Project 4 – Final Report

Team Members:

1. Saaket Raman

2. Nischal Olety Nagesh

3. Guillermo Medina Nieves

Note To The User:

For the successful execution of this program, the user must be using Python 3.11.3 and beyond. Thus, please ensure that the version of python used is up to date. The python version can be checked by running the following command line argument:

```
python --version
```

Tables in the Database:

1. Professor: PID, Name, Department_ID, Salary, Email, Tenure

2. Students: UID, First_Name, Last_Name, Date_of_Birth, GPA, Major_ID, Email, Undergraduate, Class Standing, Password, Phone, Address

3. Courses: Course_ID, Course_Name, CourseDesc, Credits

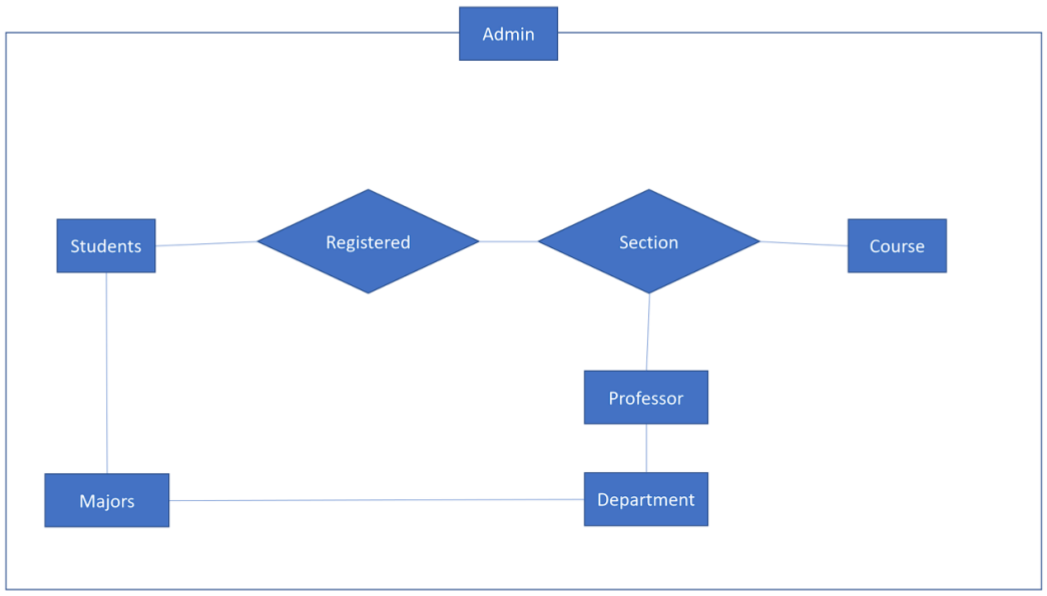
4. Department: Department_ID, Department_Name, Year_Founded, Budget, Department_Chair

5. Majors: Major_ID, Major_Name, Department_ID

6. Section: Section_ID, Course_ID, Term, Professor_ID, Section_Number

7. Registered: UID, Section_ID

8. Admin: Name, Email, Password



Welcome Page:

Upon running the program, the user will be prompted to choose between one of these three options:

1. Login using student credentials
2. Login using administrative credentials
3. Creating a new account as a student

On choosing any of the above three options

When the user chooses the option to sign up as a new user, the user will be prompted to enter a new username and a password. The program will check to ensure that the username entered by the user does not already exist and that the password is strong enough. After successfully creating new user credentials, the user will once again be prompted to login.

Attached below is a prototype of the welcome page on the terminal.

Administrator Functionalities:

The role of the administrator is to manage the upkeep of our database. Here, the administrator will have access to all the existing tables and will, thus, have authorization to edit each one as well. Below is a high-level description of administrative functionalities, which the administrator will have the option to choose from their home page:

1. Accessing a table
2. Modifying a table

3. Custom SQL Querying
4. Exit

1. Accessing a table:

On choosing this option, the administrator will be presented with the different tables that exists on the database. Thus, the administrator will be prompted to choose which table to access. Whatever table the administrator chooses, all rows of the table will be presented to the user as an output.

2. Modifying a Table:

Here, the administrator will be able to add a new row to the table, delete a row from a table, and even edit details of any row of the administrator's choice. f the ID entered by the administrator cannot be found on the table, the administrator is prompted to enter an ID again.

3. Custom SQL Querying

Our team accepts the constraint of hard coding the infinite possibilities of queries that an administrator would like to query on our database. Thus, we have implemented this option where the administrator is prompted to type an SQL of their choice. If the query is valid, the function returns the results of the query on the command line.

Within each of these options, the administrator will have the option to step out of the chosen option and choose another. Alas, the administrator will also have the option to log out.

<u>**Student Functionalities:**</u>

All students will be able to login using their University IDs (UIDs) and password. All UIDs are unique, and students will only be able to access and modify their own details, not other students. After a successfully logging into our database, the student will be able presented with one of the following options:

1. View personal information
2. Edit personal information
3. Access registered classes
4. Modifying the list of registered classes (add/remove)
5. Obtain a list of all professors within the department of the students major
6. Exit (log out)

1. View Personal Information:

On choosing this, the user will be presented with all their personal information. This includes the student's First Name, Last Name, Date of Birth, GPA, Major ID, Undergraduate status, Class Standing, Email ID, Phone Number, and Address.

Our database returns this in the form of a list.

2. Edit Personal Information:

Here, the user will be able to choose which one of their personal information they would like to change. Whatever the changes the user proposes, it will be reflected directly on the database.

3. Access Registered Classes:

The student will be able to obtain a list of classes and sections they are registered in.

4. Modifying The List Of Registered Classes:

Choosing this option, the user will be able to either add or remove a class from their list of registered classes. Either choices (add or remove) will ask the user for a Section ID, using which the class will either be added to or removed from their list of registered classes. The changes are reflected throughout the database.

5. Obtain A List Of All Professors Within The Department Of The Students Major:

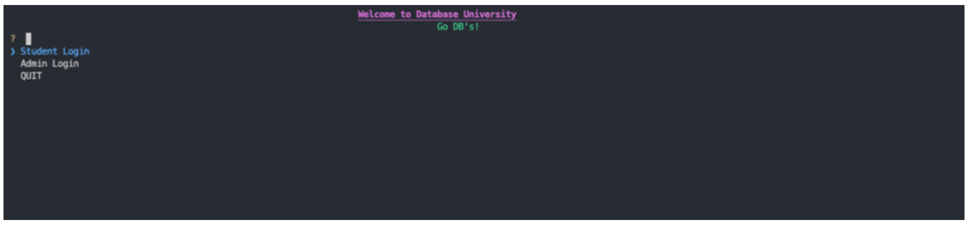
On choosing this option, the student will be able to view all professor who teach courses specific to the student's major. Below is an example output attached.

Within each of these options, the student will have the option to step out of the chosen option and choose another. Alas, the student will also have the option to log out.

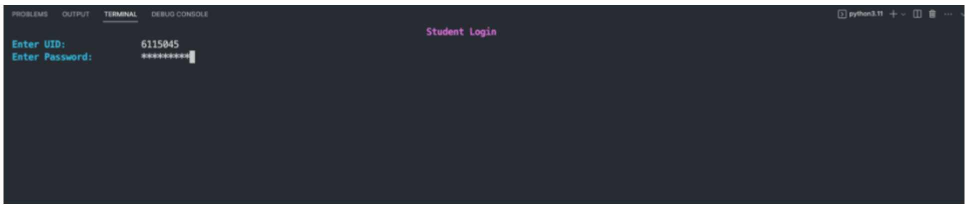
Functionalities:

Our team will be using Python's sqlite3 library to build the database and establish a connection to the database within the required functions. Attached below are the different implementations of our code and the output on the command line.

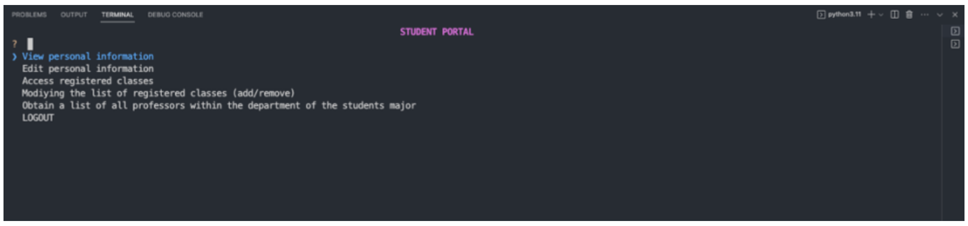
Some command line outputs:



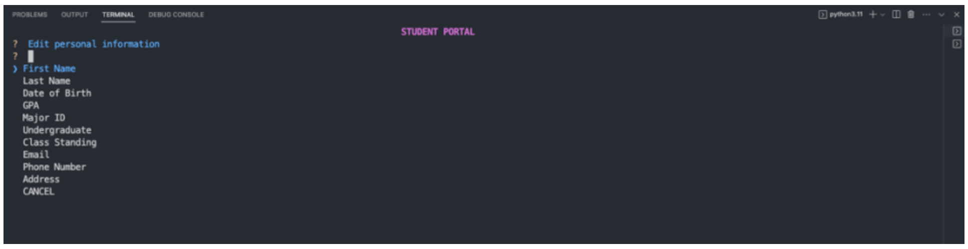
```
                              Welcome to Database University
                                         Go DB's!
?
❯ Student Login
  Admin Login
  QUIT
```



```
                              Student Login
Enter UID:          6115045
Enter Password:     *********
```

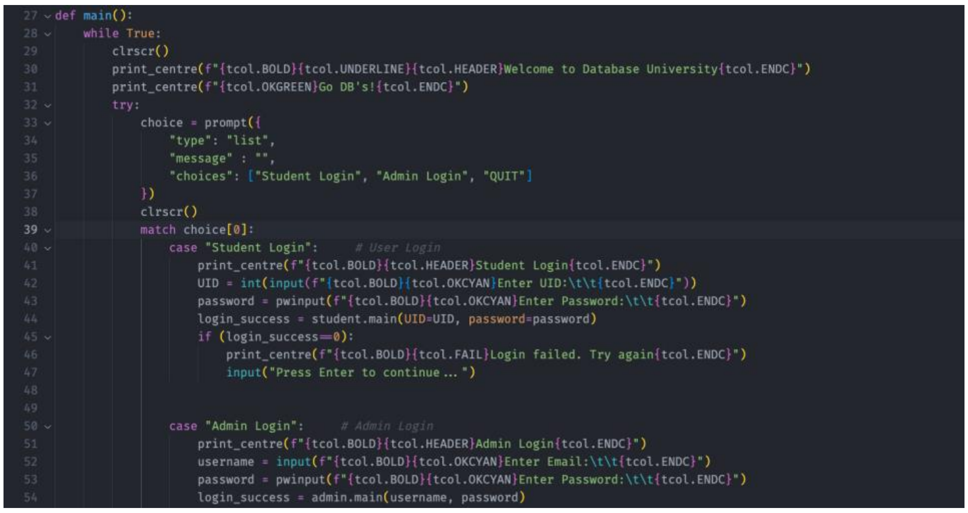


```
                              STUDENT PORTAL
?
❯ View personal information
  Edit personal information
  Access registered classes
  Modiying the list of registered classes (add/remove)
  Obtain a list of all professors within the department of the students major
  LOGOUT
```



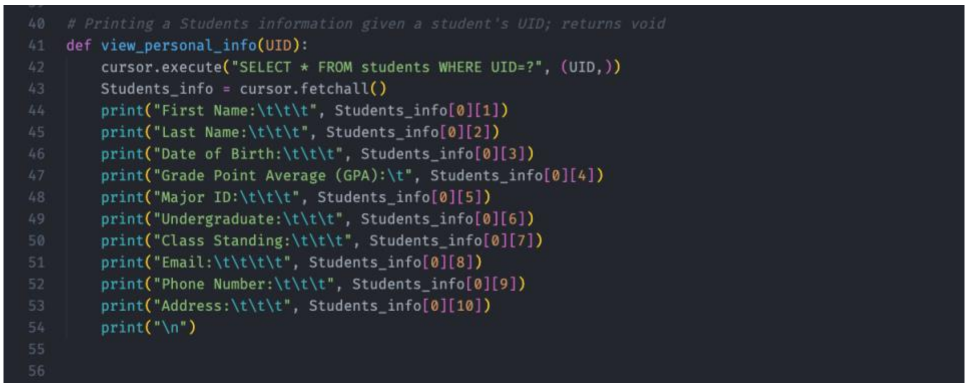
```
                              STUDENT PORTAL
?  Edit personal information
?
❯ First Name
  Last Name
  Date of Birth
  GPA
  Major ID
  Undergraduate
  Class Standing
  Email
  Phone Number
  Address
  CANCEL
```

Some example code showing our implementation:



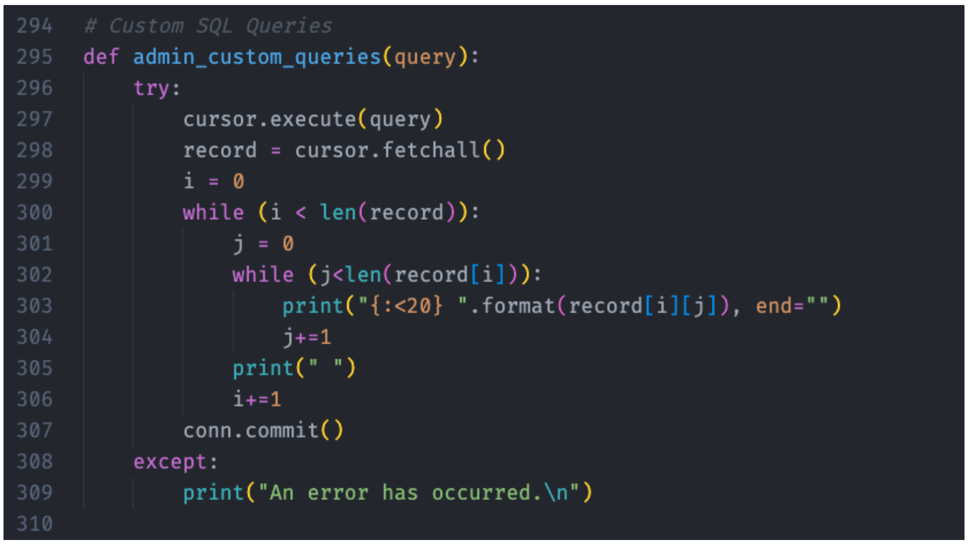
```python
def main():
    while True:
        clrscr()
        print_centre(f"{tcol.BOLD}{tcol.UNDERLINE}{tcol.HEADER}Welcome to Database University{tcol.ENDC}")
        print_centre(f"{tcol.OKGREEN}Go DB's!{tcol.ENDC}")
        try:
            choice = prompt({
                "type": "list",
                "message" : "",
                "choices": ["Student Login", "Admin Login", "QUIT"]
            })
            clrscr()
            match choice[0]:
                case "Student Login":      # User Login
                    print_centre(f"{tcol.BOLD}{tcol.HEADER}Student Login{tcol.ENDC}")
                    UID = int(input(f"{tcol.BOLD}{tcol.OKCYAN}Enter UID:\t\t{tcol.ENDC}"))
                    password = pwinput(f"{tcol.BOLD}{tcol.OKCYAN}Enter Password:\t\t{tcol.ENDC}")
                    login_success = student.main(UID=UID, password=password)
                    if (login_success==0):
                        print_centre(f"{tcol.BOLD}{tcol.FAIL}Login failed. Try again{tcol.ENDC}")
                        input("Press Enter to continue ... ")


                case "Admin Login":     # Admin Login
                    print_centre(f"{tcol.BOLD}{tcol.HEADER}Admin Login{tcol.ENDC}")
                    username = input(f"{tcol.BOLD}{tcol.OKCYAN}Enter Email:\t\t{tcol.ENDC}")
                    password = pwinput(f"{tcol.BOLD}{tcol.OKCYAN}Enter Password:\t\t{tcol.ENDC}")
                    login_success = admin.main(username, password)
```



```python
# Printing a Students information given a student's UID; returns void
def view_personal_info(UID):
    cursor.execute("SELECT * FROM students WHERE UID=?", (UID,))
    Students_info = cursor.fetchall()
    print("First Name:\t\t\t", Students_info[0][1])
    print("Last Name:\t\t\t", Students_info[0][2])
    print("Date of Birth:\t\t\t", Students_info[0][3])
    print("Grade Point Average (GPA):\t", Students_info[0][4])
    print("Major ID:\t\t\t", Students_info[0][5])
    print("Undergraduate:\t\t\t", Students_info[0][6])
    print("Class Standing:\t\t\t", Students_info[0][7])
    print("Email:\t\t\t\t", Students_info[0][8])
    print("Phone Number:\t\t\t", Students_info[0][9])
    print("Address:\t\t\t", Students_info[0][10])
    print("\n")


```



```python
# Custom SQL Queries
def admin_custom_queries(query):
    try:
        cursor.execute(query)
        record = cursor.fetchall()
        i = 0
        while (i < len(record)):
            j = 0
            while (j<len(record[i])):
                print("{:<20} ".format(record[i][j]), end="")
                j+=1
            print(" ")
            i+=1
        conn.commit()
    except:
        print("An error has occurred.\n")

```