**AMRITSAR GROUP OF COLLEGES**

Autonomous status conferred by UGC under UGC act-1956, (21), NAAC-A Grade (Formerly Known as Amritsar College of Engineering & Technology | Amritsar Pharmacy College)

INSTITUTIONAL TRAINING PROJECT REPORT

On

**"FACE DETECTION(ATTENDANCE) APP"**

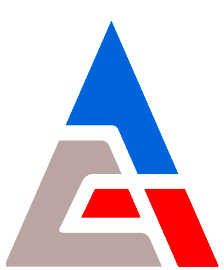
Submitted in the Partial fulfilment of the requirement for the Award of Degree of

**Bachelor of Technology**

In

**COMPUTER SCIENCE & ENGINEERING**

Batch (2023-2027)

****

**Submitted To:** **Submitted by:**

Ms. Shagun Arora Manish Kumar(2333355)  
 Kaushlendra Kumar(2333338)  
 Kundan Kumar (2333345)  
 Kundan Kumar Sah (2333346)

**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**

**Amritsar Group of Colleges, Amritsar**

**ACKNOWLEDGEMENT**

It is our proud privilege to express our heartfelt gratitude to several individuals who have assisted us, directly or indirectly, in conducting this Face Attendance Detection Project. We would like to extend our sincere thanks to my faculty guide, Ms. Shagun Arora, Assistant Professor, Department of computer science engineering, Amritsar group of college, Amritsar Punjab, for their invaluable guidance and inspiration in completing this project.

We are extremely grateful to Dr Sandeep Kad, Head of Department, and all faculty members of the computer science engineering at Amritsar group of college for their cooperation, kind guidance, and encouragement throughout this project.

We would also like to thank all our friends who have contributed to the preparation of this project report. We will always be indebted to them for their support. This project completion has indeed helped us explore new knowledge avenues related to OpenCV and C++, and we are confident it will benefit us in our future endeavors.

**DECLARATION**

We hereby declare that the project work entitled "FACE ATTENDENCE DETECTION APPLICATION" is an authentic record of our own work carried out as per the requirements of the Object-Oriented Programming System Lab for the award of the degree of B.Tech (CSE), Amritsar Group of Colleges, Amritsar, under the guidance of Ms. Shagun Arora (Assistant Professor).

Manish Kumar(2333355)  
Kaushlendra Kumar(2333338)  
Kundan Kumar (2333345)  
Kundan Kumar Sah (2333346)

**ABSTRACT**

**Face Attendance Detection Application** serves as a comprehensive The solution for automating attendance tracking using facial recognition technology. The project features two main interfaces: a user interface for students and an admin interface for attendance management. The user interface allows students to register their faces, while the admin interface provides functionalities for managing student records, viewing attendance statistics, and exporting reports.

This project utilizes C++ and incorporates essential libraries such as OpenCV for face detection and recognition, along with iostream and fstream for file handling. Key functionalities include real-time face detection, comparison with reference images, and maintaining attendance records in text files for easy retrieval. The system aims to enhance the efficiency of attendance tracking, reduce manual errors, and ensure a seamless experience for both students and administrators.

**Table of contents**

1. **INTRODUCTION** .................................................................................................1
2. **SCOPE OF PROJECT** ..........................................................................................2
3. **OBJECTIVE** ..........................................................................................................3-4
4. **METHODOLOGY**.................................................................................................5-6
5. **HARDWARE AND SOFTWARE REQUIREMENTS** ......................................7-8
6. **TOOLS USED** .......................................................................................................9
7. **SOURCE CODE** ................................................................................................. 10-21
8. **OUTPUT** ..............................................................................................................22-27
9. **REFERENCES** .....................................................................................................28

**Table of Figures**

* **Figure 1:** Main interface of Application ...................................................................... 22
* **Figure 2:** Register student face pannel.…………...................................................... 22
* **Figure 3:** Camera open for capture face..................................................................... 23
* **Figure 4:** captured face................................................................................................23
* **Figure 5:** Mark Attendance through face…........................................................... 24-25
* **Figure 6:** Update student face ………………………................................................ 26
* **Figure 7:** Delete student face …................................................................................. 26
* **Figure 8:** View registered student ……………………………………....................... 27
* **Figure 9:** Exiting pannel……………………………………..................................... 27

**INTRODUCTION**

The Face Attendance Detection Application is designed to streamline attendance tracking through advanced facial recognition technology. This system enables users to register, log in, and have their attendance marked automatically based on facial recognition. By leveraging OpenCV for face detection, the application ensures accurate identification, making attendance management efficient and reliable.

The user interface of the application provides a seamless experience for students and faculty alike. Users can easily register their facial data, log in to the system, and receive real-time updates on their attendance status. This straightforward process minimizes the hassle of manual attendance marking and promotes a more organized approach to tracking student participation.

On the administrative side, the application equips administrators with the necessary tools to manage user registrations and attendance records effectively. Admins can view attendance reports, track check-ins and check-outs, and maintain a comprehensive list of registered users. This functionality allows for quick access to essential data, enabling informed decision-making and efficient management of attendance records.

The core objective of this application is to enhance the educational experience by providing a reliable and user-friendly solution for attendance management. By utilizing facial recognition technology, the application aims to reduce administrative workload, enhance accuracy, and improve overall engagement in academic settings.

**SCOPE OF PROJECT**

The scope of the Face Attendance Detection Application project encompasses several essential functionalities designed to enhance the efficiency of attendance management in academic or organizational settings. Below is an overview of the project's scope:

1. **User Registration and Login:** Users can register their profiles in the system by providing their personal details and reference images. Once registered, users can log in to access their attendance records and other functionalities.
2. **Face Detection and Recognition**: The application employs advanced face detection and recognition algorithms to identify users during check-ins and check-outs. This feature ensures accurate attendance tracking and minimizes the chances of impersonation.
3. **Attendance Marking:** Registered users can check in and out using the face recognition system. The application automatically logs attendance, allowing for real-time tracking of users present on-site.
4. **Attendance Report Generation:** The system generates attendance reports that detail user attendance over specified periods. Administrators can access these reports to monitor attendance patterns and make informed decisions.
5. **Data Management:** The application securely stores user and attendance data in text files, ensuring easy retrieval and maintenance of historical records. This data management feature is crucial for reviewing attendance trends over time.
6. **User-Friendly Interface:** Both the user and administrator interfaces are designed to be intuitive and straightforward, enhancing the user experience. The application provides a seamless interaction platform for all users.
7. **Data Security:** The project incorporates security measures to protect user information and attendance data from unauthorized access. Ensuring the privacy and integrity of user data is a top priority

**OBJECTIVE**

The primary objectives of the Face Attendance Detection Application project are aimed at improving efficiency, accuracy, and user experience in attendance management. The key goals of the project include:

1. **Automating Attendance Marking**:
   * **Seamless Check-In/Check-Out**: Enable users to mark their attendance through facial recognition technology without the need for manual interaction. This objective focuses on making the attendance process faster, more reliable, and user-friendly.
2. **Ensuring Data Accuracy**:
   * **High Accuracy Face Recognition**: Implement advanced face detection algorithms to ensure that the system accurately recognizes users and marks their attendance. Reducing false positives and negatives is a critical objective to maintain the integrity of attendance records.
3. **Data Security**:
   * **Protecting Sensitive Information**: Securely store user information, including facial data, attendance records, and personal details. Implement encryption and secure storage mechanisms to prevent unauthorized access and ensure user privacy is maintained.
4. **Real-Time Reporting**:
   * **Instant Attendance Logs**: Generate and maintain real-time attendance logs that administrators can access at any time. This feature ensures that attendance data is up-to-date and readily available for analysis, reporting, or auditing.
5. **Efficient Data Management**:
   * **Storing and Accessing Attendance Data**: Ensure that all attendance records are securely saved and can be easily accessed for future reference. This objective emphasizes building a system that efficiently handles large volumes of data without compromising performance.
6. **User-Friendly Experience**:
   * **Simplified User Interface**: Develop an intuitive and easy-to-navigate interface for both users and administrators. This objective is focused on enhancing user experience by simplifying registration, attendance marking, and report generation processes.
7. **System Reliability and Efficiency**:
   * **Minimizing Errors and Downtime**: Ensure that the system is highly reliable, with minimal errors in face detection and recognition. The objective is to reduce system downtime and ensure smooth operation during high usage, such as during large class sessions or events.
8. **Scalability for Future Enhancements**:
   * **Expandable Framework**: Design the system with future scalability in mind, allowing for potential additions such as mobile app integration, real-time notifications, multi-location support, and more advanced analytics.
9. **Improving Institutional Efficiency**:
   * **Streamlined Attendance Management**: Automate the attendance process for institutions, saving time and reducing the workload for teachers or administrators. This objective aligns with enhancing institutional efficiency and accuracy in managing attendance data.

**METHODOLOGY**

The Face Attendance Detection Application is designed using a modular approach, where each module serves a specific function to ensure efficient and reliable performance. The following modules form the core of the system:

1. **User Management**: Handles user registration, login, and session management. This module allows users (students and staff) to securely register their facial data and manage their profiles.
2. **Face Detection and Recognition:** Utilizes the OpenCV library for face detection and recognition tasks. This module is responsible for identifying users in real-time during attendance marking.
3. **Attendance Logging:** Automatically records attendance based on successful face recognition. This module maintains logs of check-ins and check-outs, ensuring accurate attendance data is recorded.
4. **Report Generation:** Generates attendance reports for administrators, summarizing attendance statistics over selected periods. This module enables easy monitoring of attendance patterns**.**
5. **Data Storage:** Manages the storage of user and attendance data in text files. This module ensures data persistence and retrieval, allowing administrators to access historical attendance records.
6. **User Interface:** Provides a graphical interface for users and administrators to interact with the system. This module ensures that the application is user-friendly and intuitive, enhancing user experience.
7. **System Design:** Create the architecture for the application, detailing how each module interacts and the flow of data within the system.
8. **Development Environment Setup:** Set up the necessary tools and libraries, including C++, OpenCV, and any additional libraries required for file handling.
9. **Coding the Modules:** Begin coding each module, focusing on functionality, usability, and integration with other modules. Ensure that the code follows best practices and is well-documented for future reference.
10. **Testing and Debugging**: Conduct extensive testing of each module to identify and fix any bugs or issues. Testing should cover various scenarios to ensure reliability under different conditions.
11. **User Acceptance Testing:** After internal testing, conduct user acceptance testing to gather feedback from potential users and make necessary adjustments to the application based on their suggestions.
12. **Deployment:** Finalize the application and deploy it for use in an educational or organizational setting. Provide user training and support as needed during the initial rollout phase.
13. **Maintenance and Updates:** Establish a plan for ongoing maintenance, bug fixes, and future updates to enhance the application based on user feedback and technological advancements.

**HARDWARE AND SOFTWARE REQUIREMENTS**

To successfully develop and run the **Face Attendance Detection Application**, the following hardware and software components are required:

**Hardware Requirements:**

1. **Computer with Compatible CPU**: A machine with at least a dual-core processor (e.g., Intel Core i3 or equivalent) to handle real-time face detection and recognition tasks.
2. **RAM**: At least 4 GB of RAM to ensure the smooth execution of the application, particularly during the compilation and runtime of C++ code and real-time processing of face data.
3. **Webcam**: A functional webcam or external camera with decent resolution (e.g., 720p or higher) for capturing real-time facial data.
4. **Disk Space**: A minimum of 500 MB of free disk space for storing reference images, attendance logs, and the application itself.
5. **Input/Output Devices**: A keyboard and monitor to interact with the user interface of the application and review outputs.
6. **Graphics Card (optional but recommended)**: A GPU capable of handling image processing tasks for faster face detection and recognition.

**Software Requirements:**

1. **C++ Compiler**: A suitable C++ compiler such as GCC, Visual C++, or Code::Blocks for compiling the application code.
2. **Operating System**: The system can run on any operating system that supports C++ development, such as:
   * **Windows** (7, 8, 10, or later)
   * **Linux** (Ubuntu, Fedora, etc.)
   * **macOS** (10.13 or later)
3. **OpenCV Library**: The OpenCV (Open Source Computer Vision) library for handling image processing, specifically for face detection and recognition. Ensure that the appropriate version of OpenCV is installed and configured with your C++ environment.
4. **C++ Standard Libraries**: Necessary standard libraries required to compile and run the application, including file handling libraries for managing text files where attendance data is stored.
5. **IDE (Integrated Development Environment)**: Any C++ development environment, such as Code::Blocks or Visual Studio, to provide a user-friendly interface for coding, debugging, and running the program.
6. **Text Editor**: A basic text editor (e.g., Notepad++, Sublime Text) for viewing and editing configuration files or attendance logs as needed.
7. **Attendance Log Storage**: The system will store attendance data in text files located on disk, requiring software capable of reading/writing files to the appropriate location.

**TOOLS USED**

1. OpenCV (#include <opencv2/opencv.hpp>): A library for computer vision and image processing. It helps with image handling, video capture, and face detection.
2. **C++ Standard Libraries:**
   * **Input/Output (#include <iostream>):** For standard input (like cin) and output (like cout).
   * **String Manipulation (#include <string>):** For handling text data easily.
   * Dynamic Arrays (#include <vector>): For efficient storage and manipulation of collections.
   * **File System (#include <filesystem**>): For file navigation and manipulation (C++17 feature).
   * **File Handling (#include <fstream**>): For file operations using classes like ofstream (write) and ifstream (read).
3. **Namespaces:**
   * **using namespace std;:** To use standard functions without prefixing them with std::.
   * **using namespace cv;:** To use OpenCV functions directly.
   * **namespace fs = std::filesystem;:** To simplify file system operations.
4. **File Handling Functions:**
   * **ofstream:** For writing to files.
   * **ifstream:** For reading from files.
   * **open():** Opens files with specified modes (e.g., ios::in, ios::out, ios::app).
   * **close():** Closes opened files.
   * **getline():** Reads a line from the input stream.

**SOURCECODE**

//|-----------------------------------------------------------------------------------------------------|

//| Welcome To My Face Attendance (Detection) App                                   |

//|-----------------------------------------------------------------------------------------------------|

#include <opencv2/opencv.hpp>

#include <iostream>

#include <string>

#include <vector>

#include <filesystem>  //[ C++17 or higher required ]

#include <fstream>     //[ For file operations ]

using namespace cv;

using namespace std;

namespace fs = std::filesystem;

//|-----------------------------------------------|

//|Structure to store each student's face and name|

//|-----------------------------------------------|

struct Student {

    string name;

    Mat face;

};

//|-----------------------------------------------|

//| Function to detect a face and return it       |

//|-----------------------------------------------|

Mat detectFace(Mat& img, CascadeClassifier& face\_cascade) {

    vector<Rect> faces;

    Mat gray, faceROI;

    cvtColor(img, gray, COLOR\_BGR2GRAY);  //[ Convert frame to grayscale ]

    face\_cascade.detectMultiScale(gray, faces, 1.1, 4);  //[ Detect faces ]

    if (!faces.empty()) {

        faceROI = gray(faces[0]);  //[ Get the first detected face ]

    }

    return faceROI;

}

//|---------------------------------------------------|

//|Function to capture and save multiple student faces|

//|---------------------------------------------------|

void captureReferenceFaces(VideoCapture& cap, CascadeClassifier& face\_cascade, vector<Student>& students) {

    string folderPath = "C:\\Users\\manis\\Downloads\\reference\_faces\\";

    //|--------------------------------------|

    //|Create the folder if it does not exist|

    //|--------------------------------------|

    if (!fs::exists(folderPath)) {

        fs::create\_directory(folderPath);

    }

    int n;

    cout << "|-------------------------------------|\n";

    cout << "|Enter number of students to register:|\n";

    cout << "|-------------------------------------|\n";

    cin >> n;

    for (int i = 0; i < n; i++) {

        string name;

        cout << "[Enter name of student: ]" << (i + 1) << ": ";

        cin >> name;

        Mat frame, faceROI;

        namedWindow("Capture Student Face", WINDOW\_AUTOSIZE);  //[ Create window]

        while (true) {

            cap >> frame;  //[ Capture frame from camera]

            faceROI = detectFace(frame, face\_cascade);

            if (!faceROI.empty()) {

                resize(faceROI, faceROI, Size(100, 100));  // [Resize to standard size for storage]

                //|---------------------------------|

                //|Save the face image to the folder|

                //|---------------------------------|

                string filePath = folderPath + name + ".jpg";

                imwrite(filePath, faceROI);

                students.push\_back({ name, faceROI });  // [Save the student face and name in memory]

                cout << "|Captured and saved face for |" << name << ".\n";

                break;

            }

            imshow("Capture Student Face", frame);  //[ Show the frame with face detection]

            if (waitKey(300) >= 0) break;  // [Exit on key press]

        }

        destroyWindow("Capture Student Face");  // [Destroy window after capturing face]

    }

}

//|----------------------------------------------------------|

//|Function to load reference faces from the folder |

//|----------------------------------------------------------|

void loadReferenceFaces(vector<Student>& students) {

    string folderPath = "C:\\Users\\manis\\Downloads\\reference\_faces\\";

    //[ Load all images from the folder]

    for (const auto& entry : fs::directory\_iterator(folderPath)) {

        string filePath = entry.path().string();

        Mat face = imread(filePath, IMREAD\_GRAYSCALE);

        if (!face.empty()) {

            string name = entry.path().stem().string();  //[ Extract name from file path]

            resize(face, face, Size(100, 100));  // [Resize to standard size]

            students.push\_back({ name, face });

        }

    }

}

//|----------------------------------------------------------|

//|Function to compare faces and mark attendance |

//|----------------------------------------------------------|

void markAttendance(VideoCapture& cap, CascadeClassifier& face\_cascade, vector<Student>& students) {

    string attendanceFolderPath = "C:\\Users\\manis\\Downloads\\attandance\\";

    //|-------------------------------------------|

    //| Create the folder if it does not exist|

    //|-------------------------------------------|

    if (!fs::exists(attendanceFolderPath)) {

        fs::create\_directory(attendanceFolderPath);

    }

    //|-------------------------------|

    //| Find a unique filename   |

    //|-------------------------------|

    int fileCounter = 1;

    string attendanceFilePath;

    do {

        attendanceFilePath = attendanceFolderPath + "attendance\_" + to\_string(fileCounter) + ".txt";

        fileCounter++;

    } while (fs::exists(attendanceFilePath));  // [Check if the file already exists]

    ofstream attendanceFile(attendanceFilePath);

    if (!attendanceFile.is\_open()) {

        cout << "|-----------------------------|\n";

        cout << "|Error opening attendance file|\n";

        cout << "|-----------------------------|\n";

        return;

    }

    namedWindow("Mark Attendance", WINDOW\_AUTOSIZE);  //[ Create window]

    vector<string> present\_students;

    vector<string> all\_student\_names;

    set<string> absent\_students;  //[Use a set to track absent students]

    for (const auto& student : students) {

        all\_student\_names.push\_back(student.name);

    }

    while (true) {

        Mat frame, faceROI;

        cap >> frame;  //[capture frame from camera]

        faceROI = detectFace(frame, face\_cascade);

        if (!faceROI.empty()) {

            resize(faceROI, faceROI, Size(100, 100));  //[ Resize detected face to standard size]

            vector<string> detected\_students;

            for (const auto& student : students) {

                double diff = norm(student.face, faceROI, NORM\_L2);  // [Compare detected face with stored face]

                if (diff < 5000) {  //[Threshold for face matching]

                    if (find(detected\_students.begin(),detected\_students.end(),student.name)== detected\_students.end()){

                        detected\_students.push\_back(student.name);  // [Mark student as present]

                    }

                }

            }

            //|------------------------------------------------------|

            //|Write detected students to the attendance file|

            //|------------------------------------------------------|

            for (const auto& name : detected\_students) {

                if (find(present\_students.begin(), present\_students.end(), name) == present\_students.end()) {

                    present\_students.push\_back(name);

                    attendanceFile << name << " -> present.\n";

                    cout << name << " is present.\n";  // [Display on screen]

                }

            }

            //|----------------------------------------------------------------|

            //|Write absent students to the attendance file only once|

            //|----------------------------------------------------------------|

            for (const auto& name : all\_student\_names) {

                if (find(detected\_students.begin(), detected\_students.end(), name) == detected\_students.end() &&

                    find(present\_students.begin(), present\_students.end(), name) == present\_students.end() &&

                    absent\_students.find(name) == absent\_students.end()) {

                    absent\_students.insert(name);

                    attendanceFile << name << " -> absent.\n";

                }

            }

        }

        imshow("Mark Attendance", frame);  // [Show the frame with face detection]

        if (waitKey(30) >= 0) break;  // [Exit on key press]

    }

    attendanceFile.close();  // [Close the attendance file]

    destroyWindow("Mark Attendance");  //[ Destroy window after marking attendance]

}

//|--------------------------------------------------------|

//|Function to update a student's reference image|

//|--------------------------------------------------------|

void updateStudentFace(VideoCapture& cap, CascadeClassifier& face\_cascade, vector<Student>& students) {

    string folderPath = "C:\\Users\\manis\\Downloads\\reference\_faces\\";

    if (students.empty()) {

        cout << "|-----------------------------------------------------------------|\n";

        cout << "|No students registered. Please register students first.   |\n";

        cout << "|-----------------------------------------------------------------|\n";

        return;

    }

    string name;

    cout << "|----------------------------------------------------------------------------|\n";

    cout << "|Enter the name of the student whose image you want to update:|\n";

    cout << "|----------------------------------------------------------------------------|\n";

    cin >> name;

    auto it = find\_if(students.begin(), students.end(), [&name](const Student& s) { return s.name == name; });

    if (it == students.end()) {

        cout << "|------------------|\n";

        cout << "|Student not found.|\n";

        cout << "|------------------|\n";

        return;

    }

    Student& student = \*it;

    Mat frame, faceROI;

    namedWindow("Update Student Face", WINDOW\_AUTOSIZE);  // [Create window]

    while (true) {

        cap >> frame;  // [Capture frame from camera]

        faceROI = detectFace(frame, face\_cascade);

        if (!faceROI.empty()) {

            resize(faceROI, faceROI, Size(100, 100));  // [Resize to standard size]

            //|---------------------------------------------|

            //|Save the new face image to the folder|

            //|---------------------------------------------|

            string filePath = folderPath + name + ".jpg";

            imwrite(filePath, faceROI);

            student.face = faceROI;  // [Update the student face in memory]

            cout << "|updated face for| " << name << ".\n";

            break;

        }

        imshow("Update Student Face", frame);  //[ Show the frame with face detection]

        if (waitKey(30) >= 0) break;  // [Exit on key press]

    }

    destroyWindow("Update Student Face");  // [Destroy window after updating face]

}

//|----------------------------------------------------------|

//| Function to delete a student's reference image   |

//|----------------------------------------------------------|

void deleteReferenceFace(vector<Student>& students) {

    if (students.empty()) {

        cout << "|---------------------------------------------------------------|\n";

        cout << "|No students registered. Please register students first.|\n";

        cout << "|---------------------------------------------------------------|\n";

        return;

    }

    string name;

    cout << "|---------------------------------------------------------------------------|\n";

    cout << "|Enter the name of the student whose image you want to delete:|\n";

    cout << "|---------------------------------------------------------------------------|\n";

    cin >> name;

    // Find the student in the vector

    auto it = find\_if(students.begin(), students.end(), [&name](const Student& s) { return s.name == name; });

    if (it == students.end()) {

        cout << "|----------------------|\n";

        cout << "|Student not found.|\n";

        cout << "|----------------------|\n";

        return;

    }

    // Delete the face image from the filesystem

    string filePath = "C:\\Users\\manis\\Downloads\\reference\_faces\\" + name + ".jpg";

    if (remove(filePath.c\_str()) != 0) {

        cout << "|-----------------------------|\n";

        cout << "|Error deleting the image file|\n";

        cout << "|-----------------------------|\n";

    }

    else {

        cout << "|Deleted the face image for | " << name << ".\n";

    }

    // Remove the student from the vector

    students.erase(it);

}

//|--------------------------------------------------|

//| Function to view all registered students   |

//|--------------------------------------------------|

void viewRegisteredStudents(const vector<Student>& students) {

    if (students.empty()) {

        cout << "|---------------------------------------------|\n";

        cout << "|No students registered yet.                  |\n";

        cout << "|---------------------------------------------|\n";

        return;

    }

    cout << "|------------------ Registered Students ------------------|\n";

    for (const auto& student : students) {

        cout << "| " << student.name << "\n";  // Display each student's name

    }

    cout << "|-----------------------------------------------------------|\n";

}

int main() {

    CascadeClassifier face\_cascade;

    string path = "C:\\Users\\manis\\Downloads\\";  //[ Path where Haarcascade XML file is located]

    // Load Haarcascade XML file

    if (!face\_cascade.load(path + "haarcascade\_frontalface\_default.xml")) {

        cout << "|-------------------------------|\n";

        cout << "|Error loading Haarcascade file |\n";

        cout << "|-------------------------------|\n";

        return -1;

    }

    VideoCapture cap(0);  // [Open the default camera]

    if (!cap.isOpened()) {

        cout << "|----------------------|\n";

        cout << "|Error opening webcam  |\n";

        cout << "|----------------------|\n";

        return -1;

    }

    vector<Student> students;  // [Vector to store student names and faces]

    // [Load reference faces from the folder]

    loadReferenceFaces(students);

    // Main menu loop

    while (true) {  // [Loop to continuously show the menu]

        int choice;

        cout << "|----------------------------------------|\n";

        cout << "|-----Welcome to face attendance app-----|\n";

        cout << "|----------------------------------------|\n";

        cout << "|Select an option:                       |\n";

        cout << "|1. Register student faces               |\n";

        cout << "|2. Mark attendance                      |\n";

        cout << "|3. Update student face                  |\n";

        cout << "|4. Delete student face                  |\n";

        cout << "|5. View registered students             |\n";  // Option to view students

        cout << "|6. Exit                                 |\n";  // Exit option

        cout << "|Enter your choice (1, 2, 3, 4, 5, or 6):|\n";

        cout << "|----------------------------------------|\n";

        cin >> choice;

        switch (choice) {

        case 1:

            captureReferenceFaces(cap, face\_cascade, students);  // Capture student faces

            break;

        case 2:

            if (students.empty()) {

                cout << "|----------------------------------------------------------|\n";

                cout << "|No student faces registered. Please register faces first. |\n";

                cout << "|----------------------------------------------------------|\n";

            }

            else {

                markAttendance(cap, face\_cascade, students);  // Mark attendance by comparing faces

            }

            break;

        case 3:

            updateStudentFace(cap, face\_cascade, students);  // Update student face

            break;

        case 4:

            deleteReferenceFace(students);  // Call to delete student face

            break;

        case 5:

            viewRegisteredStudents(students);  // Call to view registered students

            break;

        case 6:

            cout << "|----------------------|\n";

            cout << "|Exiting the program...|\n";

            cout << "|----------------------|\n";

            return 0;  // Exit the program successfully

        default:

            cout << "|---------------------------------------------|\n";

            cout << "|Invalid choice. Please select a valid option>|\n";

            cout << "|---------------------------------------------|\n";

            break;

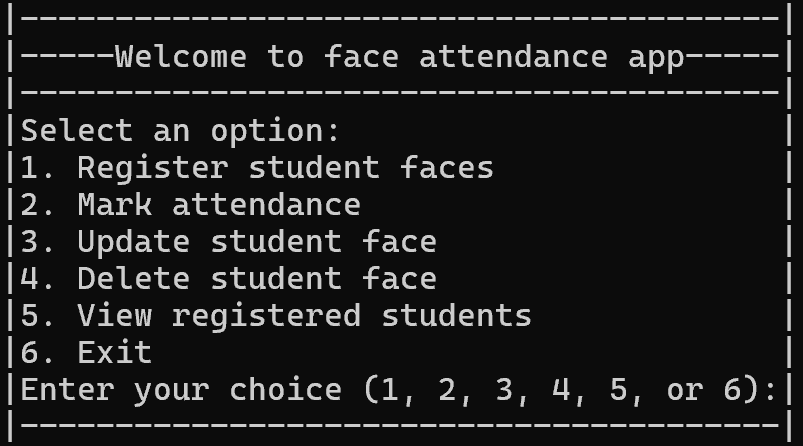
        }

    }

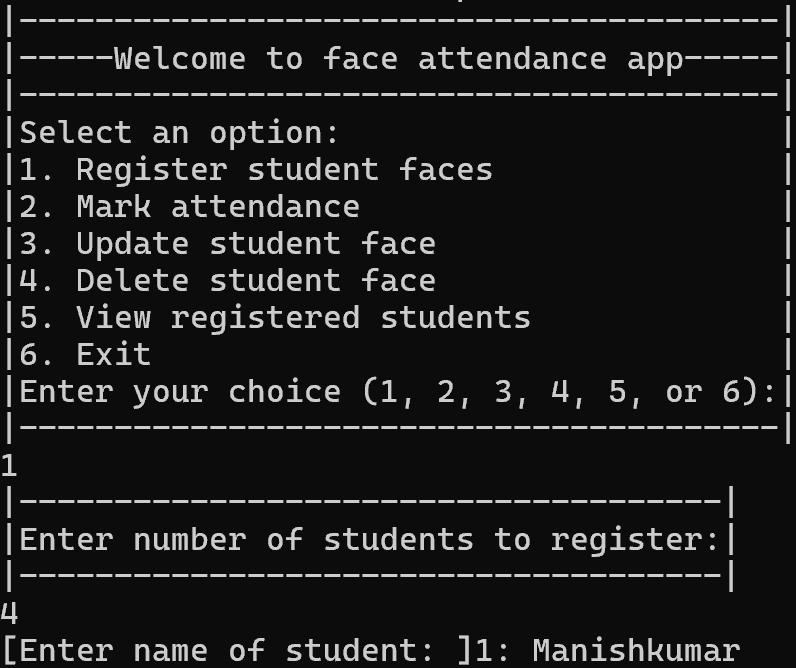
    return 0; // Indicate successful termination of the program.

}

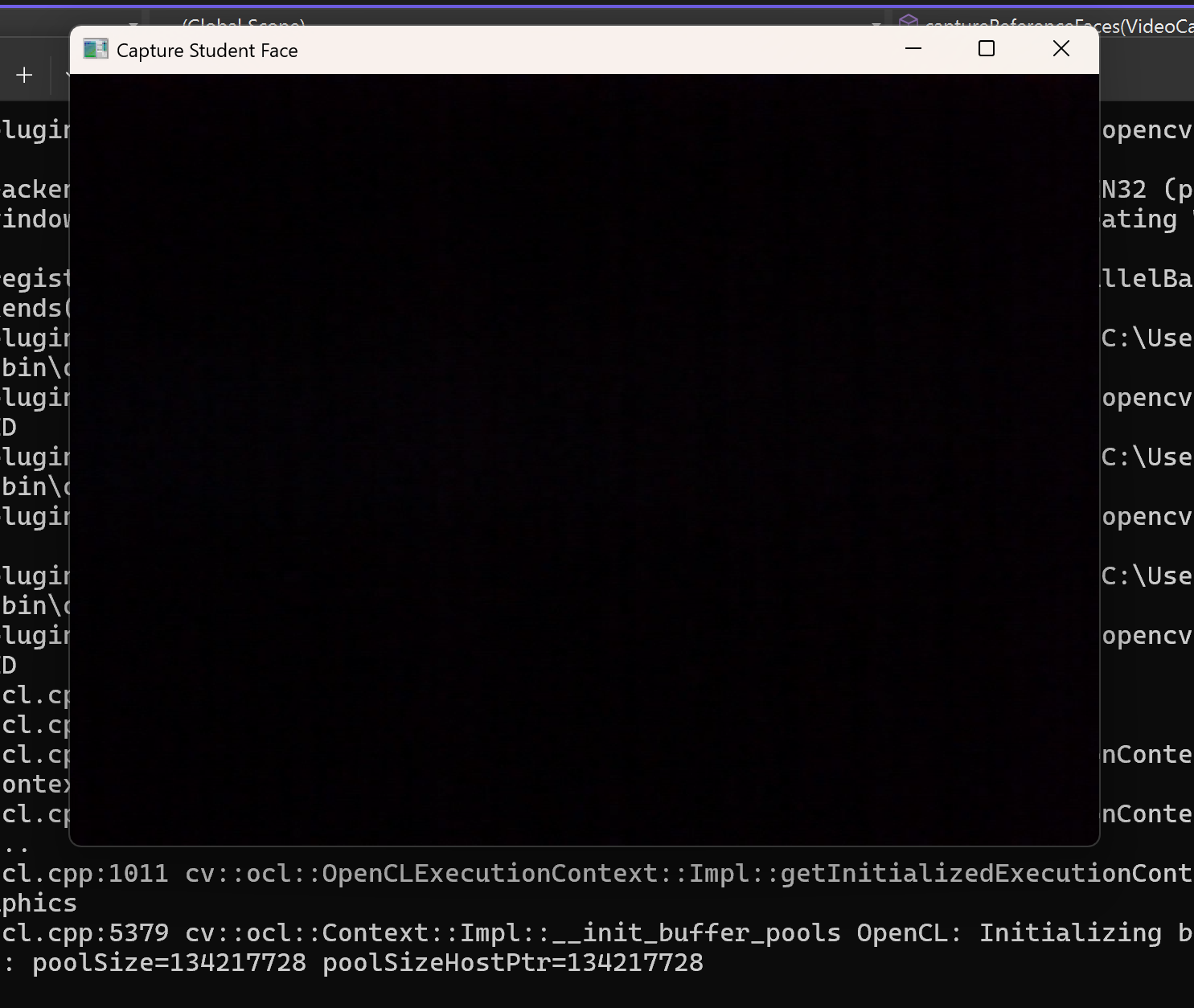
**OUTPUT**



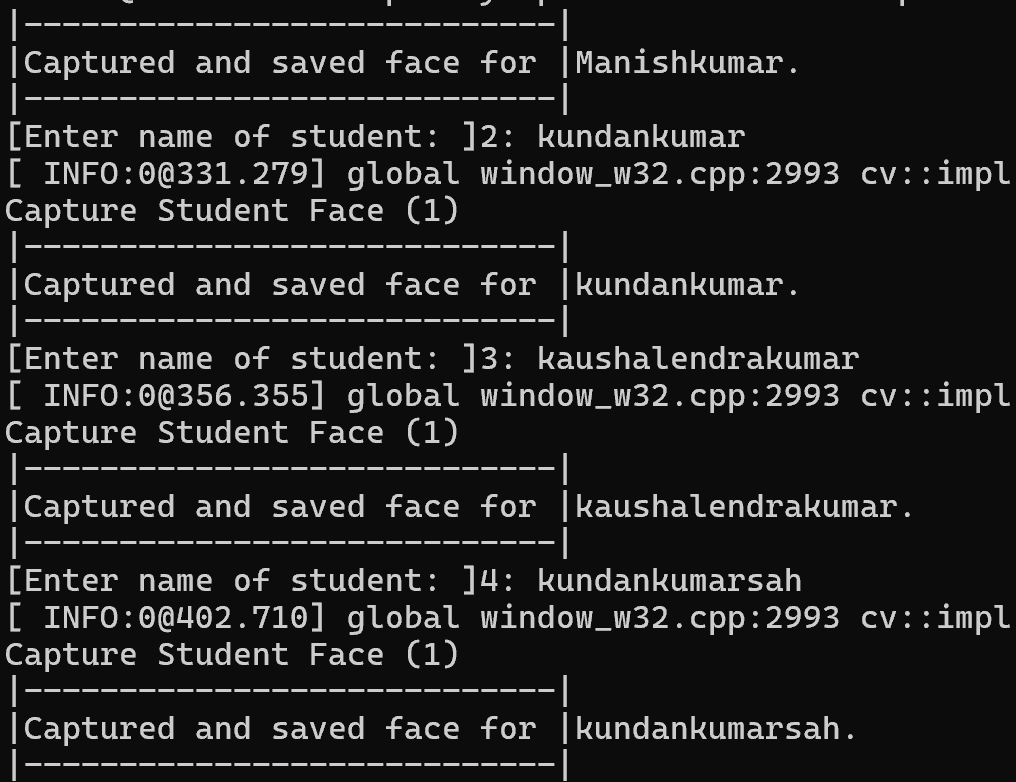
**Figure1:-Main interface of application**



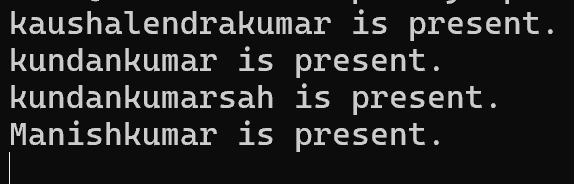
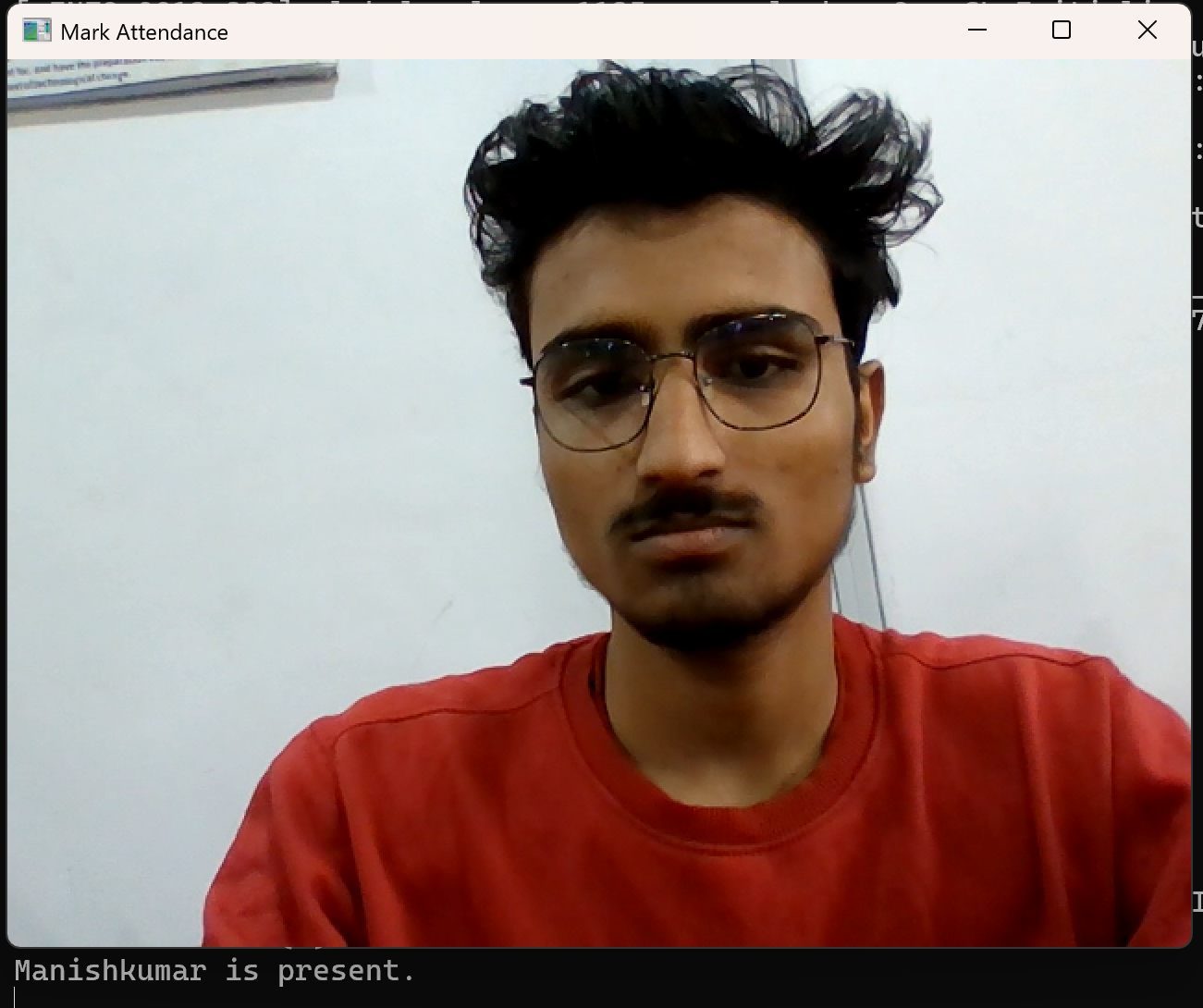
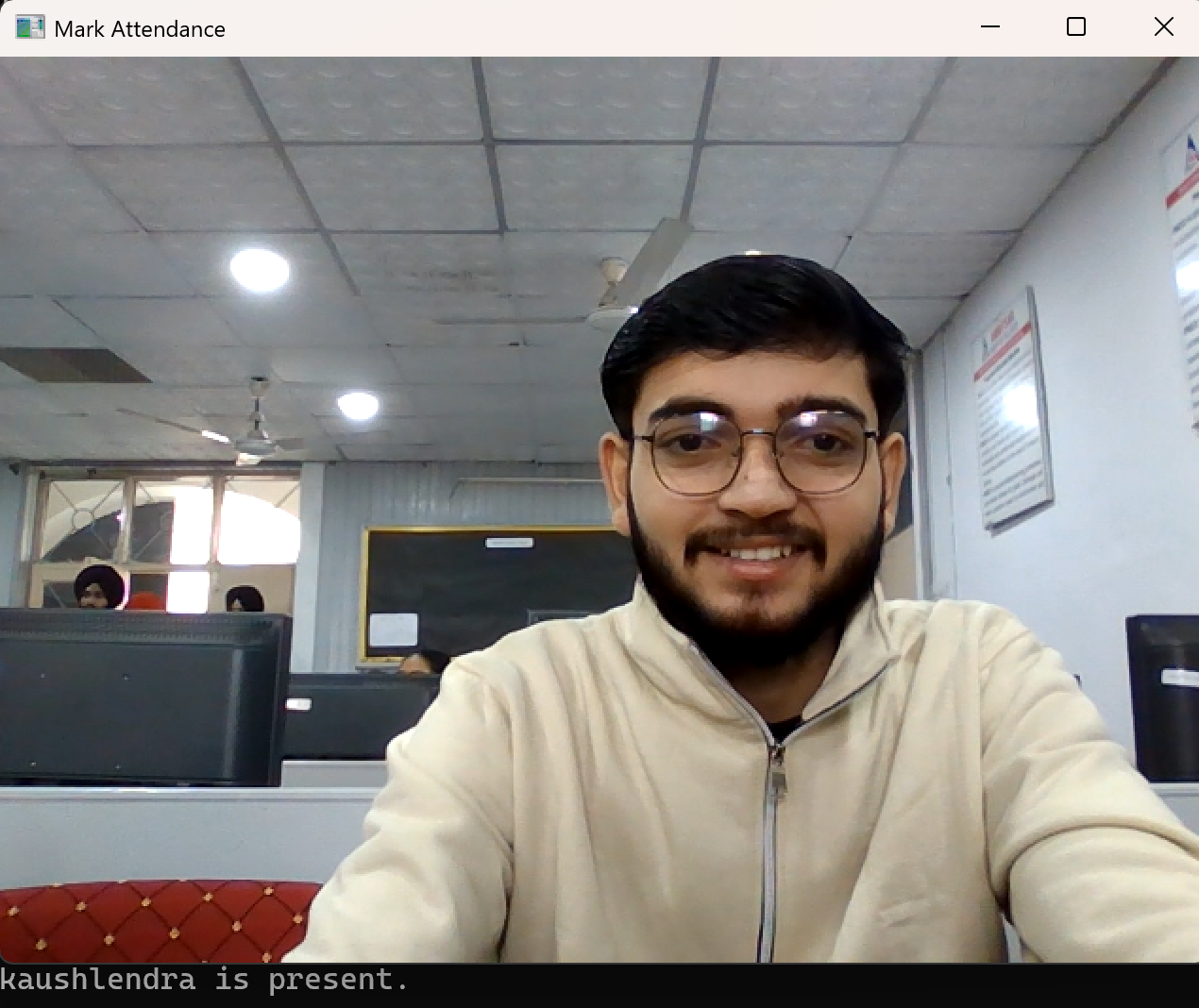
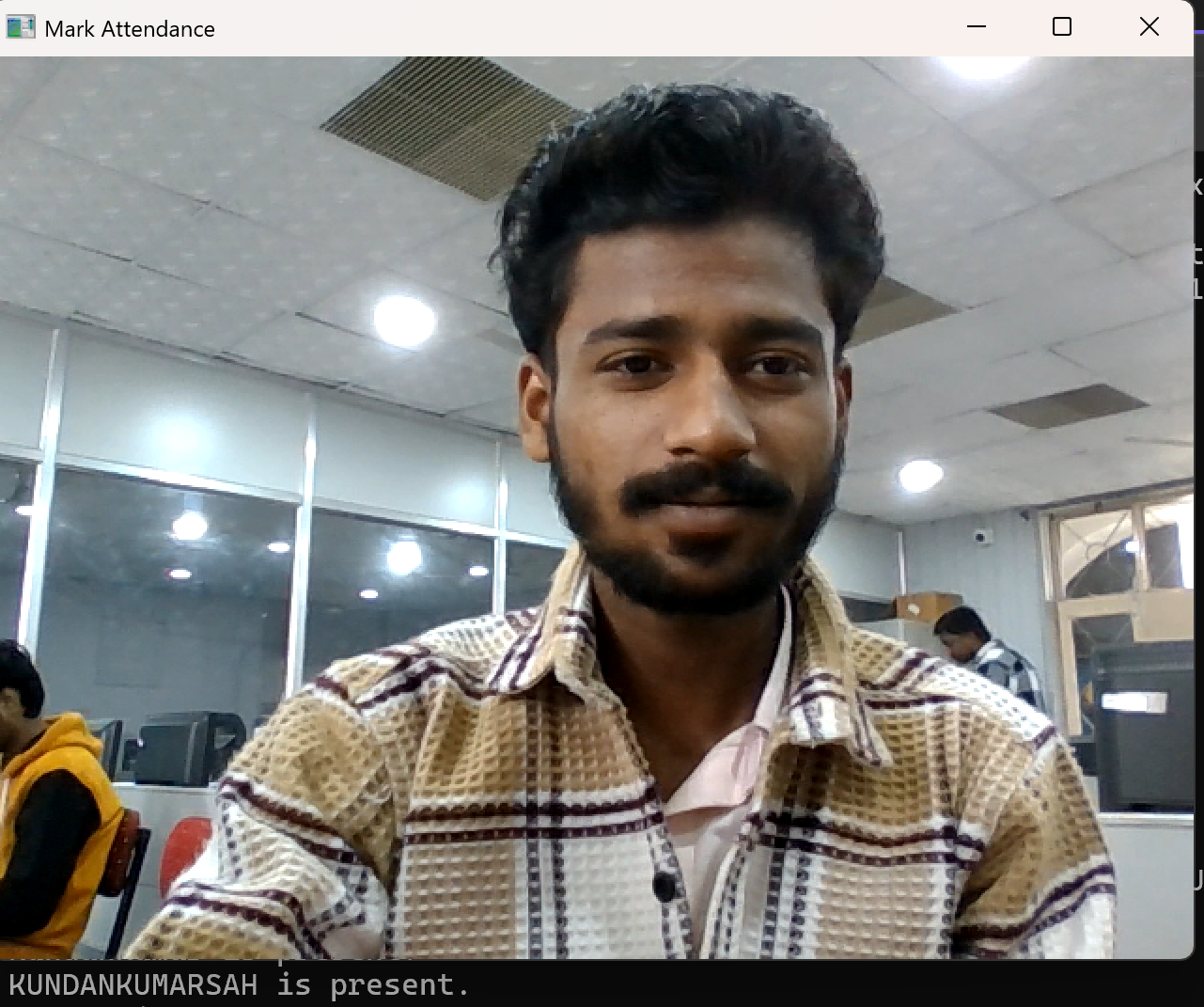
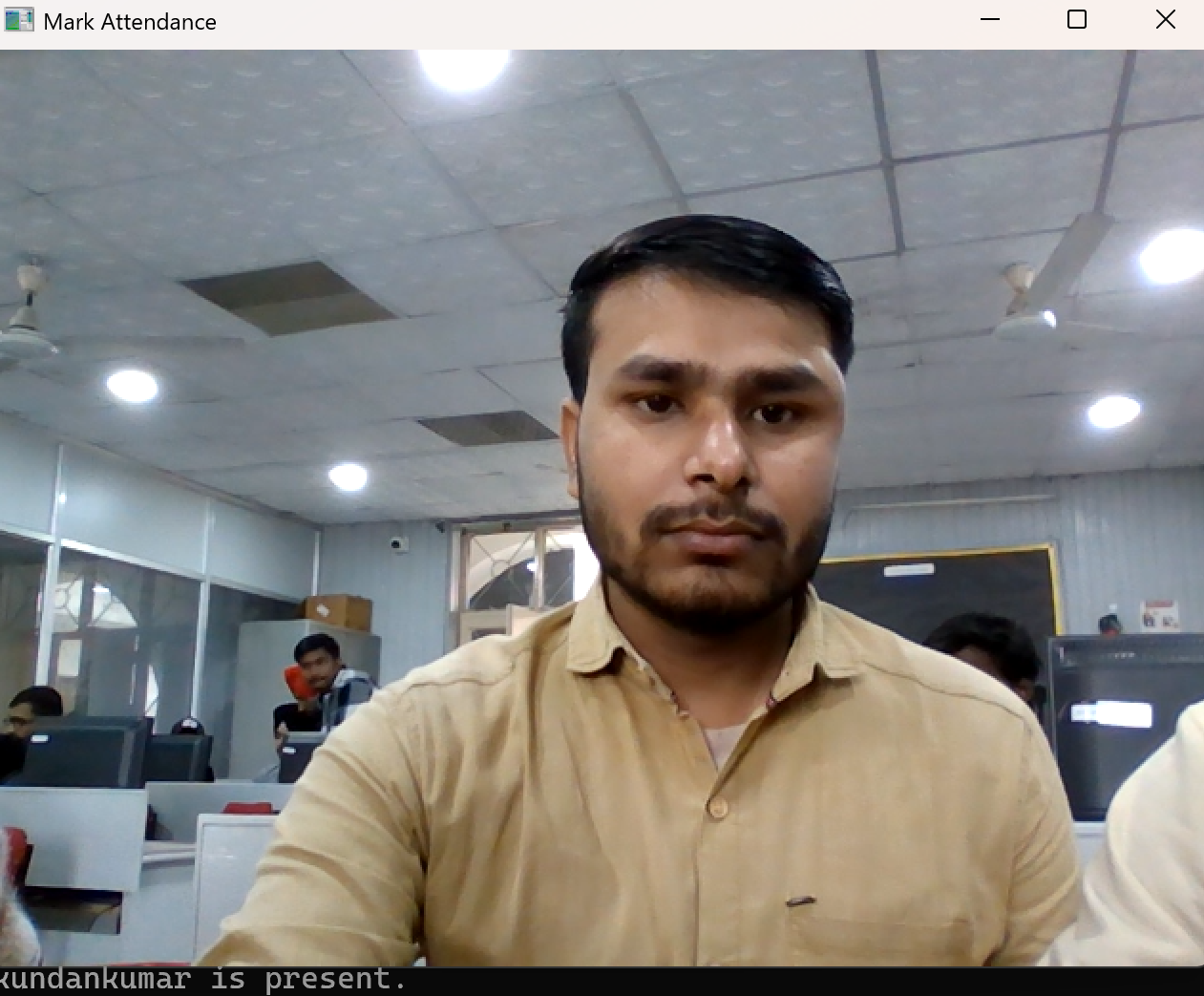
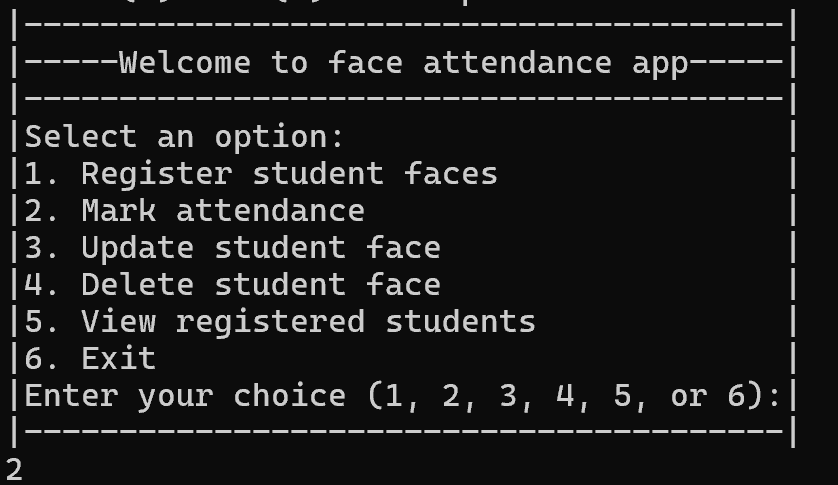
**Figure2:-Register student face**



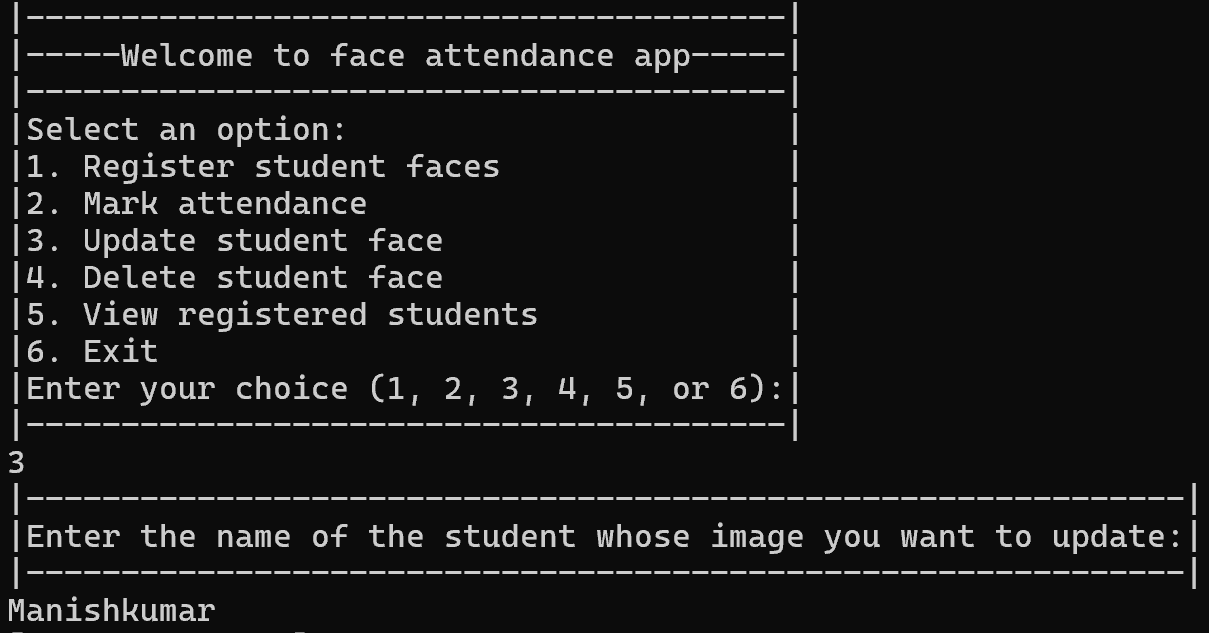
**Figure3:-Camera open for capture face**



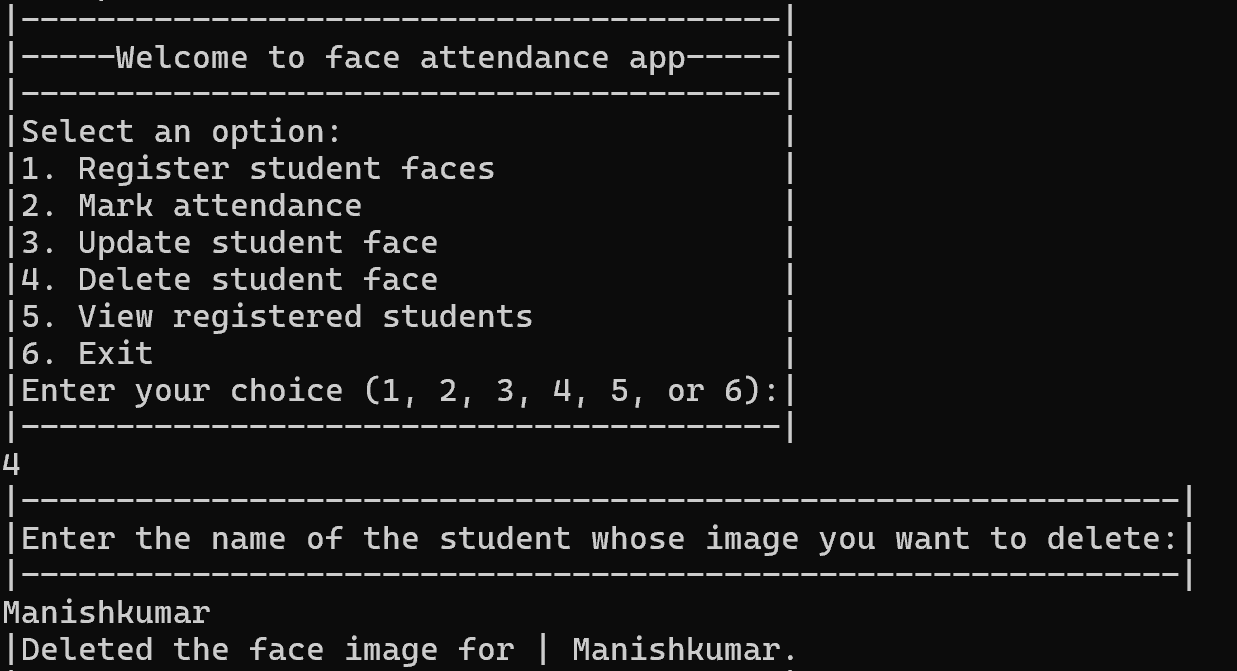
**Figure4:-capture the face**



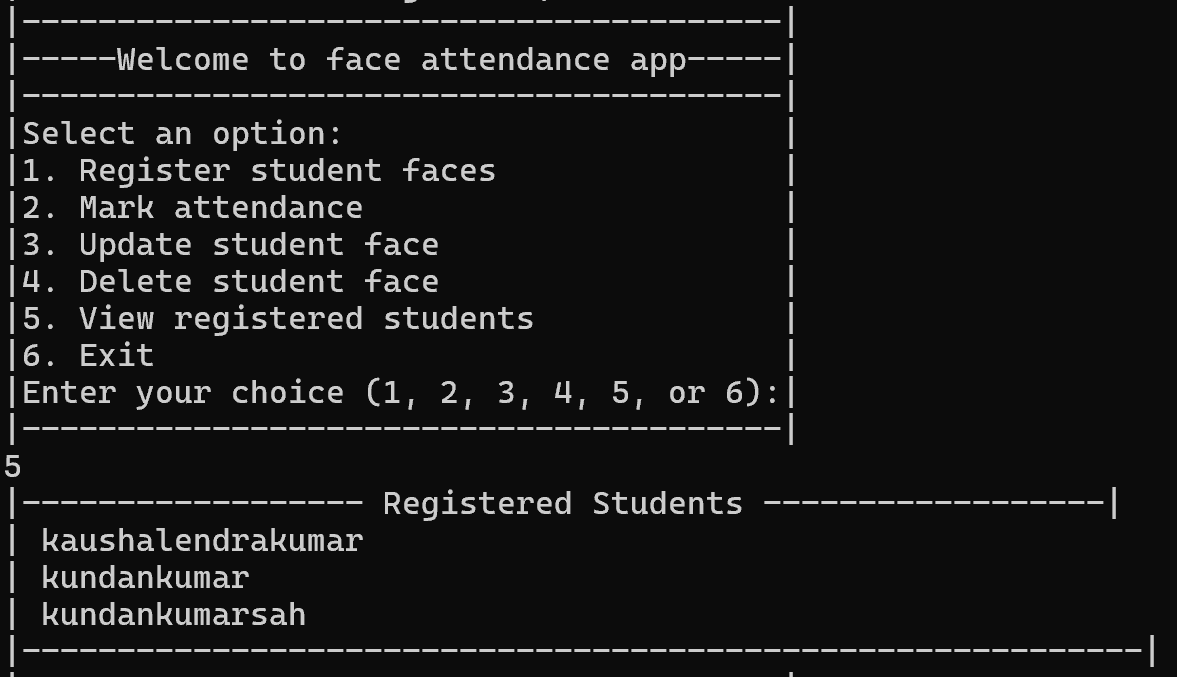
**Figure5:-Mark attendance**



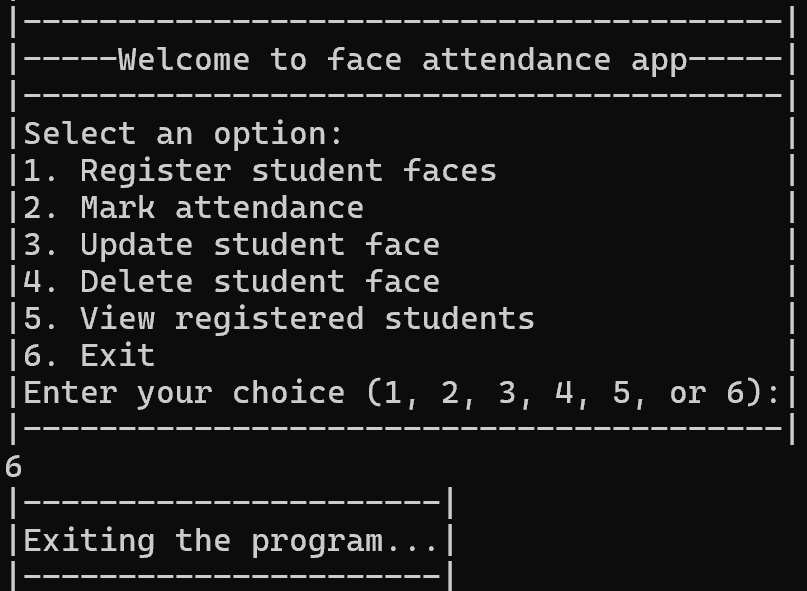
**figure6:-Update student face**



**Figure7:-Delete student face**



**Figure8:-View registered student**



**Figure9:-Exiting pannel**

**References**

1. Books :

* Learn Programming in C++ (Anshuman Sharma)
* “Object Oriented Programming with C++” (E Balagurusamy)
* "The C++ Programming Language" by Bjarne Stroustrup

1. OpenCV Documentation. (n.d.). Retrieved from https://docs.opencv.org/
2. Haar Cascades for Face Detection. (n.d.). Retrieved from <https://github.com/opencv/opencv/tree/master/data/haarcascades>
3. Digital Image Processing Using OpenCV. (n.d.). Various authors. Retrieved from relevant sources or textbooks.