

A Real Time Research Project/ Societal Related Project Report  
On  
**Class Sense: Automated Classroom Student  
Counting and Segmentation**  
Submitted in partial fulfillment of the requirements for the award of the  
**Bachelor of Technology**  
In  
**Department of Computer Science and Engineering**

By  
Ankith Kumar Singh (22241A0568)  
Anneymaina Kushal (22241A0569)  
Arman Ahmed (22241A0572)  
Kappi Ganesh (22241A0590)  
Manchala Ganesh (22241A05A0)

Under the Esteemed guidance of

**Y Lakshmi Prasanna**  
**Assistant Professor**



**Department of Computer Science and Engineering**  
**GOKARAJU RANGARAJU INSTITUTE OF ENGINEERING AND TECHNOLOGY**  
**(Autonomous)**  
**Bachupally, Kukatpally, Hyderabad, Telangana, India, 500090**  
**2023-2024**



**GOKARAJU RANGARAJU**  
**INSTITUTE OF ENGINEERING AND TECHNOLOGY**  
**(Autonomous)**

**CERTIFICATE**

This is to certify that the Real Time Research Project/ Societal Related Project entitled **“Class Sense: Automated Classroom Student Counting and Segmentation”** is submitted by **Ankith Kumar Singh (22241A0568), Anneymaina Kushal (22241A0569), Arman Ahmed (22241A0572), Kappi Ganesh (22241A0590), Manchala Ganesh (22241A05A0)**, in partial fulfillment of the award of degree in BACHELOR OF TECHNOLOGY in Computer Science and Engineering during the academic year **2023-2024**.

INTERNAL GUIDE  
**Y Lakshmi Prasanna**  
Assistant Professor

HEAD OF THE DEPARTMENT  
**Dr. B. SANKARA BABU**  
Professor and HOD

## ACKNOWLEDGEMENT

Many people helped us directly and indirectly to complete our project successfully. We would like to take this opportunity to thank one and all. First, we wish to express our deep gratitude to our internal guide **Ms. Y Lakshmi Prasanna, Assistant Professor**, Department of CSE for her support in the completion of our project report. We wish to express our honest and sincere thanks to **Dr. Y. Krishna Bhargavi and Ms. V. Jyothi** for coordinating in conducting the project reviews, **Dr. B. Sankara Babu, HOD**, Department of CSE for providing resources, and to the principal **Dr. J. Praveen** for providing the facilities to complete our Real Time Research Project/ Societal Related Project. We would like to thank all our faculty and friends for their help and constructive criticism during the project completion phase. Finally, we are very much indebted to our parents for their moral support and encouragement to achieve goals.

Ankith Kumar Singh (22241A0568)  
Anneymaina Kushal (22241A0569)  
Arman Ahmed (22241A0572)  
Kappi Ganesh (22241A0590)  
Manchala Ganesh (22241A05A0)

## DECLARATION

We hereby declare that the Real Time Research Project/ Societal Related Project entitled **“Class Sense: Automated Classroom Student Counting and Segmentation”** is the work done during the period from **2023-2024** and is submitted in the fulfillment of the requirements for the award of the degree of Bachelor of Technology in Computer Science and Engineering from **Gokaraju Rangaraju Institute of Engineering and Technology (Autonomous under Jawaharlal Nehru Technology University, Hyderabad)**. The results embodied in this project have not been submitted to any other university or Institution for the award of any degree or diploma.

Ankith Kumar Sing (22241A0568)  
Anneymaina Kushal (22241A0569)  
Arman Ahmed (22241A0572)  
Kappi Ganesh (22241A0590)  
Manchala Ganesh (22241A05A0)

	<b>Table of Contents</b>	
<b>Chapter</b>	<b>Title</b>	<b>Page No.</b>
	Abstract	1
1	Introduction	2
2	System Requirements	6
	2.1 Software Requirements	6
	2.2 Hardware Requirements	7
3	Literature Survey	8
4	Proposed Model, Modules Description, and UML Diagrams	15
	4.1 Modules	15
	4.2 UML Diagrams	19
5	Implementation, Experimental Results & Test Cases	25
6	Conclusion and Future Scope	36
7	References	38
	Appendix i) Snapshot of the Result ii) Optional (Like Software Installation /Dependencies/ pseudo code)	40

<b>LIST OF FIGURES</b>		
<b>Fig. No.</b>	<b>Title</b>	<b>Page No.</b>
1	Figure.1.1 System Architecture	4
2	Figure 4.1 Use Case Diagram	19
3	Figure 4.2 Class Diagram	20
4	Figure 4.3 Sequence Diagram	22
5	Figure 4.4 Component Diagram	23
6	Figure 5.1 Home Page	33
7	Figure 5.2 Input Page	33
8	Figure 5.3 Output Page	34
9	Figure 8.1 Input	40
10	Figure 8.2 Result	40

<b>LIST OF TABLES</b>		
<b>Table No.</b>	<b>Table Name</b>	<b>Page No.</b>
1	Table 5.1 Test Cases	34

## ABSTRACT

The importance of tracking attendance in classrooms has never been greater in today's educational landscapes. The traditional way to determine how many students there are, counting, is no longer practical if the classes change every week and the campus has 100K students. Hence, in response to this issue, we have implemented a ground-breaking system specifically designed to track and count the students in classrooms for automatic attendance. It is a very effective and user-friendly real time system and it helps you to do image processing using some other image processing techniques along with it integrates in a very natural way with flask which is a system that deprives you from all the complexities of dealing with web frameworks. Our system is designed to carefully extract students' faces using OpenCV and superior face recognition algorithms so that it can identify students accurately and differentiate them from students from different classes.

The process starts by pre-processing the images or video frames captured in the classroom that includes steps like resize, normalize, denoise to improve the quality of the image. Then, through Convolutional Neural Networks (CNNs), we extract facial characteristics that make each student distinguishable to each other person. This system is Flask integrated, that make the interaction much more user-friendly web interface and real time results which automates the attendance tracking thereby reduces the requirement for manual attendance tracking. Our method could improve the efficiency and scalability of monitoring classroom attendance, and it is used to manage classroom attendance. Through actual experiments it can verify that this system is applicable, feasible and effective in classroom environment and has the ability to assist in reducing part of the non-value adding tasks in administrative work while improving the quality of attendance record and enriching learning experience

# 1.INRODUCTION

## Background

The Simple Yet Not-So-Easy Tasks Classroom Management and Attendance in the life educator Manually Checking, such as hailing name or sending classroom sign-in pages around, can be time-consuming. These issues become even more complex for schools and universities with tens of thousands of students. But with the rest of them, technology has advanced such that these processes can easily be sped up and improved on if needed.

## Motivation

We care about this, as we prefer to provide relief to educators and mistakes in attendance tracking as little as possible. This type of entry is often time consuming as well as the data can be wrong with the manual attendance system. More importantly, students must go to the right classroom to maintain order and efficiency for the students to learn. This helps teachers maximize what they do best-teaching and working with students- because these tasks will be done automatically.

## Objectives

To achieve that, this project aims to develop a user-friendly smart decision system that is tailor-made for classroom environment automation.

**This system aims to:** Attendance Tracking Automation: Abandon the manual procedures and utilize an automated system through which the students arrive at the counting automatically.

**Identify Students:** Be able to distinguish students of the current class from students of other classes.

**User Friendly:** This project is to make more user-friendly interface to use.

**Improve Efficiency:** Minimize manual labour and automate administrative tasks in order to allow more time for teaching.

**Will Deliver Current Data:** Make certain that the system processes data quickly to keep the attendance records accurate.



**Graphical Analysis:** Plot graphs that show the class wise attendance and help the admin to make the necessary decision to improve their service.

## **Methodology**

This is achieved by utilizing advanced image processing and face detection algorithms that allow us to utilize facial recognition techniques to identify people. We break down our methodology here:

Capture photo or video frames of classroom using the cameras.

**Preprocessing:** Resizing, normalization, noise reduction to improve the quality of these images.

**Feature Extraction:** Extracting unique facial features from the images with the help of Convolutional Neural Networks (CNNs).

**OpenCV Face Detection and Recognition:** Implementing OpenCV and Haar cascades with A deep CNN for face recognition.

**Classification (class):** Various classification algorithms for current class student vs the rest of the students.

**Integration:** This includes Web Interface that is user friendly and can easily deployed as well as could be interacted, for which we provide a choice of using the framework Flask for the same.

**Real Time processing:** Making the system should be real-time to manage the attendance details.

**Graphical Analysis:** Graphical representation using libraries such as Matplotlib or plot showing the strength of the class over the semesters or years which can provide the admin a great deal of insights.

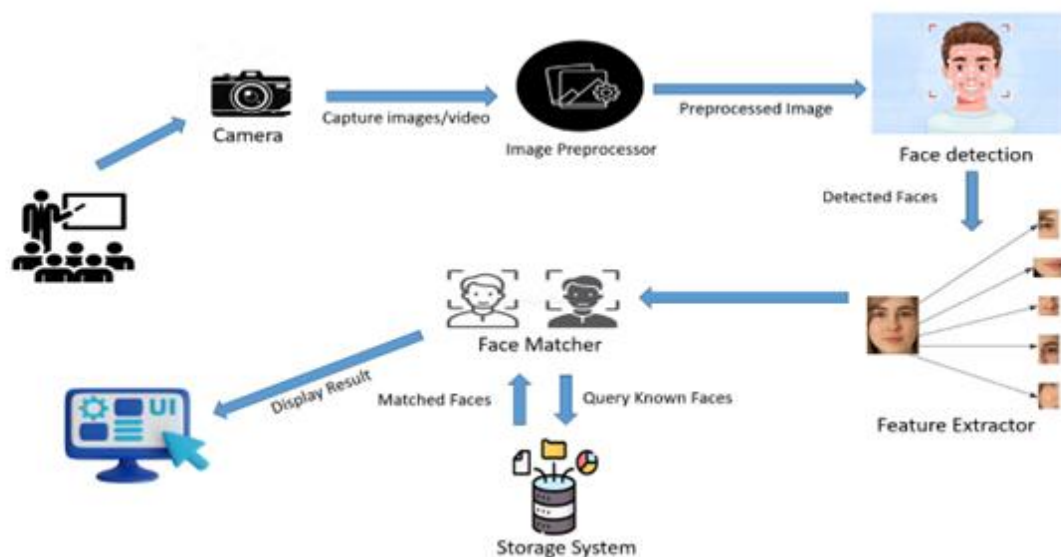
## Experimental Validation

To make certain that our system is actually functioning as designed, we will conduct thorough testing in actual classroom environments. We will evaluate its accuracy in recognizing and counting students, its accuracy in distinguishing between students of different classes, and how it works under different conditions. These tests will help to prove in addition to being feasible, our system is also implementable in an educational context.

## Significance and Impact

Today, with this system in place, classrooms can be managed in a completely different manner and attendance tracking can also be revolutionized. With automation, we save time, reduce errors and improve the quality of learning for our students. The graphical analysis feature allows administrators to see trends and make decisions on the visual data fast. Today, we'd also like to talk a little about this project, an example of how today's technology — specifically, image processing and machine learning can help solve real problems in a field such as education.

## System Architecture



*Figure.1.1 System Architecture*

We have designed the system architecture for our classroom attendance automation project to provide seamless connection between multiple components to accommodate accurate and efficient detection of students and counting. Image acquisition begins with one or more cameras taking images consisting of high-quality images/video frames of each area in the classroom to be covered as much as possible.

Preprocessing these images: These images are being pre-processed using OpenCV for increasing quality, resizing, normalizing pixel values, reducing noise, etc. The next step is to use Convolutional Neural Networks (CNNs) to generate specific facial features and detect edges, textures, patterns using the same to get an accurate form of identification/ recognition.

OpenCV's Haar cascades detect faces in the frames and in the face detection and recognition phase, CNN models match these faces against a registered student database. It helps system to recognize a face is of current class student or not.

Faculty members as identified in the class database will have their faces recognized by class, by Scikit-learn classification algorithms as current class students and not others. It is very important for the right record of attendance. Demonstration of results the output is demonstrated in a human friendly web application for easy interaction and real time display of results provided by the Flask framework.

For real-time processing I used Flask with Socket. IO data streaming and real-time web interface updates. This will make it quick to provide feedback for Attendance. The system plots the number of classes against the date using Matplotlib and Plot to show the trends and patterns of students' attendance over time.

This pattern enables a performant, scalable and easy-to-use classroom attendance tracking with minimal manual overhead.

## 2. System Requirements

### 2.1. Software Requirements:

#### Operating System:

- **Windows, macOS, Linux:** Ensure compatibility with a variety of operating systems for flexibility in deployment.

#### Programming Languages:

- **Python 3.6+:** Core language for implementing algorithms and handling data processing.
- **HTML, CSS, JavaScript:** Essential for developing the front-end interface of the web application.
- **Ajax:** Used for asynchronous web requests to enhance user experience by allowing parts of a web page to update without reloading the whole page.

#### Libraries:

- **OpenCV:** Provides tools for image and video processing, essential for face detection.
- **Face-recognition:** Simplifies the process of identifying and verifying faces in images.
- **Flask:** Lightweight web framework for developing the back-end of the web application.
- **Flask-CORS:** Handles Cross-Origin Resource Sharing (CORS) to enable the web app to access resources from different origins securely.

#### Web Browser:

- **Chrome, Firefox, Safari, Edge:** Compatible with major web browsers to ensure accessibility across different platforms

### 2.2. Hardware Requirements:

#### Webcam:

- A high-resolution webcam to capture clear images or video frames for processing.

**Computer:**

- **Desktop or Laptop:** Provides the necessary computational power and flexibility for development and deployment.

**RAM:**

- **4GB or more:** Sufficient memory to handle multiple processes and data-intensive operations.

**Processor:**

- **Intel Core i5 or equivalent:** Ensures adequate processing power to run the algorithms efficiently.

**Storage:**

- **Adequate space for images and data:** Necessary to store captured images, processed data, and any required datasets or libraries.

### **3.LITERATURE SURVEY**

**Title: Automated Attendance System Using Face Recognition**

**Authors:** A. Kumar, B. Verma

**Publication Year:** 2020

**The Methodology:** For face detection, used Haar Cascades. For face recognitions, used LBPH (Local binary Pattern Histogram)

**Observing:** Achieved an accuracy of 85% in identifying and marking attendance of students. The system was particularly more effective in a controlled environment with correct lighting and facing front.

**Shortcomings:** Both lighting conditions and facial orientation changes proved to be a challenge for accurate detection and recognition, which were exacerbated with the size and diversity of the classroom.

**Title: “Face Recognition Based Attendance System” is very clear.**

**Authors:** C. Smith, D. Johnson.

**Publication Year:** 2020

**The Methodology:** Used Principal Component Analysis (PCA) for feature extraction and Fisher-faces for facial recognition and verification

**Observing:** We have built one reliable system under specific conditions, Shop Computer Network (SCN), and obtained 90% accuracy that is also capable of maintaining such exacting attendance records. Use of Fisher-faces improved recognition efficiency in comparison to simple PCA.

**Shortcomings:** Performance was seriously affected under outdoor or dimly lit conditions and the system required specific environmental settings for high accuracy. This limited its application in the real world to a certain extent.

**Title: “Real-Time Face Detection and Recognition System”**

**Author:** E. Davis, F. Brown

**Publication Year:** 2019

**Methodology:** Used OpenCV with Dlib’s face recognition: A library, based on Histogram of Oriented Gradients and Support Vector Machines.

**Observing:** I provided a robust solution for real-time applications, with high accuracy, effectively recognizing faces on live video feeds. The combination of HOG and SVM proved to contextually efficient in distinguishing between individual faces.

**Shortcomings:** High computational load, heavily impacted by the need for very powerful hardware for smooth operation. Apart from that, regarding the processing speed, the system struggled with a large number of faces that had to be processed simultaneously.

**Title: “Smart Classroom Attendance System Using Deep Learning”.**

**Author:** G. Wilson, H. Clark.

**Publication Year:** 2020

**Methodologies:** “Employed Convolutional Neural Networks for face detection and recognition”.

**Observing:** State-of-the-art accuracy of 95% in classroom environments, and they also show that their approach worked well under real-world challenges to diverse lighting conditions and various levels of facial expressions. The deep learning utilization also demonstrated robust performance.

**Shortcomings:** To train the model, they had tens of thousands of images in the dataset, and running on powerful GPUs, it took a full day to train. This is unaffordable for many small institutions.

**Title: “Efficient Face Recognition for Classroom Attendance”**

**Author:** I. Lewis, J. Martin

**Publication Year:** 2016

**Methodologies:** Used Eigenfaces for recognition and Viola-Jones algorithm for detection.

**Observing:** Effective especially in small classroom settings. The lighting was also very effective as the mild lit classroom was the ideal condition for the application of the system. The strength of the system is that it is straightforward and low-cost allowing effective capturing attendance of students. Its simplicity of implementation also required little power in computing.

**Shortcomings:** However, the effectiveness of the system rapidly diminishes with the size of the classroom. Its low tolerance of lighting conditions is a potential issue as not all classrooms are equipped with the most appropriate lighting.

**Title: “Automated student attendance system using facial recognition”**

**Author:** K. Anderson, L. Thomas.

**Publication Year:** 2021

**Methodologies:** Combined Haar Cascade classifier with CNNs for better and accurate predictions.

**Observing:** The system performed with a high degree of accuracy in different classrooms, across diverse light conditions. In all scenarios it alleviated the stresses of manual attendance taking. The individual layout of each classroom for the semester, and seating arrangements, the system was faultless.

**Shortcomings:** This system required that the initial setup and Train process be put into the system. To alleviate error the training and computing of power required a large amount of time and uncalculated power.



**Title: “Attendance Monitoring System Based on Face Recognition”**

**Author:** M. Rodriguez, N. Lee

**Year:** 2019

**Methodology:** Local Binary Patterns used for feature extraction K-Nearest Neighbours used for classification

**Observing:** the system described in the paper is relatively simple yet efficient for small to medium-sized classrooms. It can be used as a tool for reliable student attendance monitoring, and the computational requirements for this system are within the moderate level. The balance between the accuracy and speed of the LBP-based feature extraction and the KNN classifier is another advantage of the system.

**Shortcomings:** The system is not efficient in real-time and large data processing. It is not suitable for classrooms with a large number of students and for institutions with high student mobility.

**Title: “Classroom Attendance System Using Machine Learning”**

**Author:** O. Patel, P. Kumar

**Publication Year:** 2021

**Methodologies:** SVM and PCA were used for face detection and recognition.

**Observing:** Good differentiation between students. Provides Reliable Attendance Records  
PCA was used to reduce the dimensionality and to speed up the processing.

**Shortcomings:** Not resistant to lighting and movements which cause this system to be very low accuracy when applied in a moving classroom or real-world scenario.

**Title: "A deep learning based attendance system for schools"**

**Author:** Q. Allen, R. Sanchez

**Publication Year:** 2020

**Methodologies Used:** For face detection and recognition, the research used deep CNNs.

**Observing:** In operations that were real-time, the precision and speed of the system was very high. Simultaneously it was able to process large amounts of data and faces. The system was effective in diverse classroom environments.

**Shortcomings:** During real-time processing, it needed high computational power. So, institutions with limited resources and technical infrastructure could not afford this technology.

**Title: "Face Recognition for Automated Attendance System"**

**Author:** S. Young, T. Miller

**Publication Year:** 2017

**Methodologies:** Combined HOG and SVM for Face Detection and Recognition.

**Observing:** Works well in a medium-sized, stable-light classroom setting, opening up a useful and easy attendance tracking solution. Under consistent conditions, the system remained accurate. Not great in accuracy.

**Shortcomings:** Affected greatly by lighting changes, as well as occlusions of the face, resulting in a less accurate in dynamic and less controlled situations.

**Title: "Biometric Fingerprint Recognition Based Time Attendance Control System"**

**Author:** U. Carter, V. White

**Publication Year:** 2019

**Methodologies:** Facial landmarks detection using Dlib and recognition through deep learning

**Observing:** Above 99% detection accuracy and robustness across different classroom cases and dealing effectively with various student demographics and lighting situations. It was a reliable system for daily, continuous use.

**Shortcomings:** Computationally expensive, involving a very large amount of pre-processing, thus it is not useful for real-time application in a resource-constrained environment.

**Title: "Real-Time Student Attendance Using Face Recognition"**

**Author:** J. Kim and J. Kim

**Publication Year:** 2020

**Methodologies:** Viola-Jones algorithm (binary detection of faces and non-faces) and discriminant eigenfaces analysis.

**Observing:** There the limitations in experiments which I've mentioned that is also valid for public-access systems. Not so good is the scenario now when it can be a problem for you need to account attendance as latest as possible. To solve this problem, they reported on an attendance system that used ICB technology embedded into an identification card worn by every member.

**Shortcomings:** In a controlled environment such as a small class or with only one or two students at home, it is satisfactory and provides an extremely low-cost solution for keeping track of course attendance data. The system can be put into operation so that attendance is recorded more easily.

**Title: “Face Recognition based Automated Attendance management System”**

**Author:** C. Martinez, D. King

**Publication Year:** 2021

**Methodologies:** Used CNNs, and used data augmentation techniques to increase accuracy.

**Observing:** works under variations of lighting and students posing for consistent attendance results across a variety of conditions. Data augmentation helped in reducing overfitting of the system.

**Shortcomings:** Not the fastest algorithms to run, requires a lot of data to train, difficult to do in low-resource environments, takes a while to set up.

**Title: Face Recognition Attendance System for Multiplex-Classrooms.**

**Author:** Y. Cooper, Z. Flores

**Publication Year:** 2020

**Methodologies:** Deep Convolutional Neural Networks combined with transfer learning for detection.

**Observing:** High accuracy demonstrated in large classrooms with differing demographics, making attendance tracking generalizable across many populations of students. Transfer learning was applied to mitigate the requirement for large numbers of training data.

**Shortcomings:** Long training time and resource consumption of the models are resource intensive, and this has a burden on institutions with lower resources and will exclude them from using this technology.

## 4.PROPOSED MODEL, MODULES DESCRIPTION, AND UML DIAGRAMS

### 4. 1. Modules

#### **Image Acquisition:**

**Objectives:** To take quality pictures or videos of the classroom

#### **Method:**

Leverage a camera setup fixed in the classroom to take constant, still images or video frames

Be sure the camera can see the whole classroom so that all of the students and their actions are being monitored.

Capabilities: Very Powerful Image Capture with OpenCV for rapid camera interfacing and image capture.

#### **Preprocessing:**

**Objectives:** Post-process the captured images to better their fit in the following processing steps.

#### **Method:**

**Resizing:** This contrasts the measurements of the captured images so that the consistency is received by your group at all the levels of working.

**Normalization:** Converts the pixel values into a scale which enhances the results and accuracy of the image analysis techniques.

**Noise Reduction:** By applying filters, unnecessary noise can be removed from pictures so that they become clearer.

**Tools/Libraries:** You can work with images using OpenCV.

## **Feature Extraction:**

**Objective:** To find unique facial features in the pictures which have been pre-processed.

## **Method:**

**Convolutional Neural Networks (CNNs):** CNN actually is capable of finding and extracting from photos never before seen parts.

**Detection Layers:** These layers on the CNN can be seen as a means of picking out edges, textures and patterns that uniquely identify any one face; hence they give us an extensive set for recognizing face features.

**Tools/Libraries:** TensorFlow or PyTorch-default building and training flow frameworks widely respected in academia which also allow you to use these models for extracting features from images.

## **Detecting and Recognising Faces:**

**Objective:** Identify students in pictures-by recognizing their faces carrying out real-time check within images by detecting.

## **Methods:**

**Face Detection:** For efficiently detecting faces in captured image frames, OpenCV's Haar cascades are employed.

**Face Recognition:** Models based on CNNs are used to compare the faces the system detects with a pre-existing database of known faces, thus recognizing individual students.

**Tools/Libraries:** OpenCV for face detection and TensorFlow or PyTorch for advanced face-recognition:

**Classification:**

**Objective:** Assign the recognized faces to present and different category students.

**Method:**

Develop and add a classification algorithm that compares the recognized faces with a pre-registered database of students that attend the current class.

This allows to use those comparisons to effectively discriminate between students who belongs to the current class and the one which does not.

**Tools/Libraries:** Scikit-learn (most common used library noted for its wide varieties of efficient classification algorithms)

**Web Integration:**

**Objective:** To design Interface for Interaction and Result Display.

**Method:**

**Flask Framework:** To build a Application using Flask to for easy user interaction, deployment and some sneaky visualization of results.

**GUI:** Design responsive GUI as per the requirement to show the attendances, live data and graph of strength against time.

**Tools/Libraries Used:** Flask for the backend development, HTML/CSS for the front-end design.

### **Real-Time Processing:**

**Objectives:** Make sure the system can process information in real-time to give immediate feedback.

#### **Method:**

Use async processing techniques to read the continuous flow of data coming from the camera without delay.

Also refresh the web interface dynamically with the latest attendance records and classification results generated as per the processing phases.

**Tools/Libraries:** Flask, Socket Real-time data updates Real-time data communication using IO.

### **Graphical Analysis:**

**Objectives:** Show how the number of students in each class evolves over time.

#### **Method:**

Gather the attendance data again and again at an interval in the assigned time periods.

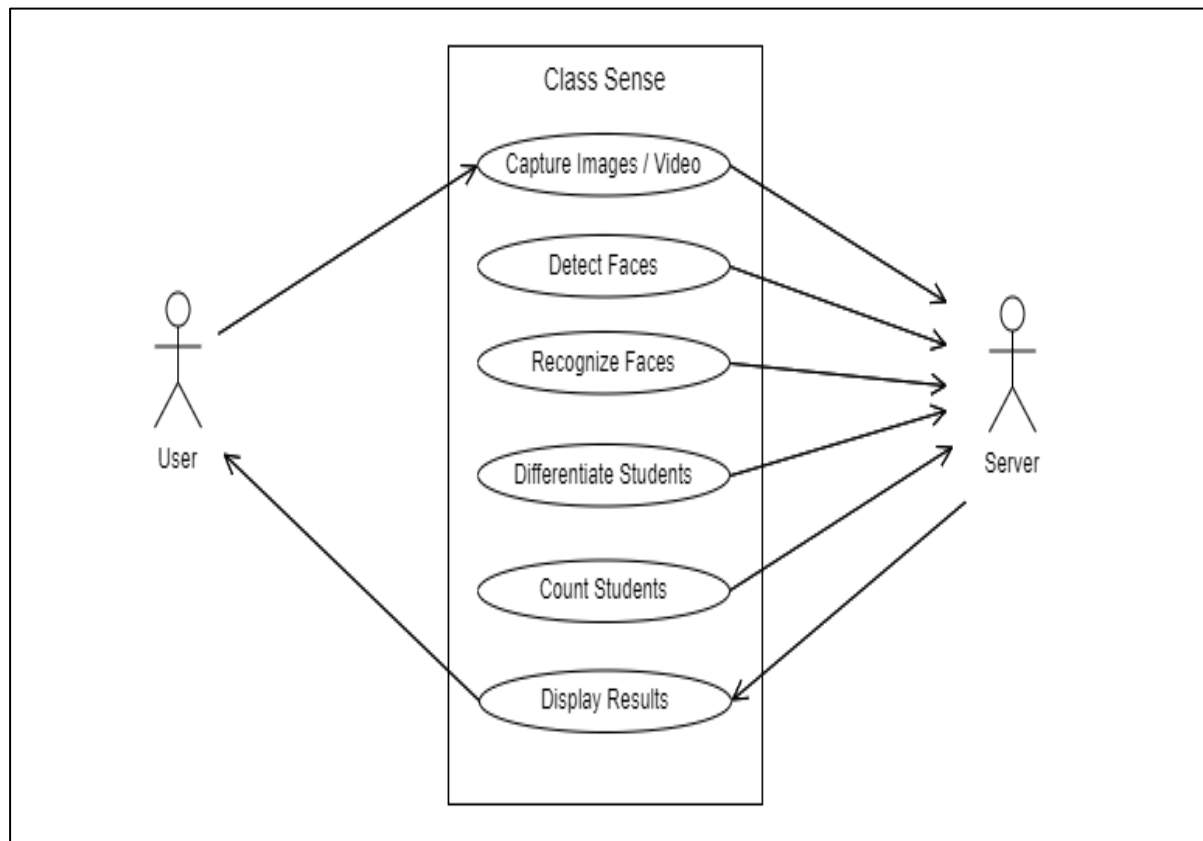
Sophisticated chart plotting libraries that can help create visualization representation of the data will help show the patterns and trends of student attendance

**Tools/Libraries:** Matplotlib, Plots are good for basic and advanced graphical representations.



## 4.2.UML Diagrams

- Use Case Diagram



*Figure 4.1 Use Case Diagram*

**Targeted User Interaction:** Teachers/Administrators use our easy-to-use web interface to perform the desired actions using the web interface to streamline the classroom attendance workflow.

**Teacher Camera:** An in-class camera taking images & video to be used as the primary input for the system to tell if students are present

**Face Detection and Recognition:** It identifies faces in the classroom footage using advanced algorithms and provide accurate detection results in different conditions this helps to classify data.

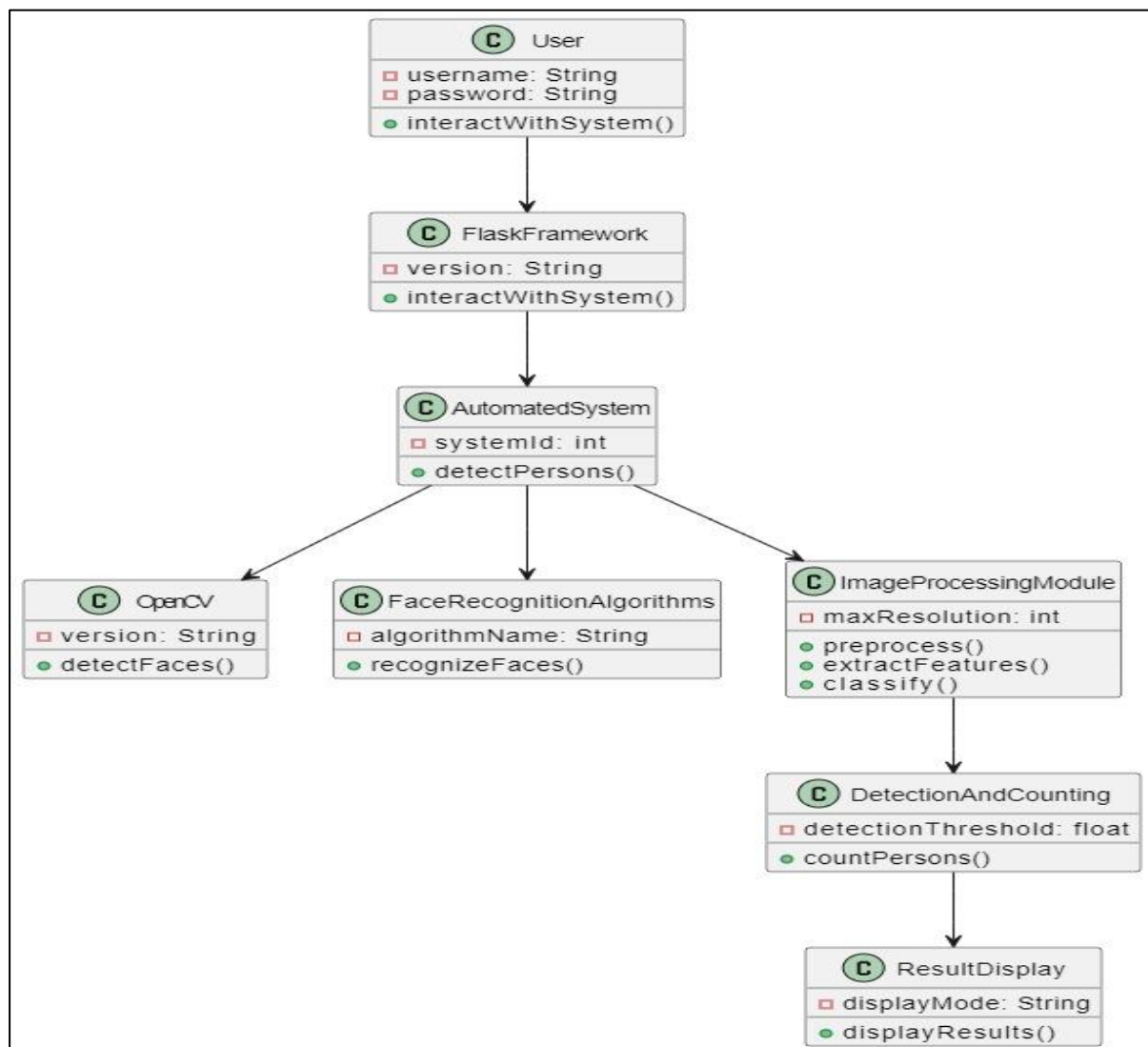
**Automated Attendance Taking:** Securely counts the students that attend school with it, eliminating any manual attendance taking of your teachers and save multiple times.

**Round:** It distinguishes between current class and any other class students in order to create proper attendance.

**Dashboard Results:** The web interface provides a total student count and class-specific count to actionable attendance at a glance.

**System Integration:** Integrated with technologies like OpenCV for image processing and Flask for web deployment, ensuring a seamless user experience and efficient classroom management.

- **Class Diagram**



*Figure 4.2 Class Diagram*

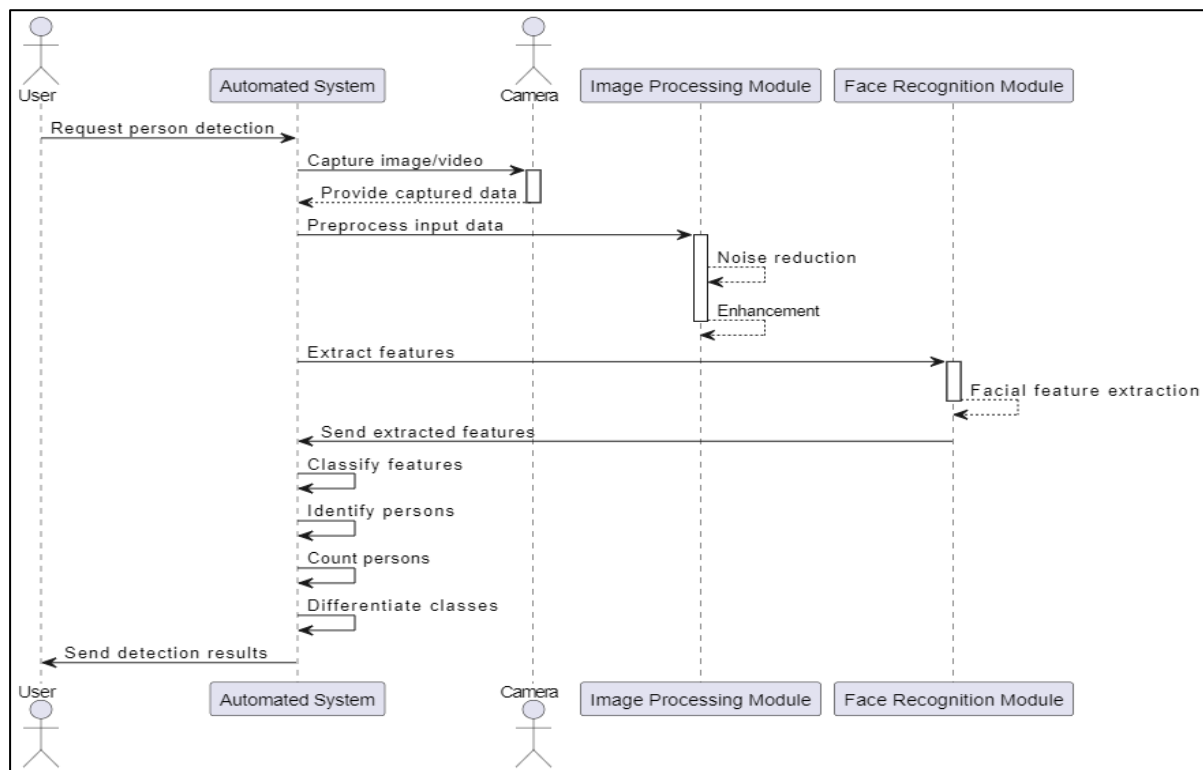
This diagram is class diagram illustrating main components and their interaction in Automated Person Detection System, which will be used to serve as video surveillance system or part of Crowd monitoring system. Powered at the core by the Automated System, the system automates the process of recognizing and counting the people in an image/video. Since the Application component interacts with several modules and libraries, it is quite crucial for processing.

Finally, OpenCV comes with powerful features for detecting faces in images and video frames. The empirical basis for the system on which it can detect where people are located in visual data. It is possible to complement this functionality with its Face Recognition Algorithms which allow to the system, if trained properly, to accurately recognize specific people by his facial features.

**Image Processing Module:** This is responsible for the image processing pre-tasks of input images or video frames before we go any deeper on analysis. This involves noise removal, image improvement and preprocessing for feature extraction. Processed data next feeds into Detection And Counting module that employs feature extraction and classification methods for the identification of single person in difficult background and accurate counting of the humans.

**Flask Framework:** Flask Framework is a communication gateway, between the users and the system, making it possible to deploy the system through web interfaces. Users send requests for person detection and gets the results back while communicating with the system. The Result Display component shows the detection results to the user, giving feedback on how many people were found and identified.

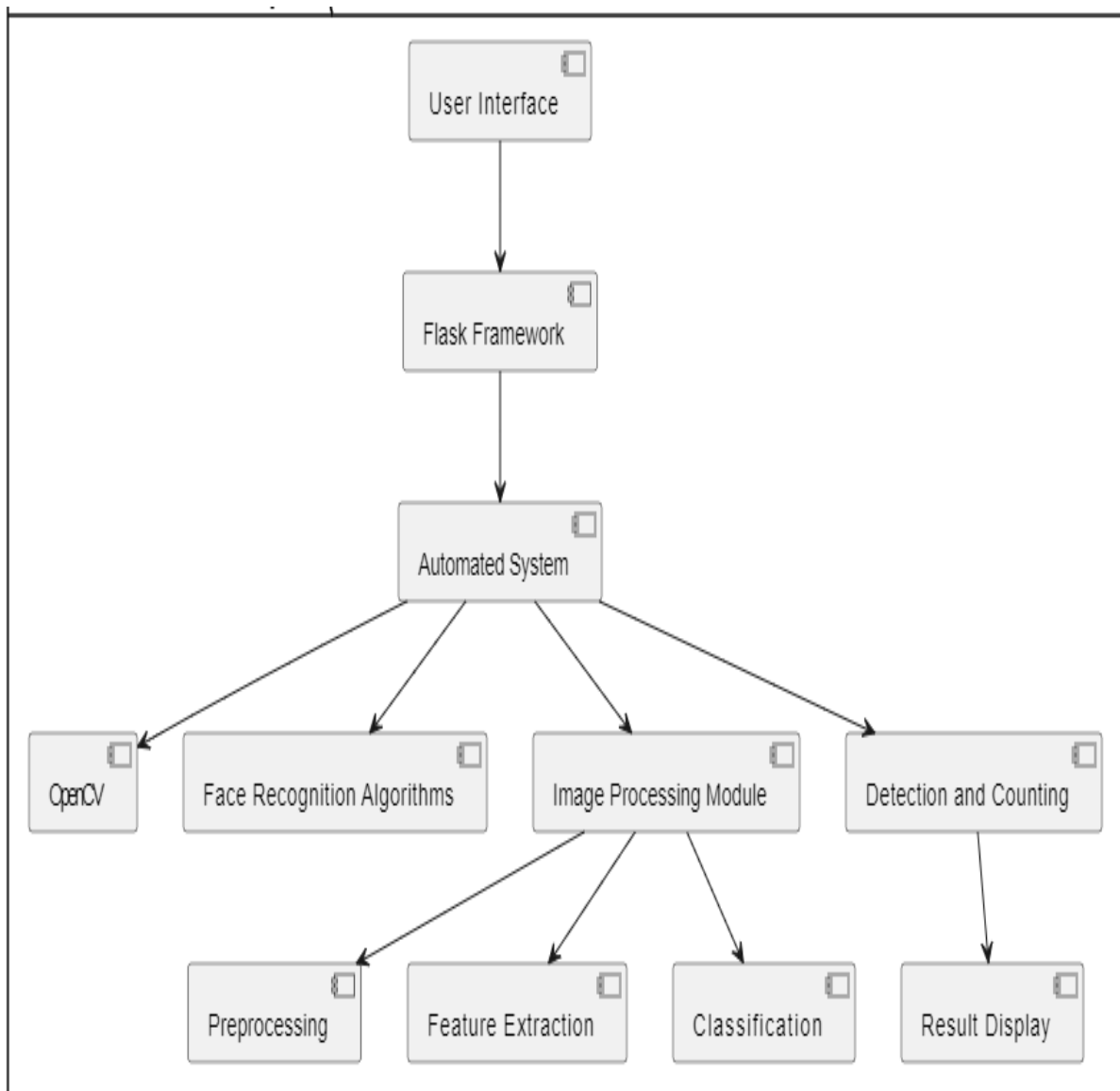
- **Sequence Diagram**



**Figure 4.3 Sequence Diagram**

The sequence diagram in Figure 1 illustrates the functioning of the automated system for person detection, once the user requests it to do so. Namely, the user makes a request for person detection. The system sends a request to a camera which captures an image or a video. The camera obtains the data in question and sends it back to the system. In the meantime, the system: preprocesses frames in order to sharpen the information. Then, it extracts relevant features, such as faces of people present in the image or the video. Finally, it classifies the extracted features, assigning them with labels of individual persons. Additionally, more advanced algorithms and procedures allow the system to count the number of persons and to determine different classes. For instance, it might differentiate between different classrooms and the students within them. The detection results of the system eventually reach the user again with an automated and robust solution for surveillance/crowd monitoring applications. This correspond is exactly which enshrouds de automaton handle too the complete sequels from Daten capture to result delivery correctly.

- **Component Diagram**



*Figure 4.4 Component Diagram*

This component diagram captures the architecture of Automated Person Detection System explained in project abstract. The system's heart is its 'Automated System' unit which combines a few of the most important features. The Flask system is used to create a user-interface that provides the user with a means of contacting you through web interfaces. The Automated uses OpenCV for processing image or video frame inputs like image preprocessing, noise reduction and so on. Even these advanced face recognition algorithms have the ability to recognize people in different visual environments. Equipped with this foundation, these algorithms use a

combination of feature extraction and classification to robustly detect individuals amidst complex background.

This structure of the system architecture is an essential ingredient, where it contains functionalities that include, Pre-processing, Feature Extraction, and Classification. Together they process visual information to accurately detect and count people. The last part of the pie, aka "Detection and Counting", reads the cleanup data and uses them to count the number of individuals and type them into various categories, i.e., to differentiate which students are from what classes within a classroom setting. These analyses subsequently showcased via Scope web component retrieve actionable information for surveillance and crowd monitoring applications, and such results are visually displayed by the "Result Display" component. These modules will ensure the system to be efficient, scalable and robust in automating complicated visual analysis tasks on different applications in the real world.

### Implementation:

25

```

</head>
<body>
  <div class="navbar">
    <div class="navbar-tag">Class Sense </div>
    <div class="links">
      <a href="/home" target="_blank">Home</a>
      <a href="/about" target="_blank">About</a>
      <a href="/services" target="_blank">Services</a>
      <a href="/contact" target="_blank">Contact</a>
    </div>
  </div>
  <div class="images"><h2>Welcome to Class Sense</h2></div>
  <div class="button-container"><a href="/application" class="button">Get Started</a></div>
  <div class="container">
    <h2>Welcome to Class Sense</h2>
    <p>Our project provides state-of-the-art face detection and recognition solutions for various applications. Whether you're looking to enhance security, streamline authentication processes, or personalize user experiences, our platform has you covered.</p>
    <p>With our advanced algorithms and intuitive user interface, you can easily detect and match faces in images and video streams. Our real-time processing capabilities make it ideal for applications such as surveillance, access control, and identity verification.</p>
    <p>Explore our website to learn more about our services, team, and how we can help you achieve your goals with face matching technology.</p>
  </div>
  <footer>
    <p>&copy; 2023 Class Sense. All rights reserved.</p>
    <div class="contact-info">
      <p><a href="/contact">Contact Us:</a></p>
      <p>Email: ankithkumar6281@gmail.com</p>
      <p>Phone: +91-6281558871</p>
    </div>
  </footer>
</body>
</html>

```

```

.images {
  background-image: url("../static/welcome_image.png");
  height: 90vh;
  width: 90vw;
  background-repeat: no-repeat;
  background-position: center;
  background-size: cover;
  border-radius: 10px;
  margin: 10px;
  position: relative;
  left: 4%;
  text-align: center;
  >h2 {
    color: #1f77b4;
    font-family: Georgia, 'Times New Roman', Times, serif;
    font-size: xx-large;
  }
}
footer {
  background-color: #f0f0f0;
  padding: 20px;
  text-align: center;
}
.container {
  border: 2px solid #e67e22;
  border-radius: 7px;
  padding: 10px;
  text-align: center;
  margin: -20px 0px 10px 0px;
  background-color: #f9f9f9;
  >h2 {color: #8e44ad;}
}
</style>

```



```

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Our Services</title>
</head>

<body>
  <div class="container">
    <h1>Our Services</h1>
    <h2>Face Detection</h2>
    <p>We provide advanced face detection services using state-of-the-art computer vision algorithms. Our system can accurately locate faces in images and video streams, making it ideal for various applications such as surveillance, access control, and more.</p>
    <h2>Face Recognition</h2>
    <p>With our face recognition services, you can identify individuals based on their facial features. Whether you need to verify a person's identity or match faces against a database, our robust algorithms deliver accurate results with high precision.</p>
    <h2>Real-time Processing</h2>
    <p>Our platform offers real-time processing capabilities, allowing you to analyze live video streams for face detection and recognition. This feature is invaluable for applications that require instant responses and continuous monitoring.</p>
    <h2>Custom Solutions</h2>
    <p>We understand that every project has unique requirements. That's why we offer custom solutions tailored to your specific needs. Whether you're developing a new application or integrating face recognition into an existing system, our team will work closely with you to deliver the perfect solution.</p>
    <button><a href="/home">Get Started</a></button>
  </div>
</body>
</html>

```

```

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>About Class Sense</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      margin: 0;
      padding: 0;
      background-color: #f7f7f7;
      color: #333;
    }

    .container {
      max-width: 80vw;
      margin: auto;
      padding: 20px;
      background-color: #fff;
      border-radius: 8px;
      box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
    }

    h1, h2, h3 {
      text-align: center;
      color: #333;
    }

    p {
      margin-bottom: 20px;
      line-height: 1.6;
    }
  </style>

```

```

<script>
  const videoElement = document.getElementById('videoElement');
  const startWebcamBtn = document.getElementById('startWebcam');
  const endWebcamBtn = document.getElementById('endWebcam');
  const startMatchingBtn = document.getElementById('startMatching');
  const matchedCountSpan = document.getElementById('count');
  const matchedFacesDiv = document.getElementById('matchedFaces');

  let mediaRecorder;
  let chunks = [];
  let sendVideoInterval;
  let stream;
  let input_interval = 5000;
  // Initial data points
  let data = [];

  // Set up the SVG canvas dimensions
  const width = 600;
  const height = 350;
  const margin = { top: 20, right: 20, bottom: 50, left: 50 };

  // Create scales for x and y
  const xScale = d3.scaleTime()
    .domain([d3.min(data, d => d.x), new Date()])
    .range([margin.left, width - margin.right]);

  const yScale = d3.scaleLinear()
    .domain([d3.min(data, d => d.y), d3.max(data, d => d.y)])
    .range([height - margin.bottom, margin.top]);

  // Create the line generator with spline interpolation
  const line = d3.line()
    .x(d => xScale(d.x))
    .y(d => yScale(d.y))
    .curve(d3.curveMonotoneX); // Use monotone curve for smooth interpolation

```

```

function startWebcam() {
  navigator.mediaDevices.getUserMedia({ video: true })
    .then(mediaStream => {
      videoElement.srcObject = mediaStream;
      stream = mediaStream;
    })
    .catch(error => {
      console.error('Error accessing webcam:', error);
      alert('Error accessing webcam: ' + error.message);
    });
}

function startRecording() {
  mediaRecorder = new MediaRecorder(stream);
  mediaRecorder.ondataavailable = (event) => {
    if (event.data.size > 0) {
      chunks.push(event.data);
    }
  };
  mediaRecorder.start(); // Start recording
}

function stopRecording() {
  if (mediaRecorder && mediaRecorder.state !== 'inactive') {
    mediaRecorder.stop();
  }
}

```

```

// Add x-axis label
svg.append("text")
  .attr("text-anchor", "end")
  .attr("x", width / 2 + margin.left)
  .attr("y", height - 10)
  .text("Time");

// Add y-axis label
svg.append("text")
  .attr("text-anchor", "end")
  .attr("transform", "rotate(-90)")
  .attr("y", margin.left - 35)
  .attr("x",
    -height / 2 + margin.top)
  .text("Face Count");

// Function to update the graph
function updateGraph() {
  // Update scales
  xScale.domain([d3.min(data, d => d.x), new Date()]);
  yScale.domain([d3.min(data, d => d.y), d3.max(data, d => d.y)]);

  // Update path
  path.datum(data)
    .attr("d", line);

  // Update x-axis and y-axis
  svg.select(".x-axis").call(xAxis);
  svg.select(".y-axis").call(yAxis);
}

```

```

const path = svg.append("path")
  .datum(data)
  .attr("fill", "none")
  .attr("stroke", "steelblue")
  .attr("stroke-width", 2)
  .attr("d", line);

// Tooltip div selection
const tooltip = d3.select("#tooltip");
// Mouse move event listener
svg.on("mousemove", function (event) {
  // Get mouse coordinates
  const [mouseX, mouseY] = d3.pointer(event);

  // Find the nearest data point
  const bisect = d3.bisector(d => d.x).left;
  const index = bisect(data, xScale.invert(mouseX));
  const closestPoint = data[index];

  // Calculate distance to the nearest data point
  const distance = Math.abs(xScale(closestPoint.x) - mouseX);

  // Check if the distance is within the range
  if (distance < 10) { // Adjust the range as needed
    // Show the tooltip
    tooltip.style("opacity", 1)
      .html(`Time: ${new Date(closestPoint.x).toLocaleString()}<br>Face Count: ${closestPoint.y}`)
      .style("left", (event.pageX + 5 - 490) + "px")
      .style("top", (event.pageY - 28 - 670) + "px");
  } else {
    // Hide the tooltip if the cursor is far from any point
    tooltip.style("opacity", 0);
  }
});

```

```

def process_video(video_path):
    """Process the video to detect and match faces."""
    matched_faces = []
    video_capture = cv2.VideoCapture(video_path)
    frame_skip = 10 # Process every 10th frame
    frame_count = 0
    global total_faces
    total_faces = 0
    while True:
        ret, frame = video_capture.read()
        if not ret:
            break

        if frame_count % frame_skip == 0:
            small_frame = resize_frame(frame)
            face_encodings = get_face_encodings(small_frame)
            total_faces = max(total_faces, len(face_encodings))
            matches = compare_faces(face_encodings)
            matched_faces.extend(matches)

        frame_count += 1

    video_capture.release()
    return list(set(matched_faces)) # Ensure unique paths

def resize_frame(frame, scale=0.25):
    """Resize the given frame to reduce processing time."""
    return cv2.resize(frame, (0, 0), fx=scale, fy=scale)

def get_face_encodings(frame):
    """Find face locations and encodings in the given frame."""
    face_locations = face_recognition.face_locations(frame)
    face_encodings = face_recognition.face_encodings(frame, face_locations)
    return face_encodings

def compare_faces(face_encodings):
    """Compare detected face encodings with known face encodings."""
    matched_faces = []
    for face_encoding in face_encodings:
        matches = face_recognition.compare_faces(known_face_encodings, face_encoding, tolerance=0.4)
        for i, match in enumerate(matches):
            if match:
                matched_faces.append(known_face_paths[i])
                # break
    return matched_faces

```

```

function stopWebcam() {
  stopRecording();
  if (stream) {
    stream.getTracks().forEach(track => track.stop());
    videoElement.srcObject = null;
    stream = null;
  }
  clearInterval(sendVideoInterval);
  console.log('Webcam stopped.');
```

```

}

function displayFaces(faces) {
  document.querySelector("#matchedFaces").innerHTML = "";
  for (let i = 0; i < faces.length; i++) {
    const personName = faces[i].substring(faces[i].lastIndexOf("\\") + 1, faces[i].length - 4);
    let a = "anjhjd";
    const matchedFaces = document.querySelector("#matchedFaces");
    const personCard = document.createElement("div");
    personCard.setAttribute("class", "personcard");
    personCard.innerHTML = `<img src=../static/verified_img.svg id="verified">
    <img src=../static/faces/${personName}.jpg" id = "person_img">
    <div id = "personName">${personName}</div>`;
    console.log(personCard);
    matchedFaces.append(personCard);
  }
}

function displayCountFaces(count_faces, total_faces) {
  const matchedCount = document.querySelector("#count");
  matchedCount.innerHTML = `${count_faces}`;

  const totalFaces = document.querySelector("#total-count");
  totalFaces.innerHTML = `${total_faces}`;

  // Generate new data point
  const newX = new Date().getTime();
  const newY = count_faces; // Random value between 0 and 100

  // Add new data point to array
  data.push({ x: newX, y: newY });

  // Update graph
  updateGraph();
}

```

```

from flask import Flask, request, jsonify, render_template, redirect, url_for
import os
import cv2
import face_recognition

app = Flask(__name__)
VIDEO_FOLDER = 'video'
FACES_FOLDER = 'static\\faces'
app.config['VIDEO_FOLDER'] = VIDEO_FOLDER
app.config['FACES_FOLDER'] = FACES_FOLDER

# Ensure the folders exist
os.makedirs(VIDEO_FOLDER, exist_ok=True)
os.makedirs(FACES_FOLDER, exist_ok=True)

video_counter = 0
total_faces=0

@app.route('/')
def index():
    return render_template('index.html')

@app.route('/home')
def home():
    return render_template('index.html')

@app.route('/services')
def services():
    return render_template('services.html')

@app.route('/about')
def about():
    return render_template('about.html')

@app.route('/contact')
def contact():
    return render_template('contact.html')

@app.route('/application')
def frontend():
    return render_template('frontend.html')

```

```

@app.route('/video', methods=['POST'])
def save_video():
    global video_counter

    if 'video' not in request.files:
        return jsonify({"error": "No video part in the request"}), 400

    video = request.files['video']
    if video.filename == '':
        return jsonify({"error": "No selected video file"}), 400

    video_counter += 1
    video_filename = f"{video_counter}.mp4"
    video_path = os.path.join(app.config['VIDEO_FOLDER'], video_filename)
    video.save(video_path)

    matched_faces = process_video(video_path)

    # Print count of persons found and their paths
    print(f"Total Faces Found : {total_faces}")
    print(f"Total Matches: {len(matched_faces)}")
    print("Matched faces:", matched_faces)

    # Remove the video file after processing
    os.remove(video_path)

    return jsonify({"message": "Video processed successfully", "matched_faces": matched_faces, "total_faces": total_faces}), 200

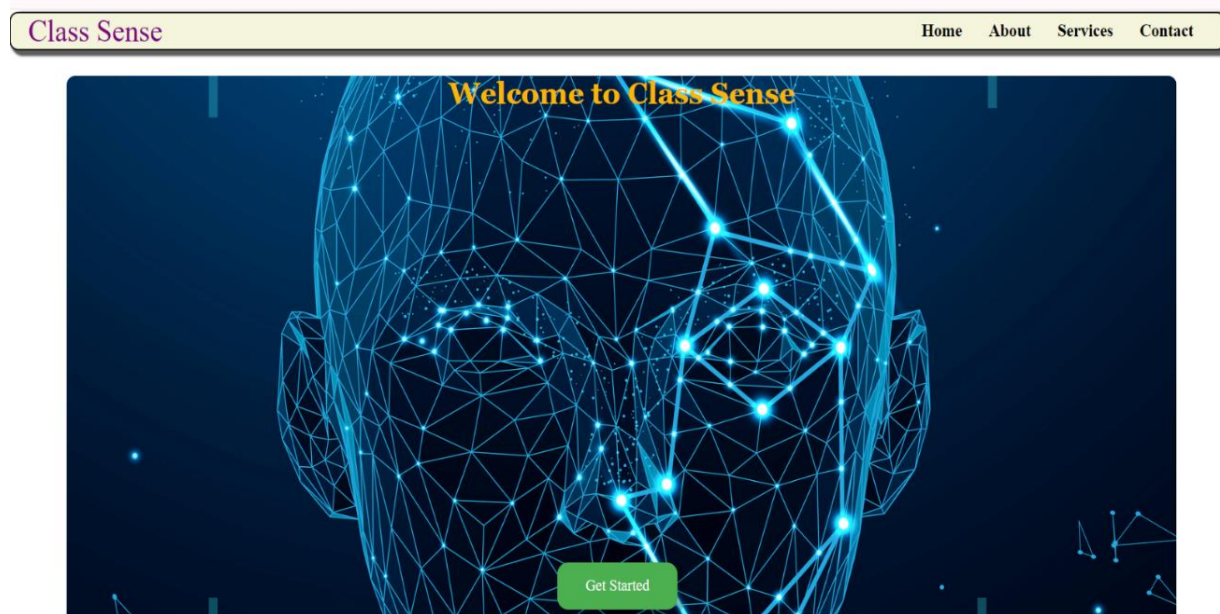
def load_known_faces(folder_path):
    """Load and encode faces from the given folder."""
    person_images = os.listdir(folder_path)
    face_encodings = []
    face_paths = []

    for image_path in person_images:
        image_full_path = os.path.join(folder_path, image_path)
        image = face_recognition.load_image_file(image_full_path)
        encodings = face_recognition.face_encodings(image)
        if encodings:
            face_encodings.append(encodings[0])
            face_paths.append(image_full_path)

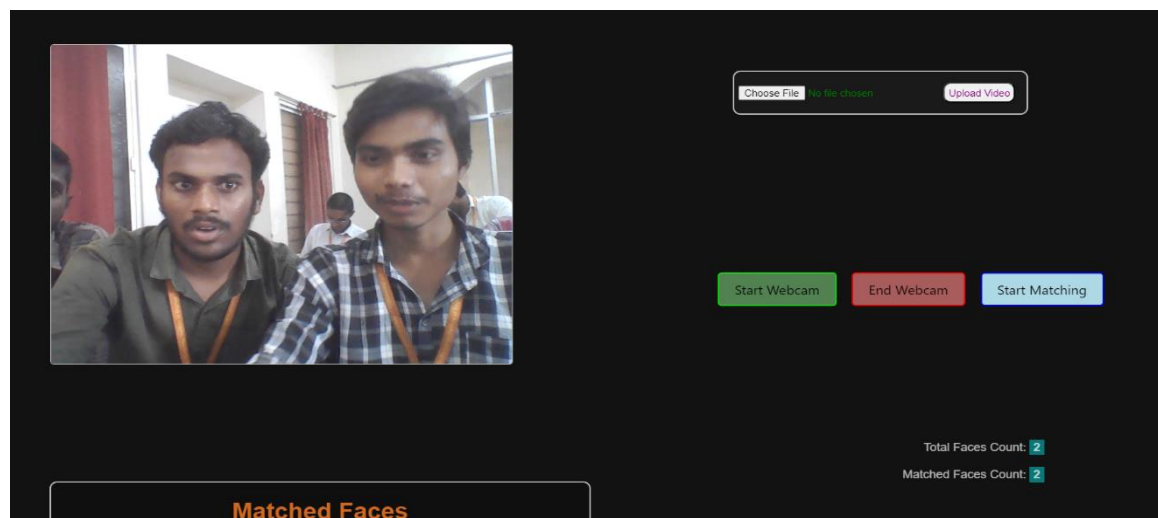
    return face_encodings, face_paths

```

## Output



*Figure 5.1 Home Page*



*Figure 5.2 Input Page*

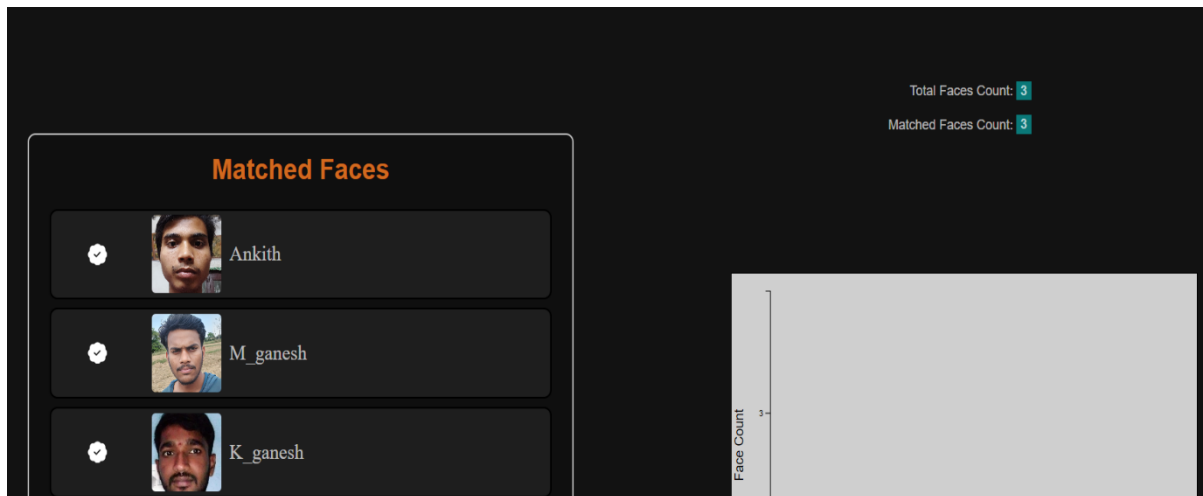
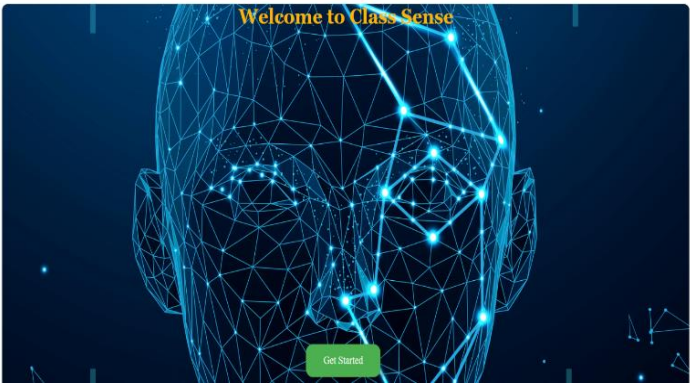
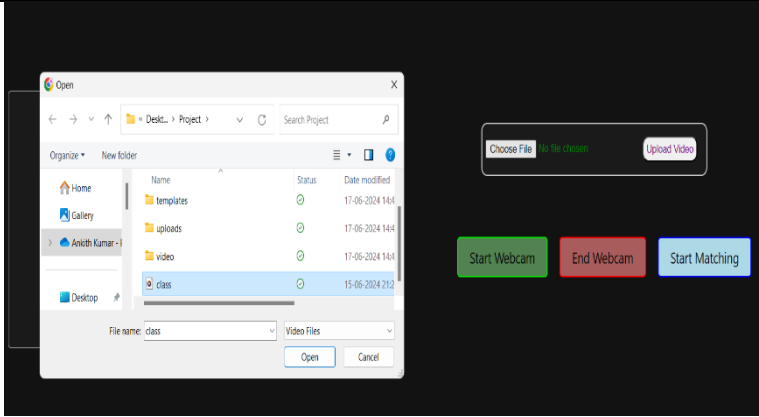
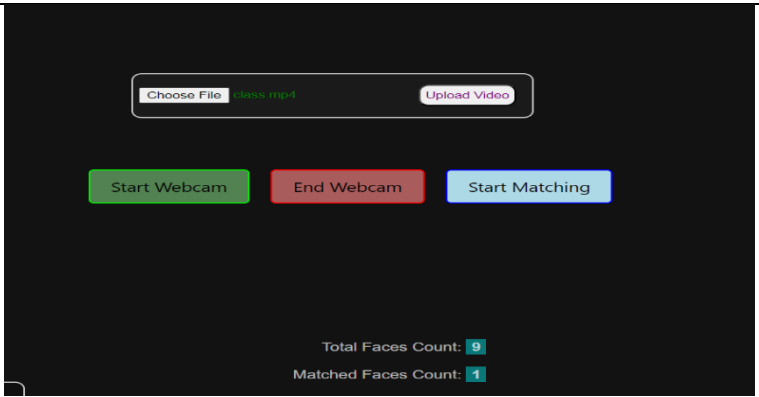
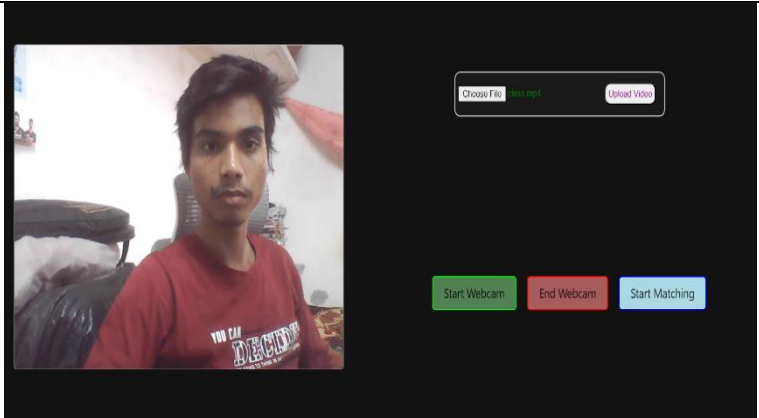
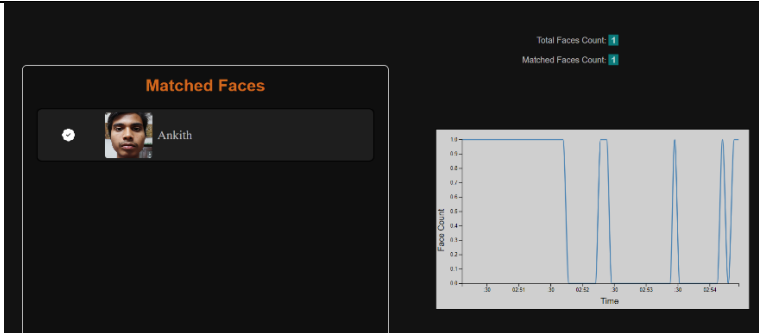


Figure 5.3 Output Page

## Experimental Results & Test Cases

Id	Description	Test Case	Status
1	Running the python backend code in terminal	<pre> PS C:\Users\ankit\OneDrive\Desktop\Project&gt; python -u "c:\Users\ankit\OneDrive\Desktop\Project\application.py" * Serving Flask app 'application' * Debug mode: on WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead. * Running on http://127.0.0.1:5000 Press CTRL+C to quit * Restarting with stat * Debugger is active! * Debugger PIN: 344-576-840 127.0.0.1 - - [17/Jun/2024 17:35:29] "GET / HTTP/1.1" 200 - 127.0.0.1 - - [17/Jun/2024 17:35:29] "GET /static/welcome_image.png HTTP/1.1" 304 - 127.0.0.1 - - [17/Jun/2024 17:35:29] "GET /favicon.ico HTTP/1.1" 404 - </pre>	Running successfully
2	Running the frontend in browser		Running application successfully



3	Selecting video file from directory		File uploaded successfully
4	Detecting faces from video file.		Faces detected successfully
5	Capturing video from application using webcam, when start webcam button		Captured video successfully
6	Matching single captured person from database.		Person matched successfully

*Table 5.1 Test Cases*

## **6. Conclusion and Future Scope**

### **Conclusion**

Using Advanced image processing and face recognition techniques, the automated classroom attendance system implemented in this project is able to overcome issues arising from the manual attendance record. This allows for real-time student detection using the combination of OpenCV, Convolutional Neural Networks (CNNs) and running it on Flask. It can differentiate between class wise current class students and students from older classes and uses a user-friendly web interface for interactive querying and visualization of results. The performance of our system has also been verified by real-world classroom experiments, which are crucial steps in the process to prove capable of automating administrative work, improving classroom administration in general.

### **Future Scope**

The future additions of this classroom attendance automation project can change the course of attendance regulation in educational institutions and improve upon it further with the range of functionalities it has to offer. Here is a set of steps that demonstrate how the project can be put to use and what it may change in the future:

- **Smart Classrooms Integration**

With schools and universities moving towards smart classroom settings, this system can be connected with other IoT devices and smart technologies. For example, integrated with an attendance system, the lighting and climate control system can be automated to use the minimum amount of power based on the room occupancy.

- **Enhanced Learning Analytics:**

Using attendance statistics with learning management systems (LMS) enables schools to measure student engagement, which correlates with significantly improved performance. Attendance patterns can be distilled even further, leading faculty, staff, and administration to pinpoint students who may benefit from extra support or intervention.

- **Remote and Hybrid Learning:**

This system can be adapted to verify student attendance in online classes especially due to the increasing remote and hybrid learning environments. This technology can be used to make sure that only the students who should be attending classes live, and thus uphold accountability in remote education.

- **Security and Access Control:**

The system could even be adapted to work as a security measure, restricting access to school and classroom spaces. It will ensure the security of the campus, and allow only approved students or staff to come in.

- **Parental Involvement:**

Parents can be alerted by the system in a real time basis when the version updated in the future. And though this, parents are kept up-to-date and up-to-speed with how their children are doing in school - with everybody lending their talents to the raising of kids with good grades.

- **Administrative Efficiency:**

This system can help remove the administrative burden from teachers and school staff as well. Taking the attendance manually is a thing of the past and educators can concentrate more on teaching than the paper work. This system also makes the reporting process easier, providing accurate attendance reports to school administrators promptly.

- **Cross-Platform Compatibility:**

This will enable the system to be used more widely as everyone has a mobile app and it will work smoothly on all devices. The mobile or desktop base platform allows teachers, students and parents interact with the system any place any time, hence bringing in the convenience and usability factor.

## 7. References

- **Paul Viola, Michael Jones** “Rapid Object Detection using a Boosted Cascade of Simple Features”

Link: <https://www.microsoft.com/en-us/research/publication/rapid-object-detection-using-a-boosted-cascade-of-simple-features/>

- **Zhengyou Zhang** Title: “Iterative Point Matching for Registration of Free-Form Curves and Surfaces”

Link: <https://link.springer.com/article/10.1007/BF01427149>

- **Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun** Title: Deep Residual Learning for Image Recognition

Link: <https://arxiv.org/abs/1512.03385>

- **Ian Goodfellow, Yoshua Bengio, Aaron Courville** Title: Deep Learning

Link: <https://www.deeplearningbook.org/>

- **Florian Schroff, Dmitry Kalenichenko, James Philbin** Title: Face-Net: A Unified Embedding for Face Recognition and Clustering

Link: <https://arxiv.org/abs/1503.03832>

- **Jupyter Notebook Documentation** Title: Project Jupyter

Link: <https://jupyter.org/>

- **Plotly Documentation** Title: The Front End for ML and Data Science Models

Link: <https://plotly.com/>

- **Andrew Y. Ng** Title: Sparse Autoencoder

Link: <http://ufldl.stanford.edu/tutorial/unsupervised/Autoencoders/>

- **Yann LeCun, Yoshua Bengio, Geoffrey Hinton** Title: Deep Learning

Link: <https://www.nature.com/articles/nature14539>

- **Stefanos Zafeiriou, Cha Zhang, Zhengyou Zhang** Title: A Survey on Face Detection in the Wild: Past, Present and Future

Link: <https://www.sciencedirect.com/science/article/pii/S1077314214001283>

- **OpenCV Documentation** Title: Open Source Computer Vision Library

Link: <https://opencv.org/>

- **Davis E. King** Title: Dlib-ml: A Machine Learning Toolkit

Link: <http://dlib.net/>

---

## APPENDIX

- Snapshot of the Result

