UNIVERSITY MANAGEMENT SYSTEM full report

ABSTRACT

UNIVERSITY MANAGEMENT SYSTEM (UMS) deals with the maintenance of university coordinators, instructors and student information with in the university. UMS is an automation system, which is used to store the information about , faculty, student, courses e.t.c.

Starting from registration of a new student in the university, it maintains all the details regarding the marks of the students and the courses that he/she registered. The project deals with retrieval of information through a OIBS based campus wide portal. It collects related information from all the departments of an organization and maintains files, which are used to generate reports in various forms to measure individual and overall performance of the students. Development process of the system starts with System analysis. System analysis involves creating a formal model of the problem to be solved by understanding requirements.

PURPOSE OF THE SYSTEM

UNIVERSITY MANAGEMENT SYSTEM [UMS] deals with the maintenance of coordinators, instructors and student information with in the university. This project of UMS involved the automation of student information that can be implemented in different college managements like cordinators of certain faculity are able to change informations about students or instructors in that faculity at the same time system gives them opportunity check which courses assigned to him/her or which courses he/she registered to.The project deals with retrieval of information through an OIBS based campus wide portal. It collects related information from all the departments of an organization and maintains files, which are used to generate reports in various forms to measure individual and overall performance of the students.

EXISTING SYSTEM

The system starts with registration of new staff and students. When the subjects are to be allocated to the faculty, the Head of the Department should enter everything in the Excel sheets. Then the staff enters corresponding subject's attendance and marks of a student then those must also be entered in the Excel sheets and validations are to be done by the user itself. So there will be a lot of work to be done and must be more conscious during the entrance of details. So, more risk is involved.

INTRODUCTION

After analyzing the requirements of the task to be performed, the next step is to analyze the problem and understand its context. The first activity in the phase is studying the existing system and other is to understand the requirements and domain of the new system. Both the activities are equally important, but the first activity serves as a basis of giving the functional specifications and then successful design of the proposed system. Understanding the properties and requirements of a new system is more difficult and requires creative thinking and understanding of existing running system is also difficult, improper understanding of present system can lead diversion from solution.

ANALYSING MODEL

The model that is basically being followed is the WATER FALL MODEL, which states that the phases are organized in a linear order. First of all the feasibility study is done. Once that part is over the requirement analysis and project planning begins. The design starts after the requirement analysis is complete and the coding begins after the design is complete. Once the programming is completed, the testing is done. In this model the sequence of activities performed in a software development project are:

- Requirement Analysis
- Project Planning
- System design
- Detail design
- Coding
- Unit testing
- System integration & testing

Here the linear ordering of these activities is critical. End of the phase and the output of one phase is the input of other phase. The output of each phase is to be consistent with the overall requirement of the system.

Some of the qualities of spiral model are also incorporated like after the people concerned with the project review completion of each of the phase the work done. WATER FALL MODEL was being chosen because all requirements were known beforehand and the objective of our software development is the computerization/automation of an already existing manual working system.

FEASIBILITY STUDY

Preliminary investigation examine project feasibility, the likelihood the system will be useful to the organization. The main objective of the feasibility study is to test the Technical, Operational and Economical feasibility for adding new modules and debugging old running system. All system is feasible if they are unlimited resources and infinite time. There are aspects in the feasibility study portion of the preliminary investigation:

- Technical Feasibility
- Operational Feasibility
- Economical Feasibility

TECHNICAL FEASIBILITY

Technical Feasibility centers on the existing computer system hardware, software, etc. and to some extent how it can support the proposed addition. This involves financial considerations to accommodate technical enhancements. Technical support is also a reason for the success of the project. The techniques needed for the system should be available and it must be reasonable to use. Technical Feasibility is mainly concerned with the study of function, performance, and constraints that may affect the ability to achieve the system. By conducting an efficient technical feasibility we need to ensure that the project works to solve the existing problem area. Since the project is designed with Tkinter module in Python as Front end and SQLite as Back end, it is easy to install in all the systems wherever needed. It is more efficient, easy and user-friendly to understand by almost everyone. Huge amount of data can be handled efficiently using SQLite as back end. Hence this project has good technical feasibility.

OPERATIONAL FEASIBILITY

People, by their very nature, instantly change, and, as you know, computers contribute to change. An assessment should be made of how strong the response of user personnel to the development of a computerized system can be. The staff is used to computerized systems. Such systems are becoming more common day after day for evaluating software developers. Therefore, this system is functionally feasible. Since this system is technically, economically, and functionally feasible, this system is considered feasible.

ECONOMICAL FEASIBILITY

The role of interface design is to reconcile the differences that prevail among the software engineer's design model, the designed system meet the end user requirement with economical way at minimal cost within the affordable price by encouraging more of proposed system. Economic feasibility is concerned with comparing the development cost with the income/benefit derived from the developed system. In this we need to derive how this project will help the management to take effective decisions. Economic Feasibility is mainly concerned with the cost incurred in the implementation of the software. Since this project is developed using Tkinter model with Python and SQLite which is more commonly available and even the cost involved in the installation process is not high. Similarly it is easy to recruit persons for operating the software since almost all the people are aware of Tkinter modul in pythron and SQLite . Even if we want to train the persons in these area the cost involved in training is also very less. Hence this project has good economic feasibility. The system once developed must be used efficiently. Otherwise there is no meaning for developing the system. For this a careful study of the existing system and its drawbacks are needed. The user should be able to distinguish the existing one and proposed one, so that one must be able to appreciate the characteristics of the proposed system, the manual one is not highly reliable and also is considerably fast. The proposed system is efficient, reliable and also quickly responding.

S/w and H/w requirements

1.Enviroments:

- Servers:
- Operating System Server:- Microsoft Windows 2007r or Higher
- Data Base Server: Microsoft SQLite Studio
- Clients Microsoft Internet Explorer, Google Chrome etc
- Tools: Any python editor
- User interface: Tkinter model
- Code behind: Python

2.Requirements:

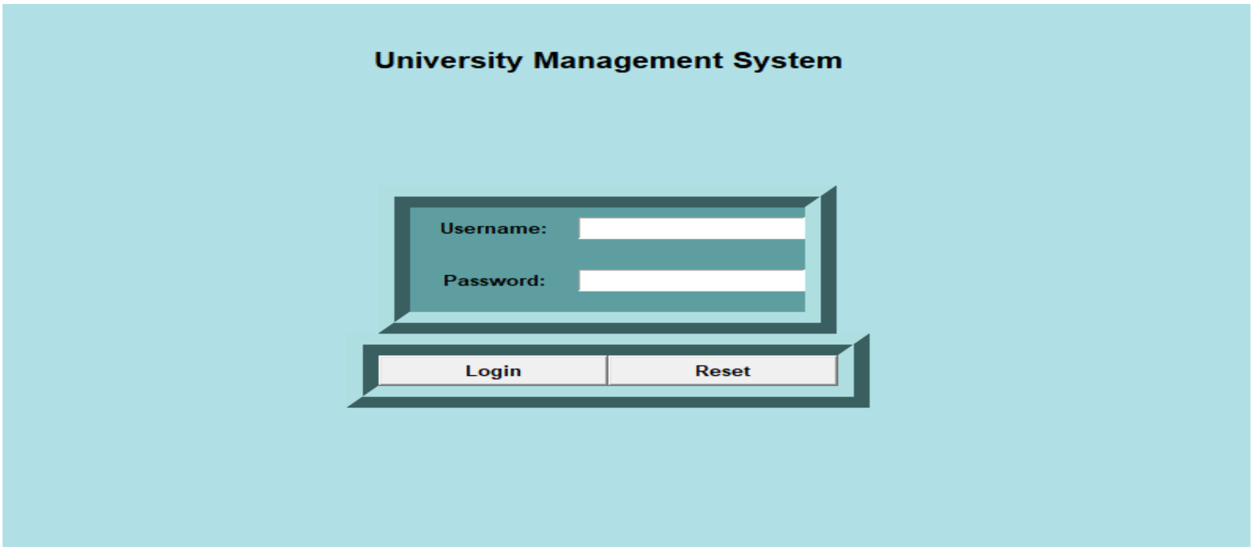
- Hardware requirements:
  Number of Description
  1 PC with 2 GB hard-disk and 256 MB RAM
- Software requirements:
  Number Description
  1. Windows 7 or Higher
  2. MS-SQLite
  3. Python(Pycharm compiler)
  4. Any internet Services

Coordinator

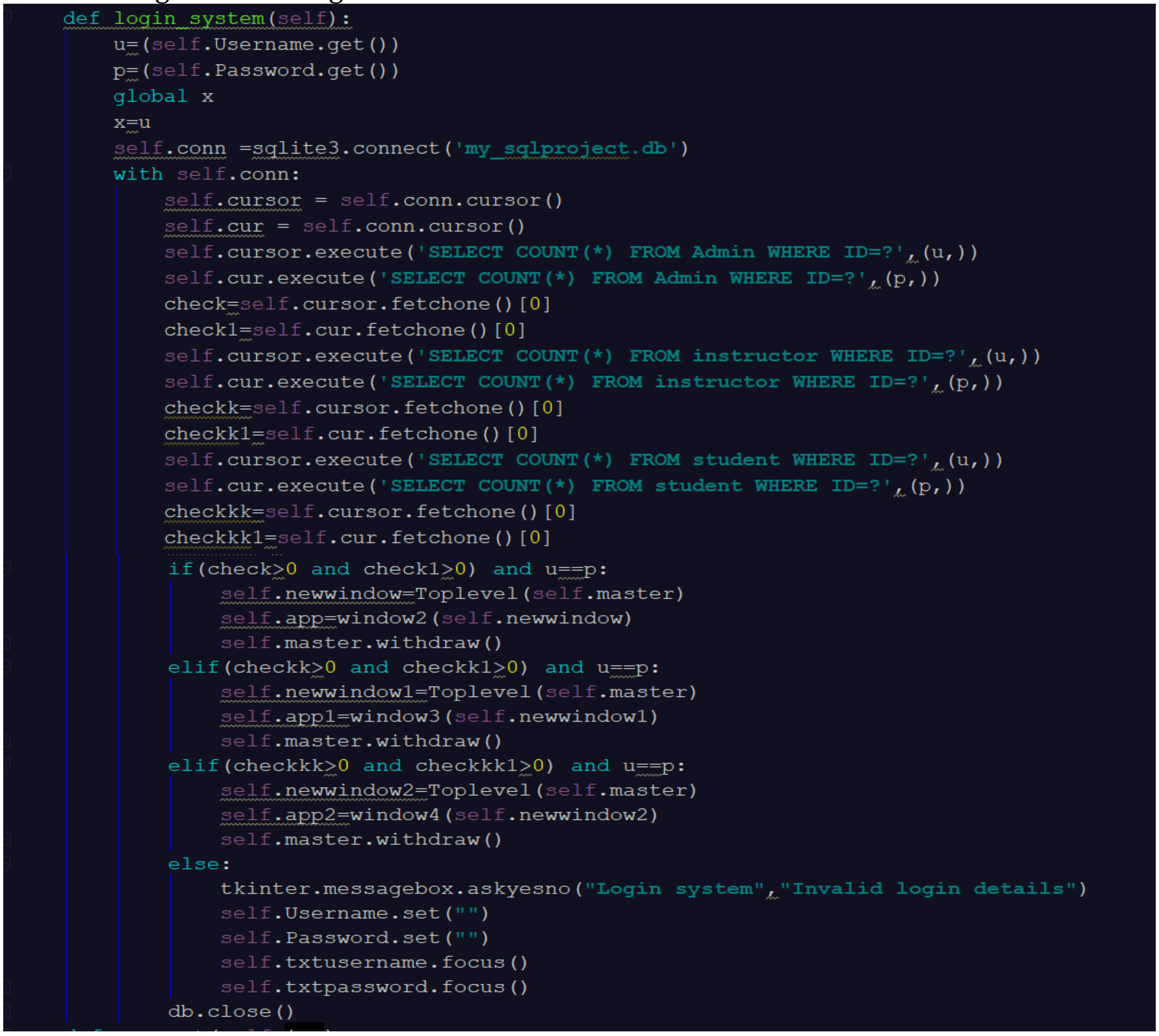
When the program runs , it will appear a window with frame includes a empty cells for Username and Password and button to login and reset .All departments have their own coordinators, which are admins. Firstly, in order to use the system, they must use their IDs as username and password. A new window will appear for adding new instructor or student with thier details. And also assign course to instructor and to student . When user press the button "Add" it will add them to Database of University Management.

Normal flow:

1. User enter login details.
   System verifies login details and displays coordinator page.
2. User enter course, instructor and student details and click to the add button.
   System will verifies details and add to the databse.
3. User will assign course to instructor. System verifies course and instructor details and add to database.
4. User will assign instructor to student. System verifies instructor and student details and add to database.



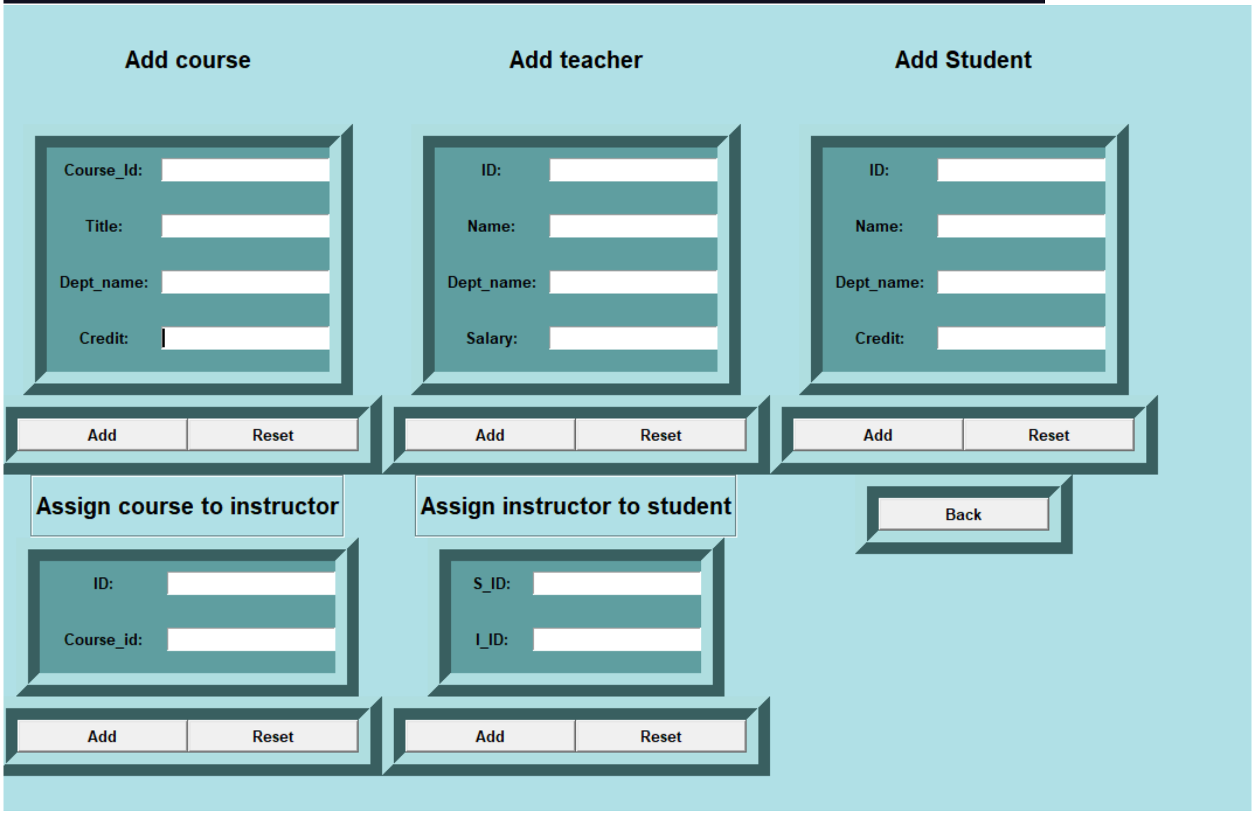
#code for login!!!!when login function clicks this function works



```
    def login_system(self):
        u=(self.Username.get())
        p=(self.Password.get())
        global x
        x=u
        self.conn =sqlite3.connect('my_sqlproject.db')
        with self.conn:
            self.cursor = self.conn.cursor()
            self.cur = self.conn.cursor()
            self.cursor.execute('SELECT COUNT(*) FROM Admin WHERE ID=?',(u,))
            self.cur.execute('SELECT COUNT(*) FROM Admin WHERE ID=?',(p,))
            check=self.cursor.fetchone()[0]
            check1=self.cur.fetchone()[0]
            self.cursor.execute('SELECT COUNT(*) FROM instructor WHERE ID=?',(u,))
            self.cur.execute('SELECT COUNT(*) FROM instructor WHERE ID=?',(p,))
            checkk=self.cursor.fetchone()[0]
            checkk1=self.cur.fetchone()[0]
            self.cursor.execute('SELECT COUNT(*) FROM student WHERE ID=?',(u,))
            self.cur.execute('SELECT COUNT(*) FROM student WHERE ID=?',(p,))
            checkkk=self.cursor.fetchone()[0]
            checkkk1=self.cur.fetchone()[0]
            if(check>0 and check1>0) and u==p:
                self.newwindow=Toplevel(self.master)
                self.app=window2(self.newwindow)
                self.master.withdraw()
            elif(checkk>0 and checkk1>0) and u==p:
                self.newwindow1=Toplevel(self.master)
                self.app1=window3(self.newwindow1)
                self.master.withdraw()
            elif(checkkk>0 and checkkk1>0) and u==p:
                self.newwindow2=Toplevel(self.master)
                self.app2=window4(self.newwindow2)
                self.master.withdraw()
            else:
                tkinter.messagebox.askyesno("Login system","Invalid login details")
                self.Username.set("")
                self.Password.set("")
                self.txtusername.focus()
                self.txtpassword.focus()
            db.close()
```

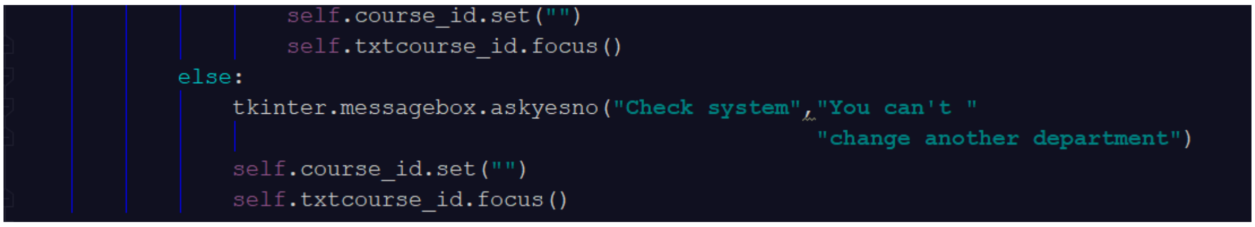
#When reset button click this function works

```
def reset(self):
    self.Username.set("")
    self.Password.set("")
    self.txtusername.focus()
    self.txtpassword.focus()
```



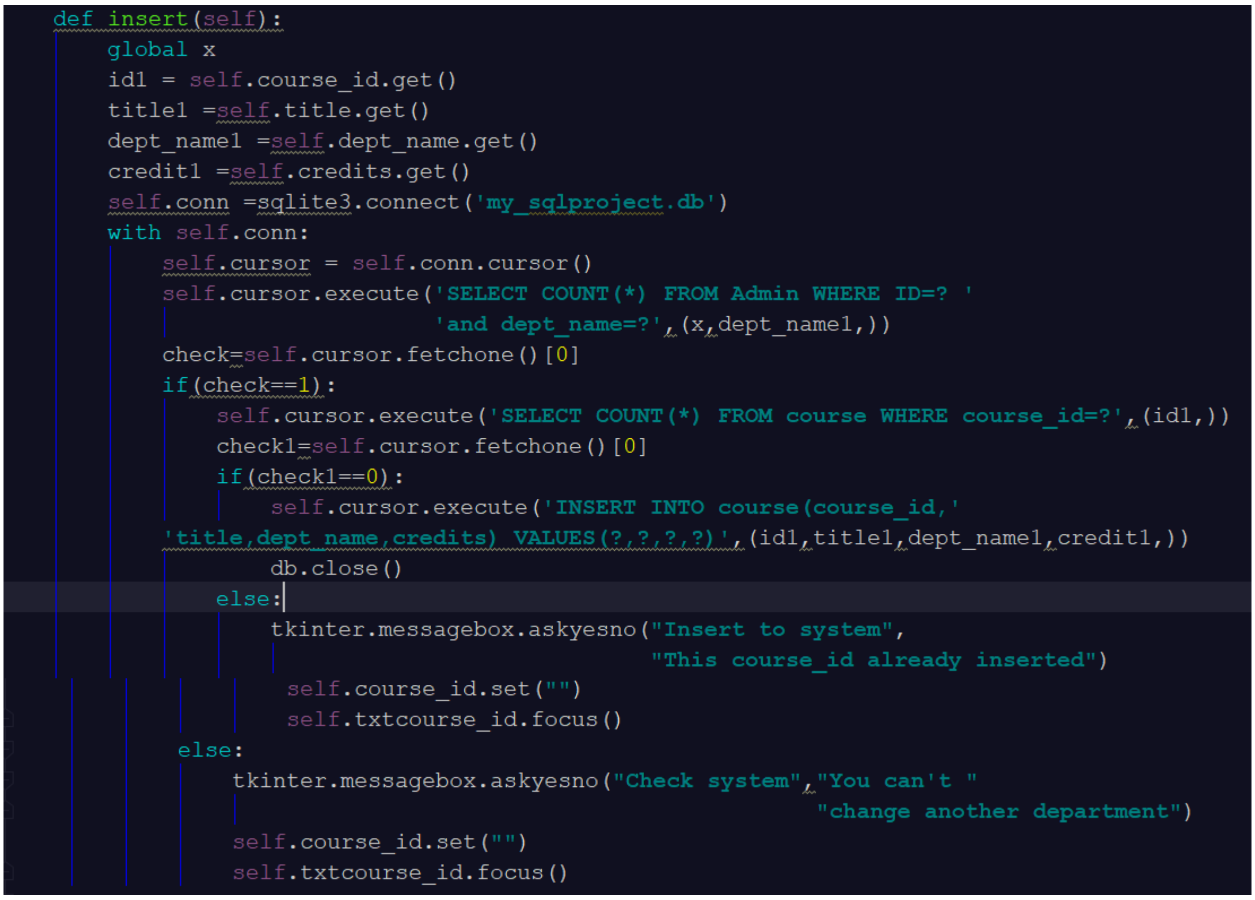
#ADD course. When add button clicks this function works

```
def insert(self):
    global x
    id1 = self.course_id.get()
    title1 =self.title.get()
    dept_name1 =self.dept_name.get()
    credit1 =self.credits.get()
    self.conn =sqlite3.connect('my_sqlproject.db')
    with self.conn:
        self.cursor = self.conn.cursor()
        self.cursor.execute('SELECT COUNT(*) FROM Admin WHERE ID=? '
                            'and dept_name=?',(x,dept_name1,))
        check=self.cursor.fetchone()[0]
        if(check==1):
            self.cursor.execute('SELECT COUNT(*) FROM course WHERE course_id=?',(id1,))
            check1=self.cursor.fetchone()[0]
            if(check1==0):
                self.cursor.execute('INSERT INTO course(course_id,'
        'title,dept_name,credits) VALUES(?,?,?,?)',(id1,title1,dept_name1,credit1,))
                db.close()
            else:
                tkinter.messagebox.askyesno("Insert to system",
                                            "This course_id already inserted")
```



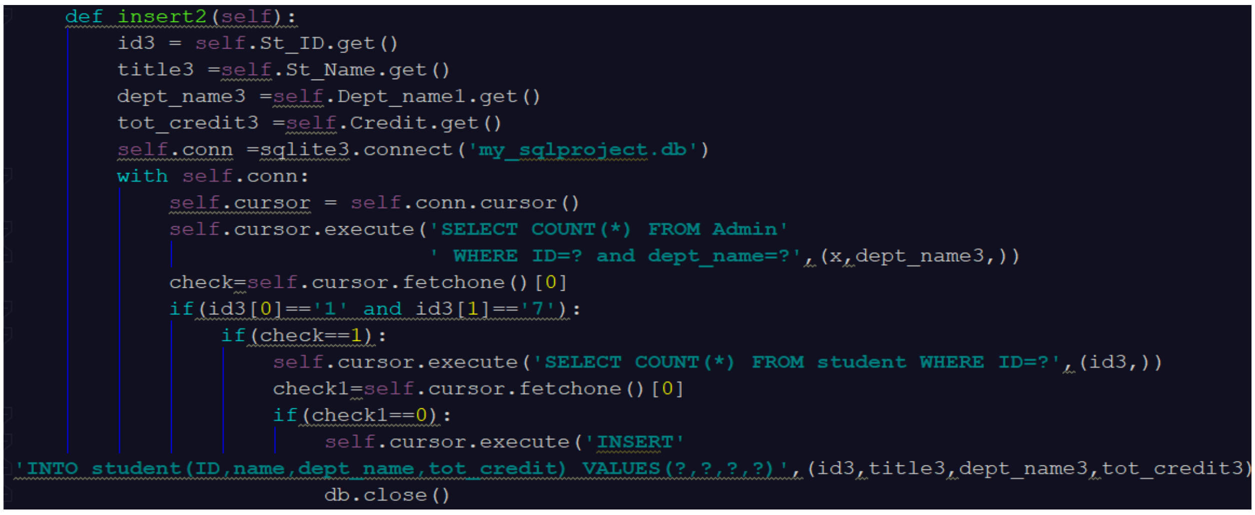
```
                self.course_id.set("")
                self.txtcourse_id.focus()
        else:
            tkinter.messagebox.askyesno("Check system","You can't "
                                         "change another department")
            self.course_id.set("")
            self.txtcourse_id.focus()
```

#ADD instructor. When add button clicks this function works

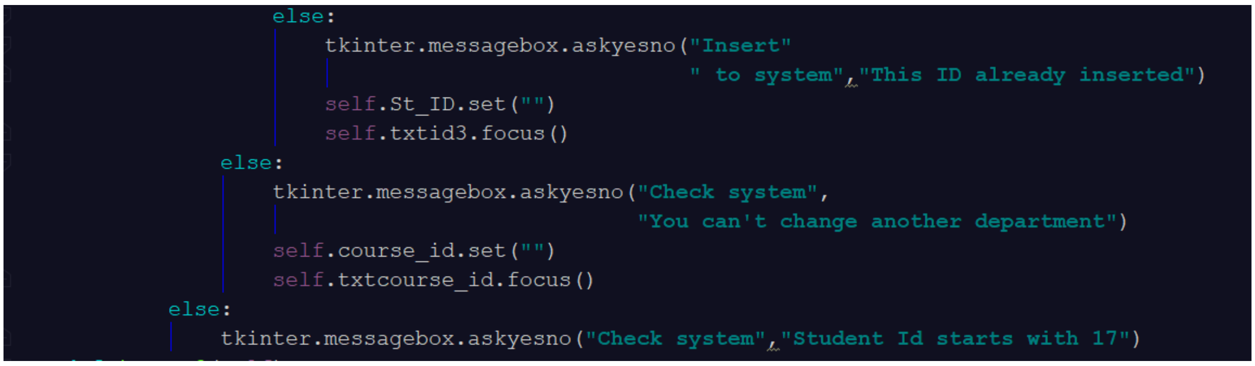


```
    def insert(self):
        global x
        id1 = self.course_id.get()
        title1 =self.title.get()
        dept_name1 =self.dept_name.get()
        credit1 =self.credits.get()
        self.conn =sqlite3.connect('my_sqlproject.db')
        with self.conn:
            self.cursor = self.conn.cursor()
            self.cursor.execute('SELECT COUNT(*) FROM Admin WHERE ID=? '
                                'and dept_name=?',(x,dept_name1,))
            check=self.cursor.fetchone()[0]
            if(check==1):
                self.cursor.execute('SELECT COUNT(*) FROM course WHERE course_id=?',(id1,))
                check1=self.cursor.fetchone()[0]
                if(check1==0):
                    self.cursor.execute('INSERT INTO course(course_id,'
            'title,dept_name,credits) VALUES(?,?,?,?)',(id1,title1,dept_name1,credit1,))
                    db.close()
                else:
                    tkinter.messagebox.askyesno("Insert to system",
                                                "This course_id already inserted")
                    self.course_id.set("")
                    self.txtcourse_id.focus()
            else:
                tkinter.messagebox.askyesno("Check system","You can't "
                                             "change another department")
                self.course_id.set("")
                self.txtcourse_id.focus()
```

#ADD student. When add button clicks this function works

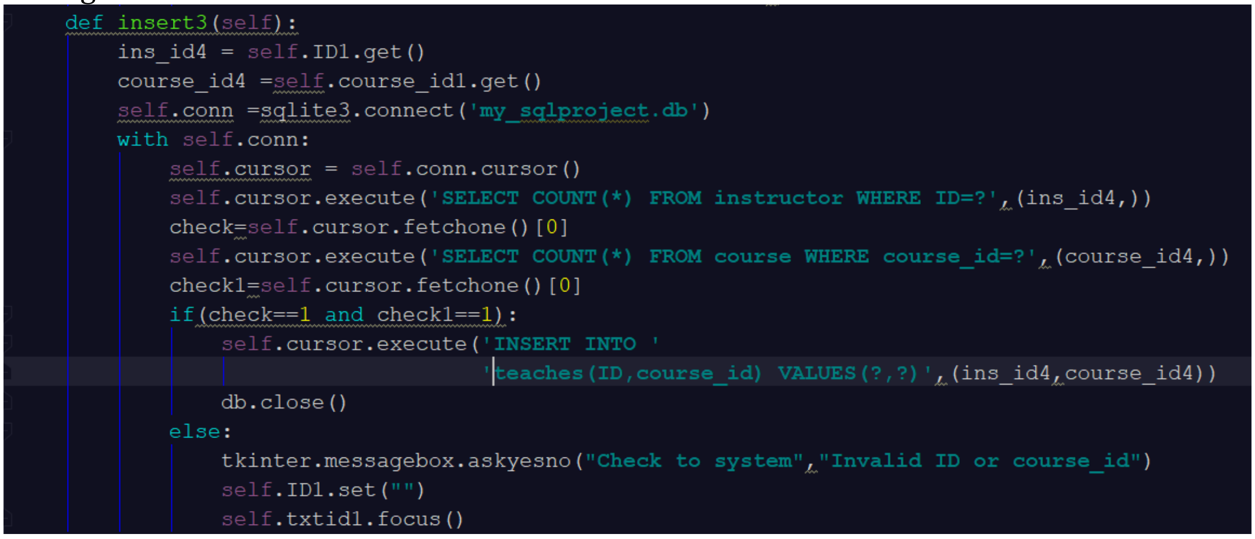


```
    def insert2(self):
        id3 = self.St_ID.get()
        title3 =self.St_Name.get()
        dept_name3 =self.Dept_name1.get()
        tot_credit3 =self.Credit.get()
        self.conn =sqlite3.connect('my_sqlproject.db')
        with self.conn:
            self.cursor = self.conn.cursor()
            self.cursor.execute('SELECT COUNT(*) FROM Admin'
                                ' WHERE ID=? and dept_name=?',(x,dept_name3,))
            check=self.cursor.fetchone()[0]
            if(id3[0]=='1' and id3[1]=='7'):
                if(check==1):
                    self.cursor.execute('SELECT COUNT(*) FROM student WHERE ID=?',(id3,))
                    check1=self.cursor.fetchone()[0]
                    if(check1==0):
                        self.cursor.execute('INSERT'
'INTO student(ID,name,dept_name,tot_credit) VALUES(?,?,?,?)',(id3,title3,dept_name3,tot_credit3)
                        db.close()
```



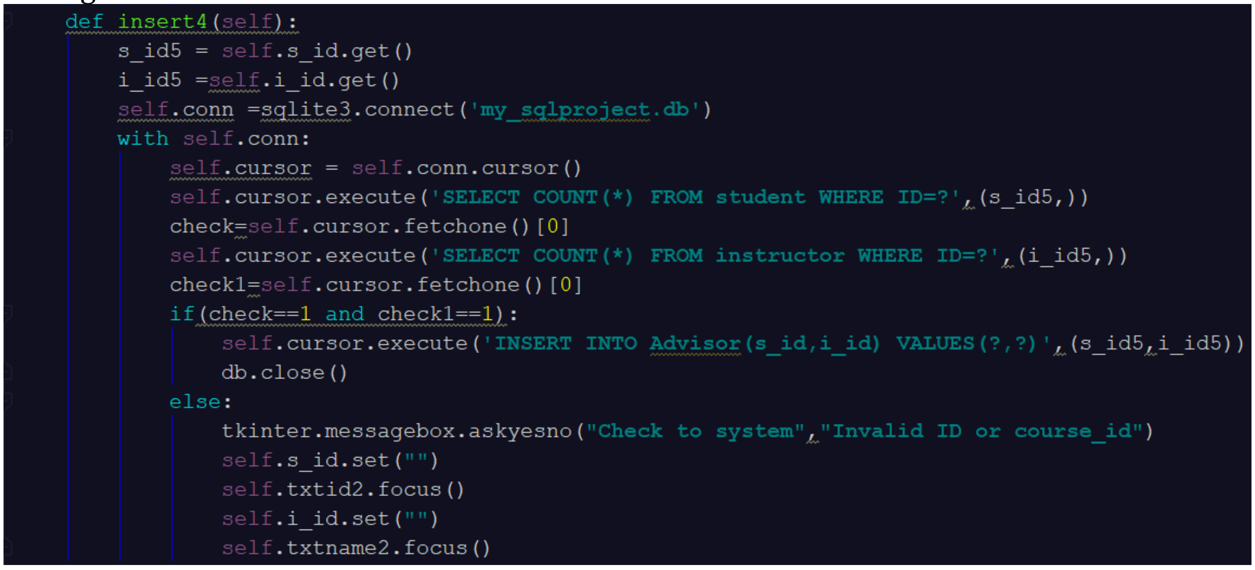
```
                    else:
                        tkinter.messagebox.askyesno("Insert"
                                                    " to system","This ID already inserted")
                        self.St_ID.set("")
                        self.txtid3.focus()
                else:
                    tkinter.messagebox.askyesno("Check system",
                                                "You can't change another department")
                    self.course_id.set("")
                    self.txtcourse_id.focus()
            else:
                tkinter.messagebox.askyesno("Check system","Student Id starts with 17")
```

#Assign course to instructor.When add button clicks this function works



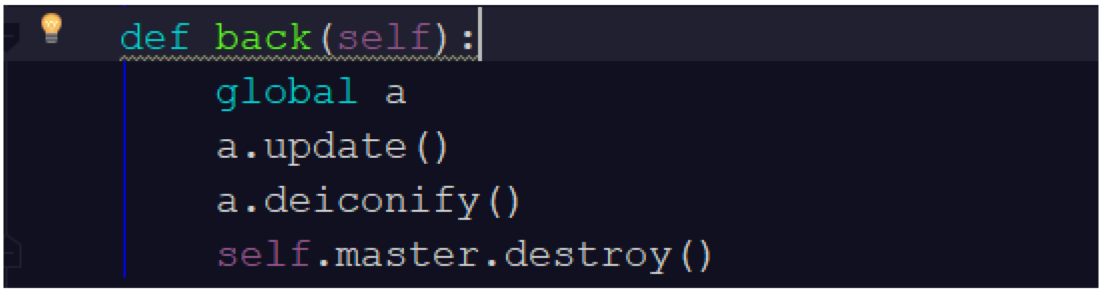
```
    def insert3(self):
        ins_id4 = self.ID1.get()
        course_id4 =self.course_id1.get()
        self.conn =sqlite3.connect('my_sqlproject.db')
        with self.conn:
            self.cursor = self.conn.cursor()
            self.cursor.execute('SELECT COUNT(*) FROM instructor WHERE ID=?',(ins_id4,))
            check=self.cursor.fetchone()[0]
            self.cursor.execute('SELECT COUNT(*) FROM course WHERE course_id=?',(course_id4,))
            check1=self.cursor.fetchone()[0]
            if(check==1 and check1==1):
                self.cursor.execute('INSERT INTO '
                                    'teaches(ID,course_id) VALUES(?,?)',(ins_id4,course_id4))
                db.close()
            else:
                tkinter.messagebox.askyesno("Check to system","Invalid ID or course_id")
                self.ID1.set("")
                self.txtid1.focus()
```

#Assign instructor to student. When add button clicks this function will work



```
    def insert4(self):
        s_id5 = self.s_id.get()
        i_id5 =self.i_id.get()
        self.conn =sqlite3.connect('my_sqlproject.db')
        with self.conn:
            self.cursor = self.conn.cursor()
            self.cursor.execute('SELECT COUNT(*) FROM student WHERE ID=?',(s_id5,))
            check=self.cursor.fetchone()[0]
            self.cursor.execute('SELECT COUNT(*) FROM instructor WHERE ID=?',(i_id5,))
            check1=self.cursor.fetchone()[0]
            if(check==1 and check1==1):
                self.cursor.execute('INSERT INTO Advisor(s_id,i_id) VALUES(?,?)',(s_id5,i_id5))
                db.close()
            else:
                tkinter.messagebox.askyesno("Check to system","Invalid ID or course_id")
                self.s_id.set("")
                self.txtid2.focus()
                self.i_id.set("")
                self.txtname2.focus()
```

#When BACK button clicks this function will work



```
    def back(self):
        global a
        a.update()
        a.deiconify()
        self.master.destroy()
```

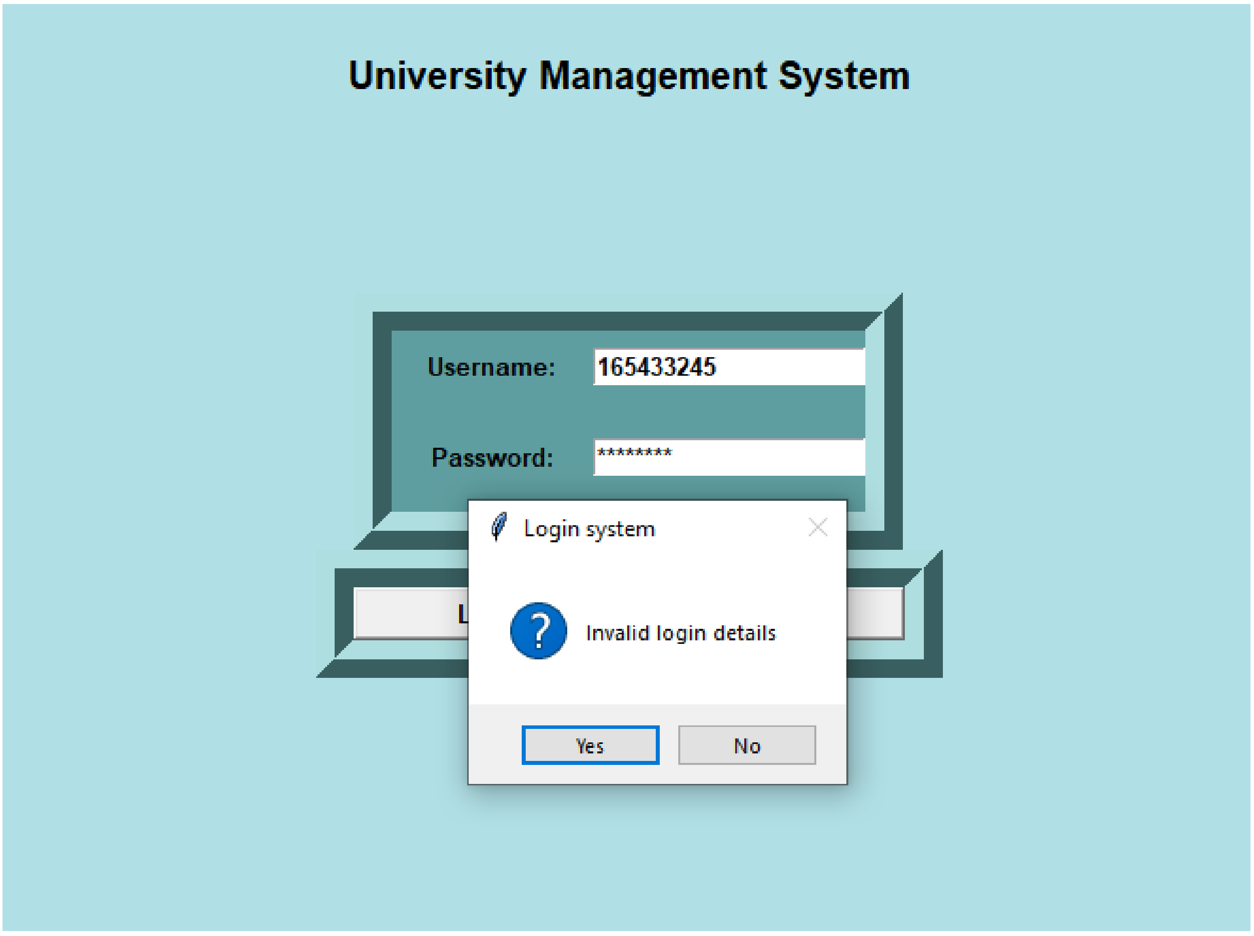
Alternative flow:

1. If login details incorrect, system displays «invalid login details»
2. If deparment name doesn't match with coordinator detail, system displays "you can't change another department".

2.1. If course_id , student_id, instructor_id already exists in database , system displays "this ID is already inserted".

3. If course_id or instructor_id is incorrect, system displays «invalid course_id or instructor_id».
4. If instructor_id or student_id is incorrect, system displays «invalid student_id or instructor_id».

1.


University Management System

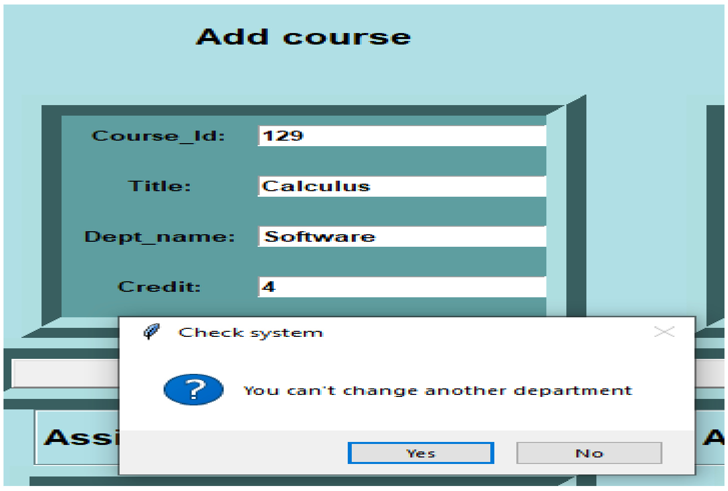
Username: 165433245

Password: ********

Login system

Invalid login details

2.


Add course

Course_Id: 129

Title: Calculus

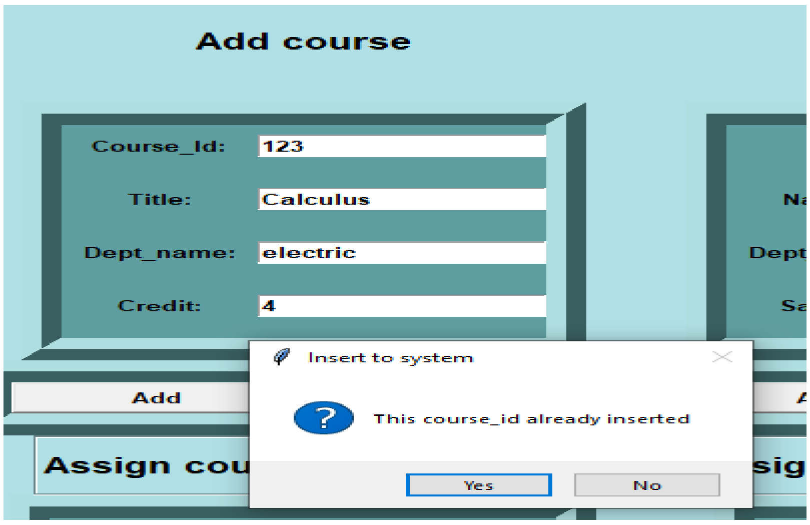
Dept_name: Software

Credit: 4

Check system

You can't change another department

2.1


Add course

Course_Id: 123

Title: Calculus

Dept_name: electric

Credit: 4

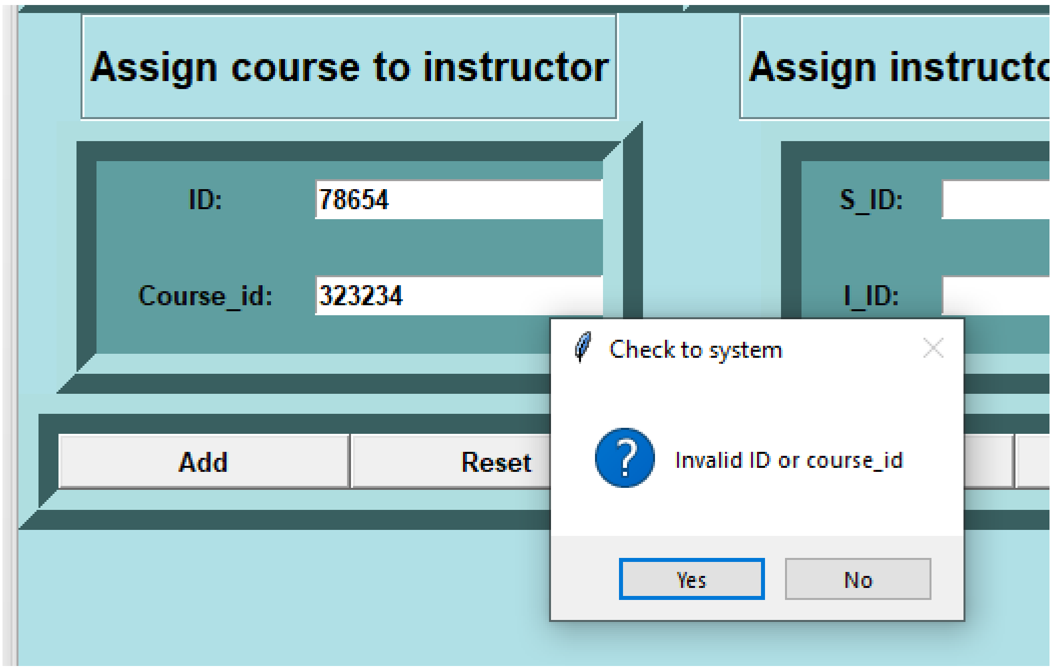
Insert to system

This course_id already inserted

Yes No

#for add instructor and add student same messagebox which is in add course

3.


Assign course to instructor

ID: 78654

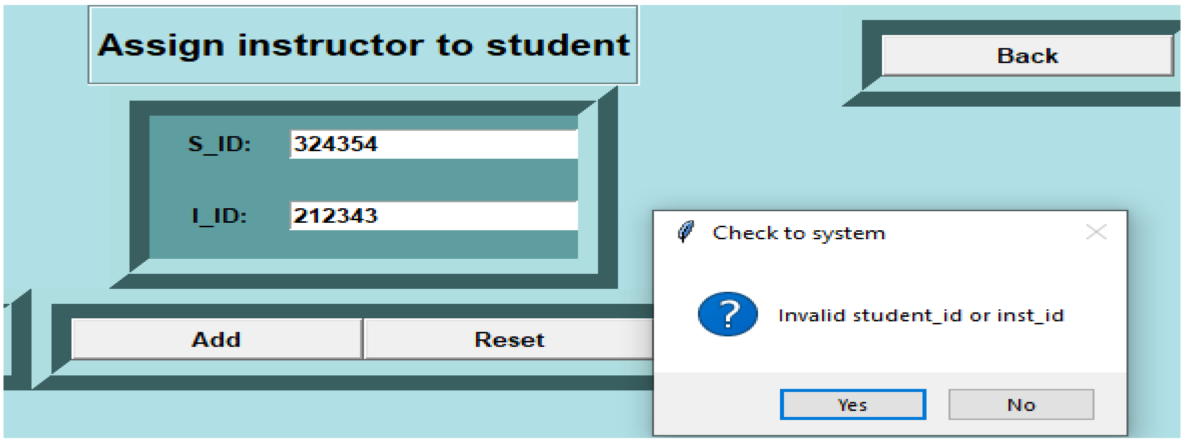
Course_id: 323234

Check to system

Invalid ID or course_id

Yes No

4.


Assign instructor to student

S_ID: 324354

I_ID: 212343

Check to system

Invalid student_id or inst_id

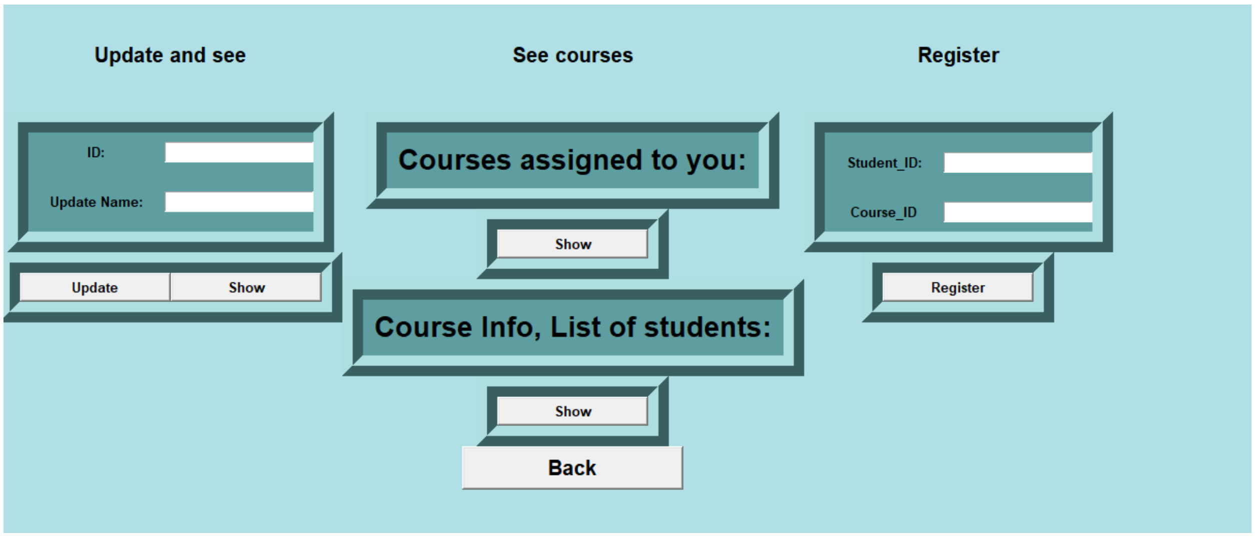
Yes No

Instructor

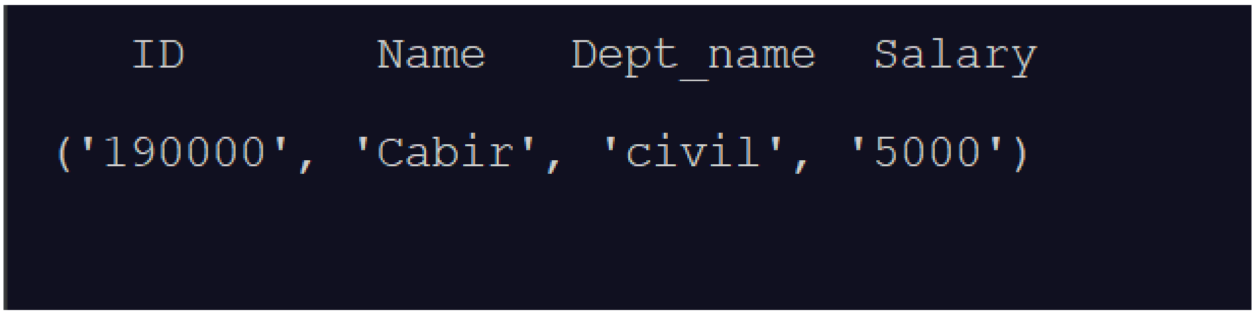
When a program runs instructor also will do the same things as coordinator. After logining to the system there will be appear a window with functions such as update his/her details, see courses assigned to him/her, course information and list fo students, and register a new student to course. By pressing button "Update" he/she will be able to update information, the button "Show" will show according to requirement, same with "Register" button

Normal flow:

1. User enter login details. System verifies and display instructor page.
2. User enter his/her ID to update details. System verifies ID and update details.
3. User can see his/her details by clicking «show» button. System displays user information.
4. User can see courses assigned to him/her by clicking «show » button. System displays courses assigned to instructor.
5. User can see course details. System displays course information, list of students registered , etc.
6. User enter student_id and course_id to register . System verifies information and add to the database.

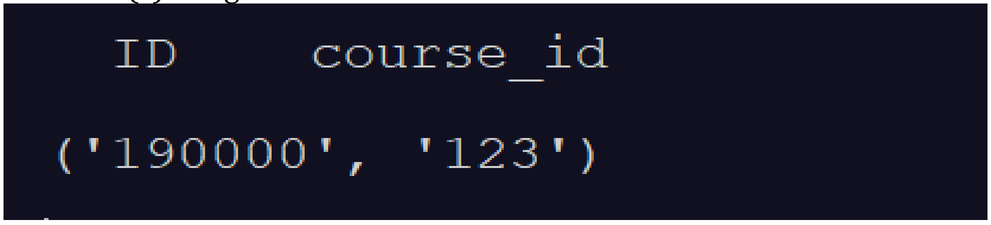


3.



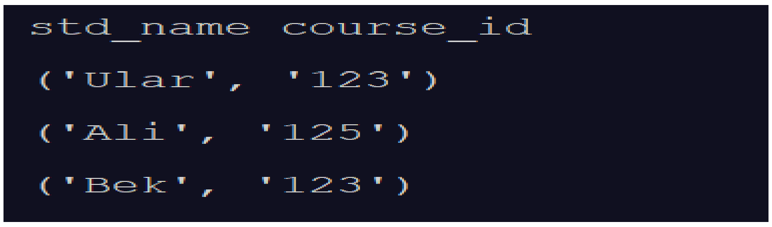
```
    ID          Name    Dept_name   Salary

 ('190000',  'Cabir',  'civil',  '5000')
```

4. course(s) assigned to instructor



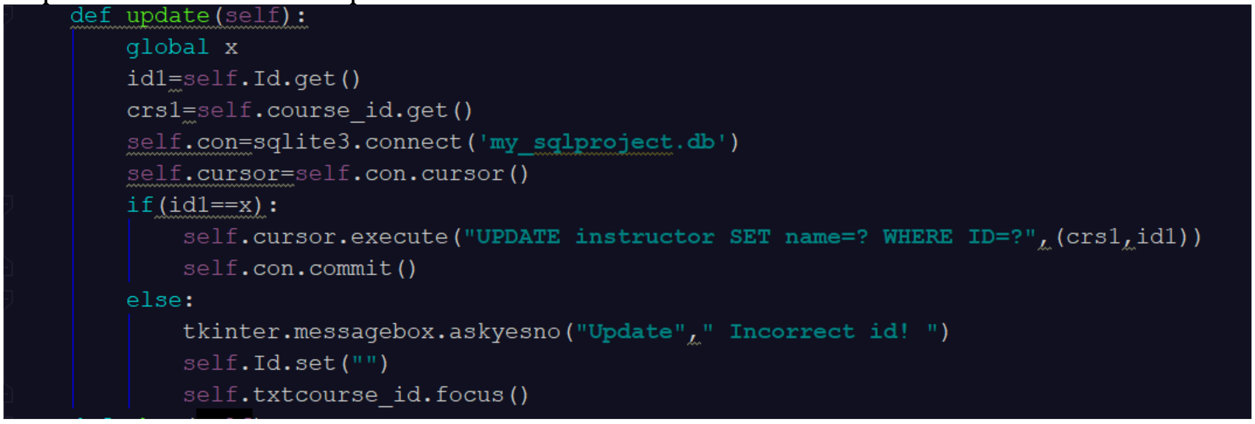
```
   ID      course_id

 ('190000',  '123')
```

5.



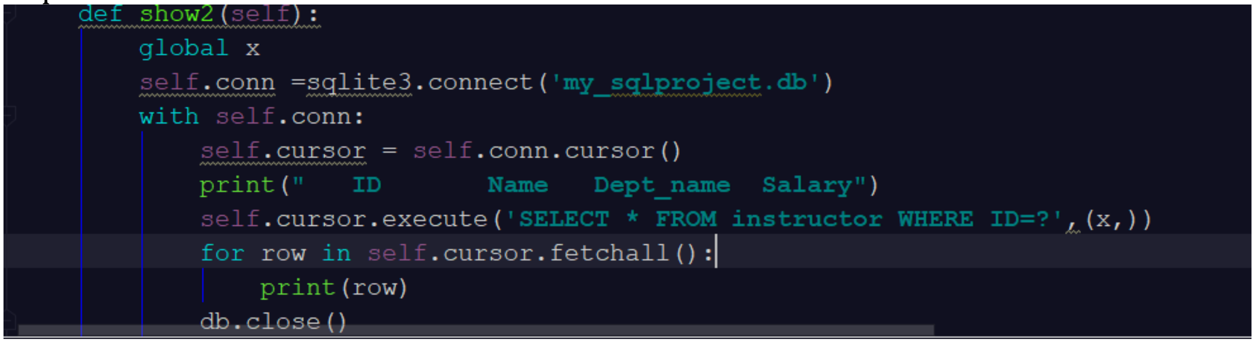
```
std_name  course_id
('Ular',  '123')
('Ali',  '125')
('Bek',  '123')
```

#update and see. When update button clicks this function will work



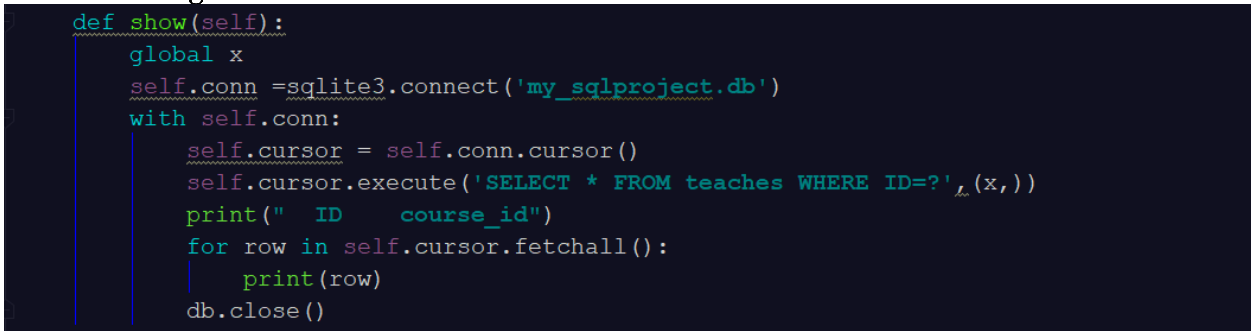
```
    def update(self):
        global x
        id1=self.Id.get()
        crs1=self.course_id.get()
        self.con=sqlite3.connect('my_sqlproject.db')
        self.cursor=self.con.cursor()
        if(id1==x):
            self.cursor.execute("UPDATE instructor SET name=? WHERE ID=?",(crs1,id1))
            self.con.commit()
        else:
            tkinter.messagebox.askyesno("Update"," Incorrect id! ")
            self.Id.set("")
            self.txtcourse_id.focus()
```

#update and see. When show button clicks this function will work



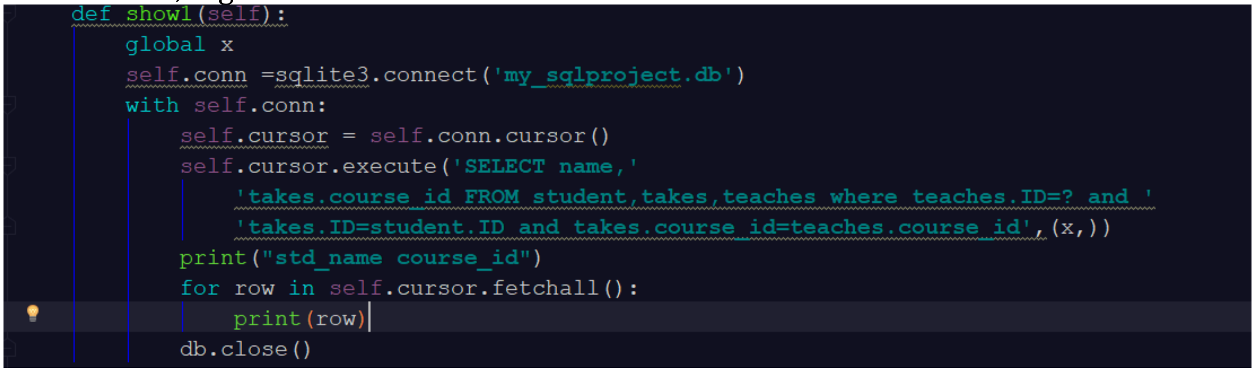
```
    def show2(self):
        global x
        self.conn =sqlite3.connect('my_sqlproject.db')
        with self.conn:
            self.cursor = self.conn.cursor()
            print("   ID        Name    Dept_name   Salary")
            self.cursor.execute('SELECT * FROM instructor WHERE ID=?',(x,))
            for row in self.cursor.fetchall():
                print(row)
            db.close()
```

#courses assigned to instructor. When show button clicks this function will work



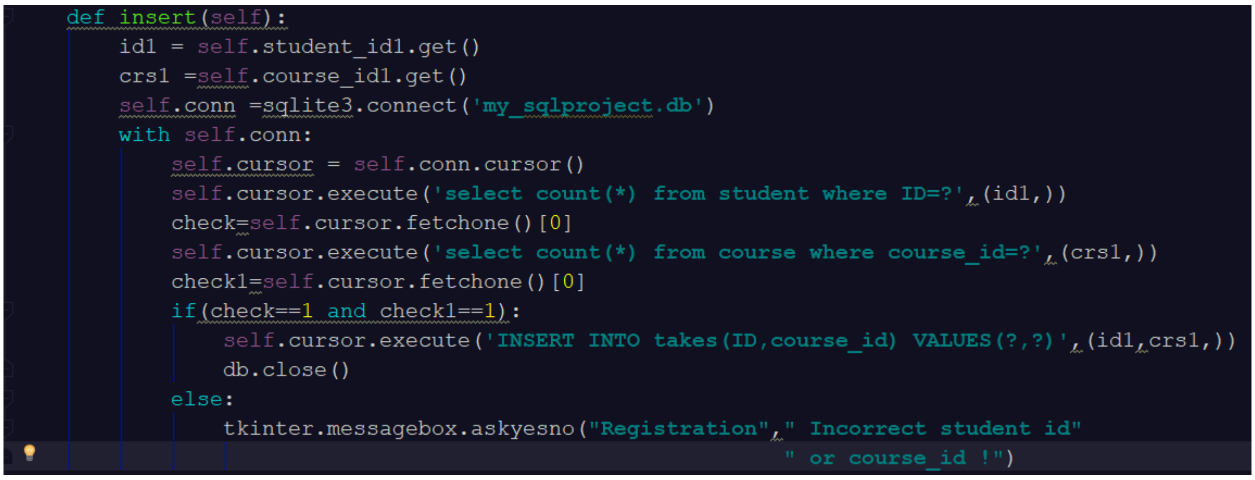
```
    def show(self):
        global x
        self.conn =sqlite3.connect('my_sqlproject.db')
        with self.conn:
            self.cursor = self.conn.cursor()
            self.cursor.execute('SELECT * FROM teaches WHERE ID=?',(x,))
            print("  ID     course_id")
            for row in self.cursor.fetchall():
                print(row)
            db.close()
```

#course info, registered students. When show button clicks this function will work



```
    def show1(self):
        global x
        self.conn =sqlite3.connect('my_sqlproject.db')
        with self.conn:
            self.cursor = self.conn.cursor()
            self.cursor.execute('SELECT name,'
                'takes.course_id FROM student,takes,teaches where teaches.ID=? and '
                'takes.ID=student.ID and takes.course_id=teaches.course_id',(x,))
            print("std_name course_id")
            for row in self.cursor.fetchall():
                print(row)
            db.close()
```

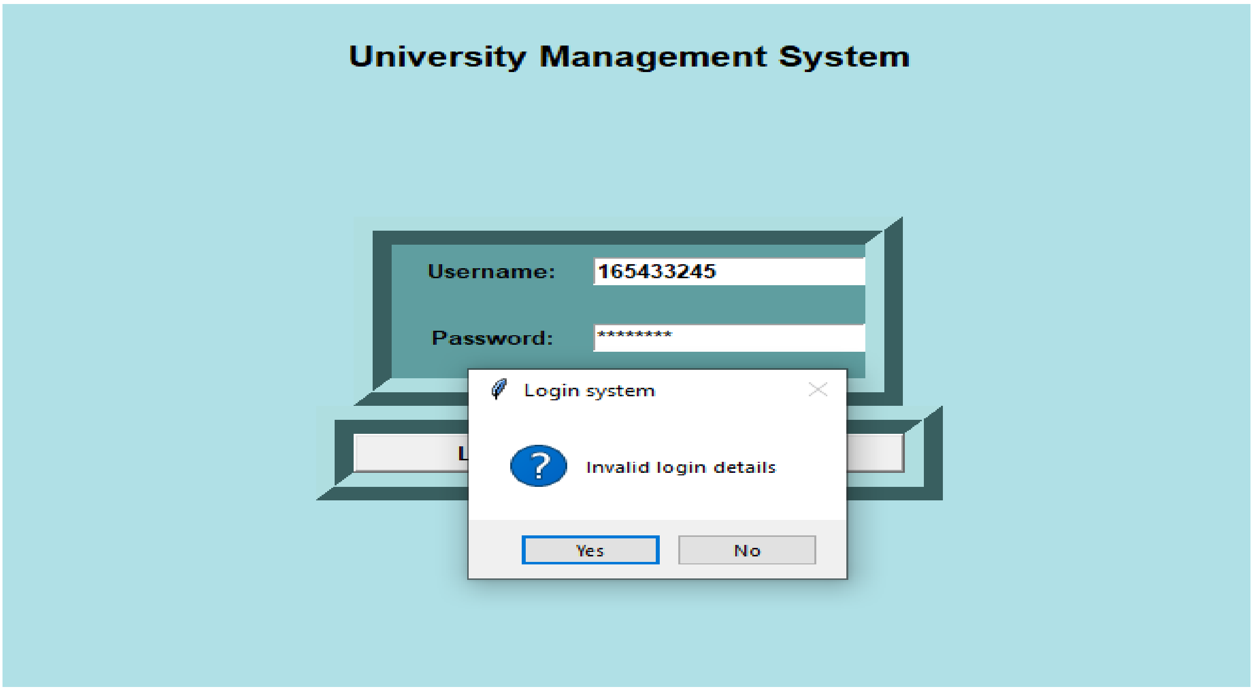
#Register. When show button clicks this function will work



```
def insert(self):
    id1 = self.student_id1.get()
    crs1 =self.course_id1.get()
    self.conn =sqlite3.connect('my_sqlproject.db')
    with self.conn:
        self.cursor = self.conn.cursor()
        self.cursor.execute('select count(*) from student where ID=?',(id1,))
        check=self.cursor.fetchone()[0]
        self.cursor.execute('select count(*) from course where course_id=?',(crs1,))
        check1=self.cursor.fetchone()[0]
        if(check==1 and check1==1):
            self.cursor.execute('INSERT INTO takes(ID,course_id) VALUES(?,?)',(id1,crs1,))
            db.close()
        else:
            tkinter.messagebox.askyesno("Registration"," Incorrect student id"
                                                      " or course_id !")
```

Alternative flow:

1. If login details incorrect, system displays «invalid login details»
2. If Id entered by user incorrect, system displays «incorrect id».
3. If student_id or course_id is incorrect, system reply message «invalid student_id or course_id».
4. If student_id or instructor_id is nor same
5. If instructor id is not starting with 19, system displays error message «Instructor id starts with 19»



University Management System

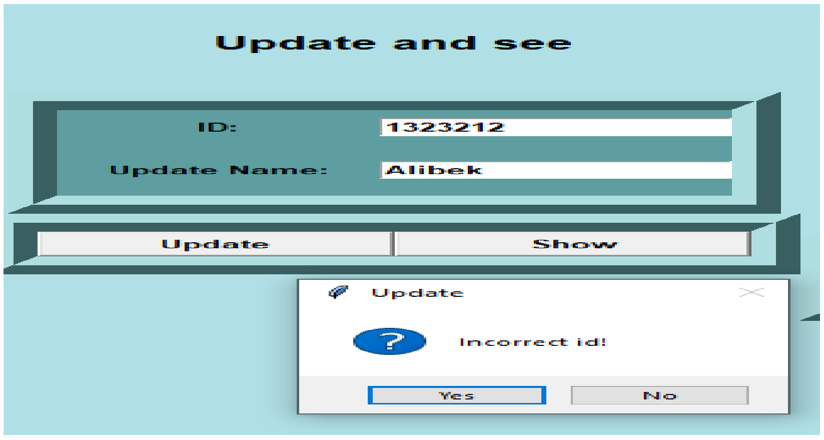
Username: 165433245

Password: *********

Login system

Invalid login details

2.



Update and see

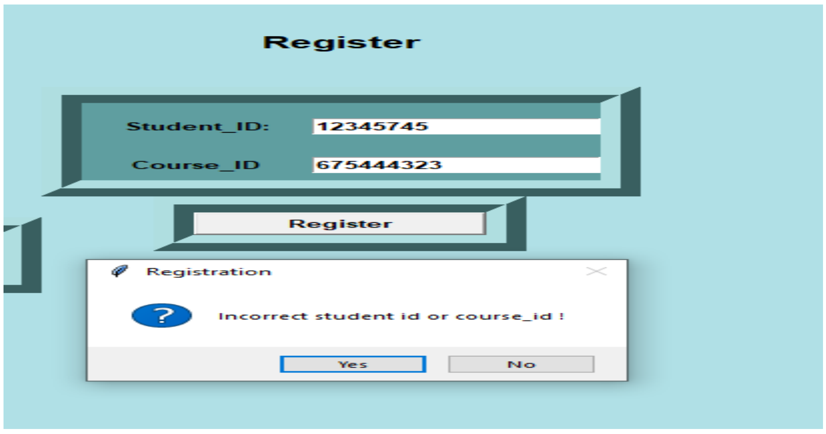
ID: 1323212

Update Name: Alibek

Update

Incorrect id!

6.



```
Student_ID: 12345745
Course_ID 675444323
```

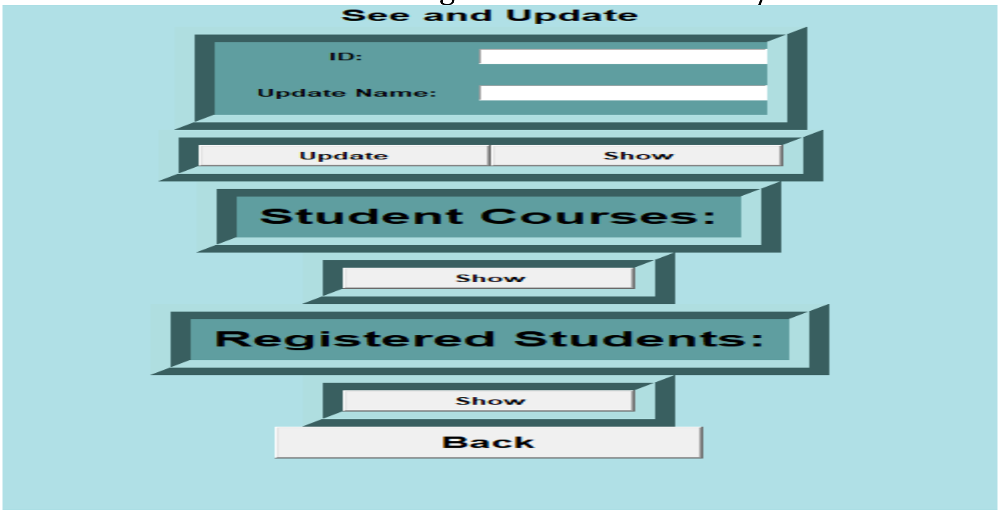
Incorrect student id or course_id !

Student

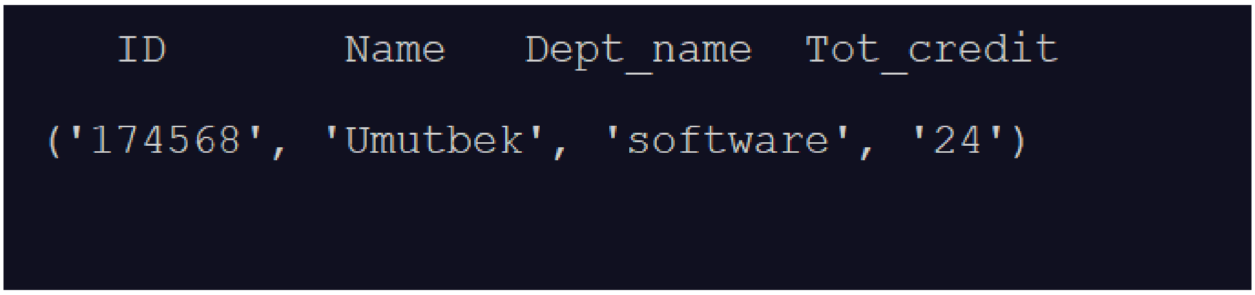
University management system allows to the students to use it by typing his/her own username and password. After logining to the system appears a new window with some function which allows students to see and update his/ her personal information. See course registered to him/her and the list of students in that course.

Normal flow:

1. User enter login details. System verifies and display student page.
2. User enter his/her ID to update details. System verifies ID and update details.
3. User can see his/her details by clicking «show» button. System displays user information.
4. User can see the courses registered to him/her by clicking «show» button. System displays course information.
5. User can see the list of registered students in his/her class.

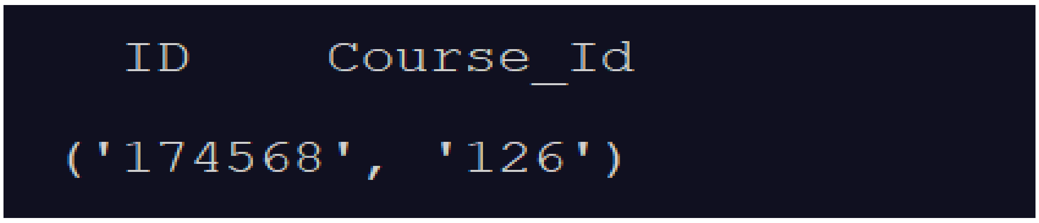


3.



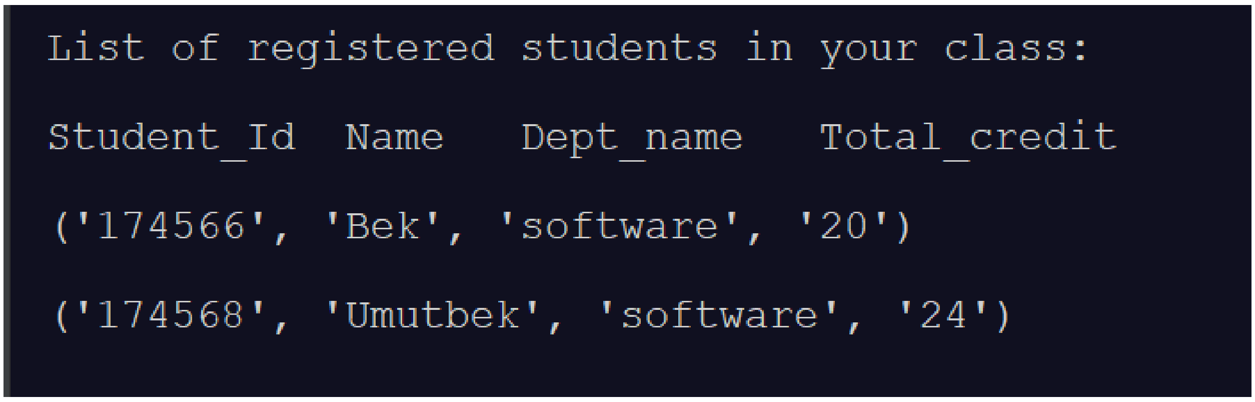
```
    ID         Name     Dept_name   Tot_credit

('174568', 'Umutbek', 'software', '24')
```

4.



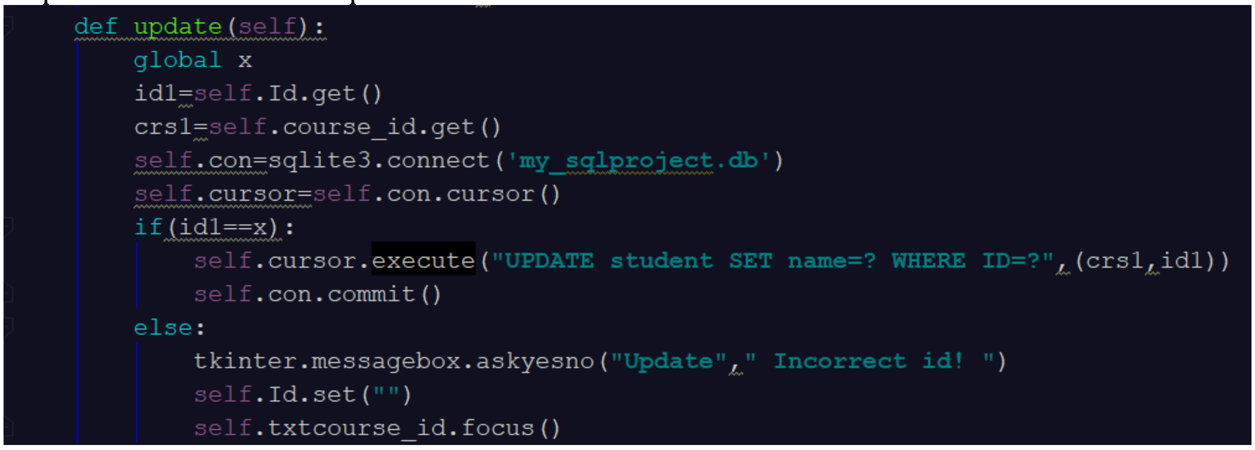
```
   ID      Course_Id
 ('174568', '126')
```

5.



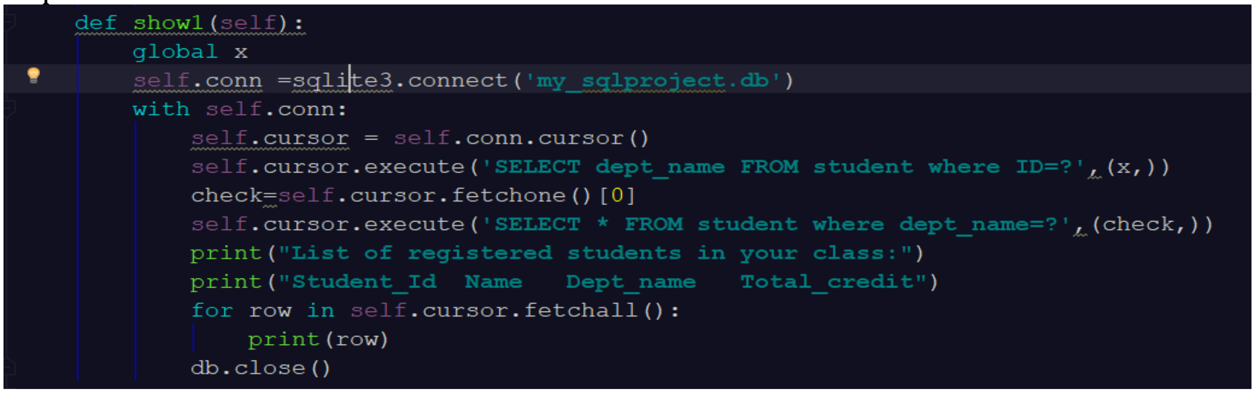
```
List of registered students in your class:
Student_Id  Name   Dept_name   Total_credit
('174566', 'Bek', 'software', '20')
('174568', 'Umutbek', 'software', '24')
```

#update and see. When update button clicks this function will work



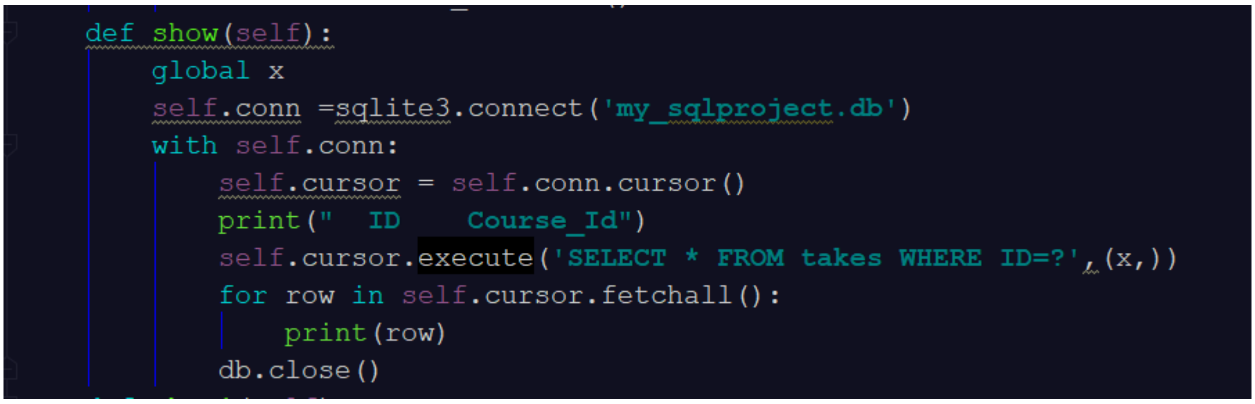
```
    def update(self):
        global x
        id1=self.Id.get()
        crs1=self.course_id.get()
        self.con=sqlite3.connect('my_sqlproject.db')
        self.cursor=self.con.cursor()
        if(id1==x):
            self.cursor.execute("UPDATE student SET name=? WHERE ID=?",(crs1,id1))
            self.con.commit()
        else:
            tkinter.messagebox.askyesno("Update"," Incorrect id! ")
            self.Id.set("")
            self.txtcourse_id.focus()
```

#update and see. When show button clicks this function will work



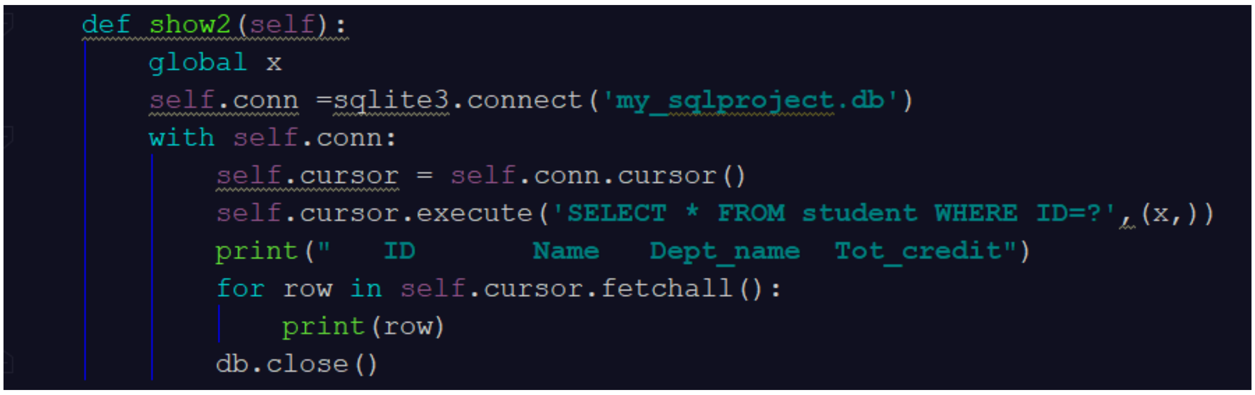
```
    def show1(self):
        global x
        self.conn =sqlite3.connect('my_sqlproject.db')
        with self.conn:
            self.cursor = self.conn.cursor()
            self.cursor.execute('SELECT dept_name FROM student where ID=?',(x,))
            check=self.cursor.fetchone()[0]
            self.cursor.execute('SELECT * FROM student where dept_name=?',(check,))
            print("List of registered students in your class:")
            print("Student_Id  Name   Dept_name   Total_credit")
            for row in self.cursor.fetchall():
                print(row)
            db.close()
```

#student courses. When show button clicks this function will work



```
    def show(self):
        global x
        self.conn =sqlite3.connect('my_sqlproject.db')
        with self.conn:
            self.cursor = self.conn.cursor()
            print("  ID     Course_Id")
            self.cursor.execute('SELECT * FROM takes WHERE ID=?',(x,))
            for row in self.cursor.fetchall():
                print(row)
            db.close()
```

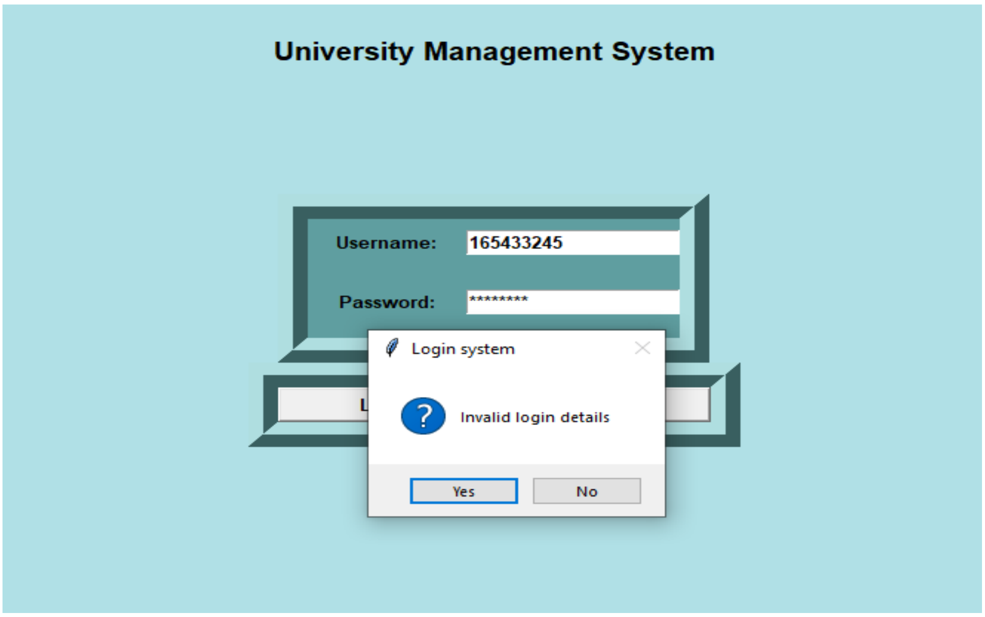
#list of students. When show button clicks this function will work



```
    def show2(self):
        global x
        self.conn =sqlite3.connect('my_sqlproject.db')
        with self.conn:
            self.cursor = self.conn.cursor()
            self.cursor.execute('SELECT * FROM student WHERE ID=?',(x,))
            print("   ID        Name     Dept_name   Tot_credit")
            for row in self.cursor.fetchall():
                print(row)
            db.close()
```

Alternative flow:

1. If entered login details incorrect , system will display error message «Invalid login details».
2. If user entered ID is incorrect, system displays error message «Incorrect id».
3. If student id is not starting with 17, system displays error message «Student id starts with 17»



University Management System

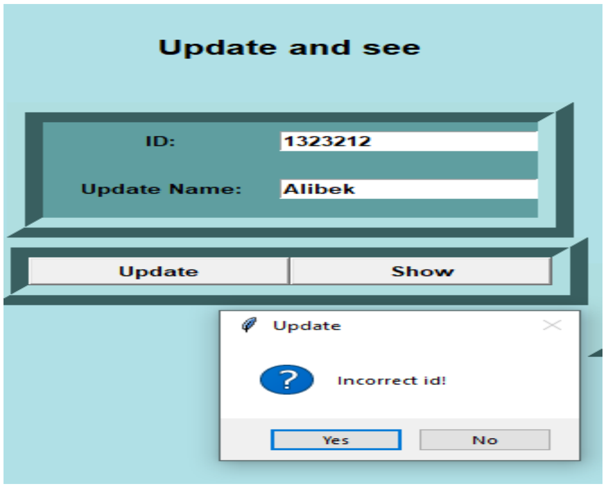
Username: 165433245

Password: ********

Login system

Invalid login details

Yes No



Update and see

ID: 1323212

Update Name: Alibek

Update Show

Update

Incorrect id!

Yes No