A picture containing text, sign

Description automatically generated

FINAL PROJECT REPORT

Project Topic: Face recognition attendance system

**Module : 7COM1039-0509-2022 Advanced Computer Science Masters Project**

Icon

Description automatically generated with medium confidence

**Carried Out by: PARTH BABUBHAI DAVRA**

**University Of Hertfordshire**

**Student ID: 20063130**

**MSc Computer Science Modular - Artificial Intelligence and Robotics**

**September 2021-2022**

**Declaration:**

Parth Babubhai Davra, a student at the University of Hertfordshire pursuing a MSc in Computer Science (Artificial Intelligence and Robotics), undertakes this project as coursework for the 7COM1039-0509-2022 Advanced Computer Science Masters Project module. I am fully aware of all the requirements and guidelines needed to fulfill my contributions for this project. I understand the academic integrity policies regarding ethics approval, originality, and citations for this module. I acknowledge that the work done is my own and proper referencing has been provided where applicable.

I would like to express my gratitude to ***Dr. Rawad Hammad***for his exceptional guidance and feedback concerning the project. I also thank the ***UH Ethics Approval Team*** for their direction on the ethics approval process required for the survey conducted to support my research question.

Finally, I sincerely appreciate module leader ***Dr. Myasar Tabany and the module team*** for being outstanding instructors throughout the module and making it an enriching learning experience that yielded strong project outcomes.

This report is submitted in partial fulfilment of the requirement for the degree of Master of Science in 7COM1039-0509-2022 Advance Computer Science Master Project at the University of Hertfordshire (UH).

It is my own work except where indicated in the report.

*I did not use human participants in my MSc Project.*

I hereby *give / withhold* permission for the report to be made available on the university website provided the source is acknowledged (delete as necessary).

Table of Contents

[Abstract: 6](#_Toc145959980)

[Introduction: 7](#_Toc145959981)

[Background: 8](#_Toc145959982)

[**Face Recognition Concepts:** 9](#_Toc145959983)

[**Review of Related Work:** 10](#_Toc145959984)

[Literature Review: 11](#_Toc145959985)

[RESEARCH QUESTIONS: 17](#_Toc145959986)

[Hypothesis: 17](#_Toc145959987)

[Problem statement: 17](#_Toc145959988)

[Aim: 18](#_Toc145959989)

[Objective: 19](#_Toc145959990)

[Risks involved with this research: 20](#_Toc145959991)

[Ethical Risks: 20](#_Toc145959992)

[Legal Risks: 20](#_Toc145959993)

[Professional Risks: 21](#_Toc145959994)

[Societal Risks: 21](#_Toc145959995)

[Existing Scenario Analysis: 22](#_Toc145959996)

[Paper-based Attendance 22](#_Toc145959997)

[Swipe Card Attendance 23](#_Toc145959998)

[Fingerprint-based Attendance 24](#_Toc145959999)

[Methodology: 26](#_Toc145960000)

[Web Application User Interface 29](#_Toc145960001)

[API Server Middleware 29](#_Toc145960002)

[Project Development Approach 29](#_Toc145960003)

[Project Workflow 30](#_Toc145960004)

[**Workflow diagram** 33](#_Toc145960005)

[Problems and Challenges faced while developing my Proposed system: 34](#_Toc145960006)

[Comparative Analysis: 34](#_Toc145960007)

[Conclusion: 37](#_Toc145960008)

[Summary of Findings: 37](#_Toc145960009)

[Analysis of Challenges and Limitations: 38](#_Toc145960010)

[Network Reliability Considerations: 38](#_Toc145960011)

[Future Challenges in Face recognition attendance system: 40](#_Toc145960012)

[Addressing Spoofing and Presentation Attacks 40](#_Toc145960013)

[Managing Diversity and Addressing Bias 40](#_Toc145960014)

[Scaling Capacity and Uptime 40](#_Toc145960015)

[Flow Chart 42](#_Toc145960016)

[Login 42](#_Toc145960017)

[Student/ Staff and visitor Form 43](#_Toc145960018)

[Change Password 44](#_Toc145960019)

[Website Images 45](#_Toc145960020)

[Login 45](#_Toc145960021)

[User List 45](#_Toc145960022)

[Add Student/ visitor details: 46](#_Toc145960023)

[Staff List: 46](#_Toc145960024)

[Add staff Details: 47](#_Toc145960025)

[User Attendance: 47](#_Toc145960026)

[Staff Attendance: 48](#_Toc145960027)

[Classroom List: 48](#_Toc145960028)

[Add classroom number: 49](#_Toc145960029)

[Change Password 49](#_Toc145960030)

[Face Detection: 50](#_Toc145960031)

[User Images: 51](#_Toc145960032)

[Unknown person image sends to the admin 51](#_Toc145960033)

[External webcam: 52](#_Toc145960034)

[Connector for external webcam 52](#_Toc145960035)

[References: 53](#_Toc145960036)

[Appendices: 57](#_Toc145960037)

# **Abstract:**

Tracking attendance is an often-tedious task, in institutions. Traditional methods like paper sign-in sheets are time-consuming, disruptive, and prone to errors. As a result, there has been a growing interest in automating attendance using facial recognition technology. The goal is to utilize computer vision techniques to identify and track students in time from classroom camera feeds. Despite the accuracy of this approach, there are still challenges to overcome, such as managing large databases and adapting to real-world variations addressing privacy and ethical concerns.

In this paper, I present the design and implementation of an automated attendance system that utilizes recognition. The system captures images within the classroom. Employs convolutional neural network algorithms to detect facial features. These encoded face representations are then compared against a database containing student images collected at the beginning of each term. When a match is found the student's attendance record is automatically updated in the database.

Testing conducted on a dataset of images under controlled conditions demonstrated matching accuracy and effective marking of attendance. However certain limitations were identified regarding scalability, hardware dependencies, and access control measures. Accommodating environments. It is essential to address risks such as spoof attacks (attempting to deceive the system) and biases in recognition accuracy as well, as ensure data security before deploying this system in real-world scenarios.

In the future, it is important to prioritize enhancing the accuracy of face recognition systems, in situations improving the efficiency of indexing and searching large-scale face databases, and implementing privacy and security measures.

Although the prototype system shows promise in automating attendance systems through face recognition there are challenges related to scalability, reliability, and ethical concerns that need to be addressed before these biometric attendance systems can be widely adopted in corporate settings. This project offers insights, into the capabilities of such systems and highlights key areas that require further research and innovation to overcome these challenges.

# **Introduction:**

Face recognition technology has become increasingly popular in years, as a solution for automating attendance monitoring in educational institutions. Traditional manual methods, such as calling out names or collecting signatures are time-consuming, tedious, and prone to errors. As a result, researchers and institutions have been motivated to explore automated attendance systems that utilize technology like face recognition.

The fundamental concept behind attendance systems based on face recognition is to employ cameras and computer vision algorithms to detect and identify student's faces in time as they enter the classroom. This allows their attendance to be instantly marked in a database without any intervention. Such systems offer advantages over paper-based methods of taking attendance;

1. Accuracy: Face recognition systems can accurately identify students eliminating instances of buddy punching or proxy attendance that may occur with methods.

2. Time Savings: The automatic face recognition process takes a few seconds per student saving class time that would otherwise be spent on calling out names or collecting signatures.

3. Real-time Tracking: Face recognition provides up-to-date information on who has arrived and allows for the identification of latecomers or early departures.

4. Record Keeping: Attendance records are automatically stored in databases making it easy to retrieve and analyze them for purposes such, as generating attendance reports and identifying patterns of absenteeism.

5. Contactless Process: Face recognition technology allows for convenient attendance tracking ensuring both hygiene and ease of use. Recent research papers highlight the development and implementation of face-recognition attendance systems, in institutions around the world.

One particular study conducted in 2020 titled "Implementing a Face Recognition-Based Attendance Monitoring System for Educational Institutions " describes a system deployed at a university in Malaysia. The system utilizes a Raspberry Pi equipped with an 8MP Pi camera module to capture classroom images. An algorithm based on OpenCV is employed to detect and recognize faces by comparing them against an existing database. The attendance of recognized students is recorded by updating a MySQL database. A dedicated website enables access to attendance reports filtered by students or subjects demonstrating the accuracy and reliability of the system during testing.

Another research paper from 2020 named "Utilizing Face Recognition and Machine Learning for Attendance Tracking, with Location Identification " proposes an approach using OpenCV, the face recognition Python package, and the Google Sheets API. In this system, real-time images captured by classroom cameras are processed through a face recognition model. Upon identifying known students their attendance is automatically recorded in a Google Sheet using the Sheets API. This solution offers cost data storage on cloud platforms.

In 2019 a research paper titled "A Face Recognition System, for Attendance Tracking in a Nigerian University" introduced a system called DLFacer. This system utilized a network based on the Faster R CNN algorithm to extract facial features from classroom images. It then accurately. Recognized students, recording their attendance in a MongoDB database. The researchers trained the system using a curated dataset of university students and integrated it with an intuitive web interface. In testing DLFacer achieved an accuracy rate of 97 99%.

Another research paper from 2019 titled "Smart Attendance System using Face Recognition " presented a portable attendance device based on the Internet of Things (IoT). This device employed face recognition technology utilizing OpenCV for face detection and recognition algorithms. The experiments conducted with this prototype demonstrated its ability to significantly reduce the time required for attendance marking when compared to other methods.

In 2018 another paper named "Facial Recognition Technology in Schools; Addressing Critical Questions and Concerns" offered a perspective on implementing face recognition-based attendance systems in institutions. It raised concerns about risks such as technology overreach, authoritarian usage, and perpetuating racial biases. The paper also argued that these systems could potentially disrupt the classroom environment in ways and questioned whether their use could ever be ethically justified within schools.

These papers collectively showcase techniques, like OpenCV, Deep Learning, and IoT being employed to develop face recognition-based attendance systems.

The typical process involves identifying faces extracting features matching them with a database of student images and updating attendance records in a database or cloud service.

In some cases, the reported accuracy levels are quite high ranging from 97% to 99%. This indicates that face recognition has progressed to a point where it can reliably automate attendance marking. However, there are still challenges when it comes to accounting for variations, in student’s facial appearances from day to day dealing with lighting conditions, and addressing privacy concerns mentioned in some research papers.

## **Background:**

Face recognition technology has seen rapid advances in the past decade, enabling its increasing application in various domains including security, marketing, and attendance monitoring. In academia, manual student attendance tracking during lectures is a routine yet tedious task. Institutions are now actively exploring automated attendance management systems using face recognition to save time and improve accuracy. This background provides an overview of face recognition concepts and reviews work done on applying this technology for automated attendance systems in educational institutions.

### **Face Recognition Concepts:**

Face recognition refers to an automated process that identifies or verifies a person’s identity using images or videos of their face. The process involves comparing features, from the image with a database of faces to find the closest match.

#### **There are stages in a face recognition system:**

* **Face Detection:** This initial step involves detecting and isolating faces in input images or video frames. Techniques such as Haar cascades, HOG, and convolutional neural networks are commonly used for this purpose.
* **Feature Extraction:** Once the face is detected distinguishing features are extracted to create a representation of the face. These features typically include measurements related to the positions and sizes of eyes, nose, mouth, and jaw edges.
* **Face Enrolment:** In identification mode the extracted facial representations (faceprints) are stored in a database along with corresponding person identities. This enrolled face database is used for matching purposes.
* **Face Matching:** The test faceprint is compared against the enrolled faceprints using models like distance to find the closest match.
* **Decision:** If there is a match, within a predefined confidence threshold the test face is identified as belonging to an individual; otherwise, it is rejected as unknown.

Various popular algorithms utilized in face recognition include Principal Component Analysis (PCA) Linear Discriminant Analysis (LDA) Independent Component Analysis (ICA) Neural Networks and Deep Learning models.

Attendance tracking, in institutions, has traditionally relied on paper registers, signatures or teachers manually calling out names. However, these methods are time-consuming, tedious, and susceptible to practices like buddy punching. To address these challenges automated solutions such as RFID cards fingerprint/iris biometrics and facial recognition are being explored as alternatives.

**Among these options face recognition offers advantages that make it particularly suitable for automating attendance management:**

* **Contactless Identification:** Students can be identified simply by using a camera without the need for any contact.
* **Non-Invasive:** Capturing facial images is considered intrusive compared to biometric methods like fingerprint or iris scans.
* **Difficulty in Spoofing:** It is significantly more challenging for someone to mark proxy attendance using someone’s Photograph compared to RFID cards.
* **Utilization of Existing Infrastructure:** Many classrooms already have installed cameras that can be leveraged for implementing face recognition technology.

**The typical workflow of a face recognition based attendance system involves the steps:**

* **Student Enrolment**: Reference images of students are. Stored in a face database along, with their corresponding ID/name.
* **Monitoring Classroom Attendance**: Cameras installed in the classroom capture either real-time video or periodic images. Transmit them to the system during class hours.
* **Identifying Faces**: The system utilizes computer vision techniques to detect and extract the faces of students attending the class from the camera feed.
* Recognizing Faces: The extracted faces are compared with a database of enrolled students to determine their identities and verify their presence.
* **Recording Attendance**: The identified student IDs are utilized to mark attendance in a database.
* **Generating Reports**: Administrators have the ability to query the database and generate attendance reports, for classes individual students, or customized time periods.

### **Review of Related Work:**

Several research efforts have investigated applying face recognition to automate attendance in academic environments. Here is a review of key works in this domain:

Ghalib et al. (2019) developed a portable IoT-based face recognition device for automating attendance. It uses OpenCV for face detection and recognition. On detecting a student's face, their attendance is marked by updating a central database. Testing during actual lectures showed significant time savings compared to the manual method.

Anushka et al. (2018) implemented a face recognition system using OpenCV and a linear SVM classifier. It captures classroom images, detects faces, matches them against an enrolled student database, and logs attendance in a spreadsheet. The system achieved 98.3% accuracy in testing.

Syafeeza et al. (2014) proposed a convolutional neural network architecture for face recognition that is robust to pose, illumination, and expression changes. They report over 99% accuracy on the AR face database, demonstrating the potential for high performance.

Yamini et al. (2019) developed an attendance system using OpenCV and Python. It detects and recognizes student faces from the classroom camera feed and logs attendance in a text file. They use Haar cascades for face detection and LBPH features for face recognition.

Guoqiang et al. (2013) proposed a 3D face recognition approach by matching curves extracted from depth images. They tested on the FRGC v2.0 dataset, achieving a 97.3% recognition rate to confirm the viability of attendance systems.

Arsenovic et al. (2017) designed a real-time face recognition attendance system called FaceTime using deep convolutional neural networks. Testing on YouTube Faces DB showed excellent accuracy of 98.74% despite challenges like varying pose, lighting, and expressions.

Pisarevsky et al. (2017) developed a system fusing face recognition and audio processing to enhance accuracy. The face recognition module uses PCA and achieves 95-97% accuracy in its testing. The audio processing identifies the lecturer's voice to reduce false face matches.

Czuszynski et al. (2018) implemented a Raspberry Pi-based system using a haar cascade classifier for face detection and Histograms of Oriented Gradients combined with linear SVM for recognition. It could recognize 15 students with 95% accuracy.

Chew et al. (2019) proposed using pre-trained deep convolutional face detectors like FaceNet and DeepID3 to overcome the limitations of classic methods like LBP and Eigenfaces. Their testing on YouTube Faces showed over 99% accuracy is achievable using deep learning models.

The review shows that advances in computer vision and deep learning have enabled face recognition attendance systems to achieve high accuracy exceeding 95% in many cases. The technology has matured sufficiently for real-world deployment. Most works focus on improving recognition accuracy and processing speed. However, few papers have addressed the ethical and privacy concerns that could arise when deploying face recognition on students. This is an important area needing further analysis.

# **Literature Review:**

The traditional process of taking attendance in educational institutions and organizations is time-consuming, tedious, and prone to errors. Calling out names or circulating an attendance sheet can disrupt classroom activities and meetings. To address these issues, automated attendance systems using facial recognition technology have been proposed as an efficient alternative. Facial recognition for attendance involves capturing images of students/employees, extracting facial features from the images, and comparing them to a database of authorized faces to detect matches and mark attendance automatically (Arsenovic et al., 2017). This literature review examines prior research on developing and implementing facial recognition-based attendance systems.

A key consideration in facial recognition is the accuracy of the system in correctly identifying authorized individuals. Shah, Bhandare, and Bhirud (2021) developed a system using Principal Component Analysis (PCA) and Discrete Cosine Transform (DCT) that achieved 95% accuracy in recognizing 40 students. Varadharajan et al. (2016) report 96% accuracy in recognizing 10 students using Viola Jones and LBPH facial recognition algorithms. Arsenovic et al. (2017) achieved 100% precision and recall using deep convolutional neural networks to train a system on over 5000 images of 100 students. The studies affirm the capability of facial recognition to reliably identify students and employees if trained adequately on a sizable database of facial images.

Popular tools and techniques used for developing facial recognition attendance systems include OpenCV, Deep Learning, and IoT devices. Shah et al. (2021) and Varadharajan et al. (2016) used OpenCV libraries in Python to implement real-time face detection and recognition pipelines. Arsenovic et al. (2017) employed deep convolutional neural networks for robust face identification. Yadav and Bhole (2019) designed an IoT-based portable attendance device using a Raspberry Pi microcontroller. The studies demonstrate the applicability of both classical machine learning and deep learning approaches using standard libraries and devices for building automated attendance solutions.

An intuitive user interface improves the ease of use and adoption of attendance systems. Awasthi and Awasthi (2022) developed a graphical user interface using Python Tkinter for students to input IDs and capture images. Bussa et al. (2020) created a web application with the Django framework to display attendance statistics and records. Pandey, Pitale, and Sharma (2020) implemented a system accessible through desktop and mobile apps. The research highlights the need for customizable and responsive multi-platform interfaces to enhance user experience.

Storing attendance data securely is critical for integrity and privacy. Many studies employ local CSV files and databases to log attendance records (Pandey et al., 2020; Bussa et al., 2020). Shah et al. (2021) used MySQL tables accessed via PHP scripts. However, cloud-based storage on platforms like Google Sheets enables convenient access and collaboration (Awasthi & Awasthi, 2022). Regular backups must be maintained to prevent data loss. Automated timestamping of attendance events enables accurate record-keeping for future reference.

Despite the potential benefits, facial recognition attendance systems face challenges in real-world deployment. Variations in lighting, poses, expressions, and occlusion can affect recognition accuracy (Shah et al., 2021). Enrolling a large number of authorized individuals in the database is cumbersome (Varadharajan et al., 2016). Privacy concerns over collecting and storing biometric data are increasing (Yadav & Bhole, 2019). Slow processing on large databases can lead to delays (Bussa et al., 2020). Further research is needed to address these limitations for seamless real-time attendance tracking.

Some studies propose hybrid systems integrating facial recognition with other biometrics like fingerprint scans to improve accuracy and security. Al-Muhaidhri and Hussain (2019) developed a system using both facial and fingerprint recognition to avoid issues when either modality fails independently. Multimodal biometric systems provide redundancy and help address the limitations of unimodal approaches. However, this increases system complexity and costs. The tradeoffs must be evaluated based on the application context and requirements.

Miniaturized embedded devices are gaining attention for portable attendance systems. Yadav and Bhole (2019) designed a handheld IoT device with Wi-Fi connectivity for classroom attendance. Raspberry Pi cameras have been used to develop pocket-sized embedded recognition systems (Bussa et al., 2020). These embedded solutions offer advantages like mobility, low power, and cost-effectiveness. However, they have limited computing capability suitable only for small databases. Cloud offloading could help overcome performance constraints.

Automated attendance systems can save time, reduce errors, and improve productivity in organizations. Yadav and Bhole (2019) found over 90%-time savings compared to manual attendance. Bussa et al. (2020) reduced wrong attendance markings from 15% to 2%. Varadharajan et al. (2016) highlighted enhanced classroom discipline using real-time alerts. Shi et al. (2021) discussed increased employee productivity and engagement in workplaces. Despite the benefits, concerns over consent, data privacy, and bias in facial analysis algorithms persist (Shah et al., 2021). Legal and ethical policies must regulate biometric data collection and use. Overall, facial recognition attendance systems have demonstrated value, but thoughtful oversight is required as adoption increases.

This literature review synthesized key research on using facial recognition for automated attendance monitoring. The technique shows reliable accuracy and performance if trained adequately on representative facial datasets. Both classical and deep learning techniques have proved effective using standard libraries like OpenCV. Cloud platforms offer convenient storage and access to attendance records. However, challenges around privacy, biases and scalability need to be addressed through legally compliant, and ethical practices. With further research on robust and inclusive algorithms, facial recognition promises efficient and ubiquitous automation of attendance systems.

Face recognition technology has become an important biometric method for automated attendance systems due to its non-invasive nature and high accuracy compared to other techniques. Ogbuju et al. (2020) proposed an attendance record system called "DLFacer" using face recognition for a Nigerian university. The aim was to automate the traditional manual attendance process which was time-consuming and prone to inaccuracies like ghost marking.

The methodology combined CRISP-DM and object-oriented design. A dataset of 20 images of 9 student faces was collected and Faster R-CNN (FRCNN) algorithm was used for modeling. The system architecture had 4 modules - user verification, face detection, recognition and attendance marking. It was implemented as a web application with CCTV camera input. Results showed 97-99% accuracy in recognizing student faces and marking attendance. The system prevented ghost marking, and was more efficient than previous works like Anushka et al. (2018) at 98.3% accuracy.

Several face recognition techniques have been applied for automating attendance systems. Anushka et al. (2018) used linear SVM and deep learning for a 98.3% accurate system that could handle real-time variations. Riya et al. (2015) utilized Bluetooth tags and Android app for automated detection and recording as the lecturer moved around. Sunaryono et al. (2019) also developed an Android app with face recognition. Kewalramani (2018) and Sbeha et al. (2017) applied principal component analysis (PCA) method for feature extraction. The PCA eigenfaces approach was also used by Pooja et al. (2017) along with histogram of gradients and local binary patterns for face authentication.

Syafeeza et al. (2014) proposed a convolutional neural network architecture that could process occlusion, expression, and illumination changes with 85-99% accuracy. Similarly, Ozmen and Yurtkan (2015) combined Viola-Jones face detection with non-subsample contourlet features and a classifier for automated attendance marking on Yale and BU-3D face datasets. Yamini et al. (2019) used Viola-Jones and LBP histograms with a CCTV camera for the classroom attendance system. While these PCA and machine learning methods were effective, deep learning approaches like FRCNN have shown higher accuracy and robustness.

The DLFacer system by Ogbuju et al. (2020) demonstrates a real-world application of deep learning-based face recognition for automating attendance in Nigerian universities. The results are significant for reducing time consumption, preventing inaccuracies, and enforcing lecture attendance policies. The system is cost-effective by avoiding specialized hardware or software. The high 97-99% accuracy surpasses previous works and can be further improved using larger datasets. While privacy issues need to be addressed, the research provides a useful model for automating attendance recording in African educational institutions still reliant on manual processes.

Overall, face recognition is gaining prominence for automated attendance systems due to high user comfort compared to other biometric techniques. The initial PCA and machine learning approaches have been superseded by deep convolutional networks like FRCNN that show greater accuracy under variabilities. Real-world applications use CCTV and mobile cameras with cloud platforms for cost-effective deployment. Key challenges are privacy protection, accuracy improvements with larger datasets, and seamless integration with student information systems. As cameras and computing resources improve further, deep face recognition promises to become a ubiquitous technology for attendance automation across educational institutions and workplaces globally.

Facial recognition technology has gained prominence as a method for automated attendance monitoring in educational institutions. Tan (2018) developed a facial recognition-based attendance system for monitoring attendance in universities. The aim was to improve efficiency, accuracy, and accessibility compared to traditional manual systems prone to issues like ghost marking of absent students.

The system used Raspberry Pi with a Pi camera module and OpenCV for face detection and recognition. A local face database was created using student images. The captured faces during attendance were matched with this database to identify students and mark attendance in a MySQL database. The attendance records could be accessed through a website developed using WordPress. Testing showed high accuracy in recognizing and marking student attendance automatically.

Several studies have explored face recognition for automating attendance monitoring. Bhise et al. (2015) utilized Near Field Communication (NFC) tags and a mobile app where students tapped their tags on the lecturer's phone equipped with an embedded camera. This allowed fast attendance marking but reliability issues persist as tags could be misused. Senthamil Selvi et al. (2014) used cameras to capture employee images and match them against a face database. However, the system lacked portability by needing a PC.

Kumar Yadav et al. (2015) developed a fingerprint biometric attendance system using two microcontrollers. Although efficient, connectivity to a PC hampered portability. Hussain et al. (2014) employed Radio Frequency Identification (RFID) tags and readers for tracking student attendance. The web portal allowed remote access but still lacked portability.

Syafeeza et al. (2014) proposed a Convolutional Neural Network (CNN) architecture for face recognition which could handle variabilities like occlusion, expression, and illumination. Testing showed 85-99% accuracy but application to attendance was not explored. Ozmen and Yurtkan (2015) combined Viola-Jones face detection with contour features and a classifier for marking attendance on Yale and BU-3D face datasets. However, convenience and accessibility factors were not addressed.

The Raspberry Pi-based system by Tan (2018) allowed both efficiencies in attendance marking and remote accessibility of records through the website. The portable and self-powered nature using WiFi and battery is also ideal for classroom settings. However further testing is required on larger university datasets to evaluate accuracy and robustness.

Kewalramani (2018) also used Raspberry Pi and face recognition to automatically mark attendance in a CSV file. The system could hold multiple classes and subjects but lacked a centralized database and remote access via the website. Pooja et al. (2017) detected faces in classroom videos, extracted features like Histogram of Gradients, and used eigenfaces classification to recognize individuals and mark attendance. However, video processing can be computationally intensive.

Saha et al. (2017) evaluated different classifiers like Support Vector Machines (SVM) and CNN for detecting and recognizing faces. CNN gave the highest accuracy of 97.5% on the Yale face dataset, showing promise for attendance systems. Riya et al. (2015) developed a mobile app linked Bluetooth-enabled attendance system where the lecturer's proximity automatically recorded student attendance. This allowed fast and paperless attendance without facial recognition.

Several issues persist in existing systems like privacy of captured images, processing overhead, lack of portability, and website accessibility. The Raspberry Pi platform can address many of these concerns using its portable and wireless features, alongside OpenCV for efficient face processing. MQTT protocols can enable real-time communication between Pi and the database. Access control policies need to be implemented for privacy protection.

While facial recognition accuracy continues to see enhancements with emerging approaches like CNN, practical challenges relevant to educational institutions need to be considered as well. Factors like cost, portability, large user groups, ease of use, and accessibility impact adoption. Thus, a balance between automation accuracy and user-centric design is required while developing attendance systems.

Accurate and efficient student attendance recording is an essential administrative function in educational institutions. However, traditional paper-based attendance has several drawbacks such as being time-consuming, prone to errors, and allowing scope for proxy attendance (Jadhav et al., 2017).

To address these issues, educational institutions are adopting automated biometric attendance systems based on fingerprint, iris, RFID, and face recognition. Among these, face recognition is emerging as a popular choice due to being contactless, non-invasive, and easy to deploy (Wang et al., 2017). This literature review examines recent studies on face recognition-based attendance systems with the aim of analyzing the techniques used and key findings of implementing such systems in educational settings.

Bhise et al. (2015) developed a Near Field Communication (NFC) and camera-enabled mobile app-based attendance system where each student is issued a unique NFC tag during enrollment. For taking attendance, teachers tap these NFC tags using their mobile devices which also capture the student's photo. The system then transmits this data to the college server for verification and attendance marking.

Key benefits of this system are the ease of use of NFC technology and fast connectivity speed which greatly streamline the attendance process. However, limitations include a lack of automation in detecting proxy attendance and dependence on the teacher's mobile device for system functioning.

Selvi et al. (2014) implemented a face recognition attendance system where student facial images are captured using a camera and matched against a stored database of images. On finding a match, the system marks the attendance of that student in the database.

This system uses a skin-color-based face detection algorithm to improve accuracy. A key advantage stated is the storage of attendance data on a highly secure server. Limitations of this study are the use of a fixed standalone system rather than a portable device and suitability mainly for staff attendance rather than large student bodies.

Hussain et al. (2014) developed an RFID-tagged student ID card-based system where students place their cards before an RFID reader to register attendance. The key difference from NFC-based approaches is the storage of student details and attendance on a web portal for easy online access.

However, dependency on wired RFID readers reduces portability while the use of RFID cards instead of biometrics also allows potential proxy attendance.

Lad et al. (2017) implemented a system using iris scanning for student attendance. Students have to present themselves before an iris scanner which verifies identities against a database and records attendance.

This approach automates the entire process and minimizes proxy attendance changes due to the use of biometrics. However, iris scanning requires active student cooperation and extensive infrastructure.

Prabhavathi et al. (2017) developed a classroom face recognition system using high-definition cameras and computer vision algorithms. Captured faces are matched against an enrolled student database for marking attendance.

The limitations identified are the effects of lighting and angles on face recognition accuracy. The system also cannot differentiate between students physically present and video footage.

Analyzing these studies highlights certain key aspects to consider when designing effective and practical automated attendance systems based on face recognition:

Using biometrics like face, iris, or fingerprints instead of RFID cards or NFC tags prevents proxy attendance more reliably. Among biometrics, face recognition provides a good balance of accuracy and usability.

Portable and real-time systems using technologies like small cameras, embedded systems, and machine learning algorithms enable unobtrusive and convenient student attendance monitoring.

Robust face detection and recognition algorithms are essential to address challenges like varying lighting, facial expressions, and angles. Hybrid approaches can improve accuracy.

Storing attendance data on secure servers and providing web portal access ensures transparency, and easy access and prevents tampering.

Although initial setup requires time and expense, automated attendance systems greatly reduce manual effort and paperwork over the long term.

Student behavior and acceptability are important considerations for the feasibility of any new attendance monitoring system.

Regular enhancement of facial databases and algorithms is required to account for students joining each academic year.

Thus, face recognition has strong potential for automated attendance systems owing to its contactless nature, but factors like accuracy, security, infrastructure, and costs need to be strategically addressed.

Automated face recognition attendance systems offer significant benefits over traditional paper-based or manual systems in educational environments. They enable convenient, real-time attendance monitoring without disrupting classes while enhancing the accuracy and reliability of records. However, robust algorithms and infrastructure capabilities are essential for acceptable performance across varying conditions. Overall, the studies reviewed highlight the feasibility of face recognition for efficient and automated attendance management after considering critical design factors.

# **RESEARCH QUESTIONS:**

1. How does the deployment of a face recognition attendance system affect educational institutions' overall efficiency and productivity?
2. What are the primary advantages and drawbacks face recognition attendance system?

## **Hypothesis:**

**The null hypothesis**

***H0:*** No, the system-based face recognition attendance system is not an optimal solution overall.

**The alternative hypothesis**

***Halt:*** Yes, the system-based face recognition attendance system is the perfect optimal solution.

## **Problem statement:**

Keeping track of attendance manually in university classrooms is a hassle. Consumes a lot of time. Professors have to pass around attendance sheets or call out names, which interrupts their lectures. This whole process can take, up to 20 minutes for classes eating into instruction time. Another big issue is proxy attendance, where students mark their friends as present. Dealing with all the paperwork and generating attendance reports requires a deal of work.

Automated attendance systems that use fingerprint readers, RFID cards, or NFC tags have their limitations. Fingerprint scanning leads to queues and invades personal space. RFID cards can be easily passed around for others to mark proxy attendance. NFC-based mobile apps require participation from both students and faculty members. What I need is an automated system.

Some universities have introduced automated swipe card systems where students tap their ID cards on readers when they enter classes. However, this still creates queues during times. There are also problems with students swiping cards for classmates allowing proxy attendance to continue unchecked. The maintenance of card readers and integrating them with university databases requires effort.

Fortunately, recent advancements in facial recognition technology offer a solution to overcome the challenges faced by methods and card-based tracking systems. By utilizing video feeds, from existing surveillance cameras in classrooms, I can accurately. Identify students in time using advanced computer vision techniques.

This automated system, for tracking attendance through face recognition has the potential to replace processes resolve issues related to proxy marking, and eliminate the need for hardware such as card readers. Additionally, it can provide analytics on aspects like latecomers, early departures, and performance metrics.

However, performing time recognition on large groups of people under different conditions poses a significant challenge in the field of computer vision. To achieve performance levels algorithms must be finely tuned using datasets and hardware that are tailored to university environments. Moreover, integrating this system with existing university information systems to access enrolment data and attendance records adds another layer of complexity. It is crucial to prioritize user privacy protection throughout this process.

Therefore, the main objective is to develop an automated attendance system that utilizes cutting-edge facial recognition technology specifically designed for university settings. This system should be efficient, accurate, resistant to tampering or manipulation cost-effective, and user-friendly so that faculty members can focus more on tasks, than administrative duties. Implementation of such a system has the potential to significantly enhance productivity and improve learning outcomes within the university ecosystem.

## **Aim:**

The objective of this project is to create a system that automates student attendance monitoring using facial recognition technology. The system will use CCTV cameras to capture images in classrooms and employ computer vision techniques, such, as neural networks to detect faces in real time. It will then extract features from the detected faces and encode them into faceprint templates. These templates will be matched against a database of enrolled student’s facial images to identify who is present in the class.

Once recognized successfully the system will automatically update the attendance records for students in the database. It aims to achieve accuracy when faced with challenges like different poses, lighting conditions, expressions, or obstructions. Moreover, it should be able to withstand attempts at spoofing using photographs or videos and provide face enrollment and recognition capabilities for student populations.

To facilitate ease of use administrators will have access to a user interface that offers real-time attendance dashboards, configuration settings, and report generation features. Additionally, there will be notifications for unenrolled individuals detected by the system. Ultimately this envisioned system aims to eliminate the tracking of attendance while providing detailed insights, into participation patterns.

The ultimate goal is to develop an efficient and trustworthy facial recognition system that can seamlessly monitor and log student attendance without causing any disruption.

This approach offers an advantage; it not only saves faculty time but also improves the accuracy of records when compared to traditional paper-based or manual methods that are prone, to errors and proxy marking. The project aims to explore the potential and limitations of utilizing face recognition algorithms in automating administrative tasks, within academic institutions.

## **Objective:**

I aim to develop a Smart Attendance System that's both compact and self-sufficient. Our goal is to enhance the speed of the attendance recording process compared to the system, which took around 3 seconds, per student. The system should have RAM capacity to accommodate the attendance database efficiently. It should be capable of recognizing individual's faces by utilizing a face database. Additionally, I will create a database for tracking attendance records. To facilitate access for administrators I will provide a user online interface to view the attendance database. Moreover, I will enable students or employees to store their facial data in the database through a graphical user interface (GUI). Lastly, it should promptly notify users, about the success or failure of face recognition operations.

# **Risks involved with this research:**

## **Ethical Risks:**

* **Invasion of Privacy:** Using cameras and facial recognition software to monitor attendance allows for the tracking of individuals' movements and behaviors, without their consent. This is a violation of privacy and undermines liberties. Even if students are informed about the system's use the power dynamics between institutions and students who are required to attend classes make it difficult to provide consent. Collecting and processing data such as images without offering an opt-out choice goes against the idea of individuals having control over their own information.
* **Potential Misuse of Data:** While the stated purpose may be attendance tracking there is a risk that the face images and patterns collected by the system could be misused for surveillance or shared with entities like law enforcement without proper oversight. Using data gathered for one purpose. Later using it for other reasons raises ethical concerns related to data misuse. Additionally, the lack of transparency regarding algorithms raises questions about accountability in cases where discrimination or errors may occur.
* **Algorithmic Bias:** Similar to all AI systems the accuracy of face recognition technology is influenced by the training data used. Algorithmic bias can result in error rates and false negatives for minority groups if these systems are trained on datasets that do not adequately represent populations. This could disproportionately affect attendance records, for marginalized communities.

Although it is important to implement measures such, as data collection and bias testing it is challenging to achieve algorithmic neutrality due to deeply ingrained societal biases.

The widespread utilization of surveillance methods like cameras and facial recognition normalizes the idea of monitoring and the erosion of privacy among younger individuals. This normalization could potentially result in a society where mass surveillance and the loss of autonomy are considered the norm. It is crucial to consider the short-term advantages of these systems in comparison to their long-term impact, on shaping attitudes and behaviors towards surveillance.

## **Legal Risks:**

* **Violations of GDPR:** When it comes to the processing of data, such, as images explicit consent is required under GDPRs "special category." Using recognition for monitoring attendance without students ' approval is likely to go against GDPR regulations. If violations occur substantial fines of up to 4% of turnover can be imposed under the GDPR accountability framework. It is crucial to incorporate consent processes and data protection impact assessments into the system design to ensure compliance.
* **Violations of the Equality Act:** Algorithmic biases can lead to rates of negative or inaccurate attendance logging for protected groups, including ethnic minorities, women, disabled individuals, and LGBTQ+ individuals. This could potentially contribute to discrimination under the Equality Act. Institutions must strive for accuracy across all demographics through testing and address any disparities that may arise. The lack of transparency makes it more challenging to identify biases.
* **Infringements on Human Rights Act:** The Human Rights Act integrates rights from the European Convention on Human Rights into UK law including privacy rights, family life rights, freedom of expression rights, and the prohibition against discrimination. Widespread monitoring using facial recognition technology could potentially violate these rights. It is necessary to demonstrate a balance, between attendance goals and invasion of privacy in order for such monitoring practices to be lawful.

## **Professional Risks:**

* **The negative impact caused by criticism:** When the media highlights privacy breaches, data leaks, or biases, in algorithms, it can harm the reputation of institutions. This backlash from the public and the subsequent loss of trust can also affect student enrolment. It's important to note that the damage to reputation resulting from publicized failures may outweigh any perceived benefits derived from implementing an attendance system. To address this, it is crucial to establish policies conduct audits, and have crisis response plans in place.
* **Additional training requirements for staff:** In order to comply with privacy laws and mitigate risks effectively administrators, IT personnel, and other individuals involved in running the system need training. They should be educated on topics such as data protection measures, bias testing procedures, and appropriate usage policies. It's essential to consider the costs associated with continually training personnel as processes evolve.
* **Increased legal responsibility:** Schools and universities have a duty of care towards their students. By implementing a software system that carries risks there is an increased legal liability for issues like data breaches or failures in recognition systems. Without security measures and testing protocols, in place, fines, lawsuits, damages claims, and settlement costs could arise. Therefore, it is necessary to implement control measures to minimize liability.

## **Societal Risks:**

* **Loss of Trust in Human Connections:** When classrooms are under surveillance with cameras and tracking systems it can damage the sense of trust and connection among students and teachers. People may hesitate to communicate because they fear being watched. This can have an impact on learning and undermine the overall social experience within the classroom. The presence of technology-mediated supervision diminishes the natural learning environment.
* **Exclusion of Marginalized Communities:** The visibility of surveillance tools, such as cameras can make groups, like minorities and women feel uncomfortable due to experiences of prejudice. As a result, they may choose to avoid spaces with monitoring leading to self-exclusion from interactions and learning opportunities. Invisible forms of exclusion emerge when these populations fear participating.
* **Normalization of Surveillance Culture**: Introducing surveillance and tracking at an age normalizes these practices for students. This normalization could potentially result in a society that tolerates monitoring while sacrificing privacy rights. It is crucial for students to be educated about the motivations, behind these technologies well as their associated risks in order to develop critical perspectives instead of blindly accepting them.

# **Existing Scenario Analysis:**

## **Paper-based Attendance**

One of the traditional methods of taking attendance is, by using paper sheets or conducting roll calls. In this approach, the professor manually marks students as present or absent either by calling out their names or passing around a paper sheet for signatures.



However, relying on paper-based attendance has limitations:

* It is time-consuming. Disrupts the flow of the class when names are called out or papers are circulated.
* Attendance records can be falsified through impersonation or forged signatures.
* Paper sheets can easily get lost, damaged, or tampered with.
* The method does not account for timeliness as even late arrivals may be marked as present.
* There is no automated analysis with this system to derive insights from attendance data.
* Professors have to convert the data into a digital format for analysis.
* Errors can easily occur during transcription from paper records into formats.
* Storing physical sheets over periods becomes cumbersome.

Overall relying on paper-based attendance creates a workload for staff members while also being susceptible, to inaccuracies and manipulation. Analyzing and reporting long-term attendance trends becomes challenging with records.

## **Swipe Card Attendance**

Some universities have implemented a system where students can use swipe cards to record their attendance by swiping their ID cards through a reader when entering the classroom.



This eliminates the need, for signing paper sheets and offers advantages over paper-based systems.

* It speeds up the check-in process as there is no longer a requirement to manually sign paper sheets.
* It reduces the risk of falsification since it relies on each student’s personalized ID card.
* Digital records are easier to store, analyze and export.
* Timestamps provide details on arrivals and early departures.

However, there are some limitations to consider:

* There is a cost associated with installing card readers and network infrastructure.
* Data analysis still requires effort. Is reactive rather than proactive.
* There is a possibility of someone swiping a student’s card if it is given to them.
* The analytics capabilities are limited. There is no automation based on attendance data.

In summary, while swipe cards improve efficiency and accuracy, in tracking attendance they do not possess analysis features. Additionally, it's important to consider the installation costs.

## **Fingerprint-based Attendance**

Some educational institutions utilize fingerprint scanners to automate attendance tracking. Each student provides their fingerprint during every class ensuring a match to their identity.



Here are some advantages of using biometric fingerprint scanning compared to card swiping methods:

* It offers a way of uniquely identifying students using their biometrics making it nearly impossible to counterfeit.
* The system maintains a level of accuracy, in recording and matching student identities.
* Scaling the system to include classrooms is cost effective requiring minimal additional expenses.

However, there are drawbacks associated with this approach:

* The initial investment in acquiring scanners and setting up the networking infrastructure can be quite high.
* Concerns regarding hygiene arise due to users touching the scanners regularly.
* The analytics capabilities provided by these systems may be limited, often relying on reporting and analysis.
* Privacy concerns revolve around the misuse. Mishandling of biometric data.

While fingerprint scanning improves the verification process it still lacks some of the automation and advanced analytics features that are possible, with software-driven attendance systems.

**Proposed System for the Project:**

**My proposed project system consists of following Hardware:**

* Webcam
* Computer system
* External Webcam
* Connectors

**The Software used for proposed system.**

* Python with PyCharm
* NodeJS with express framework
* Angular

**The Database used for proposed system.**

* PostgreSQL

# **Methodology:**

The objective of this project is to create an automated system, for monitoring attendance using facial recognition technology. The system will use a camera to capture images of students and staff members' faces and then compare their features with a database of known individuals to identify them. Once a face is recognized the system will record the person's attendance in a database.

To achieve this, I will utilize Python as the programming language along with libraries such as OpenCV, face\_recognition, NumPy, and PostgreSQL. Additionally, I built a user interface using Angular. Employ Node.js as a middleware server. The ultimate goal of this project is to develop a reliable solution for recognition and tracking that eliminates the need for manual attendance taking.

Setting up the development environment involves working with Python and its associated libraries through the PyCharm IDE. PyCharm offers a workspace for Python development that includes debugging capabilities, version control integration (Git and GitHub), and project management features. I will be utilizing the free student version of PyCharm.

For version control purposes all source code will be committed to a GitHub repository in order to facilitate collaboration among team members and maintain a record of changes made.

To enable facial recognition functionality, I will need to install libraries such as OpenCV, NumPy, face\_recognition, etc. which are readily available, through the Python Package Index (PyPI).

To install these components, you have two options. You can. Use PyCharms package manager GUI. Install them via pip on the command line.

For database connectivity, I will be using PostgreSQL. It can be installed on your development machine or, on a server. PyCharms database tools will help you configure the connection seamlessly.

When it comes to developing the frontend and Node.js backend Visual Studio Code is the tool of choice. You can. Run these components locally during development using VSCs integrated terminal and task runners.

Now let’s talk about how I organize our source code.

In the Python facial recognition module, you'll find components in their directories.

**The main entry point script**

A configuration module that holds application settings.

A database interaction module is responsible for executing SQL queries.

An image capture and processing module that utilizes OpenCV.

And a face recognition module built on top of the face\_recognition library.

As for the Angular source code, I follow best practices for its structure.

Each page component and service has its own directory.

Shared modules, models, services, and utilities are organized separately.

Assets like images and stylesheets are kept in their designated folders.

Similarly, in the Node.js backend, I organize routes, controllers, utilities, and configurations in an orderly manner.

To keep track of changes effectively I will use Git with commits to our GitHub repository. For additions or updates to our codebase feature branches will be created to isolate those changes before merging them.

Now let’s take a look, at how our facial recognition workflow operates:

1. Application start-up.

I load settings and configure webcams.

Initialize face recognition models.

Capture a frame; The webcam captures an image frame. Converts it into a NumPy array.

Detect faces: Utilize face\_recognition to identify faces in the captured image.

Encode faces: Leveraging face\_recognition generates 128 encodings, for each detected face.

Identify faces: Compare the face encodings with known faces stored in the database using distance metrics to find the match and recognize the person.

Log attendance: If a recognized face is found log the user's ID and timestamp into the database using the PostgreSQL INSERT statement with the time.

Alert for faces: In case there is no match found save the image of the person. Flag it as unknown. An email containing both the image and an unidentified ID will be sent to an administrator for review.

Stream video: Continuously process frames from the webcam to identify faces in time. Implement optimizations like skipping frames to improve performance.

To ensure reusability encapsulate facial recognition processes within a Python class. The entry point script initiates webcam functionality. Contains the event loop, for processing frames.

Database Structure

In the system, PostgreSQL will be used to store user profiles and attendance logs.

**The following tables will be created:**

**Users table:**

Id

User\_id

Name

Email

User\_type (student/staff/admin/visitor)

Password

**Attendance table:**

ID

Student\_id (referencing Users table)

Created\_at

Updated\_at

Deleted\_at

Is\_unknown

Class\_id

The Users table contains registered profile information while the Attendance table references the user\_id and records the check-in timestamp.

If necessary additional tables can be utilized to store face images and other application data. The Angular admin interface will provide database management capabilities.

## **Web Application User Interface**

For administrators and basic users, Angular will offer a frontend interface, with the following components:

1. Login Page: Includes fields for username/password input and a login button.
2. Admin Dashboard: Comprises a navigation menu, header, and content area.
3. User Management: Allows viewing a list of users and adding users through a form.
4. Attendance View: Displays tables showing check-ins sorted by user and date.
5. Account Settings: Enables updating the admin password.

The interface will be designed to be responsive and mobile-friendly using Angular Material components. Input validation on forms will ensure data integrity before submitting it to the API. Authenticated sessions will be required for accessing routes.

The dashboard sidebar facilitates navigation, between sections of the application.

The top section of the page shows the username and an option to log out. In the area, there are forms and tables that display information based on the view.

## **API Server Middleware**

To interact with the frontend and database a Node.js Express server will be used to implement a REST API. The main routes include:

/login: This route authenticates user credentials and returns a JWT token.

/users: This route retrieves a list of users. Allows for adding users.

/attendance: This route gets attendance records. Logs records.

When handling these routes parameterized SQL queries will be. JSON responses will be returned. The database logic is encapsulated to simplify API usage.

The Express server manages authenticated sessions using JSON web tokens. Upon login, a token is provided that must be included in requests. Decorators are utilized to verify and extract tokens for each route handler.

## **Project Development Approach**

The initial prototype will prioritize recognition using a sample image dataset. Once face detection and identification algorithms are functioning correctly the application will be expanded incrementally with the steps:

1. Transitioning from images to live webcam capture.
2. Expanding the PostgreSQL database schema.
3. Developing API routes. Implementing database logic.
4. Creating interface pages.
5. Setting up a Node.js middleware server.
6. Connecting frontend, backend, and facial recognition functionality.
7. Adding features such, as email alerts.
8. Conducting testing and refining algorithms for efficiency.

This flexible approach ensures that a functional solution is delivered early with time to improve its performance and reliability. Testing will be conducted throughout the implementation of each component.

In the stages I will focus on strengthening security optimizing speed and adding any remaining features. Thorough documentation will provide information, about the systems architecture, dependencies, APIs, and configuration.

This development methodology outlines a plan for building an automated attendance monitoring system supported by an architecture. The project leverages established technologies such as Angular, Node.js, PostgreSQL and OpenCV to create a facial recognition solution that can adapt to needs. By adopting an approach, I ensure visibility into our progress while maintaining a stable codebase that grows incrementally. Rigorous testing and comprehensive documentation efforts will result in a system ready, for production use.

## **Project Workflow**

The first step, in creating an automated system for tracking attendance using recognition involves gathering a dataset of images from classmates and known individuals. This dataset will be used to train and test the face recognition algorithms.

To create the dataset, I will collect facial images of classmates and staff from online sources ensuring that they are taken under controlled conditions. Each individual should provide one image where they are looking directly at the camera with an expression and the lighting should be consistent. I will crop the images to focus on the face and save them in PNG format at a size, such as 250x250 pixels.

For each image, I will use a structure like "[user\_id].png". In our database, I will store information such as names, ID numbers, and corresponding image filenames for each person. This structured dataset is crucial for evaluating the accuracy of face recognition during development.

Once I have assembled sample data the next phase involves installing libraries and dependencies. The core environment relies on Python 3.6 or with packages like NumPy, OpenCV, and face\_recognition installed. OpenCV allows us to capture and process images in time. Adam Geitgeys' face\_recognition library efficiently detects faces. Encodes them for recognition.

To facilitate development, on Windows, Mac, or Linux systems PyCharm Community Edition provides a Python IDE.

During the development process, I will make use of PyCharms debugging, version control, and project management features. Our source code will be. Tracked in a GitHub repository.

For managing user profiles and attendance logs I rely on PostgreSQL, as the database. PyCharm seamlessly integrates with database tools allowing us to handle connections and run queries directly within the IDE. To ensure compatibility with our facial recognition application please ensure that PostgreSQL 10 or a later version is installed and accessible.

Once the environment is set up our initial prototypes will concentrate on face recognition using a precompiled dataset. To simplify this task, I utilize the face\_recognition library which provides a to-use API for various machine learning algorithms. By loading test images into memory and calling the API each face is encoded into a 128 vector descriptor.

To identify faces I compare their encodings against those from the dataset using metrics such as distance or cosine similarity. The closest vector distance indicates a match which is confirmed if it falls below a threshold. The accuracy of recognition improves as our dataset encompasses variations in features.

I will conduct experiments to determine parameters such as model training epochs, distance thresholds, and image sizes. Rigorous testing will be performed to assess recognition rates under angles, expressions, and lighting conditions. All encoding and identification logic has been encapsulated into a FaceRecognizer class within our codebase.

Once I have achieved a recognition module using images successfully our next milestone involves transitioning, to real-time video processing.

The VideoCapture class, in OpenCV, allows us to access and process frames from a webcam stream. Each frame is converted into a NumPy array. Face detection using face\_recognition generates the encodings to identify individuals.

By comparing these encodings with the data stored in memory our application can annotate recognized individuals in time as they appear on the camera. During development, I have a script that tests recognition on video displaying the captured frames and annotations in an OpenCV window.

Once I have confirmed the algorithms with input, I shift our focus towards storage. I design a PostgreSQL database with a User’s table containing profile information and an Attendance table that logs check-ins. The database logic handles INSERT and SELECT queries through methods for abstraction.

When recognized faces are detected the application automatically inserts the user ID and timestamp into the Attendance table. For faces, I save them as image files and send them via email to be reviewed by an administrator. System logs keep track of metrics such as recognized users, unknown faces detected, and successful check-ins.

With the integration of our database attention is now directed towards creating a user interface. An Angular 12 frontend acts as a portal, for managing our system while the Node.js Express framework implements a REST API that connects the frontend to both recognition and database components.

The Angular CLI helps generate a project that includes components, like routers, services, guards, and core modules. The responsive template comes with pages such as login, dashboard, user management, and attendance page. Reactive forms ensure that user input is validated before being sent to the API. To ensure security route guards require authentication through JSON web tokens.

To streamline interactions with the Face Recognizer API and database I use the Express server. The server’s routes handle tasks such as user login, profile registration retrieving attendance records, and recording check-ins. Middleware is in place to authenticate requests and handle errors consistently. The frontend communicates with the backend asynchronously using HTTP.

During the stages of development, I focus on optimizing performance and enhancing reliability. I conduct tests to measure frame rates and recognition latency in order to identify any bottlenecks. By skipping analysis on frames, I strike a balance between responsiveness and accuracy. Additionally, our application allows for the configuration of parameters like model complexity based on computing resources.

Continuous testing is conducted to evaluate recognition using real-world images. Our administrator interface enables fine-tuning of thresholds and retraining of models using datasets. Detailed logs provide visibility into any runtime issues encountered while our error handling aims for failures when problems arise. I also provide documentation covering dependencies, API configuration options, and deployment guidelines.

This methodology follows a phased approach where initial efforts are focused on utilizing established libraries like OpenCV and face\_recognition, for core face recognition algorithms.

Once the basic concepts have been confirmed through a prototype, on a scale the project gradually evolves into a functional system. This progression involves implementing database storage followed by user interfaces and reliable infrastructure. Through this approach, practical features are developed while leaving room, for improvements, optimizations, and documentation.

### **Workflow diagram**

A diagram of a software system

Description automatically generated

# **Problems and Challenges faced while developing my Proposed system:**

Firstly, I encountered challenges while working on the proposed system.

To begin with, I used the PyCharm integrated development environment (IDE) for our coding work. Although PyCharm is a paid software I managed to obtain a one-year student version. Since our coding work was not graphical, in nature I had to write code for each component, such as obtaining frame size. Our process started by generating a video frame.

Next, I made attempts to connect a webcam and calculate the distance between it and the face being detected. I then focused on determining the location of the face within the frame. I created a function that consolidates all previously saved images into a variable and another function that computes matching percentages for recognition.

For our database needs I chose PostgreSQL. Whenever a new face was detected, its associated data was stored in the database. If an unidentified person was found their facial image was saved in a folder. Marked as "unknown" in the database. The new face was also added to the facial image variable and an email alert with their face. It was sent to the administrator.

Once I completed work, on the Python-based facial recognition module our focus shifted towards using Angular for front-end development and Node.js for back-end development.

First, I started creating the login page and admin dashboard in Angular. After that, I focused on completing the user interface. In Node.js I developed APIs to establish a connection, between the end and the database. Unfortunately, due, to time constraints I couldn't work on the user dashboard and staff dashboard.

## **Comparative Analysis:**

existing system vs proposed system vs implemented system.

**Existing System:**

* The current method involves taking attendance on paper.
* Professors call out names. Pass around sheets for students to sign.

**Limitations:**

* This process takes up a lot of time. Disrupts lectures.
* It is easy for someone to attendance by forging signatures.
* Paper records are susceptible to damage or getting lost.
* Analyzing attendance data from paper records is not possible.

**Proposed System:**

* I propose an automated attendance system using facial recognition technology.
* Cameras in the classroom capture images. Apply face detection algorithms.
* The faces detected are compared against a database of student images.
* Attendance records are automatically updated based on recognized students.

**Benefits:**

* This new system allows for more attendance marking without disrupting classes.
* By eliminating proxy attendance, I ensure accuracy in tracking student presence.
* Digital records make it easier to analyze and store attendance data.
* Compared to systems our touchless approach ensures hygiene.

**Implemented System:**

* I used a camera module and OpenCV for capturing classroom images.
* Student facial images were stored in a database for matching against detected faces.
* Attendance was logged by updating a PostgreSQL database.
* Accessing attendance reports can be done through a website.

**Outcomes:**

* With this implemented system marking attendance has become significantly faster.
* The accuracy of detecting and recognizing students is also high.
* Proxy attendances have been completely eliminated.
* Additionally digitized records make it easier to generate reports.
* Remote access to the attendance data can be achieved through the website.

**Differences:**

* Of using MySQL our implemented system utilizes PostgreSQL as the database management system (DBMS).
* The actual implemented system provides access to the attendance records via the designated website interface.
* During testing I worked with a dataset of student images.
* While deep learning was proposed it was not implemented in this system.
* The system does not include the functionality, for detecting faces and sending notifications.

**Future Scope of Development for my Proposed System:**

To expand on the implementation my plan is to create websites specifically designed for university staff and students. This will enhance accessibility. Provide customized interfaces tailored to meet the needs of each user.

In addition, I will enhance the admin panel by introducing features such, as the ability to download attendance reports. These reports will empower administrators to conduct in-depth analyses and make data-driven decisions regarding policies, interventions, or resource allocation. For instance, they can investigate patterns of absenteeism across courses or demographic factors.

One crucial aspect I aim to prioritize is improving the reliability of our facial recognition module. Currently, our system may encounter difficulties with low-quality images that are commonly encountered in real-world conditions. To address this challenge, I will explore deep learning algorithms like neural networks to make the face detection and recognition steps more resilient. Additionally augmenting our training data with a range of images captured under lighting conditions, angles, and expressions can significantly boost accuracy.

On the staff website instructors will have access to student profiles and lecture attendance records. With this information at their fingertips, they can easily identify students who require interventions due to absences. Moreover, I will implement a notification system that alerts staff whenever unknown individuals are detected in their classrooms—enhancing security measures.

For students' convenience, our website will allow them to check their attendance statistics, for classes and time periods.

This encourages openness. Gives students the opportunity to track their attendance levels during the year, term, month, or week.

# **Conclusion:**

Automated attendance monitoring through the use of facial recognition technology has the potential to greatly improve efficiency, accuracy, and analytics when compared to paper-based or manual systems. However, it is crucial for institutions to carefully assess the legal and social risks associated with this biometric technique before implementing it.

## **Summary of Findings:**

The initial problem statement highlighted the limitations of existing attendance tracking methods, in university classrooms that rely on paper sheets or roll calls. These manual processes are time-consuming, lectures. Are prone to inaccuracies such as proxy attendances or forged signatures. Exploring systems like RFID cards and fingerprint scanners also revealed drawbacks related to cost, scalability, and student perceptions.

As a solution face recognition technology was identified as a way to automate attendance in a non-invasive manner by utilizing existing classroom camera infrastructure. However, developing a real-time face recognition system for handling variables like lighting conditions, angles, and obstructions presented software engineering challenges. Additionally ensuring encryption of data and seamless integration with university databases were recognized as complex components.

The literature review examined 15 20 studies that explored the use of face recognition for automating attendance in workplace environments. Key techniques evaluated included OpenCV along, with machine learning and deep learning algorithms applied for detecting and matching faces against enrolled image datasets.

In controlled environments, the accuracy of recognition reached 95 99%. However, challenges arose when dealing with databases and real-world variability. It was crucial to analyze the advantages well, as addresses privacy and ethical concerns.

The main goal of the project was to create a facial recognition system that's fast tamper-proof in order to eliminate manual attendance processes. To achieve this a robust algorithm for face detection and identification was necessary especially when faced with conditions. The envisioned system would automatically update attendance records. Provide access to data through a user interface with notifications.

The proposed methodology involved a phased development approach using established technologies such as OpenCV, Node.js, and PostgreSQL. Starting with face recognition on controlled test images the prototype would gradually evolve into a functional system by adding database persistence first followed by user interfaces and infrastructure improvements. Constant testing and refining of algorithms were planned to use a process.

For the implementation phase, OpenCV and face\_recognition libraries were employed for detection and matching, against a small sample image dataset. When student faces were recognized updates were made to an attendance database stored in PostgreSQL. A website was also created to enable viewing of attendance reports. The outcomes of this implementation included attendance tracking compared to manual methods while preventing proxy attendance through biometric facial verification.

## **Analysis of Challenges and Limitations:**

However, I encountered obstacles when moving from a controlled prototype to an attendance system suitable, for a real academic environment.

**Small Image Dataset Issue:**

The accuracy and reliability achieved during testing were limited because I had a number of reference images for training and validation. As I scale up to university datasets with variations the performance could potentially decline.

**Accessibility Concerns:**

At present our system operates on a computer with a camera. To expand its usage across the campus I need to make it accessible on the university network from locations simultaneously.

**User Acceptance Considerations:**

Students' opinions about using facial recognition technology for attendance purposes may vary widely ranging from enthusiasm to skepticism or even suspicion. It is crucial that I generate awareness and gain buy-in from the student community before implementing this system institution.

**Enrollment Challenges:**

Maintaining high-quality enrollment images of every student every year requires time and administrative effort. I need to explore options for automating batch enrollment processes.

Hardware Variability Obstacles:

The diverse range of camera resolutions framing techniques and lighting conditions in lecture halls make capturing facial images quite challenging. Additionally, complications arising from occlusion or motion blur can hinder recognition.

## **Network Reliability Considerations:**

Integrating our system with university databases and accessing video feeds necessitates bandwidth and low-latency networks. Any disruptions in network connectivity could have consequences, on the functioning of our system.

Scalability is a concern when it comes to processing and matching faces against databases containing thousands of students especially if it needs to be done in time. It's crucial to have optimized systems in place to handle this.

Ensuring security is of importance when dealing with images and attendance data as they contain sensitive personal information. Encryption should be implemented both during data storage. While transmitting over networks to prevent any data leaks that could lead to legal or privacy issues.

Implementing face recognition technology for automating attendance, in universities requires investment. This includes setting up cameras networking infrastructure and high-end servers. Additionally, ongoing costs such as maintenance, upgrades, and training for administrators should also be taken into account.

While face recognition technology has made advancements it's important to acknowledge that software and infrastructure enhancements are still necessary to meet the demands of ensuring reliable, accurate, and high-performance automated attendance systems, for large universities.

# **Future Challenges in Face recognition attendance system:**

Face recognition technology has made progress in years paving the way, for automated attendance systems. However, like any technology, there are challenges and limitations that need attention as face recognition attendance systems become widely used. In this discussion, I will address three challenges I am likely to encounter in the upcoming years.

## **Addressing Spoofing and Presentation Attacks**

One major challenge is potential for spoofing and presentation attacks that could deceive face recognition systems. These attacks involve using photographs, videos, or masks of authorized individuals to gain access. Since face recognition relies on analyzing features it remains susceptible to deceptive tactics.

Researchers are working on developing detection and anti-spoofing measures that involve analyzing texture and depth in images to identify fakes. Nonetheless, spoofing techniques continue to evolve enabling attackers to employ methods through 3D modeling and AI-generated imagery. Combating these attacks will require modal biometric authentication and continuous advancement in anti-spoofing techniques. Additionally, training models with real-world data will boost system resilience, against presentation attacks.

## **Managing Diversity and Addressing Bias**

Fairness and accuracy issues still plague face recognition systems, particularly when it comes to groups.

Factors such, as age, gender, skin tone, and ethnicity can have an impact on the accuracy of facial recognition systems. Result in false rejection rates for certain groups. This bias arises from limitations in training data and variations in features.

Addressing bias requires the use of diverse and balanced training datasets. The generation of data can assist in augmenting real-world data diversity. It is also crucial to test systems for fairness across groups and make necessary adjustments to decision thresholds. Implementing regulations regarding technologies can contribute to ensuring deployment and achieving fair outcomes as these technologies become more widespread. Additionally conducting research into algorithms and customized feature extractors specifically designed for underrepresented groups could enhance accuracy even more.

## **Scaling Capacity and Uptime**

As face recognition attendance systems expand to cover organizations' events like concerts, theme parks, or other crowded venues ensuring accurate real-time identification presents scalability challenges. Processing image feeds with a number of faces simultaneously can overwhelm systems and cause delays or failures.

To overcome these challenges optimizing hardware performance is essential along with leveraging AI acceleration technologies like GPUs. Upgrading to networks and storage solutions would also be beneficial. Cloud-based deployments offer capacity options that can be considered well. However, it is important to incorporate redundancy measures and ensure uptime as any downtime could have consequences for events or venues. As venues grow larger in size the use of high-resolution cameras combined with video analytics becomes indispensable, for simultaneous people tracking purposes.

Improved face detection technology has the potential to effectively filter faces and reduce the burden of identification.

To summarize there are challenges, in enhancing accuracy reducing bias, and increasing capacity as I continue to develop face recognition attendance technology. By conducting research and fostering innovation in these areas I can expect face recognition to become faster, fairer, and more reliable, in the future. It is crucial to conduct testing and deploy these systems responsibly to ensure they fulfill their potential while safeguarding interests. If developed thoughtfully facial recognition attendance could become a technology that enhances convenience and security in our spaces and events.

# **Flow Chart**

# **Login**

**A diagram of a computer program

Description automatically generated**

## **Student/ Staff and visitor Form**

A diagram of a computer program

Description automatically generated

## **Change Password**

A blue rectangle with white text

Description automatically generated

# **Website Images**

## **Login**

**A screenshot of a computer

Description automatically generated**

## **User List**

**A screenshot of a computer

Description automatically generated**

## **Add Student/ visitor details:**

**A screenshot of a computer

Description automatically generated**

## **Staff List:**

**A screenshot of a computer

Description automatically generated**

## **Add staff Details:**

**A screenshot of a computer

Description automatically generated**

## **User Attendance:**

**A screenshot of a computer

Description automatically generated**

## **Staff Attendance:**

**A screenshot of a computer

Description automatically generated**

## **Classroom List:**

**A screenshot of a computer

Description automatically generated**

## **Add classroom number:**

**A screenshot of a computer

Description automatically generated**

## **Change Password**

**A screenshot of a computer

Description automatically generated**

## **Face Detection:**

**A person sitting in bed

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

## **User Images:**

**A screenshot of a computer

Description automatically generated**

## **Unknown person image sends to the admin**

**A screenshot of a computer

Description automatically generated**

## **External webcam:**

A hand holding a webcam next to a power bank

Description automatically generated

## **Connector for external webcam**

A white rectangular device with black wires

Description automatically generated

# **References:**

Al-Muhaidhri, G., & Hussain, J. (2019) ‘Smart attendance system using face recognition’ *International Journal of Engineering Research & Technology* 8(8), 858-861.

Arsenovic, M., Sladojevic, S., Anderla, A., & Stefanovic, D. (2017) ‘FaceTime-Deep learning based face recognition attendance system’ *IEEE 15th International Symposium on Intelligent Systems and Informatics (SISY)*, Subotica, Serbia, 14-16 Sept. 2017. IEEE.

Anushka, W., Akash, U., Ruchi, S., Nevil, P., Prashant, K. (2018) ‘Face recognition-based attendance management system using machine learning’ *International Research Journal of Engineering and Technology* 5(6): 1979-1983.

Awasthi, S., & Awasthi, S. (2022) ‘Facial recognition attendance system using Python’ *International Journal of Research Publication and Reviews* 3(1), 45-49.

Bhise, R., Khichi, S., Korde, S., & Lokare, R. (2015) ‘Attendance System using NFC Technology with Embedded Camera on Mobile Device’ *International Journal of Computer Applications* 119(16).

Bhise, S., Khichi, S., Korde, S., & Lokare, P. (2015) ‘Attendance System using NFC Technology with Embedded Camera on Mobile Device’ *International Journal of Current Engineering and Technology* 5(2), 960-962.

Bussa, S., Bharuka, S., Mani, A., & Kaushik, S. (2020) ‘Smart attendance system using OpenCV based on facial recognition’ *International Journal of Engineering Research & Technology* 9(6), 958-961.

Hussain, A., Dugar, A., Deka, G., & Hannan, S. A. (2014) ‘RFID based Student Attendance System’ *International Journal of Scientific Engineering and Technology* 3(1), 64-68.

Hussain, S., Dugar, D., Deka, G., & Bhuyan, H. (2014) ‘RFID based Student Attendance System’ *IJRET: International Journal of Research in Engineering and Technology* 3, 93-96.

Jadhav, A., Jadhav, A., Ladhe, T., & Yeolekar, K. (2017) ‘Automatic Attendance System using Face Recognition’ *International Research Journal of Engineering and Technology* 4(1), 512-515.

Kewalramani, S. (2018) ‘Automatic attendance system by face recognition using machine learning’ *International Journal of Engineering Sciences and Research Technology* 7(10): 116 - 121.

Kumar Yadav, M., Singh, A., Pujari, A. K., & Mishra, K. (2015) ‘Fingerprint based attendance system using microcontroller and LabVIEW’ *International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering* 4(4), 2652-2661.

Lad, P., More, S., Parkhe, S., Nikam, P., & Chaudhari, D. (2017) ‘Student Attendance System Using Iris Recognition’ *International Journal of Advanced Research in Computer and Communication Engineering* 6(3), 156-161.

Ogbuju, E., Adetayo, A.P., Obilikwu, P. (2020) ‘A Face Recognition System for Attendance Record in a Nigerian University’ *Journal of Scientific Research and Development* 19(2): 38-45.

Ozmen, B., Yurtkan, K. (2015) ‘Automatic exam attendance system based on illumination invariant face recognition’ *International Conference on Education in Mathematics, Science and Technology* 2: 139 - 143.

Pandey, D., Pitale, P., & Sharma, K. (2020) ‘Face recognition based attendance system using Python’ *Journal of Emerging Technologies and Innovative Research* 7(10), 703-70

Pooja, H., Shivani, K., Apurva., Akshata, J., Krushna B. (2017) ‘Automatic attendance using face recognition’ *International Journal on Recent and Innovation Trends in Computing and Communication* 5(12): 24 - 26.

Prabhavathi, B., Tanuja, V., Madhu Viswanatham, V., & Babu, M. R. (2017) ‘A Smart System for Attendance Monitoring by Face Recognition’ *IOP Conference Series: Materials Science and Engineering* 263, 042049.

Riya, L., Suruchi, G., Harshil, J., Harish, N. (2015) ‘Bluetooth smart based attendance management system’ *International Conference on Advanced Computing Technologies and Applications* pp. 524 - 527.

Saha, S., Singh, R., Jain, V., Sinha, A., & Vaidya, B. (2017) ‘Face recognition for attendance system using machine learning algorithms’ *International Conference on Algorithms, Methodology, Models and Applications in Emerging Technologies* (ICAMMAET), 1-5.

Selvi, K. S., Chitrakala, P., & Jenitha, A. A. (2014) ‘Attendance Marking System Using Face Recognition’ *Journal of Computer Science and Mobile Computing* 3(2), 258 – 267.

Senthamil Selvi, P., Antony Jenitha, R., & Chitrakala, S. (2014) ‘Face recognition based attendence marking system’ *International Journal of Computer Science and Information Technologies*, 5(4), 5700-5702.

Shah, K., Bhandare, D., & Bhirud, S. (2021) ‘Face recognition-based automated attendance system’ *In Innovative Computing and Communications* (pp. 945-952). Springer, Cham.

Sunaryono, D., Siswantoro, J., Anggoro, R. (2019) ‘An android based course attendance system using face recognition’ *Journal of King Saud University - Computer and Information Sciences* In Press.

Syafeeza, A.R., Khalil-Hani, M., Liew, S.S., Bakhteri, R. (2014) ‘Convolutional neural network for face recognition with pose and illumination variation’ *International Journal of Engineering and Technology* 6(1): 44-56.

Syafeeza, A. R., Khalil-Hani, M., & Liew, S. S. (2014) ‘Convolutional neural network for face recognition with pose and illumination variation’ *International Journal of Engineering and Technology* 6(1), 44-56.

Tan, S. J. (2018). ‘Facial recognition-based attendance monitoring system for educational institution’ Universiti Tunku Abdul Rahman.

Varadharajan, E., Dharani, R., Jeevitha, S., Kavinmathi, B., & Hemalatha, S. (2016) ‘Automatic attendance management system using face detection’ *In Online International Conference on Green Engineering and Technologies* (IC-GET) (pp. 1-7).

Wang, D., Fu, R., & Luo, Z. (2017) ‘Classroom Attendance Checking Based on Face Recognition’ In Y. Tan et al. (Eds.), Social Science, Education and Humanities Research. Atlantis Press.

Yadav, V., & Bhole, G. P. (2019) ‘Cloud based smart attendance system for educational institutions’ *In International Conference on Machine Learning, Big Data, Cloud and Parallel Computing* (Com-IT-Con) (pp. 169-174). IEEE.

Yamini, K., Mohan K., Sonia, P., Yugandhar, V., Bharath, K. (2019) ‘Class attendance using face detection and recognition with opencv’ *International Research Journal of Engineering and Technology* 6(4): 3822 – 3825

<https://www.freepik.com/free-photo/blond-businessman-happy-expression_1030605.htm#query=Security%20card&position=13&from_view=search&track=ais>

<https://www.freepik.com/free-photo/close-up-fingerprint-reader_902012.htm#query=Fingerprint%20attendance&position=11&from_view=search&track=ais>

<https://www.jetbrains.com/pycharm/>

<https://code.visualstudio.com>

<https://angular.io/docs>

<https://www.microsoft.com/en-us/microsoft-365/visio/flowchart-software>

<https://nodejs.org/en/docs>

<https://opencv.org/opencv-face-recognition/>

GitHub Link: <https://github.com/parthdavra/face-recognition-python.git>

# **Appendices:**

**Main.py**

import face\_recognition

import os, sys

import cv2

import numpy as np

import math

import psycopg2

from datetime import datetime, timedelta

import random

import smtplib

from email.mime.multipart import MIMEMultipart

from email.mime.text import MIMEText

from email.mime.image import MIMEImage

import schedule

import time

def face\_confidence(face\_distance, face\_match\_threshold=0.6):

range = (1.0 - face\_match\_threshold)

linear\_val = (1.0 - face\_distance) / (range \* 2.0)

if face\_distance > face\_match\_threshold:

return str(round(linear\_val \* 100, 2)) + '%'

else:

value = (linear\_val + ((1.0 - linear\_val) \* math.pow((linear\_val - 0.5) \* 2, 0.2))) \* 100

return str(round(value, 2)) + '%'

def mail\_send(user\_id, image, class\_id):

smtp\_server = "smtp.gmail.com"

smtp\_port = 587

sender\_email = "pdavra1997@gmail.com"

sender\_password = "qadmhqepsqsfsoyf"

recipient\_email = "parthdavra97@gmail.com"

msg = MIMEMultipart()

msg['From'] = sender\_email

msg['To'] = recipient\_email

msg['Subject'] = f'Unknown person is in the class number {class\_id}'

body = f'Unknown person id is {user\_id}'

image\_file = image

with open(image\_file, 'rb') as img\_file:

image = MIMEImage(img\_file.read(), name=os.path.basename(image\_file))

msg.attach(image)

msg.attach(MIMEText(body, 'plain'))

with smtplib.SMTP(smtp\_server, smtp\_port) as server:

server.starttls()

server.login(sender\_email, sender\_password)

text = msg.as\_string()

server.sendmail(sender\_email, recipient\_email, text)

class FaceRecognition:

face\_locations = []

face\_encodings = []

face\_names = []

known\_face\_encodings = []

known\_face\_names = []

def \_\_init\_\_(self):

self.encode\_faces()

def encode\_faces(self):

for image in os.listdir('faces'):

face\_image = face\_recognition.load\_image\_file(f'faces/{image}')

face\_encoding = face\_recognition.face\_encodings(face\_image)[0]

self.known\_face\_encodings.append(face\_encoding)

self. known\_face\_names.append(image)

def run\_recognition(self):

capture\_objects = []

index = 0

conn = psycopg2.connect(database="FaceRecognitionAttendanceSystem", host="127.0.0.1", user="postgres", password="a", port="5432")

cursor = conn.cursor()

while True:

video\_capture = cv2.VideoCapture(index)

if not video\_capture.isOpened():

break

capture\_objects.append(video\_capture)

index += 1

if len(capture\_objects) == 0:

print("No cameras found!!!")

while True:

frames = []

for video\_capture in capture\_objects:

ret, frame = video\_capture.read()

frames.append(frame)

for i, frame in enumerate(frames):

process\_current\_frame = True

if process\_current\_frame:

small\_frame = cv2.resize(frame, (0, 0), fx=0.25, fy=0.25)

rgb\_small\_frame = small\_frame[:, :, ::-1]

self.face\_locations = face\_recognition.face\_locations(rgb\_small\_frame)

self.face\_encodings = face\_recognition.face\_encodings(rgb\_small\_frame, self.face\_locations)

self.face\_names = []

for face\_encoding in self.face\_encodings:

matches = face\_recognition.compare\_faces(self.known\_face\_encodings, face\_encoding)

name = 'unknown'

confidence = 'Unknown'

face\_distances = face\_recognition.face\_distance(self.known\_face\_encodings, face\_encoding)

best\_match\_index = np.argmin(face\_distances)

if matches[best\_match\_index]:

name = self.known\_face\_names[best\_match\_index]

confidence = face\_confidence(face\_distances[best\_match\_index])

now = datetime.now()

dt\_string = now.strftime("%d/%m/%Y %H")

selectQuery = "select \* from attendances where student\_id = %s ORDER BY id DESC LIMIT 1;" % int(name.split(".")[0])

cursor.execute(selectQuery)

rows = cursor.fetchall()

if len(rows) == 0:

cursor.execute("insert into attendances (student\_id,class\_id, created\_at,is\_unknown) values (%s, %s, %s, %s);", (int(name.split(".")[0]), int(i), now, 'false'))

conn.commit()

if len(rows) == 1:

for student\_data in rows:

if int(student\_data[5]) == int(i):

if (student\_data[2].strftime("%d/%m/%Y") != now.strftime("%d/%m/%Y")) or ((student\_data[2] + timedelta(hours=1)).strftime("%d/%m/%Y %H") <= dt\_string):

cursor.execute("insert into attendances (student\_id, class\_id, created\_at, is\_unknown) values (%s, %s, %s, %s);",(int(name.split(".")[0]), int(i), now, student\_data[6]))

conn.commit()

else:

cursor.execute("insert into attendances (student\_id, class\_id, created\_at, is\_unknown) values (%s, %s, %s, %s);", (int(name.split(".")[0]), int(i), now, student\_data[6]))

conn.commit()

self.face\_names.append(f'{name} ({confidence})')

process\_current\_frame = not process\_current\_frame

for (top, right, bottom, left), name in zip(self.face\_locations, self.face\_names):

top \*= 4

right \*= 4

bottom \*= 4

left \*= 4

if name.split(" ")[0] == 'unknown':

face\_image = frame[top:bottom + 35, left:right + 35]

student\_id = random.randint(10000, 99999)

image\_name = f'faces/%d.png' % student\_id

cursor.execute("insert into attendances (student\_id, class\_id, created\_at,is\_unknown) values (%s, %s, %s, %s);", (int(student\_id), int(i), datetime.now(), 'true'))

conn.commit()

selectQuery = "select \* from attendances where student\_id = %s ORDER BY id DESC LIMIT 1;" % int(student\_id)

cursor.execute(selectQuery)

rows = cursor.fetchall()

if len(rows) == 1:

cv2.imwrite(image\_name, face\_image)

face\_image = face\_recognition.load\_image\_file(f'{image\_name}')

face\_encoding = face\_recognition.face\_encodings(face\_image)[0]

self.known\_face\_encodings.append(face\_encoding)

self.known\_face\_names.append(image\_name.split("/")[1])

select\_class\_number = "select \* from class\_number where class\_id = %s;" % int(i)

cursor.execute(select\_class\_number)

row = cursor.fetchall()

if len(row) == 1:

mail\_send(int(student\_id), image\_name, row[0][2])

mail\_send(int(student\_id), image\_name, int(i))

cv2.rectangle(frame, (left, top), (right, bottom), (0, 0, 255), 2)

cv2.rectangle(frame, (left, bottom - 35), (right, bottom), (0, 0, 255), -1)

cv2.putText(frame, name, (left + 6, bottom - 6), cv2.FONT\_HERSHEY\_DUPLEX, 0.8, (255, 255, 255), 1)

cv2.imshow(f'Face Recognition{i}', frame)

if cv2.waitKey(1) & 0xFF == ord('q'):

break

for video\_capture in capture\_objects:

video\_capture.release()

cv2.destroyAllWindows()

cursor.close()

conn.close()

if \_\_name\_\_== '\_\_main\_\_':

fr = FaceRecognition()

fr.run\_recognition()

**Back-end**

**index.js**

const express= require('express');

const bodyParser = require('body-parser');

const authRoutes = require('./routes/auth');

const errorController = require('./controllers/error');

const cors = require('cors');

const path = require('path');

const app = express();

const desktopPath = path.join(require('os').homedir(), 'Desktop', 'Finall project');

app.use(desktopPath, express.static('uploads'));

//app.use('/images', express.static(path.join(\_\_dirname, '../../FaceRecognitionAttendanceSystem/faces')));

const ports = process.env.PORT || 3000;

app.use(bodyParser.json({ limit: '10mb' }));

app.use(bodyParser.urlencoded({ extended: true, limit: '10mb' }));

app.use(

bodyParser.urlencoded({

extended: true,

})

)

app.use(cors());

app.use((req, res, next) =>{

res.setHeader('Access-Control-Allow-Origin', '\*');

res.setHeader('Access-Control-Allow-Methods', 'GET, POST, PUT, DELETE');

res.setHeader('Access-Control-Allow-Headers', 'Content-Type, Authorization');

next();

});

app.use('/auth', authRoutes);

app.use(errorController.get404);

app.use(errorController.get500);

app.listen(ports, () => console.log(`Listening on port ${ports}`))

**util/database.js**

const {Pool} = require('pg');

const config = require('../config/config.json');

const pool = new Pool({

host: config.host,

user: config.user,

database: config.database,

password: config.password,

port: config.port

})

module.exports = pool;

**Routers/auth.js**

const express = require('express');

const {body} = require('express-validator');

const router = express.Router();

const authController = require('../controllers/auth');

const auth = require('../middleware/auth');

const multer = require('multer');

const path = require('path');

const desktopPath = path.join(require('os').homedir(), 'Desktop', 'Finall project/face-recognition-python/FaceRecognitionAttendanceSystem/faces');

const storage = multer.diskStorage({

destination: function (req, file, cb) {

cb(null, desktopPath);

},

filename: function (req, file, cb) {

const min = 10000, max = 99999;

cb(null, Math.floor(Math.random() \* (max - min + 1)) + min + '.'+file.originalname.split(".")[1]); // Rename the file

},

});

const upload = multer({ storage: storage });

router.post(

'/user-form',

auth,

[

body('name').trim().not().isEmpty(),

body('user\_type').trim().not().isEmpty(),

body('email').isEmail().withMessage('Please enter a valid email.')

.normalizeEmail(),

], authController.userForm

);

router.post('/image', upload.single('file'), (req, res) => {

if (!req.file) {

return res.status(400).send('No image file received');

}

const imageUrl = req.file.filename.split(".")[0];

res.status(200).json({ imageUrl, status:200 });

});

router.post('/login', authController.login);

router.get('/staff',auth, authController.getStaff);

router.get('/students', auth, authController.getStudent);

router.get('/attendances',auth, authController.getAttendances);

router.get('/user/:id',auth, authController.getUser);

router.get('/rooms',auth, authController.getRooms);

router.post('/class', auth, authController.addClass);

router.get('/staff-attandances',auth, authController.getStaffAttendances);

router.post('/change-password', authController.changePassword);

module.exports = router;

models/user.js

const db = require('../util/database');

module.exports = class User {

constructor(user\_id,name,email,password,user\_type){

this.user\_id = user\_id;

this.name = name;

this.email = email;

this.password = password;

this.user\_type = user\_type;

}

static async find(email){

return db.query(`SELECT \* FROM users WHERE email = $1;`, [email])

}

static save(user){

return db.query(

`INSERT INTO users (name, email, password, user\_type, user\_id) VALUES ($1, $2, $3, $4, $5);`,[user.name, user.email, user.password, user.user\_type, parseInt(user.user\_id)]

)

}

static getStaff(){

return db.query(

`SELECT \* FROM users where user\_type = 'staff'`

)

}

static getStudentAndVisitor(){

return db.query(

`SELECT \* FROM users where user\_type = 'student' OR user\_type= 'visitor'`

)

}

static getAttendances(){

return db.query(

`SELECT \* FROM attendances LEFT JOIN users ON attendances.student\_id = users.user\_id LEFT JOIN class\_number ON attendances.class\_id = class\_number.class\_id where users.user\_type != 'staff' OR attendances.is\_unknown = true`

)

}

static getStaffAttendances(){

return db.query(

`SELECT \* FROM attendances LEFT JOIN users ON attendances.student\_id = users.user\_id LEFT JOIN class\_number ON attendances.class\_id = class\_number.class\_id where users.user\_type = 'staff'`

)

}

static getUser(userId){

return db.query(`SELECT \* FROM users WHERE user\_id = $1;`, [userId])

}

static updatePassword(user){

console.log(user);

return db.query(

`UPDATE users SET password = $1 where user\_id = $2 `, [user.password, parseInt(user.user\_id)]

)

}

};

models/room.js

const db = require('../util/database');

module.exports = class Rooms {

constructor(class\_id, class\_number){

this.class\_id = class\_id;

this.class\_number = class\_number;

}

static getRooms(){

return db.query(

`SELECT \* FROM class\_number`

)

}

static getRoomById(id){

return db.query(

`SELECT \* FROM class\_number where class\_id = $1`, [id]

)

}

static addClass(id,class\_number){

return db.query(

`INSERT INTO class\_number (class\_id, class\_number) VALUES ($1, $2);`,[ id, class\_number]

)

}

};

middleware/ auth.js

const jwt = require('jsonwebtoken');

module.exports = (req, res, next) => {

const authHeader = req.get('Authorization');

if (!authHeader) {

const error = new Error('Not authenticated.');

error.statusCode = 401;

throw error;

}

const token = authHeader.split(' ')[1];

let decodedToken;

try {

// jwt.verify both decodes and verifies the token and is validated with the secret

decodedToken = jwt.verify(token, 'secretfortoken');

} catch (err) {

err.statusCode = 500;

throw err;

}

// check if token unverified i.e. undefined

if (!decodedToken) {

const error = new Error('Not authenticated.');

error.statusCode = 401;

throw error;

}

req.isLoggedIn = true;

req.userId = decodedToken.userId;

req.email = decodedToken.email;

next();

};

Controllers/auth.js

const { validationResult} = require('express-validator');

const bcrypt = require('bcryptjs');

const jwt = require('jsonwebtoken');

const User = require('../models/user');

const Rooms = require('../models/rooms');

exports.userForm = async (req, res, next) => {

const errors = validationResult(req);

if(!errors.isEmpty()) return

const {name,email,password,user\_type,user\_id} = req.body;

let hashedPassword ;

try{

const user = await User.find(email);

if(user.rows.length > 0){

return res.status(202).json({message: 'Email address already exist!'})

}

if(password){

hashedPassword = await bcrypt.hash(password,12)

}else

{

hashedPassword = '';

}

const userDetails = {

name,

email,

password: hashedPassword,

user\_type,

user\_id: user\_id

}

const result = await User.save(userDetails);

const userData = await User.getUser(userDetails.user\_id);

res.status(201).json({message: 'User registered!', data: userData})

} catch(err){

if(!err.statusCode){

err.statusCode = 500;

}

next(err);

}

}

exports.login = async (req, res, next) => {

const email = req.body.email;

const password = req.body.password;

try {

const user = await User.find(email);

if (user.rows.length !== 1) {

const error = new Error('A user with this email could not be found.');

error.statusCode = 401; // 401: not authenticated

throw error;

}

const storedUser = user.rows[0];

if (storedUser.user\_type != 'admin') {

const error = new Error('You are not addmin');

error.statusCode = 401; // 401: not authenticated

throw error;

}

const isEqual = await bcrypt.compare(password, storedUser.password);

if (!isEqual) {

const error = new Error('Wrong password!');

error.statusCode = 401;

throw error;

}

const token = jwt.sign(

{

email: storedUser.email,

userId: storedUser.user\_id,

},

'secretfortoken',

{ expiresIn: '1h' }

);

res.status(200).json({ token: token, userId: storedUser.user\_id });

} catch (err) {

if (!err.statusCode) {

err.statusCode = 500;

}

next(err);

}

};

exports.changePassword = async (req, res, next) => {

try {

const {password,new\_password,user\_id} = req.body;

const user = await User.getUser(user\_id);

const storedUser = user.rows[0];

if (user.rows.length == 1) {

let hashedPassword ;

if (storedUser.user\_type != 'admin') {

const error = new Error('You are not addmin');

error.statusCode = 401; // 401: not authenticated

throw error;

}

const isEqual = await bcrypt.compare(password, storedUser.password);

if (!isEqual) {

const error = new Error('Wrong password!');

error.statusCode = 401;

throw error;

}

hashedPassword = await bcrypt.hash(new\_password,12)

console.log(hashedPassword);

const userDetails = {

password: hashedPassword,

user\_id: user\_id

}

if(hashedPassword){

let data = await User.updatePassword(userDetails);

res.status(201).json({ msg: 'Password has been upated!!', data: data});

}

}else{

res.status(404).json({ msg: 'You are not Admin'});

}

} catch (err) {

if (!err.statusCode) {

err.statusCode = 500;

}

next(err);

}

};

exports.getStaff = async (req, res, next) => {

try {

const staff = await User.getStaff();

res.status(200).json({ data: staff.rows });

} catch (err) {

if (!err.statusCode) {

err.statusCode = 500;

}

next(err);

}

};

exports.getStudent = async (req, res, next) => {

try {

const getStudent = await User.getStudentAndVisitor();

res.status(200).json({ data: getStudent.rows });

} catch (err) {

if (!err.statusCode) {

err.statusCode = 500;

}

next(err);

}

};

exports.getAttendances = async (req, res, next) => {

try {

const getAttendance = await User.getAttendances();

res.status(200).json({ data: getAttendance.rows });

} catch (err) {

if (!err.statusCode) {

err.statusCode = 500;

}

next(err);

}

};

exports.getStaffAttendances = async (req, res, next) => {

try {

const getAttendance = await User.getStaffAttendances();

res.status(200).json({ data: getAttendance.rows });

} catch (err) {

if (!err.statusCode) {

err.statusCode = 500;

}

next(err);

}

};

exports.getUser = async (req, res, next) => {

try {

const getUser= await User.getUser(req.params.id);

res.status(200).json({ data: getUser.rows });

} catch (err) {

if (!err.statusCode) {

err.statusCode = 500;

}

next(err);

}

};

exports.getRooms = async (req, res, next) => {

try {

const getRooms= await Rooms.getRooms();

res.status(200).json({ data: getRooms.rows });

} catch (err) {

if (!err.statusCode) {

err.statusCode = 500;

}

next(err);

}

};

exports.addClass = async (req, res, next) => {

try {

const {class\_number,class\_id} = req.body;

const getRooms= await Rooms.getRoomById(class\_id);

if(getRooms.rows.length == 0){

const addClass= await Rooms.addClass(class\_id, class\_number);

res.status(200).json({ data: addClass.rows });

}else{

res.status(201).json({ message: 'This camera is already assign to class' });

}

} catch (err) {

if (!err.statusCode) {

err.statusCode = 500;

}

next(err);

}

};

Controllers/error.js

exports.get404 = (req, res, next) => {

const error = new Error('Not found.');

error.statusCode = 404;

next(error);

}

exports.get500 = (error, req, res, next) => {

const data = error.data;

res.status(error.statusCode || 500);

res.json({

error:{

message: error.message,

data: data,

},

});

};

Config/config.json

{

"host": "127.0.0.1",

"user": "postgres",

"database": "FaceRecognitionAttendanceSystem",

"password": "a",

"port": 5432

}

Front-end

**Main.ts**

import { enableProdMode } from '@angular/core';

import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';

import { AppModule } from './app/app.module';

import { environment } from './environments/environment';

import 'hammerjs';

if (environment.production) {

enableProdMode();

}

platformBrowserDynamic().bootstrapModule(AppModule);

**index.html**

<!doctype html>

<html class="light-style customizer-hide layout-menu-fixed">

<head>

<base href='/'>

<meta charset="utf-8" />

<link rel="apple-touch-icon" sizes="76x76" href="./assets/img/apple-icon.png" />

<link rel="icon" type="image/png" href="./assets/img/favicon.png" />

<meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1" />

<title>Face Recognization System</title>

<meta content='width=device-width, initial-scale=1.0, maximum-scale=1.0, user-scalable=0' name='viewport' />

<meta name="viewport" content="width=device-width" />

<!-- Fonts and icons -->

<link rel="preconnect" href="https://fonts.googleapis.com" />

<link rel="preconnect" href="https://fonts.gstatic.com" crossorigin />

<link

href="https://fonts.googleapis.com/css2?family=Public+Sans:ital,wght@0,300;0,400;0,500;0,600;0,700;1,300;1,400;1,500;1,600;1,700&display=swap"

rel="stylesheet" />

<link rel="stylesheet" href="./assets/fonts/boxicons.css" />

<link href="https://maxcdn.bootstrapcdn.com/font-awesome/latest/css/font-awesome.min.css" rel="stylesheet">

<link href='https://fonts.googleapis.com/css?family=Roboto:400,700,300|Material+Icons' rel='stylesheet'

type='text/css'>

<script type="text/javascript" src="https://maps.googleapis.com/maps/api/js?key=YOUR-KEY-HERE"></script>

<script>window.global = window;</script>

</head>

<body>

<app-root>

<div class="loader">

<svg class="circular" viewBox="25 25 50 50">

<circle class="path" cx="50" cy="50" r="20" fill="none" stroke-width="2" stroke-miterlimit="10" />

</svg>

</div>

</app-root>

</body>

</html>

**Environments/ environment.ts**

export const environment = {

production: false,

webApiUrl: 'http://localhost:3000/auth/',

};

**App/ app.routing.ts**

import { NgModule } from '@angular/core';

import { CommonModule, } from '@angular/common';

import { BrowserModule } from '@angular/platform-browser';

import { Routes, RouterModule } from '@angular/router';

import { AdminLayoutComponent } from './layouts/admin-layout/admin-layout.component';

import { LoginLayoutComponent } from './layouts/login-layout/login-layout.component';

const routes: Routes = [

{

path: '',

redirectTo: 'login',

pathMatch: 'full',

}, {

path: 'admin',

component: AdminLayoutComponent,

children: [{

path: '',

loadChildren: () => import('./layouts/admin-layout/admin-layout.module').then(m => m.AdminLayoutModule)

}]

}, {

path: '',

component: LoginLayoutComponent,

children: [{

path: '',

loadChildren: () => import('./layouts/login-layout/login-layout.module').then(m => m.LoginLayoutModule)

}]

}

];

@NgModule({

imports: [

CommonModule,

BrowserModule,

RouterModule.forRoot(routes, {

useHash: false

})

],

exports: [

],

})

export class AppRoutingModule { }

**app/ app.module.ts**

import { BrowserAnimationsModule } from '@angular/platform-browser/animations';

import { NgModule } from '@angular/core';

import { FormsModule, ReactiveFormsModule } from '@angular/forms';

import { HttpClientModule, HTTP\_INTERCEPTORS } from "@angular/common/http";

import { RouterModule } from '@angular/router';

import { AppRoutingModule } from './app.routing';

import { ComponentsModule } from './components/components.module';

import { AppComponent } from './app.component';

import { AdminLayoutComponent } from './layouts/admin-layout/admin-layout.component';

import { LoginLayoutComponent } from './layouts/login-layout/login-layout.component';

import { NotifierModule } from 'angular-notifier';

import { NgxSpinnerModule } from 'ngx-spinner';

@NgModule({

imports: [

BrowserAnimationsModule,

FormsModule,

ReactiveFormsModule,

HttpClientModule,

ComponentsModule,

RouterModule,

AppRoutingModule,

NgxSpinnerModule,

NotifierModule.withConfig({

// Custom options in here

position: {

horizontal: {

position: "middle",

//distance: 50,

},

vertical: {

position: "bottom",

distance: 50,

gap: 10,

},

},

//behaviour: {

// autoHide: false

//}

}),

],

declarations: [

AppComponent,

AdminLayoutComponent,

LoginLayoutComponent

],

providers: [

],

bootstrap: [AppComponent]

})

export class AppModule { }

**app/ app.component.html**

<router-outlet></router-outlet>

**footer.component.html**

<footer class="footer ">

<div class="container-fluid">

<nav class="pull-left">

<ul>

<li>

<a href="https://www.creative-tim.com">

Creative Tim

</a>

</li>

<li>

<a href="https://creative-tim.com/about-us">

About Us

</a>

</li>

<li>

<a href="http://blog.creative-tim.com">

Blog

</a>

</li>

<li>

<a href="https://www.creative-tim.com/license">

Licenses

</a>

</li>

</ul>

</nav>

<div class="copyright pull-right">

&copy;

{{test | date: 'yyyy'}}, made with love by

<a href="https://www.creative-tim.com" target="\_blank">Creative Tim</a> for a better web.

</div>

</div>

</footer>

**navbar.component.html**

<!-- Navbar -->

<nav class="layout-navbar container-fluid navbar navbar-expand-xl navbar-detached align-items-center bg-navbar-theme"

id="layout-navbar">

<div class="layout-menu-toggle navbar-nav align-items-xl-center me-3 me-xl-0 d-xl-none">

<a class="nav-item nav-link px-0 me-xl-4" (click)="sidebarToggle()">

<i class="bx bx-menu bx-sm"></i>

</a>

</div>

<div class="navbar-nav-right d-flex align-items-center" id="navbar-collapse">

<!-- Search -->

<ol class="breadcrumb mb-0">

<li class="breadcrumb-item fs-6" \*ngFor="let pageData of getTitle()?.pastPage">

<a routerLink="{{pageData.pastUrl}}">{{ pageData.pastLinkName }}</a>

</li>

<li class="breadcrumb-item fs-6 active">{{getTitle().currentPageName}}</li>

</ol>

<!-- /Search -->

<ul class="navbar-nav flex-row align-items-center ms-auto">

<!-- User -->

<li class="nav-item navbar-dropdown dropdown-user dropdown">

<a class="nav-link dropdown-toggle hide-arrow" href="javascript:void(0);" data-bs-toggle="dropdown">

<div class="avatar avatar-online">

<img src="../../../assets/img/user.png" alt class="w-px-40 h-auto rounded-circle" />

</div>

</a>

<ul class="dropdown-menu dropdown-menu-end">

<li>

<a class="dropdown-item" (click)="logout()">

<i class="bx bx-power-off me-2"></i>

<span class="align-middle">Log Out</span>

</a>

</li>

</ul>

</li>

<!--/ User -->

</ul>

</div>

</nav>

<!-- / Navbar -->

**navbar.component.ts**

import { Component, OnInit, ElementRef } from '@angular/core';

import { ROUTES } from '../sidebar/sidebar.component';

import { Location, LocationStrategy, PathLocationStrategy } from '@angular/common';

import { Router } from '@angular/router';

import { StorageService } from 'app/shared/storage.service';

@Component({

selector: 'app-navbar',

templateUrl: './navbar.component.html',

styleUrls: ['./navbar.component.css']

})

export class NavbarComponent implements OnInit {

private listTitles: any[];

location: Location;

mobile\_menu\_visible: any = 0;

private sidebarVisible: boolean;

constructor(location: Location, private storageService: StorageService, private element: ElementRef, private router: Router) {

this.location = location;

this.sidebarVisible = false;

}

ngOnInit() {

this.listTitles = ROUTES.filter(listTitle => listTitle);

const navbar: HTMLElement = this.element.nativeElement;

this.router.events.subscribe((event) => {

this.sidebarClose();

});

}

sidebarOpen() {

const body = document.getElementsByTagName('body')[0];

body.classList.add('layout-menu-expanded');

this.sidebarVisible = true;

};

sidebarClose() {

const body = document.getElementsByTagName('body')[0];

body.classList.remove('layout-menu-expanded');

};

sidebarToggle() {

var $toggle = document.getElementsByClassName('layout-menu-toggle')[0];

this.sidebarOpen();

};

logout() {

this.storageService.removeValue('token');

this.storageService.removeValue('userId');

this.storageService.removeValue('isLogin');

this.router.navigate(['login']);

}

getTitle() {

var titlee = this.location.prepareExternalUrl(this.location.path());

if (titlee.charAt(0) === '#') {

titlee = titlee.slice(1);

}

if (titlee.charAt(0) === '?') {

titlee = titlee.slice(1);

}

if (titlee.includes('add')) {

return {

pastPage: [],

currentPageName: ''

};

}

else if (titlee.includes('users')) {

return {

pastPage: [],

currentPageName: 'Users'

};

}

else if (titlee.includes('attandance')) {

return {

pastPage: [],

currentPageName: 'Attandances'

};

}

else if (titlee.includes('staff-attandance')) {

return {

pastPage: [],

currentPageName: 'Staff Attandances'

};

}

else if (titlee.includes('staff')) {

return {

pastPage: [],

currentPageName: 'Staffs'

};

}

else if (titlee.includes('room')) {

return {

pastPage: [],

currentPageName: 'Room Assign'

};

}

else if (titlee.includes('change-password')) {

return {

pastPage: [],

currentPageName: 'Change Password'

};

}

else {

return {

pastPage: [],

currentPageName: ''

};

}

}

}

**sidebar.component.html**

<aside id="layout-menu" class="layout-menu menu-vertical menu bg-menu-theme">

<div class="app-brand demo">

<a href="javascript:void(0);" class="app-brand-link">

<span class="max-width app-brand-text fw-bolder text-capitalize">Face Recognization Attandance System</span>

</a>

<a href="javascript:void(0);" (click)="sidebarClose()"

class="layout-menu-toggle menu-link text-large ms-auto d-block d-xl-none">

<i class="bx bx-chevron-left bx-sm align-middle"></i>

</a>

</div>

<div class="menu-inner-shadow"></div>

<ul class="menu-inner py-1 sidebar-wrapper">

<!-- Users -->

<li class="menu-item" routerLinkActive="active">

<a routerLink="users" class="menu-link">

<i class='menu-icon tf-icons bx bxs-group'></i>

<div data-i18n="Analytics">Users</div>

</a>

</li>

<!-- Staff -->

<li class="menu-item" routerLinkActive="active">

<a routerLink="staff" class="menu-link">

<i class='menu-icon tf-icons bx bxs-group'></i>

<div data-i18n="Analytics">Staff</div>

</a>

</li>

<!-- Attandance -->

<li class="menu-item" routerLinkActive="active">

<a routerLink="attandance" class="menu-link">

<i class='menu-icon tf-icons bx bxs-spreadsheet'></i>

<div data-i18n="Analytics">Attandance</div>

</a>

</li>

<!-- Staff Attandance -->

<li class="menu-item" routerLinkActive="active">

<a routerLink="staff-attandance" class="menu-link">

<i class='menu-icon tf-icons bx bxs-spreadsheet'></i>

<div data-i18n="Analytics">Staff Attandance</div>

</a>

</li>

<!-- Room Assign -->

<li class="menu-item" routerLinkActive="active">

<a routerLink="room" class="menu-link">

<i class='menu-icon tf-icons bx bxs-spreadsheet'></i>

<div data-i18n="Analytics">Room Assign</div>

</a>

</li>

<!-- Change Password -->

<li class="menu-item" routerLinkActive="active">

<a routerLink="change-password" class="menu-link">

<i class='menu-icon tf-icons bx bxs-spreadsheet'></i>

<div data-i18n="Analytics">Change Password</div>

</a>

</li>

</ul>

</aside>

**sidebar.component.ts**

import { Component, OnInit } from '@angular/core';

declare const $: any;

declare interface RouteInfo {

path: string;

title: string;

icon: string;

class: string;

}

export const ROUTES: RouteInfo[] = [

{ path: '/admin/dashboard', title: 'Dashboard', icon: 'dashboard', class: '' },

{ path: '/admin/user-profile', title: 'User Profile', icon:'person', class: '' },

{ path: '/admin/table-list', title: 'Table List', icon:'content\_paste', class: '' },

{ path: '/admin/typography', title: 'Typography', icon:'library\_books', class: '' },

{ path: '/admin/icons', title: 'Icons', icon:'bubble\_chart', class: '' },

{ path: '/admin/maps', title: 'Maps', icon:'location\_on', class: '' },

{ path: '/admin/notifications', title: 'Notifications', icon:'notifications', class: '' },

{ path: '/admin/upgrade', title: 'Upgrade to PRO', icon:'unarchive', class: 'active-pro' },

];

@Component({

selector: 'app-sidebar',

templateUrl: './sidebar.component.html',

styleUrls: ['./sidebar.component.css']

})

export class SidebarComponent implements OnInit {

menuItems: any[];

constructor() { }

ngOnInit() {

this.menuItems = ROUTES.filter(menuItem => menuItem);

}

isMobileMenu() {

if ($(window).width() > 991) {

return false;

}

return true;

};

sidebarClose() {

//const navbar: HTMLElement = this.element.nativeElement;

//this.toggleButton = this.navbar.getElementsByClassName("layout-menu-toggle")[0];

const body = document.getElementsByTagName("body")[0];

//this.toggleButton.classList.remove("toggled");

//this.sidebarVisible = false;

body.classList.remove("layout-menu-expanded");

}

}

**components.module.ts**

import { NgModule } from '@angular/core';

import { CommonModule } from '@angular/common';

import { RouterModule } from '@angular/router';

import { FooterComponent } from './footer/footer.component';

import { NavbarComponent } from './navbar/navbar.component';

import { SidebarComponent } from './sidebar/sidebar.component';

import { HeaderComponent } from './header/header.component';

@NgModule({

imports: [

CommonModule,

RouterModule,

],

declarations: [

FooterComponent,

NavbarComponent,

HeaderComponent,

SidebarComponent

],

exports: [

FooterComponent,

NavbarComponent,

SidebarComponent,

HeaderComponent

]

})

export class ComponentsModule { }

**login-layout.component.ts**

import { Component, OnInit, ViewChild, AfterViewInit } from '@angular/core';

import { Location, LocationStrategy, PathLocationStrategy, PopStateEvent } from '@angular/common';

import { Router, NavigationEnd, NavigationStart } from '@angular/router';

import PerfectScrollbar from 'perfect-scrollbar';

import \* as $ from "jquery";

import { filter, Subscription } from 'rxjs';

@Component({

selector: 'app-admin-layout',

templateUrl: './login-layout.component.html',

styleUrls: ['./login-layout.component.scss']

})

export class LoginLayoutComponent implements OnInit {

private \_router: Subscription;

private lastPoppedUrl: string;

private yScrollStack: number[] = [];

constructor( public location: Location, private router: Router) {}

ngOnInit() {

const isWindows = navigator.platform.indexOf('Win') > -1 ? true : false;

if (isWindows && !document.getElementsByTagName('body')[0].classList.contains('sidebar-mini')) {

// if we are on windows OS we activate the perfectScrollbar function

document.getElementsByTagName('body')[0].classList.add('perfect-scrollbar-on');

} else {

document.getElementsByTagName('body')[0].classList.remove('perfect-scrollbar-off');

}

const elemMainPanel = <HTMLElement>document.querySelector('.main-panel');

const elemSidebar = <HTMLElement>document.querySelector('.sidebar .sidebar-wrapper');

this.location.subscribe((ev:PopStateEvent) => {

this.lastPoppedUrl = ev.url;

});

this.router.events.subscribe((event:any) => {

if (event instanceof NavigationStart) {

if (event.url != this.lastPoppedUrl)

this.yScrollStack.push(window.scrollY);

} else if (event instanceof NavigationEnd) {

if (event.url == this.lastPoppedUrl) {

this.lastPoppedUrl = undefined;

window.scrollTo(0, this.yScrollStack.pop());

} else

window.scrollTo(0, 0);

}

});

this.\_router = this.router.events.pipe(filter(event => event instanceof NavigationEnd)).subscribe((event: NavigationEnd) => {

elemMainPanel.scrollTop = 0;

elemSidebar.scrollTop = 0;

});

if (window.matchMedia(`(min-width: 960px)`).matches && !this.isMac()) {

let ps = new PerfectScrollbar(elemMainPanel);

ps = new PerfectScrollbar(elemSidebar);

}

const window\_width = $(window).width();

let $sidebar = $('.sidebar');

let $sidebar\_responsive = $('body > .navbar-collapse');

let $sidebar\_img\_container = $sidebar.find('.sidebar-background');

if(window\_width > 767){

if($('.fixed-plugin .dropdown').hasClass('show-dropdown')){

$('.fixed-plugin .dropdown').addClass('open');

}

}

$('.fixed-plugin a').click(function(event){

// Alex if we click on switch, stop propagation of the event, so the dropdown will not be hide, otherwise we set the section active

if($(this).hasClass('switch-trigger')){

if(event.stopPropagation){

event.stopPropagation();

}

else if(window.event){

window.event.cancelBubble = true;

}

}

});

$('.fixed-plugin .badge').click(function(){

let $full\_page\_background = $('.full-page-background');

$(this).siblings().removeClass('active');

$(this).addClass('active');

var new\_color = $(this).data('color');

if($sidebar.length !== 0){

$sidebar.attr('data-color', new\_color);

}

if($sidebar\_responsive.length != 0){

$sidebar\_responsive.attr('data-color',new\_color);

}

});

$('.fixed-plugin .img-holder').click(function(){

let $full\_page\_background = $('.full-page-background');

$(this).parent('li').siblings().removeClass('active');

$(this).parent('li').addClass('active');

var new\_image = $(this).find("img").attr('src');

if($sidebar\_img\_container.length !=0 ){

$sidebar\_img\_container.fadeOut('fast', function(){

$sidebar\_img\_container.css('background-image','url("' + new\_image + '")');

$sidebar\_img\_container.fadeIn('fast');

});

}

if($full\_page\_background.length != 0){

$full\_page\_background.fadeOut('fast', function(){

$full\_page\_background.css('background-image','url("' + new\_image + '")');

$full\_page\_background.fadeIn('fast');

});

}

if($sidebar\_responsive.length != 0){

$sidebar\_responsive.css('background-image','url("' + new\_image + '")');

}

});

}

ngAfterViewInit() {

this.runOnRouteChange();

}

isMaps(path){

var titlee = this.location.prepareExternalUrl(this.location.path());

titlee = titlee.slice( 1 );

if(path == titlee){

return false;

}

else {

return true;

}

}

runOnRouteChange(): void {

if (window.matchMedia(`(min-width: 960px)`).matches && !this.isMac()) {

const elemMainPanel = <HTMLElement>document.querySelector('.main-panel');

const ps = new PerfectScrollbar(elemMainPanel);

ps.update();

}

}

isMac(): boolean {

let bool = false;

if (navigator.platform.toUpperCase().indexOf('MAC') >= 0 || navigator.platform.toUpperCase().indexOf('IPAD') >= 0) {

bool = true;

}

return bool;

}

}

**login-layout.module.ts**

import { NgModule } from '@angular/core';

import { RouterModule } from '@angular/router';

import { CommonModule } from '@angular/common';

import { FormsModule, ReactiveFormsModule } from '@angular/forms';

import { LoginLayoutRoutes } from './login-layout.routing';

import { MatButtonModule } from '@angular/material/button';

import { MatInputModule } from '@angular/material/input';

import { MatRippleModule } from '@angular/material/core';

import { MatFormFieldModule } from '@angular/material/form-field';

import { MatTooltipModule } from '@angular/material/tooltip';

import { MatSelectModule } from '@angular/material/select';

import { AdminLoginComponent } from 'app/pages/admin-login/admin-login.component';

@NgModule({

imports: [

CommonModule,

RouterModule.forChild(LoginLayoutRoutes),

FormsModule,

ReactiveFormsModule,

MatButtonModule,

MatRippleModule,

MatFormFieldModule,

MatInputModule,

MatSelectModule,

MatTooltipModule,

],

declarations: [

AdminLoginComponent

]

})

export class LoginLayoutModule { }

**login-layout.routing.ts**

import { Routes } from '@angular/router';

import { AdminLoginComponent } from 'app/pages/admin-login/admin-login.component';

export const LoginLayoutRoutes: Routes = [

{ path: 'login', component: AdminLoginComponent },

];

**login-layout.service.ts**

import { Injectable } from '@angular/core';

import { HttpClient, HttpHeaders } from '@angular/common/http';

import { CommonService } from 'app/shared/common.service';

@Injectable({

providedIn: 'root'

})

export class LoginLayoutService {

constructor(public http: HttpClient, public commonService: CommonService) { }

login(data: any) {

return this.http.post(this.commonService.rootData.rootUrl + 'login', data)

}

}

**users.component.html**

<div class="container-fluid flex-grow-1 container-p-y">

<div class="card">

<div class="card-datatable table-responsive">

<div class="dataTables\_wrapper dt-bootstrap5 no-footer">

<div class="card-header flex-column flex-md-row">

<div class="head-label text-center">

<h5 class="card-title mb-0">&nbsp;</h5>

</div>

<div class="dt-action-buttons text-end pt-3 pt-md-0">

<div class="dt-buttons">

<button class="dt-button create-new btn btn-primary" (click)="addNew()" type="button">

<span><i class="bx bx-plus me-sm-2"></i>

<span class="d-none d-sm-inline-block">Add New Users</span>

</span>

</button>

</div>

</div>

</div>

<div class="row">

<div class="col-sm-12 col-md-6">

<div class="dataTables\_length" id="DataTables\_Table\_0\_length">

<label>Show

<select name="DataTables\_Table\_0\_length" [(ngModel)]="itemsPerPage"

(change)="itemsPerPageChange($event.target.value)" class="form-select">

<option value="5">5</option>

<option value="10">10</option>

<option value="25">25</option>

<option value="50">50</option>

<option value="75">75</option>

<option value="100">100</option>

</select> entries

</label>

</div>

</div>

<div class="col-sm-12 col-md-6 d-flex justify-content-center justify-content-md-end">

<div id="DataTables\_Table\_0\_filter" class="dataTables\_filter">

<label>Search:

<input type="search" (input)="search()" [(ngModel)]="searchTerm" class="form-control"

placeholder="Search here ..." aria-controls="DataTables\_Table\_0">

</label>

</div>

</div>

</div>

<table class="table dataTable ">

<thead>

<tr>

<th>No.</th>

<th>User Id</th>

<th>Name</th>

<th>Email</th>

<th>User Type</th>

</tr>

</thead>

<tbody class="">

<tr

\*ngFor="let item of userList | paginate: { itemsPerPage: l, currentPage: p };let collectionIndex = index">

<td>{{ l \* (p - 1) + collectionIndex +1 }}</td>

<td>{{item.user\_id}}</td>

<td>{{item.name}}</td>

<td>{{item.email}}</td>

<td>{{item.user\_type}}</td>

</tr>

<tr \*ngIf="noData">

<td colspan="5">

<h4 class="text-center mt-3">No Data Found</h4>

</td>

</tr>

</tbody>

</table>

<div class="row">

<div class="col-sm-12 col-md-6">

<div class="dataTables\_info">Showing {{userList.length > 0 ? l \* (p - 1) + 1 : 0}} to {{l \*

(p - 1) + l < userList.length ? l \* (p - 1) + l : userList.length}} of {{userList.length}}

entries</div>

</div>

<div class="col-sm-12 col-md-6">

<div class="dataTables\_paginate paging\_simple\_numbers">

<pagination-template #pagination="paginationApi" (pageChange)="p = $event">

<ul class="pagination">

<li class="page-item prev" [class.disabled]="pagination.isFirstPage()">

<a \*ngIf="!pagination.isFirstPage()" class="page-link cursor-pointer"

(click)="pagination.previous()">

<i class="tf-icon bx bx-chevron-left"></i> </a>

<div \*ngIf="pagination.isFirstPage()" class="page-link">

<i class="tf-icon bx bx-chevron-left"></i>

</div>

</li>

<li \*ngFor="let page of pagination.pages" class="page-item"

[class.active]="pagination.getCurrent() === page.value">

<a (click)="pagination.setCurrent(page.value)"

class="page-link cursor-pointer"

\*ngIf="pagination.getCurrent() !== page.value">

<span>{{ page.label }}</span>

</a>

<div class="page-link" \*ngIf="pagination.getCurrent() === page.value">

<span>{{ page.label }}</span>

</div>

</li>

<li class="page-item next" [class.disabled]="pagination.isLastPage()">

<a \*ngIf="!pagination.isLastPage()" class="page-link cursor-pointer"

(click)="pagination.next()"><i class="tf-icon bx bx-chevron-right"></i>

</a>

<div \*ngIf="pagination.isLastPage()" class="page-link"><i

class="tf-icon bx bx-chevron-right"></i> </div>

</li>

</ul>

</pagination-template>

</div>

</div>

</div>

</div>

</div>

</div>

</div>

**users.component.ts**

import { Component, OnInit } from '@angular/core';

import { Router } from '@angular/router';

import { AdminLayoutService } from 'app/layouts/admin-layout/admin-layout.service';

@Component({

selector: 'app-users',

templateUrl: './users.component.html',

styleUrls: ['./users.component.css']

})

export class UsersComponent implements OnInit {

l: number;

p: number = 1;

itemsPage: any;

userList: any[] = [];

allUserList: any[] = [];

itemsPerPage: number;

noData: boolean = false;

searchTerm = '';

constructor(public router: Router, public adminLayoutService: AdminLayoutService) { }

ngOnInit() {

this.l = this.itemsPerPage = 10;

this.getUserList();

}

addNew() {

this.router.navigate([`admin/add/users`])

}

getUserList() {

this.adminLayoutService.getUserList().subscribe((Response: any) => {

this.allUserList = Response.data;

this.userList = this.allUserList;

if (this.allUserList.length > 0) {

this.noData = false;

}

else {

this.noData = true;

}

}, (error) => {

this.noData = true

})

}

itemsPerPageChange(val: any) {

this.l = this.itemsPerPage = parseInt(val);

this.p = 1;

}

search() {

this.userList = this.allUserList.filter((val) => JSON.stringify(val).toLowerCase().includes(this.searchTerm.toLowerCase()));

this.p = 1;

if (this.userList.length == 0) {

this.noData = true;

} else {

this.noData = false;

}

}

}

**storage.service.ts**

import { Injectable } from '@angular/core';

@Injectable({

providedIn: 'root'

})

export class StorageService {

constructor() { }

getValue(key: any): any {

return localStorage.getItem(key);

}

setValue(key: any, value: any): void {

localStorage.setItem(key, value);

}

removeValue(key: any): void {

localStorage.removeItem(key);

}

}

export class StorageKey {

}

**staff.component.html**

<div class="container-fluid flex-grow-1 container-p-y">

<div class="card">

<div class="card-datatable table-responsive">

<div class="dataTables\_wrapper dt-bootstrap5 no-footer">

<div class="card-header flex-column flex-md-row">

<div class="head-label text-center">

<h5 class="card-title mb-0">&nbsp;</h5>

</div>

<div class="dt-action-buttons text-end pt-3 pt-md-0">

<div class="dt-buttons">

<button class="dt-button create-new btn btn-primary" (click)="addNew()" type="button">

<span><i class="bx bx-plus me-sm-2"></i>

<span class="d-none d-sm-inline-block">Add New Staff</span>

</span>

</button>

</div>

</div>

</div>

<div class="row">

<div class="col-sm-12 col-md-6">

<div class="dataTables\_length" id="DataTables\_Table\_0\_length">

<label>Show

<select name="DataTables\_Table\_0\_length" [(ngModel)]="itemsPerPage"

(change)="itemsPerPageChange($event.target.value)" class="form-select">

<option value="5">5</option>

<option value="10">10</option>

<option value="25">25</option>

<option value="50">50</option>

<option value="75">75</option>

<option value="100">100</option>

</select> entries

</label>

</div>

</div>

<div class="col-sm-12 col-md-6 d-flex justify-content-center justify-content-md-end">

<div id="DataTables\_Table\_0\_filter" class="dataTables\_filter">

<label>Search:

<input type="search" (input)="search()" [(ngModel)]="searchTerm" class="form-control"

placeholder="Search here ..." aria-controls="DataTables\_Table\_0">

</label>

</div>

</div>

</div>

<table class="table dataTable ">

<thead>

<tr>

<th>No.</th>

<th>Staff Id</th>

<th>Name</th>

<th>Email</th>

</tr>

</thead>

<tbody class="">

<tr

\*ngFor="let item of staffList | paginate: { itemsPerPage: l, currentPage: p };let collectionIndex = index">

<td>{{ l \* (p - 1) + collectionIndex +1 }}</td>

<td>{{item.user\_id}}</td>

<td>{{item.name}}</td>

<td>{{item.email}}</td>

</tr>

<tr \*ngIf="noData">

<td colspan="4">

<h4 class="text-center mt-3">No Data Found</h4>

</td>

</tr>

</tbody>

</table>

<div class="row">

<div class="col-sm-12 col-md-6">

<div class="dataTables\_info">Showing {{staffList.length > 0 ? l \* (p - 1) + 1 : 0}} to {{l \*

(p - 1) + l < staffList.length ? l \* (p - 1) + l : staffList.length}} of

{{staffList.length}} entries</div>

</div>

<div class="col-sm-12 col-md-6">

<div class="dataTables\_paginate paging\_simple\_numbers">

<pagination-template #pagination="paginationApi" (pageChange)="p = $event">

<ul class="pagination">

<li class="page-item prev" [class.disabled]="pagination.isFirstPage()">

<a \*ngIf="!pagination.isFirstPage()" class="page-link cursor-pointer"

(click)="pagination.previous()">

<i class="tf-icon bx bx-chevron-left"></i> </a>

<div \*ngIf="pagination.isFirstPage()" class="page-link">

<i class="tf-icon bx bx-chevron-left"></i>

</div>

</li>

<li \*ngFor="let page of pagination.pages" class="page-item"

[class.active]="pagination.getCurrent() === page.value">

<a (click)="pagination.setCurrent(page.value)"

class="page-link cursor-pointer"

\*ngIf="pagination.getCurrent() !== page.value">

<span>{{ page.label }}</span>

</a>

<div class="page-link" \*ngIf="pagination.getCurrent() === page.value">

<span>{{ page.label }}</span>

</div>

</li>

<li class="page-item next" [class.disabled]="pagination.isLastPage()">

<a \*ngIf="!pagination.isLastPage()" class="page-link cursor-pointer"

(click)="pagination.next()"><i class="tf-icon bx bx-chevron-right"></i>

</a>

<div \*ngIf="pagination.isLastPage()" class="page-link"><i

class="tf-icon bx bx-chevron-right"></i> </div>

</li>

</ul>

</pagination-template>

</div>

</div>

</div>

</div>

</div>

</div>

</div>

**staff.component.ts**

import { Component, OnInit } from '@angular/core';

import { Router } from '@angular/router';

import { AdminLayoutService } from 'app/layouts/admin-layout/admin-layout.service';

@Component({

selector: 'app-staff',

templateUrl: './staff.component.html',

styleUrls: ['./staff.component.css']

})

export class StaffComponent implements OnInit {

l: number;

p: number = 1;

itemsPage: any;

staffList: any[] = [];

allStaffList: any[] = [];

itemsPerPage: number;

noData: boolean = false;

searchTerm = '';

constructor(public adminLayoutService: AdminLayoutService, public router: Router) { }

ngOnInit() {

this.l = this.itemsPerPage = 10;

this.getStaffList();

}

addNew() {

this.router.navigate([`admin/add/staff`])

}

getStaffList() {

this.adminLayoutService.getStaffList().subscribe((Response: any) => {

this.allStaffList = Response.data;

this.staffList = this.allStaffList;

if (this.allStaffList.length > 0) {

this.noData = false;

}

else {

this.noData = true;

}

}, (error) => {

this.noData = true

})

}

itemsPerPageChange(val: any) {

this.l = this.itemsPerPage = parseInt(val);

this.p = 1;

}

search() {

this.staffList = this.allStaffList.filter((val) => JSON.stringify(val).toLowerCase().includes(this.searchTerm.toLowerCase()));

this.p = 1;

if (this.staffList.length == 0) {

this.noData = true;

} else {

this.noData = false;

}

}

}

**admin-login.component.html**

<!-- Content -->

<div class="container-xxl">

<div class="authentication-wrapper authentication-basic container-p-y">

<div class="authentication-inner">

<!-- Register -->

<div class="card">

<div class="card-body">

<!-- Logo -->

<div class="app-brand justify-content-center">

<a href="javascript:void(0);" class="app-brand-link gap-2">

<span class="app-brand-text demo text-body fw-bolder text-capitalize">Face Recognization

Attandance System</span>

</a>

</div>

<!-- /Logo -->

<h4 class="mb-2">Welcome to out system !👋</h4>

<p class="mb-4">Please sign-in to your account</p>

<form class="mb-3" [formGroup]="loginForm">

<div class="mb-3">

<label for="email" class="form-label">Email</label>

<input type="text" class="form-control" name="email-username" formControlName="email"

placeholder="Enter your email or username" autofocus />

<div \*ngIf="submittedLoginData && fLoginData.email.errors" class="text-right">

<div \*ngIf="fLoginData.email.errors.required">

<span class="error\_msg">Email ID is Required.</span>

</div>

<div \*ngIf="!fLoginData.email.errors.valid && !fLoginData.email.errors.required">

<span class="error\_msg">Please Enter Valid Email ID.</span>

</div>

</div>

</div>

<div class="mb-3 form-password-toggle">

<div class="d-flex justify-content-between">

<label class="form-label" for="password">Password</label>

</div>

<div class="input-group input-group-merge">

<input [type]="hide1 ? 'text' : 'password'" class="form-control" name="password"

formControlName="password"

placeholder="&#xb7;&#xb7;&#xb7;&#xb7;&#xb7;&#xb7;&#xb7;&#xb7;&#xb7;&#xb7;&#xb7;&#xb7;"

aria-describedby="password" />

<span (click)="hide1 = !hide1" [attr.aria-label]="'Hide password'"

[attr.aria-pressed]="hide1" class="input-group-text cursor-pointer"><i

class="bx {{hide1 ? 'bx-show' : 'bx-hide'}}"></i></span>

</div>

<div \*ngIf="submittedLoginData && fLoginData.password.errors" class="text-right">

<div \*ngIf="fLoginData.password.errors.required">

<span class="error\_msg">Password is Required.</span>

</div>

</div>

</div>

<div class="mb-3">

<button class="btn btn-primary d-grid w-100" (click)="login()">Sign in</button>

</div>

</form>

</div>

</div>

<!-- /Register -->

</div>

</div>

</div>

<!-- / Content -->

**admin-login.component.ts**

import { Component, OnInit } from '@angular/core';

import { Router } from '@angular/router';

import { HttpClient } from '@angular/common/http';

import { StorageKey, StorageService } from "app/shared/storage.service";

import { FormBuilder, FormGroup, Validators } from "@angular/forms";

import { LoginLayoutService } from 'app/layouts/login-layout/login-layout.service';

@Component({

selector: 'app-admin-login',

templateUrl: './admin-login.component.html',

styleUrls: ['./admin-login.component.css']

})

export class AdminLoginComponent implements OnInit {

loginForm: FormGroup | any;

hide1 = false;

get fLoginData() {

return this.loginForm.controls;

}

submittedLoginData = false;

constructor(private router: Router, private fb: FormBuilder, public storageService: StorageService, public loginLayoutService: LoginLayoutService) { }

ngOnInit(): void {

this.defaultloginForm();

}

defaultloginForm() {

this.loginForm = this.fb.group({

email: ["", [Validators.required, Validators.pattern(/^([a-zA-Z0-9\_.+-])+\@(([a-zA-Z0-9-])+\.)+([a-zA-Z0-9]{2,4})+$/),]],

password: ["", [Validators.required]],

});

}

login() {

this.submittedLoginData = true;

if (this.loginForm.invalid) {

return;

}

let loginObj = {

email: this.loginForm.value.email,

password: this.loginForm.value.password,

}

this.loginLayoutService.login(loginObj).subscribe((response: any) => {

this.storageService.setValue('token', response.token);

this.storageService.setValue('userId', response.userId);

this.storageService.setValue('isLogin', true);

this.router.navigate(["/admin/users"]);

})

}

}

**add-user.component.html**

<div class="container-fluid flex-grow-1 container-p-y">

<div class="card">

<div class="dataTables\_wrapper dt-bootstrap5 no-footer">

<div class="card-header flex-md-row">

<div class="head-label text-center">

<h5 class="card-title mb-0">Add</h5>

</div>

<div class="dt-action-buttons text-end pt-md-0">

<div class="dt-buttons">

<button class="dt-button create-new btn btn-outline-primary" (click)="cancel()"><span><i

class='bx bx-arrow-back me-sm-2'></i><span class="d-none d-sm-inline-block">Back to

Lst</span></span></button>

</div>

</div>

</div>

<div class="card-body">

<form [formGroup]="addForm">

<div class="row">

<div class="col-xl-3 col-lg-3 col-sm-6 col-xs-6">

<div class="mb-3">

<label class="form-label required" for="basic-default-email">Name</label>

<input type="text" class="form-control" id="basic-default-company" placeholder="Name..."

formControlName="name" />

<div \*ngIf="submittedData && faddData.name.errors">

<div \*ngIf="faddData.name.errors.required">

<span class="error\_msg">Name is Required.</span>

</div>

</div>

</div>

</div>

<div class="col-xl-3 col-lg-3 col-sm-6 col-xs-6">

<div class="mb-3">

<label class="form-label required" for="basic-default-email">Email</label>

<input type="text" class="form-control" id="basic-default-company" placeholder="Email"

formControlName="email" />

<div \*ngIf="submittedData && faddData.email.errors">

<div \*ngIf="faddData.email.errors.required">

<span class="error\_msg">Email ID is Required.</span>

</div>

<div \*ngIf="!faddData.email.errors.valid && !faddData.email.errors.required">

<span class="error\_msg">Please Enter Valid Email ID.</span>

</div>

</div>

</div>

</div>

<div class="col-xl-3 col-lg-3 col-sm-6 col-xs-6">

<div class="mb-3">

<label class="form-label required" for="basic-default-email">User Type</label>

<ng-select [items]="userTypeList" class="form-control p-0"

[readonly]="userType == 'staff'" bindLabel="userTypeList" [clearable]="true"

bindValue="userTypeList" placeholder="User Type" [searchable]="false"

formControlName="user\_type" (change)="userTypeChange()">

</ng-select>

<div \*ngIf="submittedData && faddData.user\_type.errors">

<div \*ngIf="faddData.user\_type.errors.required">

<span class="error\_msg">User Type is Required.</span>

</div>

</div>

</div>

</div>

<div class="col-xl-3 col-lg-3 col-sm-6 col-xs-6"

\*ngIf="faddData.user\_type.value == 'student' || faddData.user\_type.value == 'staff'">

<div class="mb-3">

<label class="form-label" for="basic-default-email">Password</label>

<input type="password" class="form-control" id="basic-default-company"

placeholder="Password" formControlName="password" />

<div \*ngIf="submittedData && faddData.password.errors">

<div \*ngIf="faddData.password.errors.required">

<span class="error\_msg">Password is Required.</span>

</div>

</div>

</div>

</div>

<div class="col-xl-3 col-lg-3 col-sm-6 col-xs-6">

<div class="mb-3">

<label class="form-label required" for="basic-default-email">Face Image</label>

<input type="file" requiredFileType="image/png" class="form-control" accept="image/\*" id="basic-default-company" placeholder="image"

formControlName="image" (change) = "onSelectFile($event)" />

<div \*ngIf="submittedData && faddData.image.errors">

<div \*ngIf="faddData.image.errors.required">

<span class="error\_msg">Face Image is Required.</span>

</div>

<div \*ngIf="!faddData.image.errors.valid && !faddData.image.errors.required">

<span class="error\_msg">Please Uplaod Face Image.</span>

</div>

</div>

</div>

</div>

</div>

<button type="submit" class="btn btn-primary" (click)="save()">Save</button>

</form>

</div>

</div>

</div>

</div>

**add-user.component.ts**

import { Component, OnInit } from '@angular/core';

import { FormBuilder, FormGroup, Validators, FormControl } from '@angular/forms';

import { ActivatedRoute, Router } from '@angular/router';

import { AdminLayoutService } from 'app/layouts/admin-layout/admin-layout.service';

@Component({

selector: 'app-add-user',

templateUrl: './add-user.component.html',

styleUrls: ['./add-user.component.css']

})

export class AddUserComponent implements OnInit {

selectedImage: File;

addForm: FormGroup;

userType: any;

userTypeList: any[] = [];

submittedData: boolean = false;

image: any;

get faddData() {

return this.addForm.controls;

}

constructor(public fb: FormBuilder, public adminLayoutService: AdminLayoutService, public route: ActivatedRoute, public router: Router) {

}

ngOnInit(): void {

this.defaultForm();

let type: any;

this.route.params.subscribe((x: any) => {

type = x.id

})

if (type == 'staff') {

this.userTypeList = ['staff'];

this.userType = type;

this.addForm.controls.user\_type.setValue(type);

this.addForm.controls.password.setValidators([Validators.required]);

}

else {

this.userTypeList = ['student', 'visitor'];

this.addForm.controls.password.clearValidators();

}

}

defaultForm() {

this.addForm = this.fb.group({

name: ['', [Validators.required]],

email: ['', [Validators.required, Validators.pattern(/^([a-zA-Z0-9\_.+-])+\@(([a-zA-Z0-9-])+\.)+([a-zA-Z0-9]{2,4})+$/)]],

password: [''],

user\_type: [null, [Validators.required]],

image: ['',[Validators.required]],

})

}

cancel() {

if (this.userType == 'staff') {

this.router.navigate(['admin/staff']);

}

else {

this.router.navigate(['admin/users']);

}

}

userTypeChange() {

if (this.addForm.value.user\_type == 'student') {

this.addForm.controls.password.setValidators([Validators.required]);

}

else {

this.addForm.controls.password.clearValidators();

this.addForm.controls.password.updateValueAndValidity();

}

}

onSelectFile(e){

if(e.target.files){

this.selectedImage = e.target.files[0];

}

}

save() {

if (this.addForm.invalid) {

this.submittedData = true;

return

}

const formData = new FormData();

formData.append('file',this.selectedImage);

this.adminLayoutService.saveImage(formData).subscribe((Response: any) => {

if(Response.status == 200){

let saveObj = {};

if (this.addForm.value.user\_type == 'student' || this.addForm.value.user\_type == 'staff') {

saveObj = {

name: this.addForm.value.name,

email: this.addForm.value.email,

user\_type: this.addForm.value.user\_type,

password: this.addForm.value.password,

user\_id: +Response.imageUrl

}

}

else {

saveObj = {

name: this.addForm.value.name,

email: this.addForm.value.email,

user\_type: this.addForm.value.user\_type,

user\_id: +Response.imageUrl

}

}

this.adminLayoutService.saveData(saveObj).subscribe((Response: any) => {

if (this.userType == 'staff') {

this.router.navigate(['admin/staff']);

}

else {

this.router.navigate(['admin/users']);

}

})

}

});

}}

**attandance.component.html**

<div class="container-fluid flex-grow-1 container-p-y">

<div class="card">

<div class="card-datatable table-responsive">

<div class="dataTables\_wrapper dt-bootstrap5 no-footer">

<div class="row">

<div class="col-sm-12 col-md-6">

<div class="dataTables\_length" id="DataTables\_Table\_0\_length">

<label>Show

<select name="DataTables\_Table\_0\_length" [(ngModel)]="itemsPerPage"

(change)="itemsPerPageChange($event.target.value)" class="form-select">

<option value="5">5</option>

<option value="10">10</option>

<option value="25">25</option>

<option value="50">50</option>

<option value="75">75</option>

<option value="100">100</option>

</select> entries

</label>

</div>

</div>

<div class="col-sm-12 col-md-6 d-flex justify-content-center justify-content-md-end">

<div id="DataTables\_Table\_0\_filter" class="dataTables\_filter">

<label>Search:

<input type="search" (input)="search()" [(ngModel)]="searchTerm" class="form-control"

placeholder="Search here ..." aria-controls="DataTables\_Table\_0">

</label>

</div>

</div>

</div>

<table class="table dataTable ">

<thead>

<tr>

<th>No.</th>

<th>Student Id</th>

<th>Date and Time</th>

<th>User Type</th>

<th>Class Number</th>

</tr>

</thead>

<tbody class="">

<tr

\*ngFor="let item of attandanceList | paginate: { itemsPerPage: l, currentPage: p };let collectionIndex = index">

<td>{{ l \* (p - 1) + collectionIndex +1 }}</td>

<td>{{item.student\_id}}</td>

<td>{{item.created\_at | date :'dd/MM/yyyy hh:mm a'}}</td>

<td>{{item.user\_type ? item.user\_type : 'Unknown' }}</td>

<td>{{item.class\_number ? item.class\_number : '-' }}</td>

</tr>

<tr \*ngIf="noData">

<td colspan="4">

<h4 class="text-center mt-3">No Data Found</h4>

</td>

</tr>

</tbody>

</table>

<div class="row">

<div class="col-sm-12 col-md-6">

<div class="dataTables\_info">Showing {{attandanceList.length > 0 ? l \* (p - 1) + 1 : 0}} to {{l

\*

(p - 1) + l < attandanceList.length ? l \* (p - 1) + l : attandanceList.length}} of

{{attandanceList.length}} entries</div>

</div>

<div class="col-sm-12 col-md-6">

<div class="dataTables\_paginate paging\_simple\_numbers">

<pagination-template #pagination="paginationApi" (pageChange)="p = $event">

<ul class="pagination">

<li class="page-item prev" [class.disabled]="pagination.isFirstPage()">

<a \*ngIf="!pagination.isFirstPage()" class="page-link cursor-pointer"

(click)="pagination.previous()">

<i class="tf-icon bx bx-chevron-left"></i> </a>

<div \*ngIf="pagination.isFirstPage()" class="page-link">

<i class="tf-icon bx bx-chevron-left"></i>

</div>

</li>

<li \*ngFor="let page of pagination.pages" class="page-item"

[class.active]="pagination.getCurrent() === page.value">

<a (click)="pagination.setCurrent(page.value)"

class="page-link cursor-pointer"

\*ngIf="pagination.getCurrent() !== page.value">

<span>{{ page.label }}</span>

</a>

<div class="page-link" \*ngIf="pagination.getCurrent() === page.value">

<span>{{ page.label }}</span>

</div>

</li>

<li class="page-item next" [class.disabled]="pagination.isLastPage()">

<a \*ngIf="!pagination.isLastPage()" class="page-link cursor-pointer"

(click)="pagination.next()"><i class="tf-icon bx bx-chevron-right"></i>

</a>

<div \*ngIf="pagination.isLastPage()" class="page-link"><i

class="tf-icon bx bx-chevron-right"></i> </div>

</li>

</ul>

</pagination-template>

</div>

</div>

</div>

</div>

</div>

</div>

</div>

**attandance.component.ts**

import { Component, OnInit } from '@angular/core';

import { AdminLayoutService } from 'app/layouts/admin-layout/admin-layout.service';

@Component({

selector: 'app-attandance',

templateUrl: './attandance.component.html',

styleUrls: ['./attandance.component.css']

})

export class AttandanceComponent implements OnInit {

l: number;

p: number = 1;

itemsPage: any;

attandanceList: any[] = [];

allAttandanceList: any[] = [];

itemsPerPage: number;

noData: boolean = false;

searchTerm = '';

constructor(public adminLayoutService: AdminLayoutService) { }

ngOnInit() {

this.l = this.itemsPerPage = 10;

this.getattandanceList();

}

getattandanceList() {

this.adminLayoutService.getAttandanceList().subscribe((Response: any) => {

this.allAttandanceList = Response.data;

this.attandanceList = this.allAttandanceList;

if (this.allAttandanceList.length > 0) {

this.noData = false;

}

else {

this.noData = true;

}

}, (error) => {

this.noData = true

})

}

itemsPerPageChange(val: any) {

this.l = this.itemsPerPage = parseInt(val);

this.p = 1;

}

search() {

this.attandanceList = this.allAttandanceList.filter((val) => JSON.stringify(val).toLowerCase().includes(this.searchTerm.toLowerCase()));

this.p = 1;

if (this.attandanceList.length == 0) {

this.noData = true;

} else {

this.noData = false;

}

}

}

**room-assign.component.html**

<div class="container-fluid flex-grow-1 container-p-y">

<div class="card">

<div class="card-datatable table-responsive">

<div class="dataTables\_wrapper dt-bootstrap5 no-footer">

<div class="card-header flex-column flex-md-row">

<div class="head-label text-center">

<h5 class="card-title mb-0">&nbsp;</h5>

</div>

<div class="dt-action-buttons text-end pt-3 pt-md-0">

<div class="dt-buttons">

<button class="dt-button create-new btn btn-primary" (click)="addNew()" type="button">

<span><i class="bx bx-plus me-sm-2"></i>

<span class="d-none d-sm-inline-block">Add New Class Room</span>

</span>

</button>

</div>

</div>

</div>

<div class="row">

<div class="col-sm-12 col-md-6">

<div class="dataTables\_length" id="DataTables\_Table\_0\_length">

<label>Show

<select name="DataTables\_Table\_0\_length" [(ngModel)]="itemsPerPage"

(change)="itemsPerPageChange($event.target.value)" class="form-select">

<option value="5">5</option>

<option value="10">10</option>

<option value="25">25</option>

<option value="50">50</option>

<option value="75">75</option>

<option value="100">100</option>

</select> entries

</label>

</div>

</div>

<div class="col-sm-12 col-md-6 d-flex justify-content-center justify-content-md-end">

<div id="DataTables\_Table\_0\_filter" class="dataTables\_filter">

<label>Search:

<input type="search" (input)="search()" [(ngModel)]="searchTerm" class="form-control"

placeholder="Search here ..." aria-controls="DataTables\_Table\_0">

</label>

</div>

</div>

</div>

<table class="table dataTable ">

<thead>

<tr>

<th>No.</th>

<th>Camera Number</th>

<th>Class Number</th>

</tr>

</thead>

<tbody class="">

<tr

\*ngFor="let item of roomList | paginate: { itemsPerPage: l, currentPage: p };let collectionIndex = index">

<td>{{ l \* (p - 1) + collectionIndex +1 }}</td>

<td>{{item.class\_id + 1}}</td>

<td>{{item.class\_number}}</td>

</tr>

<tr \*ngIf="noData">

<td colspan="5">

<h4 class="text-center mt-3">No Data Found</h4>

</td>

</tr>

</tbody>

</table>

<div class="row">

<div class="col-sm-12 col-md-6">

<div class="dataTables\_info">Showing {{roomList.length > 0 ? l \* (p - 1) + 1 : 0}} to {{l \*

(p - 1) + l < roomList.length ? l \* (p - 1) + l : roomList.length}} of {{roomList.length}}

entries</div>

</div>

<div class="col-sm-12 col-md-6">

<div class="dataTables\_paginate paging\_simple\_numbers">

<pagination-template #pagination="paginationApi" (pageChange)="p = $event">

<ul class="pagination">

<li class="page-item prev" [class.disabled]="pagination.isFirstPage()">

<a \*ngIf="!pagination.isFirstPage()" class="page-link cursor-pointer"

(click)="pagination.previous()">

<i class="tf-icon bx bx-chevron-left"></i> </a>

<div \*ngIf="pagination.isFirstPage()" class="page-link">

<i class="tf-icon bx bx-chevron-left"></i>

</div>

</li>

<li \*ngFor="let page of pagination.pages" class="page-item"

[class.active]="pagination.getCurrent() === page.value">

<a (click)="pagination.setCurrent(page.value)"

class="page-link cursor-pointer"

\*ngIf="pagination.getCurrent() !== page.value">

<span>{{ page.label }}</span>

</a>

<div class="page-link" \*ngIf="pagination.getCurrent() === page.value">

<span>{{ page.label }}</span>

</div>

</li>

<li class="page-item next" [class.disabled]="pagination.isLastPage()">

<a \*ngIf="!pagination.isLastPage()" class="page-link cursor-pointer"

(click)="pagination.next()"><i class="tf-icon bx bx-chevron-right"></i>

</a>

<div \*ngIf="pagination.isLastPage()" class="page-link"><i

class="tf-icon bx bx-chevron-right"></i> </div>

</li>

</ul>

</pagination-template>

</div>

</div>

</div>

</div>

</div>

</div>

</div>

**room-assign.component.ts**

import { Component, OnInit } from '@angular/core';

import { Router } from '@angular/router';

import { AdminLayoutService } from 'app/layouts/admin-layout/admin-layout.service';

@Component({

selector: 'app-room-assign',

templateUrl: './room-assign.component.html',

styleUrls: ['./room-assign.component.css']

})

export class RoomAssignComponent implements OnInit {

l: number;

p: number = 1;

itemsPage: any;

roomList: any[] = [];

allRoomList: any[] = [];

itemsPerPage: number;

noData: boolean = false;

searchTerm = '';

constructor(public router: Router, public adminLayoutService: AdminLayoutService) { }

ngOnInit(): void {

this.l = this.itemsPerPage = 10;

this.getRoomList();

}

getRoomList() {

this.adminLayoutService.getRoomList().subscribe((Response: any) => {

this.allRoomList = Response.data;

this.roomList = this.allRoomList;

if (this.allRoomList.length > 0) {

this.noData = false;

}

else {

this.noData = true;

}

}, (error) => {

this.noData = true

})

}

addNew() {

this.router.navigate([`admin/room/add`])

}

itemsPerPageChange(val: any) {

this.l = this.itemsPerPage = parseInt(val);

this.p = 1;

}

search() {

this.roomList = this.allRoomList.filter((val) => JSON.stringify(val).toLowerCase().includes(this.searchTerm.toLowerCase()));

this.p = 1;

if (this.roomList.length == 0) {

this.noData = true;

} else {

this.noData = false;

}

}

}

**admin-layout.component.html**

<div class="layout-wrapper layout-content-navbar">

<div class="layout-container">

<div class="sidebar">

<app-sidebar></app-sidebar>

</div>

<div class="layout-page main-panel">

<app-navbar></app-navbar>

<div>

<router-outlet ></router-outlet>

</div>

</div>

</div>

<div class="layout-overlay layout-menu-toggle"></div>

</div>

<notifier-container></notifier-container>

**admin-layout.component.ts**

import { Component, OnInit, ViewChild, AfterViewInit } from '@angular/core';

import { Location, LocationStrategy, PathLocationStrategy, PopStateEvent } from '@angular/common';

import { Router, NavigationEnd, NavigationStart } from '@angular/router';

import PerfectScrollbar from 'perfect-scrollbar';

import \* as $ from "jquery";

import { filter, Subscription } from 'rxjs';

import { StorageService } from 'app/shared/storage.service';

@Component({

selector: 'app-admin-layout',

templateUrl: './admin-layout.component.html',

styleUrls: ['./admin-layout.component.scss']

})

export class AdminLayoutComponent implements OnInit {

private \_router: Subscription;

private lastPoppedUrl: string;

private yScrollStack: number[] = [];

constructor(public storageService: StorageService, public location: Location, private router: Router) {

let storageData = JSON.parse(storageService.getValue('isLogin'));

let token = storageService.getValue('token');

if (storageData != true) {

this.router.navigate(['login']);

}

}

ngOnInit() {

let primaryBackgroundColor = localStorage.getItem("primaryBackgroundColor") ? localStorage.getItem("primaryBackgroundColor") : "#696cff"

let primaryBorderColor = localStorage.getItem("primaryBackgroundColor") ? localStorage.getItem("primaryBackgroundColor") : "#696cff"

let primaryBackgroundTextColor = localStorage.getItem("primaryBackgroundTextColor") ? localStorage.getItem("primaryBackgroundTextColor") : "#ffffff"

let primaryTextColor = localStorage.getItem("primaryTextColor") ? localStorage.getItem("primaryTextColor") : "#696cff"

document.documentElement.style.setProperty('--primary-background-color', primaryBackgroundColor);

document.documentElement.style.setProperty('--primary-border-color', primaryBorderColor);

document.documentElement.style.setProperty('--primary-background-text-color', primaryBackgroundTextColor);

document.documentElement.style.setProperty('--primary-background-shadow-color', primaryBackgroundColor + '66');

document.documentElement.style.setProperty('--primary-background-transparent-color', primaryBackgroundColor + '29');

document.documentElement.style.setProperty('--primary-text-color', primaryTextColor);

const isWindows = navigator.platform.indexOf('Win') > -1 ? true : false;

if (isWindows && !document.getElementsByTagName('body')[0].classList.contains('sidebar-mini')) {

// if we are on windows OS we activate the perfectScrollbar function

document.getElementsByTagName('body')[0].classList.add('perfect-scrollbar-on');

} else {

document.getElementsByTagName('body')[0].classList.remove('perfect-scrollbar-off');

}

const elemMainPanel = <HTMLElement>document.querySelector('.main-panel');

const elemSidebar = <HTMLElement>document.querySelector('.sidebar .sidebar-wrapper');

this.location.subscribe((ev: PopStateEvent) => {

this.lastPoppedUrl = ev.url;

});

this.router.events.subscribe((event: any) => {

if (event instanceof NavigationStart) {

if (event.url != this.lastPoppedUrl)

this.yScrollStack.push(window.scrollY);

} else if (event instanceof NavigationEnd) {

if (event.url == this.lastPoppedUrl) {

this.lastPoppedUrl = undefined;

window.scrollTo(0, this.yScrollStack.pop());

} else

window.scrollTo(0, 0);

}

});

this.\_router = this.router.events.pipe(filter(event => event instanceof NavigationEnd)).subscribe((event: NavigationEnd) => {

elemMainPanel.scrollTop = 0;

elemSidebar.scrollTop = 0;

});

if (window.matchMedia(`(min-width: 960px)`).matches && !this.isMac()) {

let ps = new PerfectScrollbar(elemMainPanel);

ps = new PerfectScrollbar(elemSidebar);

}

const window\_width = $(window).width();

let $sidebar = $('.sidebar');

let $sidebar\_responsive = $('body > .navbar-collapse');

let $sidebar\_img\_container = $sidebar.find('.sidebar-background');

if (window\_width > 767) {

if ($('.fixed-plugin .dropdown').hasClass('show-dropdown')) {

$('.fixed-plugin .dropdown').addClass('open');

}

}

$('.fixed-plugin a').click(function (event) {

// Alex if we click on switch, stop propagation of the event, so the dropdown will not be hide, otherwise we set the section active

if ($(this).hasClass('switch-trigger')) {

if (event.stopPropagation) {

event.stopPropagation();

}

else if (window.event) {

window.event.cancelBubble = true;

}

}

});

$('.fixed-plugin .badge').click(function () {

let $full\_page\_background = $('.full-page-background');

$(this).siblings().removeClass('active');

$(this).addClass('active');

var new\_color = $(this).data('color');

if ($sidebar.length !== 0) {

$sidebar.attr('data-color', new\_color);

}

if ($sidebar\_responsive.length != 0) {

$sidebar\_responsive.attr('data-color', new\_color);

}

});

$('.fixed-plugin .img-holder').click(function () {

let $full\_page\_background = $('.full-page-background');

$(this).parent('li').siblings().removeClass('active');

$(this).parent('li').addClass('active');

var new\_image = $(this).find("img").attr('src');

if ($sidebar\_img\_container.length != 0) {

$sidebar\_img\_container.fadeOut('fast', function () {

$sidebar\_img\_container.css('background-image', 'url("' + new\_image + '")');

$sidebar\_img\_container.fadeIn('fast');

});

}

if ($full\_page\_background.length != 0) {

$full\_page\_background.fadeOut('fast', function () {

$full\_page\_background.css('background-image', 'url("' + new\_image + '")');

$full\_page\_background.fadeIn('fast');

});

}

if ($sidebar\_responsive.length != 0) {

$sidebar\_responsive.css('background-image', 'url("' + new\_image + '")');

}

});

}

ngAfterViewInit() {

this.runOnRouteChange();

}

isMaps(path) {

var titlee = this.location.prepareExternalUrl(this.location.path());

titlee = titlee.slice(1);

if (path == titlee) {

return false;

}

else {

return true;

}

}

runOnRouteChange(): void {

if (window.matchMedia(`(min-width: 960px)`).matches && !this.isMac()) {

const elemMainPanel = <HTMLElement>document.querySelector('.main-panel');

const ps = new PerfectScrollbar(elemMainPanel);

ps.update();

}

}

isMac(): boolean {

let bool = false;

if (navigator.platform.toUpperCase().indexOf('MAC') >= 0 || navigator.platform.toUpperCase().indexOf('IPAD') >= 0) {

bool = true;

}

return bool;

}

}

**admin-layout.module.ts**

import { NgModule } from '@angular/core';

import { RouterModule } from '@angular/router';

import { CommonModule, DatePipe } from '@angular/common';

import { FormsModule, ReactiveFormsModule } from '@angular/forms';

import { AdminLayoutRoutes } from './admin-layout.routing';

import { MatButtonModule } from '@angular/material/button';

import { MatInputModule } from '@angular/material/input';

import { MatRippleModule } from '@angular/material/core';

import { MatFormFieldModule } from '@angular/material/form-field';

import { MatTooltipModule } from '@angular/material/tooltip';

import { MatSelectModule } from '@angular/material/select';

import { MatDatepickerModule } from "@angular/material/datepicker";

import { NgxPaginationModule } from 'ngx-pagination';

import { NgSelectModule } from '@ng-select/ng-select';

import { NgxMatNativeDateModule } from '@angular-material-components/datetime-picker';

import { MatNativeDateModule } from '@angular/material/core';

import { DirectivesModule } from 'app/shared/directives/directives.module';

import { UsersComponent } from 'app/pages/users/users.component';

import { StaffComponent } from 'app/pages/staff/staff.component';

import { AttandanceComponent } from 'app/pages/attandance/attandance.component';

import { AddUserComponent } from 'app/pages/add-user/add-user.component';

import { RoomAssignComponent } from 'app/pages/room-assign/room-assign.component';

import { AddRoomComponent } from 'app/pages/add-room/add-room.component';

import { StaffAttandanceComponent } from 'app/pages/staff-attandance/staff-attandance.component';

import { ChangePasswordComponent } from 'app/pages/change-password/change-password.component';

@NgModule({

imports: [

CommonModule,

RouterModule.forChild(AdminLayoutRoutes),

FormsModule,

ReactiveFormsModule,

MatButtonModule,

MatRippleModule,

MatFormFieldModule,

MatInputModule,

MatSelectModule,

MatTooltipModule,

NgxPaginationModule,

NgSelectModule,

MatDatepickerModule,

NgxMatNativeDateModule,

MatNativeDateModule,

DirectivesModule

],

declarations: [

UsersComponent,

StaffComponent,

AttandanceComponent,

AddUserComponent,

RoomAssignComponent,

AddRoomComponent,

StaffAttandanceComponent,

ChangePasswordComponent

],

providers: [MatDatepickerModule, DatePipe],

})

export class AdminLayoutModule { }

**admin-layout.routing.ts**

import { Routes } from '@angular/router';

import { AddUserComponent } from 'app/pages/add-user/add-user.component';

import { AttandanceComponent } from 'app/pages/attandance/attandance.component';

import { StaffComponent } from 'app/pages/staff/staff.component';

import { RoomAssignComponent } from 'app/pages/room-assign/room-assign.component';

import { UsersComponent } from 'app/pages/users/users.component';

import { AddRoomComponent } from 'app/pages/add-room/add-room.component';

import { StaffAttandanceComponent } from 'app/pages/staff-attandance/staff-attandance.component';

import { ChangePasswordComponent } from 'app/pages/change-password/change-password.component';

export const AdminLayoutRoutes: Routes = [

{ path: 'users', component: UsersComponent },

{ path: 'add/:id', component: AddUserComponent },

{ path: 'staff', component: StaffComponent },

{ path: 'attandance', component: AttandanceComponent },

{ path: 'room', component: RoomAssignComponent },

{ path: 'room/add', component: AddRoomComponent },

{ path: 'staff-attandance', component: StaffAttandanceComponent },

{ path: 'change-password', component: ChangePasswordComponent },

];

**admin-layout.service.ts**

import { HttpClient, HttpHeaders } from '@angular/common/http';

import { Injectable } from '@angular/core';

import { CommonService } from 'app/shared/common.service';

import { StorageService } from 'app/shared/storage.service';

@Injectable({

providedIn: 'root'

})

export class AdminLayoutService {

constructor(private storageService: StorageService, public http: HttpClient, public commonService: CommonService) { }

getUserList() {

let localData = this.storageService.getValue('token');

let headers = new HttpHeaders({

'Authorization': `Bearer ${localData}`

})

return this.http.get(this.commonService.rootData.rootUrl + 'students', { headers: headers })

}

getStaffList() {

let localData = this.storageService.getValue('token');

let headers = new HttpHeaders({

'Authorization': `Bearer ${localData}`

})

return this.http.get(this.commonService.rootData.rootUrl + 'staff', { headers: headers })

}

getAttandanceList() {

let localData = this.storageService.getValue('token');

let headers = new HttpHeaders({

'Authorization': `Bearer ${localData}`

})

return this.http.get(this.commonService.rootData.rootUrl + 'attendances', { headers: headers })

}

getStaffAttandanceList() {

let localData = this.storageService.getValue('token');

let headers = new HttpHeaders({

'Authorization': `Bearer ${localData}`

})

return this.http.get(this.commonService.rootData.rootUrl + 'staff-attandances', { headers: headers })

}

saveData(data: any) {

let localData = this.storageService.getValue('token');

let headers = new HttpHeaders({

'Authorization': `Bearer ${localData}`

})

return this.http.post(this.commonService.rootData.rootUrl + 'user-form', data,{ headers: headers })

}

saveImage(data: any) {

return this.http.post(this.commonService.rootData.rootUrl + 'image', data)

}

saveClass(data: any) {

let localData = this.storageService.getValue('token');

let headers = new HttpHeaders({

'Authorization': `Bearer ${localData}`

})

return this.http.post(this.commonService.rootData.rootUrl + 'class', data, { headers: headers })

}

getRoomList(){

let localData = this.storageService.getValue('token');

let headers = new HttpHeaders({

'Authorization': `Bearer ${localData}`

})

return this.http.get(this.commonService.rootData.rootUrl + 'rooms', { headers: headers })

}

changePassword(data: any) {

return this.http.post(this.commonService.rootData.rootUrl + 'change-password', data)

}

}