```
/*

Name:                   Jamilu Salisu
Registration Number:    CST/19/SWE/00409
Course:                 Software Construction - SWE2210

System Implemented:     Visitors Management System
Date of submission:     Saturday, July 17, 2021

Features of the system implemented are as follows:
- Pre-booking allows visitors and guests to pre-register prior to arriving.
- Check-In Workflows for different visitor types such as; interview, etc.
- Sign digital forms to improve data security and reduce manual labour process
- Notify hosts instantly and communication to awaiting visitor of their status
- Peace of mind to organization and everyone who enters their workplace
- Making hosting of visitors seamless

*/

#include <iostream>
#include <ctime>
#include <string>

using namespace std;

class VisitorsManagementSystem
{
private:
    /*declare and initialize the visitor's
    personal information variables*/
    string visitorFullName = "";
    string visitorPhoneNumber = "";
    string visitorContactAddress = "";

    /*declare and initialize the visiting
    information variables*/
    string purposeOfVisiting = "";
    string dateOfVisiting = "";
    string timeOfVisiting = "";

    /* Initialize purpose visit and
    prompt user to choose from the option
    available */
    const string visitingPurpose[6] = {
        "Complain",
        "Interview",
        "Meeting",
        "Personal",
        "Repair",
        "Other"};

    /* Initialize time slot and prompt user
    to choose from the option available*/
    const string visitingTime[9] = {
        "10:00am - 10:30am",
        "10:30am - 11:00am",
        "11:00am - 11:30am",
        "11:30am - 12:00pm",
        "12:00pm - 12:30pm",
        "1:30pm - 2:00pm",
        "2:00pm - 2:30pm",
        "2:30pm - 3:00pm",
        "3:00pm - 3:30pm",
    };

    /* Initialize visiting progress, set
    index[1] 'Pending' by default for every request,
    and set index[0] 'Approved' or index[2] 'Rejected'
```

```
when the host communicate the status*/
const string visitProgress[3] = {
    "Approved",
    "Pending",
    "Rejected"};

//setting index[1] 'Pending' by default as describe above
string visitRequestStatus = visitProgress[1];

//store user reply temporarily
int userResponse = 0;
int trialAttemp = 3;

/* Automatically get today's date and set
it as the visiting date for each request entry */
void setDateOfVisiting()
{
    //get and print today's date
    time_t curr_time;
    tm *curr_tm;
    char date_string[100];

    time(&curr_time);
    curr_tm = localtime(&curr_time);
    //format: day month, -year
    strftime(date_string, 50, "%d %B, %Y", curr_tm);

    dateOfVisiting = date_string;
    cout << dateOfVisiting;
}

/* This routine authenticate the admin that manage
visiting request. Its invoke by the login routine
after accepting authentication code from the user */
void setVisitStatus(string passwordEntered)
{
    if (passwordEntered == "admin")
    {
        cout << "\nAccess granted! Please confirm the visiting status:\n";
        cout << "1. Let the visitor in. \n2. Deny visitor access." << endl;

        cout << "Reply: ";
        cin >> userResponse;

        if (userResponse == 1)
        {
            //set the option 'Approved' as the status of the visit request
            visitRequestStatus = visitProgress[0];

            //send approval notification to the visitor
            sendApprovalOrRejectionNotification();
        }
        else if (userResponse == 2)
        {
            //set the option 'Rejected' as the status of the visit request
            visitRequestStatus = visitProgress[2];

            //send rejecting notification to the visitor
            sendApprovalOrRejectionNotification();
        }
        else
        {
            //show error message when user input is not within range
            printErrorOnUserReply();

            //Re-run when user input invalid character (recursive)
            setVisitStatus(passwordEntered);
        }
    } //end of if (when user enter correct password)
```

```
        //when user enter wrong password
        else
        {
            /*allow admin to attemp wrong password upto 3 more times before logout*/
            if (trialAttemp > 0)
            {
                --trialAttemp;
                cout << "Invalid authorization code, please try again.\n\n";

                //Re-run when again to accept another password (recursive)
                loginAsSecretaryOrReceptionist();
            }
            else
            {
                cout << "You\'re not authorized to get access!\n";
            }
        } //end of authenticating authorized code

    } //end of setVisitStatus

    /* Send notification to visitor when
    his/her request is approved or rejected*/
    void sendApprovalOrRejectionNotification()
    {
        string message;

        cout << "\n...sending notification to " << visitorPhoneNumber << endl;

        if (visitRequestStatus == "Approved")
        {
            message = ", we\'re pleased to inform you that your request "
                      "has been APPROVED.";
        }
        else
        {
            message = ", after reviewing your request, we have determined "
                      "that it would not be possible to accommodate your "
                      "request at this time and therefore, it\'s denied.";
        }
        cout << "Dear " << getVisitorFullName() << message << endl;

    } //end of printVisitRequestNotification

public:
    /* Prompt for user input and set it as
    the visitor's personal information*/
    void setVisitorPersonalInformation()
    {
        string full_name, phone_number, contact_address;

        cout << "Enter visitor\'s full name: ";
        getline(cin, full_name);

        cout << "Enter visitor\'s phone number: ";
        getline(cin, phone_number);

        cout << "Enter visitor\'s contact address: ";
        getline(cin, contact_address);

        //updating the class members variable with the user inputs
        visitorFullName = full_name;
        visitorPhoneNumber = phone_number;
        visitorContactAddress = contact_address;

    } //end of setVisitorPersonalInformation

    /* These routine (getters) return the
    visitor's individual personal information*/
```

```
string getVisitorFullName()
{
    return visitorFullName;
}

string getVisitorPhoneNumber()
{
    return visitorPhoneNumber;
}

string getVisitorContactAddress()
{
    return visitorContactAddress;
}

/* Prompt for visiting purpose and set it as
the purpose of visiting*/
void setPurposeOfVisiting()
{
    cout << "\nPlease select purpose of visit:\n";

    //iterate and display options for user to select purpose of visiting
    for (int i = 0; i < 6; i++)
    {
        cout << i + 1 << ". " << visitingPurpose[i] << "\n";
    }

    //Prompt for user response
    cout << "Reply: ";
    cin >> userResponse;

    /* If user reply is from 1,2...5 then the user response
    should be subtracted by 1 since array start from zero, and
    store the selected element as the purpose of visit */
    if (userResponse < 6 && userResponse >= 1)
    {
        //set if selected option is within option range
        purposeOfVisiting = visitingPurpose[userResponse - 1];
    }

    /* If user reply is exactly 6
    (indicate 'other' as purpose of visit), then
    Prompt the user for an input in one word*/
    else if (userResponse == 6)
    {
        //prompt and set user input if he/she selected 'other'
        cout << "Please enter purpose of visiting: ";
        cin >> purposeOfVisiting;
    }

    /* This indicate that user response
    is out of range*/
    else
    {
        //show error message when user input is not within range
        printErrorOnUserReply();

        //Re-run when user input invalid character (recurrence)
        setPurposeOfVisiting();
    }
} //end of setPurposeOfVisiting

/* This return the purpose of visiting */
string getPurposeOfVisiting()
{
    return purposeOfVisiting;
}

/* Prompt for convenient time of visit */
```

```
    void setTimeOfVisiting()
    {
        cout << "\nPlease choose convenient time (Duration: 30min) of visit:\n";

        //iterate and display options for user to select a convenient time
        for (int i = 0; i < 9; i++)
        {
            cout << i + 1 << ". " << visitingTime[i] << "\n";
        }

        //Prompt for user response
        cout << "Reply: ";
        cin >> userResponse;

        /* If user reply is from 1,2...9 then the user response
        should be subtracted by 1 since array start from zero, and
        store the selected element as the convenient time of visit */
        if (userResponse < 10 && userResponse >= 1)
        {
            //set selected option if reply is within option range
            timeOfVisiting = visitingTime[userResponse - 1];
        }
        //user response is out of range
        else
        {
            //show error message when user input is not within range
            printErrorOnUserReply();

            //Re-run when user input invalid character (recurrence)
            setTimeOfVisiting();
        }
    } //end of setTimeOfVisiting

    /* This return the convenient time of visiting */
    string getTimeOfVisiting()
    {
        return timeOfVisiting;
    }

    /* Display an error message when a user
    response with an option that is out of range */
    void printErrorOnUserReply()
    {
        cout << "Invalid response! Please reply with valid option." << endl;
    }

    /* Display an success message when
    a user request is logged in */
    void communicateVisitRequestToVisitor()
    {
        cout << "Dear " << getVisitorFullName() << ", your request has been submitted
successful.\n";
    } //end of printVisitRequestNotification

    /* Display visitor's personal information as well as the
    visiting information and request status per individual */
    void checkVisitorRequestStatus()
    {
        cout << "\nFull name: "
             << "\t\t" << getVisitorFullName() << "\nPhone number: "
             << "\t\t" << getVisitorPhoneNumber() << "\nContact address: "
             << "\t" << getVisitorContactAddress() <<

            "\nPurpose of visiting: "
             << "\t" << getPurposeOfVisiting() << "\nTime of visiting: "
             << "\t"
             << "on ";
        setDateOfVisiting();
```

```
        cout << " from " << getTimeOfVisiting() << "\nRequest status: "
             << "\t" << getVisitRequestStatus() << "";

        cout << "\n-----------------------------------------------------------";
    } //end of reviewVisitorRequest

    /* Prompt for authorization code to manage
    visiting request on successfully authentication */
    void loginAsSecretaryOrReceptionist()
    {
        string password;
        cout << "Please enter authorization code to get access (hint: admin): ";
        getline(cin, password);

        /*invoke and authenticate the authorization
        code entered by the user*/
        setVisitStatus(password);
    }

    /* This return the status of visiting request */
    string getVisitRequestStatus()
    {
        return visitRequestStatus;
    }
};

int main()
{
    // Creating an object of Visitors
    VisitorsManagementSystem person1;

    /* Prompt for visitor's personal information */
    cout << "### Step A: Personal Details ###\n";
    person1.setVisitorPersonalInformation();

    /* Prompt user with options to set visiting schedule */
    cout << "\n### Step B: Visit Schedule ###";
    person1.setPurposeOfVisiting();

    //set today's date
    person1.setTimeOfVisiting();

    cout << "\n";
    person1.communicateVisitRequestToVisitor();

    //display visitors logs before admin action
    person1.checkVisitorRequestStatus();

    //
    cout << "\n### Secretary/Receptionist Dashboad: Login to manage visit request ###\n";
    person1.loginAsSecretaryOrReceptionist();

    //display visitors logs after admin action
    person1.checkVisitorRequestStatus();

    return 0;
}
```