**Final Report**

**Development of Calendar Application**

**Introduction:**

The purpose of this project was to develop a calendar application in **C++** that allows users to manage events, view calendars for specific months and years, and log in securely to access their personalized calendars.

**Project Scope:**

The scope of the project included:

- Implementation of a calendar system capable of displaying months, years, and days.

- User authentication system to access personalized calendars.

- Addition of events to specific dates.

- Displaying events for a given date.

**Technologies Used:**

**- C++:** Chosen as the programming language due to its versatility and efficiency for system-level programming.

**- Object-Oriented Programming (OOP) Principles:** Employed to create modular and reusable code, enhancing maintainability and scalability.

**Key Features Implemented:**

**- User Authentication**: Implemented a login system requiring a unique account number and password for access.

**- Calendar Generation:** Developed functions to generate calendars for specific months and years, including support for leap years.

**- Event Management:** Enabled users to add events to specific dates, with validation to ensure accurate date-time input.

**Design and Implementation:**

**- Class Structure:** Designed a class hierarchy with a base **`Calendar`** class and a derived **`calendar\_2024`** class for specific calendar functionalities.

**- Data Structures:** Used arrays to store account information and event details efficiently.

**- Algorithms:** Employed algorithms to calculate leap years, determine the number of days in a month, and find the starting day of a month.

**Challenges Faced:**

**- Date Validation:** Ensuring accurate date-time input and validating it against the current date posed a challenge due to the complexity of handling different date formats and leap years.

**-User Interface:** Designing a user-friendly interface while adhering to the command-line environment was challenging, requiring clear prompts and error messages.

**Conclusion:**

In conclusion, the development of the calendar application project provided valuable insights into C++ programming and OOP principles. Despite challenges, the project achieved its objectives and laid a foundation for future enhancements and improvements.

**References:**

- [C++ Standard Template Library Documentation] (https://en.cppreference.com/w/cpp/header)

- [Object-Oriented Programming in C++] (https://www.learncpp.com/cpp-tutorial/)

- [C++ Programming Language] (https://www.cplusplus.com/doc/)

- [Leap Year Calculation Algorithm] (https://en.wikipedia.org/wiki/Leap\_year#Algorithm)

**Application Code & Output**

#include <iostream>

#include <bits/stdc++.h>

using namespace std;

class calendar\_2024;

class Calendar {

private:

static const int MAX\_ACCOUNTS = 1000;

struct Account

{

string AcNumber;

string password;

string name;

};

Account accounts[MAX\_ACCOUNTS];

int accountCount = 3;

int changeACnumber = 0;

int month;

int year;

public:

string Month[12] = {"January", "February", "March", "April", "May", "June", "July", "August", "September", "October", "November", "December"};

Calendar()

{

initializeAccounts(); }

void initializeAccounts() {

accounts[0] = {"01406501", "Rafi014", "Tahamid hasan rafi"};

accounts[1] = {"01795561", "Alvi017", "Alvi hasan"};

accounts[2] = {"016", "016", "Kamrul Hasan"}; }

void getData(){

string response;

cout << "---WELCOME ABS() CALENDAR---\nDo you have any Account in this Calendar?\n"

<< "'yes' "

<< "or "

<< "'no'\n";

cin >> response;

if (response == "y" || response == "Y" || response == "Yes") {

getExistingAccountData();

}

else if (response == "n" || response == "N" || response == "NO") {

createNewAccount();

}

else {

cout << "Type 'yes' or 'no'\n:";

getData();

}

}

void OpenCalendar() {

cout << "Calendar is open!" << endl;

int month, year;

cout << "Enter month (1-12): ";

cin >> month;

cout << "Enter year: ";

cin >> year;

Calendar cal(month, year);

cout << endl;

if (month == 1) {

cout << " Calendar: " << Month[0] << "/" << year << endl;

}

else if (month == 2){

cout << " Calendar: " << Month[1] << "/" << year << endl; }

else if (month == 3){

cout << " Calendar: " << Month[2] << "/" << year << endl;

}

else if (month == 4) {

cout << " Calendar: " << Month[3] << "/" << year << endl;

}

else if (month == 5) {

cout << " Calendar: " << Month[4] << "/" << year << endl;

}

else if (month == 6) {

cout << " Calendar: " << Month[5] << "/" << year << endl;

}

else if (month == 7) {

cout << " Calendar: " << Month[6] << "/" << year << endl;

}

else if (month == 8) {

cout << " Calendar: " << Month[7] << "/" << year << endl;

}

else if (month == 9){

cout << " Calendar: " << Month[8] << "/" << year << endl;

}

else if (month == 10) {

cout << " Calendar: " << Month[9] << "/" << year << endl;

}

else if (month == 11){

cout << " Calendar: " << Month[10] << "/" << year << endl;

}

else if (month == 12) {

cout << " Calendar: " << Month[11] << "/" << year << endl;

}

cal.print();}

void getExistingAccountData(){

string enteredNumber, enteredPassword;

cout << "Enter your AC number:";

cin >> enteredNumber;

cout << "Enter your Password:";

cin >> enteredPassword;

matchData(enteredNumber, enteredPassword);

}

void matchData(string enteredNumber, string enteredPassword);

void createNewAccount(){

changeACnumber = changeACnumber + 188;

accountCount++;

if (accountCount < MAX\_ACCOUNTS){

int phone;

int index = accountCount;

cout << "Enter your name:";

cin >> ws;

getline(cin, accounts[index].name);

cout << "Enter your password:";

cin >> accounts[index].password;

cout << "Enter your phone number:";

cin >> phone;

accounts[index].AcNumber = generateAccountNumber();

cout << "\n-After creating a new account-\n"

<< "AC number: " << accounts[index].AcNumber << "\n"

<< "AC name: " << accounts[index].name << "\n"

<< "AC password: " << accounts[index].password << "\n";

cout << "-----THANK YOU!-----\n";

getData();

}

else{

cout << "Can't create a new account. Maximum limit you have reached.\n";

} }

string generateAccountNumber() {

return "1" + to\_string(changeACnumber) + "000";

}

bool isLeapYear(int year){

return (year % 4 == 0 && year % 100 != 0) || (year % 400 == 0);

}

int getDaysInMonth(int month, int year){

switch (month) {

case 4:

case 6:

case 9:

case 11:

return 30;

case 2:

return isLeapYear(year) ? 29 : 28;

default:

return -1; }

case 1:

case 3:

case 5:

case 7:

case 8:

case 10:

case 12:

return 31;

}

int getStartingDay(int month, int year){

if (month < 3){

month += 12;

year--;}

int k = year % 100;

int j = year / 100;

return (1 + 13 \* (month + 1) / 5 + k + k / 4 + j / 4 + 5 \* j) % 7; }

Calendar(int m, int y) : month(m), year(y) {}

void print(){

cout << " " << setw(5) << "SAT" << setw(5) << "SUN" << setw(5) << "MON" << setw(5)

<< "TUE" << setw(5) << "WED" << setw(5) << "THU" << setw(5) << "FRI" << endl;

int daysInMonth = getDaysInMonth(month, year);

int startingDay = getStartingDay(month, year);

// Adjust starting position

for (int i = 0; i < startingDay; ++i){

cout << setw(5) << ""; }

// Print days sequentially

for (int day = 1; day <= daysInMonth; ++day) {

cout << setw(5) << day;

// If it's the last day of the week, start a new line

if ((startingDay + day) % 7 == 0) {

cout << endl;}

}

cout << endl;

addNewEvent(); }

bool isDateValid(string date){

time\_t now = time(0);

tm \*ltm = localtime(&now);

int currentYear = 1900 + ltm->tm\_year;

int currentMonth = 1 + ltm->tm\_mon;

int currentDay = ltm->tm\_mday;

int enteredDay, enteredMonth, enteredYear, enteredHour, enteredMinute;

sscanf(date.c\_str(), "%d/%d/%d %d:%d", &enteredDay, &enteredMonth, &enteredYear, &enteredHour, &enteredMinute);

if (enteredYear < currentYear || (enteredYear == currentYear && enteredMonth < currentMonth) || (enteredYear == currentYear && enteredMonth == currentMonth && enteredDay < currentDay))

return false;

else

return true;}

void addNewEvent() {

string date, time, eventDescription;

cout << "Enter Event date and time (format: DD/MM/YYYY HH:MM): ";

cin >> date;

if (!isDateValid(date)) {

cout << "Invalid date and time. Please enter date and time based on current date and time." << endl;

addNewEvent(); }

else {

cout << "Enter event description: ";

cin.ignore();

getline(cin, eventDescription);

cout << "Event added successfully!" << endl;

} } };

class calendar\_2024 : public Calendar{

public:

bool isLeapyear242024(int year24) {

if ((year24 % 4 == 0 && year24 % 100 != 0) || (year24 % 400 == 0))

return true;

else

return false;

}

int daysInmonth242024(int month24, int year24){

if (month24 == 2){

if (isLeapyear242024(year24))

return 29;

else

return 28;

}

else if (month24 == 4 || month24 == 6 || month24 == 9 || month24 == 11)

return 30;

else

return 31;

}

void printCalendar2024() {

int year = 2024;

cout << "CALENDAR FOR YEAR " << year << endl;

cout << "===================" << endl;

string months[12] = {"January", "February", "March", "April", "May", "June",

"July", "August", "September", "October", "November", "December"};

int daysInMonth[12] = {31, 29, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};

int startDay = 2; // January 1, 2024 is a Tuesday (0 for Sunday, 1 for Monday, ..., 6 for Saturday)

for (int month = 0; month < 12; ++month)

{

cout << endl << months[month] << endl;

cout << "---------------------" << endl;

cout << "Sun Mon Tue Wed Thu Fri Sat" << endl;

// Print leading spaces for the first week

for (int i = 0; i < startDay; ++i)

cout << " ";

// Print days

for (int day = 1; day <= daysInMonth[month]; ++day) {

printf("%3d ", day);

startDay++;

if (startDay % 7 == 0)

cout << endl;

}

// If the last week is not complete, move to the next line

if (startDay % 7 != 0)

cout << endl;

// Update startDay for the next month

startDay %= 7;

}}};

void Calendar::matchData(string enteredNumber, string enteredPassword) {

for (int i = 0; i <= accountCount; i++) {

if (enteredNumber == accounts[i].AcNumber && enteredPassword == accounts[i].password){

calendar\_2024 calendar2024;

calendar2024.printCalendar2024(); // Call without any arguments

OpenCalendar();

return;

} }

cout << "Account number and password do not match !! \n";

cout << "Enter valid AC number and password:\n";

getExistingAccountData();

}

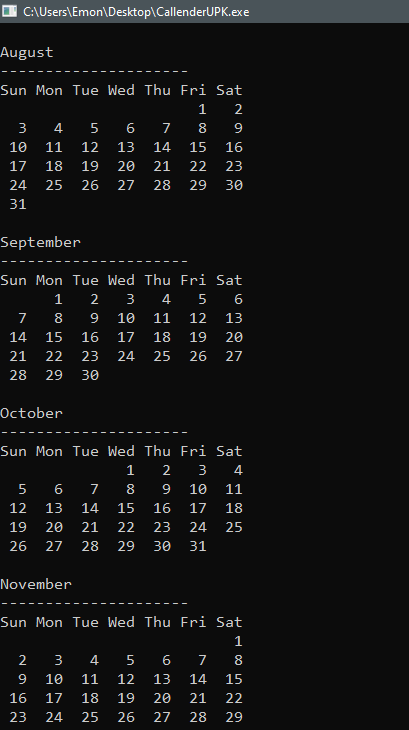
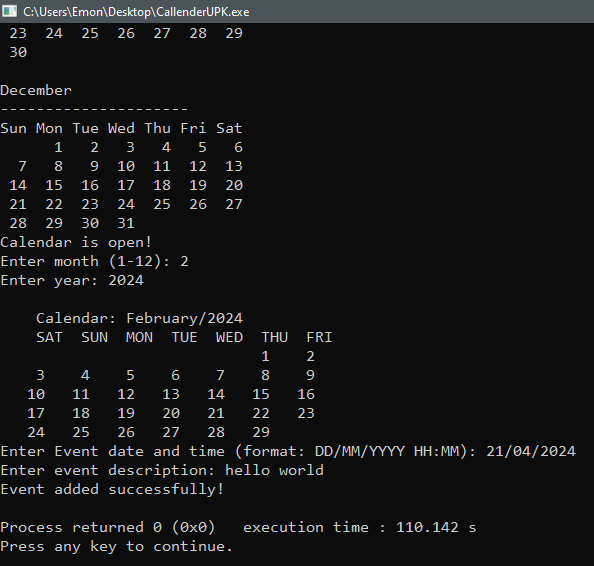
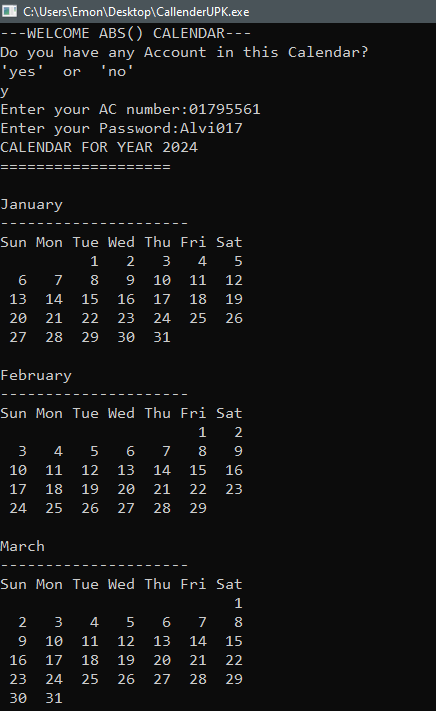
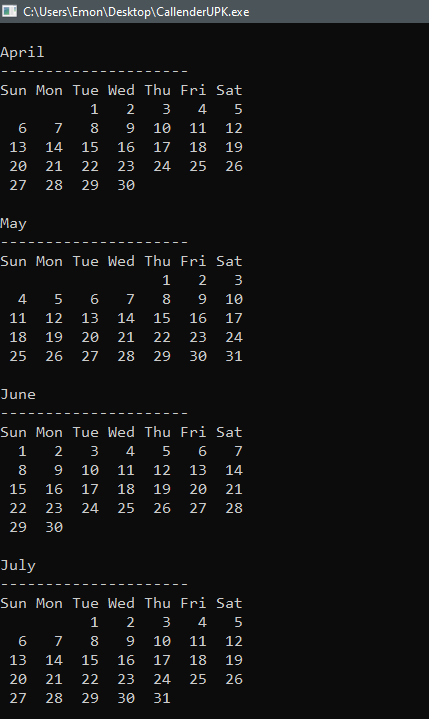
int main() {

Calendar c;

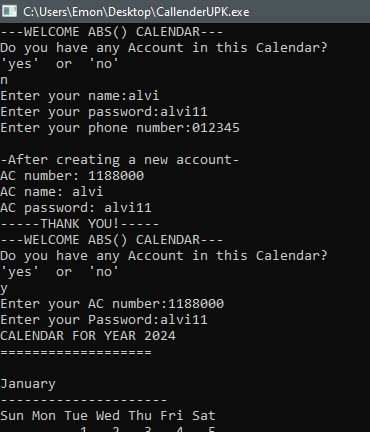
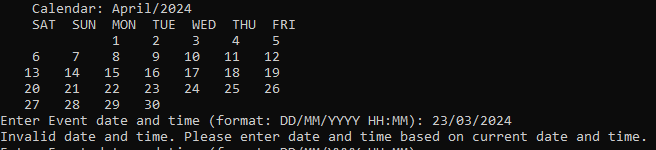
c.getData();

return 0;

}



When the response Yes and enter the current date



When the response No and enter the invalid event date