A

Project Report on

**C-In C-Out**

Submitted in partial fulfilment of completion of the course

**Advanced Diploma in IT, Networking and Cloud**

**Submitted by:**

Sumit Kumar Singh

Mantu Kumar Rana

Madan H S

Mayank Pathak

**Under the Guidance of:**

Priya Kalra (IBM Mentor)

Nashit Humam (Edunet mentor)

Ankur Mishra (Edunet Mentor)



**Index**

## [Abstract](#Abstract) ……………………………… 2

## [Acknowledgement](#Acknowledgement) ………….………………………………. 4

## [Team Composition & Workload](#TeamCompositionWorkload) ………..……………………………….. 4

## [Table of Contents](#TableofContents) ……………..…………………………… 5

* [Introduction to Problem](#Introductiontoproblem) …………………………………… 5
* [Literature Review](#LiteratureReview) …………………………………… 6
* [Proposed Solution](#ProposedSolution) …………………………………… 7
* [Requirements](#Requirements) …………………………………… 8

#### [User Requirements](#UserRequirements) ……………………………………… 9

#### [Design Documentation](#DesignDocumentation) ……………………………………… 10

#### [Implementation Details](#ImplementationDetails) ……………………………………… 14

#### [Testing](#Testing) ………………………………………. 17

#### [Deployment](#Deployment) ………………………………………. 20

#### [Future Scope](#FutureScope) ……………………………………….. 22

#### [Conclusion](#Conclusion) ………………………………………… 23

Appendix A (project code)

Appendix B (screenshot of project)

**Abstract:**

The project involves the development of a Face Attendance System specifically for institutions such as schools, universities, offices, and hospitals. The primary objective of this system is to streamline and automate the attendance management process using facial recognition technology. A key feature of the system is its ability to count the number of people have entered & exited a room. This feature enhances safety, particularly in environments with fluctuating attendance.

**Key Features:**

**Facial Recognition:** The system employs advanced facial recognition algorithms to identify and verify individuals based on their facial features. This eliminates the need for traditional methods such as manual sign-ins or RFID cards, providing a seamless and secure attendance process.

**Institution-Specific Customization:** The system is designed with a focus on four key types of institutions: schools, universities, offices, and hospitals. Each institution has unique attendance requirements and regulations, and the system is designed to meet these specific needs.

* **Schools/Universities:** The system can manage student attendance, and generate reports for academic performance evaluations.
* **Offices:** The system supports employee attendance tracking, shift management, and integration with payroll systems.
* **Hospitals:** The system can be used to monitor the attendance of medical staff, including doctors, nurses, ensuring that shifts are adequately covered.

**Admin-Only Access:** To ensure data security and maintain the integrity of the attendance records, only authorized administrators can log in and access the system. These administrators are typically high-level personnel, such as school principals, university administrators, office HR managers, or hospital administrators.

**User-Friendly Interface:** The system features a user-friendly interface that is easy to navigate, making it accessible for non-technical users. The design ensures that administrators can quickly perform their tasks without the need for extensive training.

**Bi-directional Counting:** Tracks and records the number of people entering and exiting a room through each entry point.

**Accuracy and Precision:** Utilizes advanced image processing to distinguish between individuals, even in crowded conditions.

**Acknowledgement:**

I would like to express my deepest gratitude to everyone who contributed to the successful completion of this project. First and foremost, I am thankful to my mentors and instructors, whose guidance, encouragement, and expertise were invaluable throughout this journey. Their insightful feedback and unwavering support greatly enhanced the quality and functionality of the Face Attendance System.

I extend my sincere thanks to my peers and colleagues for their collaborative spirit and constructive discussions, which were instrumental in refining and improving the project. Their diverse perspectives and suggestions helped shape the final product.

This project would not have been possible without the contributions of each and every one of you. Thank you for your support, inspiration, and trust.

**Team Composition & Workload:**

Sumit Kumar Singh : Frontend / Backend Development / Database / Deployment

Mantu Kumar Rana: Backend Development / Database  
Madan H S: Frontend Development / Resources

Mayank Pathak: Frontend Development / Documentation

**Table of Contents**

**Introduction to problem:**

In today's dynamic environments such as schools, universities, offices, and hospitals, managing attendance and monitoring the flow of people are critical tasks that directly impact security, operational efficiency, and resource allocation. Traditional methods of attendance tracking, often reliant on manual sign-ins or swipe cards, are not only time-consuming but also prone to errors and fraud, such as buddy punching or unauthorized access. These methods fail to provide real-time data and are inadequate for environments where large numbers of people frequently enter and exit rooms, making it difficult to maintain accurate occupancy records. The challenge is further compounded by the need for a system that can seamlessly integrate into existing infrastructures, scale across different environments, and operate efficiently in varied conditions, such as fluctuating lighting and crowded spaces. As organizations strive for greater efficiency, security, and data-driven decision-making, there is a clear need for an advanced solution that can not only automate attendance but also provide accurate, real-time people counting to monitor occupancy levels effectively.

**Literature review:**

Face recognition technology has been extensively researched and implemented across various domains due to its accuracy and non-intrusive nature. Early studies in biometric authentication explored fingerprint and iris recognition, but face recognition emerged as a preferred method due to its ease of use and ability to function in non-cooperative environments. Advanced algorithms, particularly deep learning models like Convolution Neural Networks (CNNs), to enhance the precision and reliability of face recognition systems. Most research focuses on optimizing algorithmic performance in controlled settings, but there is limited exploration of these systems challenges, such as varying lighting conditions, diverse facial expressions, and partial occlusions. Moreover, the integration of face recognition technology with existing institutional frameworks, such as attendance management systems in schools and hospitals, has not been thoroughly investigated. The existing gaps, the development of a face attendance system tailored specifically for schools, universities, offices, and hospitals is both relevant and necessary. This project seeks to provide a comprehensive solution that enhances efficiency while addressing privacy concerns. Early approaches relied on manual counting and sensor-based methods like infrared beams, which offered limited accuracy and scalability. With advancements in computer vision and AI, facial recognition technology emerged as a more reliable solution for tracking individuals. Recent literature highlights the effectiveness of using deep learning algorithms for real-time face detection and people counting, significantly improving accuracy over traditional methods.

**Proposed Solution:**

**Face Recognition Technology:**

* **Advanced Algorithms:** The system will use Convolution Neural Networks (CNNs) for accurate and reliable face detection and recognition. This technology will ensure that faces are identified quickly and correctly, even in varying conditions such as different lighting or facial expressions.
* **Real-Time Processing:** The system will be capable of processing and recognizing faces in real-time, allowing for immediate attendance marking as individuals enter the premises.

**Role-Based Access Control:**

* **Admin-Only Access:** Only authorized personnel (admins) will have access to the attendance system’s management interface. This role-based access control will ensure that attendance data is secure and that only designated individuals can modify or view records.
* **User Interface:** A user-friendly interface will be developed, using tools like Streamlit, to allow admins to easily manage attendance records, view reports, and handle exceptions.

**Monitor Activity:**

* **Real-Time People Counting:** Implement a system that provides continuous, real-time tracking of the number of people present, entering, and exiting the room.
* **Occupancy data:** It provides real-time occupancy data, ensuring precise monitoring and enhancing security. The system is designed to handle various environmental challenges, such as crowded spaces.

**Requirements:**

**Technology stack:**

* Streamlit**:** For building the user interface, allowing admins to interact with the system, manage attendance records.

**Backend:**

* **Python:** Core programming language for implementing face recognition algorithms and handling backend logic.
* **OpenCV:** Library for computer vision tasks, such as capturing and processing images from cameras.
* **Face Recognition Library:** For facial feature extraction and recognition.
* **Yolov10n:** To train & test & predict by using pre-trained & making models.

**Database:**

**SQLite3:** For storing user data, attendance records, and facial feature vectors.

**Machine Learning/AI:**

**Tensor Flow/PyTorch:** For developing and training custom face recognition models.

**Hardware:**

**High-Resolution Cameras:** For capturing clear images or video streams to ensure accurate face recognition.

**Software:**

**Windows:** Can be used for local development and deployment, especially if using Streamlit.

**Deployment Environment:**

Streamlit is a powerful tool for deploying data science and machine learning applications with a focus on simplicity and quick deployment. It allows you to build web applications directly from Python scripts.

**User Requirements:**

* **Accurate face recognition:** The system can accurately identify individuals based on their facial features, even in varying lighting conditions and with different facial expressions.
* **Real-time attendance tracking:** The system can capture and record attendance data in real-time, providing immediate feedback on presence or absence.
* **User management:** The admins are allowed for easy addition, modification, and deletion of user profiles, including personal information and facial data.
* **Attendance reports:** The system should generate detailed attendance reports for individuals, groups, and overall attendance.
* **Data security:** The systems have ensured the privacy and security of facial data and attendance information.
* **Bi-directional People Counting:** The system is designed to track the number of people entering and exiting each room, updating the count instantly.
* **Occupancy Monitoring:** Provided real-time data on the number of people currently inside a room.
* **Multi-Entry Support:** The system can handle multiple entry and exit points simultaneously.

**Design Documentation:**

### ****Introduction to Design Documentation****

### ****Flow chart:****

### 

* **Scope:** We are setting up to design a system comprising of two modules. The first module (face detector) is a mobile component, which is basically a camera application that captures student faces and stores them in a file using computer vision face detection algorithms and face extraction techniques. The second module is a desktop application that does face recognition of the captured images (faces) in the file, marks the students register and then stores the results in a database for future analysis.

### ****System Architecture****

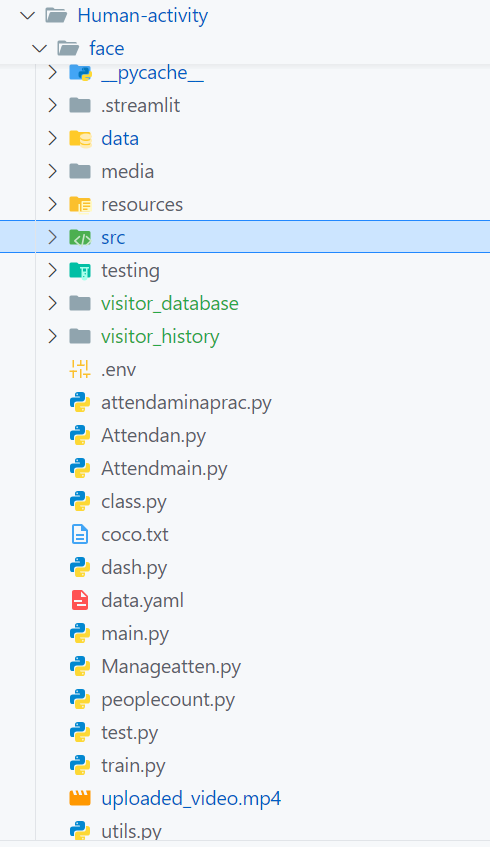
* **Placement and Setup:**
  + Cameras are strategically placed at entry and exit points of rooms or buildings to capture clear images of individuals as they move through these areas. The placement ensures that every person entering or exiting is within the camera's field of view.
  + Depending on the environment (e.g., school, hospital, office), cameras may be installed at different heights and angles to optimize facial recognition and minimize blind spots.
* **Facial Recognition:**
  + The facial recognition module uses advanced algorithms such as Face Net or Deep Face to analyze the captured images and identify individuals based on their facial features. This module is designed to work with high accuracy, even in challenging conditions like poor lighting or varying facial expressions.
  + The module includes pre-processing steps, such as image normalization and alignment, to enhance recognition accuracy. It also handles scenarios like partial face visibility and faces covered by masks or glasses.
* **People Counting:**
  + The people counting module employs object detection and tracking algorithms, such as YOLO (You Only Look Once) and SORT (Simple Online and Real-time Tracking), to count the number of people entering and exiting a room or building.
  + This module tracks individuals as they move across the camera's field of view, ensuring accurate counts by accounting for entry and exit movements. It is designed to handle complex scenarios, such as overlapping individuals or simultaneous movements.
* **Algorithms Used:**
  + **Facial Recognition:** Deep learning models like Face Net for embedding generation and SVM (Support Vector Machine) for classification.
  + **People Counting:** YOLOv10n for real-time object detection, and SORT for tracking the detected objects across frames.
* Overview **and Interaction:**
* The UI Dashboard is the primary interface through which users interact with the system. It provides a user-friendly, intuitive platform for managing and monitoring the system’s operations.
* **Real-Time Monitoring:**
* Users can monitor the current status of rooms or buildings, viewing live counts of people inside, as well as any recent entries or exits. The dashboard also provides real-time facial recognition results, showing who has been identified by the system.
* **Reporting Features:**
* The dashboard offers powerful reporting tools, allowing users to generate reports on attendance patterns, peak occupancy times, and more. These reports can be customized to meet specific needs and can be exported in various formats for further analysis or record-keeping.

#### ****Database Design:****

* + **User Profiles Table:** Stores information about each individual recognized by the system, including their unique ID, name, facial embedding.
  + **Attendance Records Table:** Logs each recognition event, including the individual’s ID, timestamp, and location of recognition. This table is crucial for tracking attendance over time.
  + **People Count Logs Table:** Maintains records of the number of individuals entering and exiting each monitored area, along with timestamps and any relevant location data.
  + **Encryption:** All sensitive data, including facial embeddings and personal information, is encrypted both at rest and in transit. Encryption methods like AES (Advanced Encryption Standard) are employed to ensure data security.
  + **Access Control:** Access to the database is controlled through role-based access controls (RBAC), ensuring that only authorized personnel can view or modify sensitive data. Additionally, the system logs all access attempts and changes to the database, providing an audit trail for security purposes.

**Implementation Details:**

All our code is written in Python language. First here is our project directory structure and files.

****

[Main] => It contains main page .

[Dash] => It contains dashboard & contents.

[Attendmain] => It contains database related.

[Attendan] => It contains take attendance.

[Class] => It contains pre trained models of yolo.

[Manageatten] => It contains all train & test of face attend.

[Peoplecount] => It contains people counting & its components

[Test] => It contains all testing phrases of Face attendance.

[Train] => It contains all training.

**Technology Stack:**

* **Frontend:** Streamlit for creating the web-based interface.
* **Backend:** Python for server-side processing and facial recognition.
* **Facial Recognition Libraries:** OpenCV, dlib, and face recognition for detecting and recognizing faces.
* **Database:** SQLite for storing user profiles and attendance records.
* **Deployment Platform:** Streamlit Cloud

**Frontend Implementation:**

**Setup:** Install Streamlit (pip installs Streamlit).

**Code Structure:** A Python file (main.py) for the Streamlit app.

**UI Components:** login forms, dashboards, and report generation features, main menu and much more

**Backend Implementation:**

* **Facial Recognition:**
  + **Capture Video Feed:** Using OpenCV to access the camera and capture images.
  + **Process Frames:** Detect and recognize faces using the face recognition library.
  + **Store and Match Faces:** Save facial feature vectors in the database and compare them to identify individuals.
* **Database Management:**
  + **Schema Design:** Defining tables for users and attendance records.
  + **Operations:** Implement functions to add, update, and retrieve data from the database.

**Authentication and Access Control:**

* **Secure Login:** Implemented login functionality with password protection.
* **Role-Based Access:** Only admins can access system features.

**Deployment:**

* **Local Deployment:** Testing the application locally using Streamlit (Streamlit run app.py).
* **Cloud Deployment:** Deploying the app on Streamlit Cloud.

**Security Measures:**

* **Data Encryption:** Encrypt sensitive data like facial features and attendance records.
* **Secure Access:** Using secure authentication methods for admin access.

**People Counting Process:**

* **Object Detection:** The system detects human figures in the camera feed using models like YOLO.
* **Tracking:** Detected individuals are tracked as they move across the camera’s field of view. Algorithms like SORT (Simple Online and Real-time Tracking) or Deep SORT are employed to maintain consistent tracking.
* **Entry/Exit Counting:** The system counts individuals as they cross virtual lines or zones set up at entry and exit points. This count is dynamically updated to reflect the current occupancy.

**Testing:**

**Face Recognition Accuracy Testing**

* **Objective:** To ensure the system correctly identifies and verifies faces.
* **Test Cases:**
  + Testing the system with various facial angles, lighting conditions, and expressions.
  + Test recognition with and without accessories like glasses, hats, or masks.
  + Check for false positives (incorrectly identifying a face) and false negatives (failing to recognize a registered face).
  + Anti – Spoofing

**People Counting Algorithm:**

* **Objective:** To ensure the people’s counting accuracy.
* **Test Cases:**
* Test the counting accuracy with different crowd sizes, entry/exit speeds, and movements.
* Ensure that the system correctly tracks individuals entering and exiting simultaneously without duplicating counts.

**Performance Testing**

* **Objective:** To evaluate the system's speed and efficiency under different conditions.
* **Test Cases:**
  + Measure the time taken for face recognition under normal and peak usage.
  + Tested system performance with a large number of users in the database.
  + Checked system response time when multiple users attempt to log attendance.

**Security Testing**

* **Objective:** To ensure that the system is secure and that sensitive data is protected.
* **Test Cases:**
  + Tested authentication processes to ensure only authorized users can access the system.
  + Verify that data transmission between the client and server is encrypted.
  + Test the system’s resistance to spoofing attempts (e.g., using photos or videos for face recognition).

**User Interface (UI) Testing:**

* **Objective:** To ensure that the system's UI is user-friendly and functions as expected.
* **Test Cases:**
  + Verify that all UI elements (buttons, forms, dashboards) function correctly.
  + Test the system's usability on different devices (desktops, tablets).
  + Check that the system provides clear feedback for successful and failed actions.
  + Test the responsiveness of the interface, ensuring it adjusts correctly to different screen sizes.

**Database Testing:**

* **Objective:** To verify that the database operations are functioning correctly.
* **Test Cases:**
  + Test the accuracy of data entry, updates, deletions, and face enroll.
  + Verify that attendance records are correctly stored, updated, and retrieved.
  + Check database performance under high load.

**Integration Testing:**

* **Objective:** To ensure that all system components work together seamlessly.
* **Test Cases:**
  + Verified that the facial recognition module correctly interacts with the database.
  + Tested the integration between the frontend (Streamlit) and backend (Python) components.
  + Check that the system correctly logs attendance and generates reports as expected.

**Reliability and Availability Testing**

* **Objective:** To ensure the system remains functional over time and under different conditions.
* **Test Cases:**
  + Test the system’s ability to run continuously without failure.
  + Verify that the system can recover from unexpected failures or crashes.
  + Check the system's availability under different network conditions.

**Error Handling:**

* **Objective:** To ensure the system handles network issues & backup.
* **Test cases:**
  + - Tested how the system handles situations such as network interruptions, camera malfunctions, or database connection issues.
    - Verified that the system provides appropriate error messages and recovers gracefully from failures.

**Deployment:**

**Introduction to Streamlit:**

Streamlit is an open-source Python library that allows you to create and deploy data-driven applications quickly and easily. Streamlit is ideal for deploying the face attendance system due to its simplicity, interactivity, and ability to integrate seamlessly with Python's data processing and machine learning libraries.

**Developing the Application:**

* Create a virtual environment to manage these dependencies and prevent conflicts.
* **User Interface (UI) Design:** Utilizing Streamlit's widgets like buttons, sliders, and file uploaders to create an interactive UI for the face attendance system.
* **Integrating Facial Recognition:** Incorporate the facial recognition module into the Streamlit app to capture images, process them, and match them with stored data.
* **Data Management:** Using Python’s database libraries SQLite3 to store and manage attendance data securely.

**Testing Locally:**

* Before deployment, thoroughly tested the application locally to ensure all components (UI, facial recognition, and database) work as expected.
* Use Streamlit’s built-in tools to test different user interactions and application behavior.

**Deploying the Application:**

* **Deployment Options:**
  + **Streamlit Sharing:** A quick and easy way to deploy your Streamlit app for free, directly from a GitHub repository.
  + **Cloud Platforms:** Deploy on cloud services like AWS, or Google Cloud Platform (GCP) for more control and scalability.

**Post-Deployment Support:**

* **Ongoing Maintenance:**
  + Schedule regular maintenance tasks, such as software updates, hardware checks, and database backups, to keep the system running smoothly and securely.
  + Address any issues that arise promptly, including bug fixes, performance optimizations, and hardware replacements.
* **User Support:**
  + Provide ongoing support to users, offering a helpdesk or ticketing system to handle any queries, troubleshooting, or feature requests.
  + Collect user feedback regularly to identify areas for improvement and to plan future enhancements.

**Future Scope:**

**Advanced AI Integration:**

* Enhance the facial recognition system with more sophisticated AI models for improved accuracy and faster processing, even in challenging environments (e.g., low light or different angles).

**Cross-Platform Accessibility:**

* Develop mobile and desktop applications to complement the web-based system, enabling users to access attendance data and admin features across multiple devices.

**Scalability:**

* Expand the system to support larger institutions with thousands of users, incorporating cloud-based solutions for distributed processing and data storage.

**Additional Biometric Features:**

* Integrate other biometric methods like fingerprint or iris recognition to create a multi-factor authentication system, enhancing security.

**Integration with Existing Systems:**

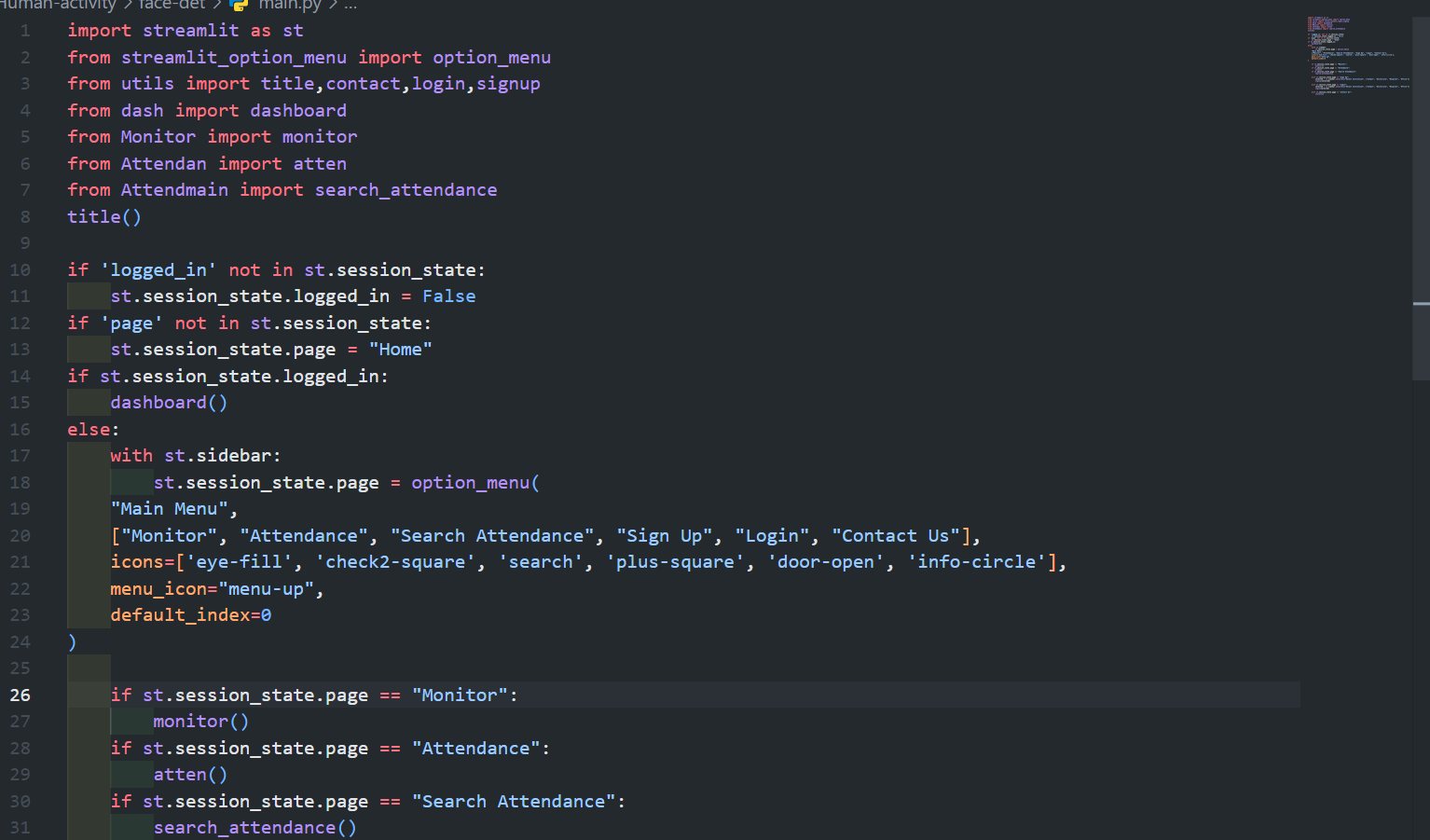
* Expand compatibility with existing HR, payroll, and academic management systems to streamline operations and automate processes like payroll calculation or student grading.

### Conclusion:

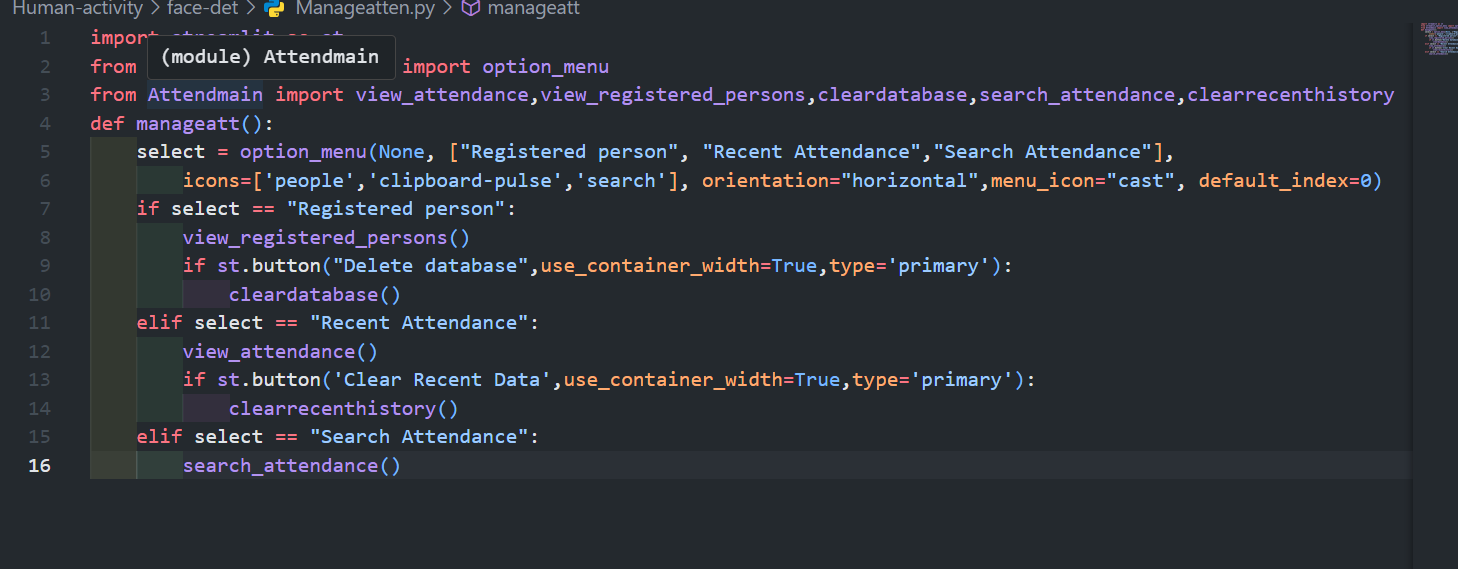
The development of a face attendance system with people counting features represents a significant advancement in managing attendance and monitoring occupancy across diverse environments such as schools, universities, offices, and hospitals. By automating attendance tracking and providing real-time insights into room occupancy, this system addresses the limitations of traditional methods, offering enhanced accuracy, efficiency, and security. The implementation of advanced facial recognition algorithms, coupled with reliable people counting technology, enables organizations to streamline operations, optimize resource utilization, and ensure compliance with safety protocols. The system's adaptability to different environments and its ability to scale make it a versatile solution suitable for a wide range of applications.

In conclusion, the face attendance system with people counting features not only meets current needs but also offers a pathway to future enhancements that will continue to drive improvements in security, efficiency, and user experience across various industries.

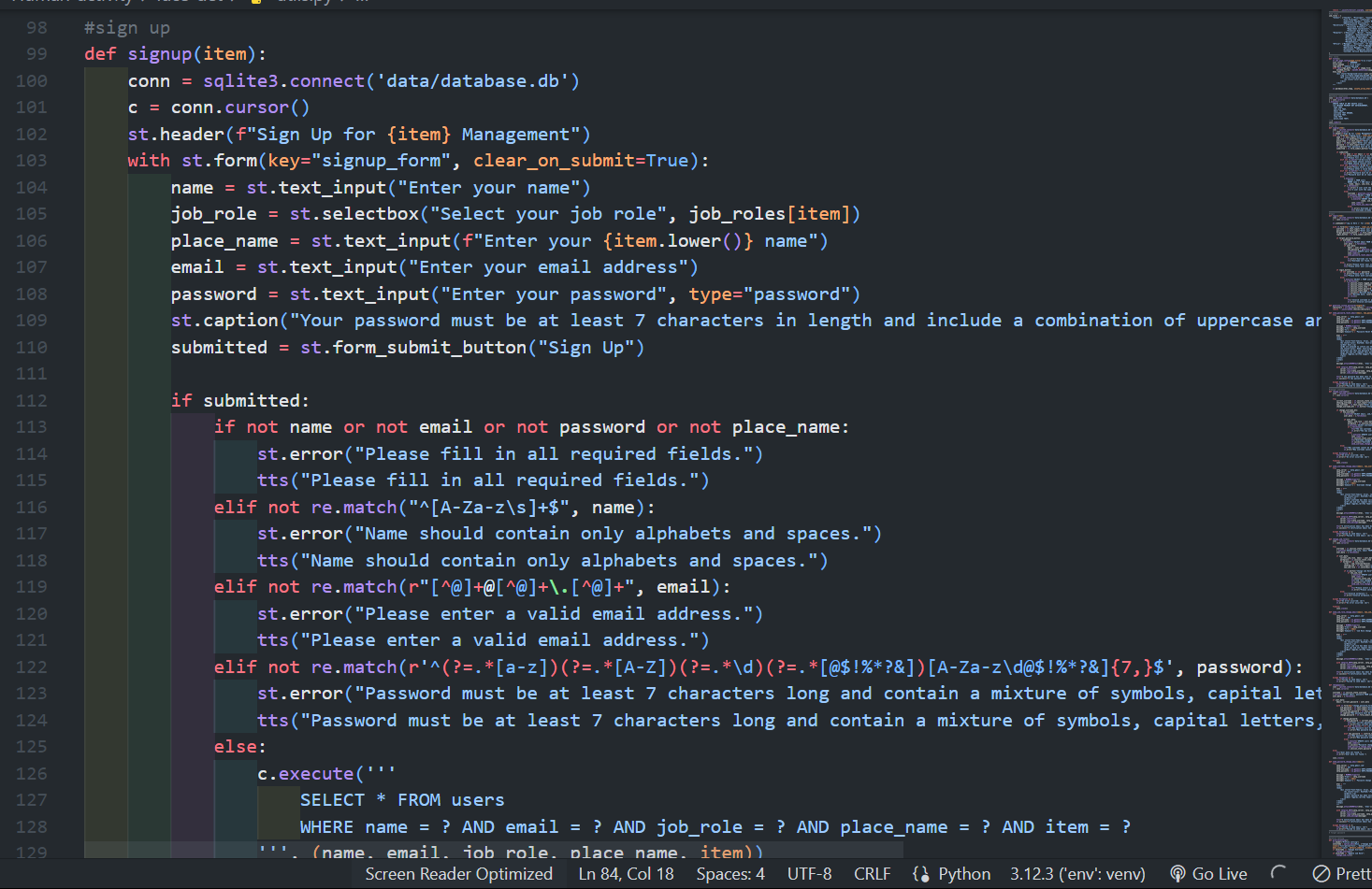
**Appendix – A:**

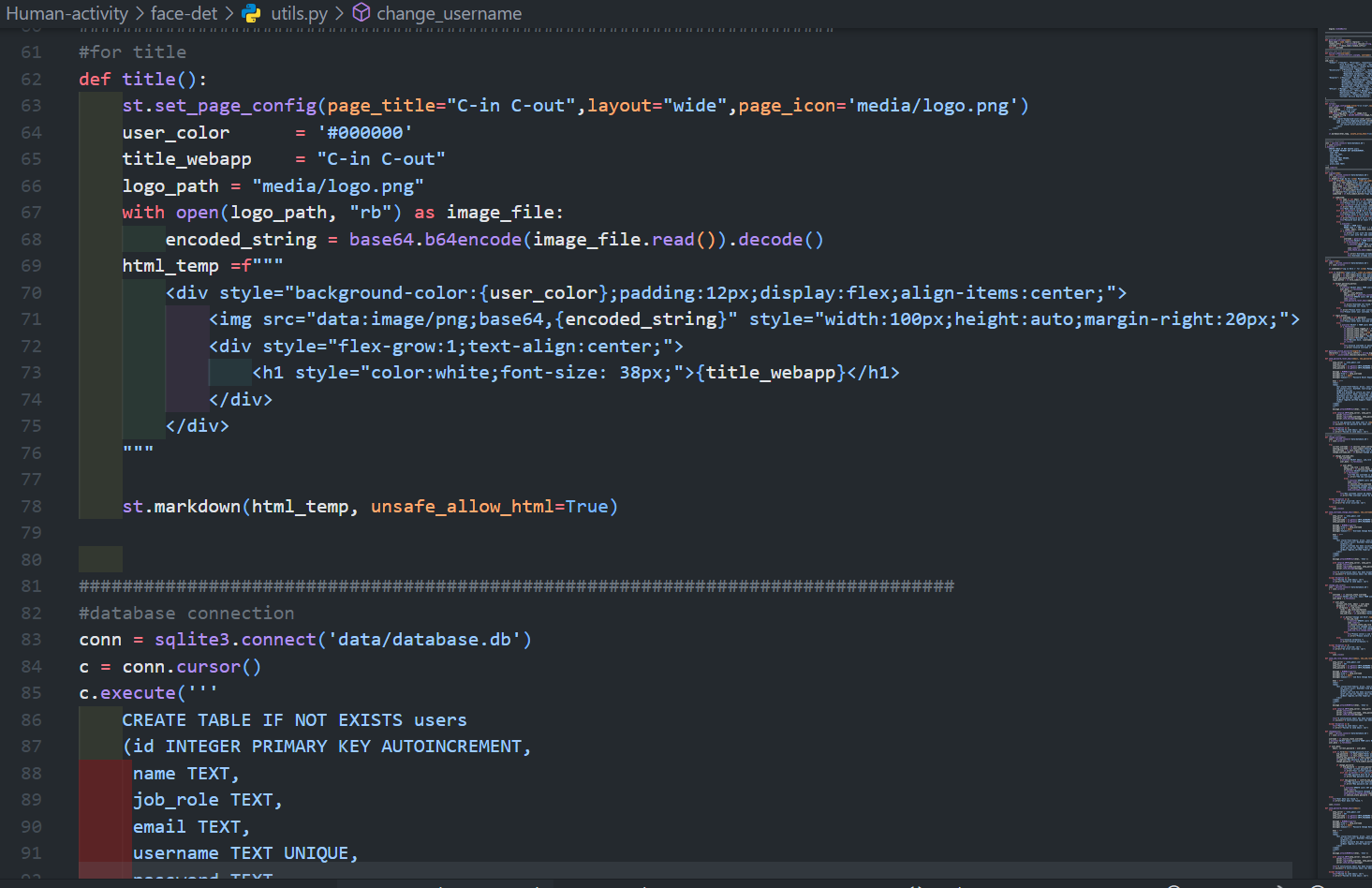


[ Main.py file ]

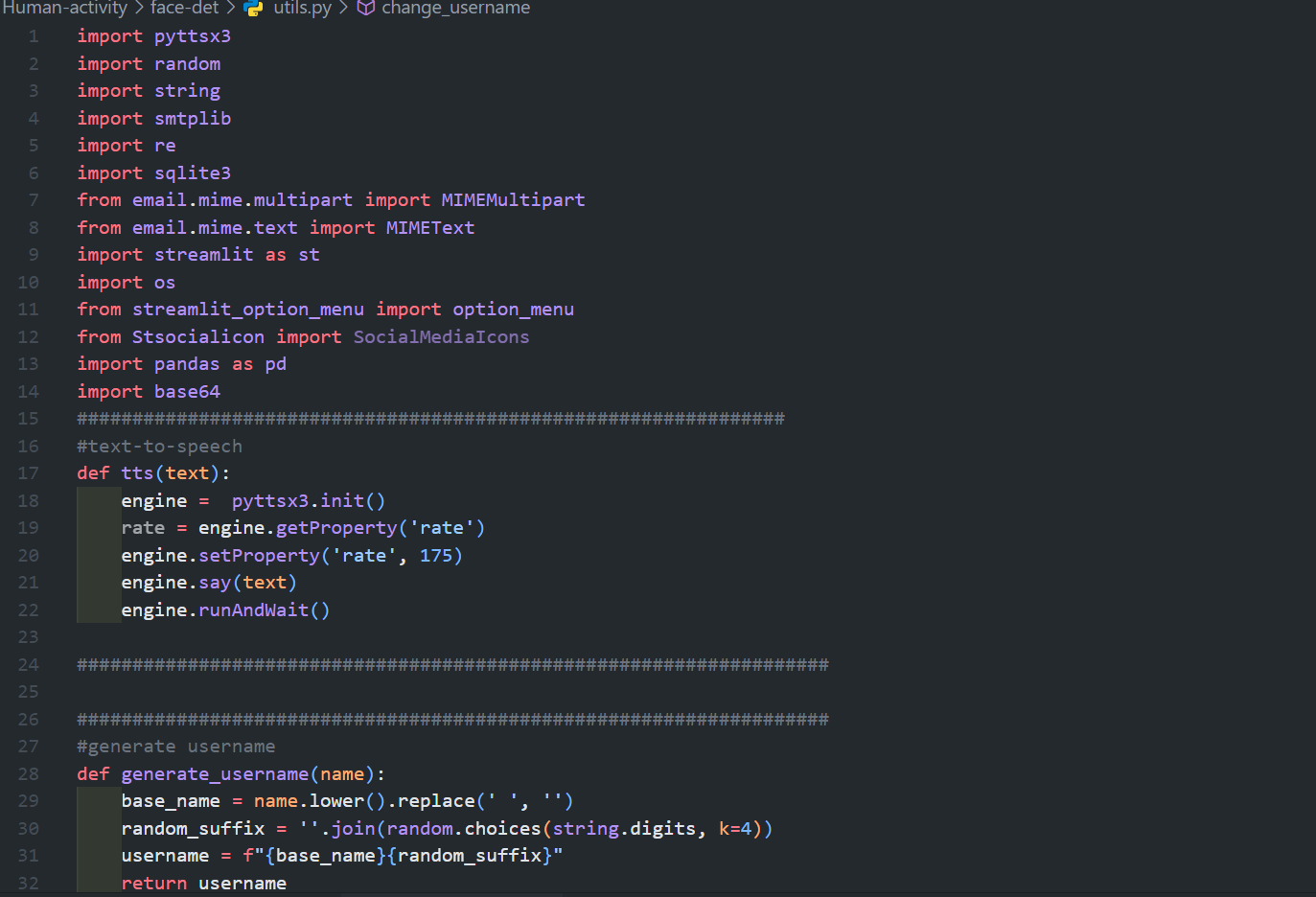


[Manageatten.py file]

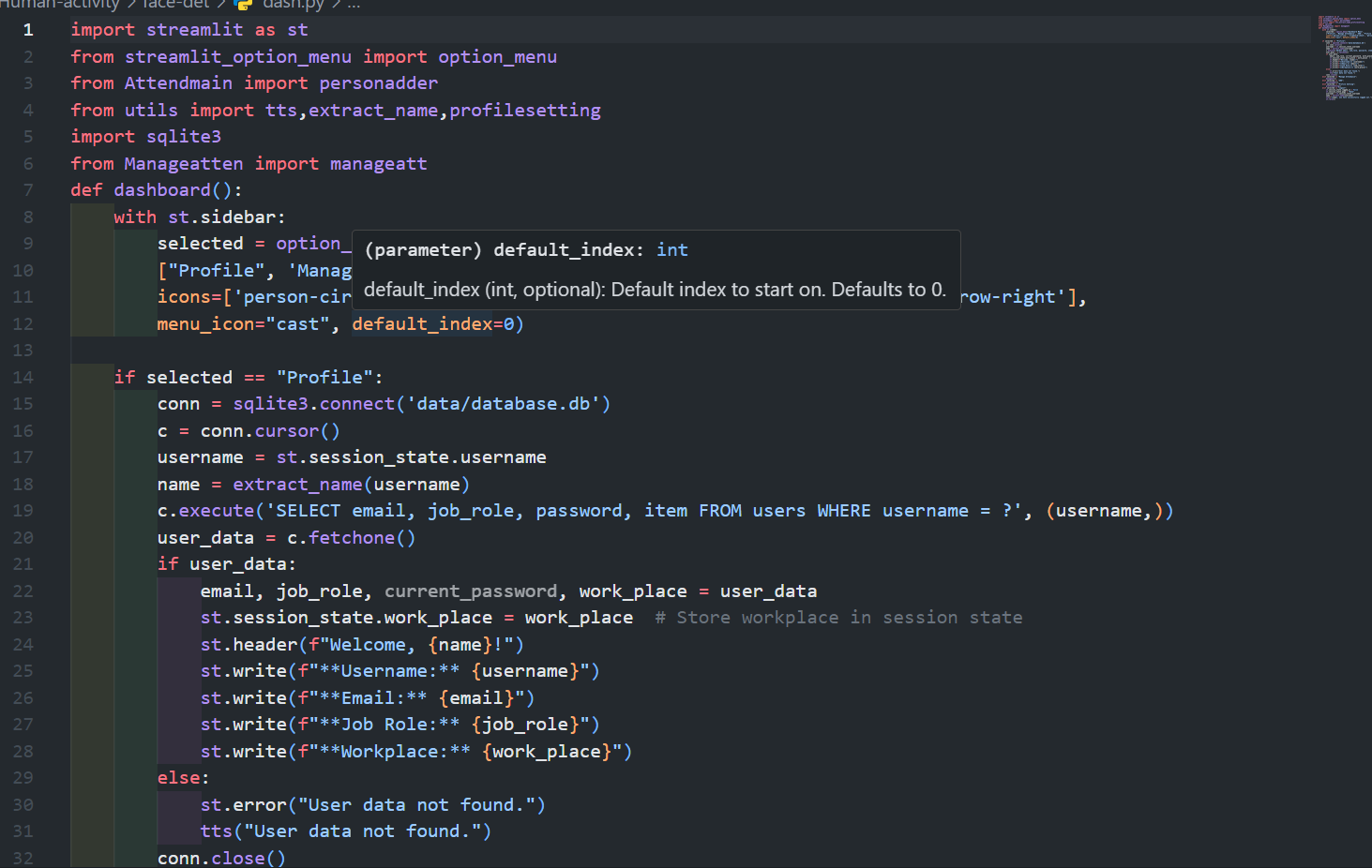


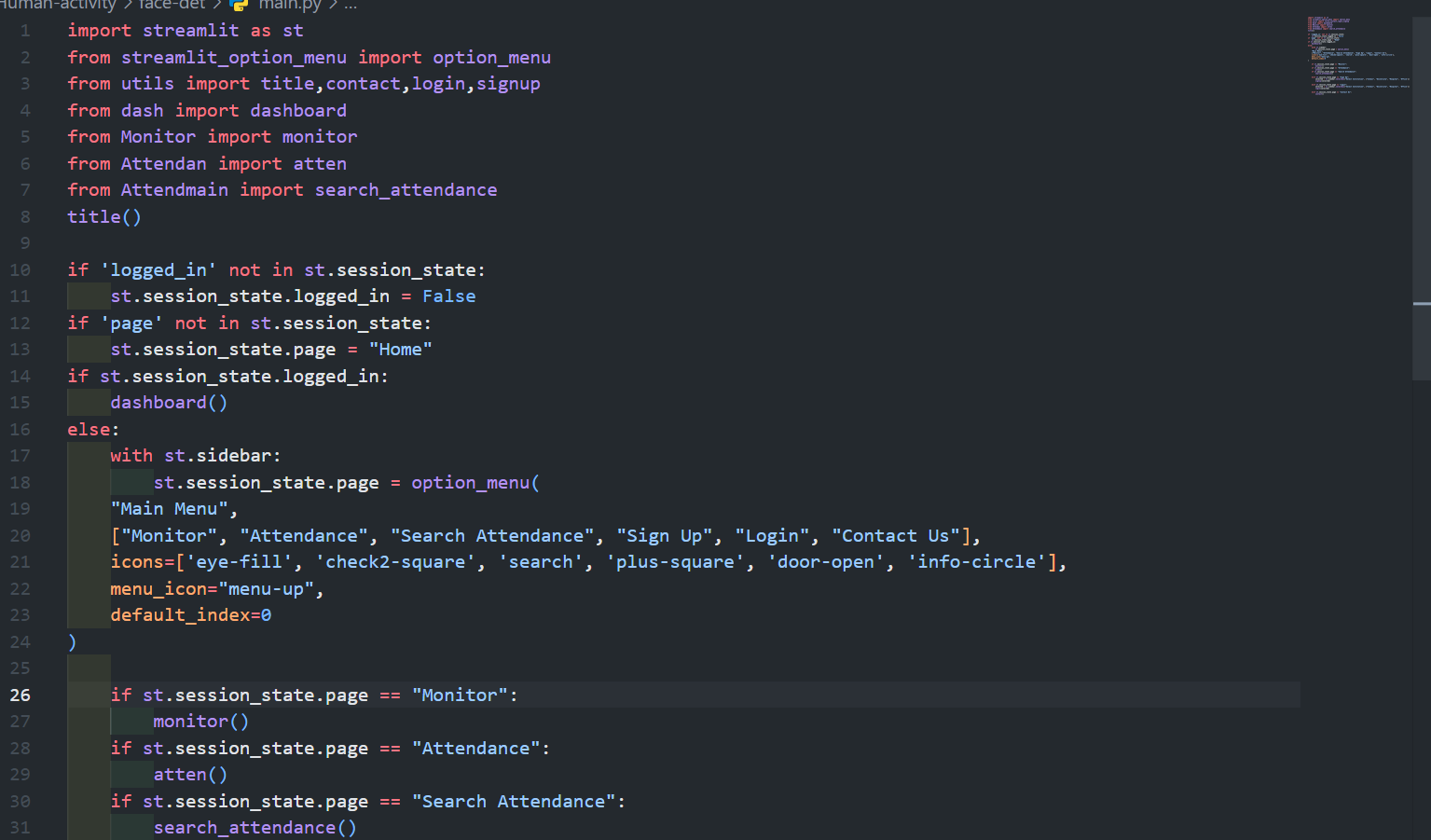
****

**[Utils.py]**

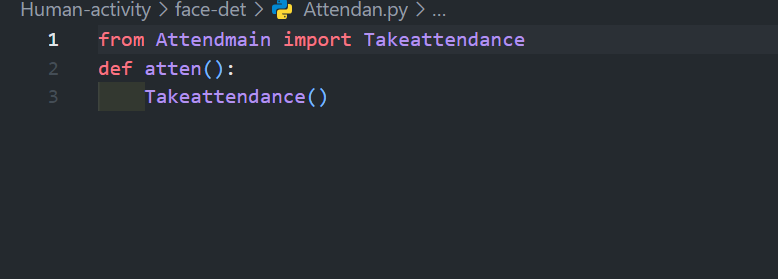
****

**[Utils.py file]**

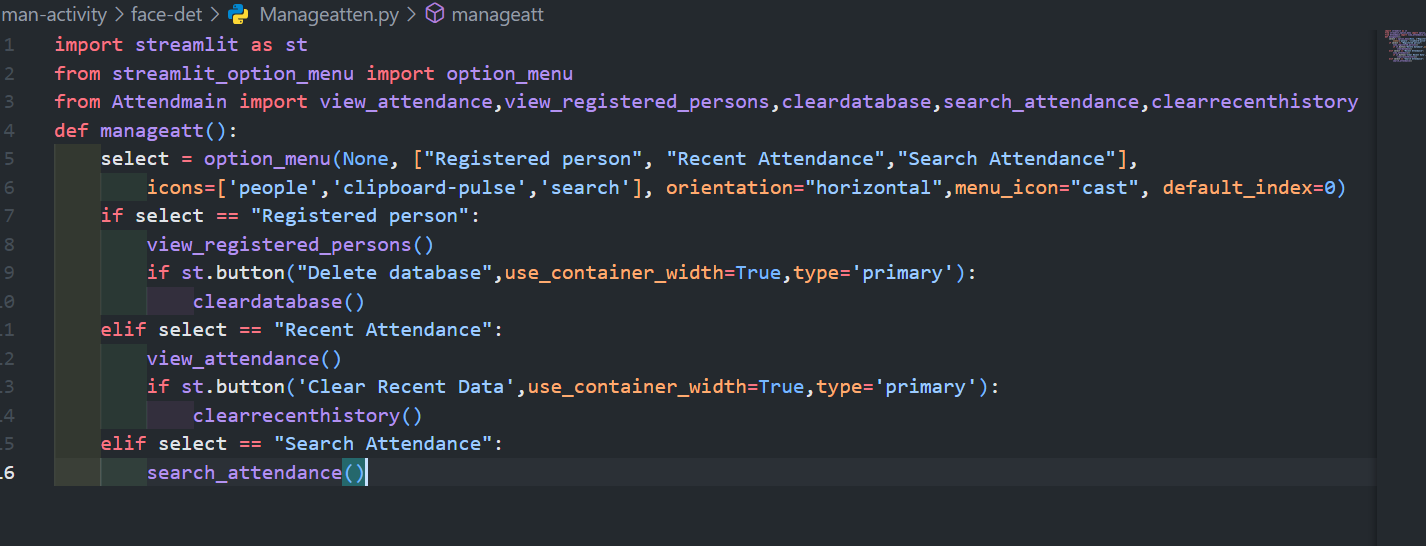
****

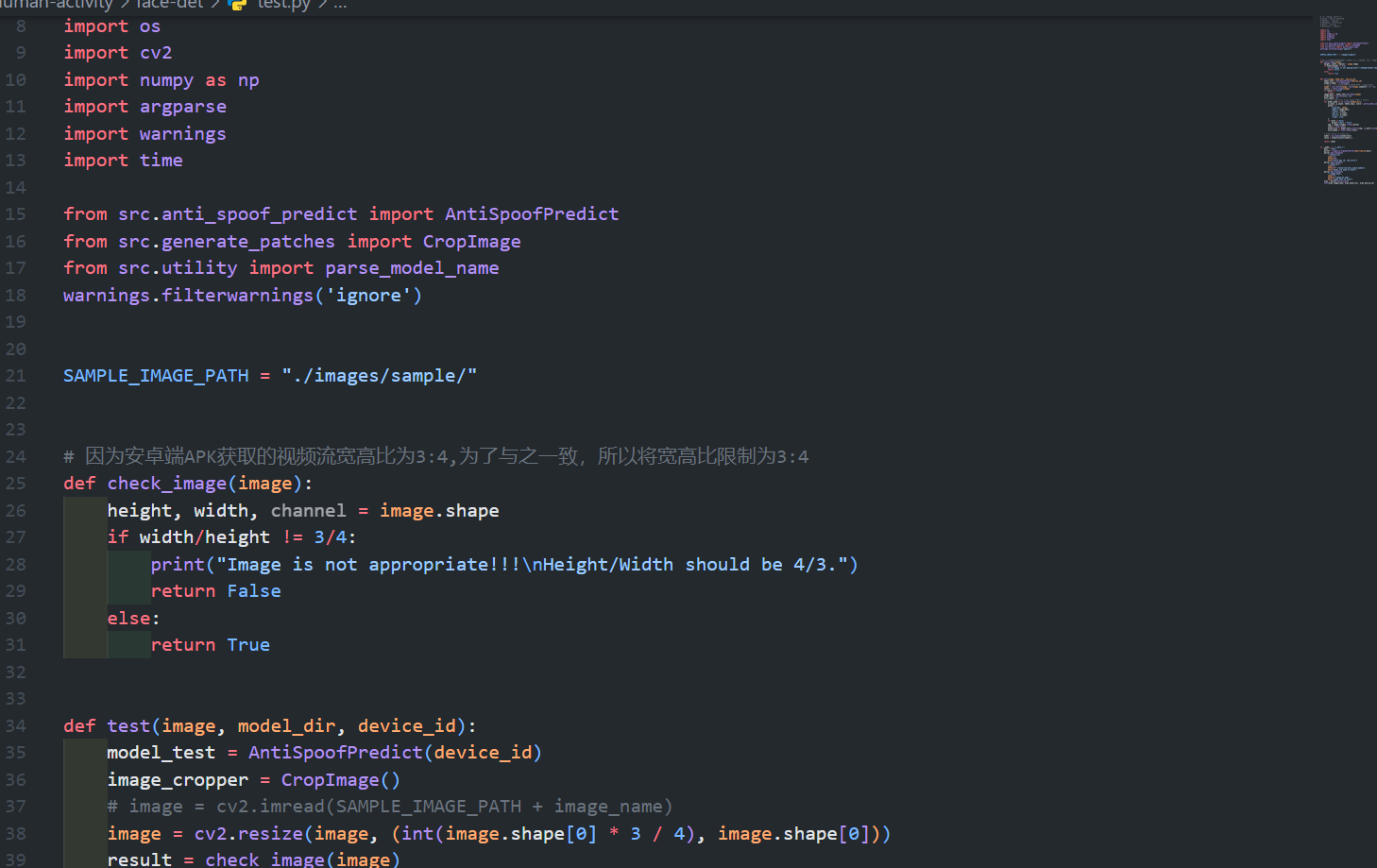
****

**[main.py]**

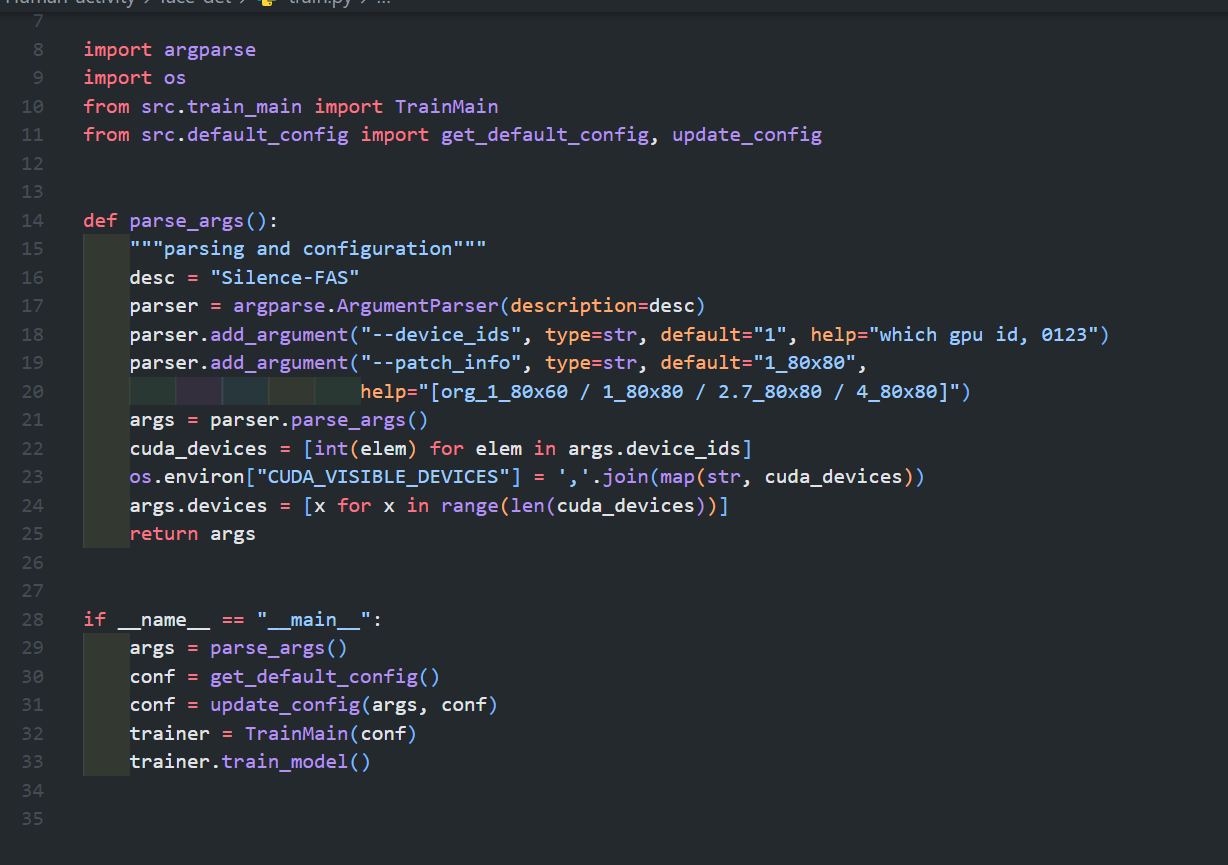
****

**[Attendan.py]**

**[Manageatten.py]**

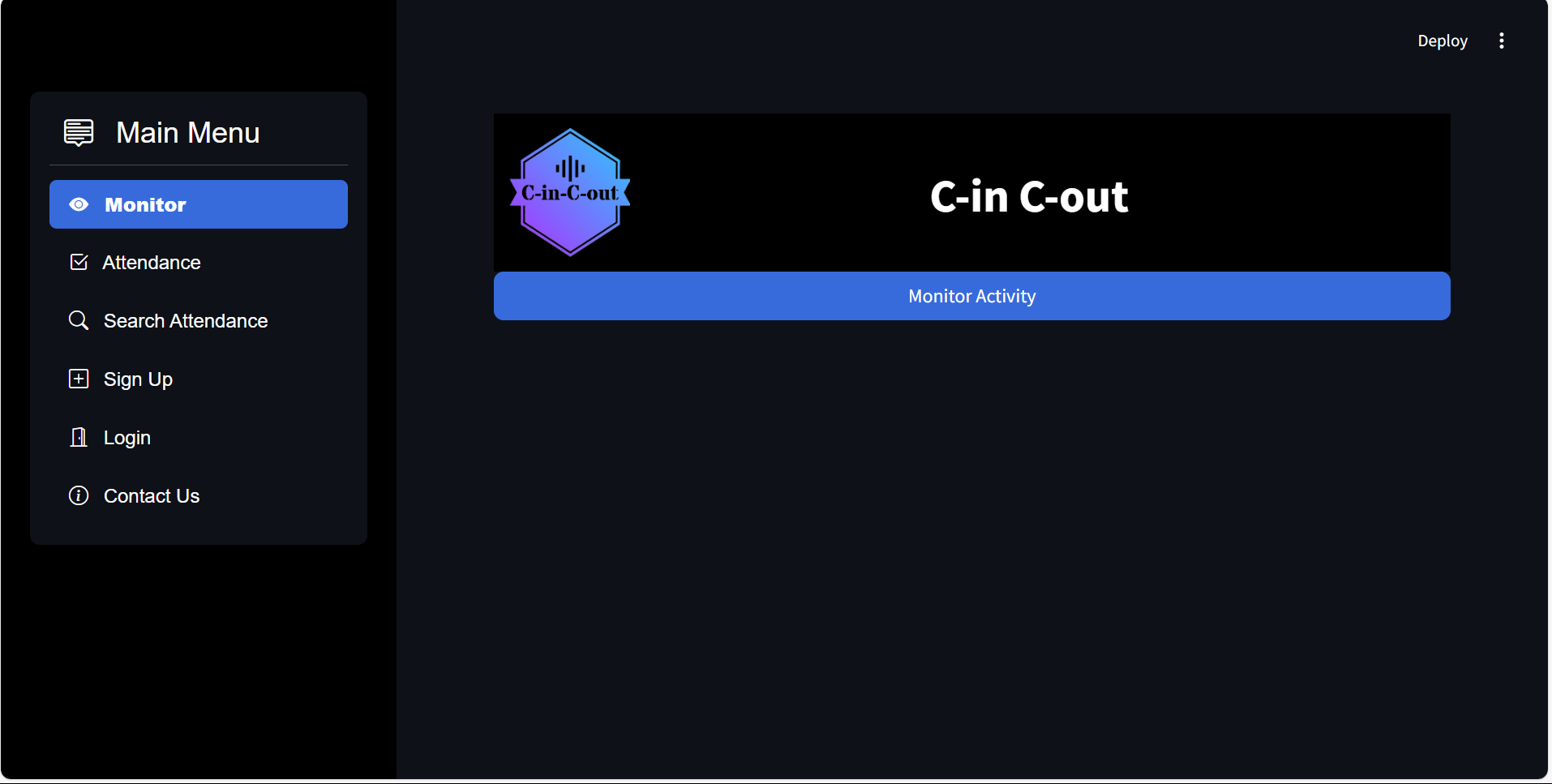
****

**[test.py]**

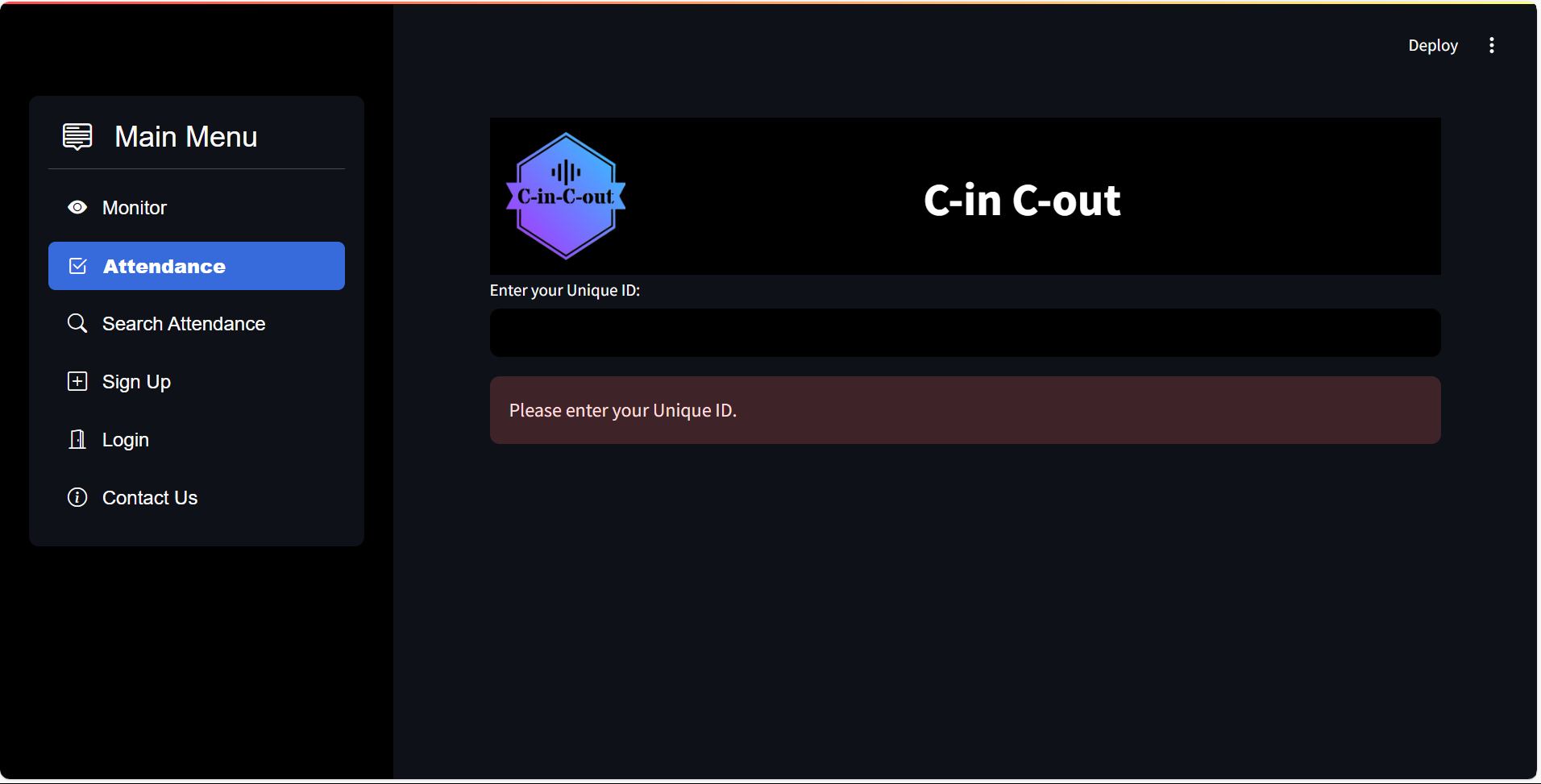
****

**[train.py]**

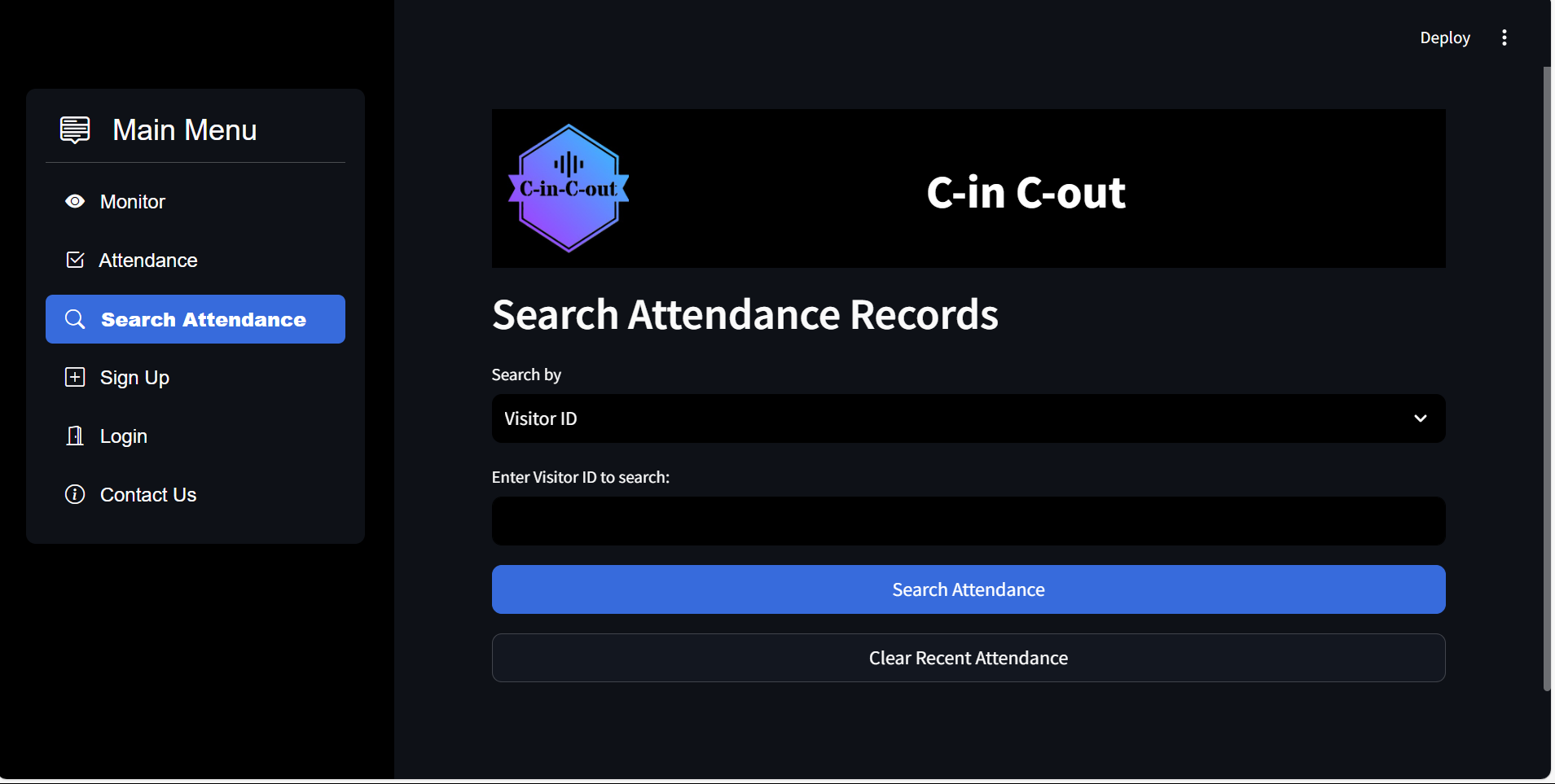
**Appendix – B:**

****

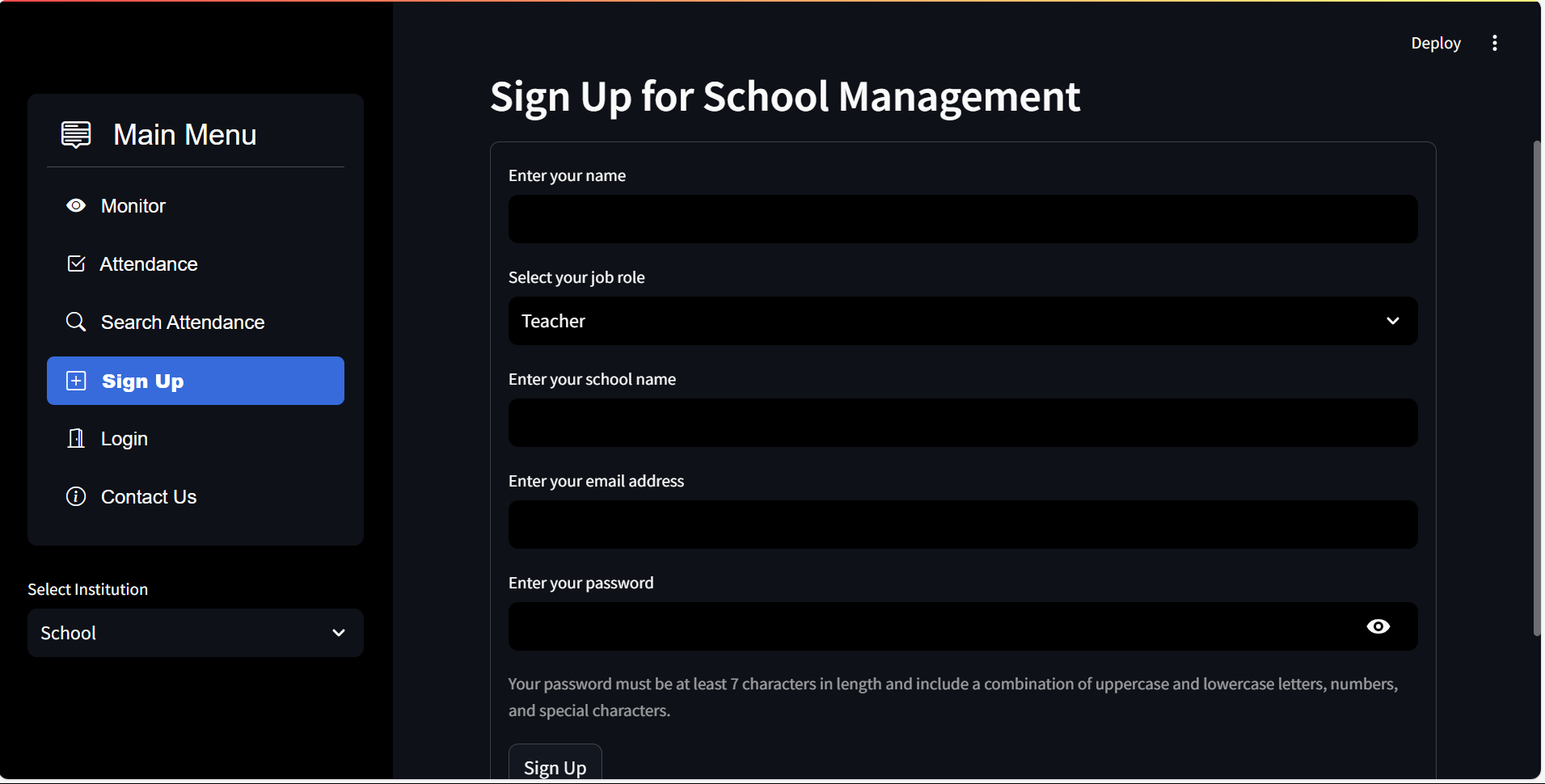
**[Monitor page to monitor the activity]**

****

**[Attendance page to mark the attendance by uniqueID]**

****

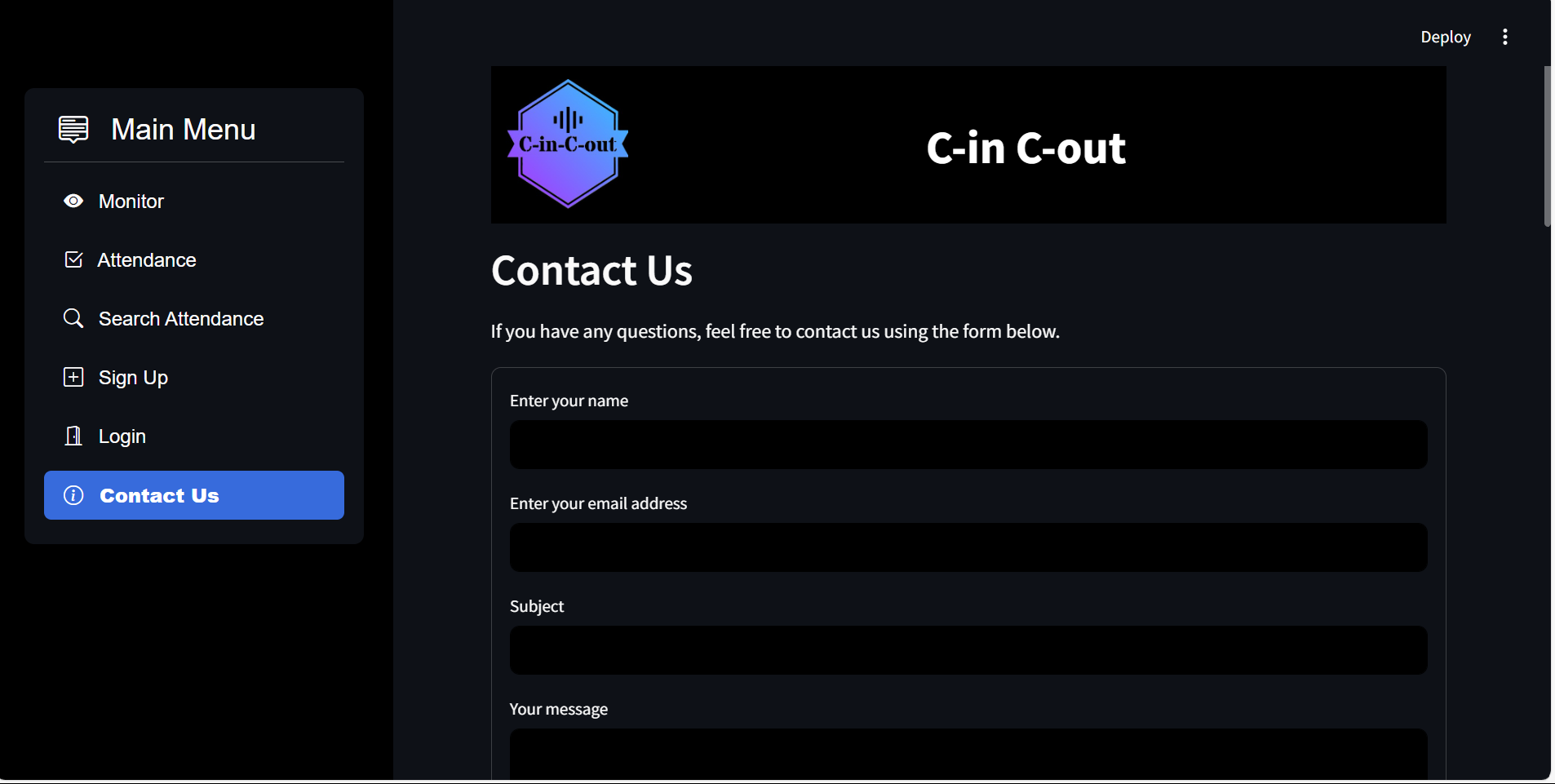
**[To search the attendance]**

****

**[Signup page to signup according to their management]**

****

**[Login according to their management]**

****

**[Contact us form to contact for queries]**