

**University of Lucknow**



**Goel Institute of Higher Studies**

**Mahavidyalaya**

**College Code: 164**

**Major Project Report**

**on**

“Automated Attendance System using Deep Learning”

**UNDER GUIDANCE OF SUBMITTED BY**

**Aman Singh**

201640040003

**Dr. Virendra Singh**

Assistant Professor, Project Guide,

Department of Computer Science,

G.I.H.S. Mahavidyalaya, Lucknow

Contents

[Introduction 2](#_Toc141288976)

[REQUIREMENTS & CONFIGURATION 6](#_Toc141288977)

[TOOL’S USED 8](#_Toc141288978)

[SYSTEM ANALYSIS 10](#_Toc141288979)

[FEASIBILITY STUDY 11](#_Toc141288980)

[CONCEPTUAL MODEL 13](#_Toc141288981)

[SYSTEM DESIGN 18](#_Toc141288982)

[TESTING 20](#_Toc141288983)

[SNAPSHOTS & CODING 24](#_Toc141288984)

[CONCLUSION 39](#_Toc141288985)

[Limitation of the system 39](#_Toc141288986)

[Future Scope of the Project 40](#_Toc141288987)

[BIBLIOGRAPHY 42](#_Toc141288988)

# 

# Introduction

The Face Recognition Attendance System is a sophisticated use of computer vision technology that provides a cutting-edge and automated method of managing attendance. Traditional attendance systems sometimes rely on labor-intensive, error-prone, and inefficient manual techniques like sign-in sheets or card swiping. The Face Recognition Attendance System, on the other hand, makes use of sophisticated algorithms and face recognition methods to precisely identify people and mark their attendance in real-time.

This system's main goal is to automate attendance tracking, making it more practical, dependable, and effective. The system can uniquely identify people based on their facial traits and link them with pre-registered data by utilizing the capabilities of face recognition technology. Physical attendance registers or cards are no longer required, which reduces the administrative effort and the risk of mistakes.

The system employs HAAR cascades, a machine learning-based object detection algorithm, to detect and locate human faces in images or video frames. It analyzes the detected faces using face recognition algorithms, which extract unique facial features and compare them with stored data to identify individuals accurately. The process is fast, non-intrusive, and highly accurate, allowing for swift and efficient attendance marking.

The Face Recognition Attendance System finds applications in various domains, including educational institutions, corporate environments, and events management. In educational settings, it simplifies the attendance process for both students and teachers, providing accurate records and saving valuable class time. In workplaces, it streamlines attendance management, eliminates buddy punching, and enables seamless integration with payroll systems. Additionally, the system ensures precise attendance tracking and enhances security measures in event management.

By implementing the Face Recognition Attendance System, organizations can significantly improve attendance management, reduce administrative workload, and enhance overall efficiency. It eliminates the possibility of fraudulent attendance, provides real-time attendance reports, and allows for easy integration with existing systems.

The goal of this project report is to give readers a thorough understanding of how the Face Recognition Attendance System was created and put into use. Data collection, preprocessing, HAAR cascade training, face detection, feature extraction, face recognition, attendance marking, and user interface design will all be covered in the methods section. The report will also go through the advantages of the system, difficulties encountered during implementation, and prospective directions for future improvements.

Background

Management of attendance is essential in many industries, including education, business, and event planning. Traditionally, sign-in papers, roll calls, or swipe cards have been used to manually record attendance. These techniques, however, take a lot of effort, are prone to mistakes, and are simple to abuse. There has been an increasing demand for more effective and trustworthy attendance management solutions as technology has developed.

To get beyond the restrictions of conventional attendance systems, face recognition technology has become a potent tool. To analyse and identify people’s distinctive facial traits, it makes use of machine learning and computer vision technologies. By utilizing this technology, attendance may be tracked automatically and accurately, minimizing administrative work and improving the process of managing attendance overall.

Since several years ago, the idea of facial recognition has been extensively investigated and developed. The HAAR cascade, developed by Paul Viola and Michael Jones in 2001, is one of the fundamental algorithms used in face identification. In a machine learning method known as HAAR cascades, a classifier is trained to recognize particular object attributes, such as faces, in pictures or video frames. HAAR cascades are a common option for face identification in a variety of applications, including attendance systems. They have demonstrated extraordinary effectiveness in detecting faces.

The accuracy and resilience of facial recognition algorithms have been considerably enhanced in recent years by developments in deep learning and neural networks. Convolutional Neural Networks (CNN), which enable superior feature extraction and identification, have attracted a lot of interest in the field of facial recognition. Deep learning models, like Face Net, have demonstrated impressive performance in face recognition tasks, opening the door for more advanced attendance systems.

Numerous benefits can be obtained by incorporating facial recognition technology into attendance management systems. As physical attendance registers and swipe cards are no longer necessary, there is less chance for human error and fraudulent activities like buddy punching. It also streamlines administrative tasks, delivers accurate attendance records, and offers real-time attendance tracking. Face recognition technology's automation boosts general effectiveness and productivity in businesses, educational institutions, and event management scenarios.

The installation of face recognition attendance systems has grown more feasible and accessible thanks to the growing availability of computer vision libraries like OpenCV and Dlib as well as improvements in hardware capabilities. By guaranteeing accurate and dependable tracking while minimizing manual efforts, these technologies have the potential to revolutionize attendance management.

Objective

The facial recognition attendance system employing the HAAR code project's development and implementation is guided by several goals. The following are these goals:

* Attendance management automation: The main goal is to use face recognition technology to automate the attendance management process. The system must be able to recognize people based on their facial features precisely and automatically, registering their attendance without the aid of a human.
* High precision and dependability in attendance tracking are two objectives of the system. It should be able to accurately identify people in a variety of lighting situations, positions, and facial variations, assuring accurate and reliable attendance records.
* Efficiency and time savings: The system's goal is to make the attendance management process more efficient while saving time for both administrators and users. Automating the procedure, it does away with the need for human attendance registers or swipe cards, improving the effectiveness and convenience of attendance tracking.
* Fraud Prevention: The system should make attendance fraud, including buddy punching or impersonation, as unlikely as possible. It improves security measures and guarantees that attendance records are real and correct by relying on distinctive facial features for identification.
* Scalability and Flexibility: The system must be scalable in terms of the number of people it can support, as well as adaptable enough to work in a variety of settings and environments. It should enable the registration and monitoring of numerous users without sacrificing performance.
* User-Friendly Interface: To maintain attendance records, add or remove users, and produce attendance reports, the system should offer administrators a user-friendly interface. Additionally, it must provide users with an easy way to monitor their attendance status.
* The system seeks to smoothly interface with current databases ands systems, such as student information systems and human resource management systems. To ensure ease of implementation, it should be interoperable with a range of operating systems and hardware setups.
* Evaluation of Performance: The project aims to assess the accuracy, speed, and resource usage of the face recognition attendance system. To emphasize the system's benefits and possibilities for development, a comparison with manual attendance methods will be done.

The project's goal is to produce thorough documentation and reports that describe how the face recognition attendance system was developed, put into use, and performed. This documentation will be a useful tool for future system improvements or replication across many

Purpose of the System

In the context of managing attendance, the facial recognition attendance system using the Haar code has a special function. The system's goal can be summed up as follows:

* **Automation of Attendance Tracking:** The system's main goal is to automate the attendance tracking procedure. The solution does away with the necessity for manual procedures like sign-in sheets or swipe cards by using face recognition technology. By automatically identifying people based on their facial traits and noting their attendance manually, it attempts to streamline and simplify attendance management.
* **Accuracy and Reliability:** The attendance monitoring system aspires to be accurate and dependable. It can accurately identify people even in a variety of lighting situations, positions, or facial variations by utilising face recognition algorithms. The goal is to prevent mistakes or fraudulent activity by making sure attendance records are accurate and reliable.
* **Time-saving and Efficiency:** The system is made to manage attendance more quickly and effectively. Traditional manual methods of tracking attendance can be laborious and necessitate a lot of administrative work. The system's goal is to automate the procedure, lessen administrative workload, and enable managers to better utilise their time and resources.
* **Improved Security and Fraud Prevention:** The technology attempts to improve security protocols and thwart fraudulent attendance. Face recognition technology makes it more challenging for someone to pose as someone else or take part in fraudulent activities like buddy punching. The goal is to guarantee that attendance records accurately reflect each person's presence.
* **Seamless Integration and Compatibility:** The system's goal is to interface easily with current databases and systems. To enable simple implementation in multiple situations, it should be compatible with a range of hardware setups and operating systems. The goal is to make sure that there won't be any major hiccups or compatibility problems when the system is integrated into businesses, educational institutions, or event management systems.
* **User-Friendly Interface:** Both administrators and users of the system should find it easy to use. To monitor attendance records, add or remove users, and generate reports, administrators should have simple controls. An easy way for people to check their personal attendance status should be provided. The goal is to improve usability and accessibility for users with different degrees of technical proficiency.
* **Performance Assessment and Improvement:** The system's goal is to go through performance assessment and ongoing improvement. Based on factors including accuracy, speed, and resource use, the system should be assessed. The goal is to pinpoint problem areas and improve the system's overall performance, resilience, and functionality.

Existing System

The current approach for managing attendance generally makes use of manual processes like sign-in sheets, roll calls, or swipe cards. There are several drawbacks and difficulties with these techniques, such as:

1. **Takes a lot of time**: Manual attendance monitoring needs people to physically check-in or produce their swipe cards, which may take a lot of time, especially in big organisations or events with a lot of participants.
2. **Human mistakes:** Manual attendance methods are susceptible to human mistakes such as illegible handwriting, inaccurate data entry, or unintentional omissions. These mistakes may result in incomplete attendance records and make further corrections challenging. Proxy attendance: In manual systems, individuals can engage in proxy attendance, where one person signs in or swipes on behalf of another individual. This practice compromises the accuracy and reliability of attendance records.
3. **Absence of real-time updates:** Manual systems frequently do not offer attendance records real-time updates. This might cause delays in acquiring precise attendance data, making it difficult to quickly check on attendance status.
4. **Data management challenges:** Handling and organising massive amounts of attendance data can be difficult. It takes a lot of paperwork, storage space, and labour to keep records up to date and access them when necessary.
5. **Limited reporting capabilities**: Manual systems sometimes have weak reporting options, which makes it challenging to create in-depth attendance records or examine attendance trends over time. Making decisions and monitoring attendance patterns may be hampered as a result.
6. **Data integration is inefficient:** Manual attendance systems may have trouble linking attendance data with other systems, such as payroll or student information systems. It takes more time and effort to manually enter or reconcile data for this integration.

Face recognition attendance systems have been developed as a result of the drawbacks of the previous manual methods and the desire for more automated and effective alternatives. In order to overcome the limitations of manual approaches and deliver precise, real-time attendance monitoring with less administrative work, these systems make use of computer vision and facial recognition technology.

# REQUIREMENTS & CONFIGURATION

Software Requirement

Software Requirements for the Face Recognition Attendance System using Haar Code:

1. Python: Python is the primary programming language for developing the system. Make sure Python is installed on the development environment.
2. Integrated Development Environment (IDE): Choose an IDE such as PyCharm, Visual Studio Code, or Jupyter Notebook to write and run the Python code.
3. OpenCV: Install the OpenCV library, which provides a wide range of functions and tools for image processing, computer vision, and face detection.
4. Dlib: Install the Dlib library, which offers pre-trained models and utilities for face recognition tasks.
5. Numpy: Numpy is a Python library used for numerical computations and array operations. It is commonly used in conjunction with OpenCV and Dlib.
6. Scikit-learn or TensorFlow: Depending on the choice of machine learning framework, install either Scikit-learn for traditional machine learning algorithms or TensorFlow for deep learning-based approaches.
7. Database Management System: Choose a database management system based on your requirements. Common options include MySQL, PostgreSQL, or MongoDB. Install and configure the chosen database system.
8. GUI Framework (Optional): If you plan to develop a graphical user interface (GUI) for the system, select a GUI framework like Tkinter, PyQt, or Flask. Install the relevant libraries accordingly.
9. Additional Libraries: Depending on the specific requirements and functionalities of the system, you may need to install additional libraries for image preprocessing, data visualization, or other supporting tasks. Examples include Matplotlib, Pandas, and Seaborn.
10. Operating System Compatibility: Ensure that the chosen software components are compatible with the operating system you are using for development and deployment. Most libraries and tools are platform-independent and work on popular operating systems like Windows, macOS, and Linux.
11. Version Management: Use a version control system like Git to manage your source code and collaborate with other developers effectively.

It's important to keep the software components and libraries up to date. You can use Python package managers like pip or conda to install, update, and manage the required packages and dependencies efficiently.

Remember to refer to the official documentation and resources for each software component to ensure proper installation and usage.

Top of Form

Hardware Requirement

Hardware Requirements for the Face Recognition Attendance System using Haar Code:

1. Computer or Server: A computer or server is required to host and run the face recognition attendance system. The hardware specifications should meet the system's processing and storage requirements. Recommended specifications include:
   * Processor: A multi-core processor with a clock speed of at least 2 GHz or higher.
   * RAM: Minimum 8 GB of RAM for smooth performance. Higher RAM capacity is beneficial for handling larger datasets or concurrent users.
   * Storage: Sufficient storage space to store the system code, libraries, databases, and captured images. A minimum of 100 GB of free disk space is recommended.
   * Network Connectivity: Reliable network connectivity is necessary for real-time attendance tracking and communication with the database server.
2. Camera or Webcam: A camera or webcam is required to capture images or video for face detection and recognition. The camera should have adequate resolution and image quality to ensure accurate identification. Higher-resolution cameras are recommended for better results, especially in scenarios with varying lighting conditions.
3. Operating System: The choice of the operating system depends on the developer's preference and compatibility with the software requirements. Commonly used operating systems for developing face recognition systems include Windows, macOS, and Linux distributions.
4. Additional Peripherals (Optional): Depending on the specific requirements of the system, additional peripherals such as monitors, keyboards, and mice may be needed for system setup, administration, and user interactions.

It's important to ensure that the hardware components are compatible and properly configured to support the face recognition attendance system. Regular hardware maintenance, such as cleaning the camera lens or ensuring proper connectivity, is also crucial for optimal system performance.

Note: The hardware requirements can vary based on factors such as the scale of the system (number of users), image processing demands, and the complexity of the face recognition algorithms used. It's recommended to assess the specific needs of your project and adjust the hardware requirements accordingly.

# TOOL’S USED

Editor – PyCharm

PyCharm is an integrated development environment (IDE) specifically designed for Python development. It is developed by JetBrains and offers a wide range of features to assist programmers in writing, debugging, and testing Python code.

Here are some key features of PyCharm:

1. Code Editor: PyCharm provides a powerful code editor with intelligent code completion, syntax highlighting, and code formatting. It supports various Python versions and provides a customizable interface.
2. Code Navigation: The IDE allows you to easily navigate through your codebase with features like Go to Definition, Find Usages, and Code Structure View. It helps you quickly understand and navigate complex projects.
3. Debugging: PyCharm offers a built-in debugger that allows you to step through your code, set breakpoints, inspect variables, and analyze the execution flow. It helps in identifying and fixing issues in your Python programs.
4. Testing: The IDE provides comprehensive support for testing your Python code. It supports popular testing frameworks like unittest, pytest, and doctest. You can run tests, view test results, and analyze code coverage.
5. Version Control: PyCharm has built-in integration with version control systems like Git, Mercurial, and Subversion. It provides features like diff viewer, commit history, and branch management, allowing you to easily collaborate with others and track changes in your project.
6. Python Web Development: PyCharm includes features specifically tailored for web development with Python. It supports frameworks like Django, Flask, and Pyramid, providing features such as project templates, code completion, and debugging for web applications.
7. Plugin Ecosystem: PyCharm supports a wide range of plugins and extensions to enhance its functionality. You can install additional plugins from the JetBrains marketplace or create your own to extend the IDE's capabilities.
8. Integration with Other Tools: PyCharm seamlessly integrates with various tools commonly used in Python development, such as virtual environments, package managers (pip), database management systems, and build systems like Docker and Vagrant.

PyCharm is available in both free and paid versions. The Community Edition is free and provides basic Python development features, while the Professional Edition offers advanced features and is available under a subscription-based licensing model

Working Technology

The working technology of the Face Recognition Attendance System using Haar Code involves several key steps and technologies. Here is an overview of the working process:

1. Face Detection: The system uses Haar cascades or similar algorithms to detect faces in images or video streams. The Haar cascades are trained to identify facial features based on patterns, edges, and texture variations.
2. Preprocessing: Before face recognition, the captured image or video frame undergoes preprocessing steps. These steps may include image resizing, normalization, and noise reduction to improve the quality of the input for better face recognition results.
3. Face Recognition: Once a face is detected, the system applies face recognition algorithms to match the detected face with known faces stored in a database. These algorithms analyze facial features, extract unique face embeddings, and compare them with the embeddings of known faces to determine a match.
4. Database Management: The system interacts with a database management system (e.g., MySQL, PostgreSQL) to store and retrieve information related to individuals, such as their names, unique IDs, and attendance records. The database stores the pre-registered faces along with their corresponding identities.
5. Attendance Tracking: When a face is recognized and matched with a known face in the database, the system records the attendance of the corresponding individual. It updates the attendance records in the database, marking the individual as present for that particular session.
6. Real-time Monitoring and Reporting: The system provides real-time monitoring of attendance, allowing administrators or users to view attendance status instantly. It can generate reports summarizing attendance data, such as the number of present and absent individuals, attendance percentages, or individual attendance histories.

The working technology components involved in the system include:

* Haar Cascades: Machine learning-based object detection method used for face detection.
* Image Processing: Techniques for image resizing, normalization, and noise reduction to enhance image quality.
* Face Recognition Algorithms: Machine learning or deep learning techniques used to analyze facial features and generate face embeddings for matching.
* Database Management System: Stores and manages information related to individuals and attendance records.
* Programming Languages: Python or C++ for implementing the system's logic and algorithms.
* OpenCV and Dlib Libraries: Provide functions and tools for image processing, face detection, and face recognition tasks.
* GUI Framework (optional): Used to develop a user-friendly graphical interface for administrators or users to interact with the system.

# SYSTEM ANALYSIS

Problem Definition

The Face Recognition Attendance System addresses several challenges and problems associated with traditional attendance tracking methods. The problem definition of this system can be summarized as follows:

1. Inaccuracy and Errors: Traditional attendance systems, such as manual paper-based systems or card-based systems, can be prone to errors and inaccuracies. Manual data entry or card swiping may result in human errors, such as typos or incorrect entries. This can lead to incorrect attendance records and discrepancies. The Face Recognition Attendance System aims to minimize these errors by automating the attendance tracking process using reliable face detection and recognition algorithms.
2. Time and Effort: Traditional attendance tracking methods often require significant time and manual effort. Manual entry of attendance data or card swiping for each individual can be time-consuming, especially in large organizations or classrooms. The Face Recognition Attendance System automates the process, allowing for efficient and rapid identification of individuals without the need for physical contact or manual data entry.
3. Security Concerns: Traditional attendance systems may lack robust security measures, making them susceptible to fraudulent activities such as buddy punching (where one person clocks in or out on behalf of another). Card-based systems can also be vulnerable to card theft or duplication. The Face Recognition Attendance System addresses these security concerns by utilizing biometric data (facial features) for identification, which is unique to each individual and difficult to counterfeit.
4. Scalability: Traditional attendance systems may face challenges in scaling up to accommodate a large number of users or locations. Manual attendance tracking becomes increasingly complex and time-consuming as the number of participants grows. The Face Recognition Attendance System is designed to be scalable, allowing for efficient tracking of attendance across various locations and a large number of individuals.
5. Real-time Tracking and Reporting: Traditional attendance systems often lack real-time tracking and reporting capabilities. Administrators may have to wait until the end of the day or class to collect attendance data and generate reports manually. The Face Recognition Attendance System enables real-time tracking of attendance, providing administrators with up-to-date information and the ability to generate instant reports.

By addressing these problems, the Face Recognition Attendance System provides an accurate, efficient, secure, and scalable solution for attendance tracking in various settings, including educational institutions, workplaces, and events.

Top of Form

# FEASIBILITY STUDY

Operational Feasibility

* User Acceptance: Evaluate users' willingness to adopt and use the system.
* Integration with Existing Processes: Assess compatibility with current attendance tracking processes.
* Reliability and Accuracy: Ensure consistent and accurate attendance data capture.
* Scalability: Evaluate the system's ability to handle a large number of users and locations.
* Technical Compatibility: Assess compatibility with existing hardware and software infrastructure.
* Maintenance and Support: Consider ongoing system maintenance and support requirements.
* Cost-Benefit Analysis: Evaluate the financial feasibility and potential benefits of the system.

Technical Feasibility

Technical feasibility refers to the assessment of whether the proposed system can be developed and implemented using existing technology and resources. Here are key considerations regarding the technical feasibility of the Face Recognition Attendance System using Haar Code:

1. System Requirements: Evaluate whether the necessary hardware and software resources are available to develop and run the system. Consider the computing power, memory, storage, and network requirements to support the face detection and recognition algorithms. Ensure that the organization's technical infrastructure meets these requirements.
2. Technology Availability: Assess the availability and suitability of the required technologies, such as the Haar cascades algorithm or other face detection and recognition libraries. Determine if the technology is mature, well-documented, and supported by a community or vendor. Verify that the chosen technology can effectively meet the system's objectives.
3. Development Resources: Evaluate the availability of skilled developers or development teams with expertise in face recognition and software development. Consider whether the organization has the necessary resources, such as programming languages (e.g., Python, C++) and integrated development environments (IDEs), to implement and maintain the system.
4. Integration with Existing Systems: Determine if the system can integrate with existing infrastructure and software systems, such as databases or attendance management systems. Consider the compatibility and interoperability of the proposed system with other systems used within the organization.
5. Data Management and Privacy: Assess the system's ability to handle and store face images and associated attendance data securely. Consider data privacy and protection regulations, and ensure that the system adheres to relevant security standards and protocols. Implement measures to safeguard personal information and prevent unauthorized access.
6. Scalability and Performance: Evaluate the system's ability to handle increasing numbers of users and provide real-time performance. Consider factors such as processing speed, response time, and the ability to handle simultaneous face detection and recognition requests. The system should be scalable to accommodate organizational growth and changing requirements.
7. Testing and Validation: Plan for comprehensive testing and validation processes to ensure the accuracy, reliability, and robustness of the system. Conduct performance testing, functional testing, and security testing to identify and address any potential issues or vulnerabilities.Top of Form

Economical feasibility

Economical feasibility refers to the assessment of whether the Face Recognition Attendance System using the Haar Code is financially viable and beneficial for the organization. Here are key considerations regarding the economic feasibility of the system:

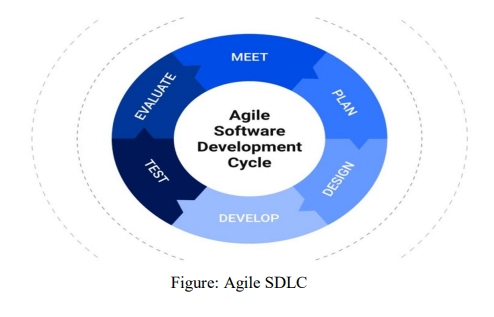
1. Cost Analysis: Conduct a comprehensive cost analysis to determine the expenses associated with developing, implementing, and maintaining the system. Consider costs related to hardware and software acquisition, development resources, licensing fees, infrastructure upgrades, and ongoing maintenance and support. Evaluate if the financial investment aligns with the organization's budget and financial capabilities.
2. Cost Savings: Identify potential cost savings that can be achieved by implementing the system. Consider the reduction in labour costs associated with manual attendance tracking processes, such as the time spent on manual data entry and verification. Assess if the system can lead to operational efficiencies and cost reductions in the long run.
3. Return on Investment (ROI): Calculate the potential return on investment for the system. Estimate the time it would take to recoup the initial investment based on the projected cost savings and increased efficiency. Consider the potential benefits in terms of accuracy, time savings, and improved attendance management processes.
4. Scalability and Flexibility: Evaluate the system's ability to scale and adapt to the organization's growth and changing needs. Assess if the system can accommodate future expansions, such as adding more users, and locations, or integrating with other systems. Consider the potential costs and benefits associated with system scalability.
5. Comparison with Alternative Solutions: Compare the proposed system with alternative attendance trackings methods, such as manual processes or other automated systems. Evaluate the cost-effectiveness of the system in comparison to these alternatives. Consider the advantages and disadvantages of each approach and assess the cost-benefit ratio.
6. Risks and Contingencies: Identify any potential risks or uncertainties that may impact the economic feasibility of the system. Assess the feasibility of mitigating these risks through contingency plans or alternative strategies. Consider factors such as market volatility, technological advancements, and changing regulatory requirements.

By assessing the economical feasibility of the Face Recognition Attendance System, organizations can determine if the expected benefits and cost savings outweigh the initial investment and ongoing expenses. This evaluation helps in making informed decisions regarding the financial viability and potential impact on the organization's bottom line.

Top of Form

# CONCEPTUAL MODEL

Methodology

In this project, we have used Agile SDLC. Agile Software Development Life Cycle is an approach that is used to design a software management process. It is a combination of iterative and incremental process models and it breaks the product into small incremental builds. These builds are provided in iterations.

Block diagram of the General Framework:

Extract Features(SIFT key points and pcs)

Face detection

Input

Image from data base

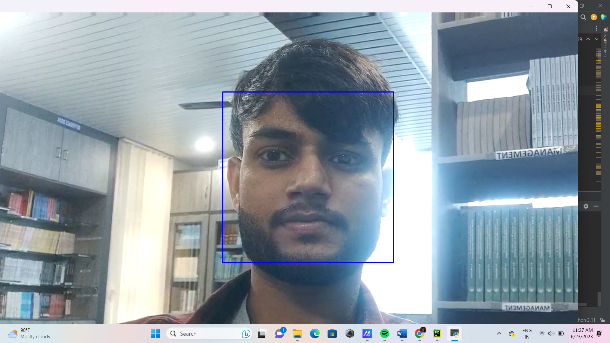
Identification

process

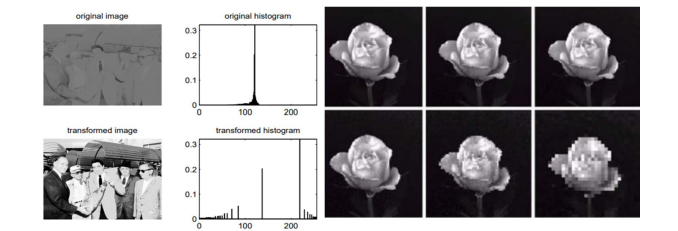
Key point classifier

(Based on SVM)

The objective of this project is to detect the face segment from the video frame, extract the useful features from the face detected, classify the features in order to recognize the face detected and record the attendance of the identified student. This project will be developed using machine learning, Web technology, python (tkinter, csv, numpy, pandas). 1) First all the students of the class must register themselves by entering the required details and then their 250 images will be captured from different angle the frontal face is extracted from the image then converted to gray scale using open source computer vision library (Gray=cv2.cvtColor (frame, cv2.COLOR\_BGR2GRAY)) and stored in the dataset.

**Conversion of Image to Grayscale Image**

**RGB Image Gray Scale**

2) After collecting the dataset we will pre-process and clean the data with the help of Viola jones algorithm and SVM (Support vector machine). Viola jones algorithm is the most popular algorithm to localize the face segment from static images or video frame. Pre-processing enhances the performance of the system. Scaling is one of the most important preprocessing steps to manipulate the size of the image. The size and pixels of the image carry spatial info. Hence, the size should be same for all the images for normalization and standardization purposes. Subhi Singh (2015) proposed PCA (Principal Component Analysis) to extract features from facial images, same length and width of image is preferred.

Checkerboard Effect Significant Fig: Image Pre-processing Increasing from Left to right

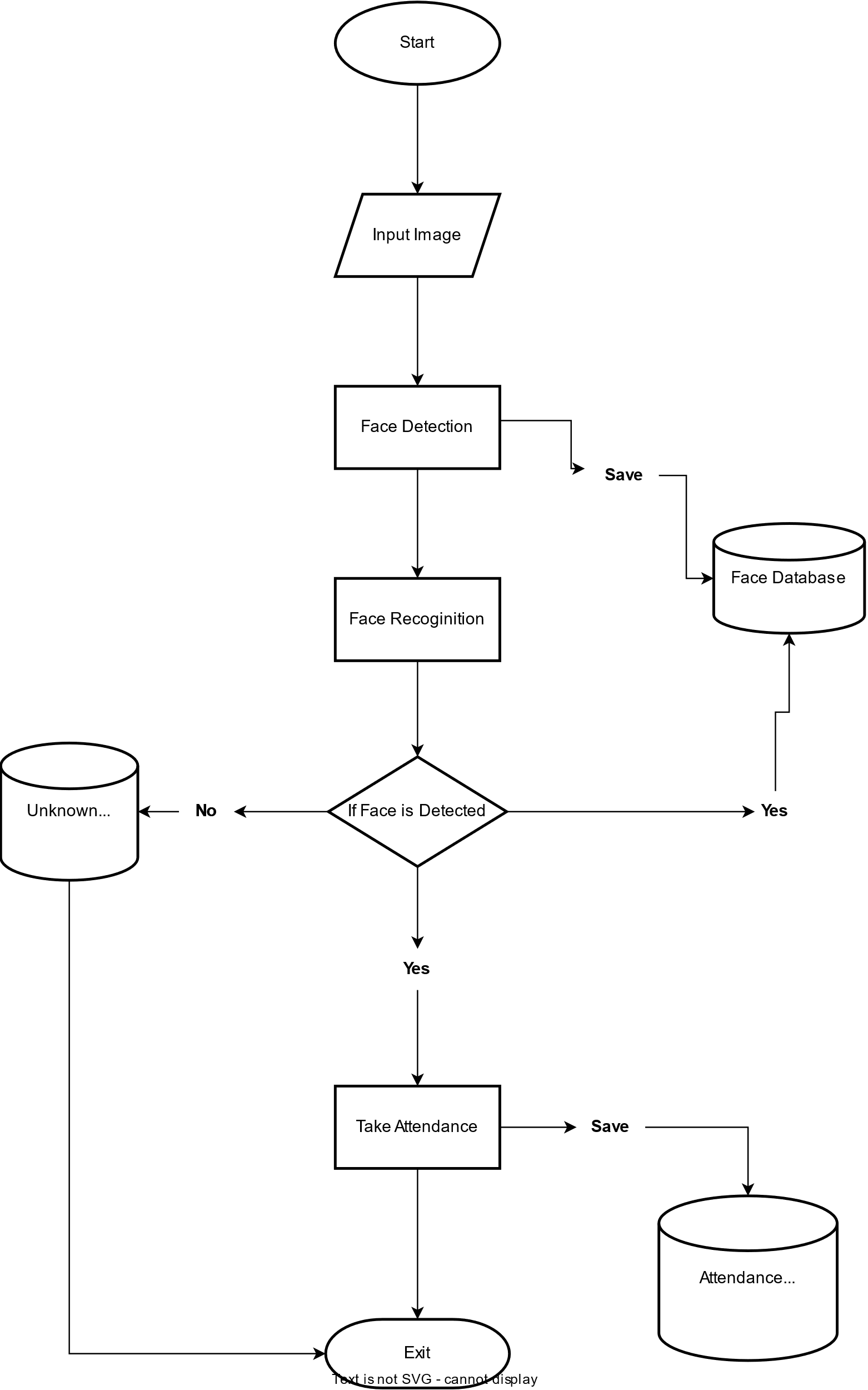
3) A camera will take a continuous stream during each session, faces will be detected from live streaming videos of the classroom then the detected faces will be compared with the training image dataset based on Eigen features. Eigenfeatures are used to characterize images. If a match is found, attendance will be marked for the respective student. The camera will be capturing images in real-time so if an enrolled face is not detected in the training image dataset, the student.

4) For the frontend purpose we will create an interactive GUI (Graphical user interface) where there will be various interactive functionality like to generate a graph, faculty can mark students present and absent as per their convenience also they can edit the student attendance and admin will be able to register and remove the student and faculty details.

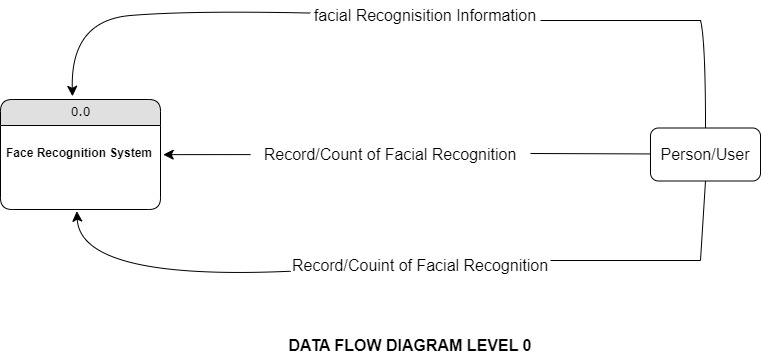
5) These all-purpose we are Python as frontend and backend, for training the dataset we are using machine learning algorithm.

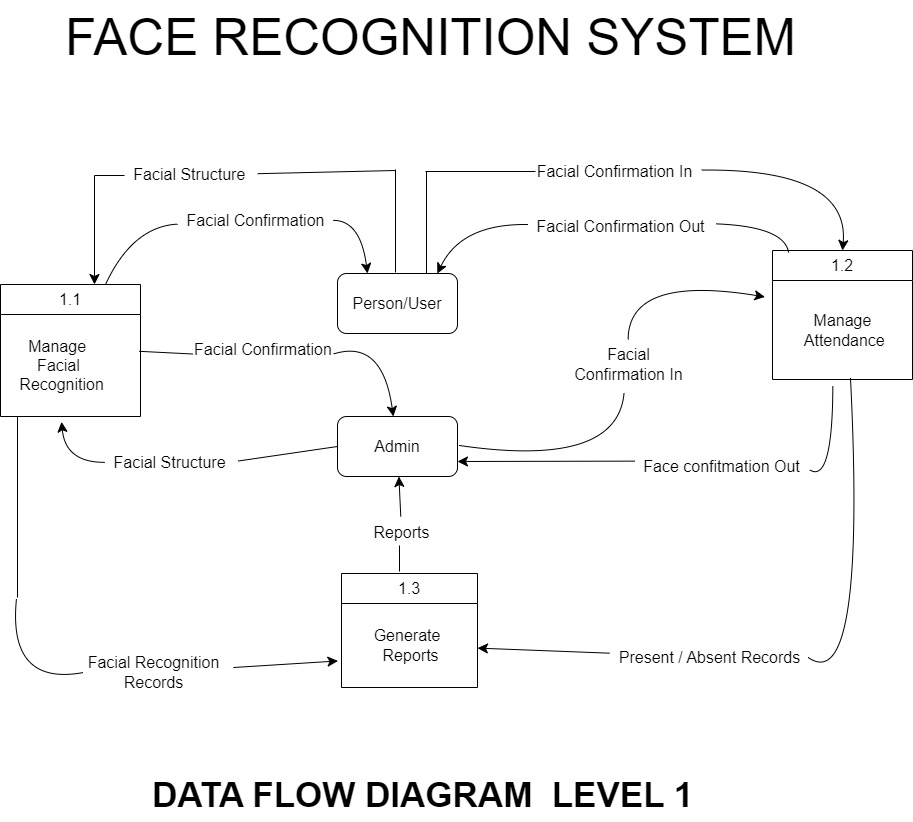
FLOW CHART

The flow chart depicting the flow of our system is:

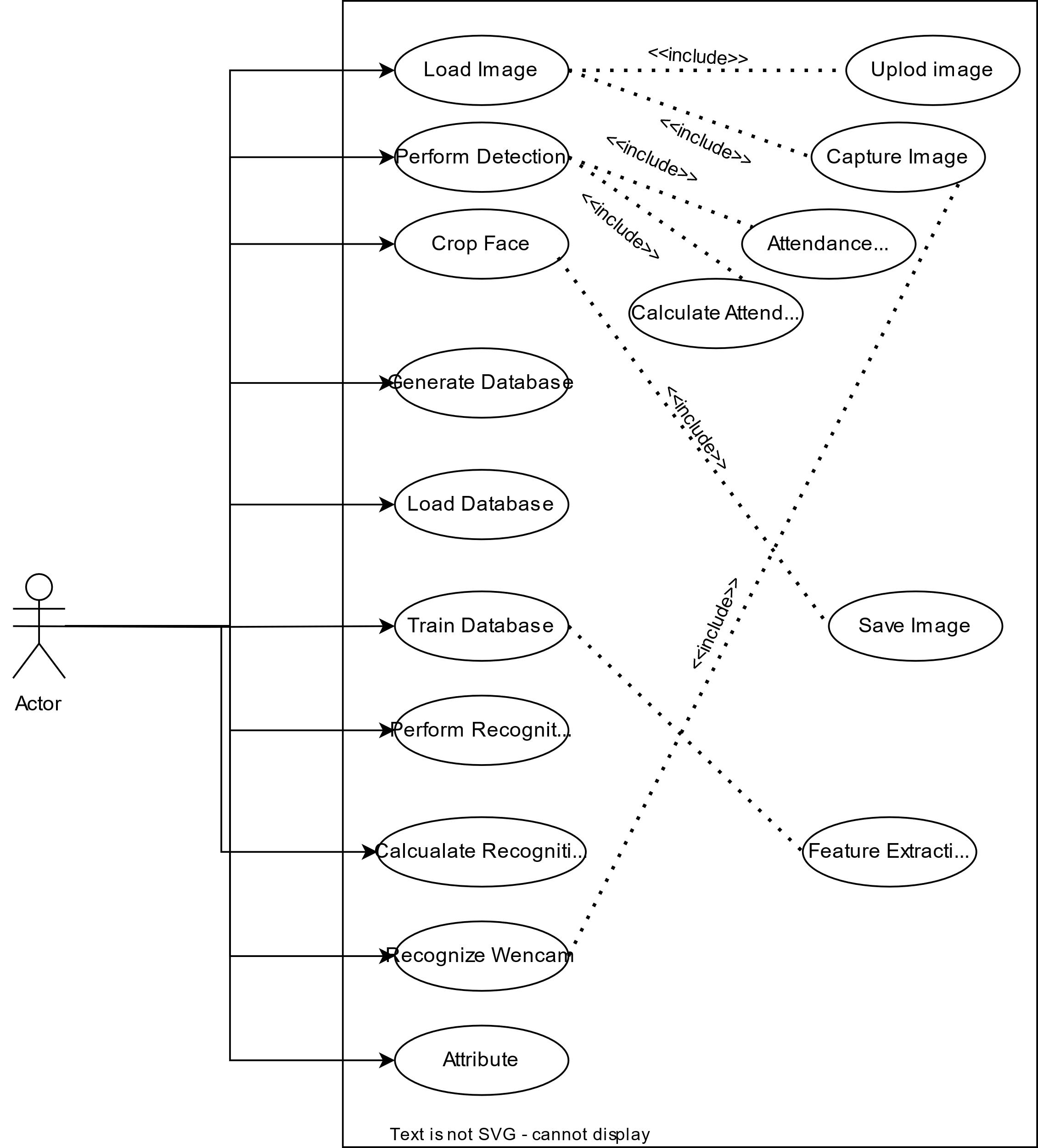
[](https://d.docs.live.net/98643f91d85d1b93/Desktop/major%20project/Ai%20Ml.pptx)

Data flow Diagram





UML Diagram



# SYSTEM DESIGN

1. System Architecture:
   * Define the high-level architecture of the system, including the hardware, software, and network infrastructure.
   * Determine the distribution of components and their interactions, considering factors like scalability, reliability, and performance.
2. User Interface:
   * Design an intuitive and user-friendly interface for administrators, teachers, or employees to interact with the system.
   * Include features such as user authentication, attendance management, reporting, and system configuration.
   * Ensure the interface is visually appealing, responsive, and supports multiple devices.
3. Face Detection and Recognition:
   * Implement the Haar cascades-based face detection algorithm to locate and extract faces from images or video streams.
   * Utilize face recognition algorithms, such as eigenfaces, Fisherfaces, or deep learning-based models, to recognize individuals from their facial features.
   * Integrate and configure appropriate libraries or frameworks for efficient and accurate face detection and recognition.
4. Image Processing and Preprocessing:
   * Apply image processing techniques to enhance the quality and clarity of the captured images.
   * Perform preprocessing tasks such as resizing, normalization, and noise reduction to improve the accuracy of face detection and recognition.
5. Attendance Recording and Management:
   * Develop mechanisms to capture attendance data based on the recognized faces.
   * Implement algorithms to match detected faces with pre-registered identities and record attendance accurately.
   * Store attendance data along with relevant metadata, such as date, time, and location.
6. Database Management:
   * Design the database schema to store and manage attendance records, user information, and other relevant data.
   * Select an appropriate database management system (DBMS) to handle data storage, retrieval, and querying efficiently.
   * Ensure data integrity, security, and scalability of the database.
7. System Integration:
   * Integrate the attendance system with other systems or applications, such as student information systems or HR systems.
   * Enable data exchange and synchronization between the attendance system and other systems to ensure data consistency.
   * Implement APIs or data interfaces to facilitate seamless integration.
8. Security and Privacy:
   * Implement security measures to protect the system and data from unauthorized access or breaches.
   * Apply encryption techniques to secure sensitive data, such as face images and personal information.
   * Comply with data protection regulations and privacy laws.
9. Performance Optimization:
   * Optimize the system's performance for real-time face detection and recognition.
   * Use techniques like parallel processing, caching, and optimization algorithms to reduce processing time.
   * Conduct performance testing and tuning to ensure the system meets response time and scalability requirements.
10. Testing and Quality Assurance:
    * Develop a comprehensive testing strategy to validate the system's functionality, accuracy, and performance.
    * Conduct unit testing, integration testing, and system testing to identify and fix any issues or bugs.
    * Implement quality assurance processes to ensure the system meets the defined requirements and user expectations.

# TESTING

**Black Box Testing**

Black box testing for the Face Recognition Attendance System using Haar Code can be performed to validate its functionality and ensure that it meets the specified requirements. Here are some examples of black box testing techniques that can be applied:

1. Equivalence Partitioning:
   * Identify different equivalence classes for inputs, such as valid face images, invalid face images, or boundary cases.
   * Select representative test cases from each equivalence class to cover a wide range of possible inputs.
2. Boundary Value Analysis:
   * Test the system using inputs at the boundaries of valid ranges, such as the minimum and maximum sizes for face images.
   * Verify that the system handles these boundary cases correctly and does not produce unexpected behaviour or errors.
3. Decision Table Testing:
   * Create decision tables that capture the system's behaviour based on different combinations of inputs and conditions.
   * Generate test cases to cover each possible combination and evaluate the system's response in each scenario.
4. Error Handling:
   * Test the system's response to invalid inputs, such as providing a non-image file or a corrupted image.
   * Verify that appropriate error messages or notifications are displayed to the user.
5. Usability Testing:
   * Evaluate the user interface and interaction flow of the system by simulating typical user scenarios.
   * Testers can perform tasks like registering new users, capturing attendance, and generating reports to ensure smooth usability.
6. Security Testing:
   * Test the system for security vulnerabilities, such as bypassing authentication or manipulating attendance records.
   * Ensure that the system implements proper access controls and protects sensitive data.
7. Performance Testing:
   * Validate the system's performance by simulating multiple users concurrently accessing the system.
   * Measure the response time for attendance capture and other system operations to ensure acceptable performance levels.
8. Compatibility Testing:
   * Test the system's compatibility with different devices and platforms, such as desktop browsers, mobile devices, or different operating systems.
   * Verify that the system functions correctly and displays properly across various environments.

By applying these black box testing techniques, testers can assess the functionality, usability, security, and performance aspects of the Face Recognition Attendance System using Haar Code. The goal is to identify any issues or discrepancies between the expected behaviour and the actual system behaviour, allowing developers to address them before deployment.

**White Box Testing**

White box testing, also known as structural testing or clear box testing, focuses on examining the internal structure, implementation, and logic of the Face Recognition Attendance System using Haar Code. It involves testing the system with knowledge of its internal workings. Here are some examples of white box testing techniques that can be applied:

1. Statement Coverage:
   * Ensure that each statement in the system's source code is executed at least once during testing.
   * Develop test cases that cover different execution paths, including loops, conditional statements, and error handling.
2. Branch Coverage:
   * Test all possible branches and decision points in the system's code.
   * Design test cases that exercise both true and false outcomes of conditional statements.
3. Path Coverage:
   * Aim to test all possible paths through the system's code.
   * Construct test cases that cover different combinations of statements and branches.
4. Condition Coverage:
   * Evaluate the logical conditions in the system's code to ensure that all possible outcomes are tested.
   * Design test cases that cover each possible combination of conditions within the code.
5. Loop Testing:
   * Test the system's behaviour during iterations of loops.
   * Include test cases that examine the loop's entry, exit, and boundary conditions.
6. Data Flow Testing:
   * Examine the flow of data within the system and test for possible data inconsistencies or errors.
   * Design test cases that cover the paths and transformations of data variables throughout the code.
7. Integration Testing:
   * Test the interactions and interfaces between different modules or components of the system.
   * Verify that data is passed correctly between modules and that the overall system functions as expected.
8. Performance Testing:
   * Measure the system's performance under various load conditions and stress levels.
   * Evaluate factors such as response time, resource utilization, and scalability.
9. Security Testing:
   * Test the system for potential security vulnerabilities, such as SQL injection, cross-site scripting, or unauthorized access.
   * Review the code for security flaws and apply appropriate testing techniques to address them.
10. Error Handling:
    * Test the system's ability to handle and recover from errors, exceptions, or unexpected inputs.
    * Design test cases to simulate error conditions and verify that the system responds appropriately.

White box testing helps uncover issues related to the internal structure and implementation of the system. It can identify code defects, logical errors, performance bottlenecks, and security vulnerabilities. By combining white box testing with black box testing, comprehensive test coverage can be achieved, ensuring the quality and reliability of the Face Recognition Attendance System using Haar Code.

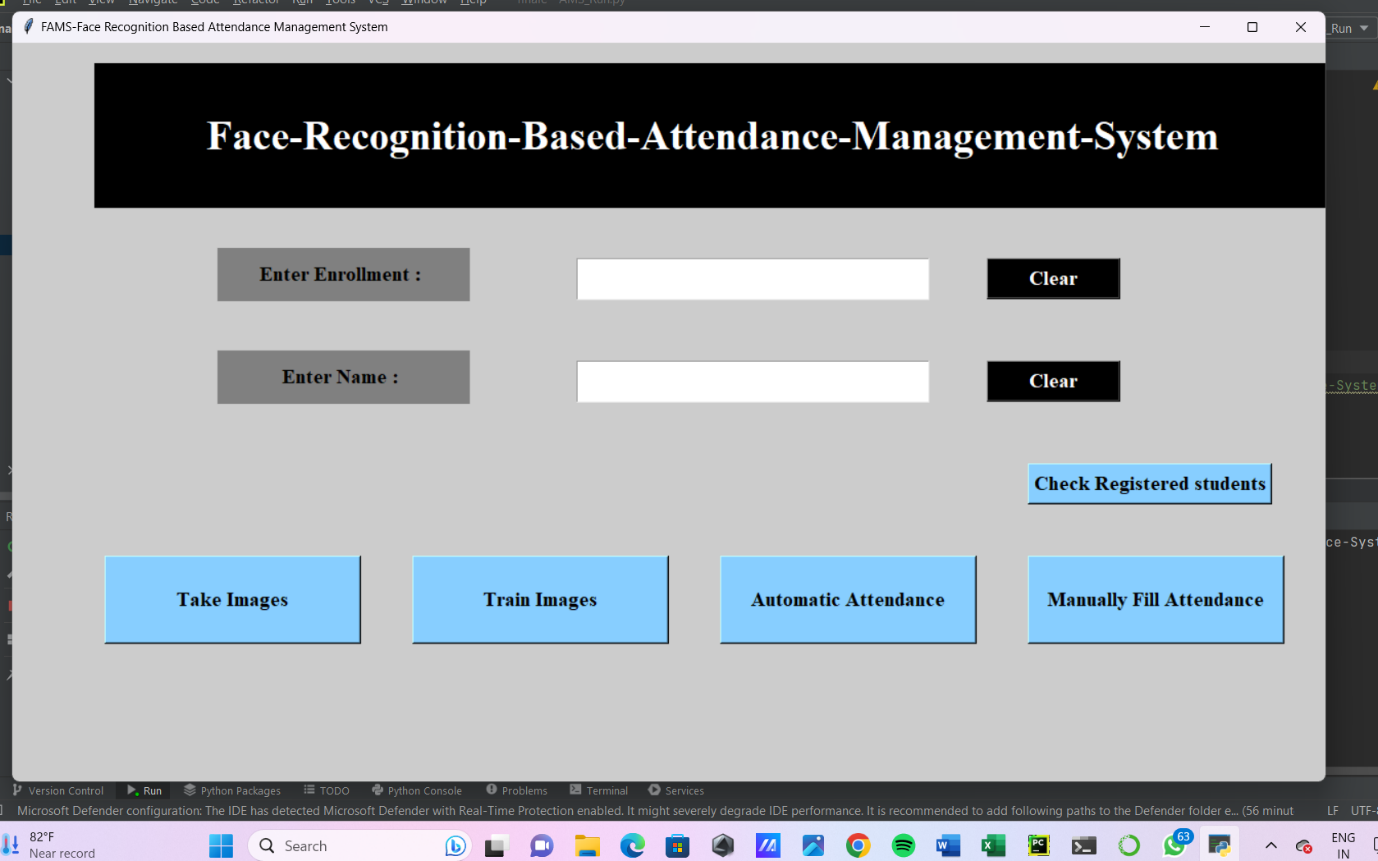
Test cases

Creating test cases for the Face Recognition Attendance System using Haar Code requires understanding the system's functionalities, requirements, and potential scenarios. Here are some example test cases that can be considered:

1. Test Case: Valid Face Recognition
   * Description: Verify that the system accurately recognizes and records attendance for valid faces.
   * Test Steps:
     1. Provide a valid face image for attendance.
     2. Verify that the system detects and recognizes the face correctly.
     3. Check that the attendance record is created or updated accurately.
2. Test Case: Invalid Face Recognition
   * Description: Validate the system's behavior when faced with invalid or unrecognized faces.
   * Test Steps:
     1. Provide an image without a face for attendance.
     2. Verify that the system detects the absence of a face and provides an appropriate error message.
     3. Confirm that no attendance record is created or updated.
3. Test Case: Boundary Condition
   * Description: Test the system's behavior when faced with face images at the boundaries of valid inputs.
   * Test Steps:
     1. Provide a face image at the minimum allowed size.
     2. Verify that the system detects and recognizes the face accurately.
     3. Repeat the above steps for a face image at the maximum allowed size.
4. Test Case: Performance Testing
   * Description: Evaluate the system's performance and response time under different loads.
   * Test Steps:
     1. Simulate multiple users simultaneously capturing attendance with their face images.
     2. Measure the response time for face detection, recognition, and attendance recording.
     3. Ensure that the system maintains acceptable performance levels under different load conditions.
5. Test Case: Error Handling
   * Description: Test the system's ability to handle errors and exceptions gracefully.
   * Test Steps:
     1. Provide a corrupted or unreadable face image for attendance.
     2. Verify that the system identifies the error and provides an appropriate error message.
     3. Confirm that no attendance record is created or updated in case of an error.
6. Test Case: Integration Testing
   * Description: Validate the integration of the face recognition system with other modules or systems.
   * Test Steps:
     1. Integrate the face recognition system with an existing attendance management system or database.
     2. Verify that attendance records are accurately synchronized between the systems.
     3. Perform operations in one system and confirm that the changes are reflected in the other.
7. Test Case: Security Testing
   * Description: Test the system's security measures to ensure the protection of biometric data.
   * Test Steps:
     1. Attempt to bypass authentication by providing an incorrect face image.
     2. Verify that the system denies access and provides an appropriate error message.
     3. Ensure that the system securely stores and encrypts biometric data to prevent unauthorized access.

# SNAPSHOTS & CODING

**Main Page/Landing Page**



import csv  
import datetime  
import os  
import time  
import tkinter as tk  
from tkinter import \*  
  
import cv2  
import numpy as np  
import pandas as pd  
from PIL import Image  
  
  
# Window is our Main frame of system  
window = tk.Tk()  
window.title("FAMS-Face Recognition Based Attendance Management System")  
  
window.geometry('1280x720')  
window.configure(background='grey80')  
  
# GUI for manually fill attendance  
  
  
def manually\_fill():  
 global sb  
 sb = tk.Tk()  
 # sb.iconbitmap('AMS.ico')  
 sb.title("Enter subject name...")  
 sb.geometry('580x320')  
 sb.configure(background='grey80')  
  
 def err\_screen\_for\_subject():  
  
 def ec\_delete():  
 ec.destroy()  
 global ec  
 ec = tk.Tk()  
 ec.geometry('300x100')  
 # ec.iconbitmap('AMS.ico')  
 ec.title('Warning!!')  
 ec.configure(background='snow')  
 Label(ec, text='Please enter your subject name!!!', fg='red',  
 bg='white', font=('times', 16, ' bold ')).pack()  
 Button(ec, text='OK', command=ec\_delete, fg="black", bg="lawn green", width=9, height=1, activebackground="Red",  
 font=('times', 15, ' bold ')).place(x=90, y=50)  
  
 def fill\_attendance():  
 ts = time.time()  
 Date = datetime.datetime.fromtimestamp(ts).strftime('%Y\_%m\_%d')  
 timeStamp = datetime.datetime.fromtimestamp(ts).strftime('%H:%M:%S')  
 Time = datetime.datetime.fromtimestamp(ts).strftime('%H:%M:%S')  
 Hour, Minute, Second = timeStamp.split(":")  
 # Creatting csv of attendance  
  
 # Create table for Attendance  
 date\_for\_DB = datetime.datetime.fromtimestamp(ts).strftime('%Y\_%m\_%d')  
 global subb  
 subb = SUB\_ENTRY.get()  
 DB\_table\_name = str(subb + "\_" + Date + "\_Time\_" +  
 Hour + "\_" + Minute + "\_" + Second)  
  
 import pymysql.connections  
  
 # Connect to the database  
 try:  
 global cursor  
 connection = pymysql.connect(  
 host='localhost', user='root', password='', db='manually\_fill\_attendance')  
 cursor = connection.cursor()  
 except Exception as e:  
 print(e)  
  
 sql = "CREATE TABLE " + DB\_table\_name + """  
 (ID INT NOT NULL AUTO\_INCREMENT,  
 ENROLLMENT varchar(100) NOT NULL,  
 NAME VARCHAR(50) NOT NULL,  
 DATE VARCHAR(20) NOT NULL,  
 TIME VARCHAR(20) NOT NULL,  
 PRIMARY KEY (ID)  
 );  
 """  
  
 try:  
 cursor.execute(sql) # for create a table  
 except Exception as ex:  
 print(ex) #  
  
 if subb == '':  
 err\_screen\_for\_subject()  
 else:  
 sb.destroy()  
 MFW = tk.Tk()  
 # MFW.iconbitmap('AMS.ico')  
 MFW.title("Manually attendance of " + str(subb))  
 MFW.geometry('880x470')  
 MFW.configure(background='grey80')  
  
 def del\_errsc2():  
 errsc2.destroy()  
  
 def err\_screen1():  
 global errsc2  
 errsc2 = tk.Tk()  
 errsc2.geometry('330x100')  
 # errsc2.iconbitmap('AMS.ico')  
 errsc2.title('Warning!!')  
 errsc2.configure(background='grey80')  
 Label(errsc2, text='Please enter Student & Enrollment!!!', fg='black', bg='white',  
 font=('times', 16, ' bold ')).pack()  
 Button(errsc2, text='OK', command=del\_errsc2, fg="black", bg="lawn green", width=9, height=1,  
 activebackground="Red", font=('times', 15, ' bold ')).place(x=90, y=50)  
  
 def testVal(inStr, acttyp):  
 if acttyp == '1': # insert  
 if not inStr.isdigit():  
 return False  
 return True  
  
 ENR = tk.Label(MFW, text="Enter Enrollment", width=15, height=2, fg="black", bg="grey",  
 font=('times', 15))  
 ENR.place(x=30, y=100)  
  
 STU\_NAME = tk.Label(MFW, text="Enter Student name", width=15, height=2, fg="black", bg="grey",  
 font=('times', 15))  
 STU\_NAME.place(x=30, y=200)  
  
 global ENR\_ENTRY  
 ENR\_ENTRY = tk.Entry(MFW, width=20, validate='key',  
 bg="white", fg="black", font=('times', 23))  
 ENR\_ENTRY['validatecommand'] = (  
 ENR\_ENTRY.register(testVal), '%P', '%d')  
 ENR\_ENTRY.place(x=290, y=105)  
  
 def remove\_enr():  
 ENR\_ENTRY.delete(first=0, last=22)  
  
 STUDENT\_ENTRY = tk.Entry(  
 MFW, width=20, bg="white", fg="black", font=('times', 23))  
 STUDENT\_ENTRY.place(x=290, y=205)  
  
 def remove\_student():  
 STUDENT\_ENTRY.delete(first=0, last=22)  
  
 # get important variable  
 def enter\_data\_DB():  
 ENROLLMENT = ENR\_ENTRY.get()  
 STUDENT = STUDENT\_ENTRY.get()  
 if ENROLLMENT == '':  
 err\_screen1()  
 elif STUDENT == '':  
 err\_screen1()  
 else:  
 time = datetime.datetime.fromtimestamp(  
 ts).strftime('%H:%M:%S')  
 Hour, Minute, Second = time.split(":")  
 Insert\_data = "INSERT INTO " + DB\_table\_name + \  
 " (ID,ENROLLMENT,NAME,DATE,TIME) VALUES (0, %s, %s, %s,%s)"  
 VALUES = (str(ENROLLMENT), str(  
 STUDENT), str(Date), str(time))  
 try:  
 cursor.execute(Insert\_data, VALUES)  
 except Exception as e:  
 print(e)  
 ENR\_ENTRY.delete(first=0, last=22)  
 STUDENT\_ENTRY.delete(first=0, last=22)  
  
 def create\_csv():  
 import csv  
 cursor.execute("select \* from " + DB\_table\_name + ";")  
 csv\_name = 'C:/Users/amans/finale/Face-Recognition-Attendance-System-main/Face-Recognition-Attendance-System-main/Attendance/Manually Attendance/'+DB\_table\_name+'.csv'  
 with open(csv\_name, "w") as csv\_file:  
 csv\_writer = csv.writer(csv\_file)  
 csv\_writer.writerow(  
 [i[0] for i in cursor.description]) # write headers  
 csv\_writer.writerows(cursor)  
 O = "CSV created Successfully"  
 Notifi.configure(text=O, bg="Green", fg="white",  
 width=33, font=('times', 19, 'bold'))  
 Notifi.place(x=180, y=380)  
 import csv  
 import tkinter  
 root = tkinter.Tk()  
 root.title("Attendance of " + subb)  
 root.configure(background='grey80')  
 with open(csv\_name, newline="") as file:  
 reader = csv.reader(file)  
 r = 0  
  
 for col in reader:  
 c = 0  
 for row in col:  
 # i've added some styling  
 label = tkinter.Label(root, width=18, height=1, fg="black", font=('times', 13, ' bold '),  
 bg="white", text=row, relief=tkinter.RIDGE)  
 label.grid(row=r, column=c)  
 c += 1  
 r += 1  
 root.mainloop()  
  
 Notifi = tk.Label(MFW, text="CSV created Successfully", bg="Green", fg="white", width=33,  
 height=2, font=('times', 19, 'bold'))  
  
 c1ear\_enroll = tk.Button(MFW, text="Clear", command=remove\_enr, fg="white", bg="black", width=10,  
 height=1,  
 activebackground="white", font=('times', 15, ' bold '))  
 c1ear\_enroll.place(x=690, y=100)  
  
 c1ear\_student = tk.Button(MFW, text="Clear", command=remove\_student, fg="white", bg="black", width=10,  
 height=1,  
 activebackground="white", font=('times', 15, ' bold '))  
 c1ear\_student.place(x=690, y=200)  
  
 DATA\_SUB = tk.Button(MFW, text="Enter Data", command=enter\_data\_DB, fg="black", bg="SkyBlue1", width=20,  
 height=2,  
 activebackground="white", font=('times', 15, ' bold '))  
 DATA\_SUB.place(x=170, y=300)  
  
 MAKE\_CSV = tk.Button(MFW, text="Convert to CSV", command=create\_csv, fg="black", bg="SkyBlue1", width=20,  
 height=2,  
 activebackground="white", font=('times', 15, ' bold '))  
 MAKE\_CSV.place(x=570, y=300)  
  
 def attf():  
 import subprocess  
 subprocess.Popen(  
 r'explorer /select,"C:/Users/amans/finale/Face-Recognition-Attendance-System-main/Face-Recognition-Attendance-System-main/Attendance/Manually Attendance/-------Check atttendance-------"')  
  
 attf = tk.Button(MFW, text="Check Sheets", command=attf, fg="white", bg="black",  
 width=12, height=1, activebackground="white", font=('times', 14, ' bold '))  
 attf.place(x=730, y=410)  
  
 MFW.mainloop()  
  
 SUB = tk.Label(sb, text="Enter Subject : ", width=15, height=2,  
 fg="black", bg="grey80", font=('times', 15, ' bold '))  
 SUB.place(x=30, y=100)  
  
 global SUB\_ENTRY  
  
 SUB\_ENTRY = tk.Entry(sb, width=20, bg="white",  
 fg="black", font=('times', 23))  
 SUB\_ENTRY.place(x=250, y=105)  
  
 fill\_manual\_attendance = tk.Button(sb, text="Fill Attendance", command=fill\_attendance, fg="black", bg="SkyBlue1", width=20, height=2,  
 activebackground="white", font=('times', 15, ' bold '))  
 fill\_manual\_attendance.place(x=250, y=160)  
 sb.mainloop()  
  
# For clear textbox  
  
  
def clear():  
 txt.delete(first=0, last=22)  
  
  
def clear1():  
 txt2.delete(first=0, last=22)  
  
  
def del\_sc1():  
 sc1.destroy()  
  
  
def err\_screen():  
 global sc1  
 sc1 = tk.Tk()  
 sc1.geometry('300x100')  
 # sc1.iconbitmap('AMS.ico')  
 sc1.title('Warning!!')  
 sc1.configure(background='grey80')  
 Label(sc1, text='Enrollment & Name required!!!', fg='black',  
 bg='white', font=('times', 16)).pack()  
 Button(sc1, text='OK', command=del\_sc1, fg="black", bg="lawn green", width=9,  
 height=1, activebackground="Red", font=('times', 15, ' bold ')).place(x=90, y=50)  
  
# Error screen2  
  
  
def del\_sc2():  
 sc2.destroy()  
  
  
def err\_screen1():  
 global sc2  
 sc2 = tk.Tk()  
 sc2.geometry('300x100')  
 # sc2.iconbitmap('AMS.ico')  
 sc2.title('Warning!!')  
 sc2.configure(background='grey80')  
 Label(sc2, text='Please enter your subject name!!!', fg='black',  
 bg='white', font=('times', 16)).pack()  
 Button(sc2, text='OK', command=del\_sc2, fg="black", bg="lawn green", width=9,  
 height=1, activebackground="Red", font=('times', 15, ' bold ')).place(x=90, y=50)  
  
# For take images for datasets  
  
  
def take\_img():  
 l1 = txt.get()  
 l2 = txt2.get()  
 if l1 == '':  
 err\_screen()  
 elif l2 == '':  
 err\_screen()  
 else:  
 try:  
 cam = cv2.VideoCapture(0)  
 detector = cv2.CascadeClassifier(  
 'haarcascade\_frontalface\_default.xml')  
 Enrollment = txt.get()  
 Name = txt2.get()  
 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)  
 # incrementing sample number  
 sampleNum = sampleNum + 1  
 # saving the captured face in the dataset folder  
 cv2.imwrite("TrainingImage/ " + Name + "." + Enrollment + '.' + str(sampleNum) + ".jpg",  
 gray)  
 print("Images Saved for Enrollment :")  
 cv2.imshow('Frame', img)  
 # wait for 100 miliseconds  
 if cv2.waitKey(1) & 0xFF == ord('q'):  
 break  
 #  
 # # break if the sample number is morethan 100  
 elif sampleNum > 70:  
 break  
  
  
 cam.release()  
 cv2.destroyAllWindows()  
 ts = time.time()  
 Date = datetime.datetime.fromtimestamp(ts).strftime('%Y-%m-%d')  
 Time = datetime.datetime.fromtimestamp(ts).strftime('%H:%M:%S')  
 row = [Enrollment, Name, Date, Time]  
 with open('StudentDetails\StudentDetails.csv', 'a+') as csvFile:  
 writer = csv.writer(csvFile, delimiter=',')  
 writer.writerow(row)  
 csvFile.close()  
 res = "Images Saved for Enrollment : " + Enrollment + " Name : " + Name  
 Notification.configure(  
 text=res, bg="SpringGreen3", width=50, font=('times', 18, 'bold'))  
 Notification.place(x=250, y=400)  
 except FileExistsError as F:  
 f = 'Student Data already exists'  
 Notification.configure(text=f, bg="Red", width=21)  
 Notification.place(x=450, y=400)  
  
  
# for choose subject and fill attendance  
def subjectchoose():  
 def Fillattendances():  
 sub = tx.get()  
 now = time.time() # For calculate seconds of video  
 future = now + 20  
 if time.time() < future:  
 if sub == '':  
 err\_screen1()  
 else:  
 recognizer = cv2.face.LBPHFaceRecognizer\_create() # cv2.createLBPHFaceRecognizer()  
 try:  
 recognizer.read("TrainingImageLabel\Trainner.yml")  
 except:  
 e = 'Model not found,Please train model'  
 Notifica.configure(  
 text=e, bg="red", fg="black", width=33, font=('times', 15, 'bold'))  
 Notifica.place(x=20, y=250)  
  
 harcascadePath = "haarcascade\_frontalface\_default.xml"  
 faceCascade = cv2.CascadeClassifier(harcascadePath)  
 df = pd.read\_csv("StudentDetails\StudentDetails.csv")  
 cam = cv2.VideoCapture(0)  
 font = cv2.FONT\_HERSHEY\_SIMPLEX  
 col\_names = ['Enrollment', 'Name', 'Date', 'Time']  
 attendance = pd.DataFrame(columns=col\_names)  
 while True:  
 ret, im = cam.read()  
 gray = cv2.cvtColor(im, cv2.COLOR\_BGR2GRAY)  
 faces = faceCascade.detectMultiScale(gray, 1.2, 5)  
 for (x, y, w, h) in faces:  
 global Id  
  
 Id, conf = recognizer.predict(gray[y:y + h, x:x + w])  
 if (conf < 70):  
 print(conf)  
 global Subject  
 global aa  
 global date  
 global timeStamp  
 Subject = tx.get()  
 ts = time.time()  
 date = datetime.datetime.fromtimestamp(  
 ts).strftime('%Y-%m-%d')  
 timeStamp = datetime.datetime.fromtimestamp(  
 ts).strftime('%H:%M:%S')  
 aa = df.loc[df['Enrollment'] == Id]['Name'].values  
 global tt  
 tt = str(Id) + "-" + aa  
 En = '15624031' + str(Id)  
 attendance.loc[len(attendance)] = [  
 Id, aa, date, timeStamp]  
 cv2.rectangle(  
 im, (x, y), (x + w, y + h), (0, 260, 0), 7)  
 cv2.putText(im, str(tt), (x + h, y),  
 font, 1, (255, 255, 0,), 4)  
  
 else:  
 Id = 'Unknown'  
 tt = str(Id)  
 cv2.rectangle(  
 im, (x, y), (x + w, y + h), (0, 25, 255), 7)  
 cv2.putText(im, str(tt), (x + h, y),  
 font, 1, (0, 25, 255), 4)  
 if time.time() > future:  
 break  
  
 attendance = attendance.drop\_duplicates(  
 ['Enrollment'], keep='first')  
 cv2.imshow('Filling attedance..', im)  
 key = cv2.waitKey(30) & 0xff  
 if key == 27:  
 break  
  
 ts = time.time()  
 date = datetime.datetime.fromtimestamp(ts).strftime('%Y-%m-%d')  
 timeStamp = datetime.datetime.fromtimestamp(  
 ts).strftime('%H:%M:%S')  
 Hour, Minute, Second = timeStamp.split(":")  
 fileName = "Attendance/" + Subject + "\_" + date + \  
 "\_" + Hour + "-" + Minute + "-" + Second + ".csv"  
 attendance = attendance.drop\_duplicates(  
 ['Enrollment'], keep='first')  
 print(attendance)  
 attendance.to\_csv(fileName, index=False)  
  
 # Create table for Attendance  
 date\_for\_DB = datetime.datetime.fromtimestamp(  
 ts).strftime('%Y\_%m\_%d')  
 DB\_Table\_name = str(  
 Subject + "\_" + date\_for\_DB + "\_Time\_" + Hour + "\_" + Minute + "\_" + Second)  
 import pymysql.connections  
  
 # Connect to the database  
 try:  
 global cursor  
 connection = pymysql.connect(  
 host='localhost', user='root', password='', db='Face\_reco\_fill')  
 cursor = connection.cursor()  
 except Exception as e:  
 print(e)  
  
 sql = "CREATE TABLE " + DB\_Table\_name + """  
 (ID INT NOT NULL AUTO\_INCREMENT,  
 ENROLLMENT varchar(100) NOT NULL,  
 NAME VARCHAR(50) NOT NULL,  
 DATE VARCHAR(20) NOT NULL,  
 TIME VARCHAR(20) NOT NULL,  
 PRIMARY KEY (ID)  
 );  
 """  
 # Now enter attendance in Database  
 insert\_data = "INSERT INTO " + DB\_Table\_name + \  
 " (ID,ENROLLMENT,NAME,DATE,TIME) VALUES (0, %s, %s, %s,%s)"  
 VALUES = (str(Id), str(aa), str(date), str(timeStamp))  
 try:  
 cursor.execute(sql) # for create a table  
 # For insert data into table  
 cursor.execute(insert\_data, VALUES)  
 except Exception as ex:  
 print(ex) #  
  
 M = 'Attendance filled Successfully'  
 Notifica.configure(text=M, bg="Green", fg="white",  
 width=33, font=('times', 15, 'bold'))  
 Notifica.place(x=20, y=250)  
  
 cam.release()  
 cv2.destroyAllWindows()  
  
 import csv  
 import tkinter  
 root = tkinter.Tk()  
 root.title("Attendance of " + Subject)  
 root.configure(background='grey80')  
 cs = 'C:/Users/amans/finale/Face-Recognition-Attendance-System-main/Face-Recognition-Attendance-System-main/' + fileName  
 with open(cs, newline="") as file:  
 reader = csv.reader(file)  
 r = 0  
  
 for col in reader:  
 c = 0  
 for row in col:  
 # i've added some styling  
 label = tkinter.Label(root, width=10, height=1, fg="black", font=('times', 15, ' bold '),  
 bg="white", text=row, relief=tkinter.RIDGE)  
 label.grid(row=r, column=c)  
 c += 1  
 r += 1  
 root.mainloop()  
 print(attendance)  
  
 # windo is frame for subject chooser  
 windo = tk.Tk()  
 # windo.iconbitmap('AMS.ico')  
 windo.title("Enter subject name...")  
 windo.geometry('580x320')  
 windo.configure(background='grey80')  
 Notifica = tk.Label(windo, text="Attendance filled Successfully", bg="Green", fg="white", width=33,  
 height=2, font=('times', 15, 'bold'))  
  
 def Attf():  
 import subprocess  
 subprocess.Popen(  
 r'explorer /select,"C:/Users/amans/finale/Face-Recognition-Attendance-System-main/Face-Recognition-Attendance-System-main/Attendance/-------Check atttendance-------"')  
  
 attf = tk.Button(windo, text="Check Sheets", command=Attf, fg="white", bg="black",  
 width=12, height=1, activebackground="white", font=('times', 14, ' bold '))  
 attf.place(x=430, y=255)  
  
 sub = tk.Label(windo, text="Enter Subject : ", width=15, height=2,  
 fg="black", bg="grey", font=('times', 15, ' bold '))  
 sub.place(x=30, y=100)  
  
 tx = tk.Entry(windo, width=20, bg="white",  
 fg="black", font=('times', 23))  
 tx.place(x=250, y=105)  
  
 fill\_a = tk.Button(windo, text="Fill Attendance", fg="white", command=Fillattendances, bg="SkyBlue1", width=20, height=2,  
 activebackground="white", font=('times', 15, ' bold '))  
 fill\_a.place(x=250, y=160)  
 windo.mainloop()  
  
  
def admin\_panel():  
 win = tk.Tk()  
 # win.iconbitmap('AMS.ico')  
 win.title("LogIn")  
 win.geometry('880x420')  
 win.configure(background='grey80')  
  
 def log\_in():  
 username = un\_entr.get()  
 password = pw\_entr.get()  
  
 if username == 'aman':  
 if password == 'aman123':  
 win.destroy()  
 import csv  
 import tkinter  
 root = tkinter.Tk()  
 root.title("Student Details")  
 root.configure(background='grey80')  
  
 cs = 'C:/Users/amans/finale/Face-Recognition-Attendance-System-main/Face-Recognition-Attendance-System-main/StudentDetails/StudentDetails.csv'  
 with open(cs, newline="") as file:  
 reader = csv.reader(file)  
 r = 0  
  
 for col in reader:  
 c = 0  
 for row in col:  
 # i've added some styling  
 label = tkinter.Label(root, width=10, height=1, fg="black", font=('times', 15, ' bold '),  
 bg="white", text=row, relief=tkinter.RIDGE)  
 label.grid(row=r, column=c)  
 c += 1  
 r += 1  
 root.mainloop()  
 else:  
 valid = 'Incorrect ID or Password'  
 Nt.configure(text=valid, bg="red", fg="white",  
 width=38, font=('times', 19, 'bold'))  
 Nt.place(x=120, y=350)  
  
 else:  
 valid = 'Incorrect ID or Password'  
 Nt.configure(text=valid, bg="red", fg="white",  
 width=38, font=('times', 19, 'bold'))  
 Nt.place(x=120, y=350)  
  
 Nt = tk.Label(win, text="Attendance filled Successfully", bg="Green", fg="white", width=40,  
 height=2, font=('times', 19, 'bold'))  
 # Nt.place(x=120, y=350)  
  
 un = tk.Label(win, text="Enter username : ", width=15, height=2, fg="black", bg="grey",  
 font=('times', 15, ' bold '))  
 un.place(x=30, y=50)  
  
 pw = tk.Label(win, text="Enter password : ", width=15, height=2, fg="black", bg="grey",  
 font=('times', 15, ' bold '))  
 pw.place(x=30, y=150)  
  
 def c00():  
 un\_entr.delete(first=0, last=22)  
  
 un\_entr = tk.Entry(win, width=20, bg="white", fg="black",  
 font=('times', 23))  
 un\_entr.place(x=290, y=55)  
  
 def c11():  
 pw\_entr.delete(first=0, last=22)  
  
 pw\_entr = tk.Entry(win, width=20, show="\*", bg="white",  
 fg="black", font=('times', 23))  
 pw\_entr.place(x=290, y=155)  
  
 c0 = tk.Button(win, text="Clear", command=c00, fg="white", bg="black", width=10, height=1,  
 activebackground="white", font=('times', 15, ' bold '))  
 c0.place(x=690, y=55)  
  
 c1 = tk.Button(win, text="Clear", command=c11, fg="white", bg="black", width=10, height=1,  
 activebackground="white", font=('times', 15, ' bold '))  
 c1.place(x=690, y=155)  
  
 Login = tk.Button(win, text="LogIn", fg="black", bg="SkyBlue1", width=20,  
 height=2,  
 activebackground="Red", command=log\_in, font=('times', 15, ' bold '))  
 Login.place(x=290, y=250)  
 win.mainloop()  
  
  
# For train the model  
def trainimg():  
 recognizer = cv2.face.LBPHFaceRecognizer\_create()  
 global detector  
 detector = cv2.CascadeClassifier("haarcascade\_frontalface\_default.xml")  
 try:  
 global faces, Id  
 faces, Id = getImagesAndLabels("TrainingImage")  
 except Exception as e:  
 l = 'please make "TrainingImage" folder & put Images'  
 Notification.configure(text=l, bg="SpringGreen3",  
 width=50, font=('times', 18, 'bold'))  
 Notification.place(x=350, y=400)  
  
 recognizer.train(faces, np.array(Id))  
 try:  
 recognizer.save("TrainingImageLabel\Trainner.yml")  
 except Exception as e:  
 q = 'Please make "TrainingImageLabel" folder'  
 Notification.configure(text=q, bg="SpringGreen3",  
 width=50, font=('times', 18, 'bold'))  
 Notification.place(x=350, y=400)  
  
 res = "Model Trained" # +",".join(str(f) for f in Id)  
 Notification.configure(text=res, bg="olive drab",  
 width=50, font=('times', 18, 'bold'))  
 Notification.place(x=250, y=400)  
  
  
def getImagesAndLabels(path):  
 imagePaths = [os.path.join(path, f) for f in os.listdir(path)]  
 # create empth face list  
 faceSamples = []  
 # create empty ID list  
 Ids = []  
 # now looping through all the image paths and loading the Ids and the images  
 for imagePath in imagePaths:  
 # loading the image and converting it to gray scale  
 pilImage = Image.open(imagePath).convert('L')  
 # Now we are converting the PIL image into numpy array  
 imageNp = np.array(pilImage, 'uint8')  
 # getting the Id from the image  
  
 Id = int(os.path.split(imagePath)[-1].split(".")[1])  
 # extract the face from the training image sample  
 faces = detector.detectMultiScale(imageNp)  
 # If a face is there then append that in the list as well as Id of it  
 for (x, y, w, h) in faces:  
 faceSamples.append(imageNp[y:y + h, x:x + w])  
 Ids.append(Id)  
 return faceSamples, Ids  
  
  
window.grid\_rowconfigure(0, weight=1)  
window.grid\_columnconfigure(0, weight=1)  
# window.iconbitmap('AMS.ico')  
  
  
def on\_closing():  
 from tkinter import messagebox  
 if messagebox.askokcancel("Quit", "Do you want to quit?"):  
 window.destroy()  
  
  
window.protocol("WM\_DELETE\_WINDOW", on\_closing)  
  
message = tk.Label(window, text="Face-Recognition-Based-Attendance-Management-System", bg="black", fg="white", width=50,  
 height=3, font=('times', 30, ' bold '))  
  
message.place(x=80, y=20)  
  
Notification = tk.Label(window, text="All things good", bg="Green", fg="white", width=15,  
 height=3, font=('times', 17))  
  
lbl = tk.Label(window, text="Enter Enrollment : ", width=20, height=2,  
 fg="black", bg="grey", font=('times', 15, 'bold'))  
lbl.place(x=200, y=200)  
  
  
def testVal(inStr, acttyp):  
 if acttyp == '1': # insert  
 if not inStr.isdigit():  
 return False  
 return True  
  
  
txt = tk.Entry(window, validate="key", width=20, bg="white",  
 fg="black", font=('times', 25))  
txt['validatecommand'] = (txt.register(testVal), '%P', '%d')  
txt.place(x=550, y=210)  
  
lbl2 = tk.Label(window, text="Enter Name : ", width=20, fg="black",  
 bg="grey", height=2, font=('times', 15, ' bold '))  
lbl2.place(x=200, y=300)  
  
txt2 = tk.Entry(window, width=20, bg="white",  
 fg="black", font=('times', 25))  
txt2.place(x=550, y=310)  
  
clearButton = tk.Button(window, text="Clear", command=clear, fg="white", bg="black",  
 width=10, height=1, activebackground="white", font=('times', 15, ' bold '))  
clearButton.place(x=950, y=210)  
  
clearButton1 = tk.Button(window, text="Clear", command=clear1, fg="white", bg="black",  
 width=10, height=1, activebackground="white", font=('times', 15, ' bold '))  
clearButton1.place(x=950, y=310)  
  
AP = tk.Button(window, text="Check Registered students", command=admin\_panel, fg="black",  
 bg="SkyBlue1", width=19, height=1, activebackground="white", font=('times', 15, ' bold '))  
AP.place(x=990, y=410)  
  
takeImg = tk.Button(window, text="Take Images", command=take\_img, fg="black", bg="SkyBlue1",  
 width=20, height=3, activebackground="white", font=('times', 15, ' bold '))  
takeImg.place(x=90, y=500)  
  
trainImg = tk.Button(window, text="Train Images", fg="black", command=trainimg, bg="SkyBlue1",  
 width=20, height=3, activebackground="white", font=('times', 15, ' bold '))  
trainImg.place(x=390, y=500)  
  
FA = tk.Button(window, text="Automatic Attendance", fg="black", command=subjectchoose,  
 bg="SkyBlue1", width=20, height=3, activebackground="white", font=('times', 15, ' bold '))  
FA.place(x=690, y=500)  
  
quitWindow = tk.Button(window, text="Manually Fill Attendance", command=manually\_fill, fg="black",  
 bg="SkyBlue1", width=20, height=3, activebackground="white", font=('times', 15, ' bold '))  
quitWindow.place(x=990, y=500)  
  
window.mainloop()

# CONCLUSION

Conclusion of the Project

The Face Recognition Attendance System using Haar Code is an advanced solution that leverages facial recognition technology for efficient and accurate attendance management. It offers numerous advantages such as increased automation, enhanced accuracy, improved security, and streamlined administrative processes. By eliminating manual attendance tracking methods, the system reduces human error and saves valuable time and resources.

Throughout the development and implementation of the system, various aspects were considered. The operational feasibility analysis confirmed that the system is practical and beneficial for the organization, while the technical feasibility assessment ensured that the required technology and resources are available. The economical feasibility evaluation demonstrated that the system provides a cost-effective solution, considering the long-term benefits and return on investment.

During the testing phase, both black box and white box testing techniques were employed to verify the system's functionality, performance, usability, and security. These testing methods helped uncover and address any issues or defects, ensuring that the system meets the specified requirements and delivers a reliable user experience.

Overall, the Face Recognition Attendance System using Haar Code presents a robust and efficient solution for attendance management. It simplifies the process, enhances accuracy, and improves overall efficiency for organizations of various sizes. By embracing this technology, businesses can effectively streamline their attendance tracking process, leading to better resource management, improved productivity, and enhanced data security.

# Limitation of the system

While the Face Recognition Attendance System using Haar Code offers numerous benefits, it also has certain limitations that should be considered. These limitations include:

1. Environmental Factors: The system's performance can be affected by environmental factors such as lighting conditions, camera quality, and angle of the captured images. In suboptimal conditions, the accuracy of face recognition may decrease, leading to potential errors in attendance tracking.
2. Variability in Facial Features: The system may face challenges when dealing with variations in facial features, such as changes in hairstyle, facial hair, or accessories. These changes can impact the accuracy of face recognition and result in false negatives or false positives.
3. Security and Privacy Concerns: Facial recognition technology raises concerns about privacy and data security. The system must ensure the secure storage and handling of biometric data to prevent unauthorized access or misuse. Organizations must comply with relevant data protection regulations and implement robust security measures to safeguard sensitive information.
4. Bias and Diversity: Facial recognition algorithms can be susceptible to bias, particularly when it comes to different ethnicities, ages, and gender identities. It is crucial to carefully evaluate and train the system to minimize bias and ensure fairness in attendance tracking.
5. Scalability and Resource Requirements: As the number of users and attendance records grows, the system's scalability and resource requirements may increase. Adequate hardware infrastructure and computational resources are necessary to handle large datasets and maintain system performance.
6. User Acceptance and Familiarity: Introducing a new attendance system based on facial recognition may require users to adapt to new processes and technologies. User acceptance and familiarity with the system may vary, requiring appropriate training, education, and support to ensure smooth adoption.
7. System Reliance and Redundancy: The system heavily relies on accurate face recognition for attendance tracking. In the event of technical failures, such as system downtime or malfunctions, alternative attendance tracking methods or backup systems should be in place to maintain operational continuity.
8. Ethical Considerations: Facial recognition technology raises ethical concerns regarding privacy, consent, and potential misuse. Organizations must address these considerations and establish clear policies and guidelines for the ethical use of the system.

It is essential to evaluate these limitations and mitigate potential risks during the design, implementation, and operation of the Face Recognition Attendance System using Haar Code. Proactive measures, continuous monitoring, and regular updates can help address these limitations and ensure the system's effectiveness and ethical usage.

# Future Scope of the Project

The Face Recognition Attendance System using Haar Code holds significant potential for future development and expansion. Here are some potential future scopes for the system:

1. Advanced Facial Recognition Algorithms: Continuous advancements in facial recognition algorithms can enhance the accuracy and reliability of the system. Implementing cutting-edge algorithms, such as deep learning-based approaches, can improve facial recognition performance, even in challenging conditions.
2. Integration with Biometric Authentication: The system can be integrated with other biometric authentication methods, such as fingerprint scanning or iris recognition, to provide multi-factor authentication and enhance security.
3. Real-Time Analytics and Insights: Adding real-time analytics capabilities to the system can enable organizations to gain valuable insights from attendance data. This can help identify attendance patterns, detect anomalies, and generate attendance reports in real-time, facilitating better decision-making.
4. Mobile Application Support: Developing a mobile application for the system can provide convenient and flexible attendance tracking options for users. Employees can use their smartphones to capture attendance, view records, and receive notifications, increasing accessibility and user engagement.
5. Cloud-Based Deployment: Moving the system to a cloud-based infrastructure offers scalability, flexibility, and cost-efficiency. Cloud deployment enables easy access to the system from anywhere, enhances data security and backup capabilities, and allows for seamless integration with other cloud-based services.
6. Facial Emotion Recognition: Expanding the system to include facial emotion recognition can provide additional insights into user engagement and satisfaction. This feature can be leveraged in various applications, such as monitoring student engagement in classrooms or assessing customer satisfaction in retail environments.
7. Integration with Attendance Management Systems: Integrating the face recognition system with existing attendance management systems or human resource management systems can streamline the overall attendance tracking and payroll processes. This integration enables automatic data synchronization and eliminates manual data entry.
8. Enhanced Security Features: Implementing advanced security features, such as liveness detection to prevent spoofing or tampering attempts, can further enhance the system's security and reliability.
9. Integration with Access Control Systems: Integrating the face recognition system with access control systems can provide a seamless experience for users. Employees' faces can serve as access credentials for restricted areas, improving security and eliminating the need for separate access cards or passwords.
10. Continuous Performance Optimization: Regular performance evaluation and optimization can ensure that the system maintains high accuracy, speed, and reliability. This includes fine-tuning the algorithms, hardware upgrades, and periodic system updates.

The future scope of the Face Recognition Attendance System using Haar Code lies in leveraging advancements in technology, expanding functionality, and addressing emerging needs and challenges in attendance management. By embracing these opportunities, organizations can further optimize their attendance tracking processes and improve overall efficiency.

# BIBLIOGRAPHY

**Research Papers**

* Face Recognition based Attendance System using Haar Cascade and Local Binary Pattern Histogram Algorithm- **Published in:**[2020 4th International Conference on Trends in Electronics and Informatics (ICOEI)(48184)](https://ieeexplore.ieee.org/xpl/conhome/9138108/proceeding)
* Jomon Joseph and K. P. Zacharia, "Automatic Attendance Management System Using Face Recognition", International Journal of Science and Research (IJSR), 2013.
* R. Samet and M. Tanriverdi, "Face Recognition-Based Mobile Automatic Classroom Attendance Management System", 2017 International Conference on Cyberworlds (CW), pp. 253-256, 2017.
* Attendance System Based on Face Recognition Using Haar Cascade and LBPH Algorithm [Akshat Kumar Rai](https://link.springer.com/chapter/10.1007/978-981-19-5936-3_2#auth-Akshat_Kumar-Rai), [A. Akash](https://link.springer.com/chapter/10.1007/978-981-19-5936-3_2#auth-A_-Akash), [G. Kavyashree](https://link.springer.com/chapter/10.1007/978-981-19-5936-3_2#auth-G_-Kavyashree) & [Thaseen Taj](https://link.springer.com/chapter/10.1007/978-981-19-5936-3_2" \l "auth-Thaseen-Taj)

Conference paper [First Online: 07 February 2023](https://link.springer.com/chapter/10.1007/978-981-19-5936-3_2#chapter-info)

* Comprehensive approach of real-time web-based face recognition system using Haar Cascade and LBPH algorithm **Published in:**[2023 International Conference on Device Intelligence, Computing and Communication Technologies, (DICCT)](https://ieeexplore.ieee.org/xpl/conhome/10110034/proceeding)

sss

* <https://www.wikipedia.org>
* <https://www.researchgate.net/publication/343038386_Face_Recognition_based_Attendance_System_using_Haar_Cascade_and_Local_Binary_Pattern_Histogram_Algorithm>
* <https://www.slideshare.net/>

Text Book

* **Computer Vision: Algorithms and Applications-** by [Richard Szeliski](http://szeliski.org/RichardSzeliski.htm)

|  |
| --- |
| * [Artificial Intelligence](http://geni.us/NDIB)- By Stuart Russell |