A

PROJECT REPORT ON

"SMART ATTENDANCE SYSTEM"

SUBMITTED
TO

SAVITRIBAI PHULE PUNE UNIVERSITY, PUNE

FOR THE PARTIAL FULFILLMENT OF

MASTER OF COMPUTER APPLICATION

(MCA-II, SEM.-III)

BY

KOKANE VISHAL DATTATRAY

UNDER THE GUIDANCE OF

PROF. YOGESH SHARMA

THROUGH

THE DIRECTOR
SINHGAD INSTITUTE OF MANAGEMENT AND COMPUTER APPLICATION
(SIMCA), NARHE, PUNE (AY. 2022-2024)

CERTIFICATE

This is to certify that, the project entitled "***Smart Attendance System*"**, being submitted for the partial fulfillment of the degree of **Master of Computer Application** by him to **Sinhgad Institute of Management and Computer Application affiliated to Savitribai Phule Pune University, Pune** is the result of the original work completed by ***Kokane Vishal Dattatray*** under the guidance of ***Prof. Yogesh Sharma***.

To the best of our knowledge and belief, this work has not been previously submitted by the award of any degree or diploma of Savitribai Phule Pune University or any other University.

Place: Narhe, Pune

Date:

Prof. Yogesh Sharma	**Prof. Navanath Choudhari**	**Dr. Vijaya Puranik**
Internal Guide	**Project Co-Ordinator**	**Director SIMCA**

External Examiner

DECLARATION

I, the undersigned hereby declare that the project titled "**Smart Attendance System",** being submitted for the award of degree of **Master of Computer Application** by me to **Sinhgad Institute of Management and Computer Application (SIMCA) affiliated to Savitribai Phule Pune University is** the result of an independent work carried out under the guidance of **Prof**. **Yogesh Sharma,** is my original work. Further I declare that this project has not been submitted to this or any Institution for the award of any degree.

Place : Pune　　　　　　　　　　Kokane Vishal Dattatray

Date :　　　　　　　　　　　　　(Student)

ACKNOWLEDGEMENT

We take this opportunity to express our gratitude towards all those who have helped us throughout the successful completion of our project Firstly, we would like to thank our Director **Dr. Vijaya Puranik**, and project coordinator **Prof. Navanath Choudhari** of Masters in Computer Application, for permitting us to complete our project and for showing faith in us and allowing us to develop **Smart Attendance System**. We must convey our gratitude to our guide **Prof. Yogesh Sharma** for giving us the constant guidance of inspiration and help in preparing the project, personally, correcting our work and providing encouragement throughout the project

Kokane Vishal Dattatray

Student Name

INDEX

1. Introduction

In the ever-evolving landscape of education and workforce management, attendance tracking remains a critical aspect of organizational efficiency. Traditional methods of attendance management are often time-consuming and prone to errors. In response to these challenges, the Smart Attendance System emerges as a revolutionary solution.

The main objective of the Smart Attendance System is to leverage advanced facial recognition technology to automate the attendance tracking process. Students' faces are scanned and identified in real-time, eliminating the need for manual data entry.

With this system, educators can access precise attendance records instantly, enhancing classroom efficiency and accountability. Students benefit from a streamlined process, ensuring their attendance is accurately recorded without disruptions.

1.1 Abstract

The Smart Attendance System (SAS) revolutionizes traditional attendance tracking by leveraging laptop cameras for facial recognition. This automated solution accurately captures attendance, eliminating manual efforts and enhancing efficiency. SAS securely uploads attendance data to the cloud, ensuring data integrity. Users can conveniently access real-time attendance records via a user-friendly mobile app. SAS streamlines attendance management, offering a seamless, secure, and intelligent approach to tracking attendance.

1.2 Existing System and Need for System

Traditional attendance systems rely on manual methods, including paper-based registers or card-swiping machines, which are susceptible to errors, time-consuming, and lack real-time data accessibility. These systems often lead to inaccuracies, making it challenging to maintain reliable attendance records. With the advancement of technology, there arises a need for a more efficient and automated attendance management system.

The Need for the System:

- Automated Data Management
- Accurate Tracking
- Real-time Accessibility
- Seamless Integration
- Enhanced Connectivity
- Enhanced Security

2. Proposed System

The Smart Attendance System is a groundbreaking solution designed to enhance the conventional attendance tracking methods in educational institutions. Leveraging state-of-the-art facial recognition technology, this system offers a myriad of advanced features, ensuring a seamless and efficient attendance management process:

- User-Friendly Interface
- Facial Recognition Technology
- Automated Attendance Tracking
- Real-time Monitoring
- Ads free experience
- Cloud-Based Storage

2.1 Objectives of Proposed System

- Efficient Attendance Management: SAS focuses on optimizing the attendance tracking process. By automating data collection through camera systems, it eliminates the need for manual attendance taking, saving valuable class time and reducing administrative workload.

- Seamless Cloud Integration: The system seamlessly integrates with cloud storage, ensuring that attendance data is securely stored and easily accessible from anywhere. This integration enhances data management, backup, and accessibility, promoting a streamlined administrative workflow.

- Real-time Monitoring and Reporting: SAS provides real-time monitoring capabilities, allowing educators to instantly access attendance records. Additionally, it generates detailed reports, aiding in comprehensive analysis and decision-making for both teachers and administrators.

- Enhanced User Experience: With an intuitive interface, SAS offers a user-friendly experience for both educators and administrators. Its simplicity ensures that users can navigate the system effortlessly, making attendance management efficient and hassle-free.

2.2 Users Summery

Smart Attendance System (SAS) caters to the needs of two primary user groups: students and educators, providing tailored features and benefits to enhance their experience:

Students:

- Effortless Attendance Tracking
- Real-time Access
- Efficient Check-In/Out
- Convenience

Educators:

- Automated Attendance Management
- Time Efficiency
- Attendance Report
- Accurate Data

2.3 Scope of the system Requirements

Functional Requirements of SAS:

- User Authentication

  Description: Users, including students and teachers, should be authenticated before accessing the attendance system.

  Features: Login credentials for each user. Secure authentication methods (e.g., username/password, biometric data).
- Attendance Recording

  Description: The system must accurately record attendance based on specified methods.

  Features: Automated attendance capture (e.g., facial recognition, RFID). Manual attendance entry for exceptional cases. Real-time recording during class sessions.

- Mobile Accessibility

  Description: Allow users to access the system via mobile devices.

  Features: Responsive design for mobile access. Mobile app for attendance tracking.
- User-Friendly Interface

  Description: Provide an intuitive and easy-to-use interface for all users.

  Features: Clear navigation and layout. Minimal training required for users.

Non-Functional Requirements of SAS:

- Usability:

  Description: Addresses the ease of use and user experience of the system.

  Requirements: The system should have an intuitive and user-friendly interface. Minimal training should be required for users to operate the system.
- Performance:

  Description: Specifies the system's responsiveness and efficiency under different conditions.

  Requirements: The system should be capable of handling simultaneous attendance recording for multiple classes. Response time for attendance capture should be within a specified limit (e.g., 2 seconds).
- Security:

  Description: Focuses on protecting data, preventing unauthorized access, and ensuring the integrity of the system.

  Requirements:Implementation of secure authentication methods (e.g., encryption, secure protocols). Access controls to restrict user permissions based on roles. Regular security audits and vulnerability assessments.
- Reliability:

  Description: Addresses the system's ability to perform consistently and reliably.

  Requirements: The system should have a high availability rate (e.g., 99.9%). Reliable facial recognition with minimal false positives/negatives.

2.4 System requirements

2.4.1 Software Requirements

- Technology & Tools
    - Python, Android, Firebase
    - Visual Studio Code
    - Android Studio
    - Web Browser Chrome

2.4.2 Hardware Requirements

- Processor: Intel/AMD dual-core or above
- RAM: 4 GB RAM
- Disk: 10 GB of SSD/HDD
- Operating system: Linux

2.4.3 Android system Requirements

- Android Version 12
- Processor: Mediatek Helio Octa-core
- RAM: 6 GB
- Storage: 64 GB

3. Requirement determination and Analysis

3.1 Fact Finding methods

- Interviews: Interviews involve face-to-face discussions with users, and subject matter experts to gather information about their needs and expectations. These interviews can be structured or unstructured and can be conducted in person or remotely.
- Observation: Observation involves watching users as they interact with the current system or with prototypes to gather information about their behavior, preferences, and pain points. This can be done in a lab or in the field, depending on the needs of the project.
- Document analysis: Document analysis involves reviewing existing documentation, such as user manuals and technical specifications, to identify requirements and constraints that may have been overlooked or not clearly communicated.
- Focus groups: Focus groups involve bringing together a small group of users to discuss their needs and expectations in a group setting. This can be used to gather both quantitative and qualitative data and to identify common themes and issues.
- Prototyping: Prototyping involves creating a simple, functional model of the website to test with users. This can be used to gather feedback and identify areas for improvement before the final system is developed.
- Brainstorming: Brainstorming involves generating ideas and solutions through a collaborative and creative process with subject matter experts. This can be used to identify new requirements and ideas that may not have been considered previously.

3.2 Feasibility study

- Technical Feasibility:

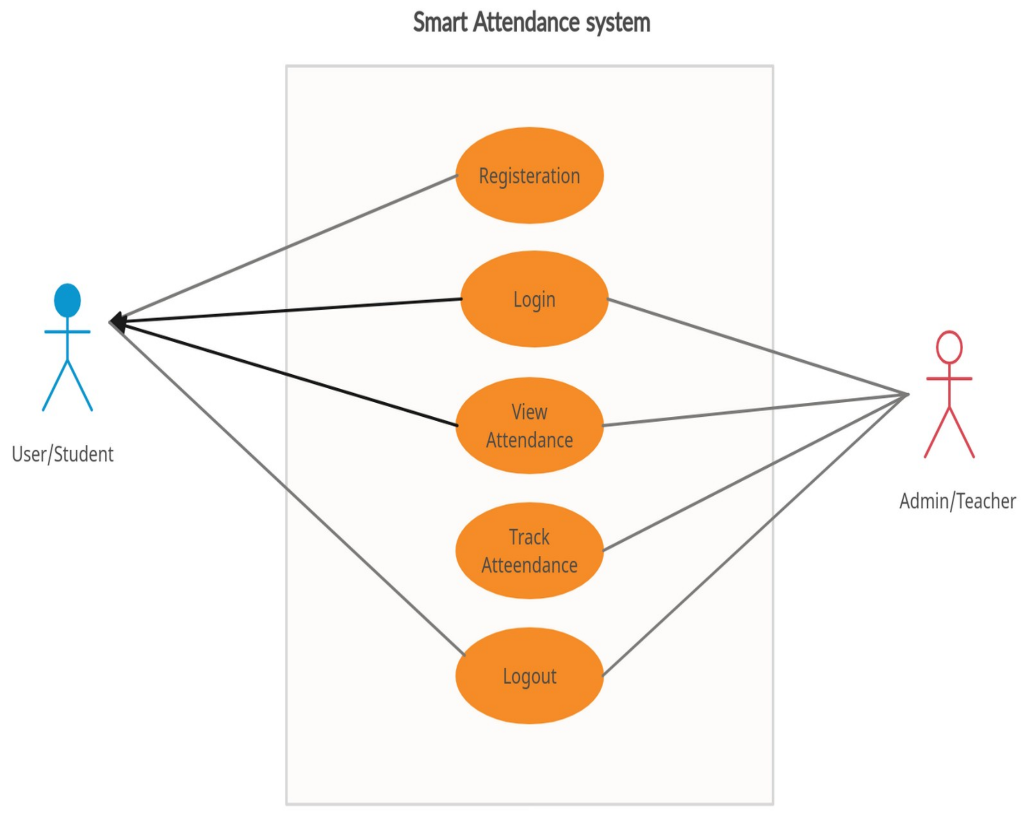
  - Availability of Technology: Assessing if the required hardware, software, and networking infrastructure needed for the Job-Link system are readily available or can be obtained within the project's constraints.
  - Skilled Resources: Determining if there are skilled developers, designers, and system administrators with the expertise needed to build and maintain the Job-Link system.
  - Integration Capability: Evaluating if the proposed system can integrate smoothly with existing systems, databases, and platforms used by job seekers and employers without significant technical difficulties.
  - Security and Privacy: Verifying that appropriate security measures can be implemented to protect user data, prevent unauthorized access, and maintain privacy and confidentiality within the Job-Link system.

- Operational Feasibility:

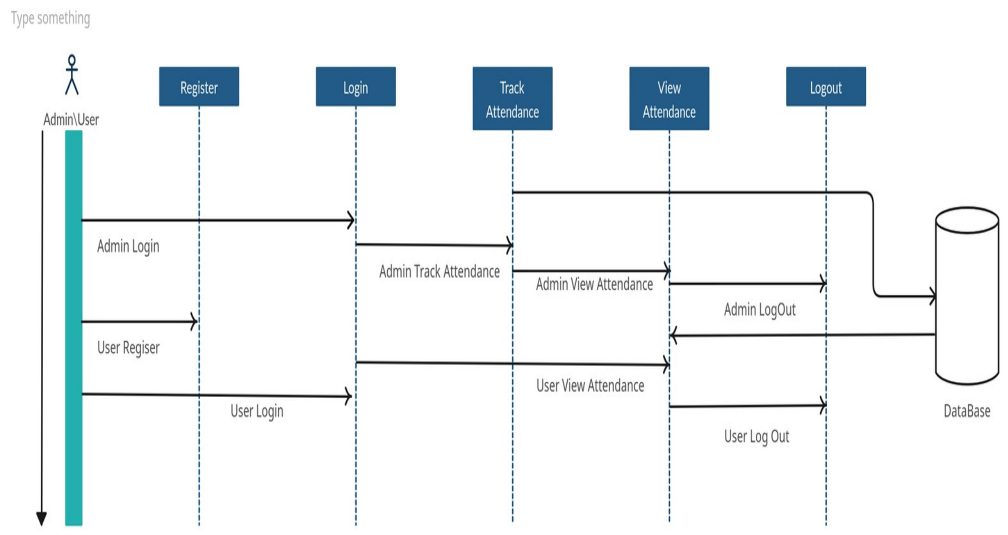
  - User Acceptance: Assessing the willingness and acceptance of potential users, including job seekers and employers, to adopt and use the Job-Link system effectively.
  - Organizational Impact: Evaluating the potential impact of implementing the system on the organization's processes, workflows, and resources, and assessing the organization's readiness for change.
  - Legal and Regulatory Compliance: Ensuring that the Job-Link system complies with applicable laws, regulations, and industry standards related to data protection, privacy, equal employment opportunity, and other relevant requirements.

4. System Analysis and Design

4.1 Use Case Diagram

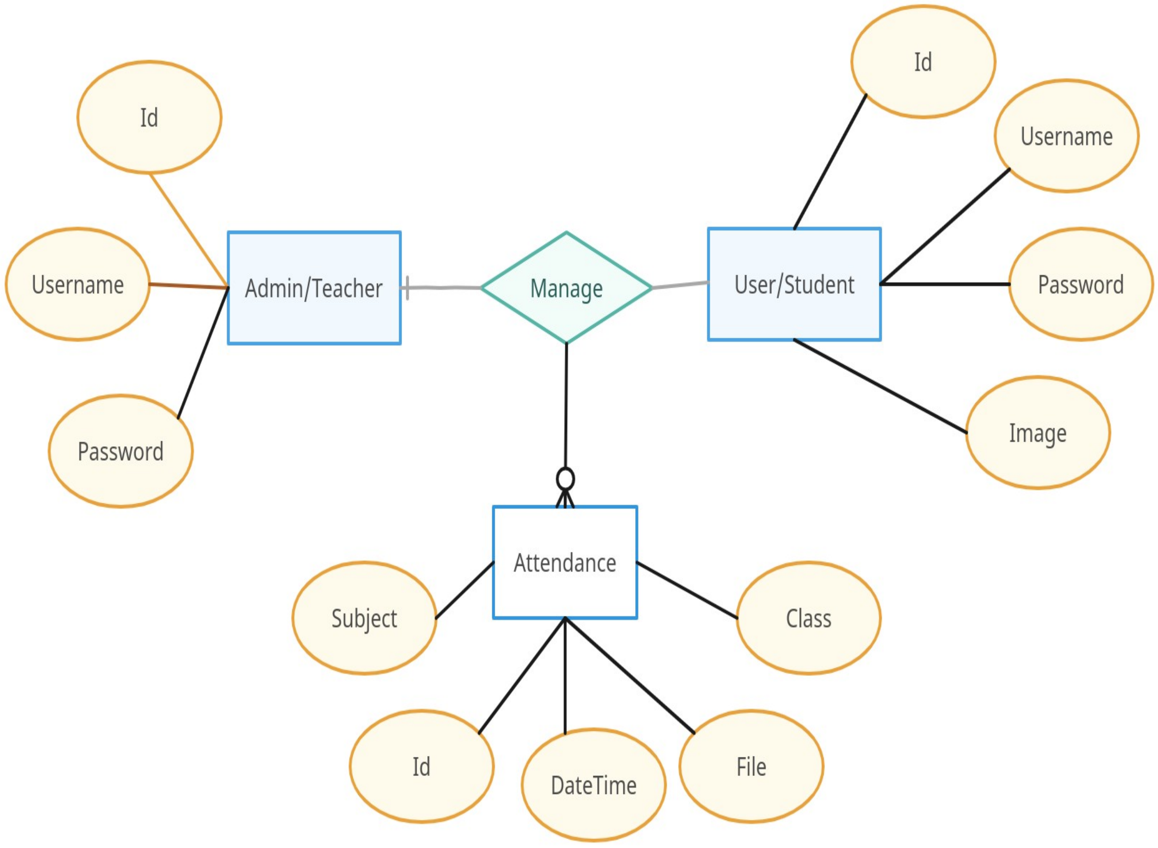


4.2 Sequence Diagram



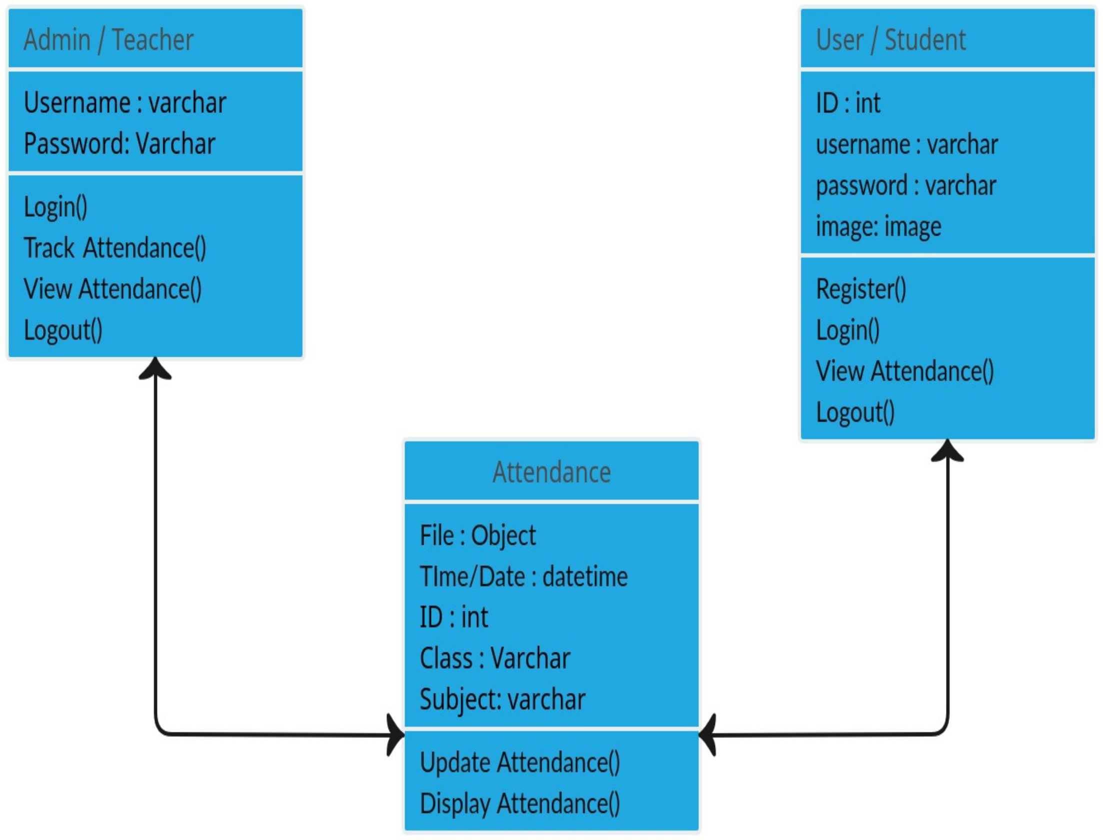
Smart Attendance System Sequence Diagram

4.3 ER-D

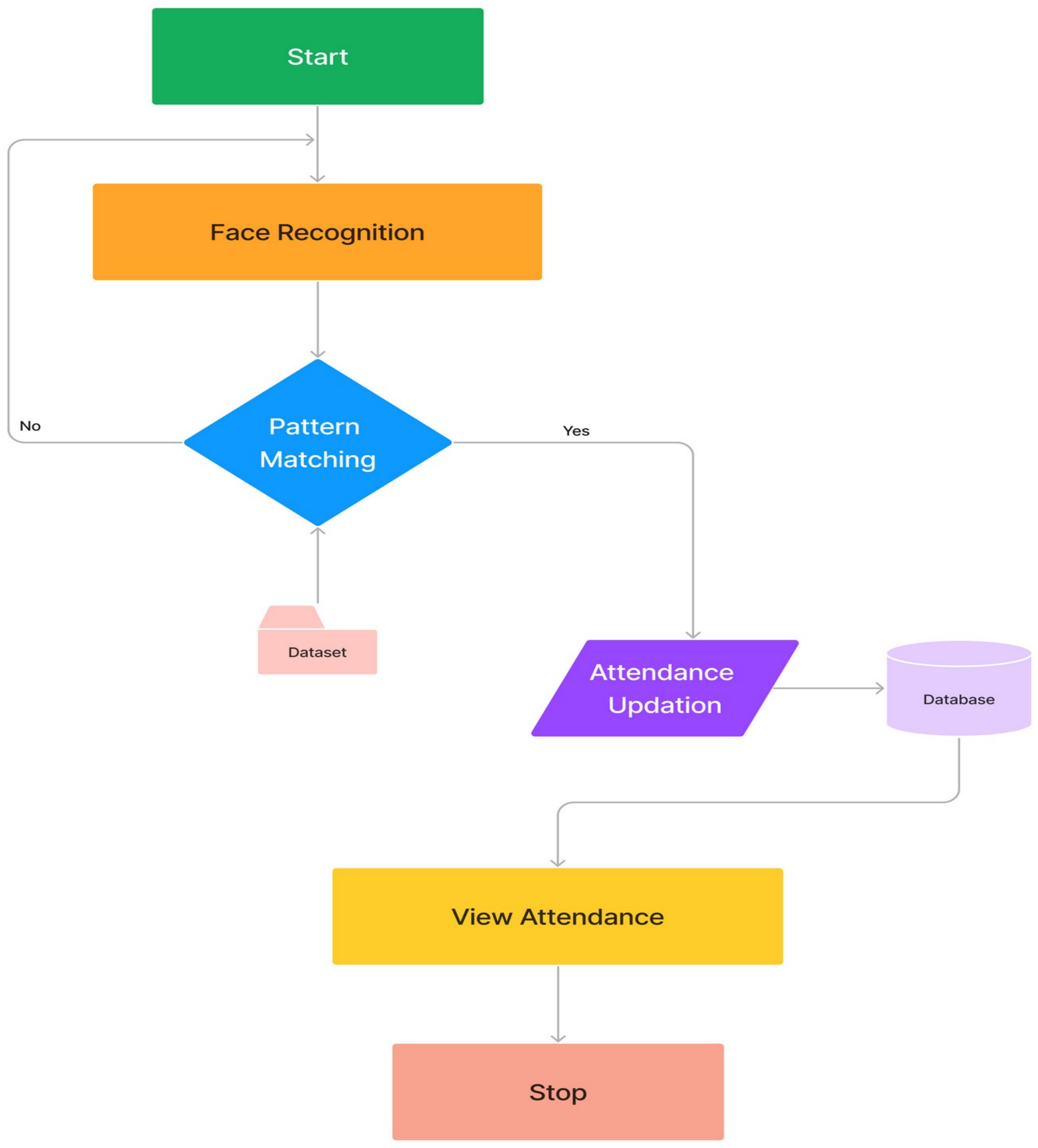


Smart Attendance system ERD

4.4 Class Diagram

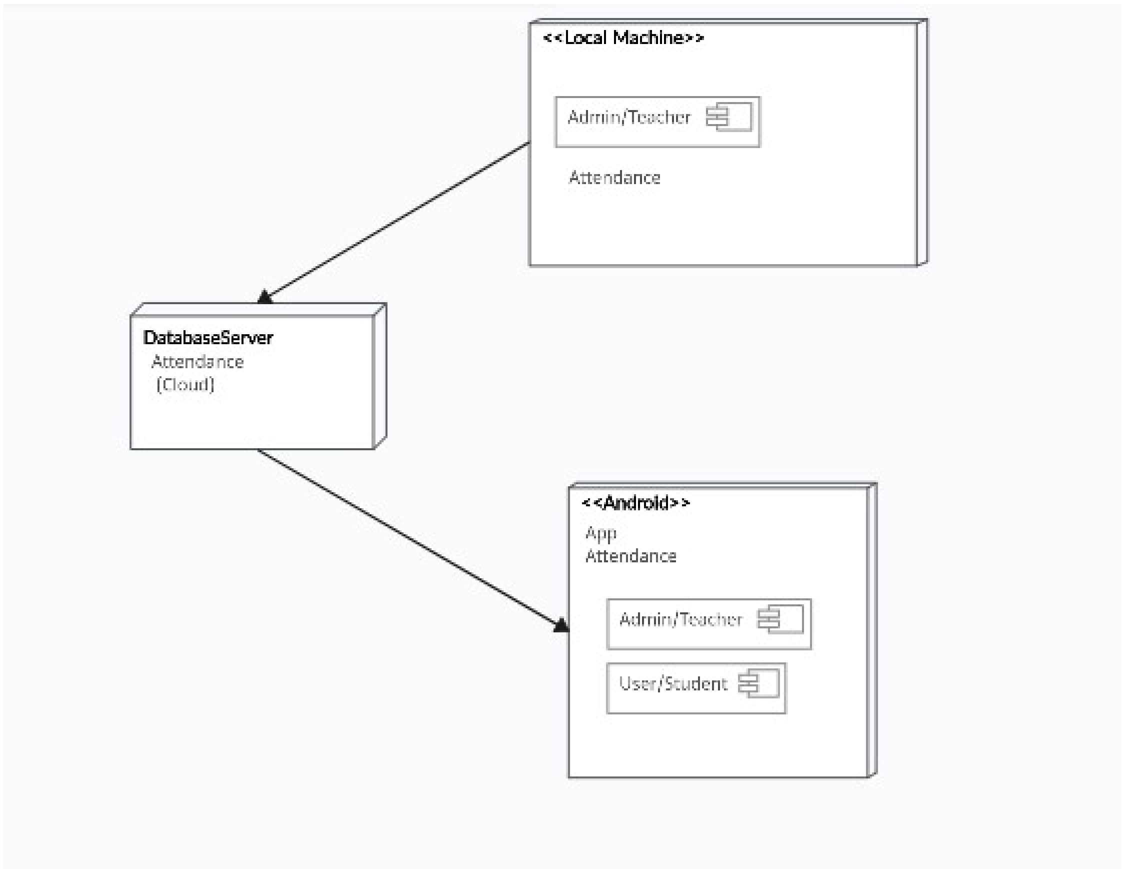


4.5 Flow Chart



Smart Attendance system Flowchart

4.5 Deployment diagram

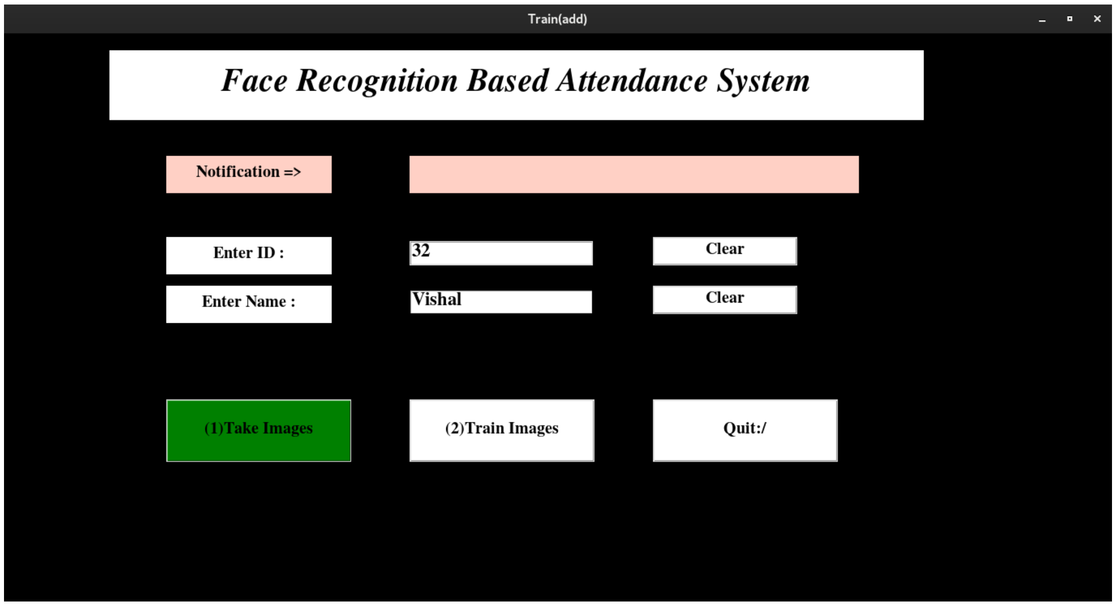


Smart Attendance system Deployment Diagram

4.7 User Interface Design (Screens)

*>>Python

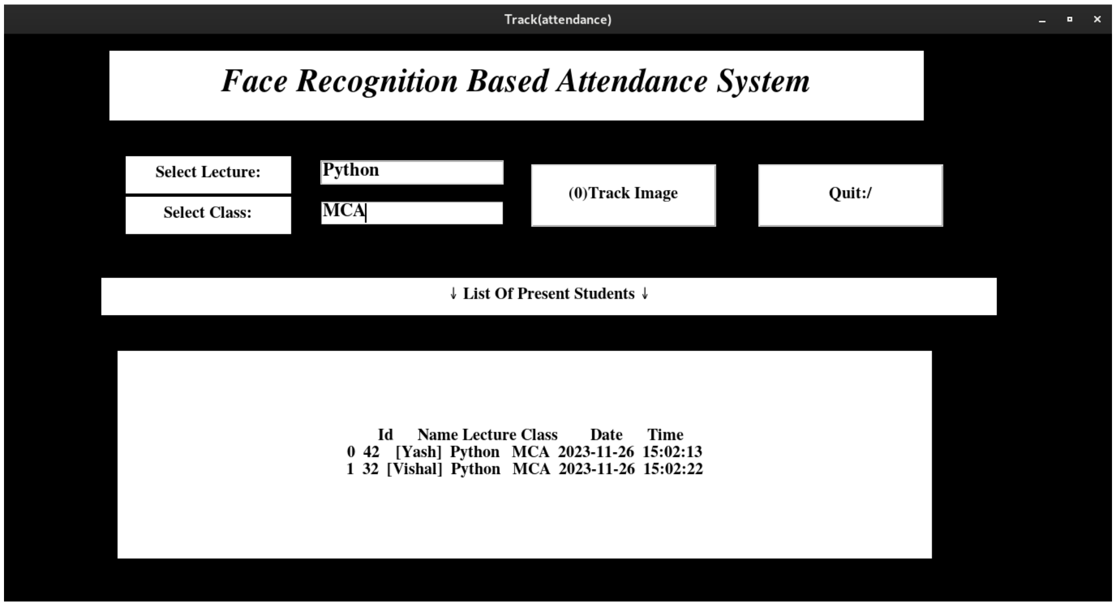
- Train images



Enter ID : 32

Enter Name : Vishal

- Track Attendance

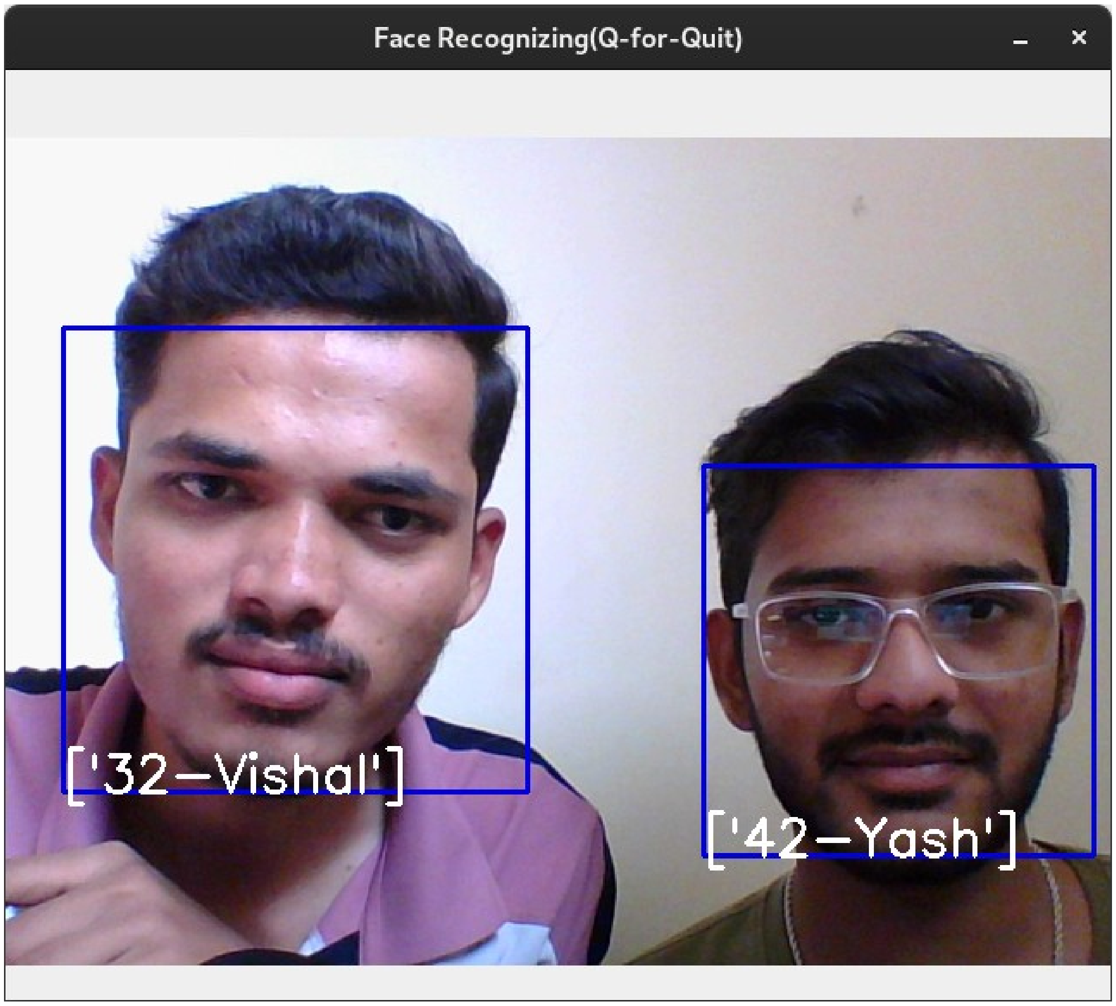

Track(attendance)
Face Recognition Based Attendance System
(0)Track Image
Quit:/
↓ List Of Present Students ↓

Select Lecture: Python

Select Class: MCA

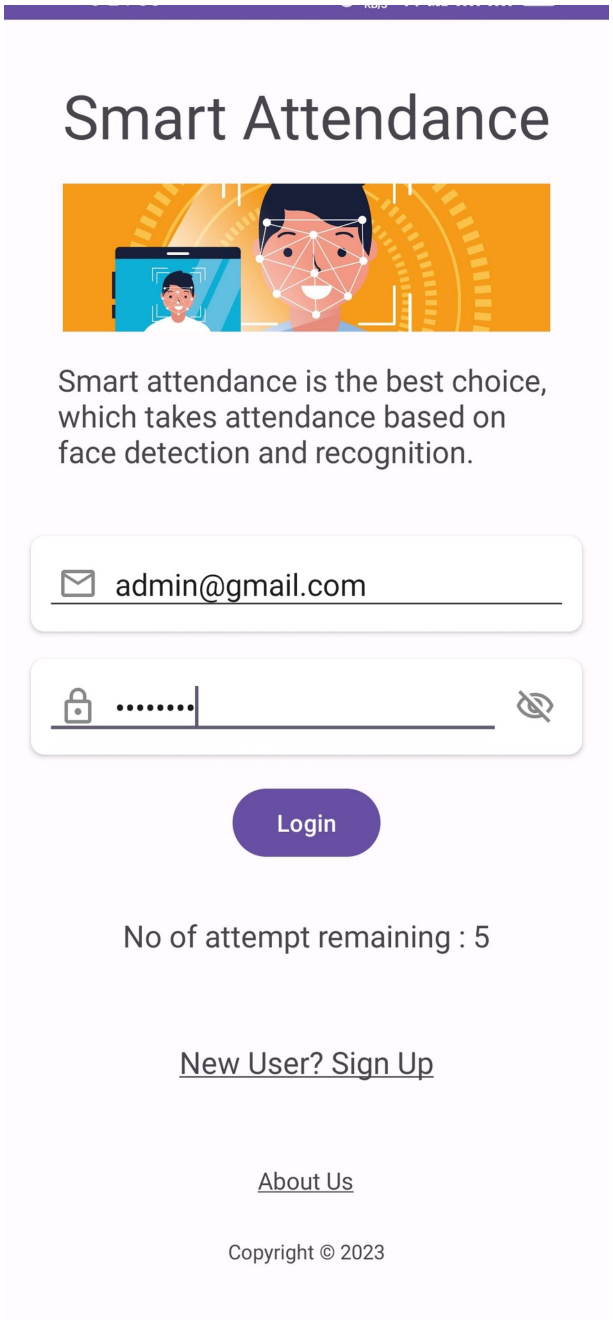
```
     Id     Name Lecture Class        Date      Time
0  42    [Yash]  Python   MCA  2023-11-26  15:02:13
1  32  [Vishal]  Python   MCA  2023-11-26  15:02:22
```

- Tracking



*>>Android Application

- Login



Smart Attendance

Smart attendance is the best choice, which takes attendance based on face detection and recognition.

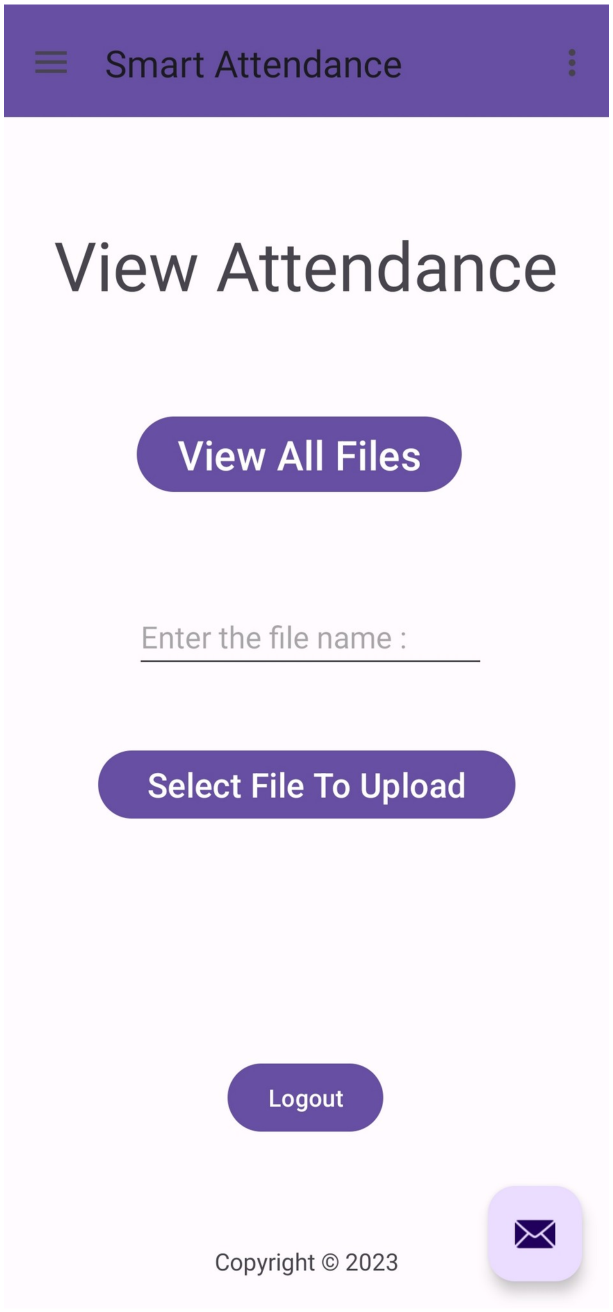
admin@gmail.com

••••••••

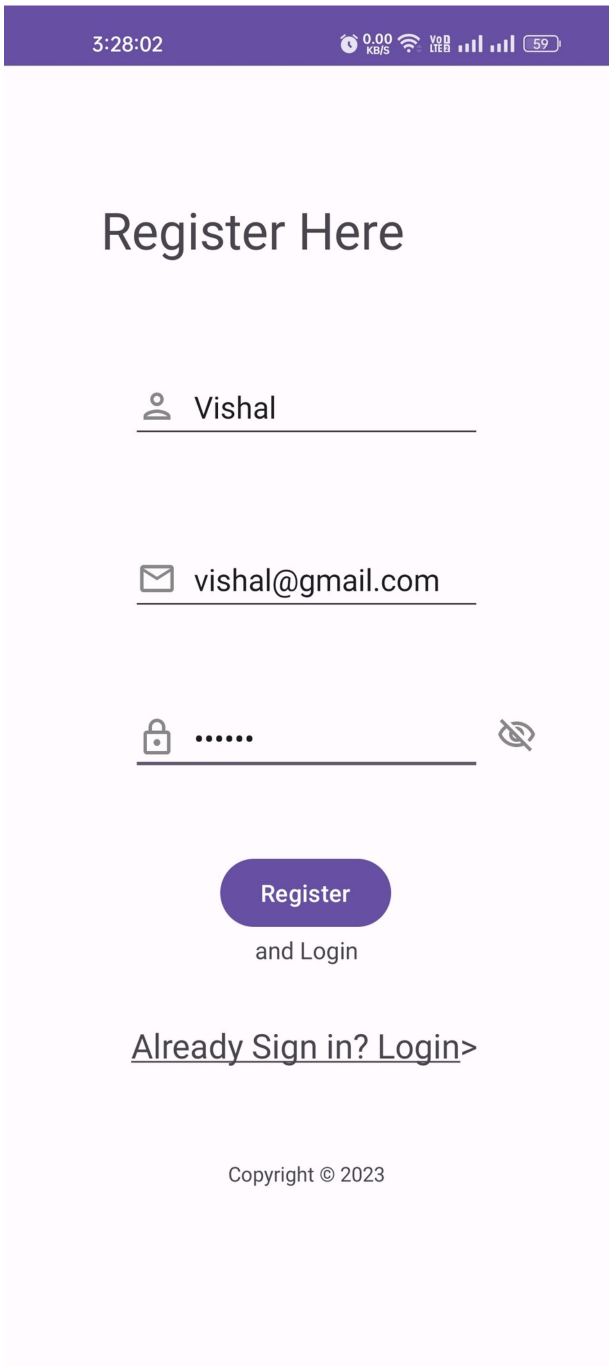
No of attempt remaining : 5

Copyright © 2023

- Dashboard


Smart Attendance
View Attendance
View All Files
Enter the file name :
Select File To Upload
Logout
Copyright © 2023

- Registration

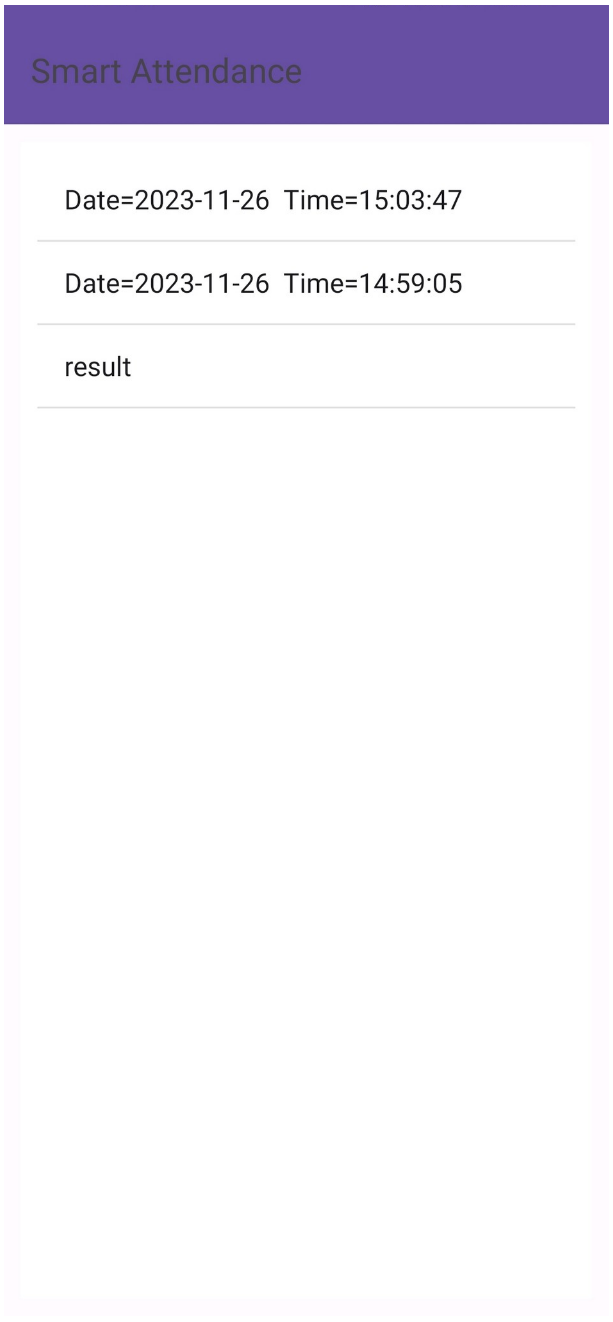

Register Here

Vishal

vishal@gmail.com

••••••

- View Attendance

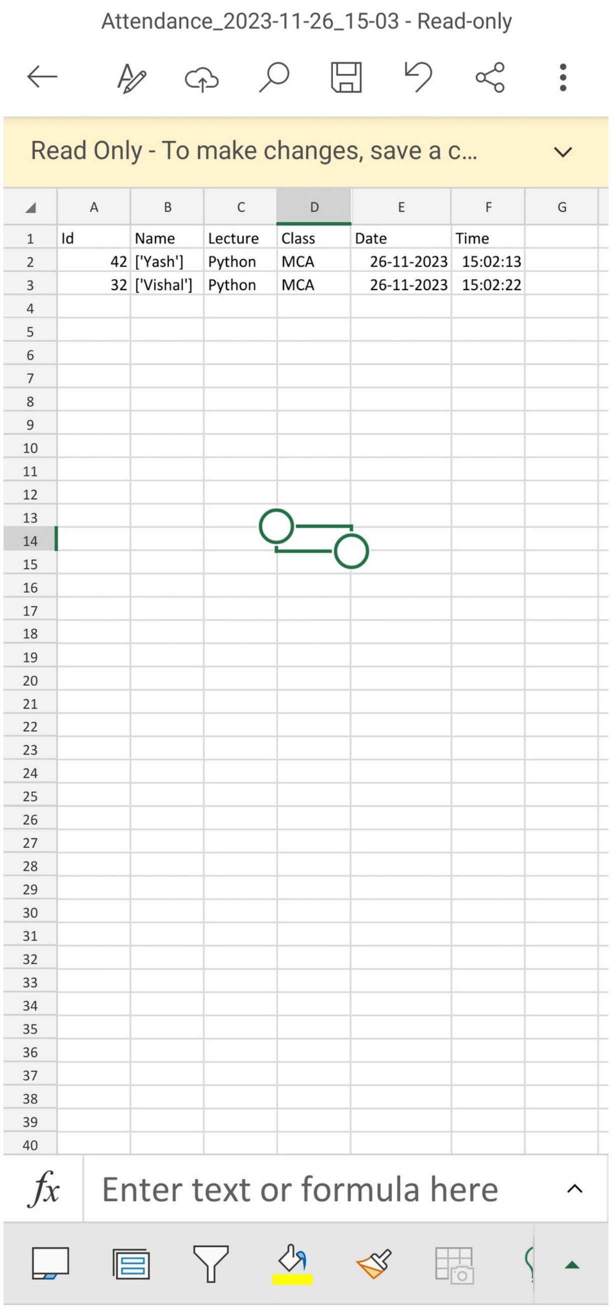


Date=2023-11-26 Time=15:03:47

Date=2023-11-26 Time=14:59:05

result

- Attendance



Attendance_2023-11-26_15-03 - Read-only

Read Only - To make changes, save a c...

	A	B	C	D	E	F	G
1	Id	Name	Lecture	Class	Date	Time	
2	42	['Yash']	Python	MCA	26-11-2023	15:02:13	
3	32	['Vishal']	Python	MCA	26-11-2023	15:02:22	

4.8 Database tables and structure

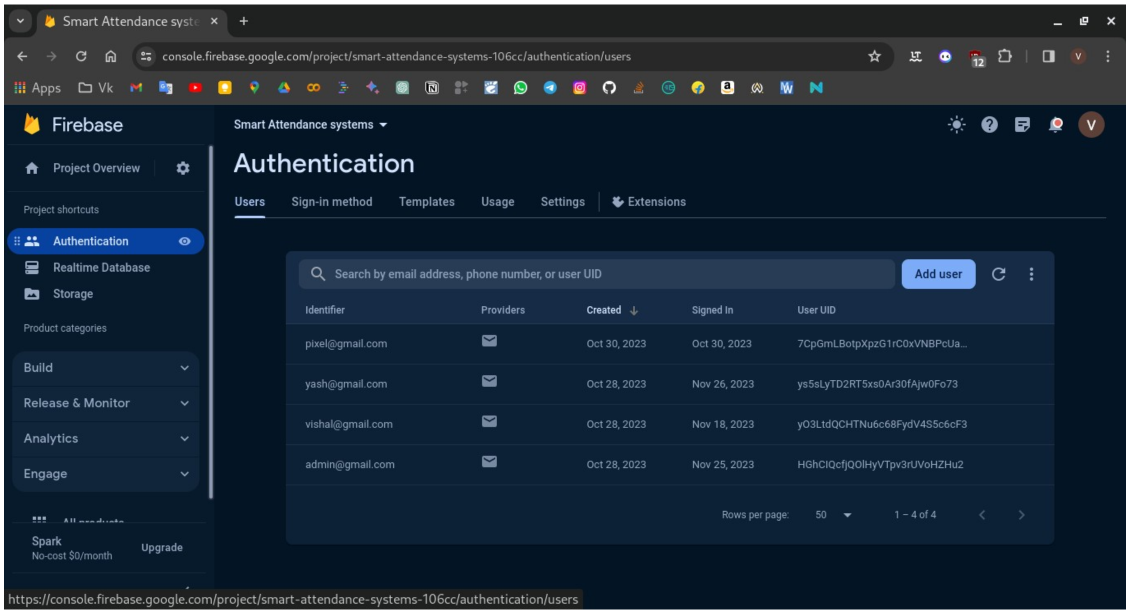
- User Table

Name	**Type**	**Key**	**Schema**	**Description**
email	varchar	Primary key	Not Null	Email id of User
username	varchar	-	Not Null	Username
password	varchar	-	Not Null	Password

- Attendance Table

Name	**Type**	**Key**	**Schema**	**Description**
id	int	Primary key	Not Null	Roll no of Student
name	varchar	-	Not Null	Name of Student
lecture	varchar	-	Not Null	Current Lecture
class	varchar	-	Not Null	Class of Students
date	date	-	Not Null	Current Date
time	time	-	Not Null	Current Time

- Users



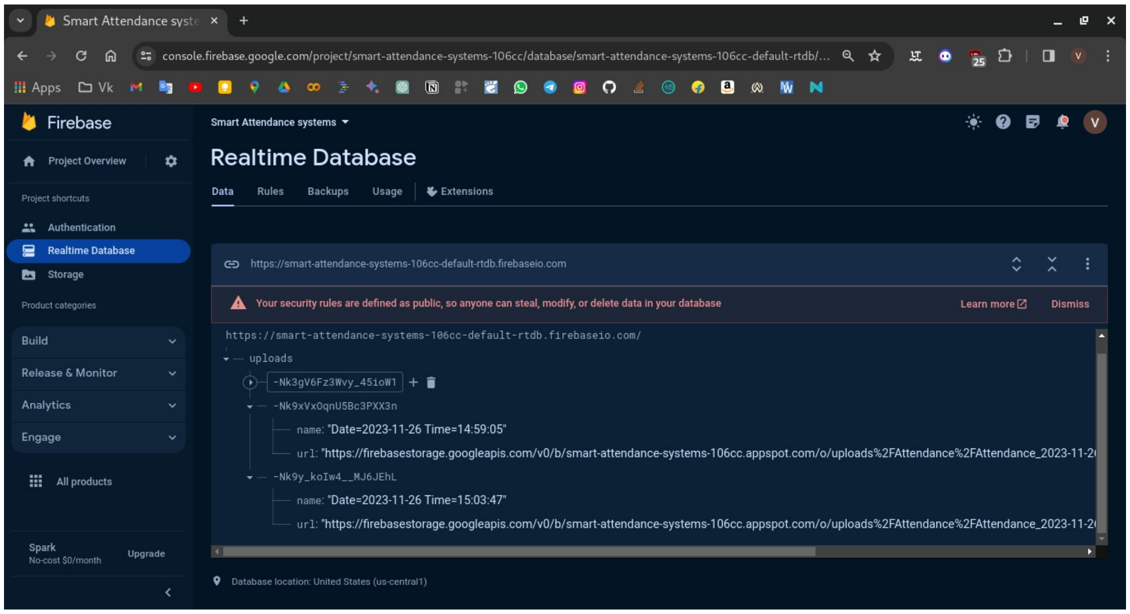
Authentication

Identifier	Providers	Created	Signed In	User UID
pixel@gmail.com	✉	Oct 30, 2023	Oct 30, 2023	7CpGmLBotpXpzG1rC0xVNBPcUa...
yash@gmail.com	✉	Oct 28, 2023	Nov 26, 2023	ys5sLyTD2RT5xs0Ar30fAjw0Fo73
vishal@gmail.com	✉	Oct 28, 2023	Nov 18, 2023	yO3LtdQCHTNu6c68FydV4S5c6cF3
admin@gmail.com	✉	Oct 28, 2023	Nov 25, 2023	HGhCIQcfjQOlHyVTpv3rUVoHZHu2

Rows per page: 50 1 – 4 of 4

https://console.firebase.google.com/project/smart-attendance-systems-106cc/authentication/users

- Cloud storage path and rtdb



Realtime Database

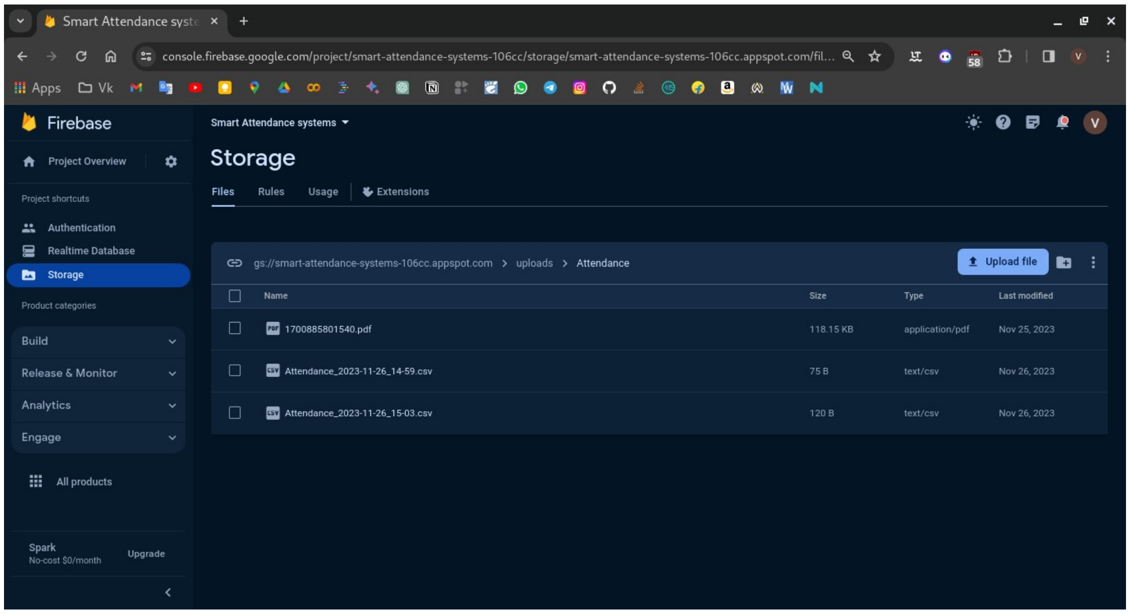
https://smart-attendance-systems-106cc-default-rtdb.firebaseio.com

Your security rules are defined as public, so anyone can steal, modify, or delete data in your database

```
https://smart-attendance-systems-106cc-default-rtdb.firebaseio.com/
uploads
    -Nk3gV6Fz3Wvy_45ioW1
    -Nk9xVxOqnU5Bc3PXX3n
        name: "Date=2023-11-26 Time=14:59:05"
        url: "https://firebasestorage.googleapis.com/v0/b/smart-attendance-systems-106cc.appspot.com/o/uploads%2FAttendance%2FAttendance_2023-11-2
    -Nk9y_koIw4__MJ6JEhL
        name: "Date=2023-11-26 Time=15:03:47"
        url: "https://firebasestorage.googleapis.com/v0/b/smart-attendance-systems-106cc.appspot.com/o/uploads%2FAttendance%2FAttendance_2023-11-2
```

Database location: United States (us-central1)

- Cloud Attendance storage



gs://smart-attendance-systems-106cc.appspot.com > uploads > Attendance

Name	Size	Type	Last modified
1700885801540.pdf	118.15 KB	application/pdf	Nov 25, 2023
Attendance_2023-11-26_14-59.csv	75 B	text/csv	Nov 26, 2023
Attendance_2023-11-26_15-03.csv	120 B	text/csv	Nov 26, 2023

5. Coding

The Coding phase of a Smart Attendance System involves implementing the necessary functionalities to collect attendance from a camera system, upload it to the cloud, and display the attendance list on a Smart Attendance App. Below is a high-level overview of the coding tasks involved in this phase:

1. Camera System Integration:

Task: Set up the camera system to capture images for attendance.

Steps: Use OpenCV or a similar library to interface with the camera.

Implement a function to capture images at regular intervals.

Integrate face detection algorithms to identify faces in the images.

2. Facial Recognition and Attendance Capture:

Task: Implement facial recognition to identify individuals and capture attendance.

Steps: Train a facial recognition model using a machine learning library (e.g., OpenCV).

Capture faces in real-time using the trained model.

Match detected faces with pre-registered student faces.

Record attendance for identified students.

3. Cloud Integration:

Task: Upload attendance data to the cloud for storage and retrieval.

Steps: Choose a cloud service provider (e.g., Firebase, AWS, Google Cloud).

Implement cloud storage to store attendance records.

Upload attendance data to the cloud after each capture session.

Ensure data security and compliance with privacy regulations.

4. Smart Attendance App Development:

Task: Develop a Smart Attendance App for users to view attendance data.

Steps: Choose a mobile app development framework (e.g., Flutter, React Native).

Create user authentication for secure access to attendance data.

Implement a dashboard to display attendance summaries.

Provide a detailed view of daily or class-wise attendance.

5. Real-time Updates and Notifications:

Task: Enable real-time updates for the Smart Attendance App.

Steps: Implement push notifications to inform users of new attendance data.

Update the app interface in real-time when new attendance records are available.

Ensure synchronization between the app and cloud-stored data.

6. User Interface (UI) Design:

Task: Design an intuitive and user-friendly interface for the Smart Attendance App.

Steps: Create screens for login, dashboard, and detailed attendance views.

Use appropriate visual elements to display attendance information.

Ensure responsiveness for different device screens.

7. Testing and Debugging:

Task: Test the entire system for functionality, security, and performance.

Steps: Conduct unit testing for individual components.

Perform integration testing to ensure seamless communication between components.

Debug and address any issues identified during testing.

8. Documentation:

Task: Document the codebase, APIs, and system architecture.

Steps: Generate API documentation for cloud interactions.

Provide inline comments in the code for better understanding.

Create a user manual for the Smart Attendance App.

9. Deployment:

Task: Deploy the Smart Attendance System for production use.

Steps: Choose a hosting environment for the camera system and cloud services.

Publish the Smart Attendance App on relevant app stores.

Monitor system performance in the production environment.

10. Maintenance and Updates:

Task: Plan for ongoing maintenance and updates.

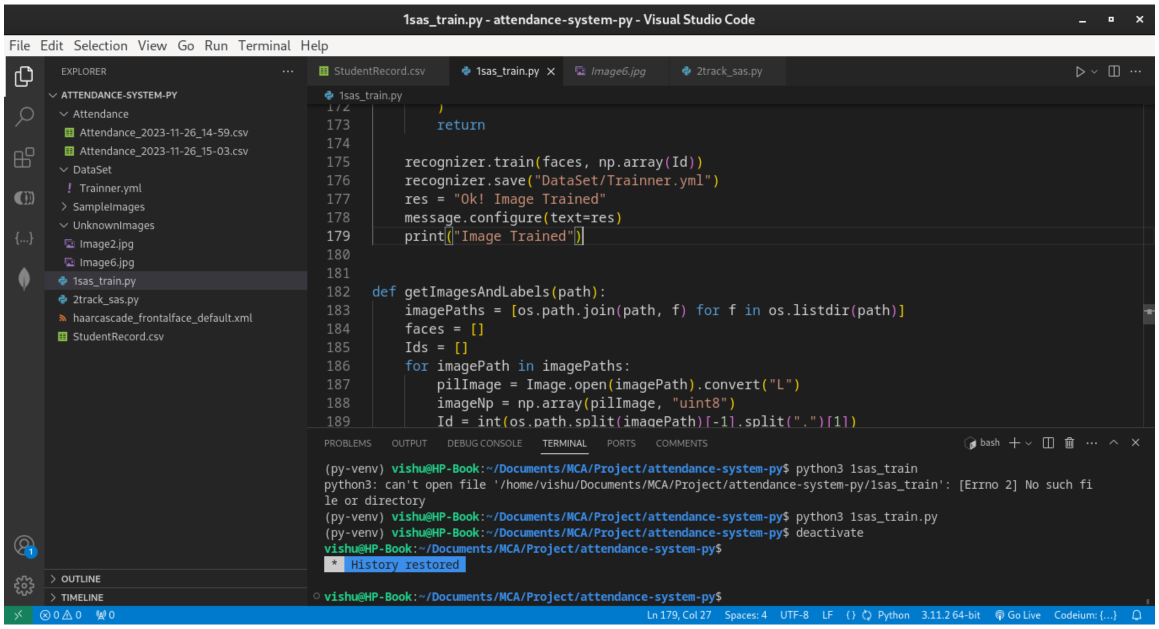
Steps: Establish a maintenance schedule for regular system checks.

Roll out updates to address security vulnerabilities or introduce new features.

Provide user support and address feedback.

// Coding Screen

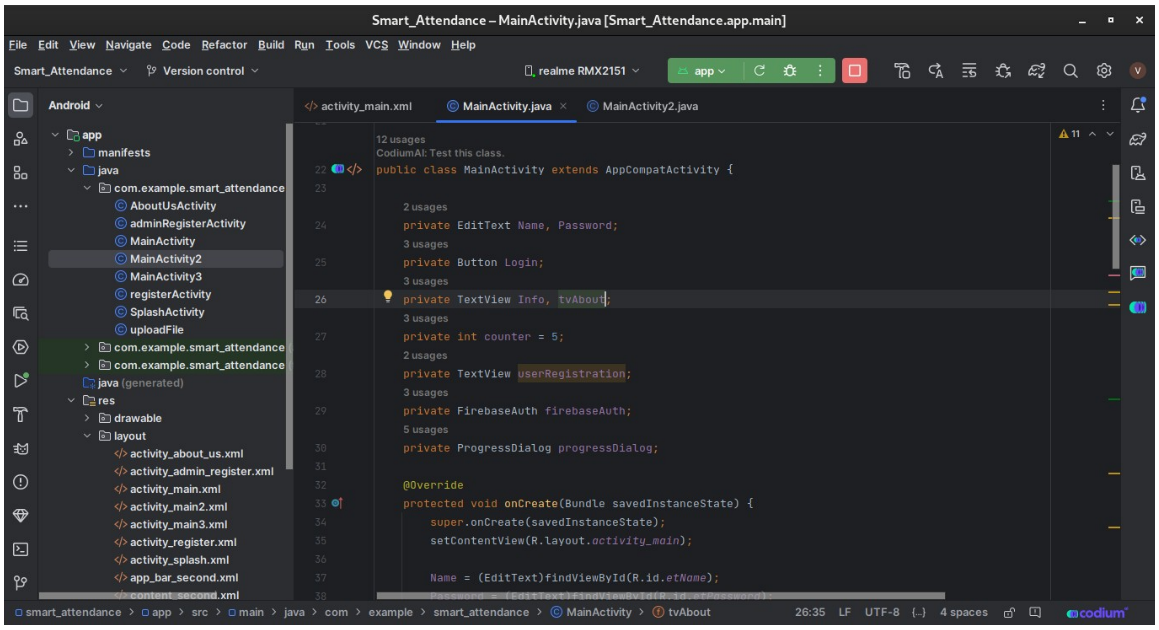
VS Code


1sas_train.py - attendance-system-py - Visual Studio Code
File Edit Selection View Go Run Terminal Help
EXPLORER
ATTENDANCE-SYSTEM-PY
Attendance
Attendance_2023-11-26_14-59.csv
Attendance_2023-11-26_15-03.csv
DataSet
Trainner.yml
SampleImages
UnknownImages
Image2.jpg
Image6.jpg
1sas_train.py
2track_sas.py
haarcascade_frontalface_default.xml
StudentRecord.csv
OUTLINE
TIMELINE
StudentRecord.csv
1sas_train.py
Image6.jpg
2track_sas.py
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS COMMENTS
bash
Ln 179, Col 27 Spaces: 4 UTF-8 LF {} Python 3.11.2 64-bit Go Live Codeium: {...}

```python
        )
            return

    recognizer.train(faces, np.array(Id))
    recognizer.save("DataSet/Trainner.yml")
    res = "Ok! Image Trained"
    message.configure(text=res)
    print("Image Trained")


def getImagesAndLabels(path):
    imagePaths = [os.path.join(path, f) for f in os.listdir(path)]
    faces = []
    Ids = []
    for imagePath in imagePaths:
        pilImage = Image.open(imagePath).convert("L")
        imageNp = np.array(pilImage, "uint8")
        Id = int(os.path.split(imagePath)[-1].split(".")[1])
```

```
(py-venv) vishu@HP-Book:~/Documents/MCA/Project/attendance-system-py$ python3 1sas_train
python3: can't open file '/home/vishu/Documents/MCA/Project/attendance-system-py/1sas_train': [Errno 2] No such fi
le or directory
(py-venv) vishu@HP-Book:~/Documents/MCA/Project/attendance-system-py$ python3 1sas_train.py
(py-venv) vishu@HP-Book:~/Documents/MCA/Project/attendance-system-py$ deactivate
vishu@HP-Book:~/Documents/MCA/Project/attendance-system-py$
 *  History restored

vishu@HP-Book:~/Documents/MCA/Project/attendance-system-py$
```

Android Studio



```java
12 usages
CodiumAI: Test this class.
public class MainActivity extends AppCompatActivity {

    2 usages
    private EditText Name, Password;
    3 usages
    private Button Login;
    3 usages
    private TextView Info, tvAbout;
    3 usages
    private int counter = 5;
    2 usages
    private TextView userRegistration;
    3 usages
    private FirebaseAuth firebaseAuth;
    5 usages
    private ProgressDialog progressDialog;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        Name = (EditText)findViewById(R.id.etName);
        Password = (EditText)findViewById(R.id.etPassword);
```

6. Testing

The Testing phase of a Smart Attendance System is crucial to ensure that the implemented functionalities work correctly, meet the specified requirements, and deliver a reliable user experience. Below is an overview of the testing tasks involved in this phase:

1. Unit Testing:
Objective: Verify the correctness of individual components and functions.
Tasks: Test facial recognition algorithms for accuracy.
Verify camera system integration.
Test cloud integration functions independently.

2. Integration Testing:
Objective: Ensure that different system components work seamlessly together.
Tasks: Test the interaction between the camera system and facial recognition.
Verify the flow of data between the Smart Attendance App and the cloud.
Confirm that real-time updates are reflected in the app.

3. Security Testing:
Objective: Identify and address potential security vulnerabilities.
Tasks: Conduct penetration testing to assess system security.
Verify secure communication between the app and cloud.
Ensure user authentication and authorization mechanisms are robust.

4. Performance Testing:
Objective: Assess system performance under different conditions.
Tasks: Test the system's ability to handle concurrent user logins.
Measure response times for attendance capture and retrieval.
Assess the impact of varying network conditions on app performance.

5. User Acceptance Testing (UAT):
Objective: Ensure that the Smart Attendance App meets user expectations.
Tasks: Engage end-users to test the app's functionality.
Collect feedback on the user interface and overall user experience.
Confirm that the app aligns with user requirements.

6. Regression Testing:
Objective: Ensure that new updates or bug fixes do not introduce new issues.
Tasks: Re-run previously conducted tests after each code update.
Confirm that existing functionalities continue to work as intended.
Address any regressions promptly.

7. Usability Testing:
Objective: Evaluate the user-friendliness of the Smart Attendance App.
Tasks: Assess the intuitiveness of the app's navigation.
Gather user feedback on the clarity of attendance displays.
Identify and address any usability issues.

8. Compatibility Testing:
Objective: Verify that the Smart Attendance App works across different devices and platforms.
Tasks: Test the app on various mobile devices (iOS and Android).
Verify compatibility with different screen sizes and resolutions.

9. Bug Tracking and Resolution:
Objective: Identify and resolve any bugs or issues discovered during testing.
Tasks: Use a bug tracking system to log and prioritize identified issues.
Address and fix bugs promptly to maintain system integrity.

10. User Training:
Objective: Provide training materials and support for end-users.
Tasks: Develop training materials, tutorials, or FAQs.
Conduct training sessions for users on how to use the Smart Attendance App.

- **Testing Table**

Test Case	**Data Input**	**Expected Output**	**Actual Output**	**Pass/Fail**
1	Valid credentials for Login	Successful login	Successful login	Pass
2	Valid Details for Registration	Successful Register & Login	Successful Register & Login	Pass
3	Invalid credentials	Error message displayed	Error message displayed	Pass
4	Empty ‘username’ field	Error message displayed	Error message displayed	Pass
5	Empty ‘email’ field	Error message displayed	Error message displayed	Pass
6	Empty ‘password’ field	Error message displayed	Error message displayed	Pass
7	Empty ‘File Name’ field	Error message displayed	Error message displayed	Pass
8	Password Wrong 5 times	Disable Login button	Disable Login button	Pass
9	Training images not Provided	Error message displayed	Error message displayed	Pass
10	Tracking details not Provided	Error message displayed	Error message displayed	Pass

7. Limitations and Enhancements

7.1 Limitations of the Smart Attendance system:

1. Accuracy Challenges:

   Facial recognition systems may exhibit errors, leading to false positives or negatives, impacting the overall accuracy of identification.

2. Environmental Factors:

   Variations in lighting conditions and camera quality can affect the system's performance, influencing its ability to accurately recognize faces.

3. Subject Variability:

   Changes in a person's appearance due to aging, alterations in hairstyle, or the use of accessories may introduce variability in recognition.

4. Privacy and Legal Concerns:

   Storing and handling biometric data raises privacy and security concerns, necessitating compliance with legal regulations and obtaining user consent.

5. Deployment Challenges:

   Implementing robust facial recognition systems may incur high costs, and regular maintenance is essential for sustained accuracy and reliability.

7.2 Future Enhancements to the Smart Attendance system:

1. Integration with Biometric Technologies:

   Explore integration with other biometric modalities, such as fingerprint recognition or iris scanning, to provide additional layers of identification and increase accuracy

2. Automatic Notifications:

   Implement automatic notifications to alert students, teachers, and administrators about attendance-related events, such as low attendance, late arrivals, or important announcements.

3. Enhanced Security Measures:

   Implement advanced security measures to prevent unauthorized access, such as multi-factor authentication, encryption of biometric data, and regular security audits.

4. Feedback Mechanism:

   Implement a feedback mechanism where users can report issues, provide suggestions, or express concerns. This can help in continuous improvement and user satisfaction.

8. Conclusion

The Smart Attendance System (SAS) brings a modern twist to traditional attendance-taking methods by using facial recognition technology. It's a promising system that automates attendance tracking, allowing real-time monitoring and tighter security measures.

However, there are challenges. Sometimes, the system may not recognize faces accurately, leading to mistakes in identifying people. Factors like varying lighting conditions or the quality of the camera used can impact how well it works. Privacy is another concern as storing facial data needs to be done securely and with user consent.

Despite these challenges, SAS offers a new way to manage attendance efficiently. It could potentially streamline processes, save time, and ensure better resource management in schools or companies. But for it to succeed, ongoing improvements and ensuring it meets privacy and security regulations will be essential. Overall, SAS holds promise but needs careful attention to its accuracy, user consent, and security measures for wider adoption and success.

9. Bibliography

- Google

  https://www.google.com
- YouTube

  https://www.youtube.com
- Python

  https://docs.python.org/3.10/
- Stack overflow

  https://stackoverflow.com
- w3schools

  https://www.w3schools.com
- Android

  https://developer.android.com/
- AI

  https://chat.openai.com/

10. Annexure

• Sample Code(train.py)

```
import tkinter as tk
from tkinter import Message, Text, messagebox
import cv2, os
import csv
import numpy as np
from PIL import Image, ImageTk
import tkinter.font as font

if (
    not os.path.exists("Attendance")
    or not os.path.exists("DataSet")
    or not os.path.exists("SampleImages")
    or not os.path.exists("UnknownImages")
):
    os.makedirs("Attendance")
    os.makedirs("DataSet")
    os.makedirs("SampleImages")
    os.makedirs("UnknownImages")

window = tk.Tk()
window.title("Train(add)")
window.configure(background="black")
window.geometry("1280x670")

lbl = tk.Label(
    window,
    text="Face Recognition Based Attendance System",
    bg="white",
    fg="black",
    width=50,
    height=2,
    font=("times", 30, "italic bold"),
)
lbl.place(x=130, y=20)

lbl1 = tk.Label(
    window,
    text="Enter ID :",
    width=20,
    height=2,
    fg="black",
    bg="white",
    font=("times", 15, " bold "),
)
lbl1.place(x=200, y=250)

txt1 = tk.Entry(window, width=20, bg="white", fg="black", font=("times", 17, "
bold "))
txt1.insert(0, "Enter ID here")
txt1.place(x=500, y=255)

lbl2 = tk.Label(
    window,
    text="Enter Name :",
    width=20,
    fg="black",
    bg="white",
    height=2,
    font=("times", 15, " bold "),
)
lbl2.place(x=200, y=310)

txt2 = tk.Entry(window, width=20, bg="white", fg="black", font=("times", 17, "
bold "))
txt2.insert(0, "Enter Name here")
txt2.place(x=500, y=315)

lbl3 = tk.Label(
    window,
    text="Notification =>",
    width=20,
    fg="black",
    bg="#FFD0C5",
    height=2,
    font=("times", 15, " bold "),
)
lbl3.place(x=200, y=150)

message = tk.Label(
    window,
    text="",
    bg="#FFD0C5",
    fg="black",
    width=55,
    height=2,
    font=("times", 15, " bold "),
)

message.place(x=500, y=150)


def clearId(event):
    if txt1.get() == "Enter ID here":
        txt1.delete(0, "end")


def clearName(event):
    if txt2.get() == "Enter Name here":
        txt2.delete(0, "end")


txt1.bind("<FocusIn>", clearId)
txt2.bind("<FocusIn>", clearName)


def clearId():
    txt1.delete(0, "end")


def clearName():
    txt2.delete(0, "end")


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


def takeImages():
    Id = txt1.get()
    name = txt2.get()

    if isNumber(Id) and name.isalpha():
        cam = cv2.VideoCapture(0)
        harcascadePath = "haarcascade_frontalface_default.xml"
        detector = cv2.CascadeClassifier(harcascadePath)
        sampleNum = 0
        while True:
            ret, img = cam.read()
            gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
            faces = detector.detectMultiScale(gray, 1.3, 5)
            for x, y, w, h in faces:
                cv2.rectangle(img, (x, y), (x + w, y + h), (255, 0, 0), 2)
                sampleNum = sampleNum + 1
                cv2.imwrite(
                    "SampleImages/ " + name + "." + Id + "." + str(sampleNum) + ".jpg",
                    gray[y : y + h, x : x + w],
                )
                cv2.imshow("Face Detecting(Q-for-Quit)", img)
            if cv2.waitKey(100) & 0xFF == ord("q"):
                break
            elif sampleNum > 60:
                break
        cam.release()
        cv2.destroyAllWindows()
        res = "Images Saved for ID : " + Id + " Name : " + name
        row = [Id, name]
        with open("StudentRecord.csv", "a+") as csvFile:
            writer = csv.writer(csvFile)
            writer.writerow(row)
        csvFile.close()
        message.configure(text=res)
    else:
        if isNumber(name):
            res = "Enter Alphabetical Name"
            message.configure(text=res)
        if Id.isalpha():
            res = "Enter Numeric Id"
            message.configure(text=res)


def trainImages():
    recognizer = cv2.face.LBPHFaceRecognizer_create()
    harcascadePath = "haarcascade_frontalface_default.xml"
    detector = cv2.CascadeClassifier(harcascadePath)
    faces, Id = getImagesAndLabels("SampleImages")

    if not faces or not Id:
        messagebox.showwarning(
            "Warning", "No training data available. Capture images first."
        )
        return

    recognizer.train(faces, np.array(Id))
    recognizer.save("DataSet/Trainner.yml")
    res = "Ok! Image Trained"
    message.configure(text=res)


def getImagesAndLabels(path):
    imagePaths = [os.path.join(path, f) for f in os.listdir(path)]
    faces = []
    Ids = []
    for imagePath in imagePaths:
        pilImage = Image.open(imagePath).convert("L")
        imageNp = np.array(pilImage, "uint8")
        Id = int(os.path.split(imagePath)[-1].split(".")[1])
        faces.append(imageNp)
        Ids.append(Id)
    return faces, Ids


clearButton1 = tk.Button(
    window,
    text="Clear",
    command=clearId,
    fg="black",
    bg="white",
    width=15,
    height=1,
    activebackground="Red",
    font=("times", 15, " bold "),
)
clearButton1.place(x=800, y=250)

clearButton2 = tk.Button(
    window,
    text="Clear",
    command=clearName,
    fg="black",
    bg="white",
    width=15,
    height=1,
    activebackground="Red",
    font=("times", 15, " bold "),
)
clearButton2.place(x=800, y=310)

takeImg = tk.Button(
    window,
    text="(1)Take Images",
    command=takeImages,
    fg="black",
    bg="white",
    width=20,
    height=3,
    activebackground="Green",
    font=("times", 15, " bold "),
)
takeImg.place(x=200, y=450)

trainImg = tk.Button(
    window,
    text="(2)Train Images",
    command=trainImages,
    fg="black",
    bg="white",
    width=20,
    height=3,
    activebackground="Green",
    font=("times", 15, " bold "),
)
trainImg.place(x=500, y=450)

quitWindow = tk.Button(
    window,
    text="Quit:/",
    command=window.destroy,
    fg="black",
    bg="white",
    width=20,
    height=3,
    activebackground="Red",
    font=("times", 15, " bold "),
)
quitWindow.place(x=800, y=450)

window.mainloop()
```