&utm_content=profile&utm_medium=androi
d_app"
        webbrowser.open(url)


if __name__ == "__main__":
    root = Tk()
    obj = Developer(root)
    root.mainloop()
```

Databases(SQL):

To create database:

```
CREATE DATABASE face_recognizer;
```

To create table, register:

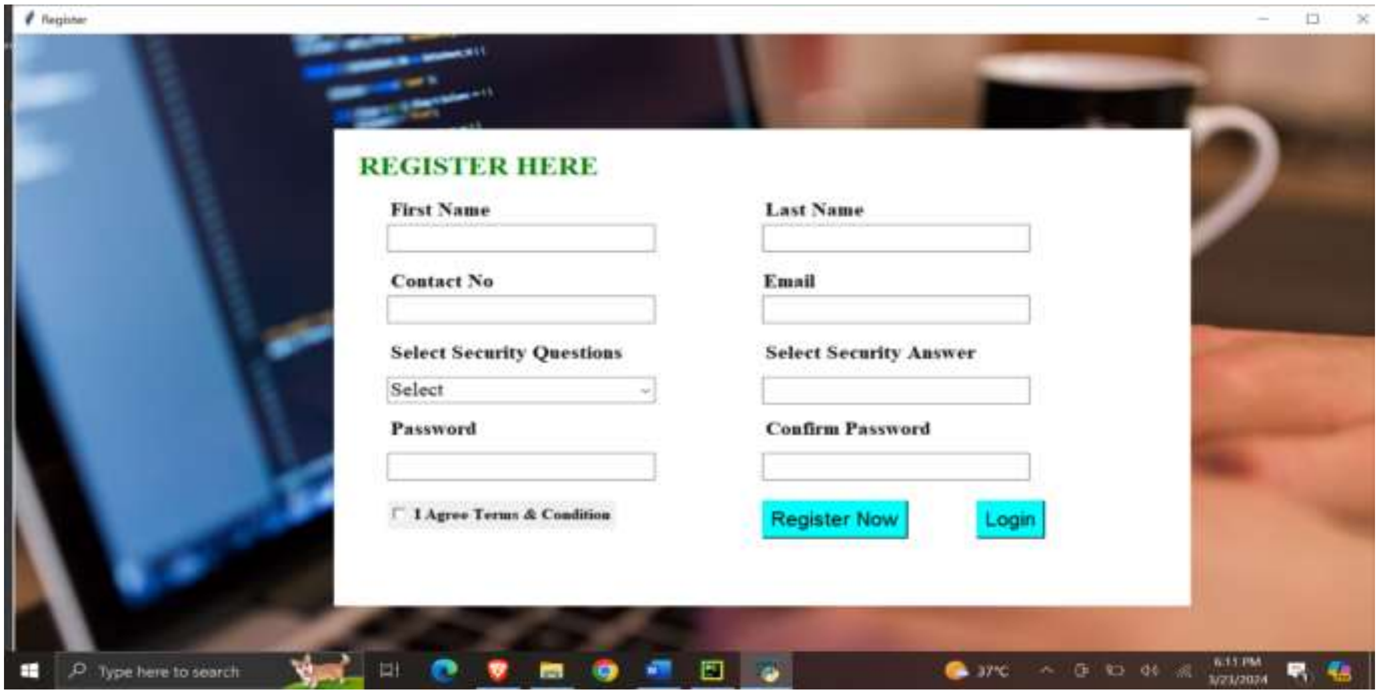
```
CREATE TABLE `register` (
  `faname` varchar(45) DEFAULT NULL,
  `iname` varchar(45) DEFAULT NULL,
  `contact` varchar(45) DEFAULT NULL,
  `email` varchar(45) NOT NULL,
  `securityQ` varchar(45) DEFAULT NULL,
  `securityA` varchar(45) DEFAULT NULL,
  `password` varchar(45) DEFAULT NULL,
  PRIMARY KEY (`email`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

To create table, student:

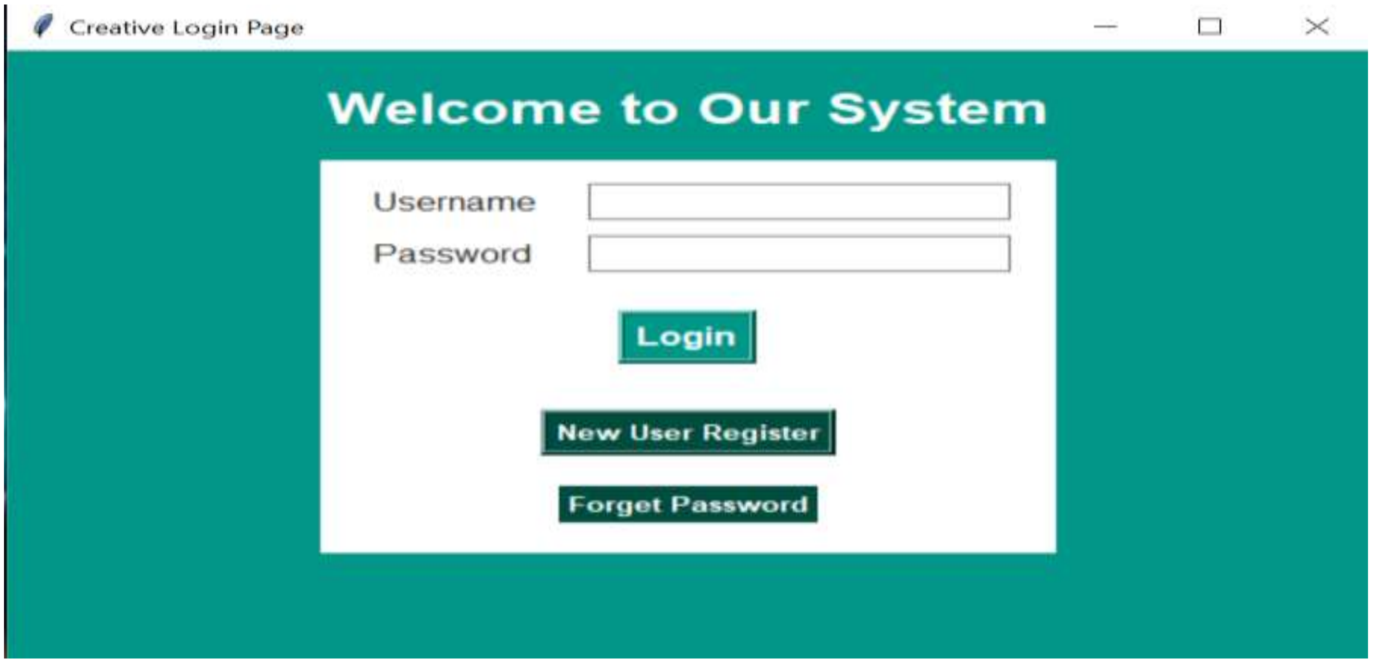
```
CREATE TABLE `student` (
  `Student_id` int NOT NULL,
  `Dep` varchar(45) DEFAULT NULL,
  `Course` varchar(45) DEFAULT NULL,
  `Semester` varchar(45) DEFAULT NULL,
  `Name` varchar(45) DEFAULT NULL,
  `Division` varchar(45) DEFAULT NULL,
  `Roll` varchar(45) DEFAULT NULL,
  `Gender` varchar(45) DEFAULT NULL,
  `Dob` varchar(45) DEFAULT NULL,
  `Email` varchar(45) DEFAULT NULL,
  `Phone` varchar(45) DEFAULT NULL,
  `Address` varchar(45) DEFAULT NULL,
  `Teacher` varchar(45) DEFAULT NULL,
  `PhotoSample` varchar(45) DEFAULT NULL,
  `Year` varchar(45) DEFAULT NULL,
  PRIMARY KEY (`Student_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

9. OUTPUT SCREES:

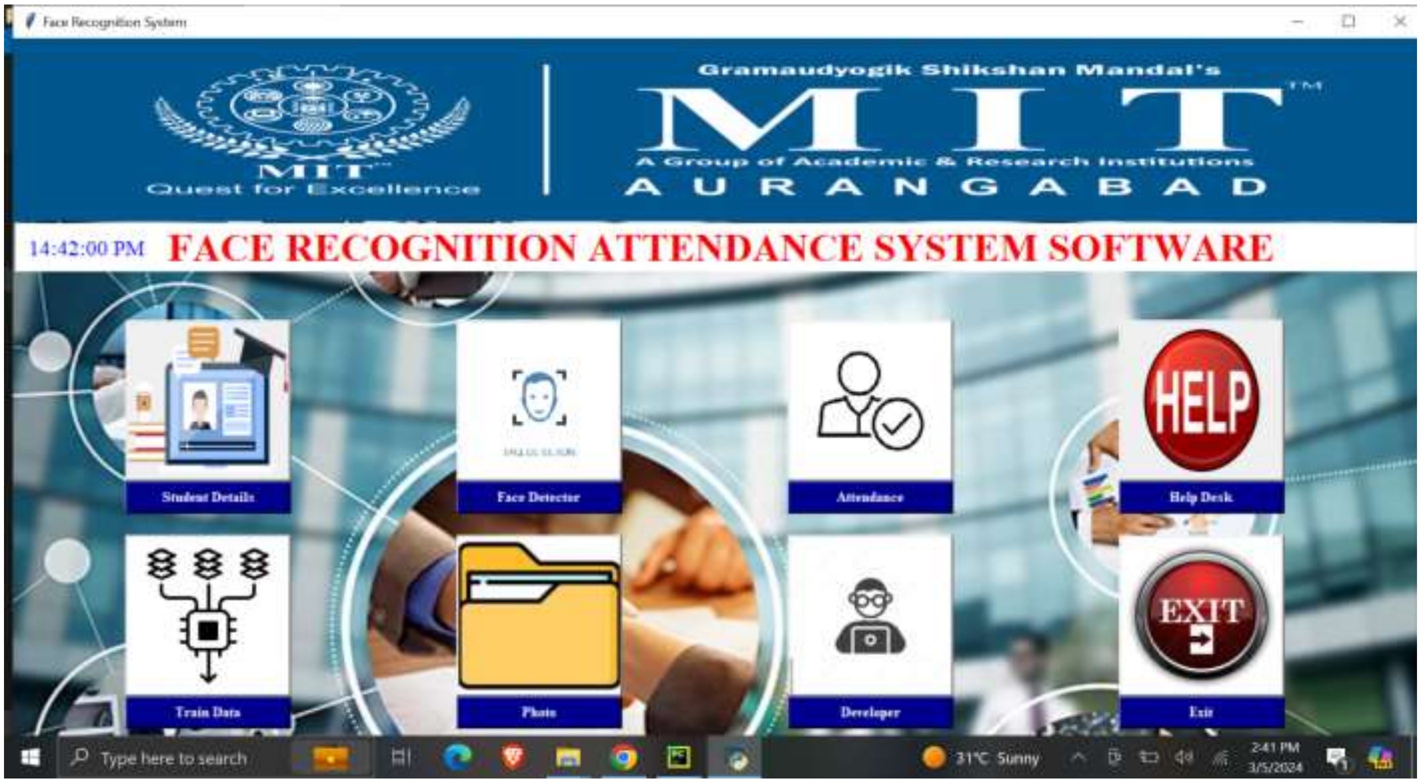
1. Registration:



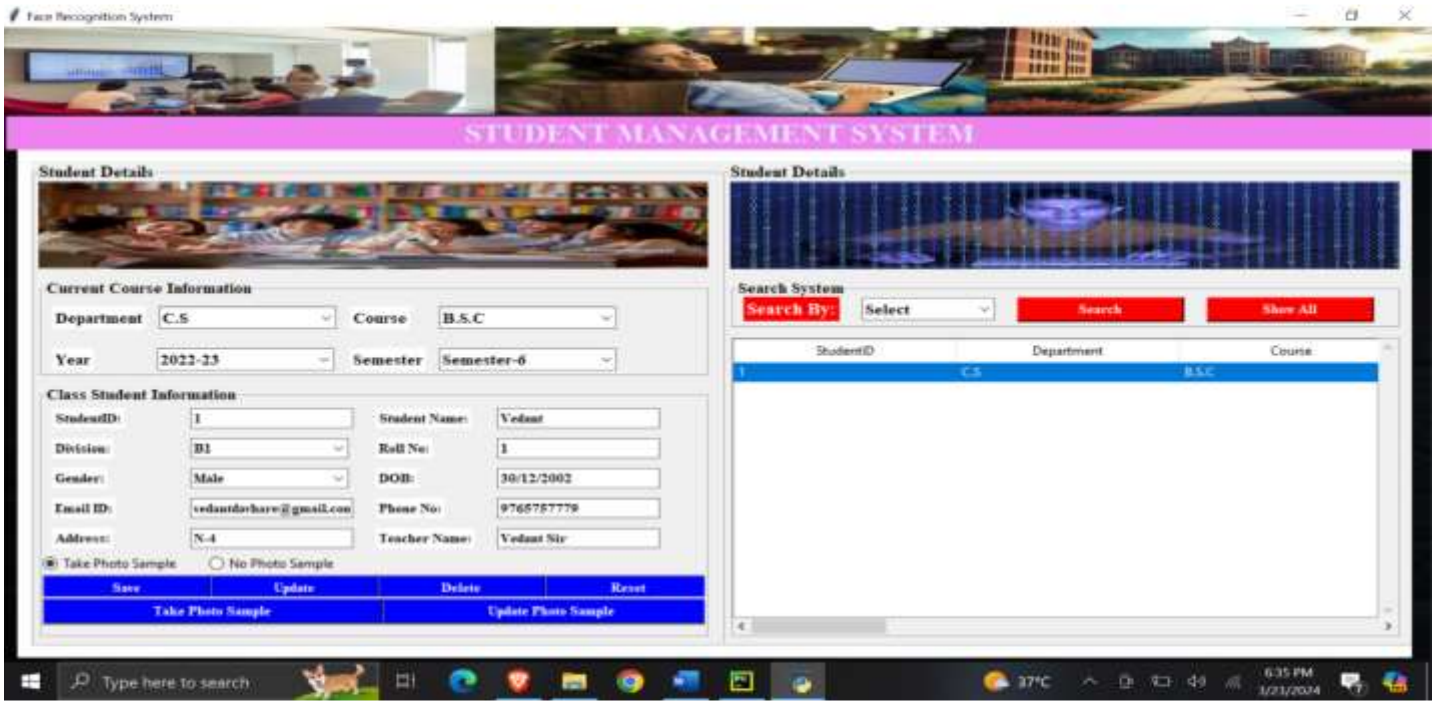
2. Login:



3. Main:



4. Student Details:



Current Course Information

Department: C.S Course: B.S.C
Year: 2022-23 Semester: Semester-6

Class Student Information

StudentID:	1	Student Name:	Vedant
Division:	B1	Roll No:	1
Gender:	Male	DOB:	30/12/2002
Email ID:	vedantdarhare@gmail.com	Phone No:	9765757779
Address:	N-4	Teacher Name:	Vedant Sir

[✓ Take Photo Sample] No Photo Sample

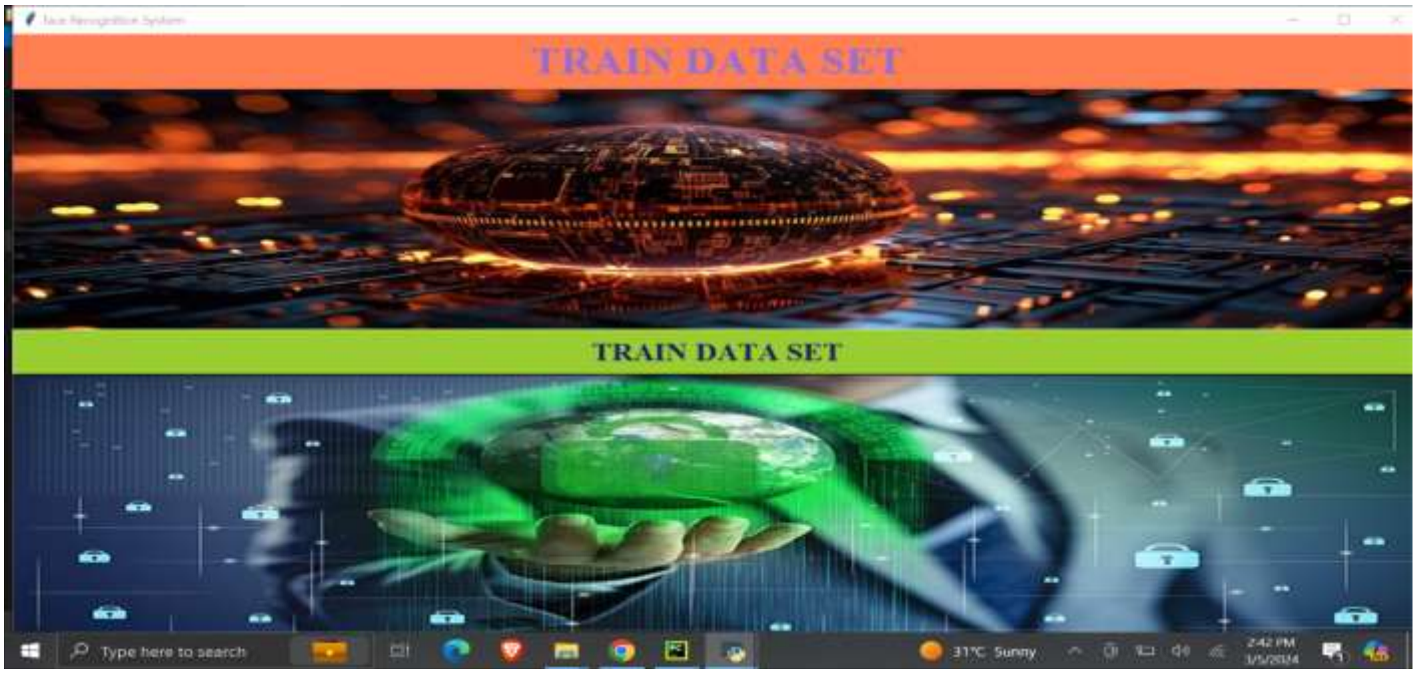
Save | Update | Delete | Reset
Take Photo Sample | Update Photo Sample

Search System

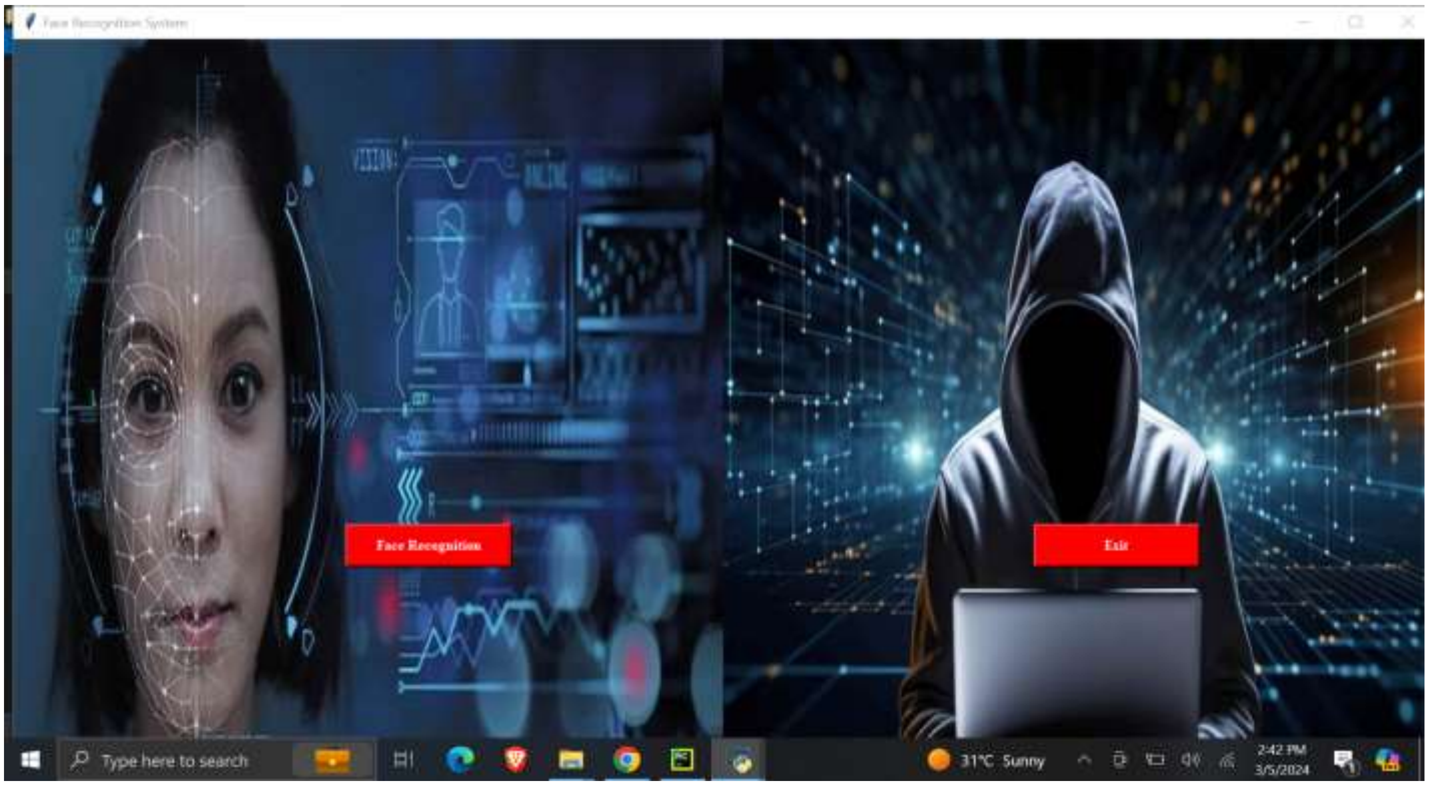
Search By: Select | Search | Show All

StudentID	Department	Course
1	C.S	B.S.C

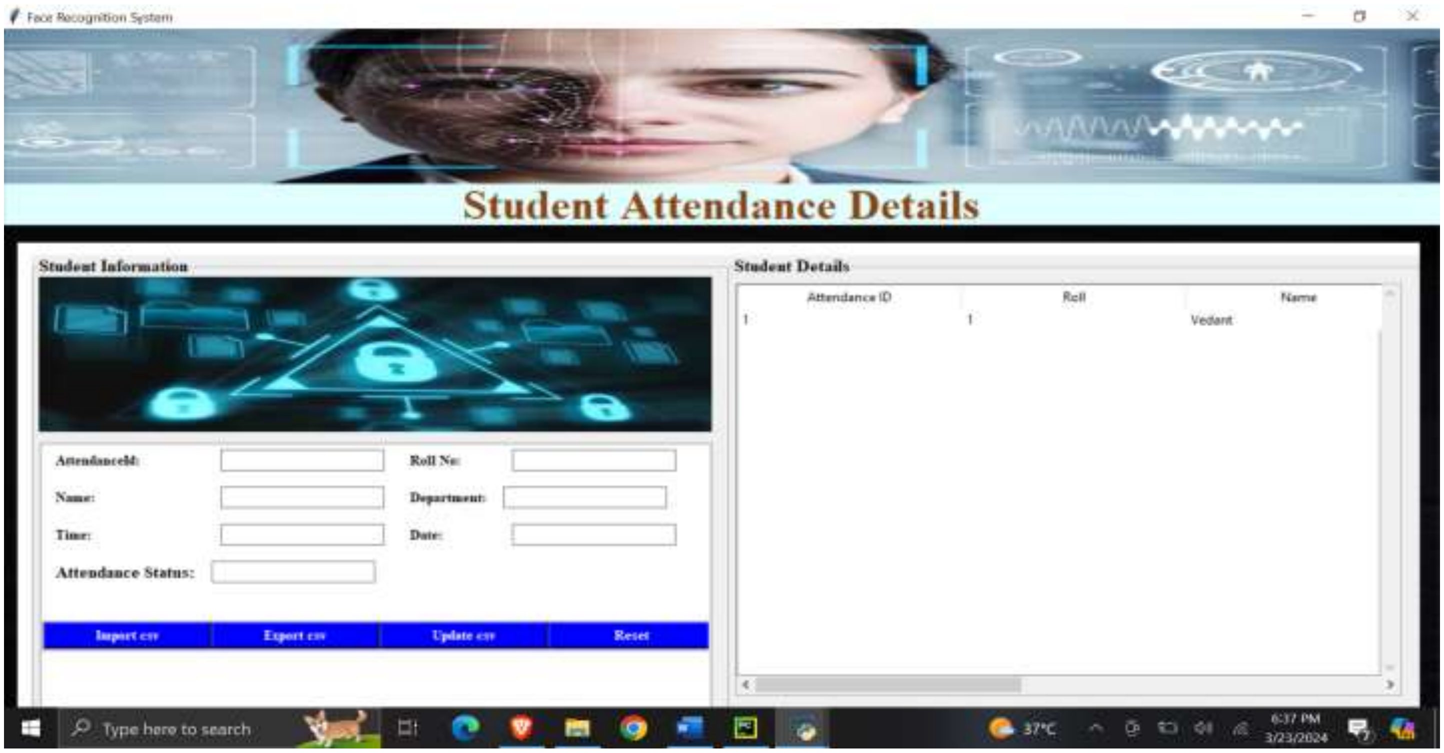
5. Train Data:



6. Face Recognition:



7. Attendance:


Face Recognition System

Student Attendance Details

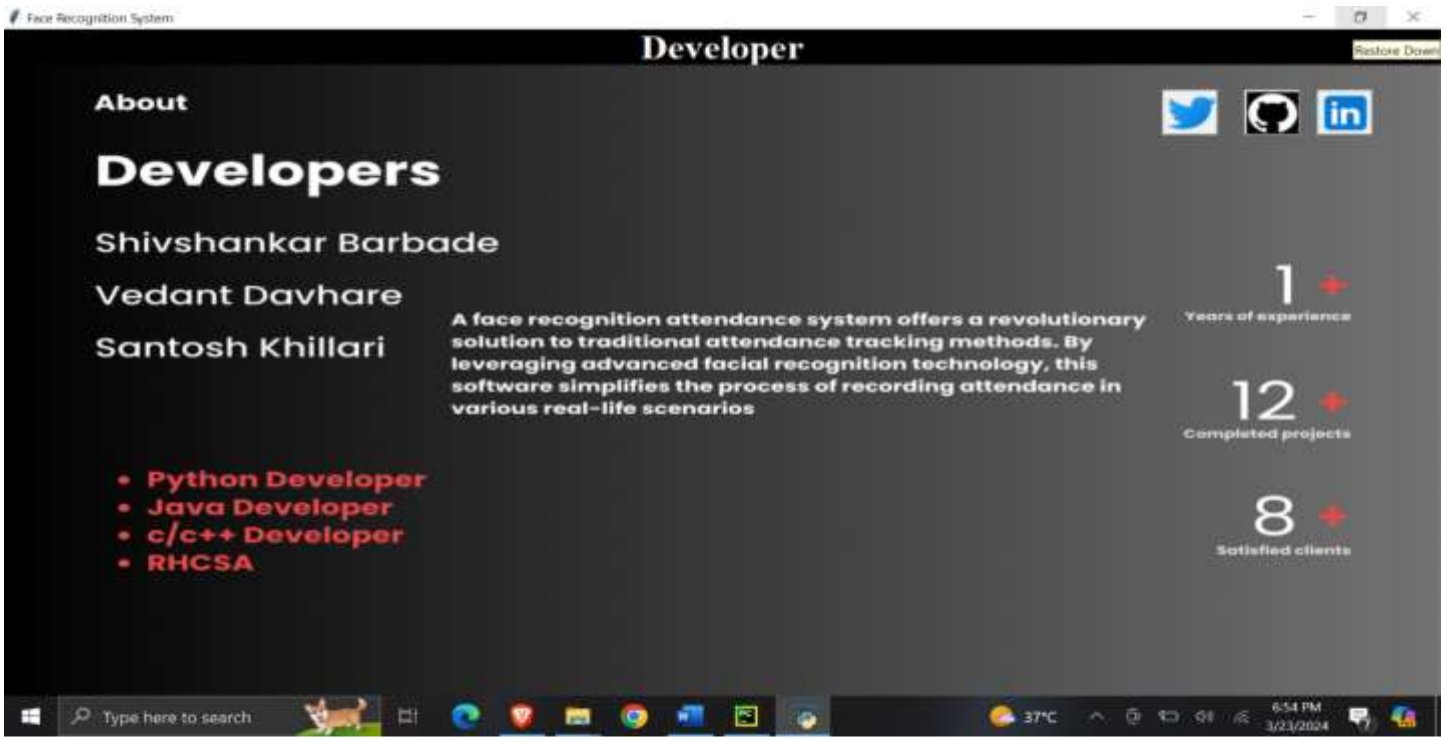
Student Information

AttendanceId:		Roll No:	
Name:		Department:	
Time:		Date:	
Attendance Status:			

Student Details

Attendance ID	Roll	Name
1	1	Vedant

8.Developer:


Face Recognition System

Developer

About

Developers

Shivshankar Barbade

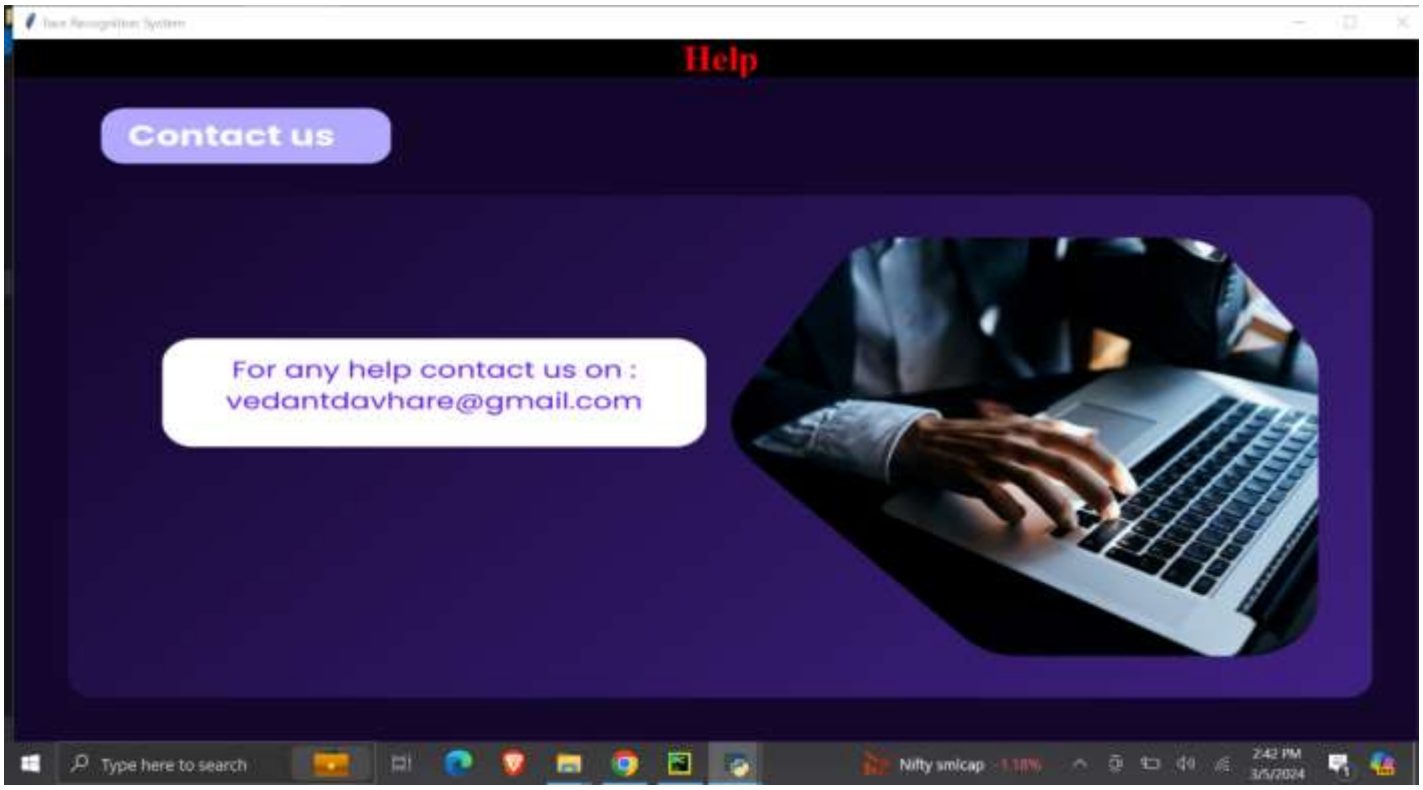
Vedant Davhare

Santosh Khillari

A face recognition attendance system offers a revolutionary solution to traditional attendance tracking methods. By leveraging advanced facial recognition technology, this software simplifies the process of recording attendance in various real-life scenarios

- Python Developer
- Java Developer
- c/c++ Developer
- RHCSA

9. Help:



For any help contact us on :
vedantdavhare@gmail.com

10. SYSTEM TESTING AND IMPLEMENTATION:

10.1 Introduction:

Software testing is a critical element of software quality assurance and represents the ultimate review of specification, design, and coding. In fact, is the one step in the software engineering process that could be viewed as destructive rather than constructive. A strategy for software testing integrates software test case design methods into a well-planned series of steps that results in the successive construction of software. Testing is the set of activities that can be planned and conducted systematically.

10.2 Strategic approach to software testing:

The software engineering process can be viewed as a spiral. Initially system engineering defines the role of software and leads to software requirement analysis where the information domain, functions, behaviour, performance, constrains and validation criteria for software are established. Moving invert among spiral, we come to design and finally to code. To develop computer software, we spiral in along streamlines that decrease the level of abstraction on each turn.

A strategy for software testing may also be viewed in the context of the spiral. Unit testing begins at the vertex of the spiral and concentrates on each unit of the software as implemented in source code. Testing process by moving outward among the spiral to integration testing, where the focus is on the design and the construction of the software architecture. Taking another turn on outward on the spiral we encounter validation testing where requirements established as part of software requirements analysis are validated against the software that has been constructed. Finally, we arrive at system testing, where the software and other system elements are tested as a whole.

10.3Unit Testing:

Unit testing focuses verification effort on the smallest unit of software design, the module. The unit testing, we have white box oriented and some modules the steps are conducted in parallel.

White-box Testing:

This type of testing ensures that:

- All independence paths have been exercised at least once.
- logical decisions have been exercised on their true and false style.
- All loops are executed at their boundaries and within their operational bounds.
- All internal data structures have been exercised to assure their validity.

To follow the concept of white-box testing we have tested each form. We
have created independently verify that Dataflow is correct, all conditions are exercised to checked their validity, all loops are executed on their boundaries.

Black-box Testing:

This type of testing ensures that:

- All possible user interactions and scenarios have been exercised at least once.
- Functional decisions and logical flows have been tested under various input conditions to verify their accuracy.
- The software's behaviour is validated across different usage patterns, ensuring that it performs correctly for all intended user actions.
- Boundary cases and edge conditions are thoroughly tested to ensure that the system behaves as expected under extreme circumstances.
- Input data variations are examined to confirm the validity and robustness of the software's processing.
- In adherence to the principles of black box testing, each feature or function has been independently verified to ensure:
- Comprehensive coverage of external behaviours without knowledge of internal code structures.
- Test cases designed to validate the functionality based on user requirements and specifications.

- The software's response to inputs and stimuli from end-users is observed and compared against expected outcomes.

- Testing is performed from a user's perspective, focusing on usability, correctness, and adherence to specifications rather than internal implementation details.

By following the concept of black box testing, we have precise examined the software's behaviour without relying on knowledge of its internal workings. This approach enables thorough validation of functionality while maintaining independence from implementation specifics, ensuring the software meets user expectations and performs reliably in diverse operating environments.

Conditional Testing:

In this part of the testing each of the conditions were tested to both true and false aspects. And all the resulting paths were tested. So that each path may be generate on particular condition is traced to uncover any possible errors.

Data Flow Testing:

This type of testing selects the path of the program according to the location of definition and use of variables. This kind of testing was used only when some local variables were declared. The **definition-use chain** method was used in this type of testing. These were particularly use full in nested statements.

Loop Testing:

In this type of testing all the loops are tested to all the limits possible. The following exercise was adopted for all loops:

- All the loops were tested at their limits, just above them an adjust below t hem.
- All the loops were skipped at least ones.
- For nested loops test the inner most loop first and then work outwards.

- For concatenated loops the value of dependent loops was set with the help of connected loop.
- Unstructured loops were resolved into nested loops or concatenated loops and tested as above.
- Each and it has been separately tested by the development team itself and all the input have validated.

10.4 Test Cases:

Test case 1-Login

Test 1:

i. Incorrect input: An empty requirement field. (username and password)

ii. Pass criteria: An appropriate error message should be display and the user shouldn’t be allowed to login.

iii. Correct input: Right username and password.

iv. Pass criteria: The user should be directed to the next form which the user is requested.

Test 2:

i. Incorrect input: Wrong username and /or wrong password.

ii. Pass criteria: the user shouldn’t be allowed to login to the system and an appropriate error message should be displayed.

iii. Correct input: Right username and password.

iv. Pass criteria: The user should belong in to the next form.

11. CONCLUSION:

The development and implementation of the Face Recognition Attendance System represent a significant milestone in modernizing attendance tracking processes. By leveraging advanced facial recognition technology, the system offers a reliable, accurate, and efficient solution for organizations seeking to streamline their attendance management operations. Throughout the project, we have successfully achieved our objectives of automating attendance tracking, improving accuracy, and enhancing user experience.

The system's architecture, incorporating front-end interfaces, back-end servers, and databases, provides a robust foundation for seamless operation and scalability. The integration of facial recognition algorithms, user management functionalities, and reporting capabilities ensures comprehensive attendance tracking and management. Moreover, security measures such as encryption, authentication, and access control safeguard sensitive attendance data, ensuring compliance with privacy regulations.

The implementation of the Face Recognition Attendance System has delivered tangible benefits to organizations, including time savings, reduced administrative burden, and improved accuracy in attendance tracking. Users have praised the system for its ease of use, real-time monitoring capabilities, and reliability.

Looking ahead, the project lays the groundwork for future enhancements and expansions. Potential areas for improvement include refining facial recognition algorithms for higher accuracy, integrating additional features such as biometric authentication, and exploring opportunities for cloud-based deployment to enhance scalability and accessibility.

In conclusion, the Face Recognition Attendance System represents a significant step forward in modernizing attendance tracking processes, offering organizations a reliable and efficient solution for managing attendance data. By leveraging advanced technology and innovative design, the system has the potential to revolutionize attendance management practices and set new standards for accuracy, efficiency, and convenience.

12. BIBLIOGRAPHY:

- Python Documentation
- MySQL Documentation
- https://youtube.com/playlist?list=PLMnmAHlVrrJJ5bzj07Fr-xGOqEMRHzaGX&si=YuSipY6aSUnhw3bb
- https://www.analyticsvidhya.com/blog/2021/07/understanding-face-recognition-using-lbph-algorithm/
- https://stackoverflow.com/questions/65627629/attendance-system-based-on-facial-recognition/