King Abdulaziz University
Faculty of Engineering
Object Oriented Programing - EE202
Fall-2023

EE-202

Project Report

Member No.	***Name***	***ID***
#1	Abdullah Binsalman	2236900
#2	Yamin Muhjeb	2235916
#3	Ahmed Al-Harthy	2237094

Table Of Content

Introduction:

This report contains a brief explanation of the project “Student Registration System” functionality, the report explains and displays information about the Classes structure that are used in this project, with the UML Diagrams for each class. Also, In the end of this report the Main file which has the full project with the GUI pages will be briefly explained.

Class structure:

Class Course:

The first class in this project is about the courses that the students take, this class consists of several methods that will be shown in the Figures below as well as the UML Diagram.

MyPage(Current) > Course.py > Course > getCourse

```python
class Course:
    def __init__(self, name = "",  title="", subjectNumber= 0, hours= 0, section=0 , mark=int(0)):
        self.title = title
        self.name = name
        self.subjectNumber = str(subjectNumber)
        self.hours = hours
        self.section = section
        self.mark = int(mark)
        self.info=[title, subjectNumber, hours, section, mark]

    def setTitle(self, newTitle):
        self.title = newTitle

    def setName(self, newName):
        self.name = newName

    def setSubjectNumber(self, newSubjectNumber):
        self.subjectNumber = newSubjectNumber

    def setHours(self, newHours):
        self.hours = newHours

    def setSection(self, newSection):
        self.section = newSection

    def setMark(self, newMark):
        self.mark = int(newMark)

    def setGrade(self, newgrade):
        self.grade = int(newgrade)
```

FIGURE 1: CLASS COURSE PART 1 OF 5

FIGURE 2: CLASS COURSE PART 2 OF 5

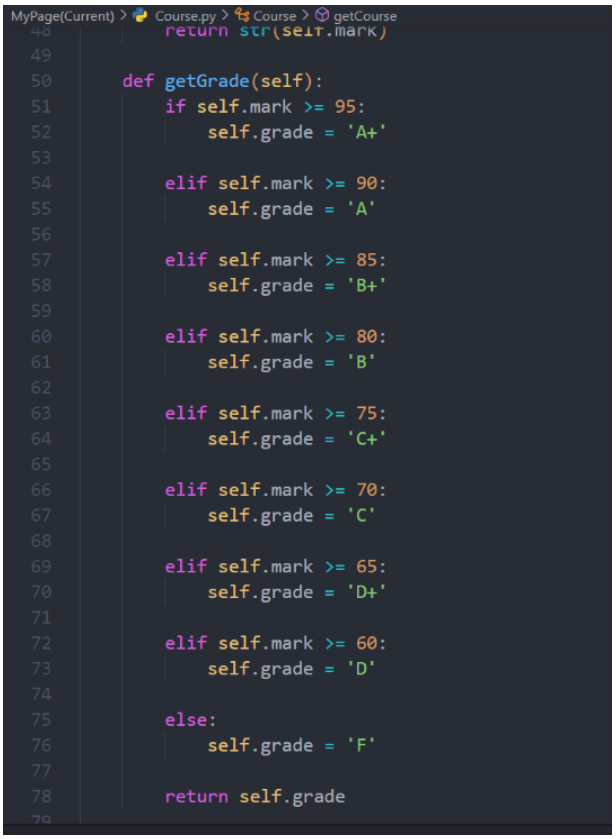

FIGURE 4: CLASS COURSE PART 3 OF 5

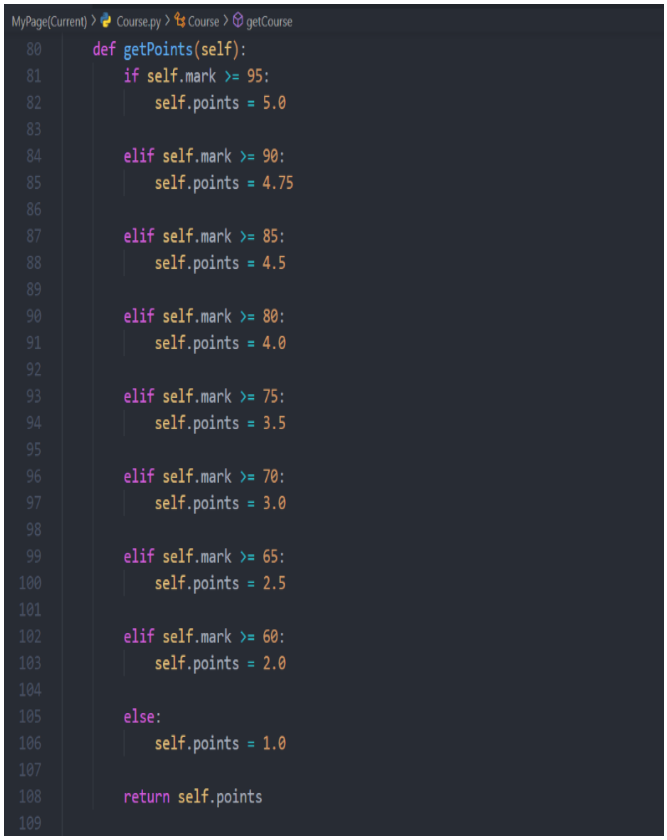

FIGURE 3: CLASS COURSE PART 4 OF 5

```
110     def getCourse(self):
111         return self.info
112 
113     def __str__(self):
114         s = f'{self.subjectNumber}' + (10-len(self.subjectNumber))*' ' + \
115             f' {self.title}' + (35-len(self.title))*' ' + \
116             f' {self.name}' + (35-len(self.name))*' ' + \
117             f' {self.hours}' + (10-len(str(self.hours)))*' ' + \
118             f' {self.section}' + (10-len(str(self.section)))*' ' + \
119             f' {self.mark}' + (10-len(str(self.mark)))*' ' + \
120             f' {self.getGrade()}' + (10-len(str(self.getGrade())))*' ' + \
121             f' {self.getPoints()}'
122 
123         return s
```

FIGURE 5: CLASS COURSE PART 5 OF 5

As shown in the above Figures (1-5) the codes for Class Course, in Figure (1) the constructor method is displayed and 6 arguments are requested to create an object from class course, and the method for each argument is shown in Figures (2-4), each argument had a method of “GET” to print the attribute of the object and the method “SET” to change the value of the attribute, and in Figure(5) the str method for the class is shown. The UML Diagram for class course will be shown in Table (1).

TABLE 1: COURSE UML DIAGRAM

Course
name: str title: str subjectNumber: str hours: int section: str mark: int
setTitle(newTitle): setName(newName): setSubjectNumber(newSubjectNumber): setHours(newHours): setSection(newSection): setMark(newMark): setGrade(newgrade): getName(): getTitle(): getSubjectnumber(): getHours(): getSection(): getMark(): getGrade(): getPoints(): getCourse(): __str__():

Class Semester:

The second class in this project is about the semesters that will have the objects of class course and then will be add to object of class students. The UML Diagram for the semester class will be shown below alongside the attributes of the class.

Main.py Class_Semester.py ×

Class_Semester.py > Semester > getPoints

```python

class Semester():

    def __init__(self,Semester_name, Courses = []):
        self.Semester_name=Semester_name
        self.Courses= list(Courses)

    def edit_semester_name(self,Semester_name):
        self.Semester_name=Semester_name

    def add_course(self,newCourses):
        self.Courses.append(newCourses)

    def get_courses(self):
        return self.Courses

    def del_course(self,REcourse):
        self.Courses.remove(REcourse)

    def GetSemName(self):
        return self.Semester_name
```

FIGURE 6: CLASS SEMESTER PART 1 OF 3

```python
    def getPoints(self):
        points = 0
        for course in self.Courses:
            points += float(course.getPoints()) * float(course.getHours())
        return points

    def getHours(self):
        hours = 0
        for course in self.Courses:
            hours += float(course.getHours())
        return hours

    def getGPA(self):
        return round(self.getPoints()/self.getHours(), 2)

```

FIGURE 7: CLASS SEMESTER PART 2 OF 3

```python
    def __str__(self):
        s = f"{self.Semester_name}\n" + \
            "-----------------------------------------------------------------------------------------------\n\n"

        for i in range(len(self.Courses)):
            s+=str(self.Courses[i])+"\n"

        s += f'\nSemester GPA: {round(self.getGPA(), 2)}\t\t Hours: {self.getHours()}\t\t Points: {self.getPoints
            "-----------------------------------------------------------------------------------------------\n"
        return s
```

FIGURE 8: CLASS SEMESTER PART 3 OF 3

As shown in Figure (6) the constructer method of class semester requires only two arguments which is the name of the semester and the objects of class course that are added to the list of an object of class semester, each attribute of the object has a “GET” and “SET” method as shown in Figures (6-7). Also, there’s are methods to remove, add or get Courses in class semester. Moreover, the string method for this class is also shown in Figure (8). And the UML Diagram for this class is shown below in Table (2).

TABLE 2: SEMESTER UML DIAGRAM

Semester
Semester_name: str Courses: list
edit_semester_name(Semester_name): add_course(newCourses): get_courses(): del_course(REcourse): GetSemName(): getPoints(): getHours(): getGPA(): __str__():

Class Student:

Lastly, the Student class is the final class to complete this project, this class will contain objects from both classes' semester and course, it will also be added to a list in the main file to be used as data. The codes and UML Diagrams for this class will be shown below in Figures (9-11) and Table (3).

```python
class Student:
    def __init__(self, name='XXXXXX', ID='181810', semesters=None):
        self.name = name
        self.ID = ID
        if semesters is None:
            self.semesters = []
        else:
            self.semesters = semesters

    def setName(self, name):
        self.name = name

    def setID(self, ID):
        self.ID = ID

    def addSemester(self, semester):
        if isinstance(semester, Semester):
            self.semesters.append(semester)

    def getName(self):
        return self.name
```

FIGURE 9: CLASS STUDENT PART 1 OF 3

```python
    def getID(self):
        return self.ID

    def getSemesters(self):
        return self.semesters



    def getHours(self):
        hours = 0
        for semester in self.semesters:
            hours += semester.getHours()
        return hours

    def getPoints(self):
        points = 0
        for semester in self.semesters:
            points += semester.getPoints()
        return points
```

FIGURE 10: CLASS STUDENT PART 2 OF 3

```
47      def getGPA(self):
48          return self.getPoints() / self.getHours()
49
50
51      def removeSemester(self, index):
52          del self.semesters[index]
53
54
55      def __str__(self):
56
57          info=f'Name: {self.name}\n'+ \
58               f'ID: {self.ID}\n' + \
59               f'GPA: {round(self.getGPA(), 2)}\n\n'
60
61          for i in range(len(self.semesters)):
62              info+=f"{self.semesters[i]}\n"
63          return info
```

FIGURE 11: CLASS STUDENT PART 3 OF 3

In the Figure (9) the constructer method for the student class is shown, this class requires three arguments, the first one is the fulls name of the student, second one is the ID number of the student, and the third and last one is an object of semester which will be add to a list in the object of class student, each attribute in this class has a “GET and “SET” method as shown in Figures (9-10), while also having three method to control the object of class semester, the first method is adding a semester to list, the second method is getting the list of semester, and the last method is to remove a semester from the list, the str method for class student is also shown in Figure (11). The UML Diagram for this class is shown in Table (3).

TABLE 3: STUDENT UML DIAGRAM

Student
name: str ID: str semesters: list
setName(name): setID(ID): addSemester(semester): getName(): getID(): getSemesters(): getHours(): getPoints(): getGPA(): __str__():

Main File:

The main file contains all the classes mentioned above, it also contains all the GUI pages which will be displayed below. The idea to gather all the work in one file helps us if we needed to add or change a GUI page without the risk of starting from the beginning. The main file has many lines of code. For Example, connecting buttons in each page of the GUI to a function that has a purpose of either showing information or executing a command and many more purposes. The full file can't be shown here in this report due to the large amount of code lines in the file, but the diagram will explain the GUI structure in a simpler way which replaces the need to show code, the code will be explained briefly in the diagram Figure (12) below. Also, the GUI pages will be show below in the Figures (13-17).

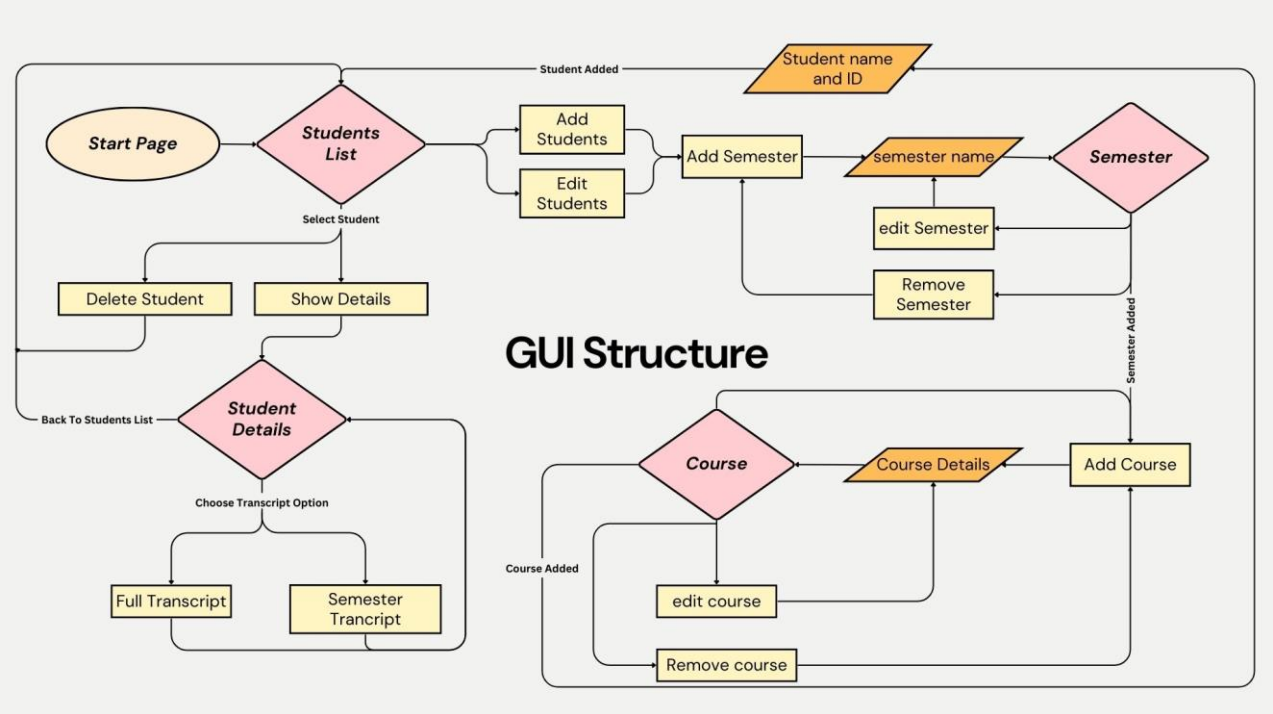

FIGURE 12: GUI STRUCTURE CODE LOOP

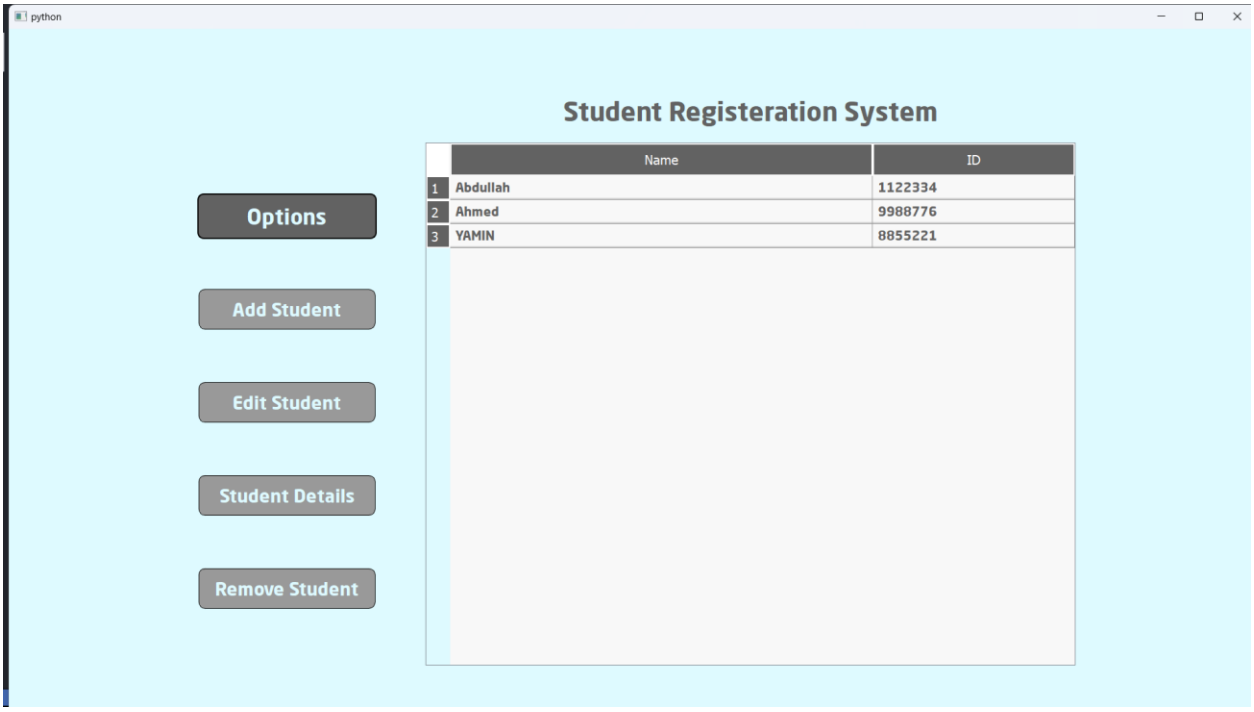

FIGURE 13: GUI PAGE 1

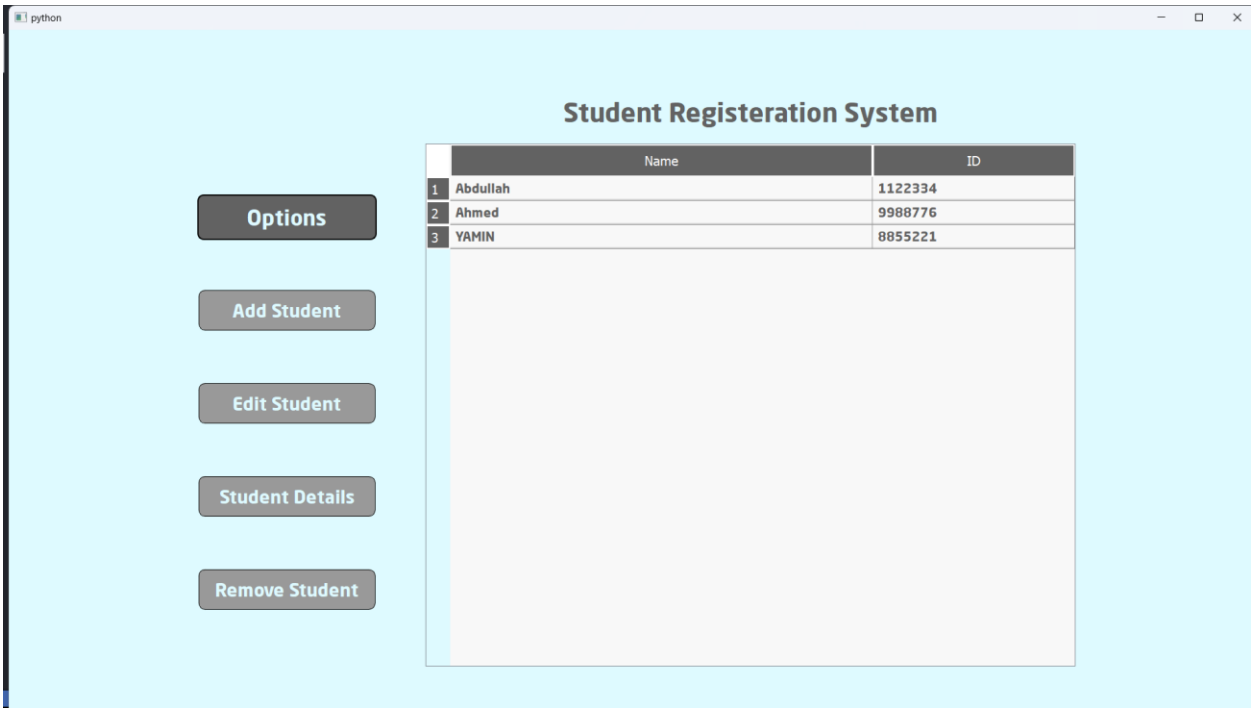

FIGURE 14: GUI PAGE 2

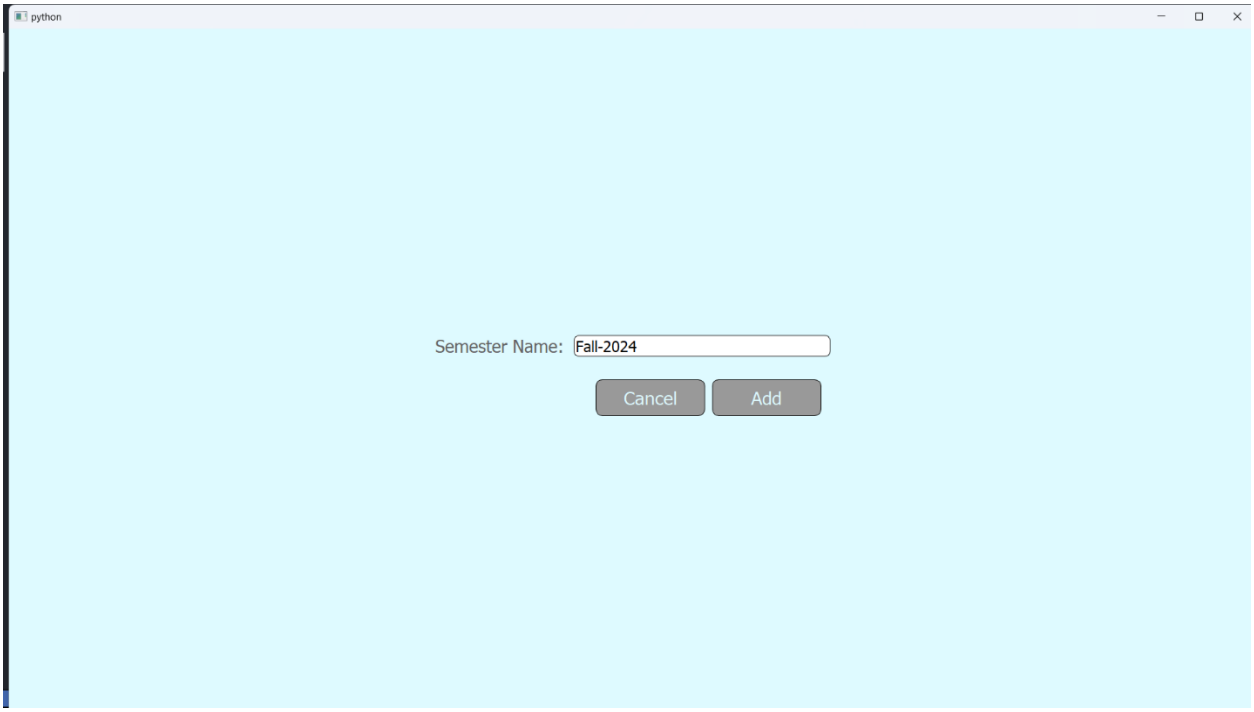

FIGURE 15: GUI PAGE 3

FIGURE 16: GUI PAGE 4

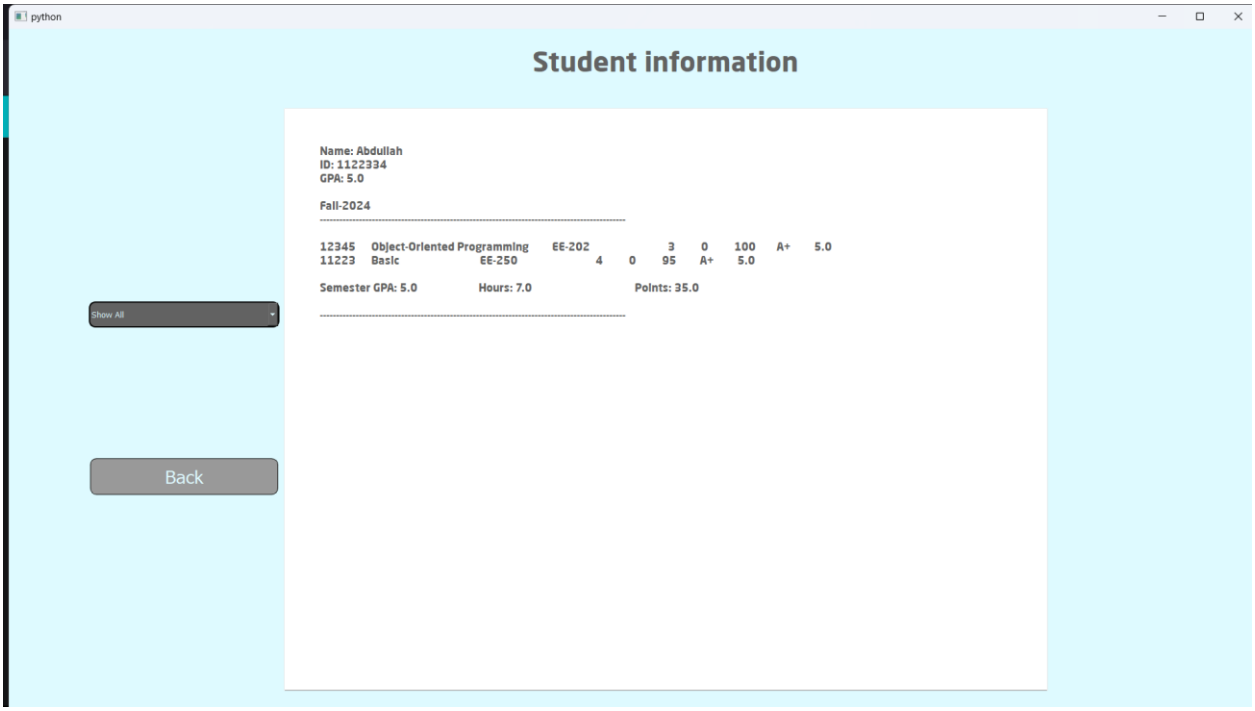

FIGURE 17: GUI PAGE 5

Conclusion:

To conclude, we learned and gained a lot of experience from doing this project, it contained a lot of aspects that the course EE-202 covered. Also, the project needed a lot of self-learning to complete, which is a very useful skill to learn especially in programing where most languages are self-learned. We hope to build on this experience in the upcoming years and develop more programs such as this project, which will help in becoming better programmers and engineers.