**REPORT**

**“STUDENT DATABASE WITH GRAPHICAL USER INTERFACE”**

**BY:**

**SANCHIT KULKARNI**

**PROBLEM STATEMENTS**

**1**

A student database using classes with dynamic memory allocation to store student’s Name, Date of Birth and automatically assign roll number.

**2**

Create a GUI based application to store student information such as Name, Date of Birth and Marks to a file while automatically assigning each student a roll number.

**Code::Blocks 17.12**

Code::Blocks is an IDE packed full of all the features you will need. It has a consistent look, feel and operation across its supported platforms. The application has been designed to be very extensible and fully configurable.

It has been built around a plugin framework, therefore Code::Blocks can be extended with plugins. Support for any kind of functionality can be added by installing/coding a plugin.

Code::Blocks' interface is both customizable and extensible with Syntax highlighting, a tabbed interface, Class Browser, a to-do list manager with different users, and many more features that are provided through plugins.

As of version 17.12, Code::Blocks comes with a GUI designer called Win32.

**Win32**

Win32 is the 32-bit application programming interface (API) for versions of Windows from 95 onwards. The API consists of functions implemented, as with Win16, in system DLLs. The core DLLs of Win32 are kernel32.dll, user32.dll, and gdi32.dll. Win32 was introduced with Windows NT. The version of Win32 shipped with Windows 95 was initially referred to as Win32c, with *c* meaning *compatibility*. This term was later abandoned by Microsoft in favor of Win32.

**AAT 1**

The following program accepts the user input of student name and date of birth and automatically assigns each student a roll number while dynamically allocating memory to the student structure variables. Number of maximum students to be entered.

**CODE**

#include <iostream>

#define maxim 5 //maximum entries macro

using namespace std;

class stud

{

char name[30];

int dob;

int rollno;

static int i;

public:

void enter()

{

cin>>name>>dob;

rollno=++i; //roll no assigned

}

void display()

{

cout<<"Name:"<<name<<"\n";

cout<<"DOB:"<<dob<<"\n";

cout<<"Rollno:"<<rollno;

}

};

int stud::i; // i is static

int main()

{

int ele,max=0,b; //ele are no of entries max is a flag which starts from 0 goes till maxim

stud \*p,\*q; //p q two pointer obj of class stud

here : cout<<"enter the no of students\n";

cin>>ele;

if(ele>maxim)

{

cout<<"Error more elements cannot be stored\n";

goto here;

}

p=new stud[ele]; //memory allocated as per the no of inputs given by user

max=max+ele;

cout<<"enter the name and dob\n";

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

{

p[i].enter();

}

while(max<maxim)

{

cout<<"enter the no of students\n";

cin>>ele;

if((max+ele)>maxim)

{

cout<<"Error more elements cannot be stored\n";

continue;

}

q=new stud[max+ele];

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

{

q[i]=p[i]; //q takes values of p

}

cout<<"enter the name and dob\n";

for(int i=max;i<max+ele;i++)

{

q[i].enter(); //remaining memory of q is filled by user

}

delete p;

p=q; //q stores back to p

max=max+ele;

}

cout<<"enter the pos\n";// pos to display

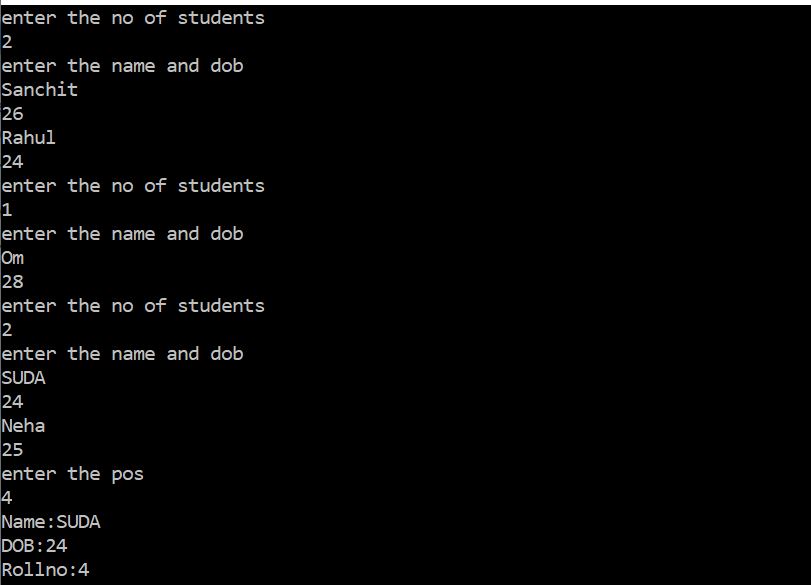
cin>>b;

p[b-1].display();

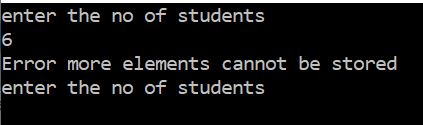
return 0;

}

**OUTPUT**



If the user exceeds the maximum limit,



**2**

The program takes in Name, Date of Birth and Marks as the input from the user and automatically assigns a roll number and stores it to a .csv file. This file can be opened as a Microsoft Excel Spread Sheet and can be modified.

**CODE**

#include <iostream>

#include<fstream>

#include<conio.h>

#include <windows.h>

#include <stdlib.h>

#include <string>

#include<sstream>

#define ENTER\_DETAILS 1

#define CHECK\_DETAILS 2

#define EXIT\_KEY 3

#define SUBMIT\_DETAILS 4

#define SUBMIT\_ROLLNO 5

#define SUBMIT\_STUDENTSNO 6

#define next 7

using namespace std;

//////////////////////////////////FUNCTIONS/////////////////////////////

void passwordDialog1(HWND);

void enter(int);

void checkDetails(int);

void AddMenus(HWND);

void AddControls(HWND);

LRESULT CALLBACK DialogProcedure1(HWND,UINT,WPARAM,LPARAM);

void registerDialogClass1(HINSTANCE);

void displayDialog1(HWND);

LRESULT CALLBACK DialogProcedure2(HWND,UINT,WPARAM,LPARAM);

void registerDialogClass2(HINSTANCE);

void displayDialog2(HWND);

LRESULT CALLBACK DialogProcedure3(HWND,UINT,WPARAM,LPARAM);

void registerDialogClass3(HINSTANCE);

void displayDialog3(HWND);

LRESULT CALLBACK WProc(HWND,UINT,WPARAM,LPARAM);

HWND hMain,hDlg1,hDlg2,hDlg3,hOut,hOut1,hName,hDOB,hRno,hNo,hpass,hkey;

HMENU hMenu;

int flag;

char name[20];

char pass[20];

char DOB[11];

char namef[20];

char DOBf[11];

char stno[4];

int student\_no;

int rno;

int k=0;

int maxim=0;

int maxilim=8;

ofstream myFile; // TO WRITE IN FILES///

struct student

{

char DOBS[8];

char nameS[20];

int roll\_no;

}\*temp,\*fina,\*mid;

//////////////////////////////////ENTER STUDENT DETAILS/////////////////////////////

void enter(int n4)

{

if(k==0)

{

temp=new student;

fina=temp;

}

for(int i=0;i<n4;i++)

{

strcpy((\*temp).DOBS,DOB);

strcpy((\*temp).nameS,name);

(\*temp).roll\_no=k++;

temp++;

}

}

/\*//////////////////////////////CHECK DETAILS//////////////////////////////////\*/

void checkDetails(int n4)

{

if(n4<=k)

{

temp=fina;

temp+=n4-1;

cout<<"Student Name :"<<(\*temp).nameS<<endl;

cout<<"Student Date Of Birth :"<<(\*temp).DOBS<<endl;

}

else

{

cout<<"Student with "<<" Roll Number "<<n4<<"does not exist."<<endl;

}

temp=fina;

for(int j=0;j<k;j++)

{

delete temp;

temp++;

}

delete fina;

}

/\*//////////////////////////////////// CREATING MENU BAR////////////////////////////////////////\*/

void AddMenus(HWND hWnd)

{

hMenu =CreateMenu();

HMENU hOptionsMenu=CreateMenu();

HMENU hSubMenu=CreateMenu();

AppendMenu(hSubMenu,MF\_STRING,ENTER\_DETAILS,"Enter");

AppendMenu(hSubMenu,MF\_STRING,CHECK\_DETAILS,"Check");

AppendMenu(hOptionsMenu,MF\_POPUP,(UINT\_PTR)hSubMenu,"Details");

AppendMenu(hOptionsMenu,MF\_STRING,EXIT\_KEY,"Exit");

AppendMenu(hMenu,MF\_POPUP,(UINT\_PTR)hOptionsMenu,"Options");

SetMenu(hWnd,hMenu);

}

//////////////////////////////////BUTTONS///////////////////////////////

void AddControls(HWND hWnd)

{ CreateWindowW(L"static",L"DATABASE APP",WS\_VISIBLE|WS\_CHILD,70,20,400,20,hWnd,NULL,NULL,NULL);

CreateWindowW(L"BUTTON",L" ADMIN",WS\_VISIBLE|WS\_CHILD|WS\_BORDER|SS\_CENTER,40,80,200,50,hWnd,(HMENU)ENTER\_DETAILS,NULL,NULL);

CreateWindowW(L"BUTTON",L" DATABASE",WS\_VISIBLE|WS\_CHILD|WS\_BORDER|SS\_CENTER,40,150,200,50,hWnd,(HMENU)CHECK\_DETAILS,NULL,NULL);

}

////////////////////////////////// ENTRY WINDOW /////////////////////////////////////

LRESULT CALLBACK DialogProcedure1(HWND hWnd,UINT msg,WPARAM wp,LPARAM lp)

{

switch(msg)

{

case WM\_CLOSE:

DestroyWindow(hWnd);

EnableWindow(hMain,true);

break;

case WM\_COMMAND:

switch(wp)

{

case next:

GetWindowText(hkey,pass,20);

displayDialog1(hWnd);

break;

case SUBMIT\_DETAILS:

GetWindowText(hName,name,20);

GetWindowText(hDOB,DOB,10);

int flag=0;

int i;

for(i=0;i<20;i++)

{

if((name[i]>=33 && name[i]<=64)) //TO TAKE ONLY VALID INPUTS////

{

flag=1;

break;

}

}

for(i=0;i<2;i++)

{

if((DOB[i]>=48 && DOB[i]<=57))//TO TAKE ONLY VALID INPUTS////

{

continue;

}

else

{

flag=1;

break;

}

}

if(flag==1)

{

MessageBoxW(hDlg1,L"Please fill valid details to proceed",L"Invalid Input",MB\_OK|MB\_ICONERROR);

return 0;

}

if(strcmp(name,"")==0||strcmp(DOB,"")==0)

{ MessageBoxW(hDlg1,L"Please fill the details to proceed",L"Missing Input",MB\_OK|MB\_ICONERROR);

return 0;

}

else

{ if(k==0)

{

temp=new student;

fina=temp;

mid=temp;

}

temp=mid;

strcpy((\*temp).DOBS,DOB);

strcpy((\*temp).nameS,name);

k+=1;

(\*temp).roll\_no=k;

temp++;

mid=temp;

myFile<<name<<","<<DOB<<endl; //WRITING IT TO A FILE////

}

DestroyWindow(hWnd);

DestroyWindow(hpass);

break;

}

break;

default:

return DefWindowProcW(hWnd,msg,wp,lp);

}

}

void registerDialogClass1(HINSTANCE hInst)

{

WNDCLASSW dialog1={0};

dialog1.hbrBackground=(HBRUSH)COLOR\_WINDOW;

dialog1.hCursor=LoadCursor(NULL,IDC\_CROSS);

dialog1.hInstance=hInst;

dialog1.lpszClassName=L"MyDialogClass1";

dialog1.lpfnWndProc=DialogProcedure1;

RegisterClassW(&dialog1);

}

void passwordDialog1(HWND hWnd)

{

hpass=CreateWindowW(L"MyDialogClass1",L"Enter password :",WS\_VISIBLE|WS\_OVERLAPPEDWINDOW,100,100,400,300,hWnd,NULL,NULL,NULL);

hkey=CreateWindowW(L"edit",L"",WS\_VISIBLE|WS\_CHILD|WS\_BORDER|ES\_MULTILINE|ES\_AUTOVSCROLL,155,105,100,50,hpass,NULL,NULL,NULL);

CreateWindowW(L"static",L"PASSWORD:",WS\_VISIBLE|WS\_CHILD,50,105,100,50,hpass,NULL,NULL,NULL);

CreateWindowW(L"Button",L"VERIFY",WS\_VISIBLE|WS\_CHILD,132,160,150,50,hpass,(HMENU)next,NULL,NULL);

}

void displayDialog1(HWND hWnd)

{

hDlg1=CreateWindowW(L"MyDialogClass1",L"Please enter Student Details",WS\_VISIBLE|WS\_OVERLAPPEDWINDOW,100,100,400,300,hWnd,NULL,NULL,NULL);

CreateWindowW(L"static",L"STUDENT INFORMATION ENTRY ",WS\_VISIBLE|WS\_CHILD,100,20,400,20,hDlg1,NULL,NULL,NULL);

CreateWindowW(L"static",L"Name:",WS\_VISIBLE|WS\_CHILD,50,50,100,50,hDlg1,NULL,NULL,NULL);

hName=CreateWindowW(L"edit",L"",WS\_VISIBLE|WS\_CHILD|WS\_BORDER|ES\_MULTILINE|ES\_AUTOVSCROLL,155,50,100,50,hDlg1,NULL,NULL,NULL);

CreateWindowW(L"static",L"DOB:",WS\_VISIBLE|WS\_CHILD,50,105,100,50,hDlg1,NULL,NULL,NULL);

hDOB=CreateWindowW(L"edit",L"",WS\_VISIBLE|WS\_CHILD|WS\_BORDER|ES\_MULTILINE|ES\_AUTOVSCROLL,155,105,100,50,hDlg1,NULL,NULL,NULL);

CreateWindowW(L"Button",L"Submit",WS\_VISIBLE|WS\_CHILD,132,160,150,50,hDlg1,(HMENU)SUBMIT\_DETAILS,NULL,NULL);

}

/\*/////////////////////////////////GETTING DETAILSFROM USER////////////////////////////////////\*/

LRESULT CALLBACK DialogProcedure2(HWND hWnd,UINT msg,WPARAM wp,LPARAM lp)

{

switch(msg)

{

case WM\_CLOSE:

DestroyWindow(hWnd);

EnableWindow(hMain,true);

break;

case WM\_COMMAND:

switch(wp)

{

case SUBMIT\_ROLLNO:

char rollno[4];

GetWindowText(hRno,rollno,4);

rno=atoi(rollno);

if(strcmp(rollno,"")==0)

{ MessageBoxW(hDlg2,L"Please fill the details to proceed",L"Missing Input",MB\_OK|MB\_ICONERROR);

return 0;

}

else if(rno>k||rno==0)

{

MessageBoxW(hDlg2,L"Roll number does not exist , please enter a valid choice",L"Exceeding Limit",MB\_OK|MB\_ICONERROR);

return 0;

}

else

{ char out2[50]="NAME : ";

temp=fina;

temp+=rno-1;

strcpy(namef,(\*temp).nameS);

strcpy(DOBf,(\*temp).DOBS);

strcat(out2,namef);

strcat(out2," ");

strcat(out2," Date Of Birth : ");

strcat(out2,DOBf);

SetWindowText(hOut1,out2);

}

break;

}break;

default:

return DefWindowProcW(hWnd,msg,wp,lp);

}

}

void registerDialogClass2(HINSTANCE hInst)

{

WNDCLASSW dialog2={0};

dialog2.hbrBackground=(HBRUSH)COLOR\_WINDOW;

dialog2.hCursor=LoadCursor(NULL,IDC\_CROSS);

dialog2.hInstance=hInst;

dialog2.lpszClassName=L"MyDialogClass2";

dialog2.lpfnWndProc=DialogProcedure2;

RegisterClassW(&dialog2);

}

void displayDialog2(HWND hWnd)

{

hDlg2=CreateWindowW(L"MyDialogClass2",L" USER WINDOW ",WS\_VISIBLE|WS\_OVERLAPPEDWINDOW,100,100,500,500,hWnd,NULL,NULL,NULL);

CreateWindowW(L"static",L"STUDENT DETAILS",WS\_VISIBLE|WS\_CHILD,100,20,400,20,hDlg2,NULL,NULL,NULL);

CreateWindowW(L"static",L"POSITION:",WS\_VISIBLE|WS\_CHILD,50,70,100,50,hDlg2,NULL,NULL,NULL);

hRno=CreateWindowW(L"edit",L"",WS\_VISIBLE|WS\_CHILD|WS\_BORDER,155,70,100,50,hDlg2,NULL,NULL,NULL);

CreateWindowW(L"Button",L"SUBMIT",WS\_VISIBLE|WS\_CHILD,132,130,150,50,hDlg2,(HMENU)SUBMIT\_ROLLNO,NULL,NULL);

hOut1=CreateWindowW(L"edit",L"",WS\_VISIBLE|WS\_CHILD|WS\_BORDER|ES\_MULTILINE|ES\_AUTOVSCROLL,50,220,300,100,hDlg2,NULL,NULL,NULL);

EnableWindow(hWnd,false);

}

LRESULT CALLBACK DialogProcedure3(HWND hWnd,UINT msg,WPARAM wp,LPARAM lp)

{

switch(msg)

{

case WM\_CLOSE:

DestroyWindow(hWnd);

EnableWindow(hMain,true);

break;

case WM\_COMMAND:

switch(wp)

{

case SUBMIT\_STUDENTSNO:

char stno[4];

GetWindowText(hNo,stno,4);

student\_no=atoi(stno);

if(strcmp(stno,"")==0)

{ MessageBoxW(hDlg3,L"Please fill the details to proceed",L"Missing Input",MB\_OK|MB\_ICONERROR);

return 0;

}

else if(student\_no >60 )

{

MessageBoxW(hDlg3,L"slots unavailable ",L"Exceeding Limit",MB\_OK|MB\_ICONERROR);

return 0;

}

else

{ flag=1;

if(k==0)

{

temp=new student;

fina=temp;

}

for(int i=0;i<student\_no;i++)

{

displayDialog1(hWnd);

}

}

break;

}break;

default:

return DefWindowProcW(hWnd,msg,wp,lp);

}

}

void registerDialogClass3(HINSTANCE hInst)

{

WNDCLASSW dialog3={0};

dialog3.hbrBackground=(HBRUSH)COLOR\_WINDOW;

dialog3.hCursor=LoadCursor(NULL,IDC\_CROSS);

dialog3.hInstance=hInst;

dialog3.lpszClassName=L"MyDialogClass";

dialog3.lpfnWndProc=DialogProcedure3;

RegisterClassW(&dialog3);

}

void displayDialog3(HWND hWnd)

{

hDlg3=CreateWindowW(L"MyDialogClass",L" Number",WS\_VISIBLE|WS\_OVERLAPPEDWINDOW,200,200,280,200,hWnd,NULL,NULL,NULL);

CreateWindowW(L"static",L"No.Of Students:",WS\_VISIBLE|WS\_CHILD,20,50,150,20,hDlg3,NULL,NULL,NULL);

hNo=CreateWindowW(L"edit",L"",WS\_VISIBLE|WS\_CHILD|WS\_BORDER,180,50,50,20,hDlg3,NULL,NULL,NULL);

CreateWindowW(L"Button",L"Submit",WS\_VISIBLE|WS\_CHILD,80,90,100,50,hDlg3,(HMENU)SUBMIT\_STUDENTSNO,NULL,NULL);

EnableWindow(hWnd,false);

}

LRESULT CALLBACK WProc(HWND hWnd,UINT msg,WPARAM wp,LPARAM lp)

{

switch(msg)

{ case WM\_COMMAND:

switch (wp)

{

case ENTER\_DETAILS:

if(k==maxilim)

{

MessageBoxW(hWnd,L"Maximum reached",L"Limit",MB\_OK|MB\_ICONEXCLAMATION);

break;

}

else

{

passwordDialog1(hWnd);

break;

}

break;

case CHECK\_DETAILS:

displayDialog2(hWnd);

break;

case EXIT\_KEY:

int val;

val=MessageBoxW(hWnd,L"Are you sure to exit?",L"Exit",MB\_YESNO|MB\_ICONEXCLAMATION);

if (val==IDYES)

{

temp=fina;

for(int j=0;j<k;j++)

{

delete temp;

temp++;

}

delete fina;

DestroyWindow(hWnd);

}

break;

}

break;

case WM\_DESTROY:

PostQuitMessage(0);

break;

case WM\_CLOSE:

temp=fina;

for(int j=0;j<k;j++)

{

delete temp;

temp++;

}

delete fina;

DestroyWindow(hWnd);

break;

case WM\_CREATE:

AddMenus(hWnd);

AddControls(hWnd);

myFile.open("Database.csv"); //OPENING A EXEL FILE////

myFile<<"NAME:"<<","<<"DOB:"<<endl;//////NAME OF THE COLUMNS//////

break;

default:

return DefWindowProcW(hWnd,msg,wp,lp);

}

}

int WINAPI WinMain(HINSTANCE hInst,HINSTANCE hPrevInst,LPSTR args,int ncmdshow)

{

WNDCLASSW wc={0};

wc.hbrBackground=(HBRUSH)COLOR\_WINDOW;

wc.hCursor=LoadCursor(NULL,IDC\_ARROW);

wc.hInstance=hInst;

wc.lpszClassName=L"MyClass";

wc.lpfnWndProc=WProc;

if(!RegisterClassW(&wc))

return -1;

registerDialogClass1( hInst);

registerDialogClass2( hInst);

hMain=CreateWindowW(L"MyClass",L"Student DataBase",WS\_OVERLAPPED|WS\_VISIBLE,50,50,300,300,NULL,NULL,NULL,NULL);

MSG msg={0};

while(GetMessage(&msg,NULL,NULL,NULL))

{

TranslateMessage(&msg);

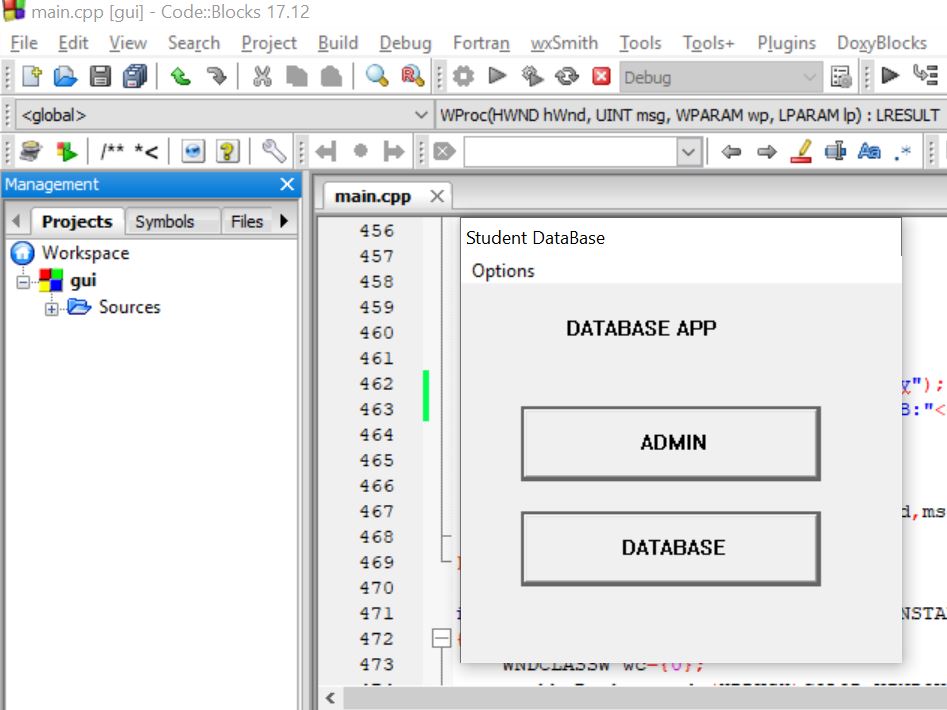
DispatchMessage(&msg);

}

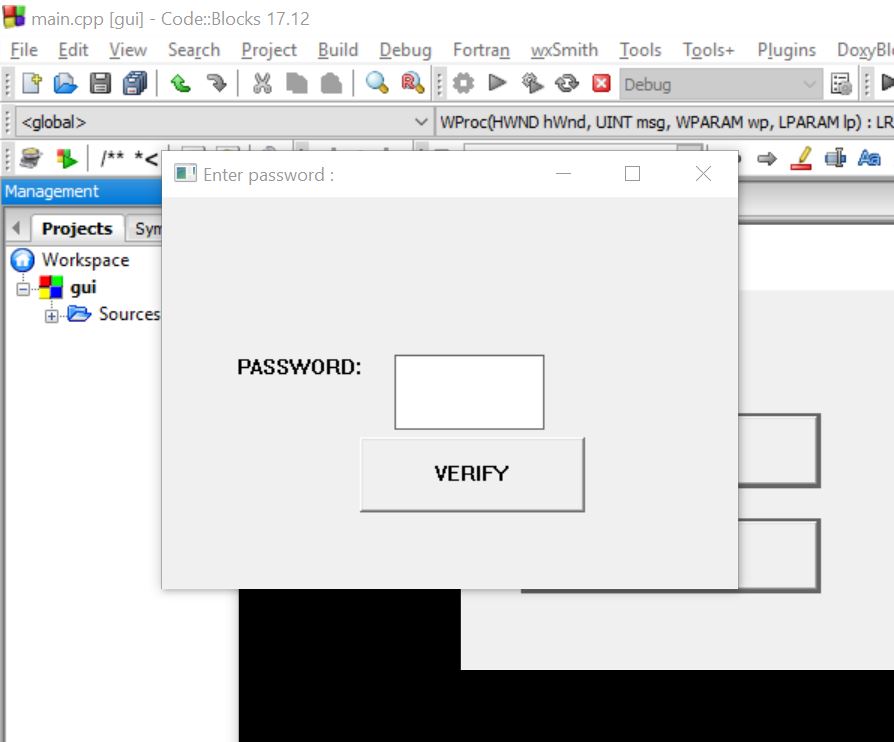
return 0;

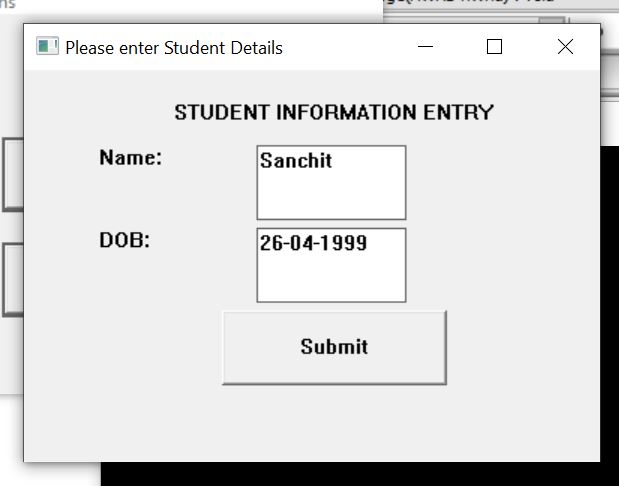
}

**OUTPUT**

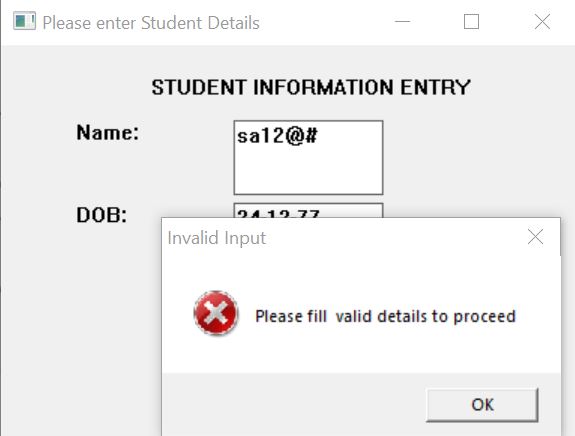


When the user clicks on ADMIN to store student data, it asks to enter password

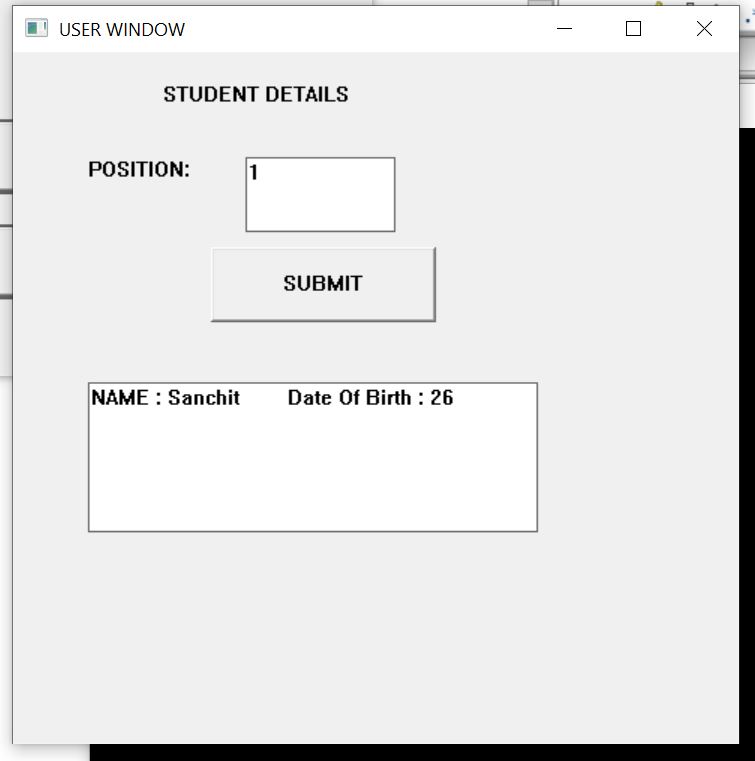




When the user gives invalid details



If the user clicks DATABASE to get student details, it asks to enter student roll number



All the student details in database are stored in excel file

