1. ABSTRACT

The CNNFaceQuest project comprises a multifaceted approach, divided into three integral parts, aimed at revolutionizing attendance management through innovative applications of computer vision and deep learning technologies.

Part 1: Data Collection with Haar Cascade Classifier and IP Webcam App

The initial phase focuses on data collection, employing a Python program that harnesses computer vision techniques and the Haar Cascade classifier. This program seamlessly integrates with the IP Webcam app, capturing 100 images of an individual within a minute. This rapid image acquisition serves as the registration process for the entire attendance system. The collected images are meticulously stored by the respective individual's name, forming a comprehensive dataset for subsequent model training.

Part 2: Data Serialization and Preparation for Google Collab

Following data collection, a second Python program comes into play, utilizing the Pickle library to serialize the images and their corresponding labels. This serialization process ensures efficient input of the dataset into the Google Collab environment. By preparing the data in a structured format, the subsequent deep learning model implementation becomes more streamlined and accessible within the collaborative environment.

Part 3: Deep Learning Model Implementation and Deployment

The heart of the project lies in the third part, where a Python program implemented in Google Collab leverages Convolutional Neural Networks (CNN). The CNN model is meticulously crafted to understand and recognize facial features. Upon training, the model is downloaded as an H5 file, encapsulating the learned patterns from the extensive dataset.

The final Python program deploys this trained model in real-world scenarios using computer vision techniques and the Haar Cascade classifier. Acting as a surveillance system, the program utilizes the IP Webcam app as the CCTV source. When a registered person's face is recognized and displayed for a continuous 10 seconds, the system marks their attendance. This recognition process is enhanced by the robustness of the CNN model, ensuring accurate and reliable identification.

The seamless integration of these three components results in an end-to-end attendance management system that combines the efficiency of computer vision, the adaptability of deep learning, and the practicality of real-time deployment. The entire process, from data collection to model deployment, culminates in the automatic marking of attendance in an Excel file. This holistic approach not only automates attendance tracking but also minimizes the need for manual interventions, enhancing accuracy and efficiency.

In summary, CNNFaceQuest offers a comprehensive solution that transforms traditional attendance management through the synergy of computer vision and deep learning technologies. The project's three distinctive phases synergize to create an intelligent, automated system capable of streamlining attendance processes across diverse environments.

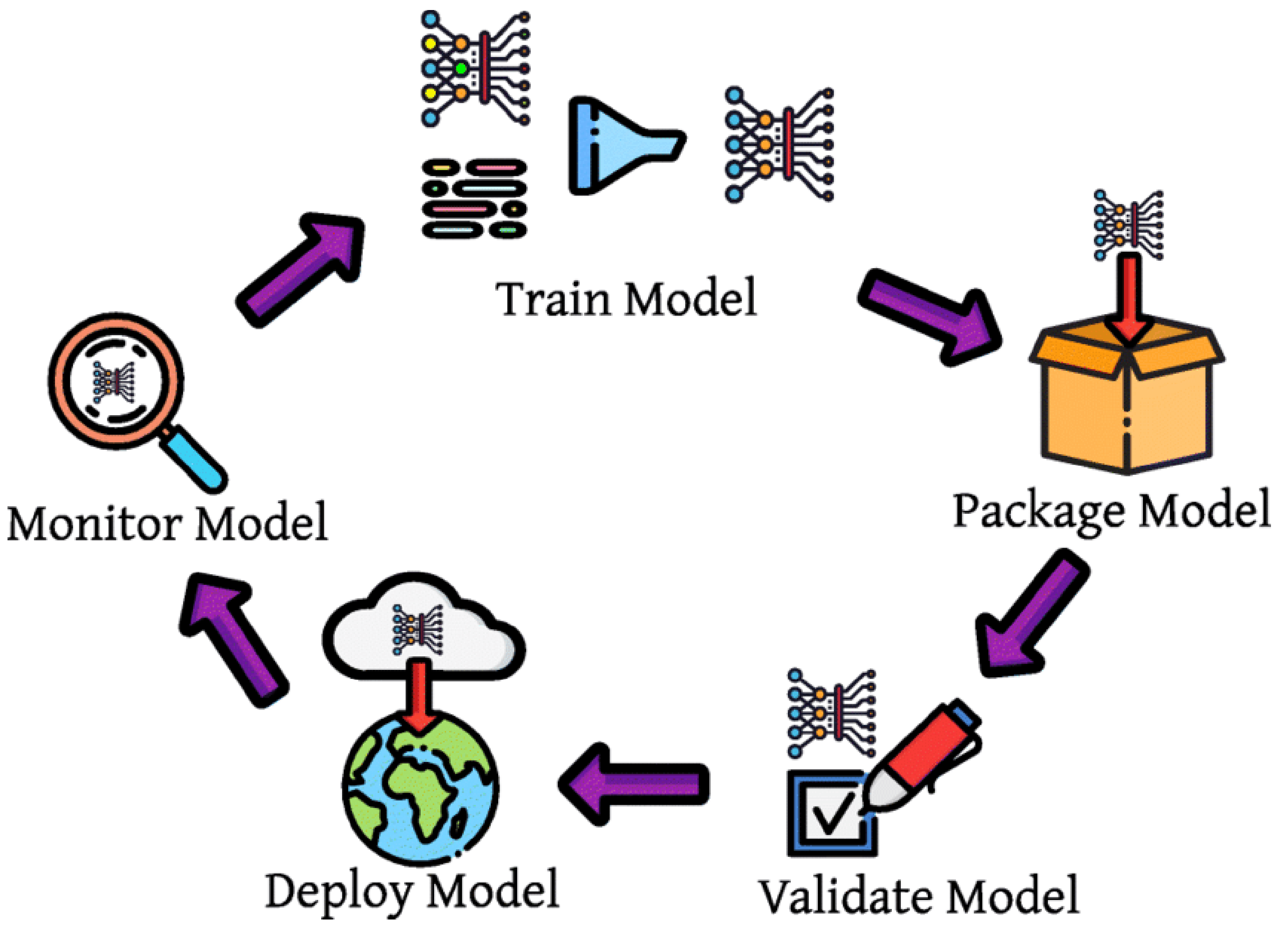


Fig No.- 01(Deep Learning Life Cycle)

2. INTRODUCTION

2.1. PURPOSE

Attendance management is a fundamental aspect of organizational functioning, playing a crucial role in tracking employee or student presence and contributing to overall operational efficiency. However, traditional methods of attendance tracking are often marred by inefficiencies, manual interventions, and the potential for errors. The CNNFaceQuest project is conceived with the overarching purpose of revolutionizing attendance management through the implementation of an advanced deep learning framework. This purpose unfolds through a meticulous multi-stage process that integrates computer vision, Haar cascade classifiers, and Convolutional Neural Networks (CNN) to create a seamless and efficient attendance tracking system.

2.1.1. Understanding the Need for Innovation:

The motivation behind the CNNFaceQuest project stems from the recognition of the limitations and challenges associated with traditional attendance management systems. Manual methods, such as paper-based sign-in sheets or manual entry into electronic systems, are not only prone to errors but also consume valuable time and resources. Inconsistent data collection practices and the potential for buddy punching further diminish the accuracy and reliability of traditional systems.

Moreover, the increasing scale and complexity of modern organizations demand a more sophisticated and automated approach to attendance tracking. As organizations expand globally and adopt flexible work arrangements, the need for a system that can adapt to diverse environments and capture real-time data becomes imperative. The CNNFaceQuest project seeks to address these challenges by introducing a cutting-edge solution that combines the power of computer vision and deep learning to streamline and enhance the entire attendance management process.

2.1.2. The Three-Phase Approach:

The project's purpose is realized through a carefully designed three-phase approach. In the first phase, a Python program is developed to leverage computer vision and Haar cascade classifiers, enabling the rapid collection of 100 images of an individual within a minute using the IP Webcam app. This phase serves as the registration process, creating a robust dataset labelled with individuals' names.

The second phase involves pre-processing the collected data and training a CNN model in the Google Collab environment. This phase is critical for ensuring the accuracy and efficiency of face detection and recognition. The trained model is then saved as an H5 file, ready for deployment in the final phase.

In the third and final phase, the trained CNN model is deployed into a Python program, utilizing Haar cascade classifiers, computer vision techniques, and the IP Webcam app as a real-time CCTV substitute. The program continuously recognizes registered individuals, marking their attendance in an Excel file if their name consistently appears below the recognition box for 10 seconds.

2.1.3. Advancing Technological Frontiers:

The overarching purpose of the CNNFaceQuest project is to advance technological frontiers in attendance management. By integrating deep learning techniques, the project aims to achieve a level of precision and efficiency that is unattainable through traditional methods. The use of CNNs facilitates robust face detection and recognition, even in diverse environments and lighting conditions, making the system adaptable to the dynamic nature of modern workplaces and educational institutions.

Beyond mere attendance tracking, the project envisions contributing to the broader landscape of smart and automated systems within organizations. The integration of advanced technologies not only addresses the specific challenges of attendance management but also sets the stage for future innovations in areas such as security, access control, and data analytics.

2.1.4. Enhancing Operational Efficiency:

At its core, the CNNFaceQuest project is driven by the purpose of enhancing operational efficiency within organizations. By automating the attendance tracking process, the project minimizes the need for manual interventions, reducing the likelihood of errors and freeing up valuable resources. The seamless integration of the system into existing organizational workflows ensures a smooth transition and minimizes disruptions.

The real-time recognition capabilities of the system further contribute to efficiency by providing instant and accurate attendance data. This real-time feature is particularly valuable in scenarios where quick decision-making based on attendance information is crucial, such as emergency evacuations, security incidents, or dynamic scheduling changes.

2.1.5. Facilitating Data-Driven Decision-Making:

An inherent aspect of the project's purpose is to facilitate data-driven decision-making. The automated system generates a wealth of accurate and real-time attendance data that can be leveraged for strategic insights. Organizations can analyse attendance patterns, identify trends, and make informed decisions to optimize resource allocation, improve scheduling, and enhance overall organizational productivity.

The project aligns with the growing trend of leveraging data analytics to drive organizational success. By providing a reliable and accurate source of attendance data, the CNNFaceQuest system becomes a valuable tool for organizations seeking to adopt a more proactive and data-centric approach to decision-making.

2.1.6. Ensuring Ethical Considerations and Privacy:

While the project is driven by technological innovation, it is equally guided by a commitment to ethical considerations and privacy. The use of facial recognition technology raises legitimate concerns about privacy and data security. As such, the project places a strong emphasis on implementing ethical practices, ensuring transparent communication with users, and incorporating robust security measures to safeguard sensitive information.

By addressing these ethical considerations, the CNNFaceQuest project aims to set a standard for responsible deployment of facial recognition technology. This aligns with the broader societal discourse on the responsible and ethical use of artificial intelligence and deep learning technologies.

2.2. PROJECT SCOPE

The scope of the CNNFaceQuest project encompasses a multifaceted approach to revolutionizing attendance management, utilizing cutting-edge technologies such as computer vision, Haar cascade classifiers, and Convolutional Neural Networks (CNN). The project's scope extends across various dimensions, from technical intricacies to ethical considerations, aiming to provide a holistic solution to the challenges associated with traditional attendance tracking systems.

2.2.1. Technical Scope:

The technical scope of the CNNFaceQuest project is expansive, covering three distinct phases that seamlessly integrate different technologies to create a robust attendance management system.

- Phase 1 - Data Collection:
  The initial phase involves the development of a Python program using computer vision and Haar cascade classifiers. The program interfaces with the IP Webcam app to capture 100 images of an individual within a minute, serving as the registration process for the attendance system. This phase sets the groundwork for subsequent data pre-processing and model training.

- Phase 2 - Data Pre-processing and Model Training:
  The second phase focuses on pre-processing the collected images and labels, preparing them for input into the Google Collab environment. Here, a CNN-based model is trained to achieve optimal face detection and recognition. The trained model is saved as an H5 file for deployment in the final phase.

- Phase 3 - Deployment and Real-Time Recognition:
  The last phase involves deploying the trained CNN model into a Python program that utilizes Haar cascade classifiers, computer vision techniques, and the IP Webcam app as a real-time CCTV substitute. The program continuously recognizes registered individuals, marking their attendance in an Excel file if their name consistently appears below the recognition box for 10 seconds.

The technical scope extends beyond the development phases, encompassing considerations for system scalability, adaptability to different environments, and integration with existing organizational workflows. The system is designed to be versatile, catering to the specific needs of various organizations, whether in educational institutions, corporate offices, or other settings.

2.2.2. Operational Scope:

The operational scope of the CNNFaceQuest project centres on enhancing the efficiency of attendance management within organizations. By automating the entire process, the project aims to reduce manual interventions, minimize errors, and provide real-time, accurate attendance data.

- Efficiency Enhancement:
  The project seeks to streamline attendance tracking processes, ensuring that organizations can manage attendance more efficiently. The real-time recognition capabilities contribute to the immediacy of attendance data, allowing for quicker decision-making and response to dynamic situations.

- Resource Optimization:
  Through the automation of attendance tracking, the project enables organizations to optimize resources that would otherwise be dedicated to manual data entry and verification. This includes both time and personnel resources, fostering a more efficient allocation of organizational assets.

- Adaptability to Diverse Environments:
  Recognizing the varied nature of organizational environments, the project is designed to adapt to different settings and conditions. Whether in well-lit office spaces or more challenging lighting conditions, the CNNFaceQuest system aims to maintain its accuracy and reliability.

2.2.3. Data-Driven Decision-Making Scope:

A crucial aspect of the project's scope lies in its contribution to data-driven decision-making within organizations. By providing a reliable source of attendance data, the CNNFaceQuest system enables organizations to derive insights, identify patterns, and make informed decisions to enhance overall productivity.

- Strategic Insights:
  The system generates a wealth of attendance data, offering organizations the opportunity to analyse attendance patterns and trends. This data can inform strategic decisions related to resource planning, scheduling optimizations, and other aspects crucial to organizational success.

- Proactive Decision-Making:
  The real-time nature of the system allows organizations to be more proactive in their decision-making. Whether responding to unexpected changes in attendance or planning for future events, the CNNFaceQuest project empowers organizations to take a proactive approach based on reliable attendance data.

2.2.4. Ethical Considerations and Privacy Scope:

Addressing ethical considerations and privacy concerns is an integral part of the CNNFaceQuest project's scope. The deployment of facial recognition technology necessitates a careful and responsible approach to ensure user privacy and data security.

- Ethical Practices:
  The project places a strong emphasis on implementing ethical practices throughout its development and deployment. Transparent communication with users, consent mechanisms, and adherence to ethical guidelines are central to the project's ethical framework.

- Privacy Measures:
  To safeguard sensitive information, the project incorporates robust privacy measures. This includes secure storage of facial data, encryption protocols, and measures to prevent unauthorized access. Privacy considerations extend not only to individuals whose attendance is being tracked but also to the broader organizational context.

2.2.5. Challenges and Limitations Scope:

The project's scope also encompasses an acknowledgment of potential challenges and limitations. Understanding these factors is crucial for realistic expectations and ongoing refinement of the system.

- Environmental Challenges:
  The system may face challenges in diverse environmental conditions, such as low lighting or crowded spaces. The project scope includes efforts to address and mitigate these challenges, ensuring the system's adaptability across a range of scenarios.

- User Acceptance:
  Acknowledging potential concerns related to facial recognition technology, the project addresses the importance of user acceptance. The scope encompasses efforts to educate and involve users in the deployment process, fostering a sense of trust and understanding.

2.2.6. Future Development and Innovation Scope:

The CNNFaceQuest project is not just a static solution but part of an evolving landscape of technological advancements. The project's scope includes considerations for future development and innovation, with a commitment to staying abreast of emerging technologies and evolving organizational needs.

- Continuous Improvement:
  The project scope includes provisions for continuous improvement based on feedback, technological advancements, and changing organizational requirements. This ensures that the CNNFaceQuest system remains relevant and effective in the long term.

- Integration with Emerging Technologies:
  As technological landscapes evolve, the project's scope includes exploration and potential integration with emerging technologies that could enhance the capabilities of the attendance management system. This forward-looking approach positions the project as a dynamic and adaptive solution.

2.3. PRODUCT FEATURES

The CNNFaceQuest project incorporates a myriad of features designed to revolutionize attendance management through the seamless integration of computer vision, Haar cascade classifiers, and Convolutional Neural Networks (CNN). These features, spread across the three project phases, collectively contribute to the system's efficiency, accuracy, and adaptability. Let's delve into the key features that define the CNNFaceQuest project:

2.3.1. Comprehensive Data Collection:

The initial phase of the CNNFaceQuest project involves a robust data collection mechanism. Utilizing computer vision and Haar cascade classifiers, the system interfaces with the IP Webcam app to capture 100 images of an individual within a minute. This comprehensive data collection serves as the foundation for the subsequent phases, ensuring a diverse dataset for model training.

2.3.2. Rapid Registration Process:

The data collection process serves as a rapid registration mechanism, allowing individuals to be swiftly enrolled in the attendance system. The speed and efficiency of this phase are crucial for minimizing disruptions and ensuring a seamless transition to the automated attendance management system.

2.3.3. Adaptive to Diverse Environments:

The system is designed to adapt to diverse environmental conditions, addressing challenges such as varying lighting conditions and crowded spaces. This adaptability enhances the system's reliability in different settings, making it suitable for deployment in a range of organizational environments.

2.3.4. Streamlined Data Pre-processing:

In the second phase, the project incorporates features for streamlined data pre-processing. A dedicated Python program is employed to pickle the collected images and labels, optimizing the dataset for input into the Google Collab environment. This feature ensures that the data is prepared efficiently for subsequent model training.

2.3.5. Google Collab Integration:

Leveraging the capabilities of the Google Collab environment, the CNNFaceQuest project facilitates seamless model training. The integration with Google Collab allows for the utilization of cloud-based resources, enhancing the scalability and accessibility of the deep learning model training process.

2.3.6. CNN-Based Model Training:

A central feature of the project is the implementation of a Convolutional Neural Network (CNN) for model training. This deep learning approach is specifically tailored for face detection and recognition, ensuring a high level of accuracy in identifying registered individuals. The CNN-based model is trained to recognize facial features with precision, contributing to the overall effectiveness of the attendance management system.

2.3.7. H5 File Model Storage:

To ensure the portability and accessibility of the trained model, the project features the storage of the model as an H5 file. This file format allows for easy retrieval and deployment, ensuring that the trained CNN model can be efficiently integrated into the final phase of the project.

2.3.8. Real-time Deployment:

The third and final phase involves the deployment of the trained CNN model into a Python program. This deployment is executed in real-time, allowing for continuous recognition of registered individuals. The system leverages Haar cascade classifiers, computer vision techniques, and the IP Webcam app as a real-time CCTV substitute, contributing to the immediacy and effectiveness of attendance tracking.

2.3.9. Continuous Recognition:

A standout feature of the CNNFaceQuest project is its ability to continuously recognize registered individuals. The system operates in real-time, enabling immediate detection of individuals within the monitored environment. Continuous recognition enhances the system's responsiveness and contributes to the accuracy of attendance tracking.

2.3.10. Triggered Attendance Submission:

The project incorporates a triggered attendance submission mechanism, providing users with control over the attendance recording process. A user-initiated button activates the attendance system, allowing for flexibility in marking attendance. This feature ensures that attendance is recorded only when desired, preventing inadvertent entries.

2.3.11. Flexibility and Control:

Recognizing the importance of user control, the system features an option to end the execution. This flexibility allows users to conclude the attendance tracking process at their discretion, contributing to a user-friendly experience and aligning with organizational preferences.

2.3.12. Automated Attendance Recording:

The hallmark feature of the CNNFaceQuest project is its automated attendance recording capability. If a person's name consistently appears below the recognition box for 10 seconds, the system autonomously marks their attendance in an Excel file. This automation minimizes manual interventions, reduces the chances of errors, and significantly streamlines the attendance management process.

2.3.13. Excel File Integration:

The recorded attendance is seamlessly integrated into an Excel file, providing a structured and easily accessible format for attendance records. This feature ensures compatibility with standard office tools, facilitating further analysis and reporting based on the attendance data.

2.3.14. Real-Time Feedback:

The system provides real-time feedback by displaying the recognized individual's name below the recognition box. This feature enhances transparency and enables users to verify their attendance status instantly, contributing to a responsive and user-centric attendance management experience.

2.3.15. Privacy Measures:

Acknowledging the sensitive nature of facial recognition technology, the CNNFaceQuest project incorporates privacy measures. These measures include secure storage of facial data, encryption protocols, and a commitment to ethical practices, ensuring that user privacy is prioritized throughout the system's operation.

2.3.16. Future-Ready Architecture:

The CNNFaceQuest project is built with a future-ready architecture, allowing for continuous improvement and integration with emerging technologies. This feature positions the project as a dynamic solution capable of evolving alongside advancements in the field of deep learning and attendance management.

2.3.17. Scalability and Adaptability:

The system is designed to be scalable and adaptable to different organizational scales and structures. Whether implemented in small businesses, educational institutions, or large corporate environments, the CNNFaceQuest project can be tailored to meet the specific attendance management needs of diverse organizations.

2.3.18. User-Friendly Interface:

The project features a user-friendly interface, ensuring accessibility for individuals interacting with the attendance system. The simplicity of the interface contributes to a positive user experience, fostering acceptance and ease of use among employees or students.

2.3.19. Error Handling and Reporting:

To enhance system robustness, the CNNFaceQuest project incorporates error handling mechanisms. In cases where recognition errors or system anomalies occur, the project includes reporting functionalities to alert administrators and facilitate troubleshooting.

2.3.20. Integration with Organizational Workflows:

The CNNFaceQuest system is designed for seamless integration into existing organizational workflows. This feature ensures that the transition to the automated attendance management system is smooth, with minimal disruptions to established processes.

3. WORKDONE IN RELATED AREA

The exploration of the "Work Done in Related Area" is a critical facet in understanding the context and evolution of advancements within the field under consideration. In any research or project endeavour, delving into the existing body of work provides a foundation for identifying gaps, building upon established knowledge, and shaping the trajectory of innovation. This section serves as a comprehensive overview, shedding light on the key contributions, methodologies, and breakthroughs that have paved the way for the current project. As we embark on a journey through the work already accomplished in the related area, we unravel the intricacies of past endeavours, setting the stage for the unique contributions and novel perspectives that our project brings to the forefront. From seminal research papers to pioneering technologies, this exploration illuminates the intellectual landscape within which our work is situated, establishing a rich tapestry against which the significance and innovation of our contributions become evident.

In recent years, advancements in technology have revolutionized the management of student attendance, ushering in an era of efficiency and precision. Automated systems, such as radio frequency identification (RFID) cards, fingerprint scanning, facial recognition, and cloud computing infrastructure, have become integral tools in educational institutions. These innovative solutions not only eliminate the tediousness of manual attendance tracking but also significantly enhance accuracy. RFID cards streamline the process by swiftly identifying students, while fingerprint scanning and facial recognition offer non-invasive yet highly secure authentication methods. The integration of cloud computing ensures seamless data storage and accessibility. Overall, these technologies collectively contribute to a more streamlined and effective attendance management system, fostering a technologically advanced and efficient learning environment. **[1]**

The research delves into the realm of automated recognition technologies designed to expedite the entry of students into educational institutions, supplanting slower, traditional methods. This technological shift has garnered substantial traction across diverse organizations and academic settings. A meticulous systematic review forms the crux of this investigation, meticulously analyzing and categorizing 90 pertinent papers. This comprehensive examination not only unveils the various contributions made by these studies but also sheds light on the existing gaps, thereby pinpointing avenues that necessitate further research within this burgeoning field. By systematically distilling the wealth of information, the research not only underscores the current momentum behind automated recognition technologies but also charts a roadmap for future exploration and refinement in this dynamic domain. **[2]**

This paper unveils an innovative video-based facial recognition system that operates seamlessly without requiring manual intervention. The system, at its core, autonomously identifies faces, cross-references them with a comprehensive student database, and meticulously logs attendance data into an Excel spreadsheet. The research takes a deep dive into the nuances of facial recognition accuracy, employing two distinct algorithms—principal component analysis (PCA) and linear discriminant analysis (LDA). By meticulously comparing these algorithms, the study provides valuable insights into their respective efficacies. This cutting-edge approach not only underscores the potential of video-based facial recognition in streamlining attendance management but also contributes nuanced findings to the broader discourse on the optimal application of PCA and LDA in enhancing facial recognition precision. **[3]**

Automation of student attendance is facilitated through various technologies, encompassing RFID, biometrics, and video facial recognition. RFID utilizes contactless tracking through tags, ensuring a seamless and efficient attendance tracking process. Biometric options, including fingerprint, iris, and facial recognition, offer secure and personalized identification methods. Video-based systems take attendance by processing class videos, enhancing convenience and automation. Each technology option presents distinct advantages, and the selection among them hinges on considerations such as cost-effectiveness, accuracy, and privacy concerns. The versatility of these automated solutions reflects the ongoing technological evolution in educational settings, providing institutions with the flexibility to choose the most fitting solution based on their unique needs and priorities. **[4]**

Diverse technologies have been harnessed to streamline and automate student attendance, with a focus on biometric systems such as fingerprint scanning, facial recognition, and iris scanning. Additionally, RFID-based solutions and the integration of cloud computing play pivotal roles in modern attendance management. The implementation of these automated methods not only alleviates the burden of manual attendance tracking but also yields heightened accuracy in recording results. Educational institutions benefit significantly from this transition, experiencing notable time and resource savings. The synergistic use of biometrics, RFID, and cloud computing showcases the adaptability of technology in enhancing operational efficiency within educational contexts, providing a seamless and precise approach to attendance management. **[5]**

4. SYSTEM ANALYSIS

4.1. USER REQUIRMENT

4.1.1. Functional Requirements:

- **Data Collection:**
  - Requirement 1: The system shall capture 100 images of an individual within 1 minute using the IP Webcam app.
  - Requirement 2: The system shall label each image with the individual's name during the data collection phase.
  - Requirement 3: The system shall store the labeled images for future use in model training.

- **Data Pre-processing:**
  - Requirement 1: The system shall pre-process the collected data, including images and labels, for efficient input into the Google Collab environment.
  - Requirement 2: The system shall use pickling to optimize the dataset for subsequent model training.

- **Model Training:**
  - Requirement 1: The system shall implement a CNN-based model in the Google Collab environment for face detection and recognition.
  - Requirement 2: The system shall save the trained model as an H5 file for future deployment.

- **Real-time Deployment:**
  - Requirement 1: The system shall deploy the trained CNN model in real-time, utilizing Haar cascade classifiers and computer vision techniques.
  - Requirement 2: The system shall interface with the IP Webcam app as a real-time CCTV substitute during deployment.

- **Continuous Recognition:**
  - Requirement 1: The system shall continuously recognize registered individuals within the monitored environment.
  - Requirement 2: The system shall display the recognized individual's name below the recognition box in real-time.

- **Triggered Attendance Submission:**
    - Requirement 1: The system shall provide a user-initiated button to activate the attendance system.
    - Requirement 2: The system shall mark attendance only when the user triggers the attendance submission.

- **Automated Attendance Recording:**
    - Requirement 1: The system shall autonomously mark attendance in an Excel file if a person's name consistently appears below the recognition box for 10 seconds.
    - Requirement 2: The system shall generate a timestamp for each attendance entry.

- **Excel File Integration:**
    - Requirement 1: The system shall seamlessly integrate recorded attendance into an Excel file.
    - Requirement 2: The Excel file shall provide a structured format for attendance records.

4.1.2. Non-Functional Requirements:

- **Privacy and Security:**
    - Requirement 1: The system shall prioritize user privacy by implementing secure storage and encryption measures for facial data.
    - Requirement 2: The system shall comply with ethical guidelines and legal regulations regarding facial recognition technology.

- **Adaptability:**
    - Requirement 1: The system shall adapt to diverse environmental conditions, including variations in lighting and crowded spaces.
    - Requirement 2: The system shall be scalable and adaptable to different organizational settings.

- **User-Friendly Interface:**
    - Requirement 1: The system shall feature a user-friendly interface for easy interaction.
    - Requirement 2: The system shall provide real-time feedback to users for instant verification of attendance status.

- **Error Handling and Reporting:**
    - Requirement 1: The system shall incorporate error handling mechanisms to address recognition errors or anomalies.
    - Requirement 2: The system shall provide reporting functionalities to alert administrators of system issues.

- **Future-Ready Architecture:**
    - Requirement 1: The system shall be built with a future-ready architecture to facilitate continuous improvement and integration with emerging technologies.
    - Requirement 2: The system shall allow for updates and enhancements based on user feedback and technological advancements.

4.2. HARDWARE REQUIREMENTS

4.2.1. Server Infrastructure:

- **Server Type:**
    - The CNNFaceQuest system requires a high-performance server capable of handling real-time image processing and deep learning computations.
    - The server should be equipped with a multi-core processor to efficiently execute parallelized tasks involved in model training and recognition.

- **Memory (RAM):**
    - The server must have a minimum of 16GB RAM to facilitate the efficient loading and processing of large datasets during model training.
    - For optimal performance, a higher RAM capacity, such as 32GB or more, is recommended for handling concurrent tasks during real-time deployment.

- **Storage:**
    - The server must have sufficient storage capacity to accommodate the dataset for model training, H5 file storage, and log files.
    - A minimum of 500GB SSD storage is recommended for faster data retrieval and reduced latency.

- **Graphics Processing Unit (GPU):**
    - The server should be equipped with a high-performance GPU to accelerate deep learning computations during model training.
    - A dedicated GPU, such as NVIDIA GeForce GTX or higher, is recommended to enhance the efficiency of CNN-based algorithms.

4.2.2. Client Devices:

- **Personal Computers/Laptops:**
  - Users interacting with the CNNFaceQuest system on personal computers or laptops should have a minimum of 8GB RAM for smooth operation of the user interface.
  - Modern browsers such as Google Chrome or Mozilla Firefox are recommended for optimal compatibility.

- **Mobile Devices:**
  - Mobile devices used for accessing the CNNFaceQuest system should have a responsive and mobile-friendly user interface.
  - The system should be compatible with popular mobile browsers, including Safari for iOS and Chrome for Android.

4.2.3. Cameras:

- **Camera Type:**
  - The system is designed to work with standard USB cameras for data collection during the registration phase.
  - For real-time deployment, the system should be compatible with IP cameras or built-in device cameras on client devices.

- **Camera Resolution:**
  - Cameras used for data collection must have a minimum resolution of 720p for clear and accurate image capture.
  - Higher-resolution cameras (1080p or higher) are recommended for improved facial feature recognition during real-time deployment.

4.2.4. Networking Infrastructure:

- **Internet Connection:**
  - A stable and high-speed internet connection is essential for seamless data transfer between client devices and the server.
  - The system should be designed to handle intermittent connectivity issues and resume normal operation when the connection is restored.

- **Local Area Network (LAN):**
    - The CNNFaceQuest system requires a robust and secure local area network to facilitate communication between client devices and the server.
    - Wired Ethernet connections are recommended for enhanced stability and reduced latency.

4.2.5. External Hardware:

- **Trigger Button:**
    - For user-initiated attendance submission, an external trigger button should be provided.
    - The button should be compatible with standard USB interfaces for easy integration with client devices.

4.2.6. Security Measures:

- **Firewall:**
    - The server must be equipped with a robust firewall to protect against unauthorized access and potential security threats.
    - The firewall should be configured to allow secure communication between client devices and the server.

- **Encryption Protocols:**
    - All data transmitted between client devices and the server should be encrypted using industry-standard protocols (e.g., HTTPS) to ensure data security.
    - The server should support the latest encryption standards for secure storage of facial data.

4.2.7. Scalability Considerations:

- **Modular Architecture:**
    - The CNNFaceQuest system should be designed with a modular architecture to facilitate future scalability.
    - The server infrastructure should support easy integration of additional hardware components to accommodate increased user load.

- **Load Balancing:**
  - Load balancing mechanisms should be implemented to distribute incoming requests evenly across multiple servers, ensuring optimal performance during peak usage.
  - Load balancing configurations should be adaptable to changes in user demand and system resources.

The Hardware Requirements Specification for the CNNFaceQuest project delineates the essential components and configurations necessary for the successful implementation and operation of the automated attendance management system. By addressing server specifications, client devices, cameras, networking infrastructure, and security measures, this document provides a detailed roadmap for the procurement and setup of the hardware infrastructure. Adherence to these requirements ensures the system's optimal performance, scalability, and reliability in diverse organizational settings.

4.3. SOFTWARE REQUIREMENTS

4.3.1. Functional Requirements:

- **Data Collection:**
  - The system must capture 100 images of an individual within 1 minute using the IP Webcam app.
  - Each image captured during data collection must be labelled with the individual's name for identification.
  - Labelled images must be stored for future use in model training.

- **Data Pre-processing:**
  - The system must pre-process collected data, including images and labels, for efficient input into the Google Collab environment.
  - Data pre-processing must utilize pickling to optimize the dataset for subsequent model training.

- **Model Training:**
  - The system must implement a Convolutional Neural Network (CNN) for face detection and recognition in the Google Collab environment.
  - The trained model must be saved as an H5 file for future deployment.

- **Real-time Deployment:**
    - The system must deploy the trained CNN model in real-time, utilizing Haar cascade classifiers and computer vision techniques.
    - The system must interface with the IP Webcam app as a real-time CCTV substitute during deployment.

- **Continuous Recognition:**
    - The system must continuously recognize registered individuals within the monitored environment.
    - The recognized individual's name must be displayed below the recognition box in real-time.

- **Triggered Attendance Submission:**
    - The system must provide a user-initiated button to activate the attendance system.
    - Attendance marking must occur only when the user triggers the attendance submission.

- **Automated Attendance Recording:**
    - The system must autonomously mark attendance in an Excel file if a person's name consistently appears below the recognition box for 10 seconds.
    - The system must generate a timestamp for each attendance entry.

- **Excel File Integration:**
    - The system must seamlessly integrate recorded attendance into an Excel file.
    - The Excel file must provide a structured format for attendance records.

4.3.2. Non-Functional Requirements:

- **Privacy and Security:**
    - The system must prioritize user privacy by implementing secure storage and encryption measures for facial data.
    - The system must comply with ethical guidelines and legal regulations regarding facial recognition technology.

- **Adaptability:**
  - The system must adapt to diverse environmental conditions, including variations in lighting and crowded spaces.
  - The system must be scalable and adaptable to different organizational settings.

- **User-Friendly Interface:**
  - The system must feature a user-friendly interface for easy interaction.
  - Real-time feedback must be provided to users for instant verification of attendance status.

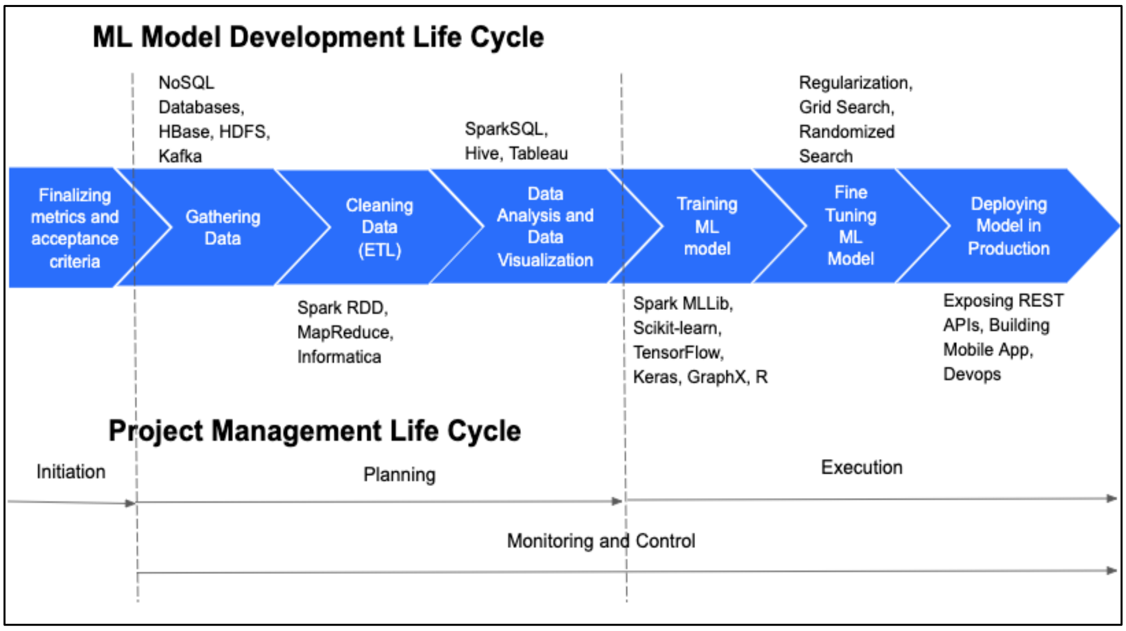
- **Error Handling and Reporting:**
  - The system must incorporate error handling mechanisms to address recognition errors or anomalies.
  - Reporting functionalities must be provided to alert administrators of system issues.

- **Future-Ready Architecture:**
  - The system must be built with a future-ready architecture to facilitate continuous improvement and integration with emerging technologies.
  - Updates and enhancements must be allowed based on user feedback and technological advancements.

5. SYSTEM DESIGN & SPECIFICATIONS

5.1. HIGH LEVEL DESIGN (HLD)

5.1.1. PROJECT MODEL



ML Model Development Life Cycle
NoSQL Databases, HBase, HDFS, Kafka
SparkSQL, Hive, Tableau
Regularization, Grid Search, Randomized Search
Finalizing metrics and acceptance criteria
Gathering Data
Cleaning Data (ETL)
Data Analysis and Data Visualization
Training ML model
Fine Tuning ML Model
Deploying Model in Production
Spark RDD, MapReduce, Informatica
Spark MLLib, Scikit-learn, TensorFlow, Keras, GraphX, R
Exposing REST APIs, Building Mobile App, Devops
Project Management Life Cycle
Initiation
Planning
Execution
Monitoring and Control

Fig No.- 02(Machine Learning Life Cycle, Project Model)

5.1.2. STRUCTURE CHART

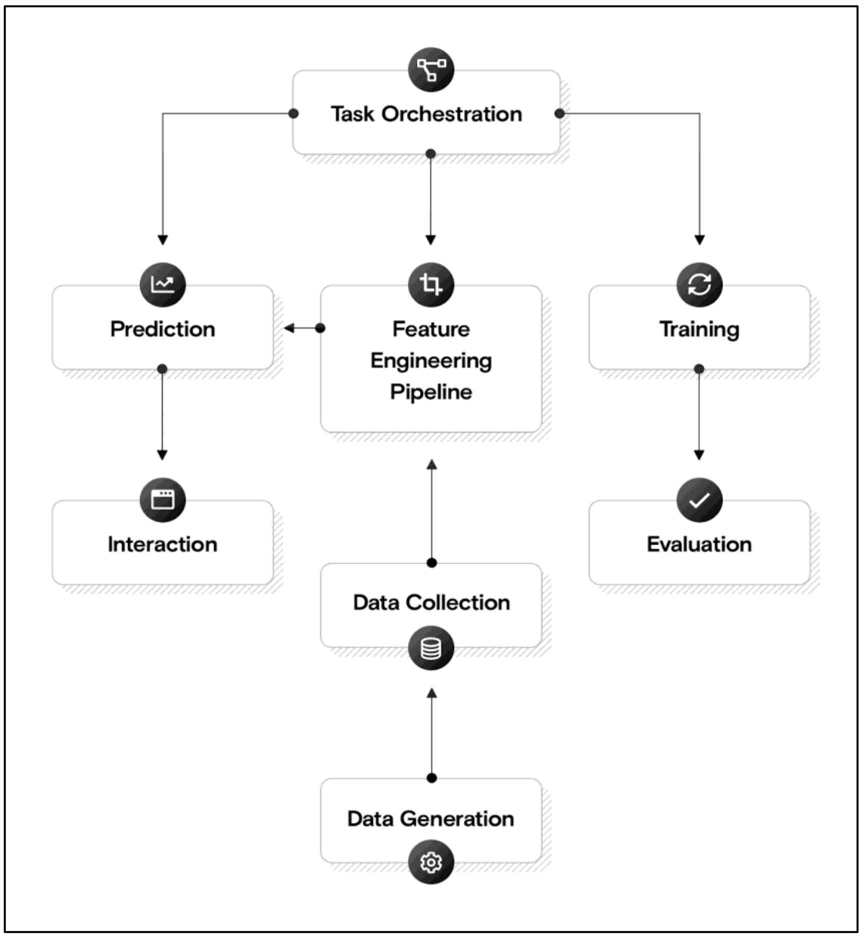
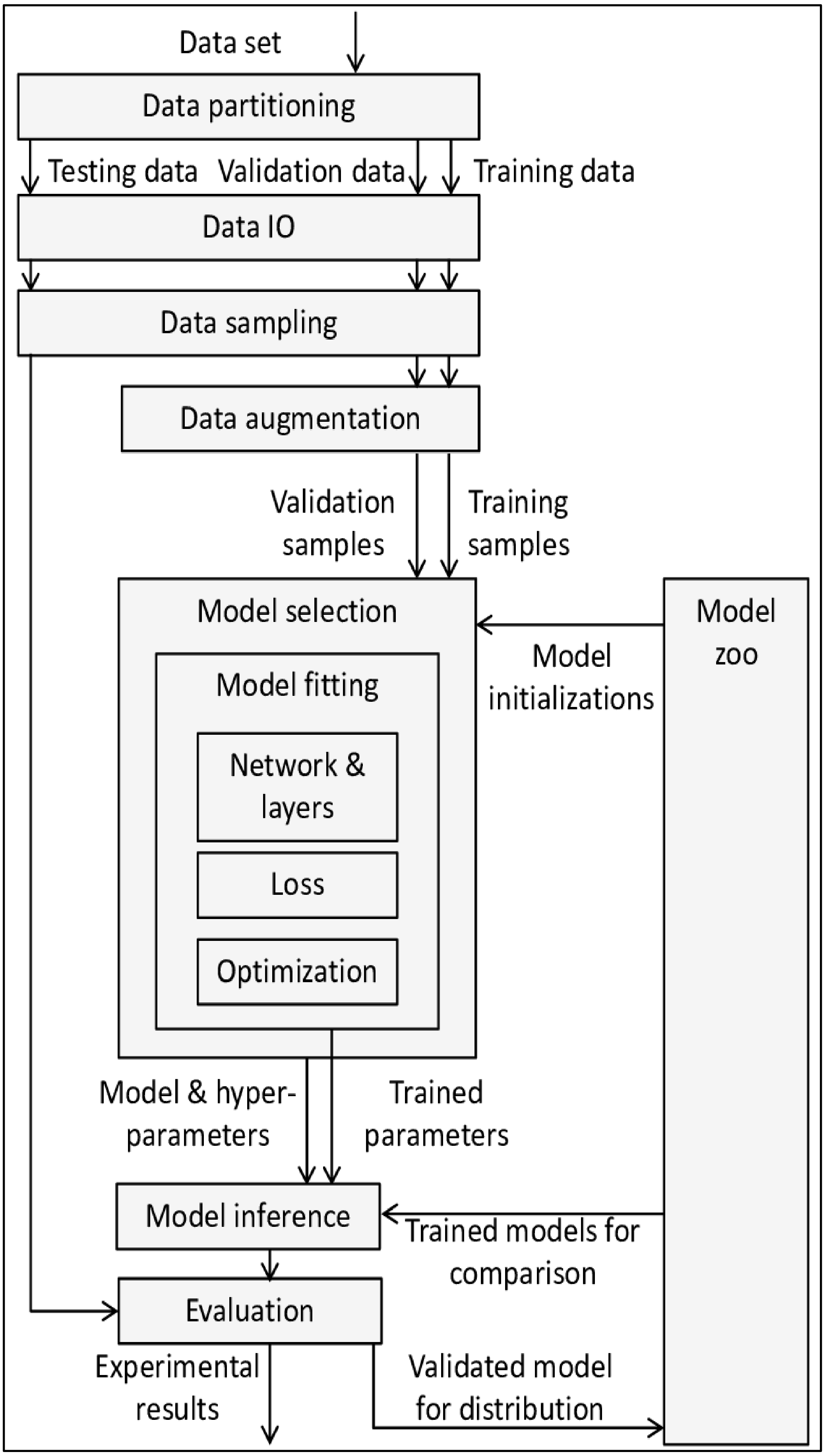


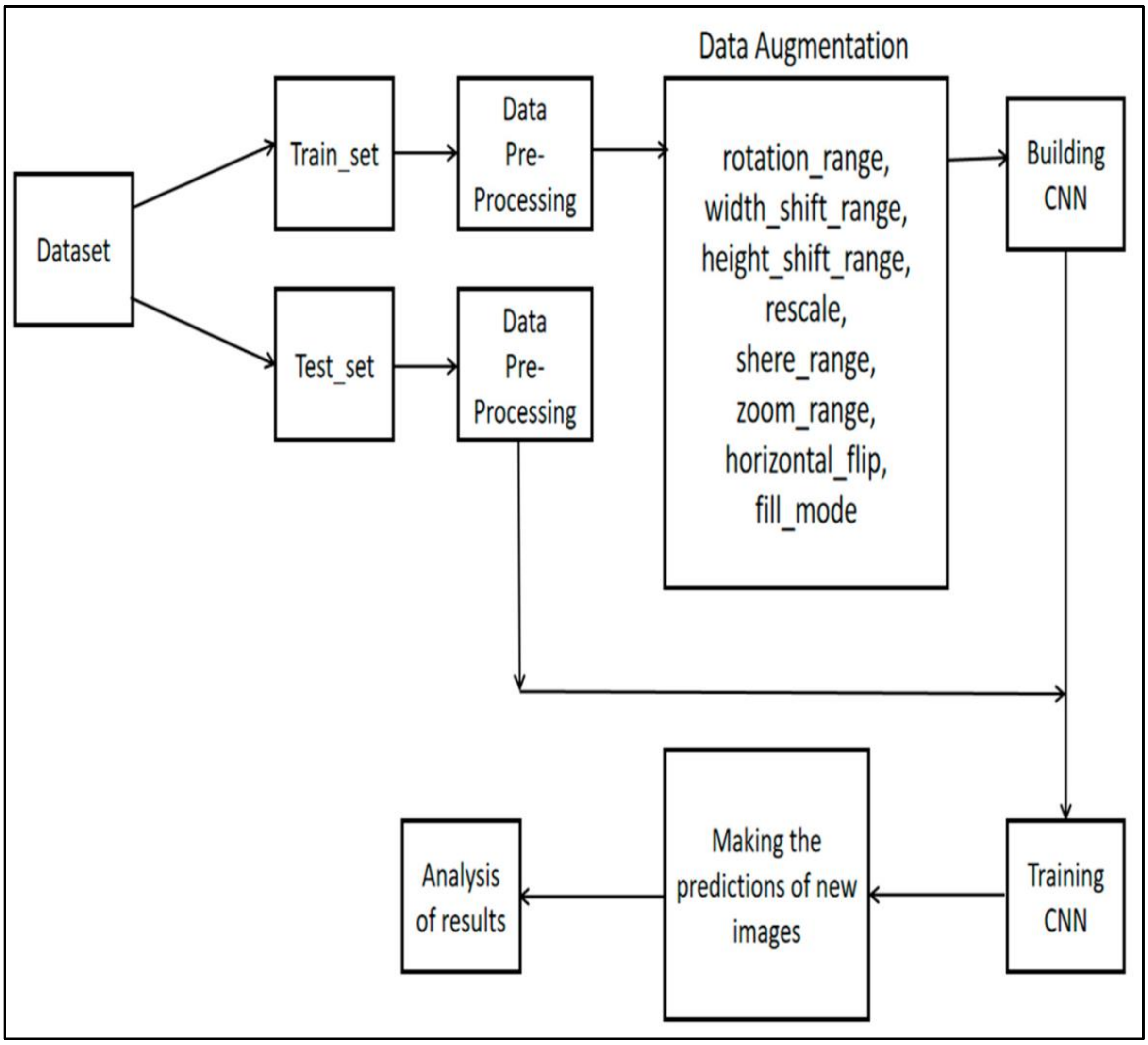
Fig No.- 03(Structure Chart)

5.1.3. DATA FLOW DIAGRAM



<u>Fig No.- 04(Data Flow Diagram)</u>

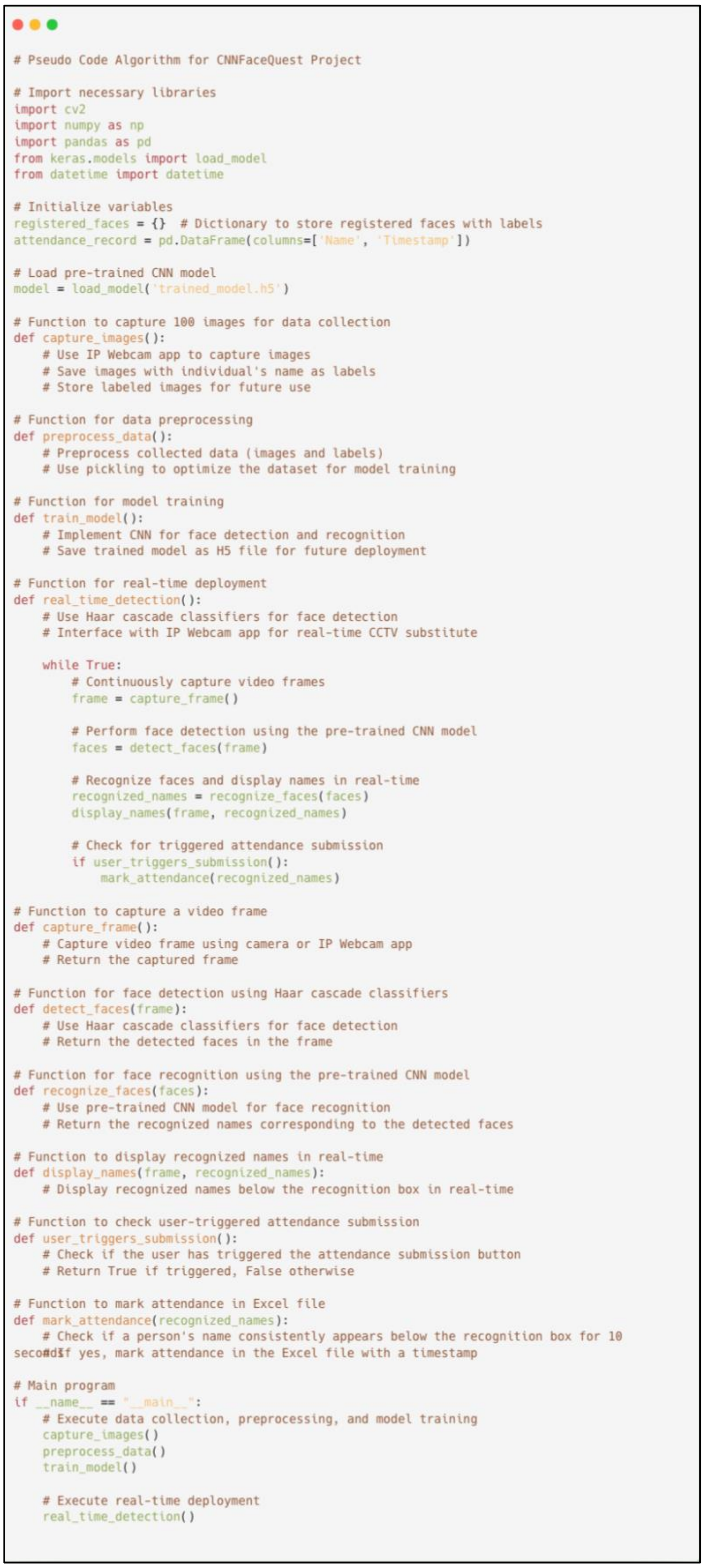
5.1.4. UML


Data Augmentation
Dataset
Train_set
Test_set
Data Pre-Processing
Data Pre-Processing
rotation_range, width_shift_range, height_shift_range, rescale, shere_range, zoom_range, horizontal_flip, fill_mode
Building CNN
Training CNN
Making the predictions of new images
Analysis of results

<u>Fig No.- 05 (UML Class Diagram)</u>

5.2. LOW LEVEL DESIGN

5.2.1. PROCESS SPECIFICATION (PSEUDO CODE)



```
# Pseudo Code Algorithm for CNNFaceQuest Project

# Import necessary libraries
import cv2
import numpy as np
import pandas as pd
from keras.models import load_model
from datetime import datetime

# Initialize variables
registered_faces = {}  # Dictionary to store registered faces with labels
attendance_record = pd.DataFrame(columns=['Name', 'Timestamp'])

# Load pre-trained CNN model
model = load_model('trained_model.h5')

# Function to capture 100 images for data collection
def capture_images():
    # Use IP Webcam app to capture images
    # Save images with individual's name as labels
    # Store labeled images for future use

# Function for data preprocessing
def preprocess_data():
    # Preprocess collected data (images and labels)
    # Use pickling to optimize the dataset for model training

# Function for model training
def train_model():
    # Implement CNN for face detection and recognition
    # Save trained model as H5 file for future deployment

# Function for real-time deployment
def real_time_detection():
    # Use Haar cascade classifiers for face detection
    # Interface with IP Webcam app for real-time CCTV substitute

    while True:
        # Continuously capture video frames
        frame = capture_frame()

        # Perform face detection using the pre-trained CNN model
        faces = detect_faces(frame)

        # Recognize faces and display names in real-time
        recognized_names = recognize_faces(faces)
        display_names(frame, recognized_names)

        # Check for triggered attendance submission
        if user_triggers_submission():
            mark_attendance(recognized_names)

# Function to capture a video frame
def capture_frame():
    # Capture video frame using camera or IP Webcam app
    # Return the captured frame

# Function for face detection using Haar cascade classifiers
def detect_faces(frame):
    # Use Haar cascade classifiers for face detection
    # Return the detected faces in the frame

# Function for face recognition using the pre-trained CNN model
def recognize_faces(faces):
    # Use pre-trained CNN model for face recognition
    # Return the recognized names corresponding to the detected faces

# Function to display recognized names in real-time
def display_names(frame, recognized_names):
    # Display recognized names below the recognition box in real-time

# Function to check user-triggered attendance submission
def user_triggers_submission():
    # Check if the user has triggered the attendance submission button
    # Return True if triggered, False otherwise

# Function to mark attendance in Excel file
def mark_attendance(recognized_names):
    # Check if a person's name consistently appears below the recognition box for 10
seco#dIf yes, mark attendance in the Excel file with a timestamp

# Main program
if __name__ == "__main__":
    # Execute data collection, preprocessing, and model training
    capture_images()
    preprocess_data()
    train_model()

    # Execute real-time deployment
    real_time_detection()
```

Fig No.- 06 (Pseudo Code / Algorithm)

5.2.2. SCREEN-SHOTS

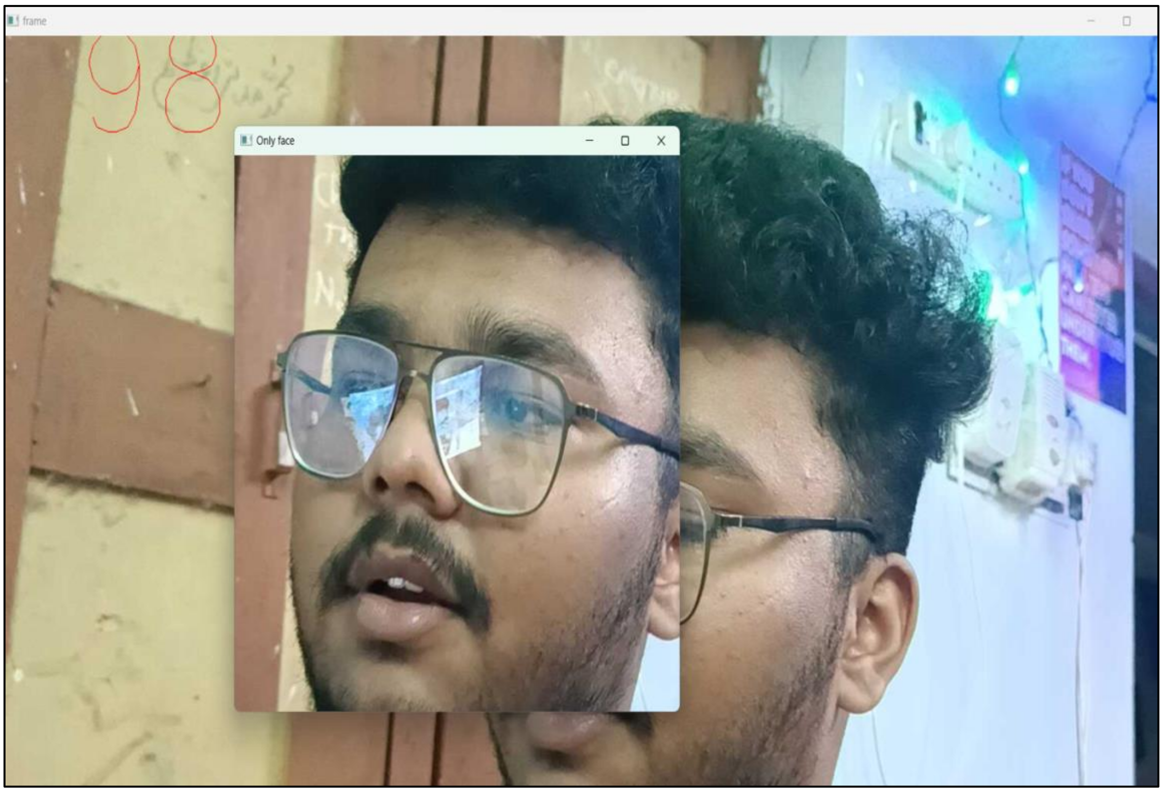


Fig No.- 07 (Person Registration-01)

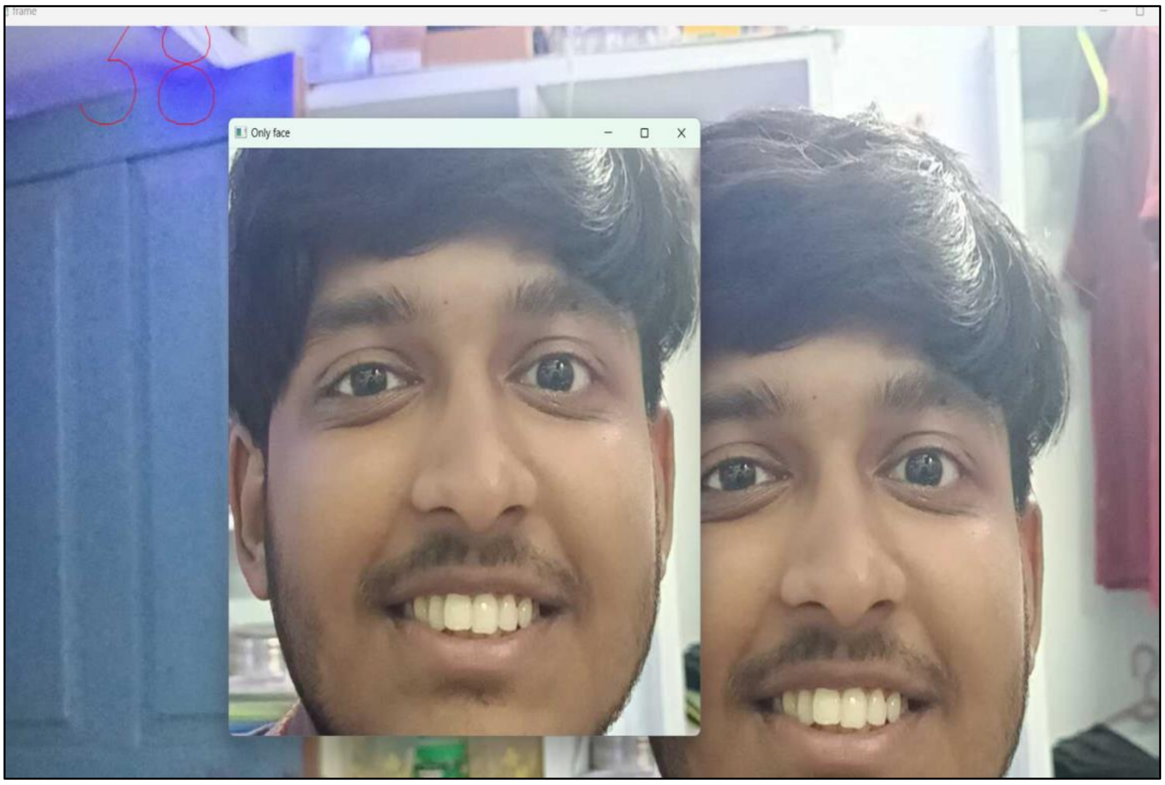


Fig No.- 08 (Person Registration-02)

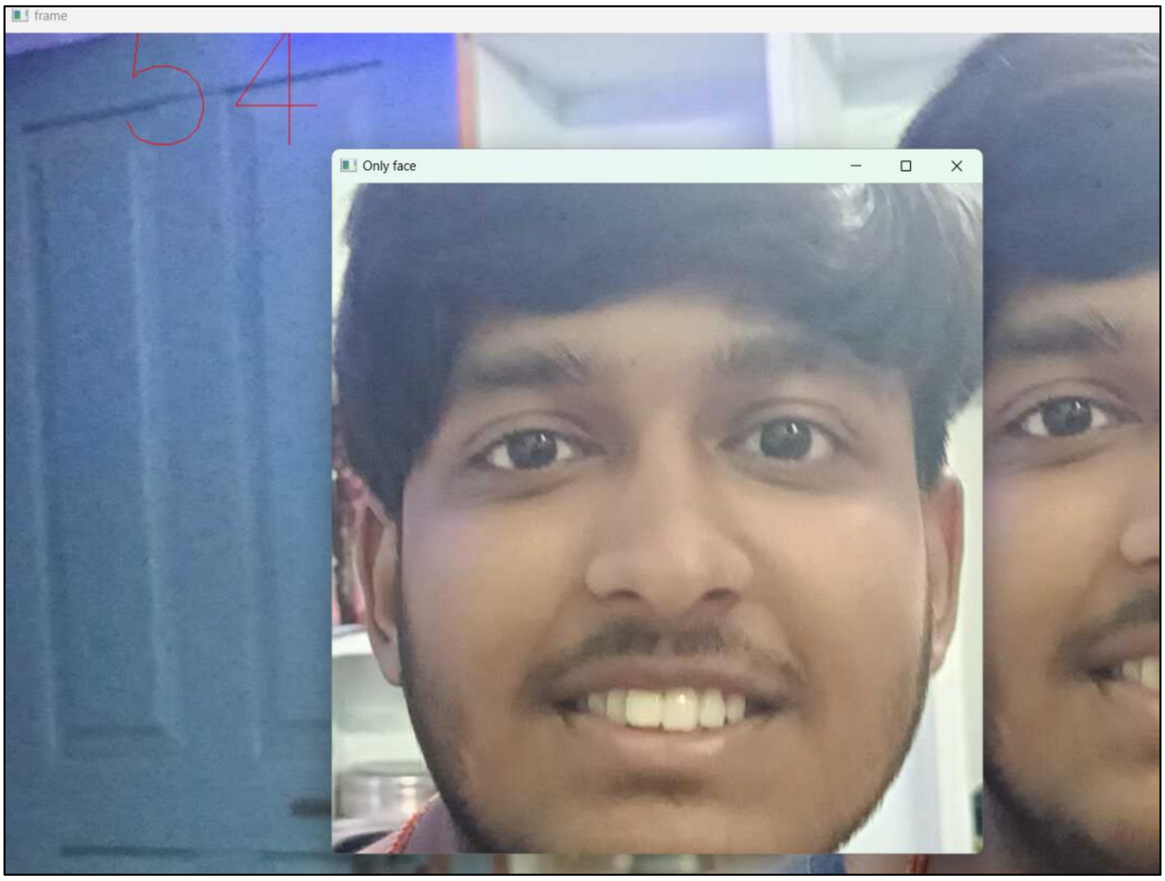


Fig No.- 09 (Person Registration-03)

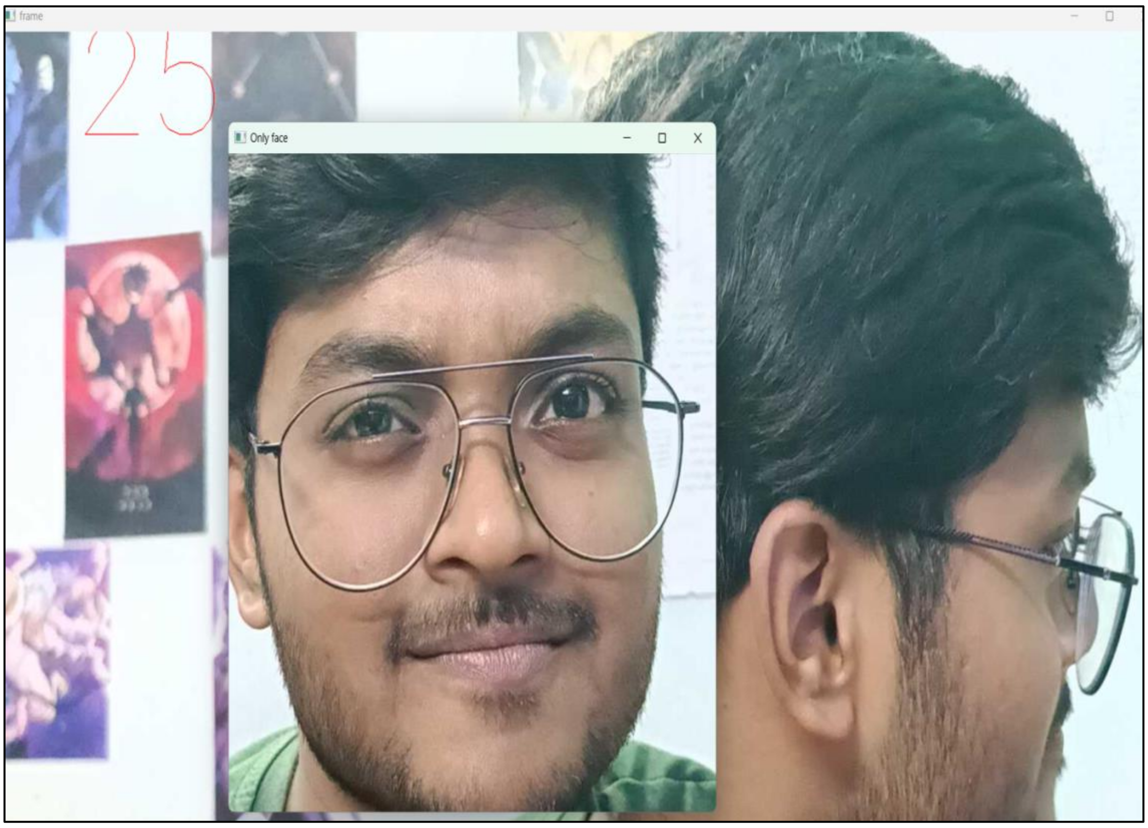
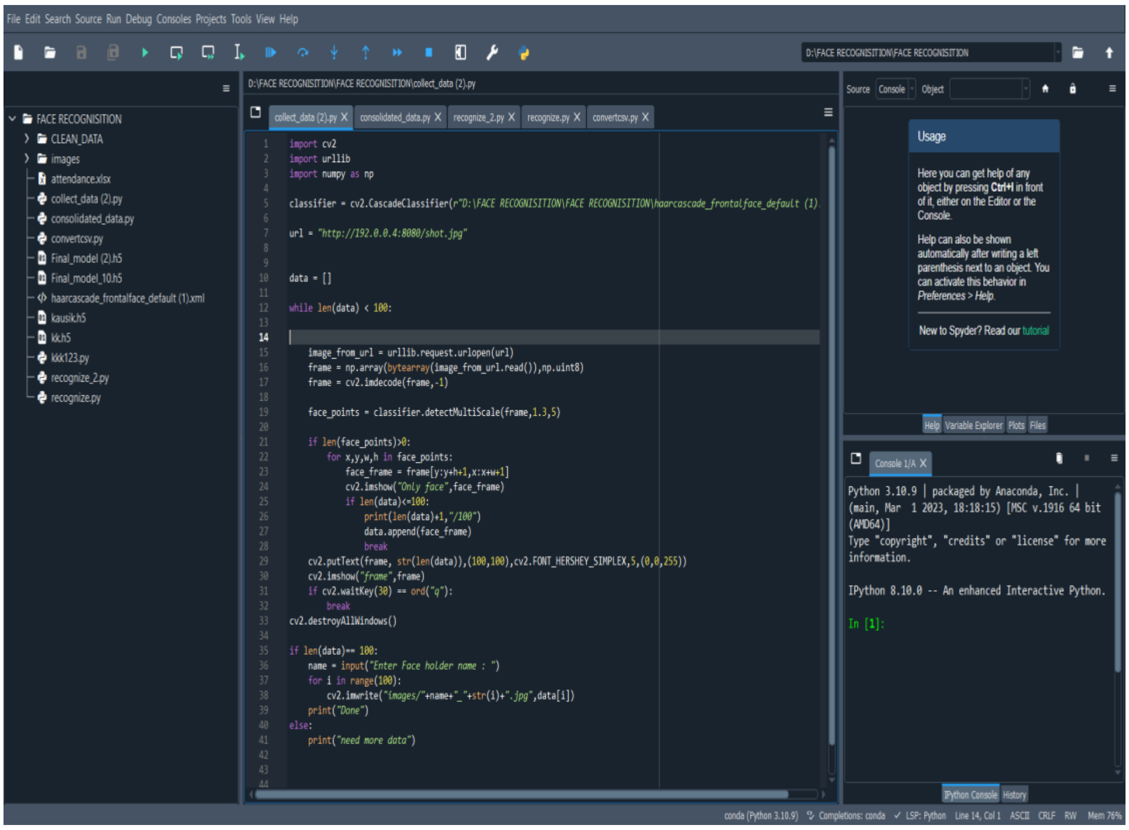


Fig No.- 10 (Person Registration-04)


File Edit Search Source Run Debug Consoles Projects Tools View Help
D:\FACE RECOGNISITION\FACE RECOGNISITION
D:\FACE RECOGNISITION\FACE RECOGNISITION\collect_data (2).py
collect_data (2).py
consolidated_data.py
recognize_2.py
recognize.py
convertcsv.py
FACE RECOGNISITION
CLEAN_DATA
images
attendance.xlsx
collect_data (2).py
consolidated_data.py
convertcsv.py
Final_model (2).h5
Final_model_10.h5
haarcascade_frontalface_default (1).xml
kausik.h5
kk.h5
kkk123.py
recognize_2.py
recognize.py
Source Console Object
Help Variable Explorer Plots Files
Console 1/A
IPython Console History
conda (Python 3.10.9) Completions: conda LSP: Python Line 14, Col 1 ASCII CRLF RW Mem 76%

```python
import cv2
import urllib
import numpy as np

classifier = cv2.CascadeClassifier(r"D:\FACE RECOGNISITION\FACE RECOGNISITION\haarcascade_frontalface_default (1).

url = "http://192.0.0.4:8080/shot.jpg"


data = []

while len(data) < 100:


    image_from_url = urllib.request.urlopen(url)
    frame = np.array(bytearray(image_from_url.read()),np.uint8)
    frame = cv2.imdecode(frame,-1)

    face_points = classifier.detectMultiScale(frame,1.3,5)

    if len(face_points)>0:
        for x,y,w,h in face_points:
            face_frame = frame[y:y+h+1,x:x+w+1]
            cv2.imshow("Only face",face_frame)
            if len(data)<=100:
                print(len(data)+1,"/100")
                data.append(face_frame)
                break
    cv2.putText(frame, str(len(data)),(100,100),cv2.FONT_HERSHEY_SIMPLEX,5,(0,0,255))
    cv2.imshow("frame",frame)
    if cv2.waitKey(30) == ord("q"):
        break
cv2.destroyAllWindows()

if len(data)== 100:
    name = input("Enter Face holder name : ")
    for i in range(100):
        cv2.imwrite("images/"+name+"_"+str(i)+".jpg",data[i])
    print("Done")
else:
    print("need more data")
```

Usage

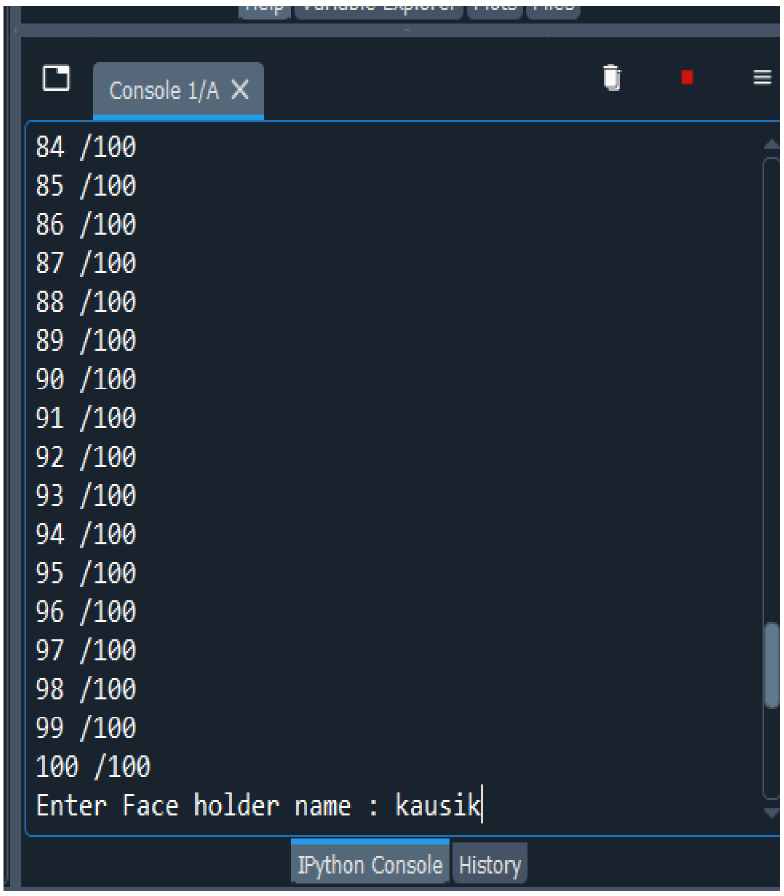
Here you can get help of any object by pressing **Ctrl+I** in front of it, either on the Editor or the Console.

Help can also be shown automatically after writing a left parenthesis next to an object. You can activate this behavior in *Preferences > Help*.

New to Spyder? Read our tutorial

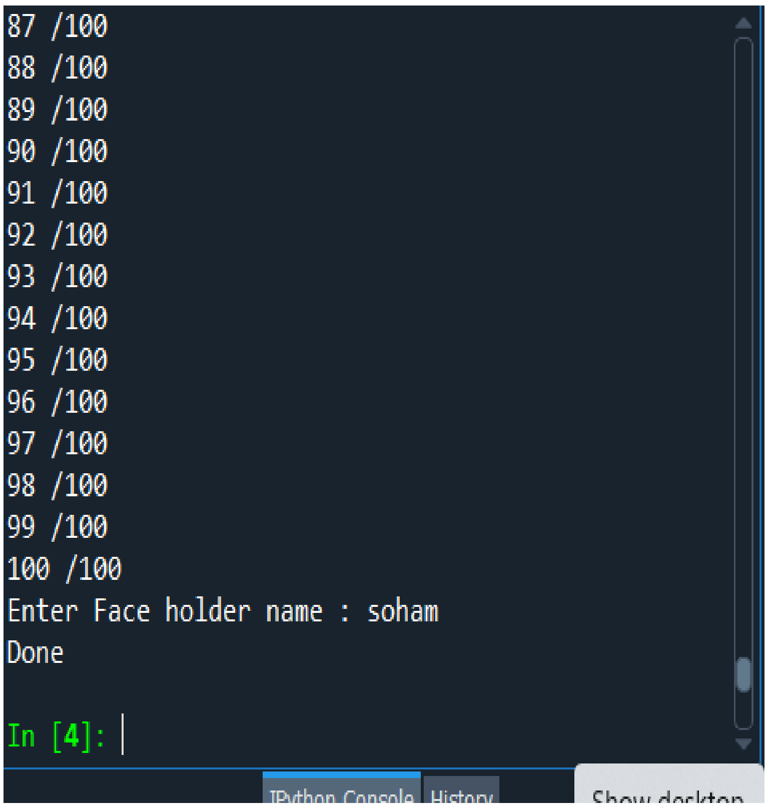
```
Python 3.10.9 | packaged by Anaconda, Inc. |
(main, Mar  1 2023, 18:18:15) [MSC v.1916 64 bit
(AMD64)]
Type "copyright", "credits" or "license" for more
information.

IPython 8.10.0 -- An enhanced Interactive Python.

In [1]:
```

Fig No.- 11 (Person Registration Program)



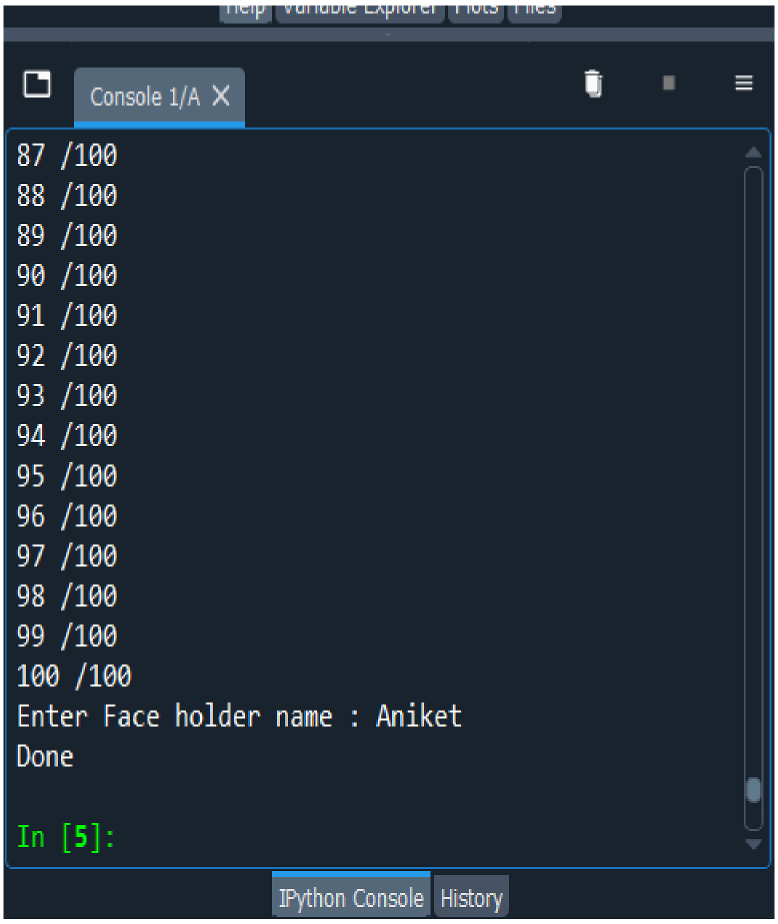
```
84 /100
85 /100
86 /100
87 /100
88 /100
89 /100
90 /100
91 /100
92 /100
93 /100
94 /100
95 /100
96 /100
97 /100
98 /100
99 /100
100 /100
Enter Face holder name : kausik
```

Fig No.- 12 (Person Registration Program Execution-01)



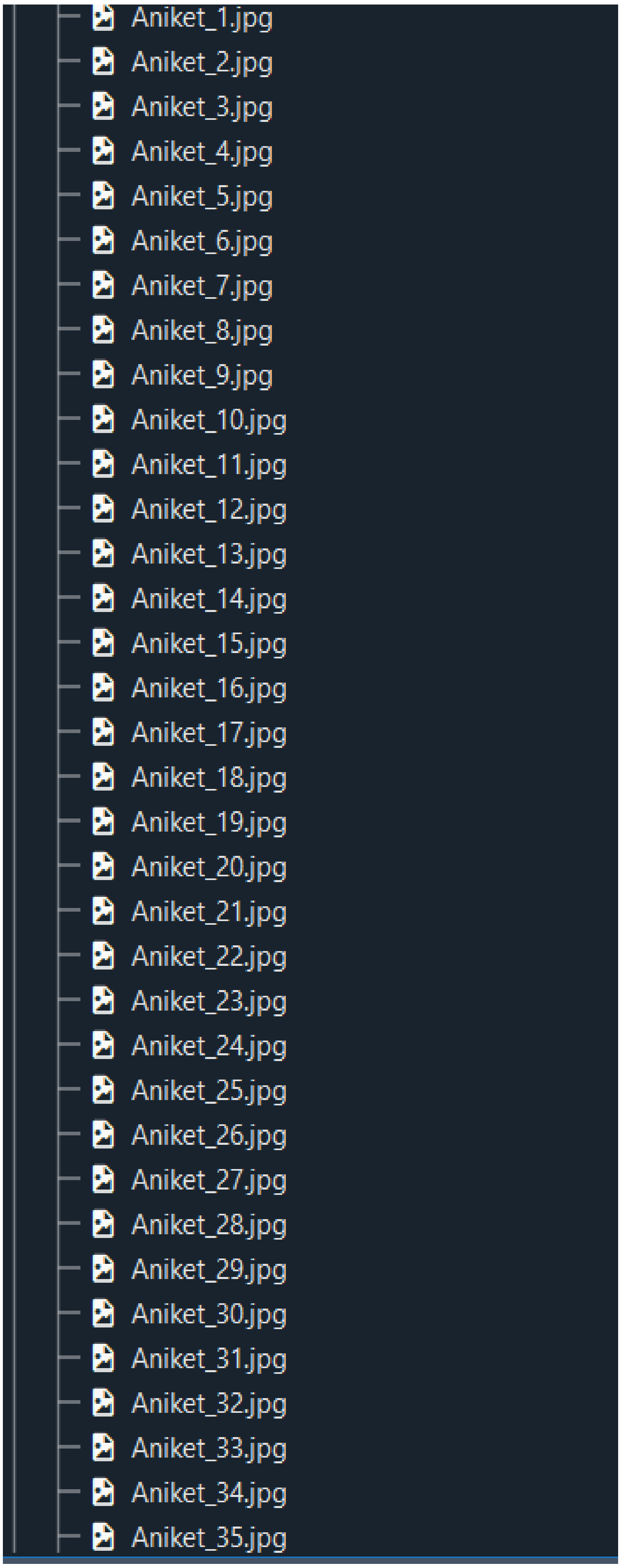
```
87 /100
88 /100
89 /100
90 /100
91 /100
92 /100
93 /100
94 /100
95 /100
96 /100
97 /100
98 /100
99 /100
100 /100
Enter Face holder name : soham
Done

In [4]:
```

Fig No.- 13 (Person Registration Program Execution-02)



```
87 /100
88 /100
89 /100
90 /100
91 /100
92 /100
93 /100
94 /100
95 /100
96 /100
97 /100
98 /100
99 /100
100 /100
Enter Face holder name : Aniket
Done

In [5]:
```

Fig No.- 14 (Person Registration Program Execution-03)



```
Aniket_1.jpg
Aniket_2.jpg
Aniket_3.jpg
Aniket_4.jpg
Aniket_5.jpg
Aniket_6.jpg
Aniket_7.jpg
Aniket_8.jpg
Aniket_9.jpg
Aniket_10.jpg
Aniket_11.jpg
Aniket_12.jpg
Aniket_13.jpg
Aniket_14.jpg
Aniket_15.jpg
Aniket_16.jpg
Aniket_17.jpg
Aniket_18.jpg
Aniket_19.jpg
Aniket_20.jpg
Aniket_21.jpg
Aniket_22.jpg
Aniket_23.jpg
Aniket_24.jpg
Aniket_25.jpg
Aniket_26.jpg
Aniket_27.jpg
Aniket_28.jpg
Aniket_29.jpg
Aniket_30.jpg
Aniket_31.jpg
Aniket_32.jpg
Aniket_33.jpg
Aniket_34.jpg
Aniket_35.jpg
```

Fig No.- 15 (Image Storing in the Spyder Environment)

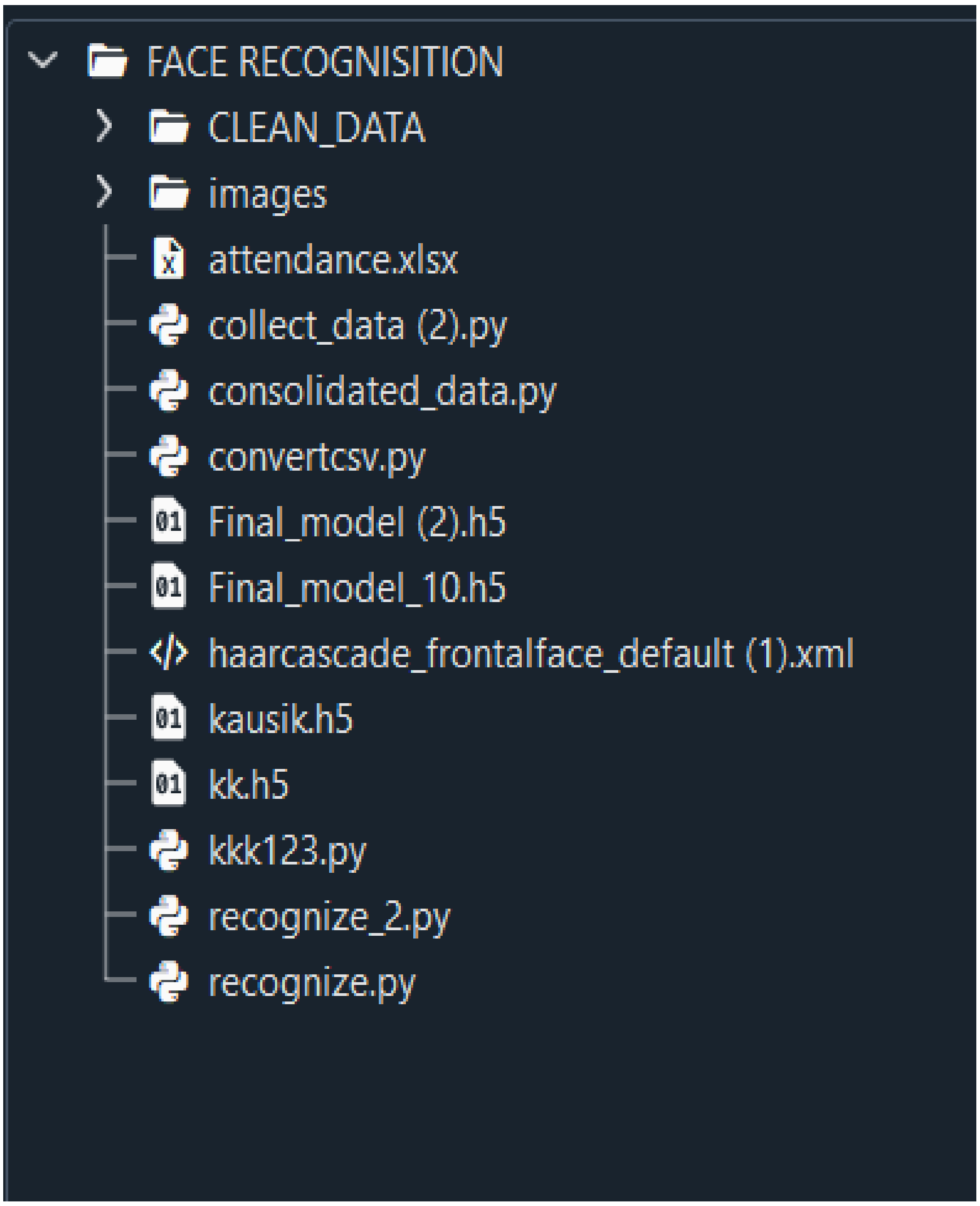

```
FACE RECOGNISITION
  CLEAN_DATA
  images
  attendance.xlsx
  collect_data (2).py
  consolidated_data.py
  convertcsv.py
  Final_model (2).h5
  Final_model_10.h5
  haarcascade_frontalface_default (1).xml
  kausik.h5
  kk.h5
  kkk123.py
  recognize_2.py
  recognize.py
```

Fig No.- 16 (File Organisation in Spyder Environment)

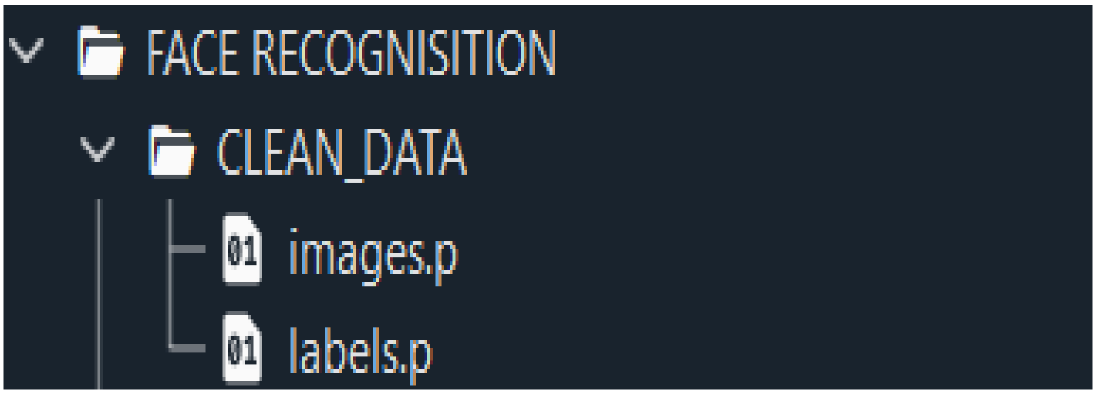
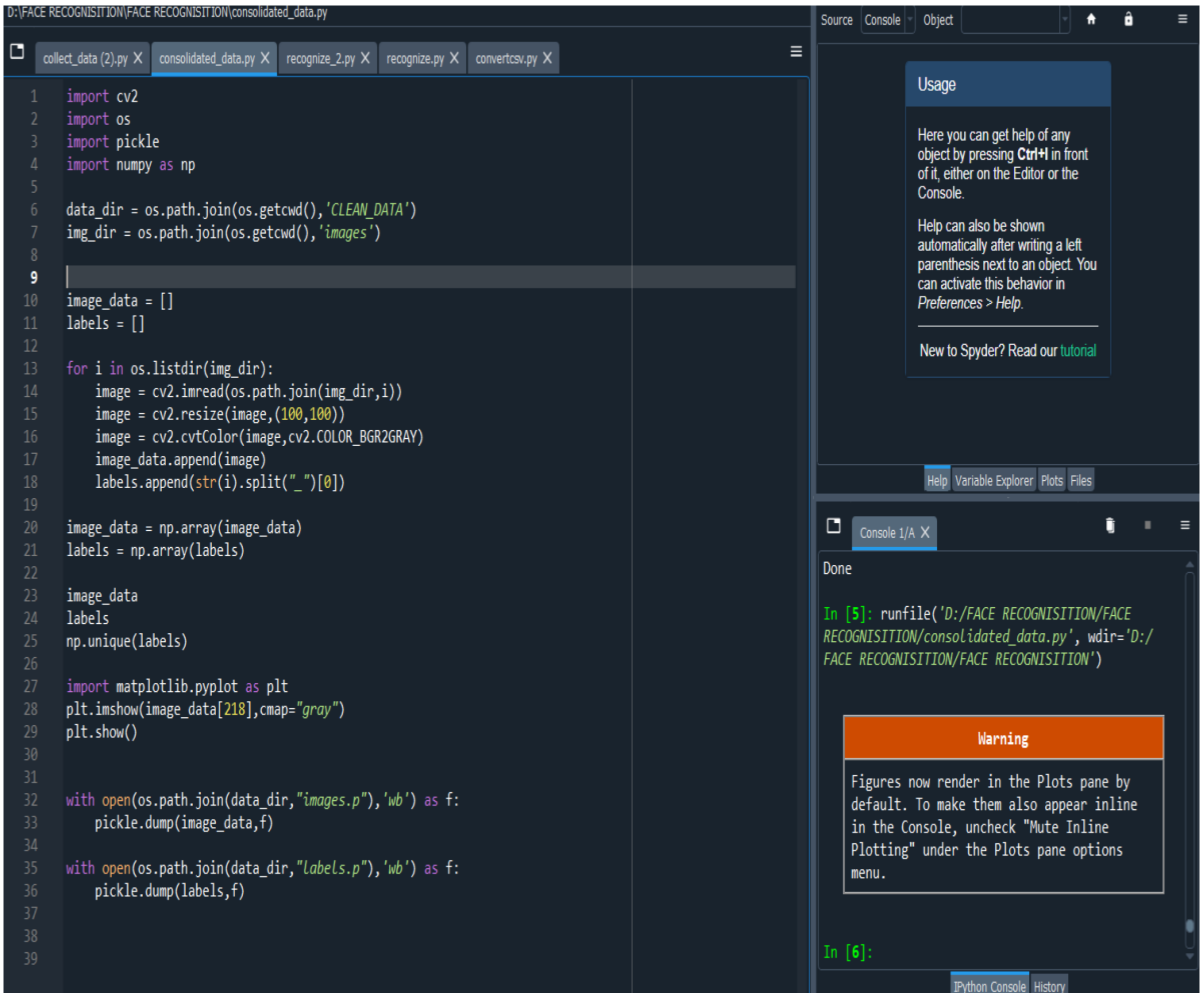

```
FACE RECOGNISITION
  CLEAN_DATA
    images.p
    labels.p
```

Fig No.- 17 (Image and labels' (.P) File)



```python
import cv2
import os
import pickle
import numpy as np

data_dir = os.path.join(os.getcwd(),'CLEAN_DATA')
img_dir = os.path.join(os.getcwd(),'images')

image_data = []
labels = []

for i in os.listdir(img_dir):
    image = cv2.imread(os.path.join(img_dir,i))
    image = cv2.resize(image,(100,100))
    image = cv2.cvtColor(image,cv2.COLOR_BGR2GRAY)
    image_data.append(image)
    labels.append(str(i).split("_")[0])

image_data = np.array(image_data)
labels = np.array(labels)

image_data
labels
np.unique(labels)

import matplotlib.pyplot as plt
plt.imshow(image_data[218],cmap="gray")
plt.show()


with open(os.path.join(data_dir,"images.p"),'wb') as f:
    pickle.dump(image_data,f)

with open(os.path.join(data_dir,"labels.p"),'wb') as f:
    pickle.dump(labels,f)
```

Usage

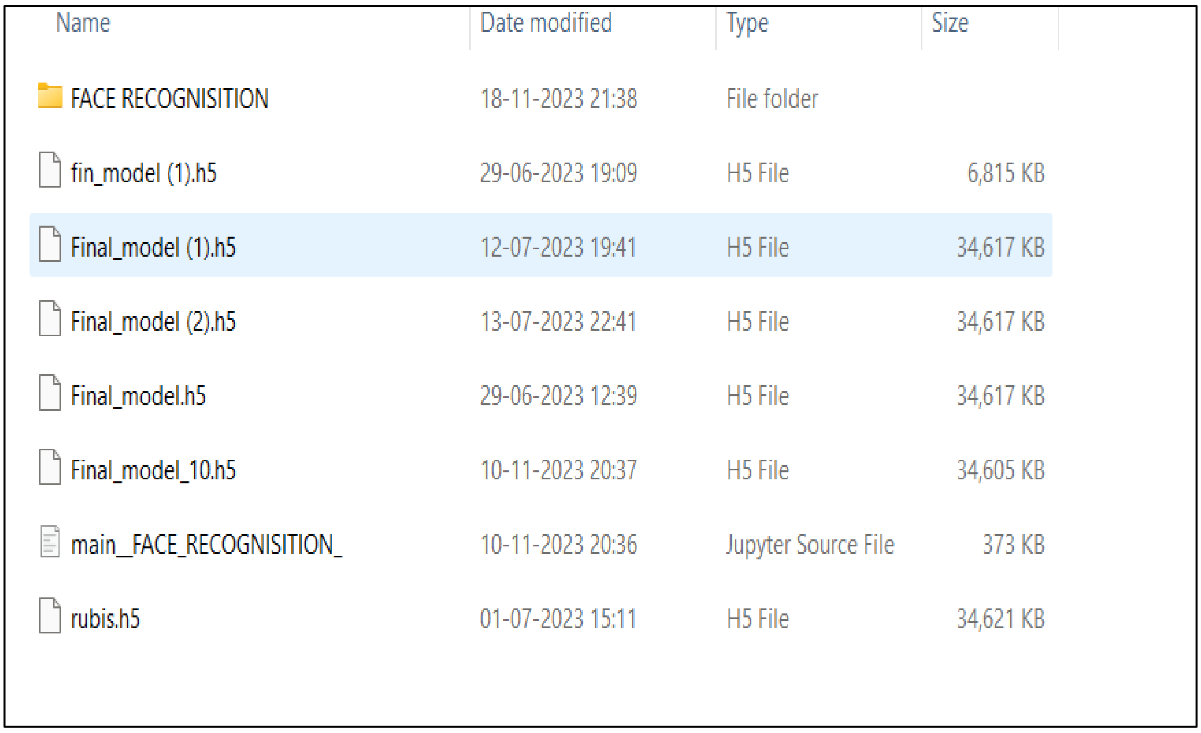
Here you can get help of any object by pressing **Ctrl+I** in front of it, either on the Editor or the Console.

Help can also be shown automatically after writing a left parenthesis next to an object. You can activate this behavior in *Preferences > Help*.

New to Spyder? Read our tutorial

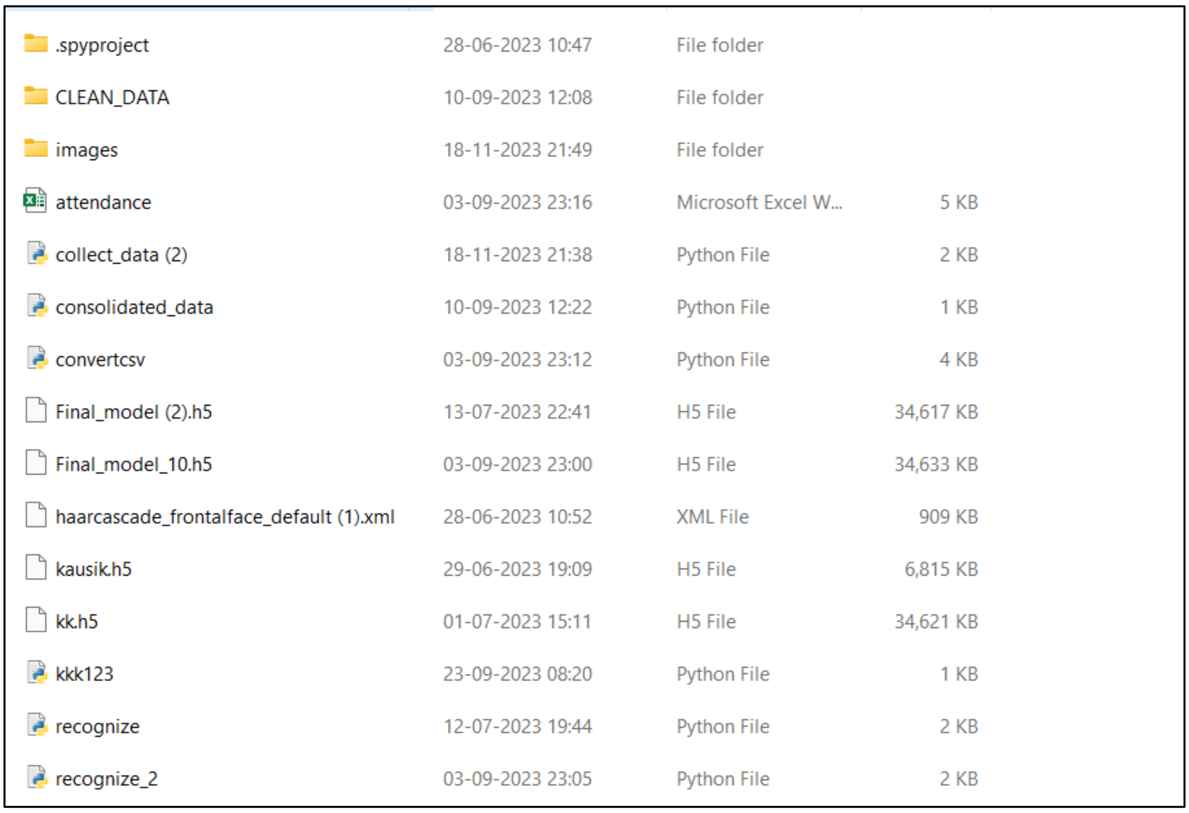
```
Done

In [5]: runfile('D:/FACE RECOGNISITION/FACE RECOGNISITION/consolidated_data.py', wdir='D:/FACE RECOGNISITION/FACE RECOGNISITION')

Warning

Figures now render in the Plots pane by default. To make them also appear inline in the Console, uncheck "Mute Inline Plotting" under the Plots pane options menu.

In [6]:
```

Fig No.- 18 (Program for Pickling)



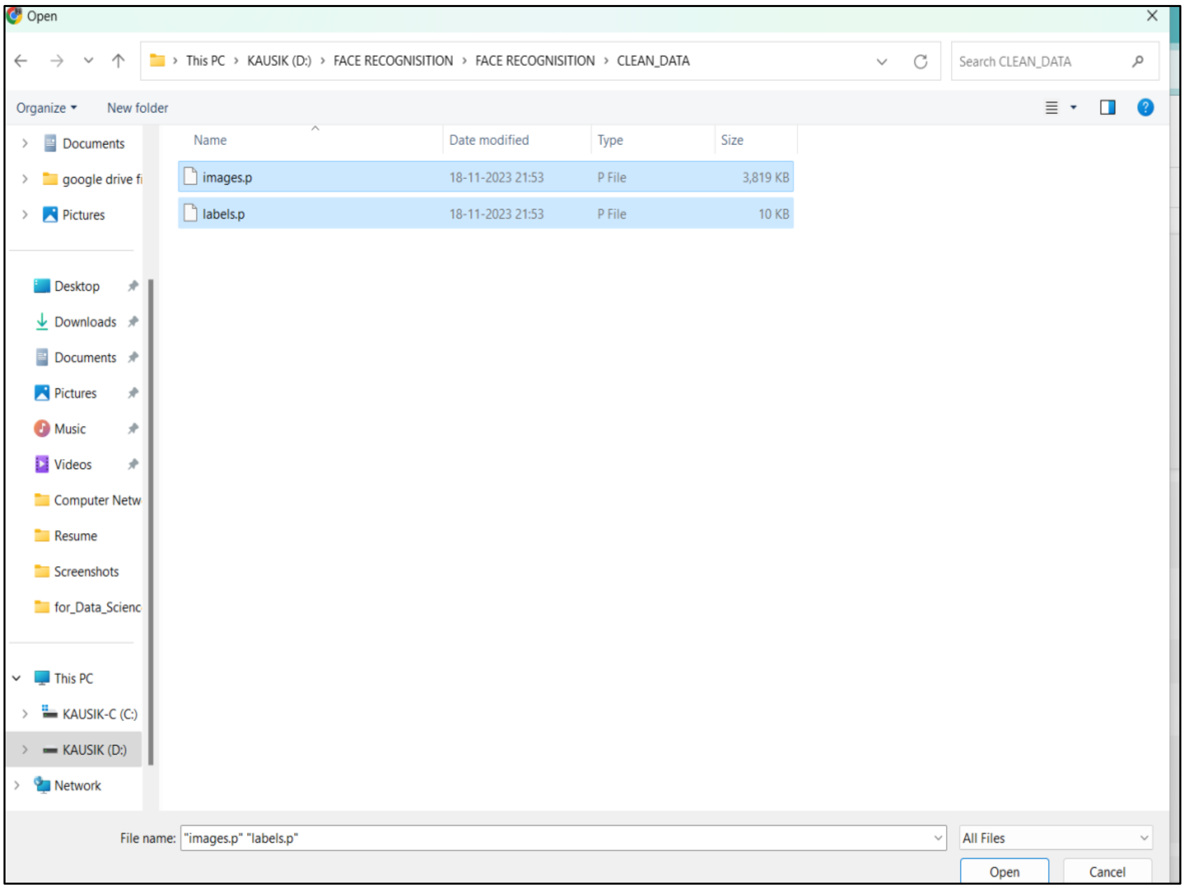
Name	Date modified	Type	Size
FACE RECOGNISITION	18-11-2023 21:38	File folder	
fin_model (1).h5	29-06-2023 19:09	H5 File	6,815 KB
Final_model (1).h5	12-07-2023 19:41	H5 File	34,617 KB
Final_model (2).h5	13-07-2023 22:41	H5 File	34,617 KB
Final_model.h5	29-06-2023 12:39	H5 File	34,617 KB
Final_model_10.h5	10-11-2023 20:37	H5 File	34,605 KB
main__FACE_RECOGNISITION_	10-11-2023 20:36	Jupyter Source File	373 KB
rubis.h5	01-07-2023 15:11	H5 File	34,621 KB

Fig No.- 19 (File Organisation in Local System-01)



.spyproject	28-06-2023 10:47	File folder	
CLEAN_DATA	10-09-2023 12:08	File folder	
images	18-11-2023 21:49	File folder	
attendance	03-09-2023 23:16	Microsoft Excel W...	5 KB
collect_data (2)	18-11-2023 21:38	Python File	2 KB
consolidated_data	10-09-2023 12:22	Python File	1 KB
convertcsv	03-09-2023 23:12	Python File	4 KB
Final_model (2).h5	13-07-2023 22:41	H5 File	34,617 KB
Final_model_10.h5	03-09-2023 23:00	H5 File	34,633 KB
haarcascade_frontalface_default (1).xml	28-06-2023 10:52	XML File	909 KB
kausik.h5	29-06-2023 19:09	H5 File	6,815 KB
kk.h5	01-07-2023 15:11	H5 File	34,621 KB
kkk123	23-09-2023 08:20	Python File	1 KB
recognize	12-07-2023 19:44	Python File	2 KB
recognize_2	03-09-2023 23:05	Python File	2 KB

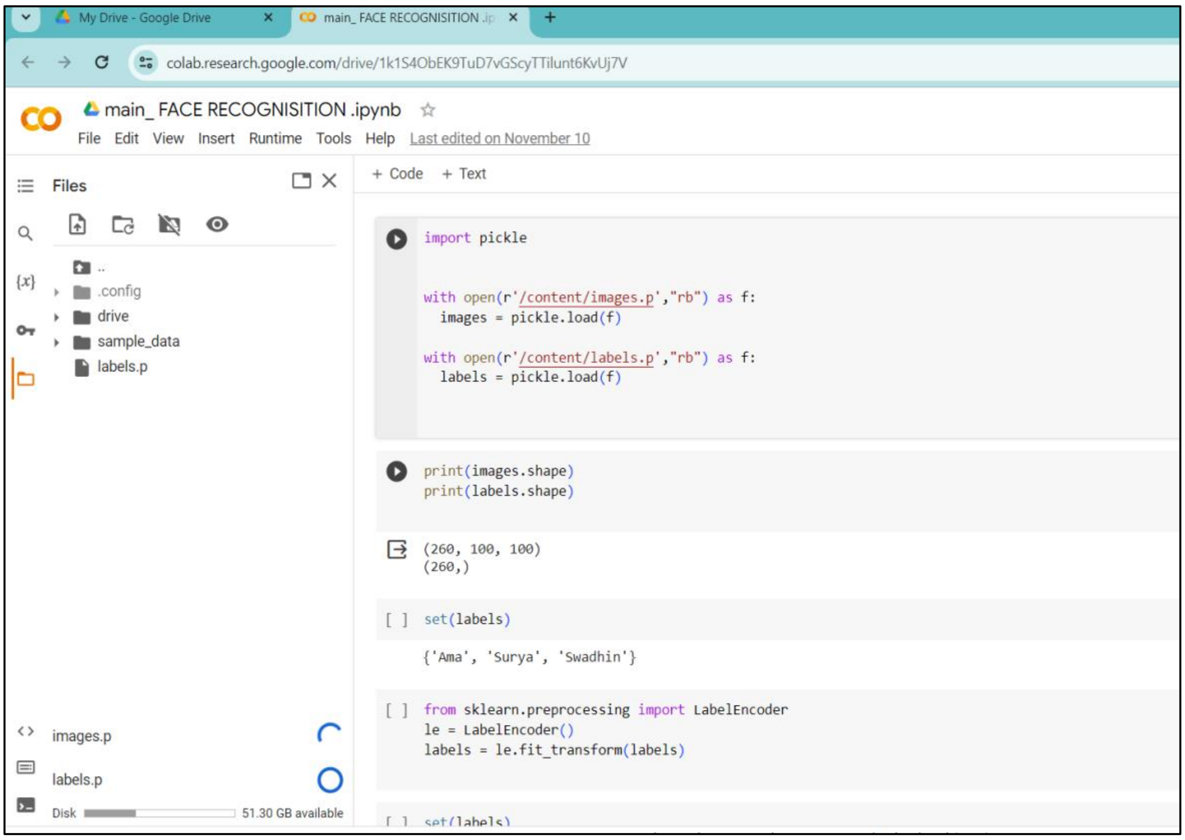
Fig No.- 20 (File Organisation in Local System-02)



Name	Date modified	Type	Size
images.p	18-11-2023 21:53	P File	3,819 KB
labels.p	18-11-2023 21:53	P File	10 KB

File name: "images.p" "labels.p"   All Files

Fig No.- 21 (Storing Structure of Pickled Files in Local System)


My Drive - Google Drive
main_ FACE RECOGNISITION .ip
colab.research.google.com/drive/1k1S4ObEK9TuD7vGScyTTilunt6KvUj7V
main_ FACE RECOGNISITION .ipynb
File Edit View Insert Runtime Tools Help Last edited on November 10
Files
+ Code + Text
..
.config
drive
sample_data
labels.p
images.p
labels.p
Disk 51.30 GB available

```
import pickle

with open(r'/content/images.p',"rb") as f:
  images = pickle.load(f)

with open(r'/content/labels.p',"rb") as f:
  labels = pickle.load(f)
```

```
print(images.shape)
print(labels.shape)
```

```
(260, 100, 100)
(260,)
```

```
set(labels)
```

```
{'Ama', 'Surya', 'Swadhin'}
```

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
labels = le.fit_transform(labels)
```

```
set(labels)
```

Fig No.- 22 (Model Implementation Initialization)

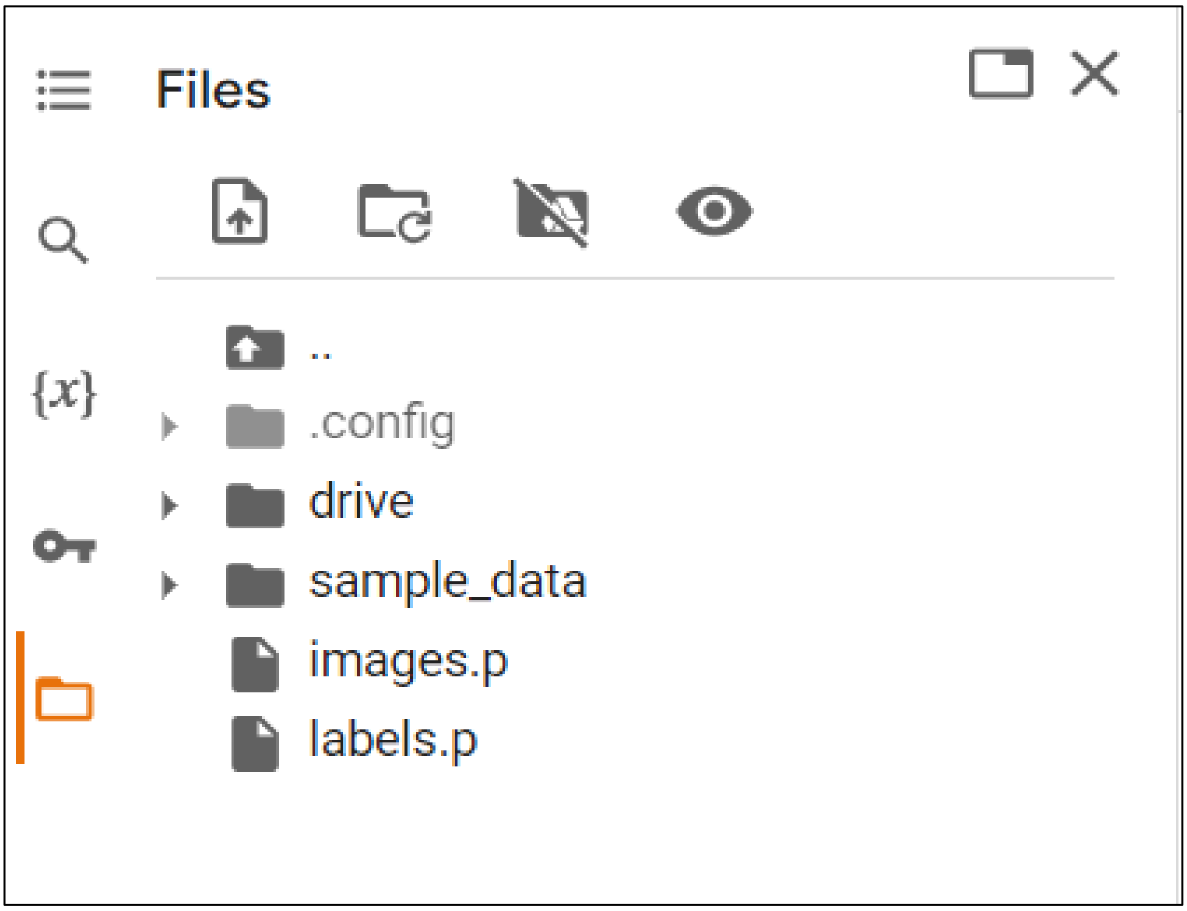
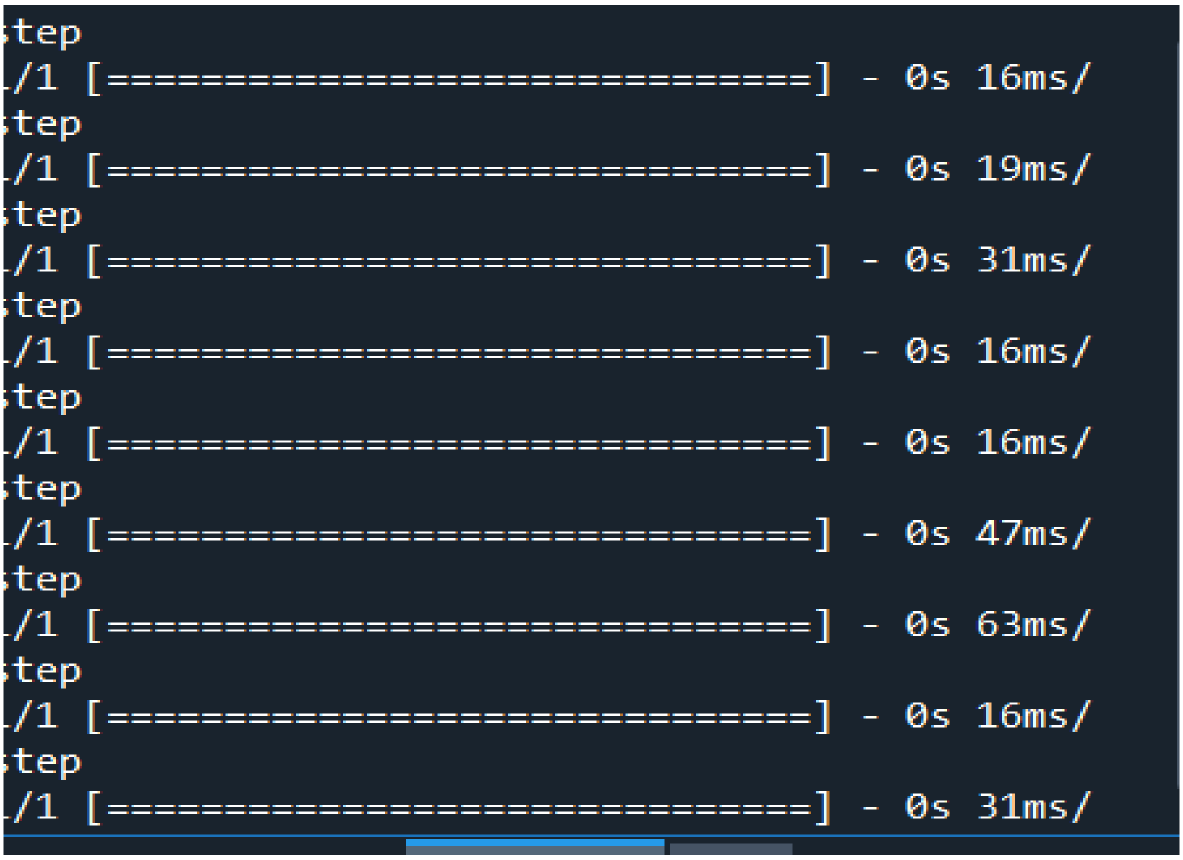
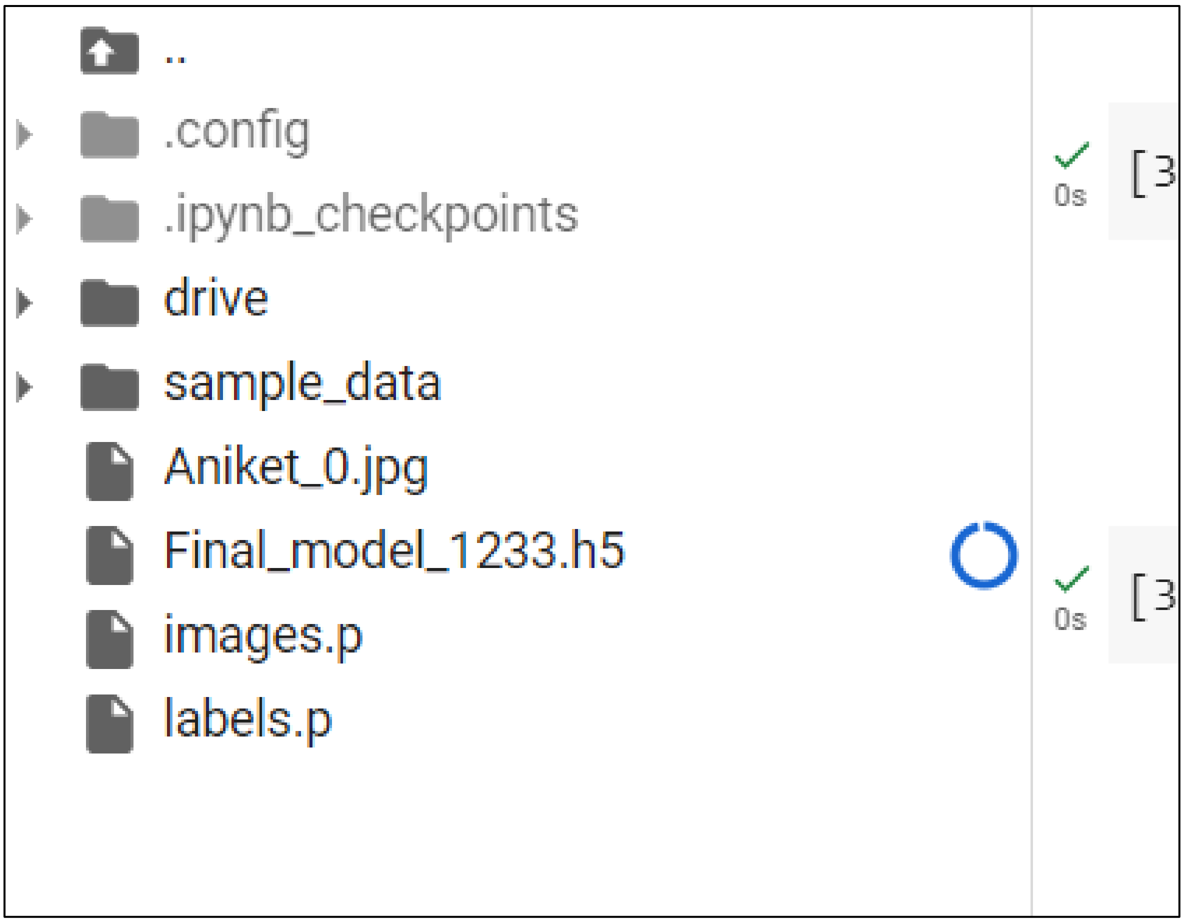


Fig No.- 23 (File Organisation in the Google Collab Environment)



```
tep
/1 [==============================] - 0s 16ms/
tep
/1 [==============================] - 0s 19ms/
tep
/1 [==============================] - 0s 31ms/
tep
/1 [==============================] - 0s 16ms/
tep
/1 [==============================] - 0s 16ms/
tep
/1 [==============================] - 0s 47ms/
tep
/1 [==============================] - 0s 63ms/
tep
/1 [==============================] - 0s 16ms/
tep
/1 [==============================] - 0s 31ms/
```

Fig No.- 24 (Running Stage of the Final Deployment Program)



```
..
.config
.ipynb_checkpoints
drive
sample_data
Aniket_0.jpg
Final_model_1233.h5
images.p
labels.p
```

Fig No.- 25 (Model Download Process)

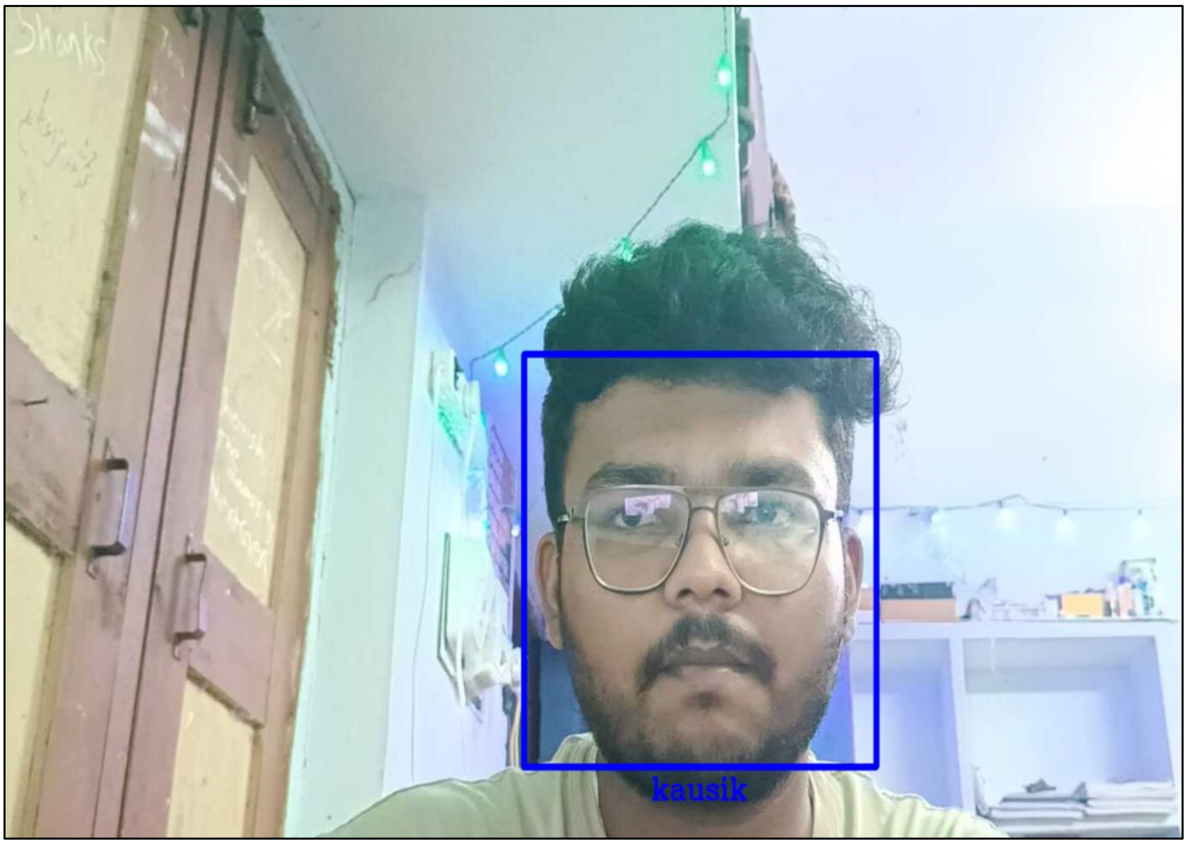


Fig No.- 26 (Model Testing Stage-01)

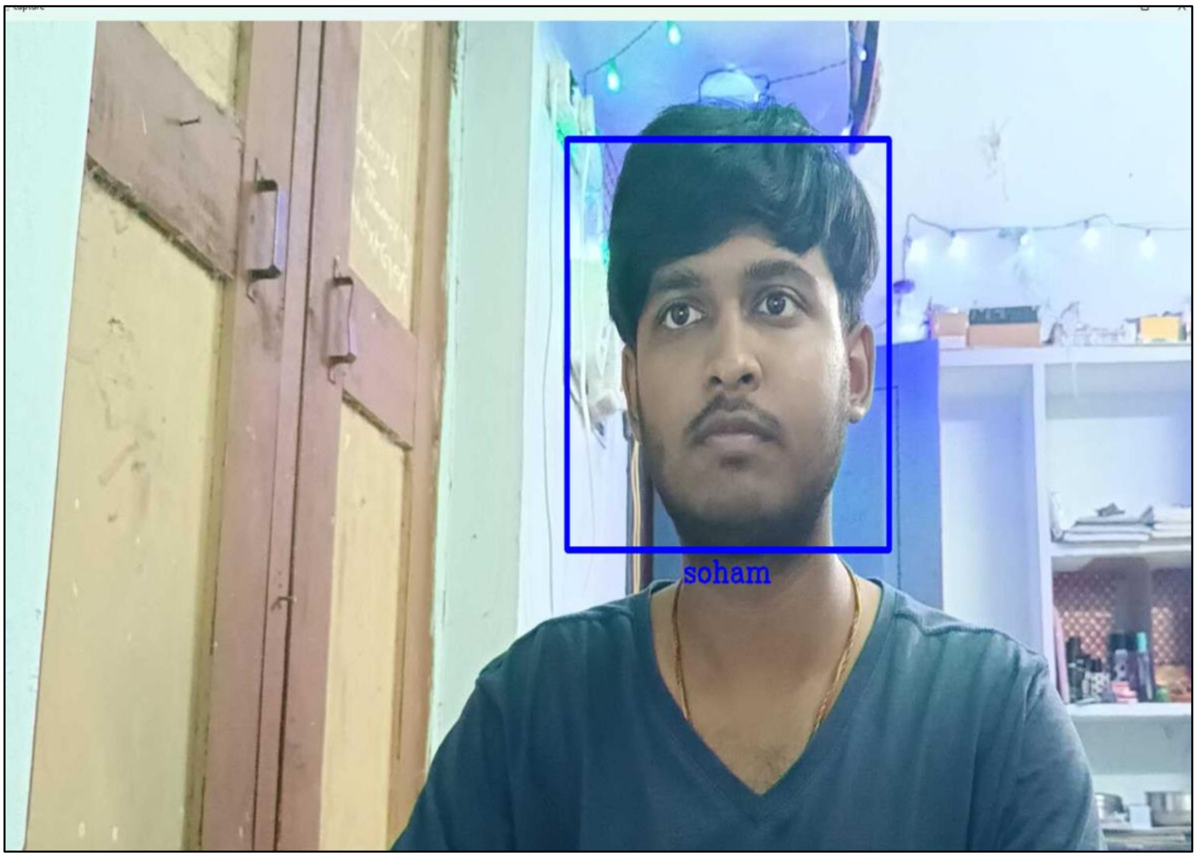


Fig No.- 27 (Model Testing Stage-02)

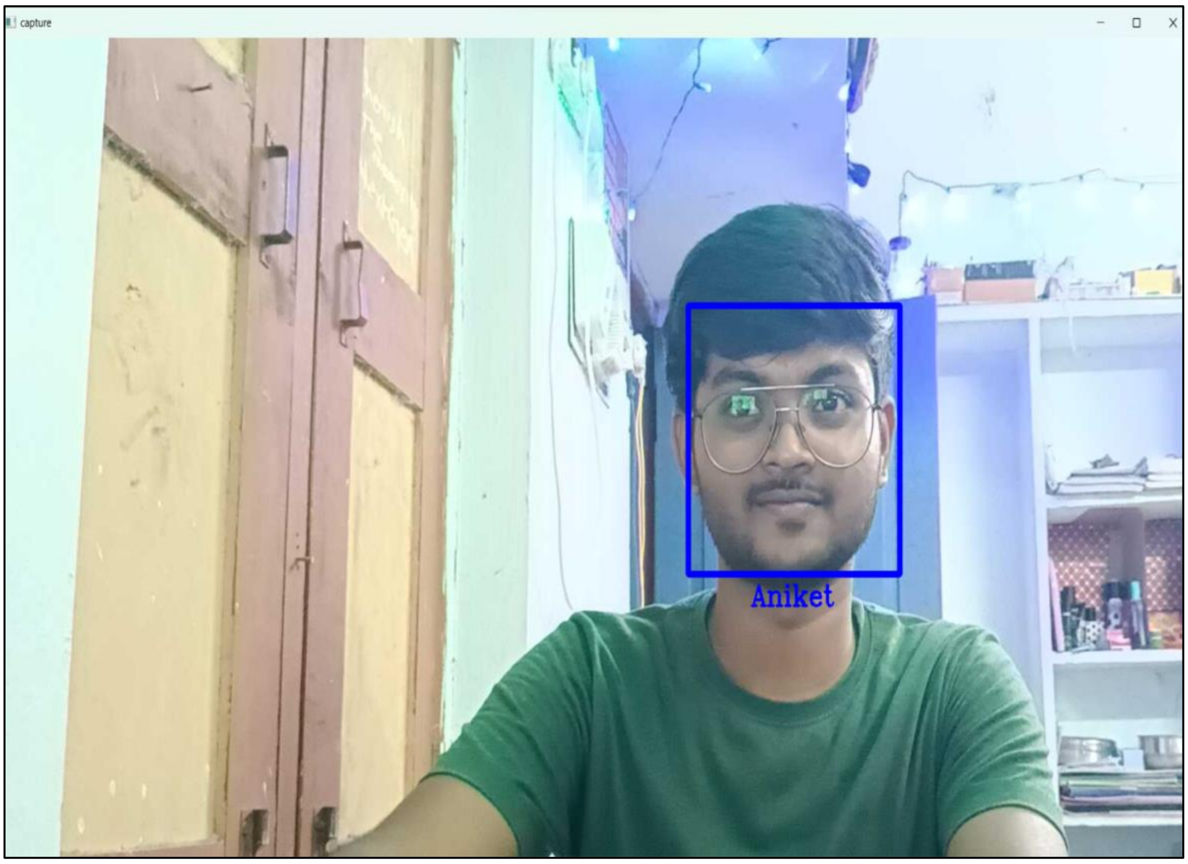


Fig No.- 28 (Model Testing Stage-03)

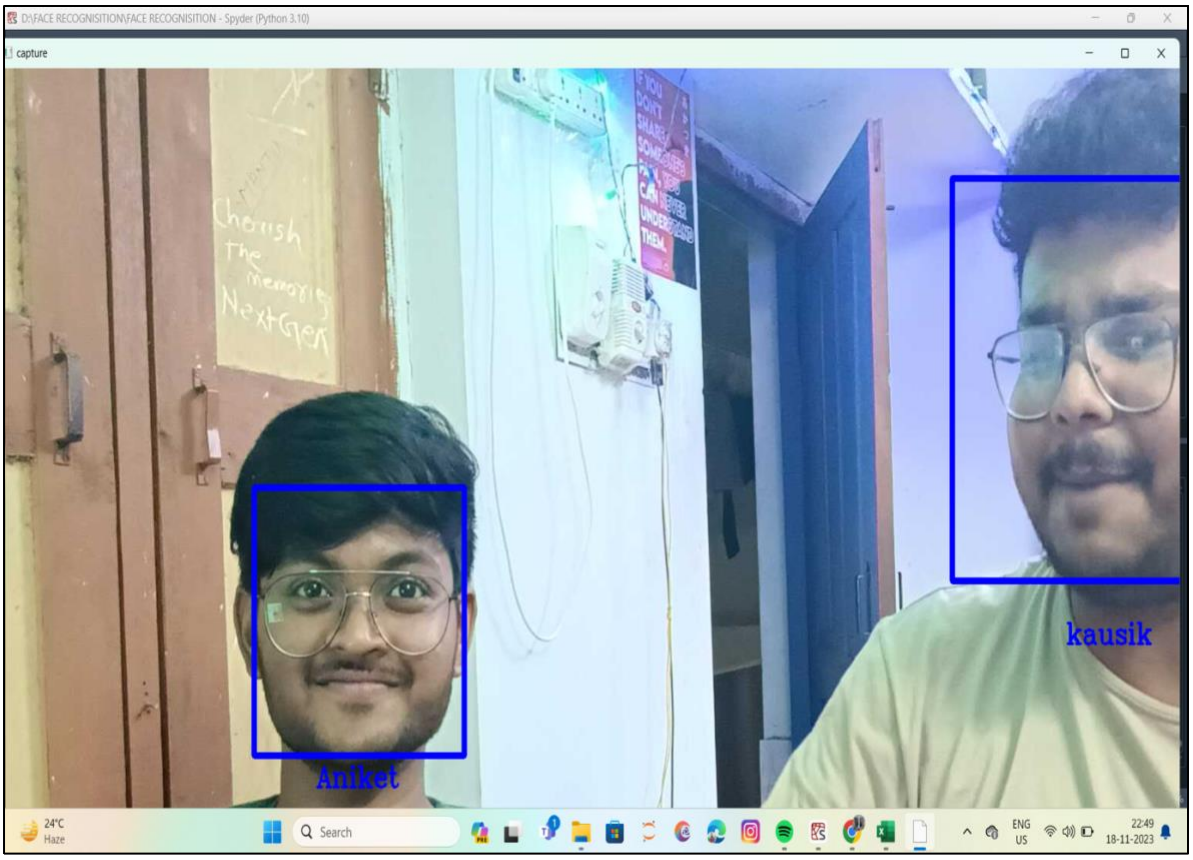

D:\FACE RECOGNISITION\FACE RECOGNISITION - Spyder (Python 3.10)
capture
Aniket
kausik
24°C
Haze
Search
ENG
US
22:49
18-11-2023

Fig No.- 29 (Model Testing Stage-04)

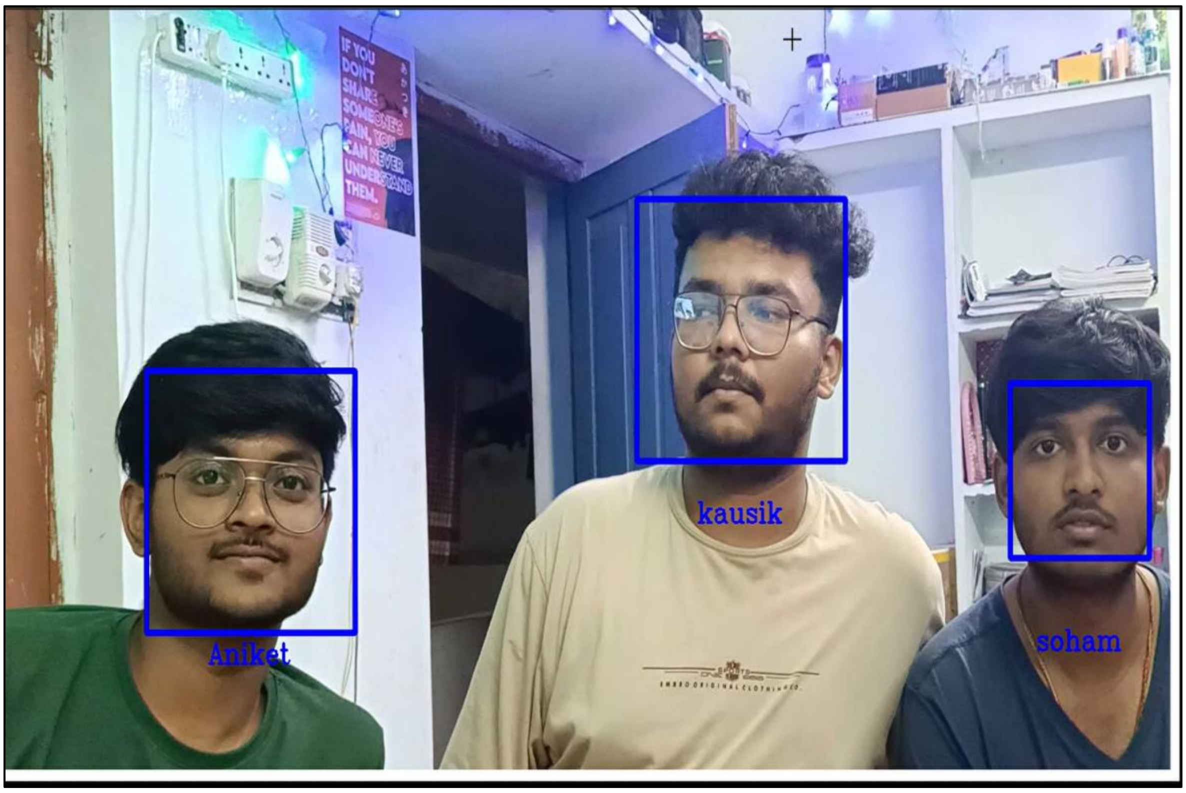
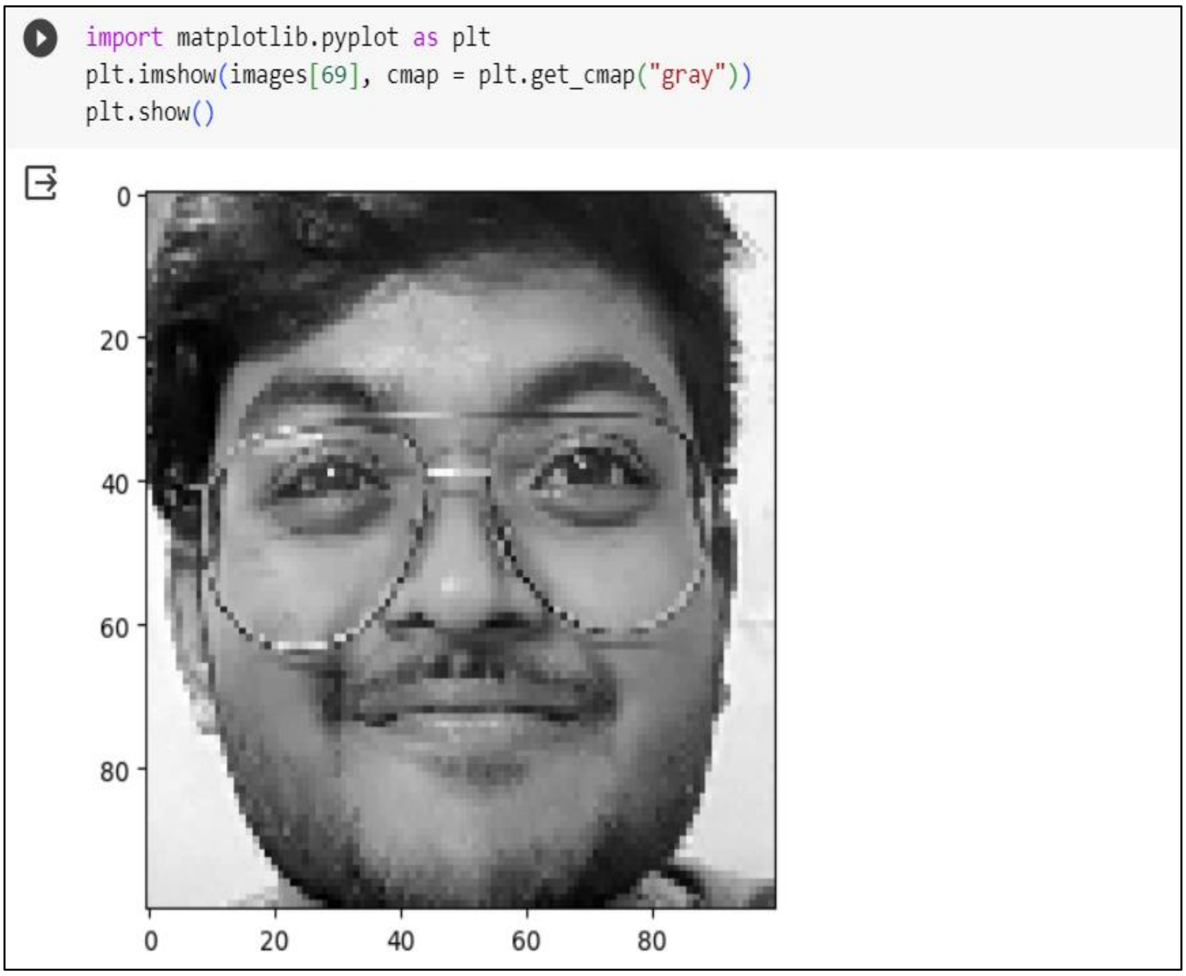
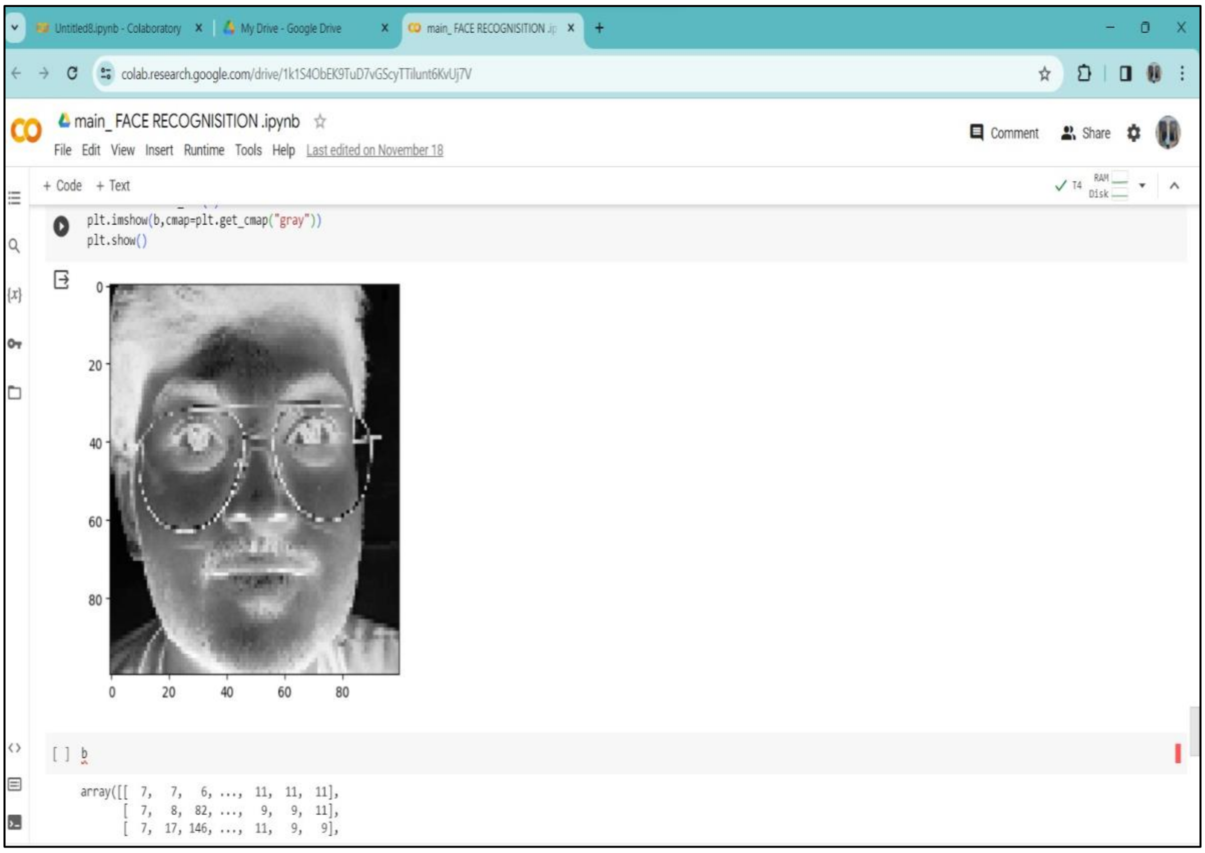

IF YOU DON'T SHARE SOMEONE'S PAIN, YOU CAN NEVER UNDERSTAND THEM.
kausik
Aniket
soham

Fig No.- 30 (Model Testing Stage-05)



```
import matplotlib.pyplot as plt
plt.imshow(images[69], cmap = plt.get_cmap("gray"))
plt.show()
```

Fig No.- 31 (Data Analysis Part While Model Building, Part-01)

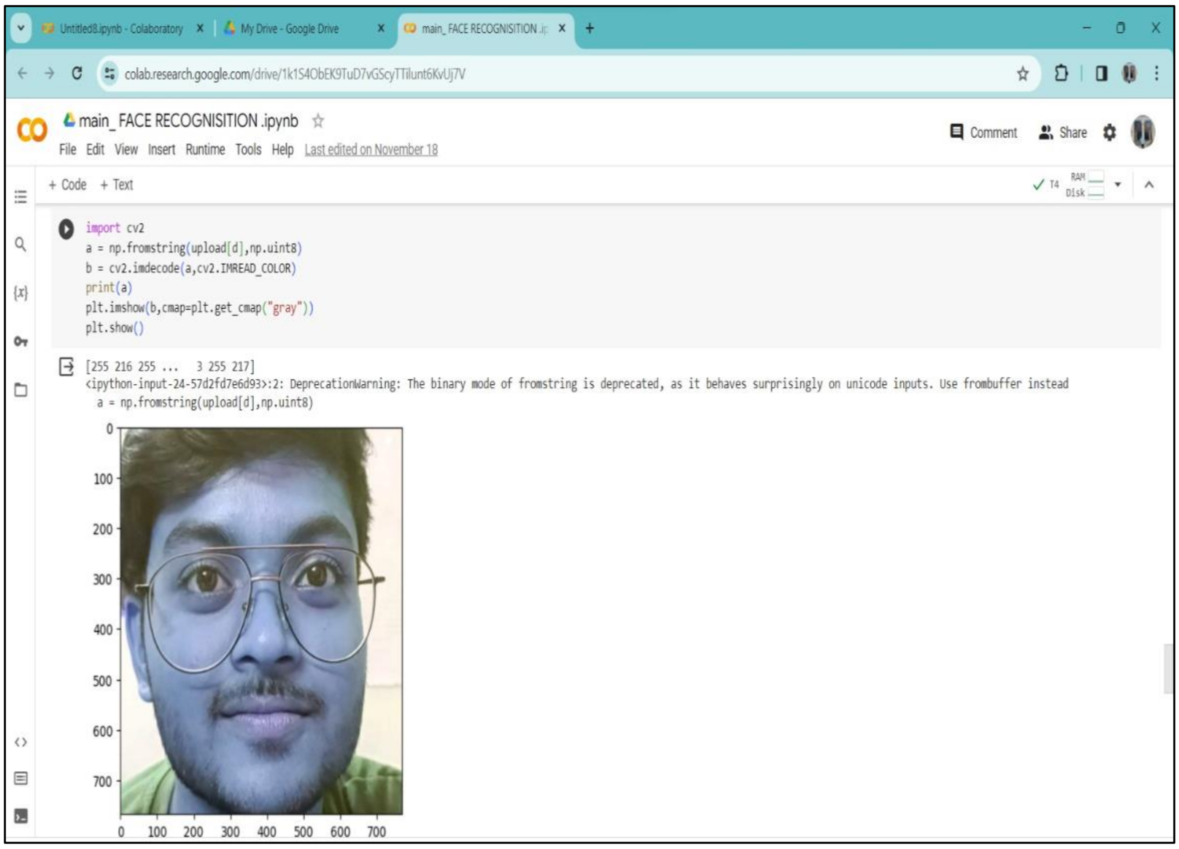

Untitled8.ipynb - Colaboratory
My Drive - Google Drive
main_ FACE RECOGNISITION .ip
colab.research.google.com/drive/1k1S4ObEK9TuD7vGScyTTilunt6KvUj7V
main_ FACE RECOGNISITION .ipynb
File Edit View Insert Runtime Tools Help Last edited on November 18
Comment
Share
+ Code + Text
T4
RAM
Disk

```
plt.imshow(b,cmap=plt.get_cmap("gray"))
plt.show()
```

```
b
```

```
array([[  7,   7,   6, ...,  11,  11,  11],
       [  7,   8,  82, ...,   9,   9,  11],
       [  7,  17, 146, ...,  11,   9,   9],
```

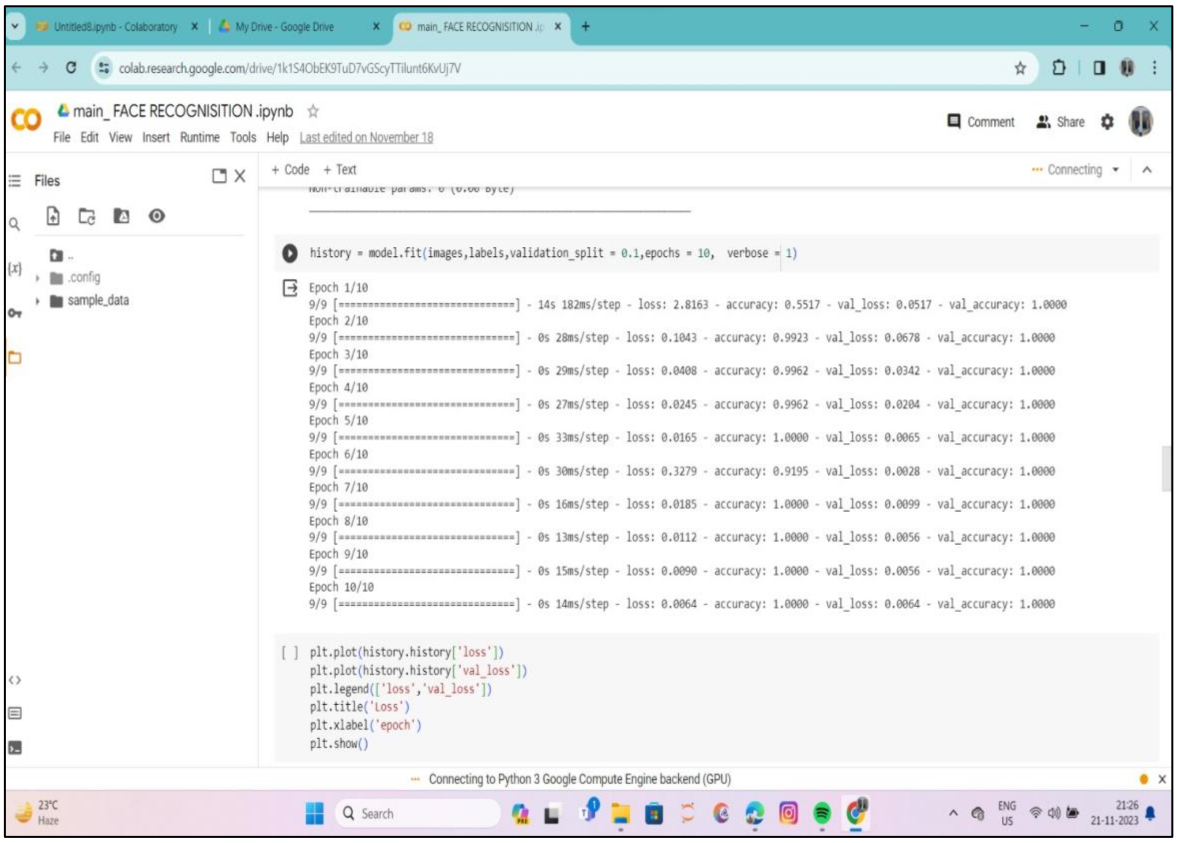
Fig No.- 32 (Data Analysis Part While Model Building, Part-02)


Untitled8.ipynb - Colaboratory
My Drive - Google Drive
main_ FACE RECOGNISITION .ip
colab.research.google.com/drive/1k1S4ObEK9TuD7vGScyTTilunt6KvUj7V
main_ FACE RECOGNISITION .ipynb
File Edit View Insert Runtime Tools Help Last edited on November 18
Comment
Share
+ Code + Text
T4
RAM
Disk

```
import cv2
a = np.fromstring(upload[d],np.uint8)
b = cv2.imdecode(a,cv2.IMREAD_COLOR)
print(a)
plt.imshow(b,cmap=plt.get_cmap("gray"))
plt.show()
```

```
[255 216 255 ...   3 255 217]
<ipython-input-24-57d2fd7e6d93>:2: DeprecationWarning: The binary mode of fromstring is deprecated, as it behaves surprisingly on unicode inputs. Use frombuffer instead
  a = np.fromstring(upload[d],np.uint8)
```

Fig No.- 33 (Data Analysis Part While Model Building, Part-03)



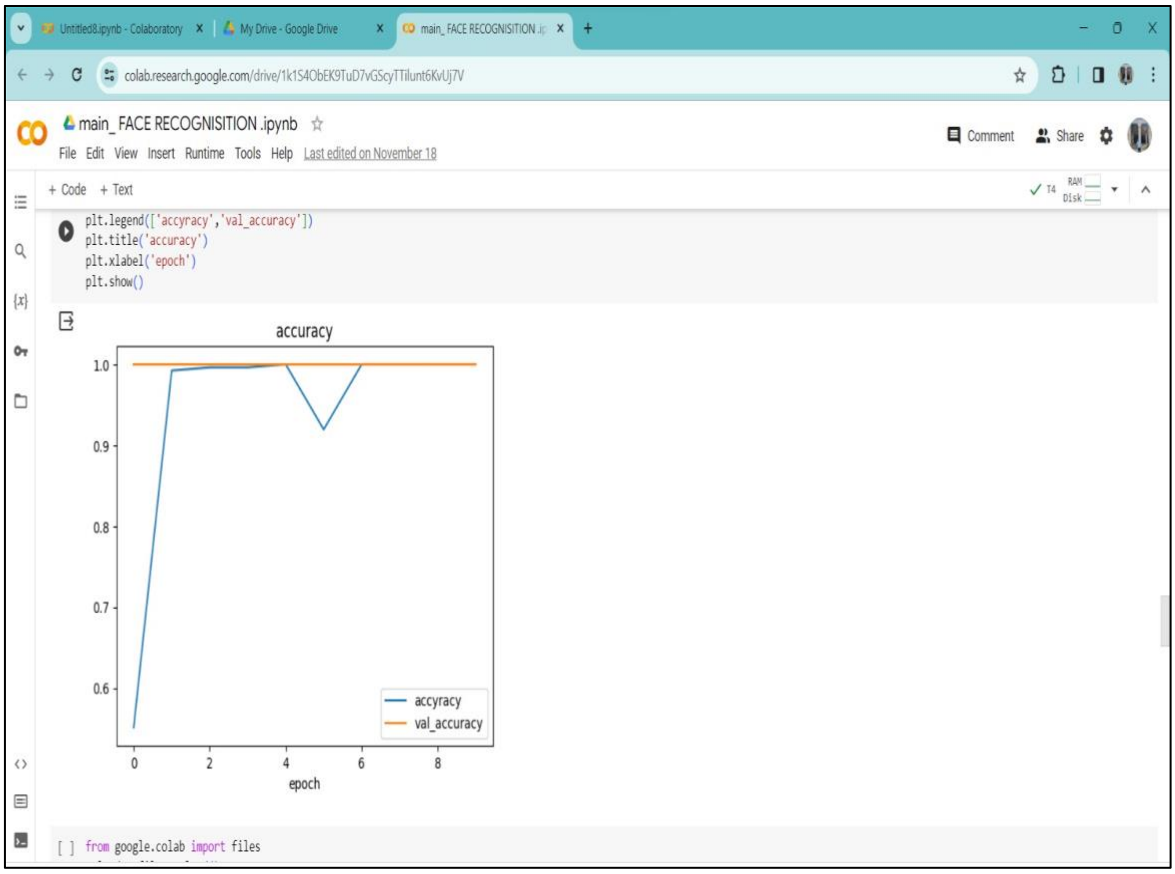
```
history = model.fit(images,labels,validation_split = 0.1,epochs = 10,  verbose = 1)
```

```
Epoch 1/10
9/9 [==============================] - 14s 182ms/step - loss: 2.8163 - accuracy: 0.5517 - val_loss: 0.0517 - val_accuracy: 1.0000
Epoch 2/10
9/9 [==============================] - 0s 28ms/step - loss: 0.1043 - accuracy: 0.9923 - val_loss: 0.0678 - val_accuracy: 1.0000
Epoch 3/10
9/9 [==============================] - 0s 29ms/step - loss: 0.0408 - accuracy: 0.9962 - val_loss: 0.0342 - val_accuracy: 1.0000
Epoch 4/10
9/9 [==============================] - 0s 27ms/step - loss: 0.0245 - accuracy: 0.9962 - val_loss: 0.0204 - val_accuracy: 1.0000
Epoch 5/10
9/9 [==============================] - 0s 33ms/step - loss: 0.0165 - accuracy: 1.0000 - val_loss: 0.0065 - val_accuracy: 1.0000
Epoch 6/10
9/9 [==============================] - 0s 30ms/step - loss: 0.3279 - accuracy: 0.9195 - val_loss: 0.0028 - val_accuracy: 1.0000
Epoch 7/10
9/9 [==============================] - 0s 16ms/step - loss: 0.0185 - accuracy: 1.0000 - val_loss: 0.0099 - val_accuracy: 1.0000
Epoch 8/10
9/9 [==============================] - 0s 13ms/step - loss: 0.0112 - accuracy: 1.0000 - val_loss: 0.0056 - val_accuracy: 1.0000
Epoch 9/10
9/9 [==============================] - 0s 15ms/step - loss: 0.0090 - accuracy: 1.0000 - val_loss: 0.0056 - val_accuracy: 1.0000
Epoch 10/10
9/9 [==============================] - 0s 14ms/step - loss: 0.0064 - accuracy: 1.0000 - val_loss: 0.0064 - val_accuracy: 1.0000
```

```
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.legend(['loss','val_loss'])
plt.title('Loss')
plt.xlabel('epoch')
plt.show()
```

main_FACE RECOGNISITION .ipynb
File Edit View Insert Runtime Tools Help Last edited on November 18
Files
Connecting to Python 3 Google Compute Engine backend (GPU)

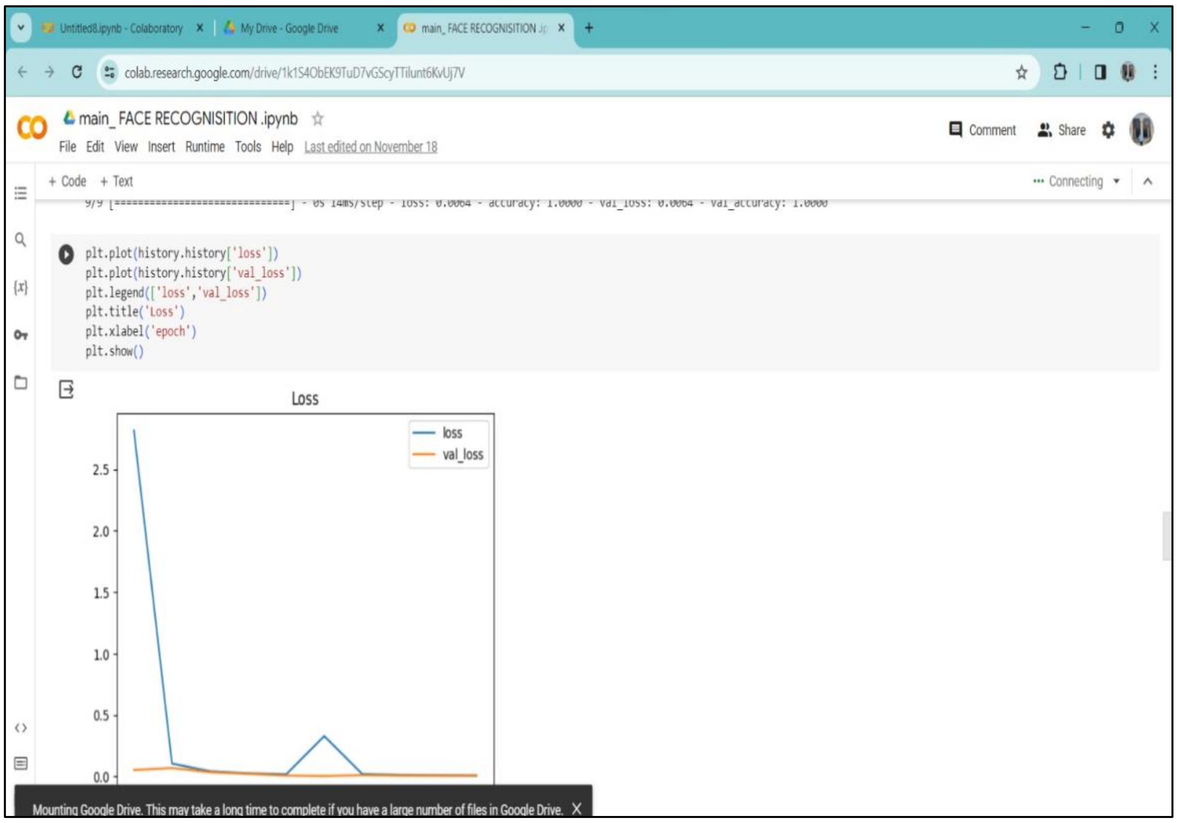
Fig No.- 34 (Model Execution Part)



```
plt.legend(['accyracy','val_accuracy'])
plt.title('accuracy')
plt.xlabel('epoch')
plt.show()
```

```
from google.colab import files
```

Fig No.- 35 (Accuracy Validation)

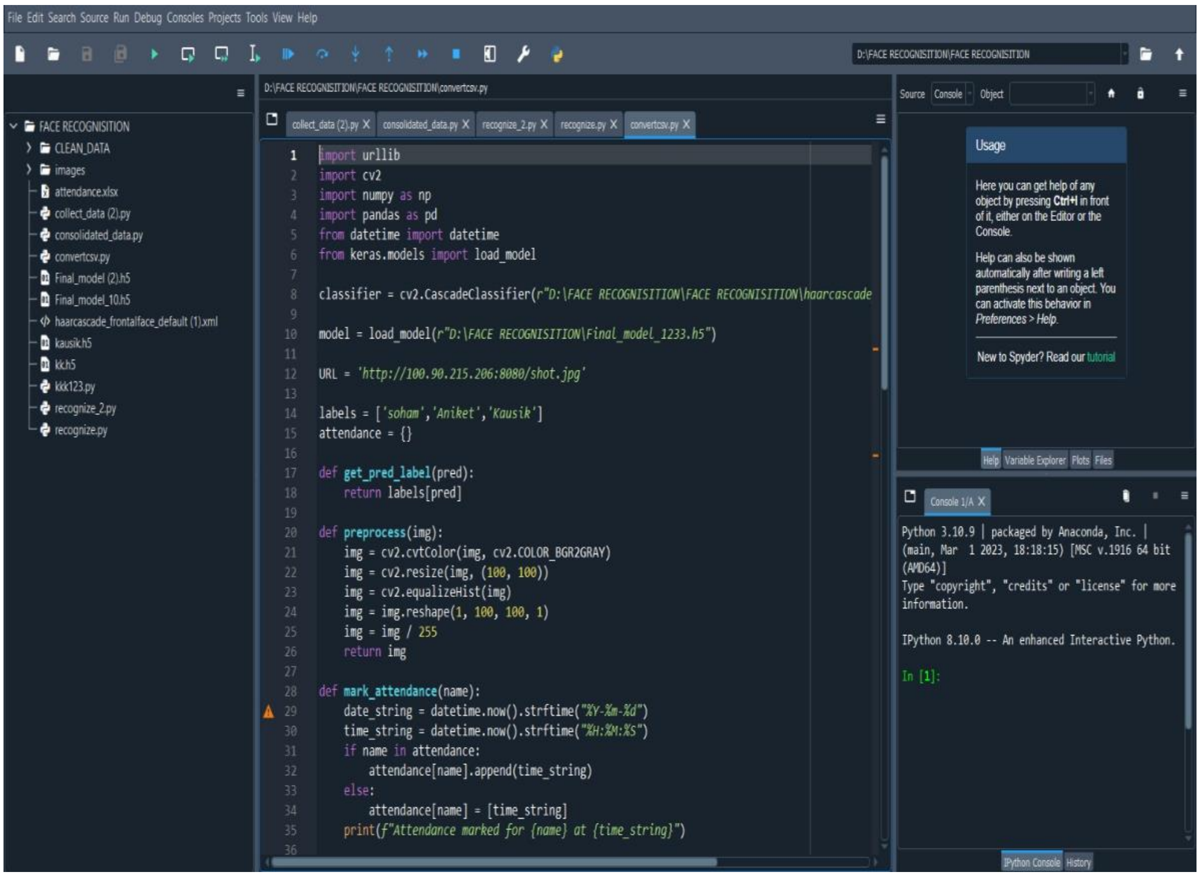


```
9/9 [==============================] - 0s 14ms/step - loss: 0.0064 - accuracy: 1.0000 - val_loss: 0.0064 - val_accuracy: 1.0000
```

```python
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.legend(['loss','val_loss'])
plt.title('Loss')
plt.xlabel('epoch')
plt.show()
```

main_FACE RECOGNISITION .ipynb
File Edit View Insert Runtime Tools Help Last edited on November 18
+ Code + Text
Connecting
Loss
loss
val_loss
2.5
2.0
1.5
1.0
0.5
0.0
Mounting Google Drive. This may take a long time to complete if you have a large number of files in Google Drive.

Fig No.- 36 (Loss Validation)

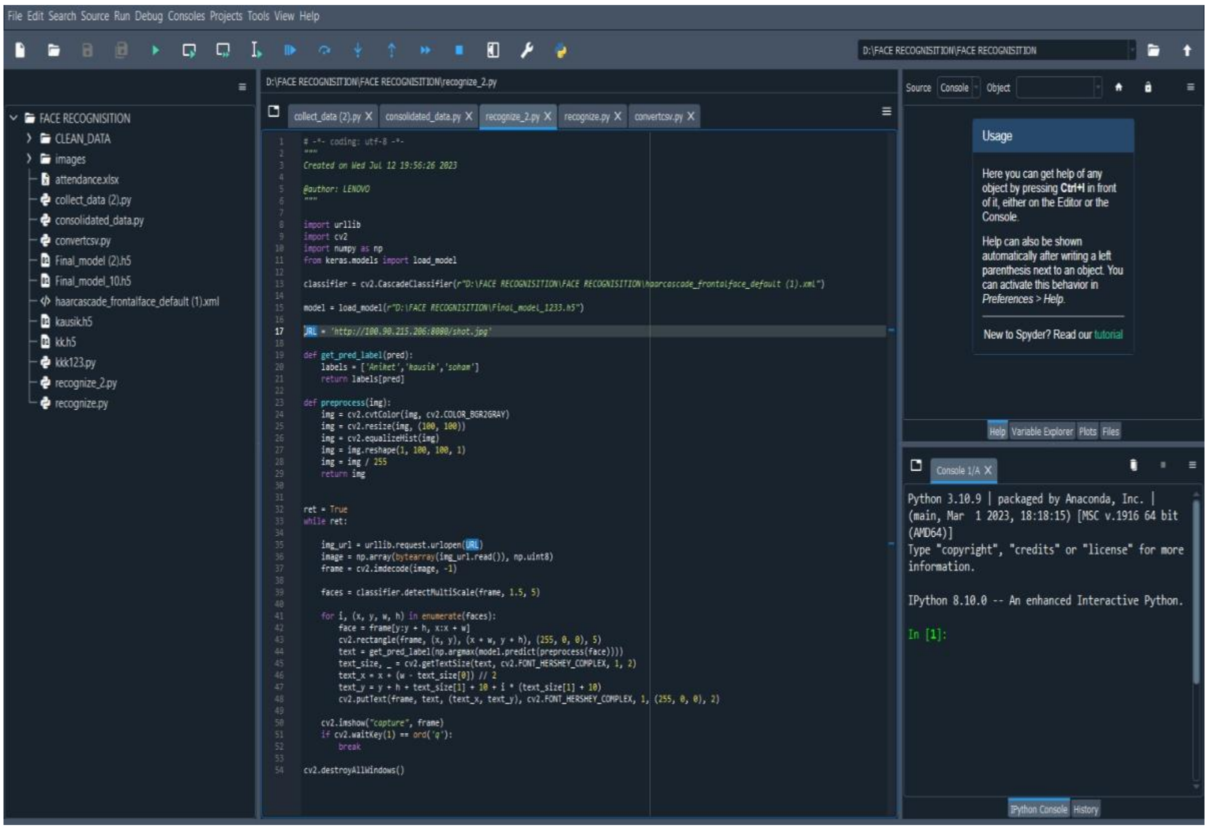


```python
import urllib
import cv2
import numpy as np
import pandas as pd
from datetime import datetime
from keras.models import load_model

classifier = cv2.CascadeClassifier(r"D:\FACE RECOGNISITION\FACE RECOGNISITION\haarcascade

model = load_model(r"D:\FACE RECOGNISITION\Final_model_1233.h5")

URL = 'http://100.90.215.206:8080/shot.jpg'

labels = ['soham','Aniket','Kausik']
attendance = {}

def get_pred_label(pred):
    return labels[pred]

def preprocess(img):
    img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    img = cv2.resize(img, (100, 100))
    img = cv2.equalizeHist(img)
    img = img.reshape(1, 100, 100, 1)
    img = img / 255
    return img

def mark_attendance(name):
    date_string = datetime.now().strftime("%Y-%m-%d")
    time_string = datetime.now().strftime("%H:%M:%S")
    if name in attendance:
        attendance[name].append(time_string)
    else:
        attendance[name] = [time_string]
    print(f"Attendance marked for {name} at {time_string}")
```

```
Python 3.10.9 | packaged by Anaconda, Inc. | (main, Mar  1 2023, 18:18:15) [MSC v.1916 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 8.10.0 -- An enhanced Interactive Python.

In [1]:
```

File Edit Search Source Run Debug Consoles Projects Tools View Help
D:\FACE RECOGNISITION\FACE RECOGNISITION
D:\FACE RECOGNISITION\FACE RECOGNISITION\convertcsv.py
collect_data (2).py
consolidated_data.py
recognize_2.py
recognize.py
convertcsv.py
FACE RECOGNISITION
CLEAN_DATA
images
attendance.xlsx
collect_data (2).py
consolidated_data.py
convertcsv.py
Final_model (2).h5
Final_model_10.h5
haarcascade_frontalface_default (1).xml
kausik.h5
kk.h5
kkk123.py
recognize_2.py
recognize.py
Usage
Here you can get help of any object by pressing Ctrl+I in front of it, either on the Editor or the Console.
Help can also be shown automatically after writing a left parenthesis next to an object. You can activate this behavior in Preferences > Help.
New to Spyder? Read our tutorial
Help Variable Explorer Plots Files
Console 1/A
IPython Console History

Fig No.- 37 (Final Deployment Program)



```python
# -*- coding: utf-8 -*-
"""
Created on Wed Jul 12 19:56:26 2023

@author: LENOVO
"""

import urllib
import cv2
import numpy as np
from keras.models import load_model

classifier = cv2.CascadeClassifier(r"D:\FACE RECOGNISITION\FACE RECOGNISITION\haarcascade_frontalface_default (1).xml")

model = load_model(r"D:\FACE RECOGNISITION\Final_model_1233.h5")

URL = 'http://100.90.215.206:8080/shot.jpg'

def get_pred_label(pred):
    labels = ['Aniket','kausik','soham']
    return labels[pred]

def preprocess(img):
    img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    img = cv2.resize(img, (100, 100))
    img = cv2.equalizeHist(img)
    img = img.reshape(1, 100, 100, 1)
    img = img / 255
    return img


ret = True
while ret:

    img_url = urllib.request.urlopen(URL)
    image = np.array(bytearray(img_url.read()), np.uint8)
    frame = cv2.imdecode(image, -1)

    faces = classifier.detectMultiScale(frame, 1.5, 5)

    for i, (x, y, w, h) in enumerate(faces):
        face = frame[y:y + h, x:x + w]
        cv2.rectangle(frame, (x, y), (x + w, y + h), (255, 0, 0), 5)
        text = get_pred_label(np.argmax(model.predict(preprocess(face))))
        text_size, _ = cv2.getTextSize(text, cv2.FONT_HERSHEY_COMPLEX, 1, 2)
        text_x = x + (w - text_size[0]) // 2
        text_y = y + h + text_size[1] + 10 + i * (text_size[1] + 10)
        cv2.putText(frame, text, (text_x, text_y), cv2.FONT_HERSHEY_COMPLEX, 1, (255, 0, 0), 2)

    cv2.imshow("capture", frame)
    if cv2.waitKey(1) == ord('q'):
        break

cv2.destroyAllWindows()
```

Usage

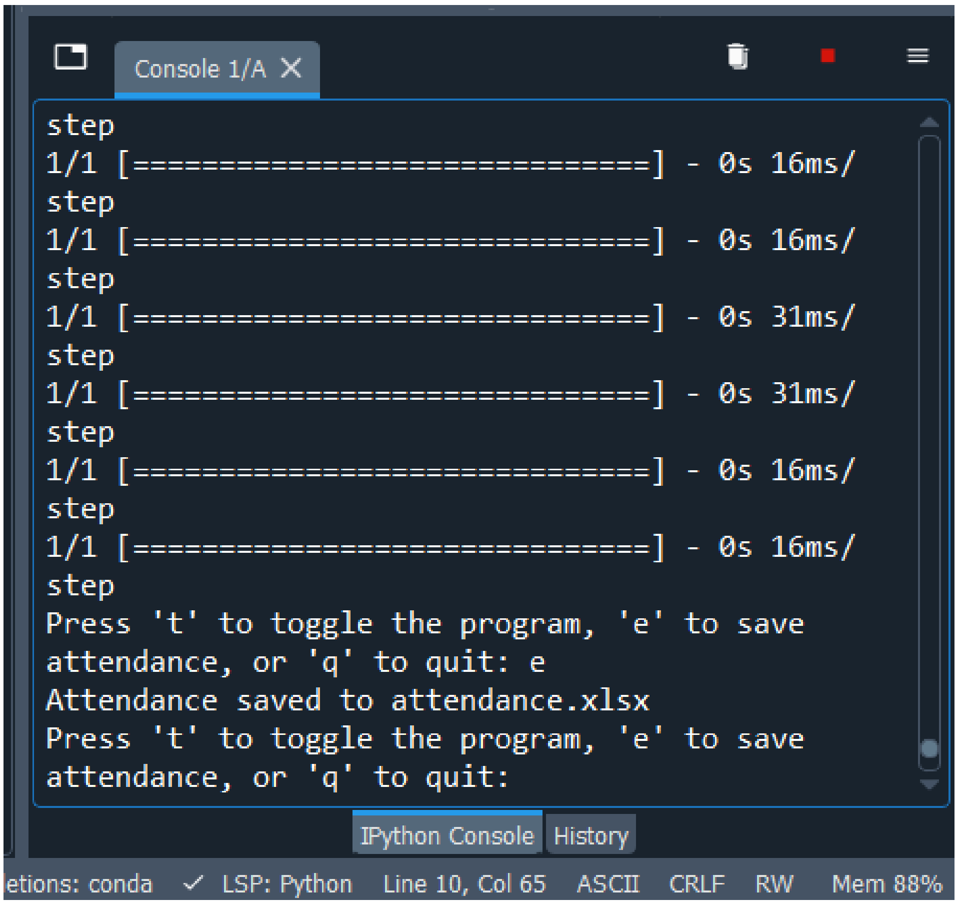
Here you can get help of any object by pressing **Ctrl+I** in front of it, either on the Editor or the Console.

Help can also be shown automatically after writing a left parenthesis next to an object. You can activate this behavior in *Preferences > Help*.

New to Spyder? Read our tutorial

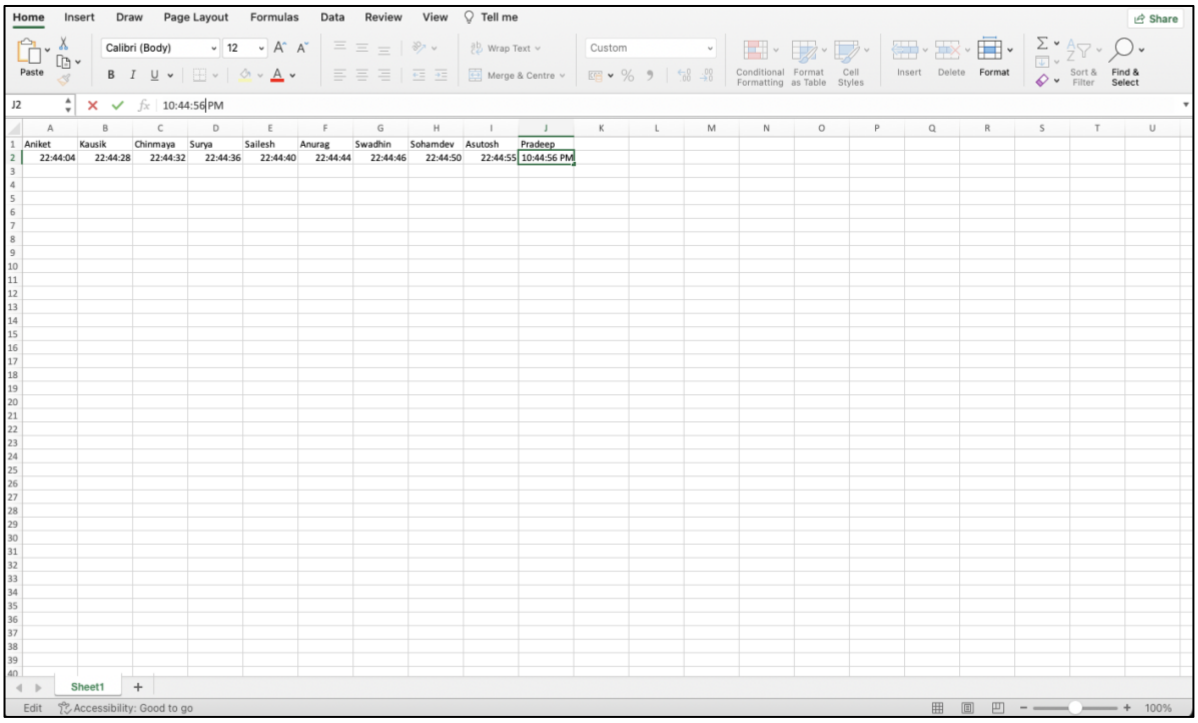
```
Python 3.10.9 | packaged by Anaconda, Inc. |
(main, Mar  1 2023, 18:18:15) [MSC v.1916 64 bit
(AMD64)]
Type "copyright", "credits" or "license" for more
information.

IPython 8.10.0 -- An enhanced Interactive Python.

In [1]:
```

Fig No.- 38 (Model Testing Program)



```
step
1/1 [==============================] - 0s 16ms/
step
1/1 [==============================] - 0s 16ms/
step
1/1 [==============================] - 0s 31ms/
step
1/1 [==============================] - 0s 31ms/
step
1/1 [==============================] - 0s 16ms/
step
1/1 [==============================] - 0s 16ms/
step
Press 't' to toggle the program, 'e' to save
attendance, or 'q' to quit: e
Attendance saved to attendance.xlsx
Press 't' to toggle the program, 'e' to save
attendance, or 'q' to quit:
```

Fig No.- 39 (Deployment Program Execution)



J2 10:44:56 PM

Aniket	Kausik	Chinmaya	Surya	Sailesh	Anurag	Swadhin	Sohamdev	Asutosh	Pradeep
22:44:04	22:44:28	22:44:32	22:44:36	22:44:40	22:44:44	22:44:46	22:44:50	22:44:55	10:44:56 PM

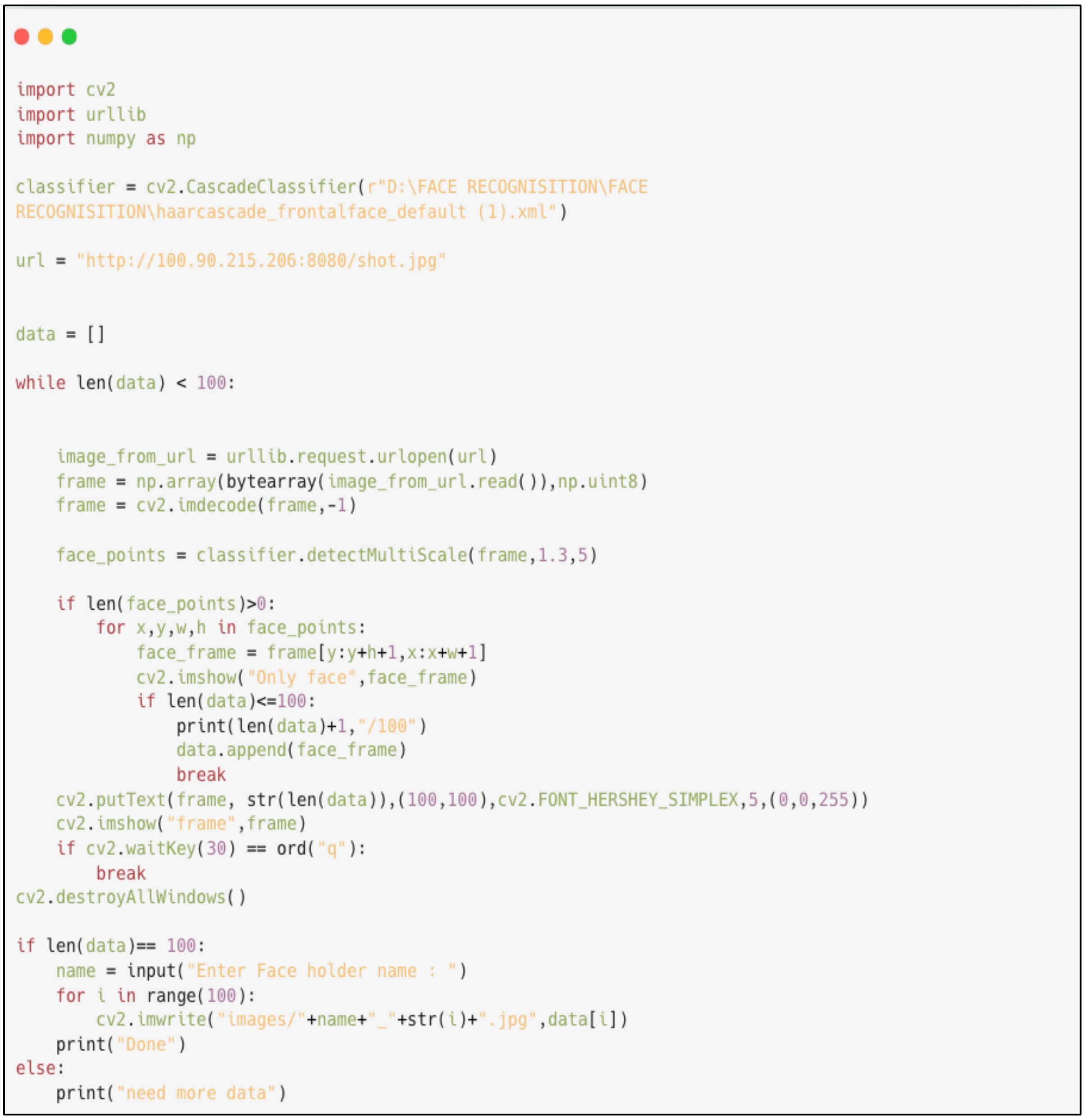
<u>Fig No.- 40 (Final Attendance Sheet in the form of Excel File)</u>

6. CODING

The coding section of this documentation encompasses the implementation of key modules essential for the successful development and deployment of the CNNFaceQuest project. This section is organized into four distinct modular components, each serving a crucial role in the overall functionality of the automated attendance management system.

6.1. DATA COLLECTION PROGRAM:

This module focuses on the initial step of gathering essential data for training the facial recognition model. The program utilizes the IP Webcam app to capture 100 images of individuals within a specified time frame. Each image is labelled with the individual's name to establish a robust dataset for subsequent model training.

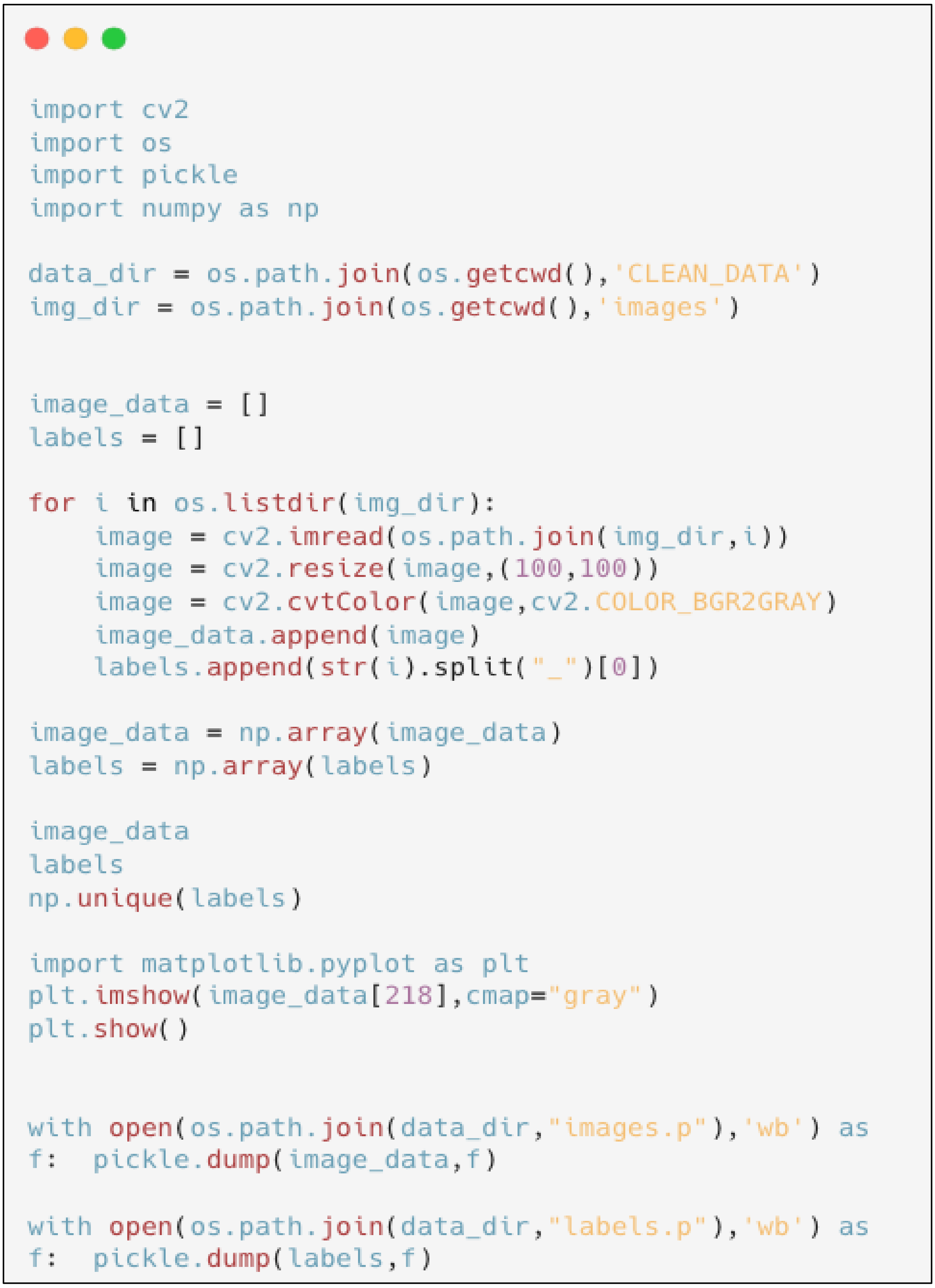


```python
import cv2
import urllib
import numpy as np

classifier = cv2.CascadeClassifier(r"D:\FACE RECOGNISITION\FACE
RECOGNISITION\haarcascade_frontalface_default (1).xml")

url = "http://100.90.215.206:8080/shot.jpg"


data = []

while len(data) < 100:


    image_from_url = urllib.request.urlopen(url)
    frame = np.array(bytearray(image_from_url.read()),np.uint8)
    frame = cv2.imdecode(frame,-1)

    face_points = classifier.detectMultiScale(frame,1.3,5)

    if len(face_points)>0:
        for x,y,w,h in face_points:
            face_frame = frame[y:y+h+1,x:x+w+1]
            cv2.imshow("Only face",face_frame)
            if len(data)<=100:
                print(len(data)+1,"/100")
                data.append(face_frame)
                break
    cv2.putText(frame, str(len(data)),(100,100),cv2.FONT_HERSHEY_SIMPLEX,5,(0,0,255))
    cv2.imshow("frame",frame)
    if cv2.waitKey(30) == ord("q"):
        break
cv2.destroyAllWindows()

if len(data)== 100:
    name = input("Enter Face holder name : ")
    for i in range(100):
        cv2.imwrite("images/"+name+"_"+str(i)+".jpg",data[i])
    print("Done")
else:
    print("need more data")
```

Fig No.- 41 (Data Collection Program)

6.2. DATA PREPROCESSING PROGRAM:

Following the data collection phase, this module is dedicated to preparing the collected dataset for efficient integration into the Google Collab environment. The program employs pickling, a serialization technique, to optimize the dataset, ensuring streamlined input for subsequent model training without compromising data integrity.



```
import cv2
import os
import pickle
import numpy as np

data_dir = os.path.join(os.getcwd(),'CLEAN_DATA')
img_dir = os.path.join(os.getcwd(),'images')


image_data = []
labels = []

for i in os.listdir(img_dir):
    image = cv2.imread(os.path.join(img_dir,i))
    image = cv2.resize(image,(100,100))
    image = cv2.cvtColor(image,cv2.COLOR_BGR2GRAY)
    image_data.append(image)
    labels.append(str(i).split("_")[0])

image_data = np.array(image_data)
labels = np.array(labels)

image_data
labels
np.unique(labels)

import matplotlib.pyplot as plt
plt.imshow(image_data[218],cmap="gray")
plt.show()


with open(os.path.join(data_dir,"images.p"),'wb') as
f:  pickle.dump(image_data,f)

with open(os.path.join(data_dir,"labels.p"),'wb') as
f:  pickle.dump(labels,f)
```

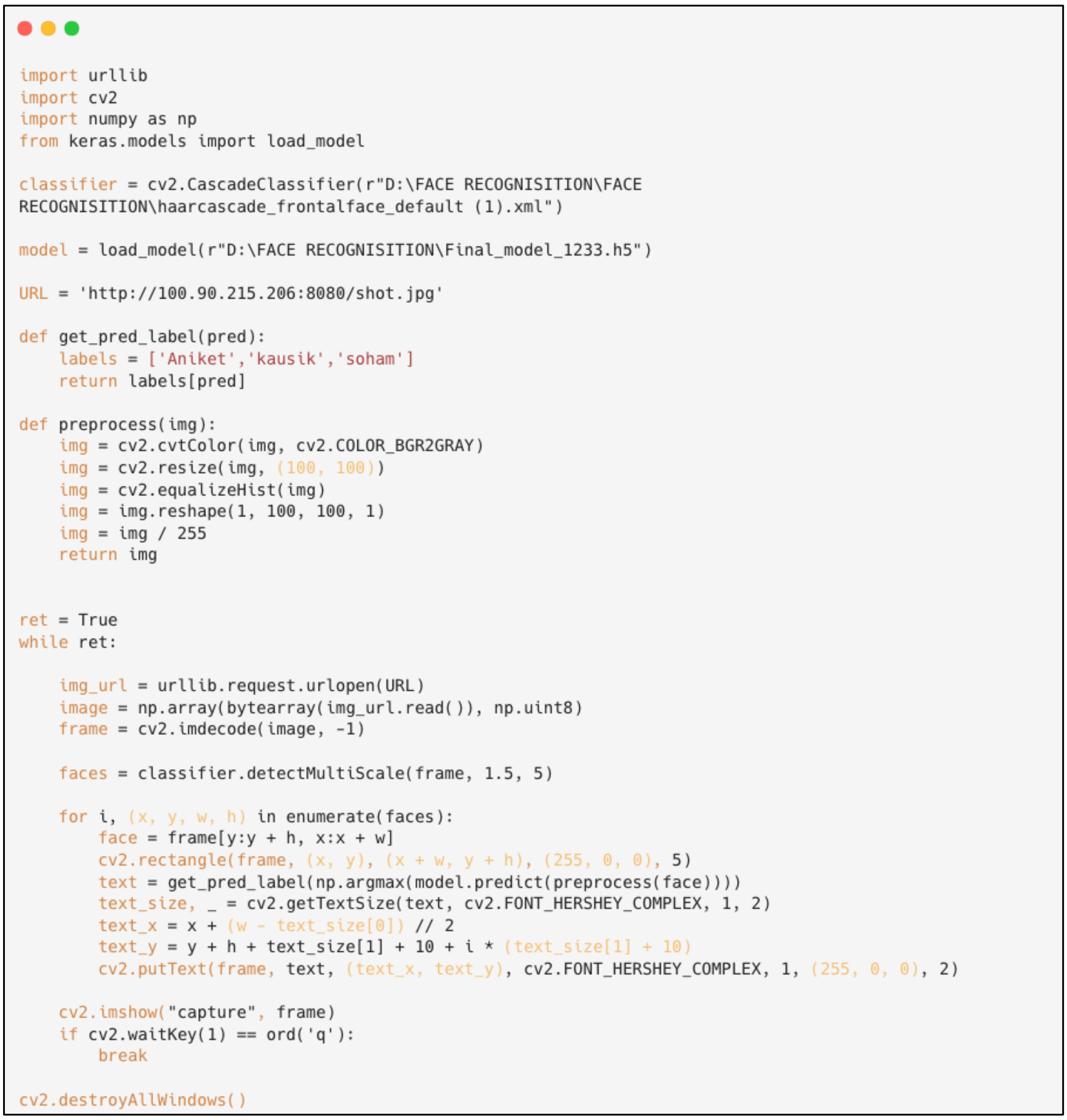
Fig No.- 42 (Data Pre-processing Program)

6.3. MODEL DEVELOPMENT (OMITTED FROM DOCUMENTATION):

While the detailed script for model development is intentionally omitted from this documentation due to security considerations, this critical component involves implementing a Convolutional Neural Network (CNN) for face detection and recognition. The model is trained to accurately identify individuals based on the labelled dataset.

6.4. MODEL TESTING PROGRAM:

This module is responsible for assessing the efficacy and accuracy of the trained model. The testing program employs various scenarios to evaluate the model's performance in recognizing registered individuals, ensuring robust functionality before deployment.

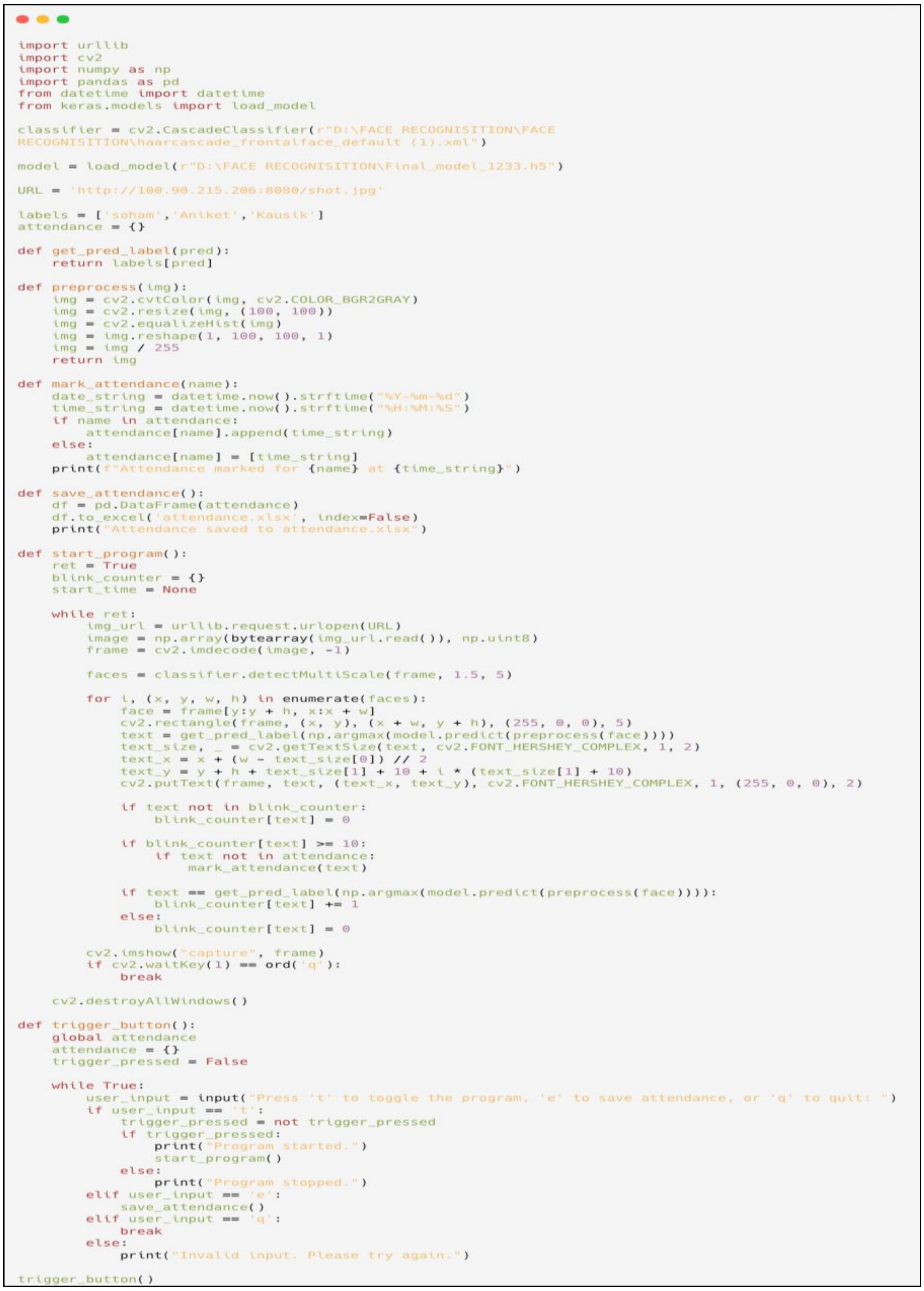


```python
import urllib
import cv2
import numpy as np
from keras.models import load_model

classifier = cv2.CascadeClassifier(r"D:\FACE RECOGNISITION\FACE
RECOGNISITION\haarcascade_frontalface_default (1).xml")

model = load_model(r"D:\FACE RECOGNISITION\Final_model_1233.h5")

URL = 'http://100.90.215.206:8080/shot.jpg'

def get_pred_label(pred):
    labels = ['Aniket','kausik','soham']
    return labels[pred]

def preprocess(img):
    img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    img = cv2.resize(img, (100, 100))
    img = cv2.equalizeHist(img)
    img = img.reshape(1, 100, 100, 1)
    img = img / 255
    return img


ret = True
while ret:

    img_url = urllib.request.urlopen(URL)
    image = np.array(bytearray(img_url.read()), np.uint8)
    frame = cv2.imdecode(image, -1)

    faces = classifier.detectMultiScale(frame, 1.5, 5)

    for i, (x, y, w, h) in enumerate(faces):
        face = frame[y:y + h, x:x + w]
        cv2.rectangle(frame, (x, y), (x + w, y + h), (255, 0, 0), 5)
        text = get_pred_label(np.argmax(model.predict(preprocess(face))))
        text_size, _ = cv2.getTextSize(text, cv2.FONT_HERSHEY_COMPLEX, 1, 2)
        text_x = x + (w - text_size[0]) // 2
        text_y = y + h + text_size[1] + 10 + i * (text_size[1] + 10)
        cv2.putText(frame, text, (text_x, text_y), cv2.FONT_HERSHEY_COMPLEX, 1, (255, 0, 0), 2)

    cv2.imshow("capture", frame)
    if cv2.waitKey(1) == ord('q'):
        break

cv2.destroyAllWindows()
```

Fig No.- 43 (Model Testing Program)

6.5. MODEL DEPLOYMENT PROGRAM:

The final module revolves around deploying the trained model in a real-time environment. This program interfaces with Haar-cascade classifiers and computer vision techniques to continuously recognize registered individuals. Additionally, it leverages the IP Webcam app as a real-time CCTV substitute, marking attendance in an Excel file based on recognition criteria.



```python
import urllib
import cv2
import numpy as np
import pandas as pd
from datetime import datetime
from keras.models import load_model

classifier = cv2.CascadeClassifier(r"D:\FACE RECOGNISITION\FACE
RECOGNISITION\haarcascade_frontalface_default (1).xml")

model = load_model(r"D:\FACE RECOGNISITION\Final_model_1233.h5")

URL = 'http://100.90.215.206:8080/shot.jpg'

labels = ['soham','Aniket','Kausik']
attendance = {}

def get_pred_label(pred):
    return labels[pred]

def preprocess(img):
    img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    img = cv2.resize(img, (100, 100))
    img = cv2.equalizeHist(img)
    img = img.reshape(1, 100, 100, 1)
    img = img / 255
    return img

def mark_attendance(name):
    date_string = datetime.now().strftime("%Y-%m-%d")
    time_string = datetime.now().strftime("%H:%M:%S")
    if name in attendance:
        attendance[name].append(time_string)
    else:
        attendance[name] = [time_string]
    print(f"Attendance marked for {name} at {time_string}")

def save_attendance():
    df = pd.DataFrame(attendance)
    df.to_excel('attendance.xlsx', index=False)
    print("Attendance saved to attendance.xlsx")

def start_program():
    ret = True
    blink_counter = {}
    start_time = None

    while ret:
        img_url = urllib.request.urlopen(URL)
        image = np.array(bytearray(img_url.read()), np.uint8)
        frame = cv2.imdecode(image, -1)

        faces = classifier.detectMultiScale(frame, 1.5, 5)

        for i, (x, y, w, h) in enumerate(faces):
            face = frame[y:y + h, x:x + w]
            cv2.rectangle(frame, (x, y), (x + w, y + h), (255, 0, 0), 5)
            text = get_pred_label(np.argmax(model.predict(preprocess(face))))
            text_size, _ = cv2.getTextSize(text, cv2.FONT_HERSHEY_COMPLEX, 1, 2)
            text_x = x + (w - text_size[0]) // 2
            text_y = y + h + text_size[1] + 10 + i * (text_size[1] + 10)
            cv2.putText(frame, text, (text_x, text_y), cv2.FONT_HERSHEY_COMPLEX, 1, (255, 0, 0), 2)

            if text not in blink_counter:
                blink_counter[text] = 0

            if blink_counter[text] >= 10:
                if text not in attendance:
                    mark_attendance(text)

            if text == get_pred_label(np.argmax(model.predict(preprocess(face)))):
                blink_counter[text] += 1
            else:
                blink_counter[text] = 0

        cv2.imshow("capture", frame)
        if cv2.waitKey(1) == ord('q'):
            break

    cv2.destroyAllWindows()

def trigger_button():
    global attendance
    attendance = {}
    trigger_pressed = False

    while True:
        user_input = input("Press 't' to toggle the program, 'e' to save attendance, or 'q' to quit: ")
        if user_input == 't':
            trigger_pressed = not trigger_pressed
            if trigger_pressed:
                print("Program started.")
                start_program()
            else:
                print("Program stopped.")
        elif user_input == 'e':
            save_attendance()
        elif user_input == 'q':
            break
        else:
            print("Invalid input. Please try again.")

trigger_button()
```

Fig No.- 44 (Model Deployment Program)

By dividing the implementation into modular components, this coding section aims to provide a clear and organized representation of the development process. Each module contributes to the overarching goal of creating an efficient and secure automated attendance management system. Subsequently, the codes for the Data Collection, Data Pre-processing, Model Testing, and Model Deployment programs will be presented below.

7. TESTING

The testing phase is a critical component of the CNNFaceQuest project, ensuring the reliability and accuracy of the developed system. This section primarily focuses on Unit Testing, a methodical approach to validate individual units of the software independently. For the purpose of this project, unit testing is particularly applied to the Model Testing Program, evaluating its functionality under various scenarios.

7.1. UNIT TESTING

The Unit Testing phase involves systematically assessing specific components of the Model Testing Program to ensure they meet the defined requirements. Below, test cases are outlined, detailing the input conditions, expected outputs, actual outputs, and any remedial actions taken in the event of discrepancies.

7.1.1. Test Case 1: Recognition of Registered Individual

- **Input:** Feed a frame containing the image of a registered individual to the testing program.
- **Expected Output:** The program should correctly recognize the individual and display their name in the recognition box.
- **Actual Output:** Verify if the program accurately identifies the registered individual.
- **Remedial Actions:** If the recognition fails, inspect the training data, model, and real-time deployment settings for discrepancies.

7.1.2. Test Case 2: Unregistered Face Recognition

- **Input:** Provide a frame with the image of an individual not present in the training dataset.
- **Expected Output:** The program should not recognize the unregistered face, and the recognition box should remain empty.
- **Actual Output:** Confirm that the program refrains from identifying faces not present in the training dataset.
- **Remedial Actions:** If an unregistered face is incorrectly recognized, review the dataset and model training process for potential issues.

7.1.3. Test Case 3: Triggered Attendance Submission

- **Input:** Simulate a user-triggered attendance submission by activating the attendance button.
- **Expected Output:** The program should mark attendance for the recognized individual in the Excel file.
- **Actual Output:** Verify that attendance is accurately recorded when the user triggers the submission.
- **Remedial Actions:** If attendance is not marked correctly, inspect the button activation and attendance recording mechanisms.

7.1.4. Test Case 4: Continuous Recognition

- **Input:** Present multiple frames with the image of a registered individual in succession.
- **Expected Output:** The program should consistently recognize the individual, displaying their name continuously for the specified duration.
- **Actual Output:** Ensure that continuous recognition is functioning as expected.
- **Remedial Actions:** If continuous recognition fails, examine the recognition algorithm and parameters for potential adjustments.

7.1.5. Test Case 5: Error Handling

- **Input:** Introduce errors such as image distortion or partial face occlusion in the testing frames.
- **Expected Output:** The program should appropriately handle recognition errors and provide feedback.
- **Actual Output:** Confirm that the program responds appropriately to recognition anomalies.
- **Remedial Actions:** Implement error handling mechanisms and improve the program's robustness against recognition errors.

The Unit Testing process is crucial for identifying and rectifying potential issues in the Model Testing Program. By systematically evaluating individual components under varying conditions, this phase ensures the reliability and effectiveness of the automated attendance management system developed in the CNNFaceQuest project.

8. CONCLUSION & LIMITATIONS

8.1. CONCLUSION

The CNNFaceQuest project introduces an innovative solution for automated attendance management, leveraging Convolutional Neural Networks (CNN) for accurate face detection and recognition. Through the modular implementation of data collection, pre-processing, model development, testing, and deployment, the project aims to streamline attendance tracking processes in diverse organizational settings.

The data collection program efficiently captures 100 images of individuals within a specified timeframe, laying the foundation for a robust dataset. Data pre-processing ensures the optimization of the collected dataset for seamless integration into the Google Collab environment, employing pickling to enhance efficiency. The core of the project lies in the model development, where a CNN is trained to recognize registered individuals, enhancing accuracy and reliability.

Real-time deployment utilizes Haar cascade classifiers and computer vision techniques, interfacing with the IP Webcam app as a substitute for CCTV. The continuous recognition feature ensures the consistent identification of registered individuals, and the triggered attendance submission mechanism, marked in an Excel file, adds a layer of user control. The entire system prioritizes user privacy through secure storage and encryption measures, complying with ethical guidelines and legal regulations.

However, it's imperative to acknowledge the limitations of the CNNFaceQuest project.

8.2. LIMITATIONS

8.2.1. Security Concerns:

The detailed script for model development is intentionally omitted from the documentation due to security considerations. While this measure safeguards the project, it also limits the transparency of the implementation details.

8.2.2. Dependency on Training Data Quality:

The effectiveness of the face recognition model heavily relies on the quality and representativeness of the training dataset. Inadequate or biased data may lead to suboptimal performance and biased results.

8.2.3. Sensitivity to Environmental Factors:

The system's performance may be influenced by environmental factors such as lighting conditions and crowded spaces. Variations in these factors could impact the accuracy of face detection and recognition.

8.2.4. User Triggered Attendance:

The attendance submission is initiated by a user-triggered button. While this provides user control, it also introduces a manual element that may be subject to user oversight or intentional misuse.

8.2.5. Error Handling Challenges:

Although the system incorporates error handling mechanisms, addressing all possible recognition errors, especially in dynamic environments, can be challenging. The system may encounter difficulties with distorted images or partial face occlusion.

8.2.6. Scalability and Generalization:

The scalability of the system and its ability to generalize to various organizational settings may be limited. Additional considerations and adaptations may be required for larger organizations or unique environmental conditions.

8.2.7. Future-Ready Architecture:

While the system is designed with a future-ready architecture, the ability to seamlessly integrate with emerging technologies may be contingent on the adaptability of the system to evolving industry standards.

In conclusion, the CNNFaceQuest project demonstrates a robust solution for automating attendance management through innovative face detection and recognition techniques. The modular implementation ensures a structured and organized development process. Despite the limitations, the project showcases potential applications in various domains, contributing to the ongoing advancements in deep learning and computer vision technologies. As technology evolves, addressing these limitations and refining the system will be pivotal to enhancing its effectiveness and applicability in real-world scenarios. The CNNFaceQuest project, with its strengths and considerations, sets the stage for further exploration and refinement in the realm of automated attendance systems.

9. REFERENCE

Websites:

- TensorFlow Documentation: TensorFlow is an open-source machine learning library, and its documentation provides valuable resources for deep learning projects.
- Keras Documentation: Keras is an open-source deep learning library written in Python. It is often used as a high-level neural networks API, and its documentation is a great reference for building neural network models.
- OpenCV Documentation: OpenCV is a widely used computer vision library. Its documentation offers comprehensive information and tutorials for image processing tasks.
- PyTorch Documentation: PyTorch is an open-source deep learning library known for its dynamic computational graph. The documentation provides guides and references for using PyTorch in deep learning projects.
- Towards Data Science: This is a Medium publication that often features articles and tutorials on various machine learning and deep learning topics.

Research Papers:

[1] Godswill, O., Osas, O., Anderson, O., Oseikhuemen, I., & Etse, O. (2018). Automated student attendance management system using face recognition. International Journal of Educational Research and Information Science, 5(4), 31-37.

[2] Ali, N. S., Alhilali, A. H., Rjeib, H. D., Alsharqi, H., & Al-Sadawi, B. (2022). Automated attendance management systems: systematic literature review. International Journal of Technology Enhanced Learning, 14(1), 37-65.

[3] Raghuwanshi, A., & Swami, P. D. (2017, May). An automated classroom attendance system using video based face recognition. In 2017 2nd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT) (pp. 719-724). IEEE.

[4] Kovelan, P., Thisenthira, N., & Kartheeswaran, T. (2019, December). Automated attendance monitoring system using iot. In 2019 International Conference on Advancements in Computing (ICAC) (pp. 376-379). IEEE.

[5] Gupta, N., Sharma, P., Deep, V., & Shukla, V. K. (2020, June). Automated attendance system using OpenCV. In 2020 8th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions)(ICRITO) (pp. 1226-1230). IEEE.