ABSTRACT

- The adoption of face recognition technology for attendance management systems represents a significant advancement in the realm of educational and organizational efficiency. The integration of Python, OpenCV, and Firebase in the development of an automated attendance system demonstrates a sophisticated approach to streamline attendance tracking processes.

- This project aims to utilize face recognition algorithms implemented in Python with the OpenCV library to accurately identify individuals and mark their attendance. Leveraging Firebase as the backend database facilitates real-time data management, enabling seamless updates of attendance records.

- Through the utilization of a webcam, the system captures live video feeds and applies face detection and recognition techniques to identify registered individuals within a given database. Upon successful recognition, attendance records are updated instantaneously in the Firebase database, providing administrators with up-to-date attendance statistics.

- The project also incorporates a graphical user interface (GUI) to enhance user interaction and display relevant information during the attendance marking process. By integrating Firebase storage, student images are efficiently managed and accessed for recognition purposes, further enhancing the system's accuracy and reliability.

- This project offers a comprehensive solution for attendance management, catering to the needs of educational institutions, corporate organizations, and other entities requiring efficient attendance tracking systems. By automating the attendance process through face recognition technology, the project contributes to increased accuracy, efficiency, and convenience in managing attendance records, ultimately facilitating a more streamlined and effective organizational workflow.

CHAPTER-1
INTRODUCTION

1.INTRODUCTION

In the realm of modern education and organizational management, the traditional methods of attendance tracking have become increasingly obsolete, often prone to errors and inefficiencies. Manual attendance taking, whether through paper-based registers or electronic systems, can be time-consuming, susceptible to manipulation, and cumbersome to manage, particularly in large-scale settings.

To address these challenges and usher in a new era of efficiency and accuracy in attendance management, the integration of cutting-edge technologies such as Python, OpenCV, and Firebase presents a compelling solution. By harnessing the power of face recognition technology, coupled with real-time data management capabilities offered by Firebase, this project endeavors to revolutionize the way attendance is monitored and recorded.

The project's overarching goal is to develop an automated attendance system that leverages facial recognition algorithms to identify individuals and mark their attendance seamlessly. Through the utilization of a webcam or other camera devices, the system captures live video feeds, detects faces, and compares them against a database of known individuals. Upon successful recognition, attendance records are updated in real-time, providing administrators with accurate and up-to-date attendance statistics.

This project represents a significant step towards modernizing attendance management systems in educational institutions, corporate organizations, and other entities requiring efficient attendance tracking mechanisms. By automating the attendance process through advanced face recognition technology, the project aims to streamline administrative workflows, improve data accuracy, and enhance overall efficiency.

1.1 AN OVERVIEW

The Automated Attendance System using Python, OpenCV, and Firebase is an innovative project designed to revolutionize the process of attendance tracking in educational institutions and organizational settings. This system leverages advanced technologies to automate attendance marking, thereby streamlining administrative tasks and improving efficiency.

At its core, the system consists of a sophisticated face recognition algorithm implemented in Python using the OpenCV library. By utilizing a webcam or similar camera device, the system captures live video feeds and employs facial recognition techniques to identify registered individuals within a designated database. This process allows for seamless and accurate attendance tracking without the need for manual intervention.

In addition to facial recognition, the system integrates Firebase, a real-time database platform, to manage attendance records and provide instant updates. Firebase enables the system to store and retrieve attendance data efficiently, ensuring that attendance records are always up-to-date and accessible to administrators. One of the key features of the system is its ability to adapt to various organizational needs and environments. Whether deployed in educational institutions, corporate offices, or other settings, the Automated Attendance System offers a flexible and customizable solution for attendance tracking.

Overall, the Automated Attendance System using Python, OpenCV, and Firebase represents a significant advancement in attendance management technology. By automating the attendance tracking process and leveraging cutting-edge technologies, the system simplifies administrative tasks, improves accuracy, and enhances overall organizational efficiency.

1.2 OBJECTIVES

The objective of the Auto Attendance System project is to create an automated attendance tracking solution leveraging AI and machine learning technologies.

CHAPTER-2
SYSTEM STUDY

2.1 EXISTING SYTEM

The current attendance tracking system relies on manual methods, such as paper-based registers or electronic spreadsheets, for recording attendance. This manual process is time-consuming and prone to errors, leading to inaccuracies in attendance records. Additionally, it requires significant administrative effort and may result in inefficiencies in attendance management. Furthermore, in the absence of an automated system, there is a lack of real-time visibility into attendance data, making it challenging for administrators to track attendance trends and identify patterns. This manual approach also limits the ability to generate timely reports and analyze attendance data effectively.

Overall, the existing manual attendance system is inefficient and lacks the capabilities required for effective attendance management in modern educational and organizational settings. There is a clear need for an automated attendance system that streamlines the process, improves accuracy, and provides real-time access to attendance data for administrators.

2.1.1 DISADVANTAGES

- Manual process prone to errors.
- Time-consuming for administrators.
- Lack of real-time visibility into attendance data.
- Inefficiencies in attendance management.
- Limited ability to generate timely reports.
- Difficulty in tracking attendance trends and patterns.
- Inability to analyze attendance data effectively.

2.2 PROPOSED SYSTEM

The proposed system aims to revolutionize traditional attendance tracking processes by harnessing cutting-edge technologies such as artificial intelligence (AI) and machine learning. At its core, the system will feature an AI-powered face recognition module, leveraging sophisticated algorithms to accurately identify individuals based on captured facial features. This module will continuously refine its recognition accuracy over time, ensuring dependable attendance marking in various environments.

Central to the proposed system is its capability for real-time attendance tracking. As individuals enter designated areas, the system will automatically detect their faces using a webcam and compare them against pre-registered profiles. Upon successful recognition, the system will promptly mark their attendance and update records in the database in real-time. This instantaneous tracking eliminates the need for manual data entry, significantly streamlining administrative processes.

Integration with a cloud-based database platform, such as Firebase, will facilitate seamless data management. Student or employee information, including names, IDs, and attendance records, will be securely stored in the cloud. This integration ensures accessibility to attendance data from any location, empowering administrators with the ability to manage and analyze attendance effortlessly.

The proposed system will boast a user-friendly interface tailored for administrators' ease of use. Through intuitive dashboards and graphical representations, administrators can monitor attendance in real-time and access comprehensive attendance reports with ease. This functionality not only simplifies data analysis but also enables informed decision-making based on attendance trends and statistics. Scalability and adaptability are key considerations in the system's design. Whether deployed in educational institutions, corporate offices, or other settings, the system will offer flexibility and customization options to meet diverse organizational needs. Additionally, stringent security measures, including robust encryption protocols and access controls, will safeguard data privacy and prevent unauthorized access or tampering.

In summary, the proposed system represents a significant advancement in attendance tracking technology, offering automation, efficiency, and reliability through the integration of AI, machine learning, and cloud-based data management. By optimizing attendance tracking processes and providing valuable insights into attendance patterns, the system aims to enhance organizational efficiency, productivity, and data security

2.2.1 ADVANTAGES

Efficiency: The proposed attendance system will offer efficient attendance tracking, reducing the time and effort required for manual processes. With automatic detection and real-time updating of attendance records, administrators can streamline administrative tasks and allocate resources more effectively.

Accuracy: By leveraging AI-powered face recognition technology, the system ensures precise attendance marking, minimizing the risk of errors associated with human input. This heightened accuracy enhances the reliability of attendance data, providing stakeholders with confidence in the system's performance.

Accessibility: The proposed system will be accessible to all stakeholders, offering a user-friendly interface and seamless integration with cloud-based databases. Administrators can easily access attendance information from any location, facilitating remote monitoring and decision-making. Additionally, the system will be designed to accommodate diverse organizational needs, ensuring accessibility for users across different settings.

CHAPTER-3

SYSTEM SPECIFICATION

3.1 HARDWARE SPECIFICATION

Processor	: AMD Ryzen 5 5500U
RAM Capacity	: 8GB
Hard Disk	: 160 GB
Mouse	: Logical Optical Mouse
Keyboard	: 104Keys
Monitor	: 16 inches
Speed	: 2.10 GHZ
Floppy Disk Drive	: 100 MB

3.2 SOFTWARE SPECIFICATION

Operating System	: Windows11
Frontend	: Open CV, Pickle
Backend	: Fire Base
Language	: Python

3.3 APPLICATION SPECIFICATION

FRONT END: Open CV

OpenCV, short for Open Source Computer Vision Library, is a powerful open-source computer vision and machine learning software library. Developed originally by Intel, it has become a go-to tool for various applications in image processing, computer vision, and machine learning. OpenCV offers a wide range of functionalities, including image and video manipulation, object detection and recognition, facial recognition, feature extraction, and more. It provides a comprehensive suite of algorithms and tools that enable developers to efficiently process and analyze visual data. OpenCV is written in C++ and has bindings for Python, making it accessible to a broad community of developers. Its cross-platform nature allows it to run seamlessly on different operating systems, including Windows, Linux, macOS, Android, and iOS. With its extensive documentation, active community support, and continuous development, OpenCV remains a cornerstone in the field of computer vision, facilitating the development of innovative solutions across various domains.

PYTHON

Python is a very popular general-purpose interpreted, interactive, object-oriented, and high-level programming language. Python is dynamically typed and garbage-collected programming language. It was created by Guido van Rossum during 1985- 1990. Like Perl, Python source code is also available under the GNU General Public License (GPL).

Python supports multiple programming paradigms, including Procedural, Object Oriented and Functional programming language. Python design philosophy emphasizes code readability with the use of significant indentation.

This tutorial gives a complete understanding of Python programming language starting from basic concepts to advanced concepts. This tutorial will take you through simple and practical approaches while learning Python Programming language.

Python Syntax compared to other programming languages

- Python was designed for readability and has some similarities to the English language. with influence from mathematics.
- Python uses new lines to complete a command, as opposed to other programming languages which often use semicolons or parentheses.

3.3.2 BACKEND: Fire Base

Firebase is a comprehensive platform developed by Google that offers various backend services and tools to help developers build and scale their applications. It provides a wide range of features, including real-time database, authentication, cloud storage, hosting, machine learning, analytics, and more, all integrated into a single platform.

One of the core components of Firebase is the Realtime Database, which is a NoSQL cloud database that allows developers to store and sync data in real-time between clients and servers. This real-time synchronization enables collaborative and responsive applications, making it ideal for applications requiring live updates and data synchronization across multiple devices.

Firebase also offers Authentication services, allowing developers to easily add user authentication and authorization to their applications, supporting various authentication methods like email/password, social login providers (Google, Facebook, etc.), and custom authentication systems.

Additionally, Firebase provides Cloud Storage, enabling developers to store and serve user-generated content such as images, videos, and other files securely in the cloud. It offers scalable and reliable storage solutions with easy integration into Firebase-powered applications.

Furthermore, Firebase offers Hosting services, allowing developers to deploy and host web applications quickly and securely with built-in SSL encryption and global content delivery network (CDN) support.
With its comprehensive set of features, ease of use, scalability, and real-time capabilities, Firebase has become a popular choice for developers looking to build modern, feature-rich applications across various platforms, including web, mobile, and desktop.

CHAPTER-4

SYSTEM DESIGN AND DEVELOPMENT

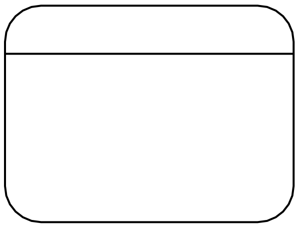
4.1 DATAFLOW DIAGRAM

A data flow diagram is graphical tool used to describe and analyze movement of data through a system. These are the central tool and the basis from which the other components are developed. The transformation of data from input to output, through processed, may be described logically and independently of physical components associated with the system. These are known as the logical data flow diagrams. The physical data flow diagrams show the actual implements and movement of data between people, departments, and workstations.

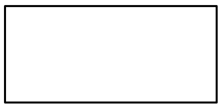
DFD SYMBOLS:

In the DFD, there are four symbols.

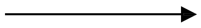
1. A square defines a source(originator) or destination of system data.
2. An arrow identifies data flow. It is the pipeline through which the information flows.
3. A circle or a bubble represents a process that transforms incoming data flow into outgoing data flows.
4. An open rectangle is a data store, data at rest or a temporary repository of data.



Process that transforms data flow.



Source or Destination of data



Data flow

Data Store

CONSTRUCTING A DFD:

Several rules of thumb are used in drawing DFD'S:

1. Process should be named and numbered for an easy reference. Each name should be representative of the process.
2. The direction of flow is from top to bottom and from left to right. Data traditionally flow from source to the destination although they may flow back to the source. One way to indicate this is to draw long flow line back to a source. An alternative way is to repeat the source symbol as a destination. Since it is used more than once in the DFD it is marked with a short diagonal.
3. When a process is exploded into lower level details, they are numbered.
4. The names of data stores and destinations are written in capital letters. Process and dataflow names have the first letter of each work capitalized.

4.2 INPUT DESIGN

Input design is a part of overall system design. The main objective during the input design as given below:

- Webcam Input: The system captures live video feed from the webcam to detect faces for attendance tracking.
- Student Images: Input design includes uploading student images to the system, which are then encoded and stored for recognition during attendance marking.
- Student Information: Student details such as name, major, and starting year are inputted into the system either manually or through a data import mechanism.

4.3 OUTPUT DESIGN

Outputs from computer systems are required primarily to communicate the results of processing to users. They are also used to provide a permanent copy of the results for later consultation. The various types of outputs in general are:

- Attendance Marking: The system provides real-time feedback on attendance marking, indicating whether a recognized face corresponds to a registered student and updating attendance records accordingly.

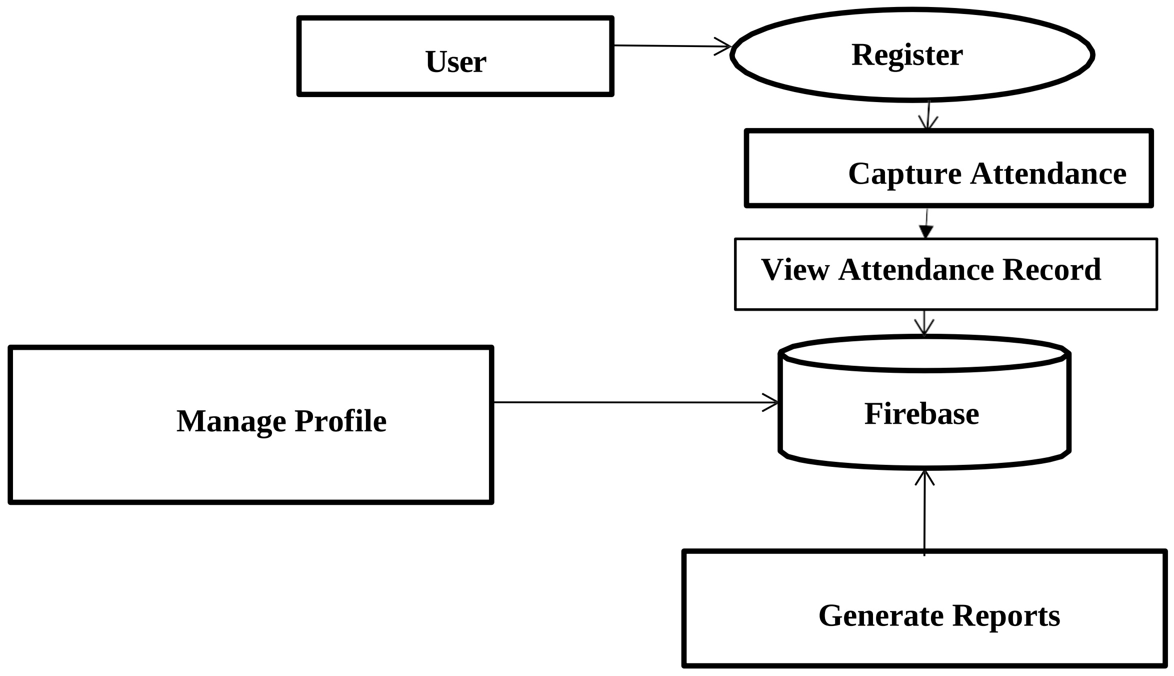
- Visual Feedback: The system displays visual feedback to users, highlighting detected faces, displaying relevant student information alongside recognized faces, and providing status updates on attendance marking.
- Attendance Reports: The system generates attendance reports, summarizing attendance records for individual students or entire classes over specified time periods.
- Error Messages: In case of errors or exceptions, the system outputs informative error messages to guide users in resolving issues effectively.

4.4 DATABASE DESIGN

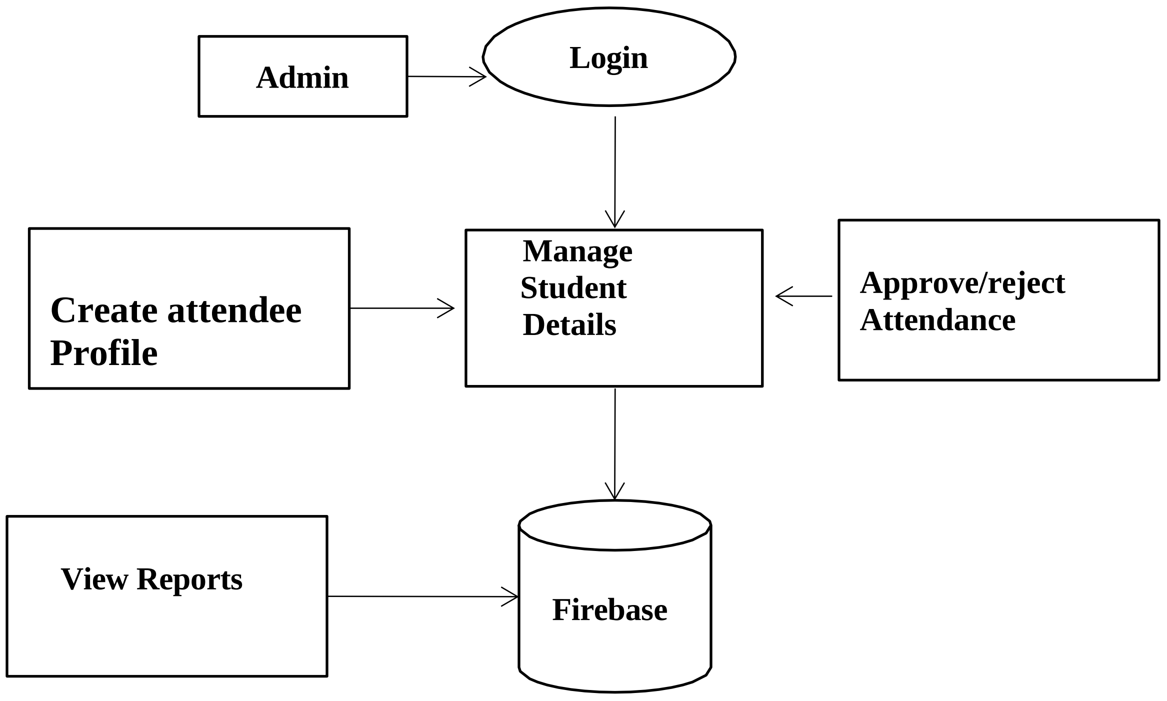
The database design for the face recognition attendance system utilizes Firebase Realtime Database, which serves as the primary storage mechanism for attendee details and attendance records. Within the database, attendee information is organized under a "Students" table, containing attributes such as unique student IDs, names, majors, starting years, total attendance counts, standing, academic years, and timestamps for the last attendance records. Additionally, attendance records, including dates and times of attendance, are likely stored either within each student's entry in the "Students" table or in a separate table for efficient tracking of attendance history. Firebase Storage is employed for securely storing attendee images, while Firebase Authentication ensures user authentication and access control.

Leveraging Firebase Realtime Database's real-time data synchronization capabilities, the system ensures that any updates to attendee details or attendance records are promptly propagated to all connected clients, maintaining data accuracy and accessibility. This design facilitates scalability, performance, and real-time data management, making it suitable for the dynamic nature of attendance tracking in the face recognition attendance system.

Data Flow Diagram
User



Admin



4.5 MODULES SPLIT UP

MODULE DESCRIPTION

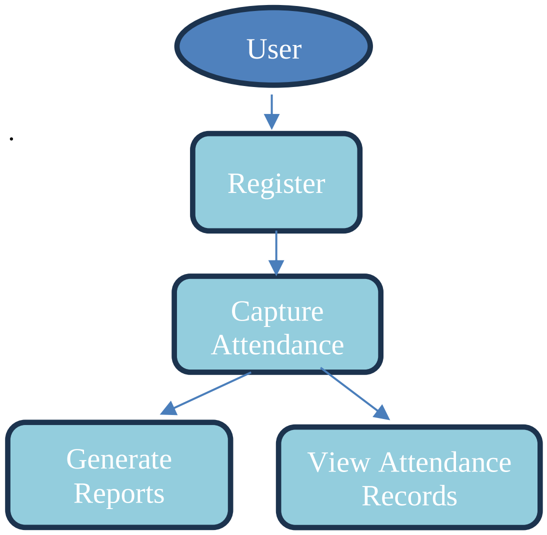
User

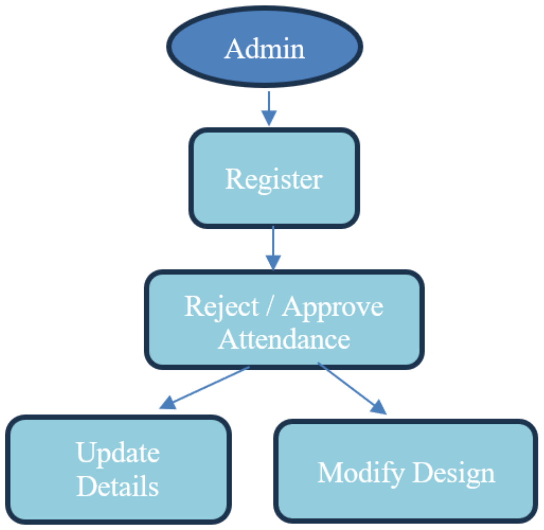
- **Register:** Users need to register by providing their basic details to access the auto attendance system.
- **Capture Attendance**: The system captures attendance by using face recognition techniques, identifying registered individuals, and marking their presence.
- **View Attendance Records**: Users can view their attendance records, including dates and times of attendance.
- **Generate Reports:** The system allows users to generate reports summarizing their attendance history, including total attendance, late arrivals, etc.
- **Manage Profile:** Users can manage their profiles by updating personal information such as name, contact details, etc.

Admin

- **Login:** Admins can access the admin dashboard by logging in with their username and password.
- **Approve/Reject Attendance Records:** Admins have the authority to approve or reject attendance records marked by the system, ensuring accuracy and reliability.
- **Create Student Profiles:** Admins can create student profiles by entering their details such as name, student ID, and other relevant information. These profiles are stored in the database for attendance tracking.
- **Manage Student Details:** Admins can manage student details, including adding new students, updating existing profiles, and removing outdated information as necessary.
- **View Attendance Reports:** Admins have access to attendance reports, allowing them to view attendance trends, monitor student participation, and identify any irregularities in attendance patterns.

4.6 ARCHITECTURAL DESIGN





4.7 TABLE DESIGN

User

User ID	Name	Major	Password	Starting Year	Total Attendance	Standing Year	Attendance Time
Int	Varchar	Varchar	Varchar	Varchar	Int	Int	Varchar
100	100	100	100	100	100	100	100
Primary key							

CHAPTER-5

IMPLEMENTATION AND TESTING

5.1 TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the

The software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of tests. Each test type addresses a specific testing requirement.

5.1.1 TYPES OF TESTS

Unit testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

Integration testing

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

Functional test

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input	: identified classes of valid input must be accepted.
Invalid Input	: identified classes of invalid input must be rejected.
Functions	: identified functions must be exercised.
Output	: identified classes of application outputs must be exercised.
Systems/Procedures	: interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

System Test

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration-oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

White Box Testing

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

Black Box Testing

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box. you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

Unit Testing

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

Test strategy and approach

Field testing will be performed manually, and functional tests will be written in detail.

Test objectives

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.

Features to be tested

- Verify that the entries are of the correct format.
- No duplicate entries should be allowed.
- All links should take the user to the correct page.

Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects. The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

Acceptance Testing

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

5.2 SYSTEM IMPLEMENTATION:

Systems implementation is the process of:

1. defining how the information system should be built (i.e., physical system design).
2. ensuring that the information system is operational and used.
3. ensuring that the information system meets quality standard (i.e., quality assurance).

Systems design

Conceptual design – what the system should do

Logical design – what the system should look to the user.

Physical design – how the system should be built.

Physical system design using structured design approach:

To produce a system that is easy to read, code, and maintain.

1. Factoring: decomposition
2. Span of control: 9 subordinate modules
3. Reasonable size: 50-100 LOC
4. Coupling: minimize inter-module dependency
5. Cohesion: single module functionality
6. Shared use: multiple calls to lower-level modules from different bosses

CHAPTER-6

CONCLUSION

In conclusion, the auto attendance system project represents a significant advancement in the domain of attendance management. By leveraging the capabilities of Python, OpenCV, Firebase, and face recognition technology, the system streamlines the process of recording attendance in various educational or organizational settings. Through the development of modules for user registration, login, and attendance tracking, the project ensures ease of use and efficient management of attendance records.

The project's implementation of facial recognition technology enables accurate identification of individuals, reducing the possibility of errors and ensuring the integrity of attendance data. Additionally, the integration with Firebase facilitates real-time data storage and retrieval, providing administrators and users with access to up-to-date attendance information from anywhere, at any time.

With its user-friendly interface and robust functionality, the auto attendance system project offers a comprehensive solution for attendance management, benefiting educational institutions, businesses, and other organizations alike. Moving forward, further enhancements and refinements could be made to expand the system's capabilities and address specific user requirements, ensuring its continued relevance and effectiveness in meeting the evolving needs of attendance tracking in various contexts.

CHAPTER-7

SCOPE FOR FURTHER ENHANCEMENT

The auto attendance system project presents several opportunities for further enhancement to improve functionality, usability, and overall effectiveness. One potential area for improvement is the implementation of enhanced security features, such as multi-factor authentication or biometric verification, to provide users with greater data protection and prevent unauthorized access. Another avenue for enhancement is the development of a mobile application version of the system, offering increased flexibility and accessibility for users to mark attendance using their smartphones from anywhere.

Furthermore, integrating data analytics and reporting capabilities into the system would enable administrators to generate insightful reports on attendance trends, patterns, and student performance. These analytics-driven insights can assist educational institutions in making data-driven decisions to optimize resource allocation, improve teaching methods, and enhance overall student engagement and success. Additionally, exploring integration with IoT devices such as smart cameras or sensors could automate attendance tracking in physical classrooms or events, eliminating the need for manual intervention and improving efficiency and accuracy in attendance recording.

BIBLIOGRAPHY

[1] Riaz Munshi, Steven Puttemans, and Roy Shilkrot, "Hands-On OpenCV 4 with Python".

[2] Adrian Kaehler and Gary Bradski, "Learning OpenCV 4: Computer Vision with Python".

[3] Ivan Vasilev and Daniel Slater, "Python Deep Learning".

[4] Himanshu Singh, "Practical Machine Learning and Image Processing: For Facial Recognition, Object Detection, and Pattern Recognition Using Python".

[5] Foster Provost and Tom Fawcett, "Data Science for Business: What You Need to Know about Data Mining and Data-Analytic Thinking".

[6] Rohit S., "Face Detection using OpenCV and Python: A Beginner's Guide" [Medium].

[7] Bhavesh Singh, "Face Recognition with OpenCV" [Towards Data Science].

[8] Joseph Redmon and Ali Farhadi, "YOLOv3: An Incremental Improvement" [arXiv].

[9] Rishabh Jain, "OpenCV Tutorial: A Guide to Learn OpenCV" [Analytics Vidhya].

[10] Shubham Jaiswal, "Building a Face Recognition System with OpenCV" [DataCamp].

REFERENCE WEBSITES

[1] https://www.pyimagesearch.com/
[2] https://learnopencv.com/
[3] https://www.analyticsvidhya.com/
[4] https://towardsdatascience.com/
[5] https://arxiv.org/
[6] https://www.datacamp.com/community/tutorials/face-detection-python-opencv
[7] https://www.analyticsvidhya.com/blog/2018/08/a-guide-to-face-detection-in-python/
[8] https://pjreddie.com/darknet/yolo/
[9] https://www.analyticsvidhya.com/learn/opencv-tutorial/
[10] https://www.datacamp.com/community/tutorials/face-recognition-python-opencv

ANNEXURES

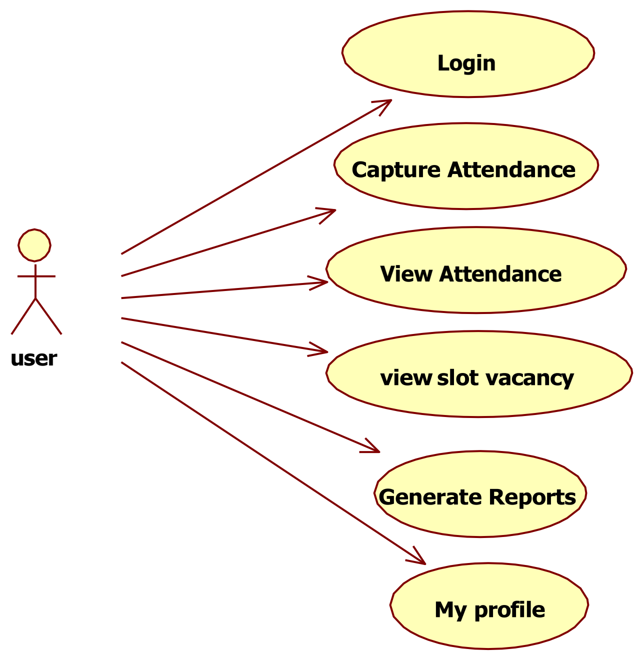
A) UML stands for Unified Modeling Language. UML is a language for specifying, visualizing, and documenting the system. This is the step while developing any product after analysis. The goal from this is to produce a model of the entities involved in the project which later need to be built. The representation of the entities that are to be used in the product being developed need to be designed.

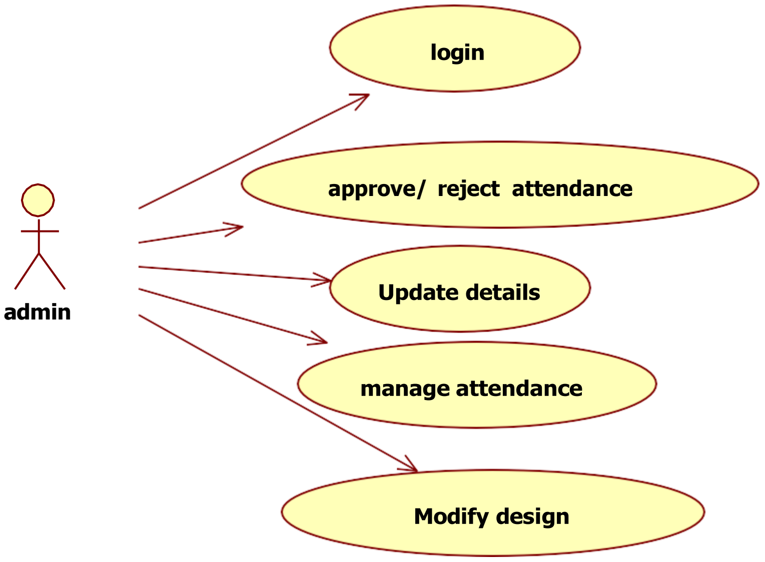
There are various kinds of methods in software design:

- Use case Diagram
- Sequence Diagram
- Collaboration Diagram

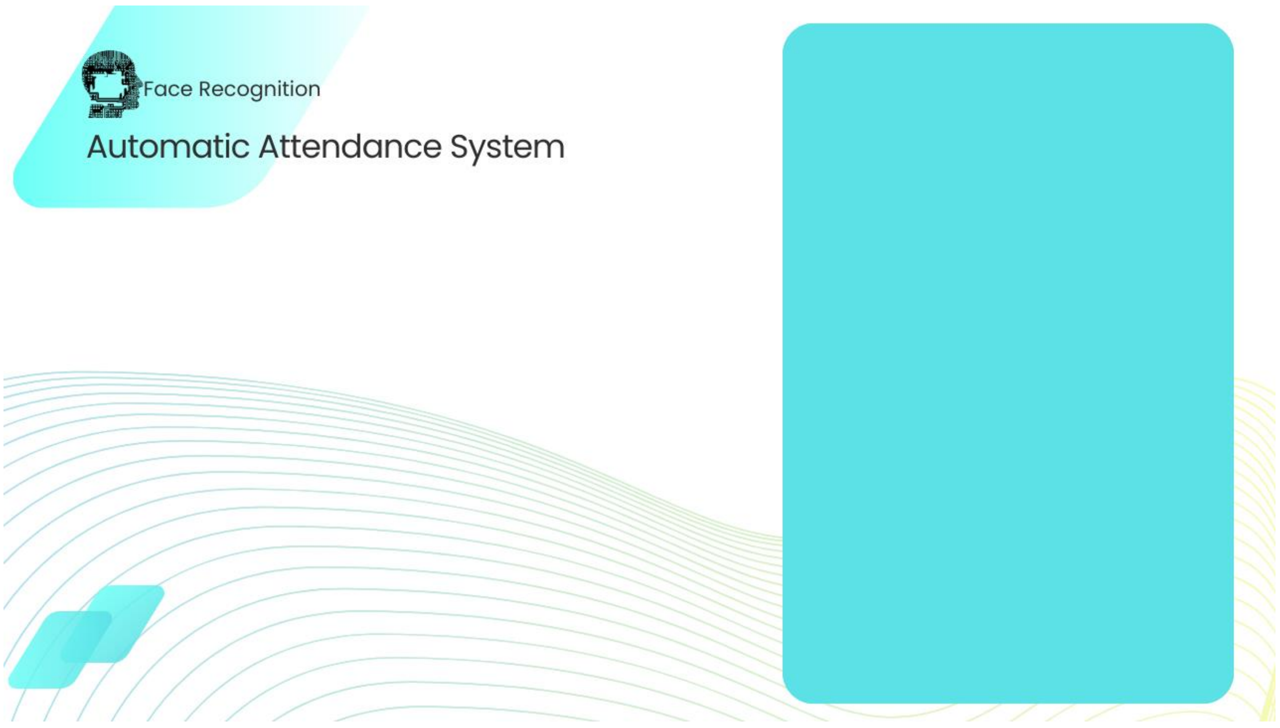
Use case Diagrams:

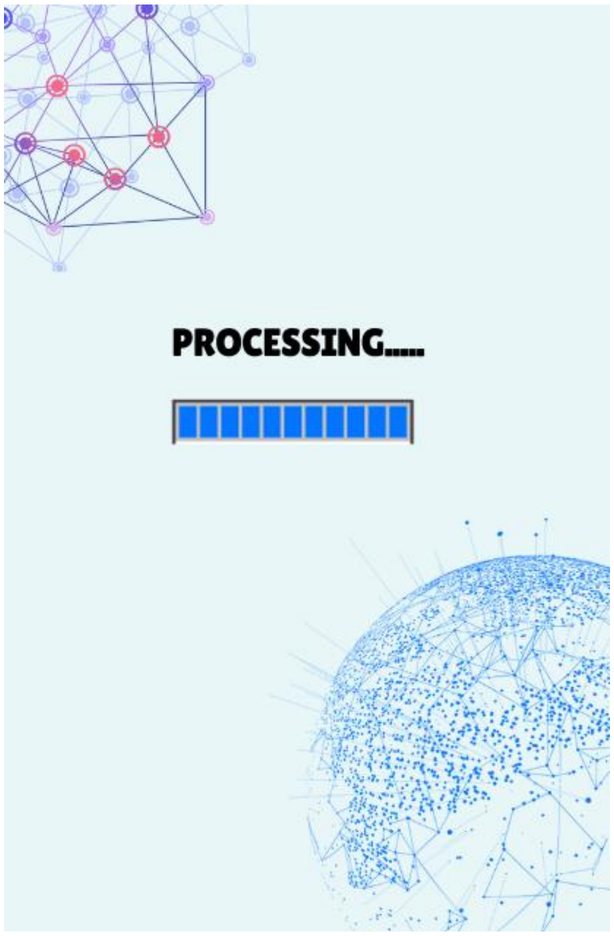
Use case diagrams model behavior within a system and helps the developers understand of what the user require. The stick man represents what's called an actor. Use case diagram can be useful for getting an overall view of the system and clarifying that can do and more importantly what they can't do.

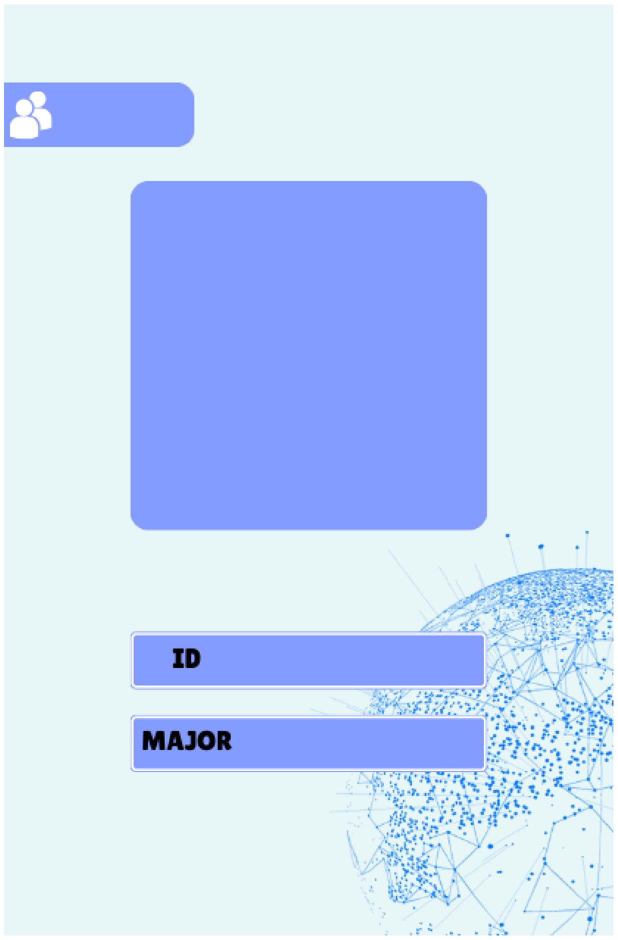


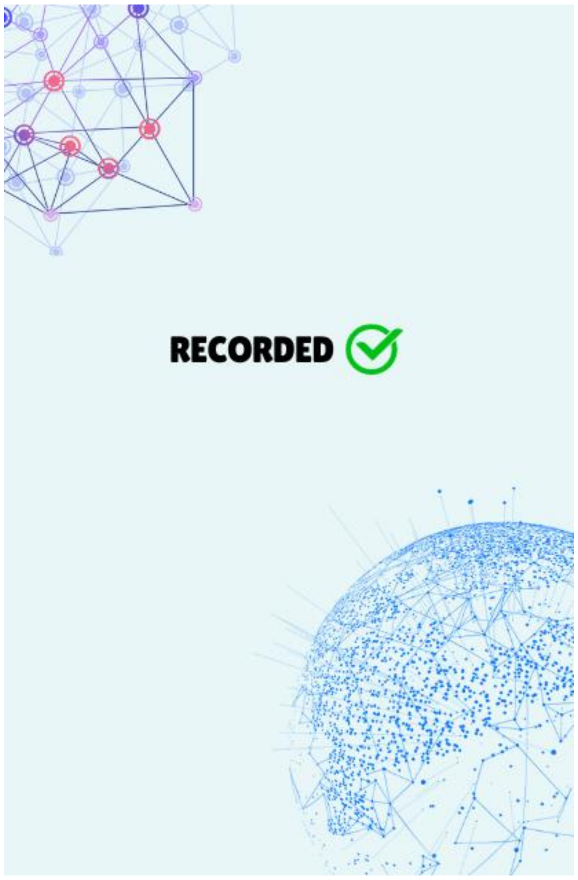


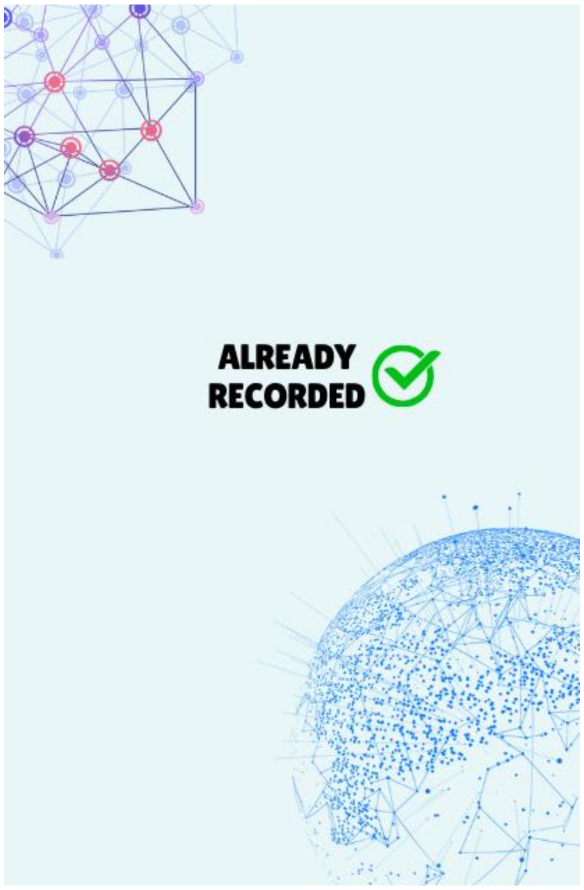
SCREENSHOTS



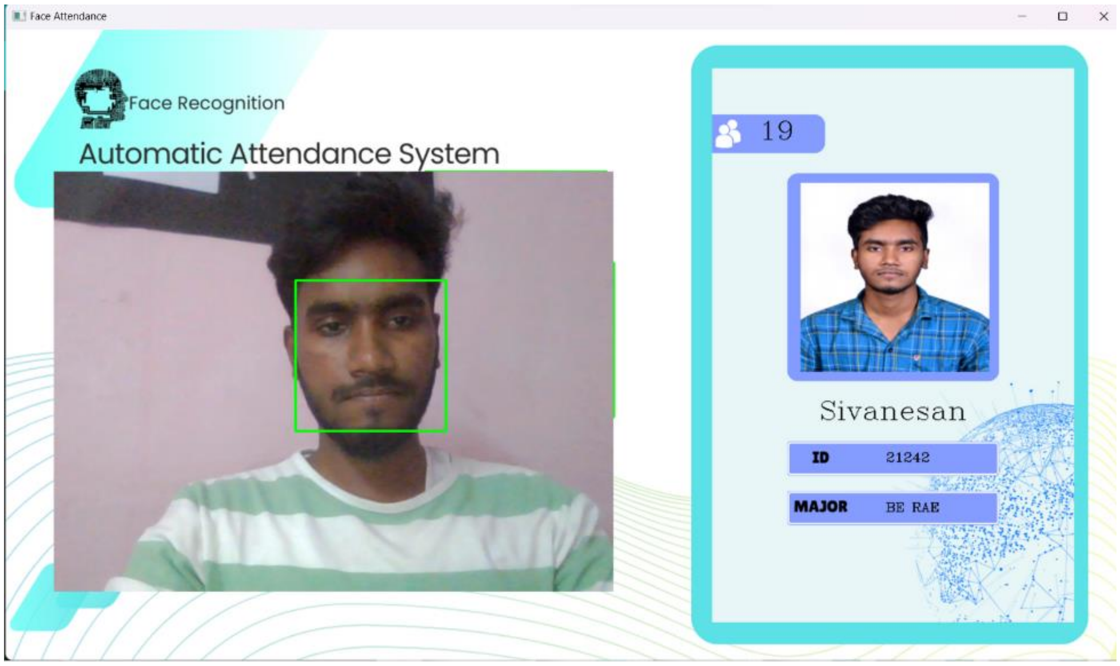

PROCESSING.....

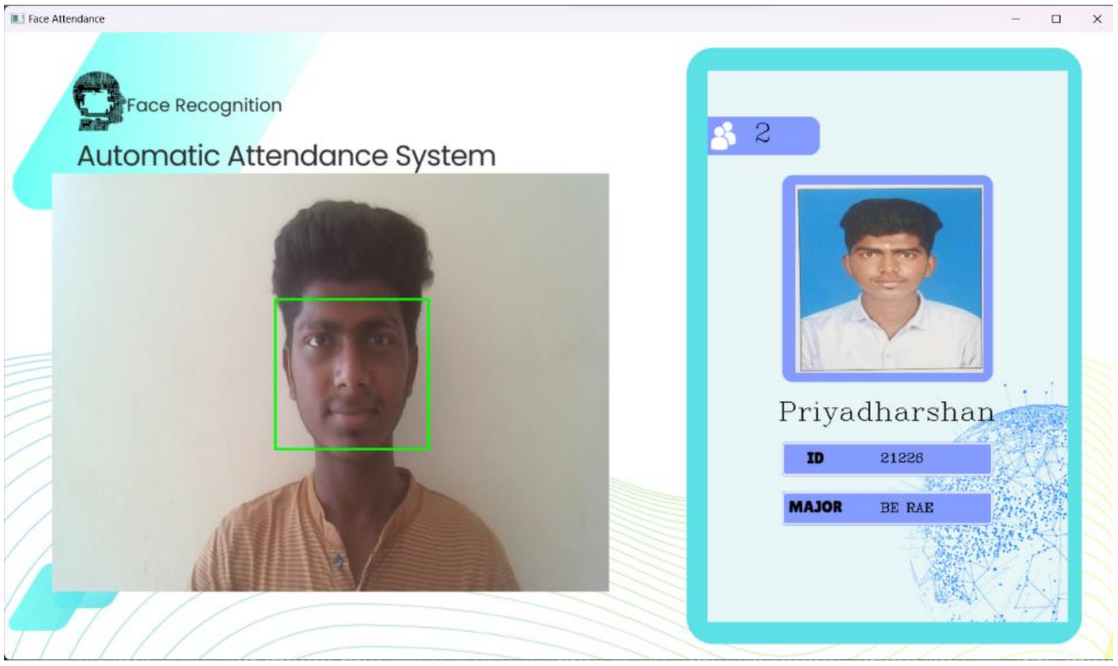

ID
MAJOR

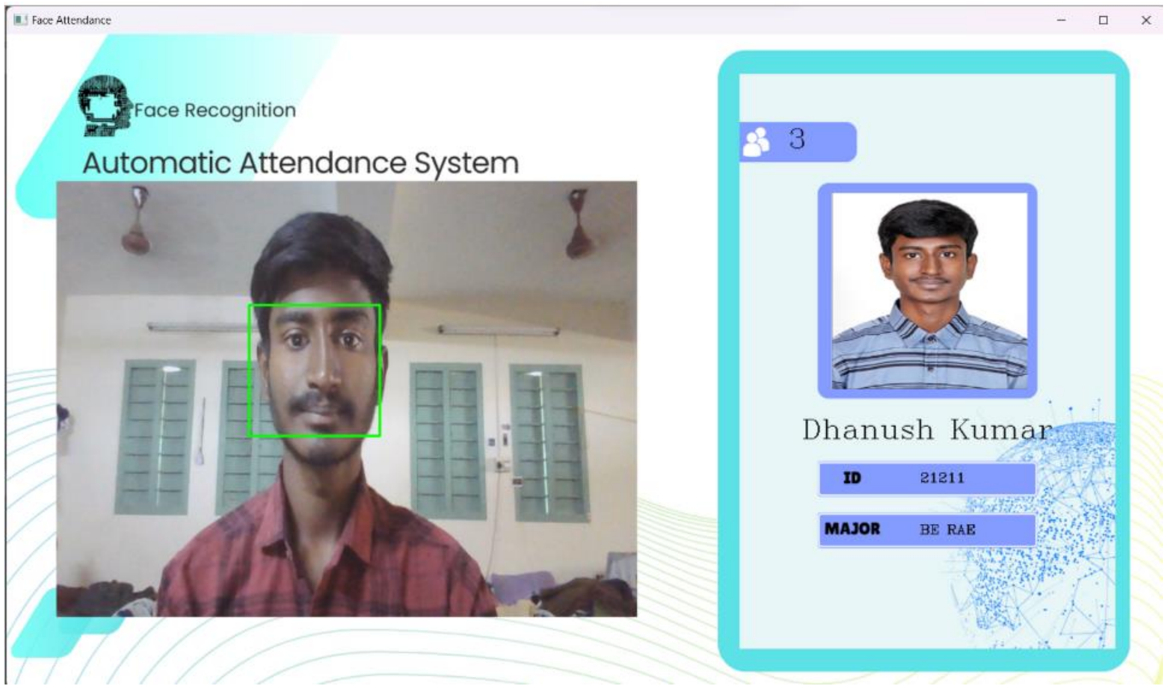

RECORDED


ALREADY RECORDED

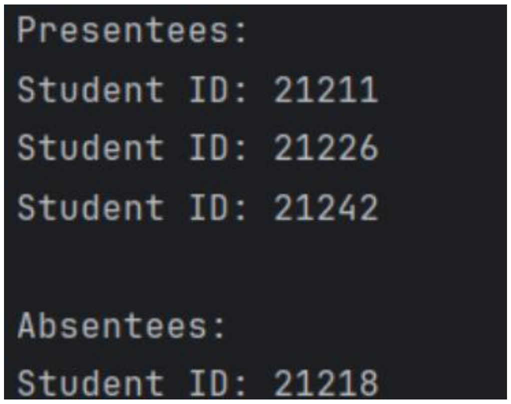
Sample Images:





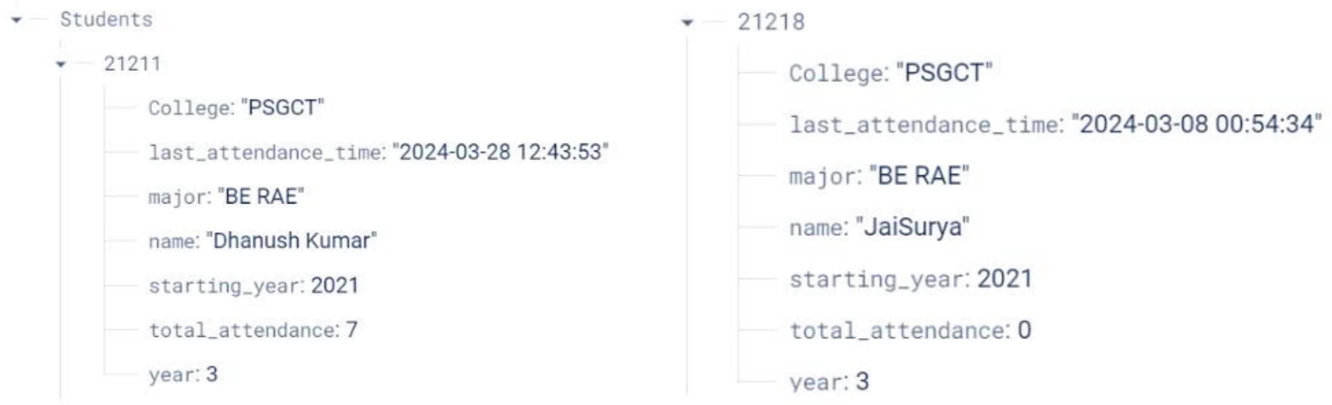


Output In PyCharm :



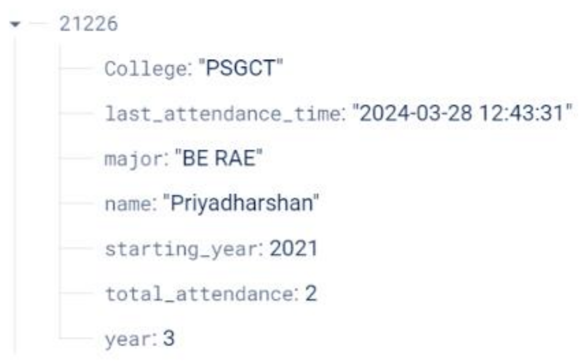
```
Presentees:
Student ID: 21211
Student ID: 21226
Student ID: 21242

Absentees:
Student ID: 21218
```

FireBase Output:

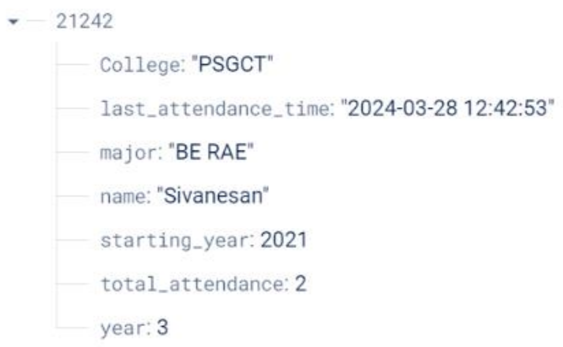


```
Students
  21211
    College: "PSGCT"
    last_attendance_time: "2024-03-28 12:43:53"
    major: "BE RAE"
    name: "Dhanush Kumar"
    starting_year: 2021
    total_attendance: 7
    year: 3
```

```
21218
  College: "PSGCT"
  last_attendance_time: "2024-03-08 00:54:34"
  major: "BE RAE"
  name: "JaiSurya"
  starting_year: 2021
  total_attendance: 0
  year: 3
```



```
21226
    College: "PSGCT"
    last_attendance_time: "2024-03-28 12:43:31"
    major: "BE RAE"
    name: "Priyadharshan"
    starting_year: 2021
    total_attendance: 2
    year: 3
```



```
21242
    College: "PSGCT"
    last_attendance_time: "2024-03-28 12:42:53"
    major: "BE RAE"
    name: "Sivanesan"
    starting_year: 2021
    total_attendance: 2
    year: 3
```

SOURCE CODE

Main.py

```
import os
import pickle
import numpy as np
import cv2
import face_recognition
import cvzone
import firebase_admin
from firebase_admin import credentials
from firebase_admin import db
from firebase_admin import storage
import numpy as np
from datetime import datetime

cred = credentials.Certificate("serviceAccountKey.json")
firebase_admin.initialize_app(cred, {
    'databaseURL': "https://autoattendance-763d3-default-rtdb.firebaseio.com/Q",
    'storageBucket': "autoattendance-763d3.appspot.com"
})
bucket = storage.bucket()

img = cv2.VideoCapture(0)
```

```
img.set(3, 640)   #setting the width of webcam
img.set(4, 480)   #setting height of webcam
if not img.isOpened():
    print("Error: Could not open camera.")
    exit()


imgBackground = cv2.imread('GUI_files/C.png')

# Importing the mode images into a list
folderModePath = 'GUI_files/Modes'
modePathList = os.listdir(folderModePath)
imgModeList = []
imgList=[]
studentIds=[]
for path in modePathList:
    imgModeList.append(cv2.imread(os.path.join(folderModePath, path)))
# print(len(imgModeList))

# Load the encoding file
print("Loading Encode File ...")
file = open('EncodeFile.p', 'rb')
encodeListKnownWithIds = pickle.load(file)
file.close()
encodeListKnown, studentIds = encodeListKnownWithIds
# print(studentIds)
print("Encode File Loaded")

modeType = 0
counter = 0
id = -1
imgStudent = []
```

```
folderPath = 'Students_Images'
pathList = os.listdir(folderPath)
print(pathList)
attendance_status = {student_id: False for student_id in studentIds}  # Initialize attendance
status
for path in pathList:
    imgList.append(cv2.imread(os.path.join(folderPath, path)))
    student_id = os.path.splitext(path)[0]
    studentIds.append(student_id)
    attendance_status[student_id] = False  # Initialize each student as absent

    # Construct the file path for the image in the local system
    fileName = f'{folderPath}/{path}'

    # Obtain a reference to the default Cloud Storage bucket for the Firebase project
    bucket = storage.bucket()

    # Create a reference to an object (file) within the Cloud Storage bucket
    blob = bucket.blob(fileName)

    # Upload the local file specified by fileName to the Cloud Storage bucket
    blob.upload_from_filename(fileName)

while True:
    success, frame= img.read()

    imgS = cv2.resize(frame, (0, 0), None, 0.25, 0.25)
    imgS = cv2.cvtColor(imgS, cv2.COLOR_BGR2RGB)

    faceCurFrame = face_recognition.face_locations(imgS)
    encodeCurFrame = face_recognition.face_encodings(imgS, faceCurFrame)
    imgBackground[162:162 + 480, 55:55 + 640] = frame
    imgBackground[44:44 + 633, 808:808 + 414] = imgModeList[modeType]

    if faceCurFrame:
        for encodeFace, faceLoc in zip(encodeCurFrame, faceCurFrame):
            matches = face_recognition.compare_faces(encodeListKnown, encodeFace)
            faceDis = face_recognition.face_distance(encodeListKnown, encodeFace)
            # print("matches", matches)
            # print("faceDis", faceDis)

            matchIndex = np.argmin(faceDis)
            # print("Match Index", matchIndex)

            if matches[matchIndex]:
                id = studentIds[matchIndex]
                if not attendance_status[id]:
                    attendance_status[id] = True  # Mark the student as present
                # print("Known Face Detected")
                # print(studentIds[matchIndex])
                y1, x2, y2, x1 = faceLoc
                y1, x2, y2, x1 = y1 * 4, x2 * 4, y2 * 4, x1 * 4
                #bbox = 55 + x1, 162 + y1, x2 - x1, y2 - y1
                #imgBackground = cvzone.cornerRect(imgBackground, bbox, rt=0)
                # Create a simple square covering the detected face with a specified color (e.g.,
green)
                color = (0, 255, 0)  # Green color in BGR format
                thickness = 2  # Fill the square



                # Draw a filled rectangle on the image
                imgBackground = cv2.rectangle(imgBackground, (x1 + 55, y1 + 162), (x2 + 55, y2 +
162), color, thickness)
                id = studentIds[matchIndex]
                if counter == 0:
                    #cvzone.putTextRect(imgBackground, "Loading", (300, 700))
                    cv2.imshow("Face Attendance", imgBackground)
                    cv2.waitKey(1)
                    counter = 1
                    modeType = 1

        if counter != 0:

            if counter == 1:
                # Get the Data
                studentInfo = db.reference(f'Students/{id}').get()
                print(studentInfo)
                # Get the Image from the storage
                blob = bucket.get_blob(f'Images/{id}.png')
                array = np.frombuffer(blob.download_as_string(), np.uint8)
                imgStudent = cv2.imdecode(array, cv2.COLOR_BGRA2BGR)
                # Update data of attendance
                datetimeObject = datetime.strptime(studentInfo['last_attendance_time'],
                                                   "%Y-%m-%d %H:%M:%S")
                secondsElapsed = (datetime.now() - datetimeObject).total_seconds()
                print(secondsElapsed)
                if secondsElapsed > 30:
                    ref = db.reference(f'Students/{id}')
                    studentInfo['total_attendance'] += 1
                    ref.child('total_attendance').set(studentInfo['total_attendance'])
                    ref.child('last_attendance_time').set(datetime.now().strftime("%Y-%m-%d
%H:%M:%S"))
                else:
                    modeType = 3
                    counter = 0
                    imgBackground[44:44 + 633, 808:808 + 414] = imgModeList[modeType]
            if modeType != 3:

                if 10 < counter < 20:
                    modeType = 2
                    cv2.waitKey(1)

                imgBackground[44:44 + 633, 808:808 + 414] = imgModeList[modeType]

                if counter <= 10:
                    cv2.putText(imgBackground, str(studentInfo['total_attendance']), (861, 125),
                                cv2.FONT_HERSHEY_COMPLEX, 1, (255, 255, 255), 1)
                    cv2.putText(imgBackground, str(studentInfo['major']), (1006, 550),
                                cv2.FONT_HERSHEY_COMPLEX, 0.5, (255, 255, 255), 1)
                    cv2.putText(imgBackground, str(id), (1006, 493),
                                cv2.FONT_HERSHEY_COMPLEX, 0.5, (255, 255, 255), 1)



                    (w, h), _ = cv2.getTextSize(studentInfo['name'],
cv2.FONT_HERSHEY_COMPLEX, 1, 1)
                    offset = (414 - w) // 2
                    cv2.putText(imgBackground, str(studentInfo['name']), (808 + offset, 445),
                                cv2.FONT_HERSHEY_COMPLEX, 1, (50, 50, 50), 1)

                    imgBackground[175:175 + 216, 909:909 + 216] = imgStudent

                counter += 1

                if counter >= 2:
                    counter = 0
                    modeType = 0
                studentInfo = []
                imgStudent = []
                imgBackground[44:44 + 633, 808:808 + 414] = imgModeList[modeType]
        else:
            modeType = 0
            counter = 0
        # cv2.imshow("Webcam", img)
        cv2.imshow("Face Attendance", imgBackground)
        cv2.waitKey(1)

        if cv2.waitKey(10) & 0xFF == 27:  # 27 is the ASCII code for the 'Escape' key
            break



    img.release()
    cv2.destroyAllWindows()
    print("Presentees:")
    for student_id, present in attendance_status.items():
        if present:
            print(f"Student ID: {student_id}")

    print("\nAbsentees:")
    for student_id, present in attendance_status.items():
        if not present:
            print(f"Student ID: {student_id}")
```
Encoding.py
```
import cv2
import face_recognition
import pickle
import os
import firebase_admin
from firebase_admin import credentials
from firebase_admin import db
from firebase_admin import  storage

cred = credentials.Certificate("serviceAccountKey.json")
firebase_admin.initialize_app(cred, {
    'databaseURL': "https://autoattendance-763d3-default-rtdb.firebaseio.com/",
    'storageBucket': "autoattendance-763d3.appspot.com"
})



# Importing student images
folderPath = 'Students_Images'
pathList = os.listdir(folderPath)
#print(pathList)
imgList = []
studentIds = []
for path in pathList:
    imgList.append(cv2.imread(os.path.join(folderPath, path)))
    studentIds.append(os.path.splitext(path)[0])

    fileName = f'{folderPath}/{path}'
    bucket = storage.bucket()
    blob = bucket.blob(fileName)
    blob.upload_from_filename(fileName)



    # print(path)
    # print(os.path.splitext(path)[0])
print(studentIds)



def findEncodings(imagesList):
    encodeList = []
    for img in imagesList:
        img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
        encode = face_recognition.face_encodings(img)[0]
        encodeList.append(encode)

    return encodeList



print("Encoding the Images...")
encodeListKnown = findEncodings(imgList)
encodeListKnownWithIds = [encodeListKnown, studentIds]
print("Encoding Completed")

file = open("EncodeFile.p", 'wb')
pickle.dump(encodeListKnownWithIds, file)
file.close()
print("Saved the encoded file.")
```

Data.py

```
import firebase_admin
from firebase_admin import credentials

from firebase_admin import db

cred = credentials.Certificate("serviceAccountKey.json")
firebase_admin.initialize_app(cred, {
    'databaseURL': "https://autoattendance-763d3-default-rtdb.firebaseio.com/"
})

ref = db.reference('Students')
```

```
},


data = {
    "21226":
        {
            "name": "Priyadharshan",
            "major": "BE RAE",
            "year": 3,
            "starting_year": 2021,
            "College": "PSGCT",
            "total_attendance": 0,
            "last_attendance_time": "2024-03-08 00:54:34"
        },
    "21242":
        {
            "name": "Sivanesan",
            "major": "BE RAE",
            "year": 3,
            "starting_year": 2021,
            "College": "PSGCT",
            "total_attendance": 0,
            "last_attendance_time": "2024-03-08 00:54:34"
        },
    "21218":
        {
            "name": "JaiSurya",
            "major": "BE RAE",
            "year": 3,
            "starting_year": 2021,
            "College": "PSGCT",
            "total_attendance": 0,
            "last_attendance_time": "2024-03-08 00:54:34"
        },
    "21211":
        {
            "name": "Dhanush Kumar",
            "major": "BE RAE",
            "year": 3,
            "starting_year": 2021,
            "College": "PSGCT",
            "total_attendance": 0,
            "last_attendance_time": "2024-03-08 00:54:34"
        },

}
for key, value in data.items():
    ref.child(key).set(value)
```