```
/*
Programmers:
Ranica Chawla and Krishi Dalal

Purpose:
This product provides a user-friendly experience for teachers to conduct class attendance in
short time period.
This saves time for teachers in the classroom and allows them to focus on class content and
learning!
To begin, simply run the program and start adding students to your class list right away!

NOTE: Only admin/staff may access the remove/add student features and must enter in the key
provided by the manager
*/

#include <iostream>
// import the necessary libraries for creating/adjusting files
#include <fstream>
#include <filesystem>
// import the necessary library files for opencv
#include <opencv2/imgcodecs.hpp>
#include <opencv2/highgui.hpp>
#include <opencv2/imgproc.hpp>
#include <opencv2/videoio.hpp>
#include <opencv2/objdetect.hpp>

// include a library to get the current date
#include <chrono>

using namespace cv;
using namespace std;

// class to hold all the colours for text
class Colours {
   public:
   // create new functions for every different colour in order to be more efficient when trying to
change the colour of the terminal
      void blue(){
         // use the corrosponding ANSI escape sequences for each colour
        cout<<"\033[0;34m";
      }
      void magenta(){
         cout<<"\033[0;35m";
      }
```

```
        void cyan(){
            cout<<"\033[0;36m";
        }
        void red(){
            cout<<"\033[0;31m";
        }
        void white(){
            cout<<"\033[0;37m";
        }
        void green(){
            cout<<"\033[0;32m";
        }
};

// class to hold all the functions of the attendance system
class attendance {
    public:
        //structure to store student data in variables
        struct {
            // use a vector to hold all the student numbers and student names that exist in the class
            vector<int> studNum;
            vector<string> studName;
        } studInfo;

        // structure to store the students numbers and names present
        struct {
            // use a vector to hold all the student numbers and student names that are marked
present
            vector<int> studNum;
            vector<string> studName;
        } studPresent;

        // functions to add and remove students and mark their attendance
        void addStudents();
        void removeStudents();
        void markAttendance();
        int action();

        // functions to create the excel sheets
        void classList();
        void attendanceList();

        //function for user password entry
        void teacherPassword();
```

```
};

// function to create/edit the list of the class
void attendance::classList() {
   // create an instance of the ofstream class to write data on files
   ofstream classList;
   // create a new excel file and declare where its path/location is
   classList.open("../resources/classList.csv");

   // add the headers for the columns for the csv file
   classList << "Student Name" << "," << "Student Number" << endl;

   // loop through all the information stored in the student information after the addStudents
function is run and append to csv file
   for (int i = 0; i < studInfo.studName.size(); i++) {
      // each cell is seperated by a comma, and each row by endl
      classList << studInfo.studName[i] << "," << studInfo.studNum[i] << endl;
   }
}

// function to create a new attendance list for the current date/time
void::attendance::attendanceList() {
   // create a variable to hold the current date and time to save the excel sheet name as and
using the chrono library
   const auto date = chrono::system_clock::now();
   // convert the current time point into a time object because c++ cannot directly convert to
string
   std::time_t time = std::chrono::system_clock::to_time_t(date);
   // convert from time object to string
   std::string dateString = std::ctime(&time);

   // create an instance of the ofstream class to write data on files
   ofstream attendanceList;

   // declare the file path for the attendance lists to the saved in
   string file_path = "../resources/attendance-lists/" + dateString + ".csv";

   // create a new class list
   attendanceList.open(file_path);

   // add the headers for the columns for the csv file
   attendanceList << "Student Name" << "," << "Student Number" << "," << "Attendance" <<
endl;
```

```
    // loop through the entire vector of students and add each student to the datasheet and if they
were marked present or not
    for (int i = 0; i < studInfo.studName.size(); i++) {
        // declare a variable to check if the student was marked present or not after looping through
the students present vector
        bool present = false;
        // loop through the entire vector of the students marked present
        for (int j = 0; j < studPresent.studName.size(); j++) {
            // check if the current student name exists inside of the students present list and if they
do, mark them present
            if (studInfo.studName[i] == studPresent.studName[j]) {
                // each cell is seperated by a comma, and each row by endl
                attendanceList << studInfo.studName[i] << "," << studInfo.studNum[i] << "," <<
"Present" << endl;
                // set present to true to let the program know they were found
                present = true;
                // break out of the loop
                break;
            }
        }
        // if the student wasn't found in the array, then they will be marked absent
        if (!present) {
            // append absent to the csv file for the corrosponding student
            attendanceList << studInfo.studName[i] << "," << studInfo.studNum[i] << "," << "Absent"
<< endl;
        }
    }
}

// create an instance of the class defined above of all the colours and is done outside the
functions so they all have access to this class
Colours colours;

// function to check if a teacher and/or admin is accessing the remove and add students feature
and not an imposter
void attendance::teacherPassword() {
    //variables storing a value for password that must be typed to proceed with adding or
removing students from the class list
    string password = "3141592653589";
    string passwordEntered;

   //prompts user to enter the password to add or remove students
    cout << "Please enter the password to continue with the action: ";
    cin >> passwordEntered;
    //while loop that continues to promt user to enter a password until a correct match is found
    while (password != passwordEntered) {
        colours.red();
        //if the stored password does not match the one entered, it will prompt user to re-enter the
correct password
        cout << "Wrong! Please re-enter the correct password: ";
        colours.white();
        cin >> passwordEntered;
    }
    // enters a new line before next text
    cout << "\n";
}

// function to add more students to the class list
void attendance::addStudents() {
    // call the teacher password so students cannot access the class list
    teacherPassword();

    //variable to store number of students being added
    int n = 0;
    colours.magenta();
    //prompts user to enter the number of students they want to add
    cout << "Enter the number of students you would like to add: ";
    colours.white();
    cin >> n;

    // checks if the input is valid and if it's not it will keep prompting user to re-enter a valid input
    while (n < 0 || cin.fail()) {
        cin.clear();
        cin.ignore(10000, '\n');

        // prompts user to re enter a valid input
        colours.red();
        cout << "Please enter a valid number: ";
        colours.white();
        cin >> n;
    }

    //for loop that allows student details to be entered and stored in their arrays
    for (int i = 0; i < n; i++) {
        // declares variables to hold the current number and name of student
        int num;
        string name;
        // get user input about the name and number of student
        colours.green();
        cout << "Student Name: ";
        colours.white();
        cin >> name;

        colours.cyan();
        cout << "Student Number: ";
        colours.white();
        cin >> num;

        // checks if the input is valid and if it's not, then it will run until it is
        while (num < 0 || cin.fail()) {
            // clears invalid input by reseting the error state
            cin.clear();
            cin.ignore(10000, '\n');

            // prompts user to re-enter a valid number
            colours.red();
            cout << "Please enter a valid number: ";
            colours.white();
            cin >> num;
        }

        // adds student name and number entered to the array holding the details to the last index
of the vector
        studInfo.studName.push_back(name);
        studInfo.studNum.push_back(num);
    }

    // call the class list function to create an excel sheet holding all the students names and
numbers
    classList();
}

// function to remove students from the existing class list
void attendance::removeStudents() {
    //calls function to make user enter password to proceed with removing students from class list
    teacherPassword();
    // variable to store number of students being added
    int n = 0;
    colours.magenta();
    //asks user to enter the number of student's they'd like to remove
    cout << "Enter the number of students you would like to remove: ";
    colours.white();
    cin >> n;

    // checks if the input is valid and if it's not, then it will run until it is
    while (n < 0 || cin.fail() || n > studInfo.studNum.size()) {
        // resets the error state
        cin.clear();
        cin.ignore(10000, '\n');
        colours.red();

        //checks for invalid input - if the number of student exceeds number of students in class list
the user will be asked to re-enter a valid number
        if (n > studInfo.studNum.size()) {
            cout << "The number of students to remove is too high! Please enter a valid number: ";
        } else {
            // user will be prompted to enter a valid number of students to remove
            cout << "Please enter a valid number: ";
        }
        colours.white();
        //input field for user to enter the number of students they want to remove
        cin >> n;
    }


    // for loop to enter the student numbers you want to remove
    for (int i = 0; i < n; i++) {
        // declares a variable to store current student number entered
        int num;
        colours.cyan();
        //prompts user to enter the student's student number
        cout << "Student Number: ";
        colours.white();
        cin >> num;

        // create a variable count in order to check if user input is valid by comparing it against the
whole vector
        bool found = false;

        // keep repeating this section of code until the break statement is declared (loop through
and remove the inputted student numbers)
        while (true) {
            // checks if the input is valid and if it's not, then it will run until it is
            while (num < 0 || cin.fail()) {
                cin.clear();
                cin.ignore(10000, '\n');

                // prompts user to re enter a valid input
                colours.red();
                cout << "Please enter a valid number: ";
                colours.white();
                cin >> num;
            }

            // for loop to determine where the student number is in the array and removes it
            for (int j = 0; j < studInfo.studNum.size(); j++) {
                // checks if the student number entered matches one if the studNum and erases it
                if (num == studInfo.studNum[j]) {
                    // erases student number and name from array at current index which the number
was found
                    studInfo.studNum.erase(studInfo.studNum.begin() + j);
                    studInfo.studName.erase(studInfo.studName.begin() + j);
                    // set the found variable to true to tell the program that the functin has been
sucessful and break out of the loop
                    found = true;
                    break;
                }
            }
            // if the found variable is still false, it will prompt the user to enter a valid input
            if (!found) {
                colours.red();
                //tells user that student number isn't found and asks to re-enter a valid number
                cout << "Student number not found! Please enter a valid number: ";
                colours.white();
                cin >> num;
            } else {
                // if a match was found, it will break out of the loop
                break;
            }
        }
    }
    // displays the student number array after a number has been removed
    // for (int i = 0; i < studInfo.studNum.size(); i++) {
    //     cout << studInfo.studNum[i] << endl;
    // }
    colours.white();
    cout << "\nThe student(s) has been removed!\n\n";
    // call the call list function again to update it with the removed student(s)
    classList();
}

// function to detect faces present in the current frame/window
bool facialDetection(Mat frame) {
    // clone the original frame
    Mat org = frame.clone();

    //convert the frame from bgr to grayscale
    Mat grayscale;
    cvtColor(org, grayscale, COLOR_BGR2GRAY);

    //array that stores information about the faces detected (e.g. coordinates)
    vector<Rect> faces;

    // create an instance of the object detection class
    CascadeClassifier faceCascade;
    faceCascade.load("../resources/haarcascade_frontalface_default.xml");

    // uses opencv's built in facial recognization algorithm to detect faces within a given
image/frame
    // the first parameter is the image that you are detecting objects from, the second is a vector
to store the data obtained from the model
    // the third and fourth are used to fine-tune the model (1.1 is the scale factor and 10 is the min
neighbors)
                                        // scale factor: how much an image will be resized to
compare to training data
                                        // min neighbors: how many detected overlaps is required
to be considered a match
    faceCascade.detectMultiScale(grayscale, faces, 1.1, 10);

    // detect all present faces in the frame through the camera
    for (int i = 0; i < faces.size(); i++) {
        // create a rectangle around the detected face(s)
        rectangle(frame, faces[i].tl(), faces[i].br(), Scalar(255, 0, 0), 5);
    }

    // checks if the array of faces is empty (meaning there is no faces detected in the frame) and
return false if it is
    return !faces.empty();
}

//function to mark students present in a class
```

```
void attendance::markAttendance() {
    // declares variables to hold the current number and name of student
    int num;
    string name;

    // access the video camera
    VideoCapture cap(0);
    // create a frame (matrix that holds various images) that takes the rows, columns, and type of
elements --> CV_8UC3 means 3 channel rgb
    Mat frame;

    while (true) {
        // provide information/instructions to the user about how to use the program or exit it
        cout << "Please enter the students information OR enter 0 to exit the program\n";

        //student enters name and their student number to mark themselves present
        colours.green();
        cout << "Student Name: ";
        colours.white();
        cin >> name;

        // break out of the program when name is 0
        if (name == "0") {
            // break out of the while loop
            break;
        }

        // promp the user to enter the sttudent number and save the current student to the variable
num
        colours.cyan();
        cout << "Student Number: ";
        colours.white();
        cin >> num;

        //while the input is not a number (but rather a string), the user will be told invalid input and
prompted to re-enter a valid number
        while (cin.fail()) {
            // clears invalid input by resetting the error state of cin
            std::cin.clear();
            std::cin.ignore(10000, '\n');

            // prompts user to re enter a valid input
            colours.red();
            cout << "Please enter a number: ";
            colours.white();
            cin >> name;
        }

        // check if the student details entered of a student were already marked present or not
        while (true) {
            // create a bool variable called exists to check if the current student is already marked
present or not
            bool exists = false;

            // loop through the student present vector that holds all the students alredy marked
present
            for (int i = 0; i < studPresent.studNum.size(); i++) {
                // check if the current number is the same as one of the others in the students present
vector
                if (num == studPresent.studNum[i]) {
                    // if it is, then the following message will be displayed
                    colours.red();
                    cout << "This student is already marked present! Enter valid student details: \n";
                    colours.white();

                    // users must enter in the student name and student number again
                    colours.green();
                    cout << "Student Name: ";
                    colours.white();
                    cin >> name;

                    colours.cyan();
                    cout << "Student Number: ";
                    colours.white();
                    cin >> num;

                    // set the exists variable to true to let the program know it still must run again
                    exists = true;
                    // break out of the for loop
                    break;
                }
            }

            // if the student details entered were repeated, the while loop will repeat, otherwise, it will
break out of the while loop
            if (!exists) {
                break;
            }
        }

        colours.white();

        // create a variable to check if the student number exists in the attendance list
        bool found = false;

        // keep rerunning this until the student number and name are both valid inputs
        while (true) {
            // define a variable to hold the index at which the student number is found in the vector
            int index = 0;

            // loop through all the numbers in the array
            for (int i = 0; i < studInfo.studNum.size(); i++) {
                // checks if the student number entered matches one of the studNum
                if (num == studInfo.studNum[i]) {
                    // change found to true to let the program know the number is valid/exists
                    found = true;
                    // store the current index at which the number was found
                    index = i;
                    // break out of the for loop
                    break;
                }
            }

            // if the found variable is still false, it will prompt the user to enter a valid input
            if (!found) {
                colours.red();
                cout << "Student number not found! Please enter a valid number: ";
                colours.white();
                cin >> num;
            // checks if the current entered name matchs the name that corrosponds with the student
    number inputted
            } else if (name != studInfo.studName[index]) {
                // prompt the user to re enter both the name and number if invalid
                colours.red();
                cout << "Student name and number does not match! Please enter valid information.
    \n";
                cout << "Student Name: ";
                colours.white();
                cin >> name;

                colours.cyan();
                cout << "Student Number: ";
                colours.white();
                cin >> num;
            } else {
                // if the number exists and the name and number matched, it will break out of the loop
                break;
            }
        }

        // create a window that will display all the information
        namedWindow("Main");

        // create a variable that holds true if there is a person detected in the camera and false if
    not
        bool present = false;

        // continuously display the main window until exited
        while (true) {
            // let the capture function open a frame
            cap >> frame;

            // run the facial detection function to see if there is anyone in front of the camera and
    save the result in the variable present
            present = facialDetection(frame);

            // display the frame
            imshow("Main", frame);

            // wait until the esc key is pressed
            if (waitKey(1) == 27) {
                // break out of the loop
                break;
            }
        }

        // if someone's face was detected in the frame, they will be marked present
        if (present) {
            // add the name and number of the current student thats present to the vectors
            studPresent.studName.push_back(name);
            studPresent.studNum.push_back(num);

            cout << "\nThe student has been marked present!\n\n";
        } else {
            // let the user know they weren't marked present or there was no one in the frame
            colours.red();
```

```
            cout << "No face found! Please re enter information and try again: \n";
        }
    }

    // call the function below to make a csv file to store the marked students and save to the
resources folder
    attendanceList();

    // close the current window
    destroyWindow("Main");
    cap.release();
}

// function to display a main menu/interface to the user to select various actions from
int attendance::action(){
    //variable that stores users choices of actions
    int choice;

    //prompts user to choose an action they would like execute
    colours.white();
    cout << "\nPlease choose one of the following actions: \n";
    colours.magenta();

    // displays the possible actions students can enter
    cout << "1. Mark Attendance\n2. Add Students\n3. Remove Students\n4. Exit\n";
    colours.white();

    // allows user to input their choice of acction
    cout << "Action: ";
    cin >> choice;
    cout << "\n";

    // while the input is not a number (but rather a string), the user will be told invalid input and
prompted to re-enter a valid number
    while (choice <= 0 || choice > 4 || cin.fail()) {
        // clears invalid input by resetting the error state of cin
        cin.clear();
        cin.ignore(10000, '\n');

        colours.red();
        // prompts user to re enter a valid input
        cout << "Please enter a valid number: ";
        colours.white();
        cin >> choice;
    }

    // return the users choice as an int to the displayFrame function where it will run the
corrosponding function
    return choice;
}

// function that is responsible to run the function that the user chose and also exiting the
program
bool displayFrame(int action, attendance &system) {
    // quit the program if the action is 4 (which means they entered they want to exit)
    if (action != 4) {
        // check what the action is and then run the corrosponding function
        switch (action) {
            case 1:
                system.markAttendance();
                break;
            case 2:
                // run the add students function if user enters they want to
                system.addStudents();
                break;
            case 3:
                // run the remove students function if the user enters they want to
                system.removeStudents();
                break;
        }
        // return false to let the software know that the user still wants to continue doing more
actions
        return false;
    } else {
        colours.blue();
        // thanks users for using our product once they are complete with their actions
        cout << "Thank you for using TxFy-Trackify! We hope you enjoyed your experience.\n";
        // return true to let the software know that the user wants to exit the program
        return true;
    }
}

// main function
int main() {
    // create an instance of the class attendance declared at the start of the code
    attendance system;

    // display the welcome message to the user at the start of the program
    colours.blue();
    cout << "Welcome to TxFy-Trackify! An app that easily allows you to do your class attendace
:)\n";
    colours.white();

    while (true) {
        // call the main menu of the program where the user can choose what action they would
like to do and save it to the integer action
        int action = system.action();

        // display the main frame until the function returns true which means they want to quit the
program
        if (displayFrame(action, system)) {
            // end the program
            break;
        }
    }

    return 0;
}
```