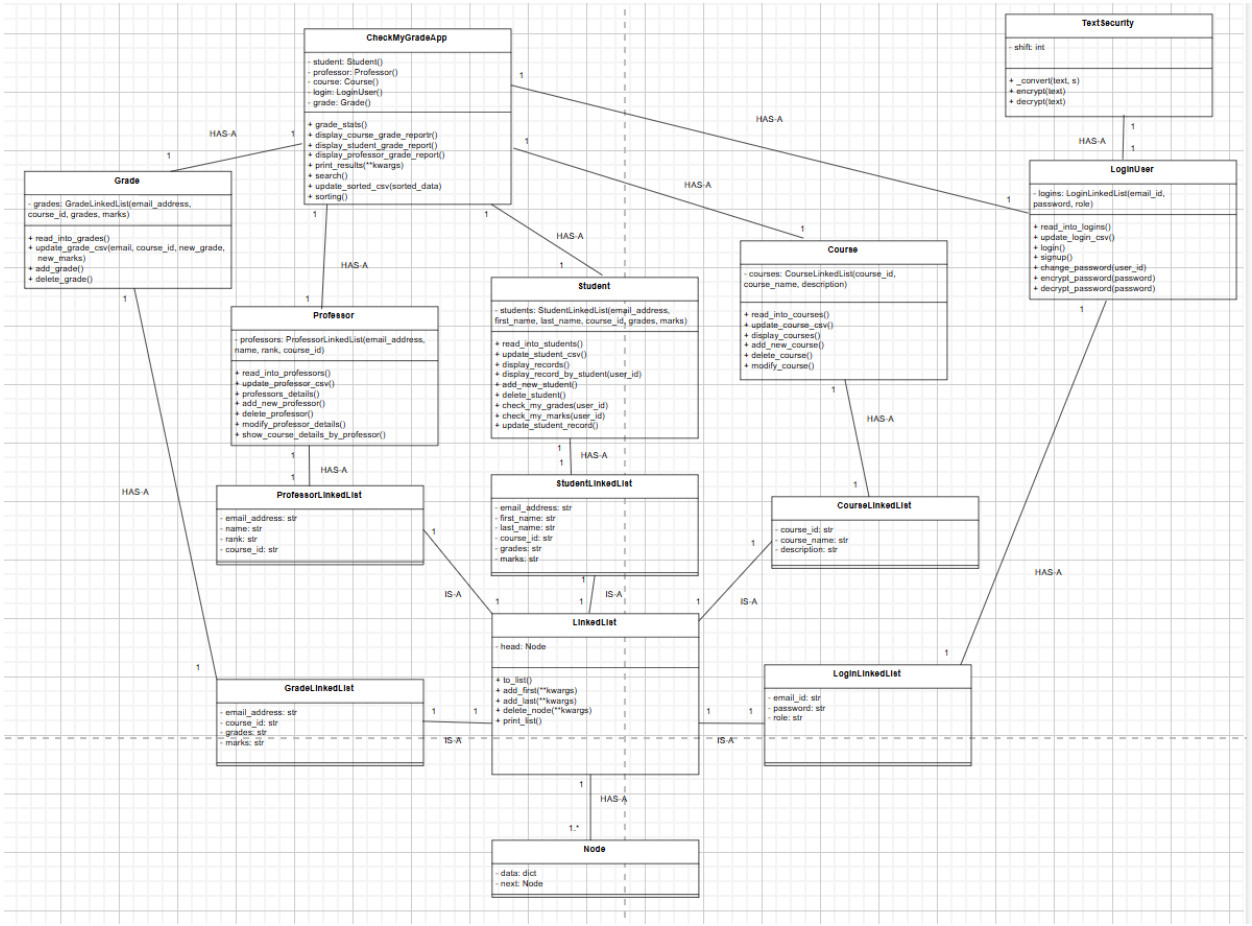
Rish Jain // rish.jain@sjsu.edu // 015731690

Professor Param

DATA 200

16 March 2025

IS-A/HAS-A Relationship Diagram



Unit Testing:

Generating 1000 student records: This unit test clears the student.csv file and uses the random and names library to fill the student.csv file with random data such as first name, last name, email address, etc. It also prints the timing of generating the 1000 records.

```python
class TestStudentManagement(unittest.TestCase):

    def generate_students(self, num_records):
        """Generate and write student records to student.csv."""
        # Reset students data for a clean start
        self.app.student.students.head = None

        start = time.time()

        with open("student.csv", mode='w', newline='') as file:
            fieldnames = ["email_address", "first_name", "last_name", "course_id", "grades", "marks"]
            writer = csv.DictWriter(file, fieldnames=fieldnames)
            writer.writeheader()
            for i in range(num_records):
                marks = random.randint(0, 100)
                grade = "A" if marks >= 90 else "B" if marks >= 80 else "C" if marks >= 70 else "D" if marks >= 60 else "F"
                first_name = names.get_first_name()
                last_name = names.get_last_name()

                writer.writerow({
                    "email_address": f"{first_name.lower()}.{last_name.lower()}{i}@university.edu",
                    "first_name": first_name,
                    "last_name": last_name,
                    "course_id": f"CS{i%10 + 101}",
                    "grades": grade,
                    "marks": str(marks)
                })

        end = time.time()
        elapsed = end - start
        print(f"{num_records} student records generated in {elapsed:.15f} seconds")
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\rishx\Downloads\SJSU DATA 200\RishJain-Lab1> & C:/Users/rishx/AppData/Local/Programs/Python/Python312/python.exe "c:/Users/rishx/Downloads/SJSU DATA 200/RishJain-Lab1/unit_test_Rish.py"
1000 student records generated in 3.39734292030345 seconds
PS C:\Users\rishx\Downloads\SJSU DATA 200\RishJain-Lab1>
```

essor_file.py | unit_test_Rish.py | student.csv | course.csv | professor.csv

student.csv > data

```
email_address,first_name,last_name,course_id,grades,marks
edwin.tong991@university.edu,Edwin,Tong,CS102,A,92
eva.kelly992@university.edu,Eva,Kelly,CS103,B,84
floyd.schultz993@university.edu,Floyd,Schultz,CS104,F,5
laura.orear994@university.edu,Laura,Orear,CS105,F,39
david.britt995@university.edu,David,Britt,CS106,C,71
michele.scott996@university.edu,Michele,Scott,CS107,F,38
nicole.malone997@university.edu,Nicole,Malone,CS108,A,98
christine.moffett998@university.edu,Christine,Moffett,CS109,C,75
jacob.britton999@university.edu,Jacob,Britton,CS110,A,92
```

Testing add student: This unit test uses mock input (since my program is interactive) to manually add a student to both the StudentLinkedList and student.csv file using the add_new_student() function. It also prints the timing.

```
 5  class TestStudentManagement(unittest.TestCase):
41      @patch("builtins.input", side_effect=["new_student@csu.edu", "John", "Doe", "CS101", 89])
42      def test_add_student(self, mock_input):
43          start = time.time()
44          self.app.student.add_new_student()
45          end = time.time()
46          elapsed = end - start
47          print(f"Add student completed in {elapsed:.15f} seconds")
48          self.assertTrue(elapsed < 1, "Add took too long")
49
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\rishx\Downloads\SJSU DATA 200\RishJain-Lab1> & C:/Users/rishx/AppData/Local/Programs/Python/Python312/python.exe "c:/Users
hx/Downloads/SJSU DATA 200/RishJain-Lab1/unit_test_Rish.py"
1000 student records generated in 3.397342920303345 secon^C
PS C:\Users\rishx\Downloads\SJSU DATA 200\RishJain-Lab1> python -m unittest unit_test_Rish.TestStudentManagement.test_add_student
Add student completed in 0.007052421569824 seconds
.
----------------------------------------------------------------------
Ran 1 test in 0.038s

OK
PS C:\Users\rishx\Downloads\SJSU DATA 200\RishJain-Lab1>
```

```
   1  email_address,first_name,last_name,course_id,grades,marks
 995  floyd.schultz993@university.edu,Floyd,Schultz,CS104,F,5
 996  laura.orear994@university.edu,Laura,Orear,CS105,F,39
 997  david.britt995@university.edu,David,Britt,CS106,C,71
 998  michele.scott996@university.edu,Michele,Scott,CS107,F,38
 999  nicole.malone997@university.edu,Nicole,Malone,CS108,A,98
1000  christine.moffett998@university.edu,Christine,Moffett,CS109,C,75
1001  jacob.britton999@university.edu,Jacob,Britton,CS110,A,92
1002  new_student@csu.edu,John,Doe,CS101,B,89
1003
```

Testing modify student: This unit test uses mock input (since my program is interactive) to manually modify a student in both the StudentLinkedList and student.csv file using the update_student_record() function. It also prints the timing.

```
class TestStudentManagement(unittest.TestCase):

    @patch("builtins.input", side_effect=["new_student@csu.edu", "nstudent@csu.edu", "new_first_name", "new_last_name", "new_cou
    def test_modify_student(self, mock_input):
        start = time.time()
        self.app.student.update_student_record()
        end = time.time()

        elapsed = end - start
        print(f"Modify student completed in {elapsed:.15f} seconds")
        self.assertTrue(elapsed < 1, "Modify took too long")
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\rishx\Downloads\SJSU DATA 200\RishJain-Lab1> & C:/Users/rishx/AppData/Local/Programs/Python/Python312/python.exe "c:/Users/ri
hx/Downloads/SJSU DATA 200/RishJain-Lab1/unit_test_Rish.py"
1000 student records generated in 3.397342920303345 secon^C
PS C:\Users\rishx\Downloads\SJSU DATA 200\RishJain-Lab1> python -m unittest unit_test_Rish.TestStudentManagement.test_add_student
Add student completed in 0.007052421569824 seconds
.
----------------------------------------------------------------------
Ran 1 test in 0.038s

OK
PS C:\Users\rishx\Downloads\SJSU DATA 200\RishJain-Lab1> python -m unittest unit_test_Rish.TestStudentManagement.test_modify_student
Modify student completed in 0.002999782562256 seconds
.
----------------------------------------------------------------------
Ran 1 test in 0.037s

OK
PS C:\Users\rishx\Downloads\SJSU DATA 200\RishJain-Lab1>
```

```
email_address,first_name,last_name,course_id,grades,marks
floyd.schultz993@university.edu,Floyd,Schultz,CS104,F,5
laura.orear994@university.edu,Laura,Orear,CS105,F,39
david.britt995@university.edu,David,Britt,CS106,C,71
michele.scott996@university.edu,Michele,Scott,CS107,F,38
nicole.malone997@university.edu,Nicole,Malone,CS108,A,98
christine.moffett998@university.edu,Christine,Moffett,CS109,C,75
jacob.britton999@university.edu,Jacob,Britton,CS110,A,92
nstudent@csu.edu,new_first_name,new_last_name,new_course,C,75
```

Testing delete student: This unit test uses mock input (since my program is interactive) to manually delete a student from both the StudentLinkedList and student.csv file using the delete_student() function. It also prints the timing.

```
5   class TestStudentManagement(unittest.TestCase):
60      @patch("builtins.input", side_effect=["nstudent@csu.edu"])
61      def test_delete_student(self, mock_input):
62          start = time.time()
63          self.app.student.delete_student()
64          end = time.time()
65          elapsed = end - start
66          print(f"Delete student completed in {elapsed:.15f} seconds")
67          self.assertTrue(elapsed < 1, "Delete took too long")
68
69
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\rishx\Downloads\SJSU DATA 200\RishJain-Lab1> & C:/Users/rishx/AppData/Local/Programs/Python/Python312/python.exe "c:/Users/ri
hx/Downloads/SJSU DATA 200/RishJain-Lab1/unit_test_Rish.py"
1000 student records generated in 3.397342920303345 secon^C
PS C:\Users\rishx\Downloads\SJSU DATA 200\RishJain-Lab1> python -m unittest unit_test_Rish.TestStudentManagement.test_add_student
Add student completed in 0.007052421569824 seconds
.
----------------------------------------------------------------------
Ran 1 test in 0.038s

OK
PS C:\Users\rishx\Downloads\SJSU DATA 200\RishJain-Lab1> python -m unittest unit_test_Rish.TestStudentManagement.test_modify_student
Modify student completed in 0.002999782562256 seconds
.
----------------------------------------------------------------------
Ran 1 test in 0.037s

OK
PS C:\Users\rishx\Downloads\SJSU DATA 200\RishJain-Lab1> python -m unittest unit_test_Rish.TestStudentManagement.test_delete_student
Student deleted successfully
Delete student completed in 0.006300926208496 seconds
.
----------------------------------------------------------------------
Ran 1 test in 0.038s

OK
PS C:\Users\rishx\Downloads\SJSU DATA 200\RishJain-Lab1>
```

```
email_address,first_name,last_name,course_id,grades,marks
floyd.schultz993@university.edu,Floyd,Schultz,CS104,F,5
laura.orear994@university.edu,Laura,Orear,CS105,F,39
david.britt995@university.edu,David,Britt,CS106,C,71
michele.scott996@university.edu,Michele,Scott,CS107,F,38
nicole.malone997@university.edu,Nicole,Malone,CS108,A,98
christine.moffett998@university.edu,Christine,Moffett,CS109,C,75
jacob.britton999@university.edu,Jacob,Britton,CS110,A,92
```

Testing add course: This unit test uses mock input (since my program is interactive) to manually add a course to both the CourseLinkedList and course.csv file using the add_new_course() function. It also prints the timing.

```
70  class TestCourseManagement(unittest.TestCase):
75
76      @patch("builtins.input", side_effect=["DATA111", "Intro to Python", "Gives an intro to Python"])
77      def test_add_course(self, mock_input):
78          start = time.time()
79          self.app.course.add_new_course()
80          end = time.time()
81          elapsed = end - start
82          print(f"Add course completed in {elapsed:.15f} seconds")
83          self.assertTrue(elapsed < 1, "Add took too long")
84
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
OK
PS C:\Users\rishx\Downloads\SJSU DATA 200\RishJain-Lab1> python -m unittest unit_test_Rish.TestStudentManagement.test_delete_studen
Student deleted successfully
Delete student completed in 0.006300926208496 seconds
.
----------------------------------------------------------------------
Ran 1 test in 0.038s

OK
PS C:\Users\rishx\Downloads\SJSU DATA 200\RishJain-Lab1> python -m unittest unit_test_Rish.TestCourseManagement.test_add_course
Course already exists
Add course completed in 0.000000000000000 seconds
.
----------------------------------------------------------------------
Ran 1 test in 0.033s

OK
PS C:\Users\rishx\Downloads\SJSU DATA 200\RishJain-Lab1> python -m unittest unit_test_Rish.TestCourseManagement.test_add_course

Course added successfully.
Add course completed in 0.001010179519653 seconds
.
----------------------------------------------------------------------
Ran 1 test in 0.040s

OK
```

course.csv > data

```
1   course_id,course_name,description
2   DATA111,Intro to Python,Gives an intro to Python
3
```

Testing modify course: This unit test uses mock input (since my program is interactive) to manually modify a course in both the CourseLinkedList and course.csv file using the modify_course() function. It also prints the timing.

```
70  class TestCourseManagement(unittest.TestCase):
84
85      @patch("builtins.input", side_effect=["DATA111", "DATA102","Intro to SQL", "Gives an intro to SQL"])
86      def test_modify_course(self, mock_input):
87          start = time.time()
88          self.app.course.modify_course()
89          end = time.time()
90          elapsed = end - start
91          print(f"Modify course completed in {elapsed:.15f} seconds")
92          self.assertTrue(elapsed < 1, "Modify took too long")
93
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
Ran 1 test in 0.033s

OK
PS C:\Users\rishx\Downloads\SJSU DATA 200\RishJain-Lab1> python -m unittest unit_test_Rish.TestCourseManagement.test_add_course

Course added successfully.
Add course completed in 0.001010179519653 seconds
.
----------------------------------------------------------------------
Ran 1 test in 0.040s

OK
PS C:\Users\rishx\Downloads\SJSU DATA 200\RishJain-Lab1> python -m unittest unit_test_Rish.TestCourseManagement.test_modify_course
No records found.
Modify course completed in 0.000998973846436 seconds
.
----------------------------------------------------------------------
Ran 1 test in 0.032s

OK
PS C:\Users\rishx\Downloads\SJSU DATA 200\RishJain-Lab1> python -m unittest unit_test_Rish.TestCourseManagement.test_modify_course
Modify course completed in 0.001005649566650 seconds
.
----------------------------------------------------------------------
Ran 1 test in 0.036s

OK
PS C:\Users\rishx\Downloads\SJSU DATA 200\RishJain-Lab1>
```

course.csv > data

```
1   course_id,course_name,description
2   DATA102,Intro to SQL,Gives an intro to SQL
3
```

Testing delete course: This unit test uses mock input (since my program is interactive) to manually delete a course from both the CourseLinkedList and course.csv file using the delete_course() function. It also prints the timing.

```
70  class TestCourseManagement(unittest.TestCase):
93
94      @patch("builtins.input", side_effect=["DATA102"])
95      def test_delete_course(self, mock_input):
96          start = time.time()
97          self.app.course.delete_course()
98          end = time.time()
99          elapsed = end - start
100         print(f"Delete course completed in {elapsed:.15f} seconds")
101         self.assertTrue(elapsed < 1, "Delete took too long")
102
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
----------------------------------------------------------------------
Ran 1 test in 0.040s

OK
PS C:\Users\rishx\Downloads\SJSU DATA 200\RishJain-Lab1> python -m unittest unit_test_Rish.TestCourseManagement.test_modify_course
No records found.
Modify course completed in 0.000998973846436 seconds
.
----------------------------------------------------------------------
Ran 1 test in 0.032s

OK
PS C:\Users\rishx\Downloads\SJSU DATA 200\RishJain-Lab1> python -m unittest unit_test_Rish.TestCourseManagement.test_modify_course
Modify course completed in 0.001005649566650 seconds
.
----------------------------------------------------------------------
Ran 1 test in 0.036s

OK
PS C:\Users\rishx\Downloads\SJSU DATA 200\RishJain-Lab1> python -m unittest unit_test_Rish.TestCourseManagement.test_delete_course
Course deleted successfully
Delete course completed in 0.000999927520752 seconds
.
----------------------------------------------------------------------
Ran 1 test in 0.033s

OK
```

course.csv > data

```
1   course_id,course_name,description
2
```

Testing add professor: This unit test uses mock input (since my program is interactive) to manually add a professor to both the ProfessorLinkedList and professor.csv file using the add_new_professor() function. It also prints the timing.

```
103  class TestProfessorManagement(unittest.TestCase):
107
108      @patch("builtins.input", side_effect=["test_prof@mycsu.edu", "Professor Test", "Test Professor", "DATA101"])
109      def test_add_professor(self, mock_input):
110          start = time.time()
111          self.app.professor.add_new_professor()
112          end = time.time()
113          elapsed = end - start
114          print(f"Add professor completed in {elapsed:.15f} seconds")
115          self.assertTrue(elapsed < 1, "Add took too long")
116
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

.
----------------------------------------------------------------------
Ran 1 test in 0.032s

OK
PS C:\Users\rishx\Downloads\SJSU DATA 200\RishJain-Lab1> python -m unittest unit_test_Rish.TestCourseManagement.test_modify_course
Modify course completed in 0.001005649566650 seconds
.
----------------------------------------------------------------------
Ran 1 test in 0.036s

OK
PS C:\Users\rishx\Downloads\SJSU DATA 200\RishJain-Lab1> python -m unittest unit_test_Rish.TestCourseManagement.test_delete_course
Course deleted successfully
Delete course completed in 0.000999927520752 seconds
.
----------------------------------------------------------------------
Ran 1 test in 0.033s

OK
PS C:\Users\rishx\Downloads\SJSU DATA 200\RishJain-Lab1> python -m unittest unit_test_Rish.TestProfessorManagement.test_add_professor
Add professor completed in 0.001000404357910 seconds
.
----------------------------------------------------------------------
Ran 1 test in 0.039s

OK
PS C:\Users\rishx\Downloads\SJSU DATA 200\RishJain-Lab1>
```

professor.csv > data

```
1  professor_id,professor_name,rank,course_id
2  test_prof@mycsu.edu,Professor Test,Test Professor,DATA101
3
```

Testing modify professor: This unit test uses mock input (since my program is interactive) to manually modify a professor in both the ProfessorLinkedList and professor.csv file using the modify_professor_details() function. It also prints the timing.

```
103  class TestProfessorManagement(unittest.TestCase):
116
117      @patch("builtins.input", side_effect=["test_prof@mycsu.edu", "testprof@mycsu.edu","Professor Testing", "Testing Professor",
118      def test_modify_professor(self, mock_input):
119          start = time.time()
120          self.app.professor.modify_professor_details()
121          end = time.time()
122          elapsed = end - start
123          print(f"Modify professor completed in {elapsed:.15f} seconds")
124          self.assertTrue(elapsed < 1, "Modify took too long")
125
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
.
----------------------------------------------------------------------
Ran 1 test in 0.033s

OK
PS C:\Users\rishx\Downloads\SJSU DATA 200\RishJain-Lab1> python -m unittest unit_test_Rish.TestProfessorManagement.test_add_professor
Add professor completed in 0.001000404357910 seconds
.
----------------------------------------------------------------------
Ran 1 test in 0.039s

OK
PS C:\Users\rishx\Downloads\SJSU DATA 200\RishJain-Lab1> python -m unittest unit_test_Rish.TestProfessorManagement.test_modify_professor
No records found.
Modify professor completed in 0.001198291778564 seconds
.
----------------------------------------------------------------------
Ran 1 test in 0.032s

OK
PS C:\Users\rishx\Downloads\SJSU DATA 200\RishJain-Lab1> python -m unittest unit_test_Rish.TestProfessorManagement.test_modify_professor
Modify professor completed in 0.001096725463867 seconds
.
----------------------------------------------------------------------
Ran 1 test in 0.031s

OK
PS C:\Users\rishx\Downloads\SJSU DATA 200\RishJain-Lab1>
```

professor.csv > data

```
1  professor_id,professor_name,rank,course_id
2  testprof@mycsu.edu,Professor Testing,Testing Professor,DATA201
3
```

Testing delete professor: This unit test uses mock input (since my program is interactive) to manually delete a professor from both the ProfessorLinkedList and professor.csv file using the delete_professor() function. It also prints the timing.

```
103  class TestProfessorManagement(unittest.TestCase):
125
126      @patch("builtins.input", side_effect=["testprof@mycsu.edu"])
127      def test_delete_professor(self, mock_input):
128          start = time.time()
129          self.app.professor.delete_professor()
130          end = time.time()
131          elapsed = end - start
132          print(f"Delete professor completed in {elapsed:.15f} seconds")
133          self.assertTrue(elapsed < 1, "Delete took too long")
134
```

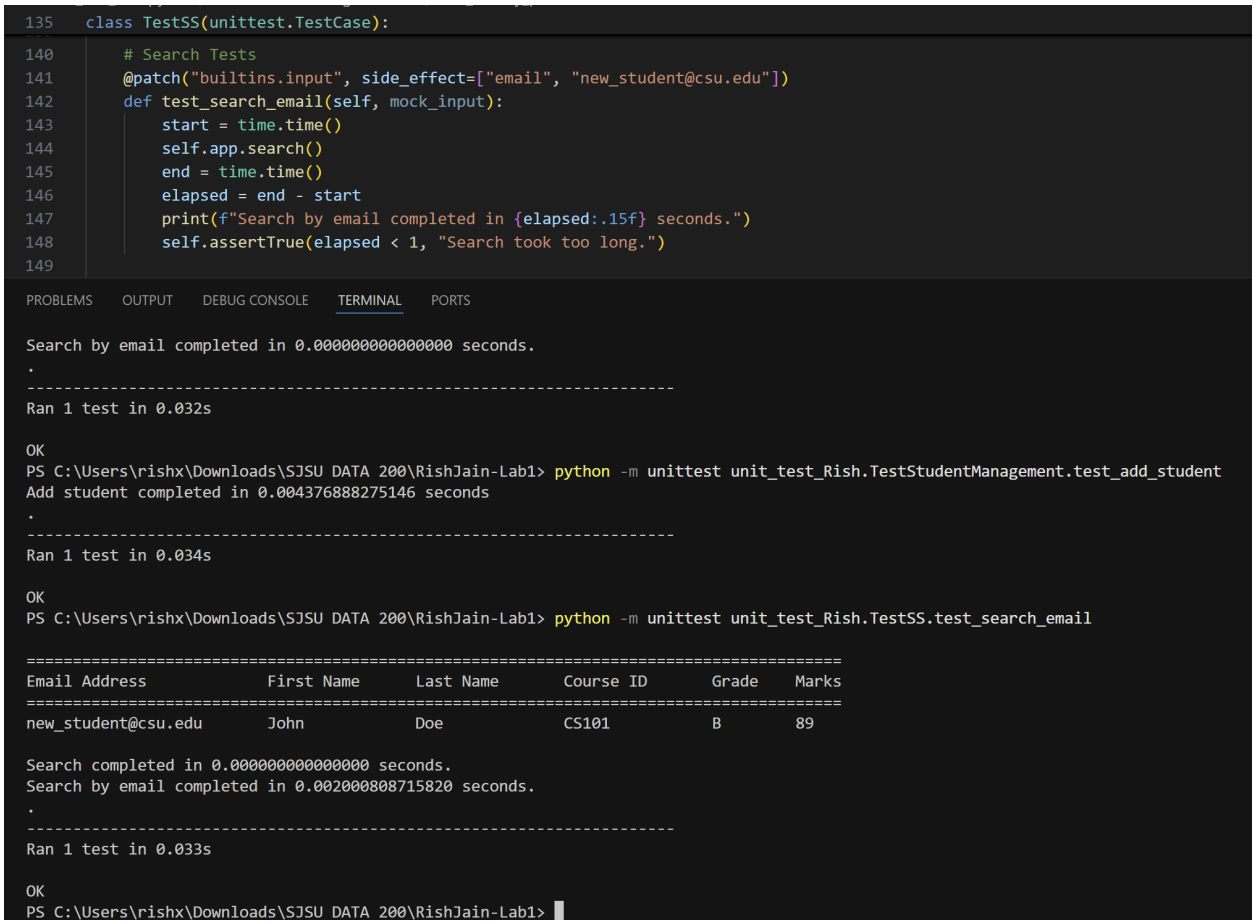
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
----------------------------------------------------------------------
Ran 1 test in 0.039s

OK
PS C:\Users\rishx\Downloads\SJSU DATA 200\RishJain-Lab1> python -m unittest unit_test_Rish.TestProfessorManagement.test_modify_professor
No records found.
Modify professor completed in 0.001198291778564 seconds
.
----------------------------------------------------------------------
Ran 1 test in 0.032s

OK
PS C:\Users\rishx\Downloads\SJSU DATA 200\RishJain-Lab1> python -m unittest unit_test_Rish.TestProfessorManagement.test_modify_professor
Modify professor completed in 0.001096725463867 seconds
.
----------------------------------------------------------------------
Ran 1 test in 0.031s

OK
PS C:\Users\rishx\Downloads\SJSU DATA 200\RishJain-Lab1> python -m unittest unit_test_Rish.TestProfessorManagement.test_delete_professor
Professor deleted successfully
Delete professor completed in 0.002103567123413 seconds
.
----------------------------------------------------------------------
Ran 1 test in 0.032s

OK
PS C:\Users\rishx\Downloads\SJSU DATA 200\RishJain-Lab1>
```

professor.csv > data

```
1   professor_id,professor_name,rank,course_id
2
```

Search student by email: This unit test uses mock input (since my program is interactive) to manually search a student email in both the StudentLinkedList and student.csv file using the search() function. It also prints the timing.



```
class TestSS(unittest.TestCase):

    # Search Tests
    @patch("builtins.input", side_effect=["email", "new_student@csu.edu"])
    def test_search_email(self, mock_input):
        start = time.time()
        self.app.search()
        end = time.time()
        elapsed = end - start
        print(f"Search by email completed in {elapsed:.15f} seconds.")
        self.assertTrue(elapsed < 1, "Search took too long.")

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

Search by email completed in 0.000000000000000 seconds.
.
----------------------------------------------------------------------
Ran 1 test in 0.032s

OK
PS C:\Users\rishx\Downloads\SJSU DATA 200\RishJain-Lab1> python -m unittest unit_test_Rish.TestStudentManagement.test_add_student
Add student completed in 0.004376888275146 seconds
.
----------------------------------------------------------------------
Ran 1 test in 0.034s

OK
PS C:\Users\rishx\Downloads\SJSU DATA 200\RishJain-Lab1> python -m unittest unit_test_Rish.TestSS.test_search_email

================================================================================
Email Address             First Name      Last Name       Course ID       Grade    Marks
================================================================================
new_student@csu.edu       John            Doe             CS101           B        89

Search completed in 0.000000000000000 seconds.
Search by email completed in 0.002000808715820 seconds.
.
----------------------------------------------------------------------
Ran 1 test in 0.033s

OK
PS C:\Users\rishx\Downloads\SJSU DATA 200\RishJain-Lab1>
```

Search student by grade: This unit test uses mock input (since my program is interactive) to manually search student grades in both the StudentLinkedList and student.csv file using the search() function. It also prints the timing.

```
135  class TestSS(unittest.TestCase):
150      @patch("builtins.input", side_effect=["grade", "A"])
151      def test_search_grade(self, mock_input):
152          start = time.time()
153          self.app.search()
154          end = time.time()
155          elapsed = end - start
156          print(f"Search by grade completed in {elapsed:.15f} seconds")
157          self.assertTrue(elapsed < 1, "Search took too long")
158
159      # Sort Tests
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

Search by email completed in 0.000000000000000 seconds.
.
----------------------------------------------------------------------
Ran 1 test in 0.032s

OK
PS C:\Users\rishx\Downloads\SJSU DATA 200\RishJain-Lab1> python -m unittest unit_test_Rish.TestStudentManagement.test_add_student
Add student completed in 0.004376888275146 seconds
.
----------------------------------------------------------------------
Ran 1 test in 0.034s

OK
PS C:\Users\rishx\Downloads\SJSU DATA 200\RishJain-Lab1> python -m unittest unit_test_Rish.TestSS.test_search_email

================================================================================
Email Address             First Name      Last Name       Course ID       Grade    Marks
================================================================================
new_student@csu.edu       John            Doe             CS101           B        89

Search completed in 0.000000000000000 seconds.
Search by email completed in 0.002000808715820 seconds.
.
----------------------------------------------------------------------
Ran 1 test in 0.033s

OK
PS C:\Users\rishx\Downloads\SJSU DATA 200\RishJain-Lab1> python -m unittest unit_test_Rish.TestSS.test_search_grade
```

```
silvia.calcote837@univers Silvia          Calcote         CS108           A        92
daniel.rock863@university Daniel          Rock            CS104           A        99
steven.ritter869@universi Steven          Ritter          CS110           A        90
harry.baldwin871@universi Harry           Baldwin         CS102           A        95
randolph.smith877@univers Randolph        Smith           CS108           A        100
frank.heston878@universit Frank           Heston          CS109           A        93
marilyn.kendall887@univer Marilyn         Kendall         CS108           A        100
ronald.micciche894@univer Ronald          Micciche        CS105           A        100
rubin.turner895@universit Rubin           Turner          CS106           A        91
eric.vazquez902@universit Eric            Vazquez         CS103           A        99
kara.smith919@university. Kara            Smith           CS110           A        98
yvonne.salley930@universi Yvonne          Salley          CS101           A        94
luis.bonner938@university Luis            Bonner          CS109           A        95
linda.ruiz942@university. Linda           Ruiz            CS103           A        99
hope.beller947@university Hope            Beller          CS108           A        96
kurt.brinlee962@universit Kurt            Brinlee         CS103           A        95
salvador.koehler983@unive Salvador        Koehler         CS104           A        99
jeffrey.neaves990@univers Jeffrey          Neaves          CS101           A        92
edwin.tong991@university. Edwin            Tong            CS102           A        92
nicole.malone997@universi Nicole           Malone          CS108           A        98
jacob.britton999@universi Jacob            Britton         CS110           A        92

Search completed in 0.000000000000000 seconds.
Search by grade completed in 0.019382476806641 seconds
.
----------------------------------------------------------------------
Ran 1 test in 0.051s
```

Sort student by email: This unit test uses mock input (since my program is interactive) to manually sort both the StudentLinkedList and student.csv file by email using the sorting() function. It also prints the timing.

```
135  class TestSS(unittest.TestCase):
158
159      @patch("builtins.input", side_effect=["email", "false"])
160      def test_sort_email(self, mock_input):
161          start = time.time()
162          self.app.sorting()
163          end = time.time()
164          elapsed = end - start
165          print(f"Sort by email completed in {elapsed:.15f} seconds")
166          self.assertTrue(elapsed < 2, "Sort took too long")
167
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

luis.bonner938@university Luis          Bonner        CS109        A      95
linda.ruiz942@university. Linda         Ruiz          CS103        A      99
hope.beller947@university Hope          Beller        CS108        A      96
kurt.brinlee962@universit Kurt          Brinlee       CS103        A      95
salvador.koehler983@unive Salvador      Koehler       CS104        A      99
jeffrey.neaves990@univers Jeffrey       Neaves        CS101        A      92
edwin.tong991@university. Edwin         Tong          CS102        A      92
nicole.malone997@universi Nicole        Malone        CS108        A      98
jacob.britton999@universi Jacob         Britton       CS110        A      92

Search completed in 0.000000000000000 seconds.
Search by grade completed in 0.019382476806641 seconds
.
----------------------------------------------------------------------
Ran 1 test in 0.051s

OK
PS C:\Users\rishx\Downloads\SJSU DATA 200\RishJain-Lab1> python -m unittest unit_test_Rish.TestSS.test_sort_email

Sorting completed in 0.000000000000000 seconds
Sorted by email successfully
Sort by email completed in 0.024667024612427 seconds
.
----------------------------------------------------------------------
Ran 1 test in 0.059s

OK
PS C:\Users\rishx\Downloads\SJSU DATA 200\RishJain-Lab1>
```

student.csv > data

```
1   email_address,first_name,last_name,course_id,grades,marks
2   aaron.perkins477@university.edu,Aaron,Perkins,CS108,F,54
3   adam.pauley241@university.edu,Adam,Pauley,CS102,F,43
4   adele.bobo243@university.edu,Adele,Bobo,CS104,F,31
5   alan.fraser101@university.edu,Alan,Fraser,CS102,C,73
6   alice.bradley646@university.edu,Alice,Bradley,CS107,F,30
7   alice.welch537@university.edu,Alice,Welch,CS108,F,36
8   alisa.galvez840@university.edu,Alisa,Galvez,CS101,B,84
9   alison.patterson813@university.edu,Alison,Patterson,CS104,A,95
10  allen.binder733@university.edu,Allen,Binder,CS104,A,93
11  alvin.crespino977@university.edu,Alvin,Crespino,CS108,F,58
```

Sort student by marks: This unit test uses mock input (since my program is interactive) to manually sort both the StudentLinkedList and student.csv file by marks using the sorting() function. It also prints the timing.

```
135  class TestSS(unittest.TestCase):
167
168      @patch("builtins.input", side_effect=["marks", "True"])
169      def test_sort_marks(self, mock_input):
170          start = time.time()
171          self.app.sorting()
172          end = time.time()
173          elapsed = end - start
174          print(f"Sort by marks completed in {elapsed:.15f} seconds")
175          self.assertTrue(elapsed < 2, "Sort took too long")
176
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
Search completed in 0.000000000000000 seconds.
Search by grade completed in 0.019382476806641 seconds
.
----------------------------------------------------------------------
Ran 1 test in 0.051s

OK
PS C:\Users\rishx\Downloads\SJSU DATA 200\RishJain-Lab1> python -m unittest unit_test_Rish.TestSS.test_sort_email

Sorting completed in 0.000000000000000 seconds
Sorted by email successfully
Sort by email completed in 0.024667024612427 seconds
.
----------------------------------------------------------------------
Ran 1 test in 0.059s

OK
PS C:\Users\rishx\Downloads\SJSU DATA 200\RishJain-Lab1> python -m unittest unit_test_Rish.TestSS.test_sort_marks

Sorting completed in 0.000999927520752 seconds.
Sorted by marks successfully
Sort by marks completed in 0.019810914993286 seconds
.
----------------------------------------------------------------------
Ran 1 test in 0.050s

OK
```

student.csv > data

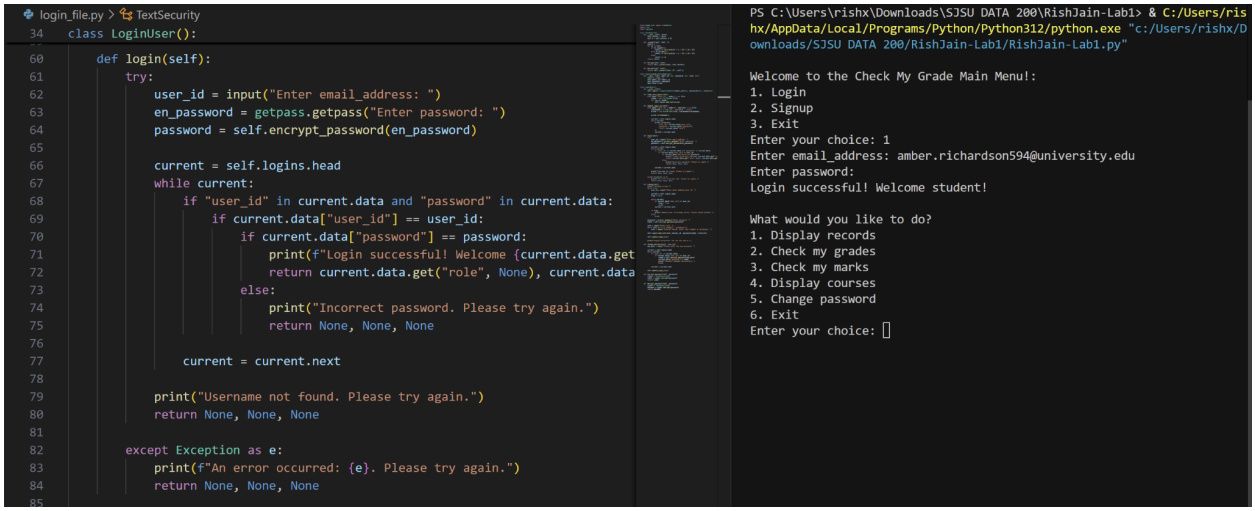
```
1   email_address,first_name,last_name,course_id,grades,marks
2   amber.richardson594@university.edu,Amber,Richardson,CS105,A,100
3   amy.gouge12@university.edu,Amy,Gouge,CS103,A,100
4   anthony.nickell513@university.edu,Anthony,Nickell,CS104,A,100
5   carolina.vanek531@university.edu,Carolina,Vanek,CS102,A,100
6   erica.mcilwain521@university.edu,Erica,Mcilwain,CS102,A,100
7   evelyn.wicker782@university.edu,Evelyn,Wicker,CS103,A,100
8   linda.abraham7@university.edu,Linda,Abraham,CS108,A,100
9   maria.bissett354@university.edu,Maria,Bissett,CS105,A,100
10  marilyn.kendall887@university.edu,Marilyn,Kendall,CS108,A,100
11  phyllis.gottfried475@university.edu,Phyllis,Gottfried,CS106,A,100
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

Lab 1 Screenshots

Note: Each add/modify/delete edits the subsequent linked list and updates the csv file overall (Student/Course/Professor/GradeLinkedList and the student/course/professor/grade.csv).

Login: This function checks if there is an email address and password in the login.csv file and if so, it takes the password entered. That entered password becomes encrypted into a variable and that variable is compared to the encrypted password in the csv file. If they match, then the login is successful and says: Welcome, “role”! whether it’s student or professor. The function returns the role, email, and password because role is used to determine different accesses and certain emails should only be able to see their own records.



```
login_file.py > TextSecurity
34  class LoginUser():
60      def login(self):
61          try:
62              user_id = input("Enter email_address: ")
63              en_password = getpass.getpass("Enter password: ")
64              password = self.encrypt_password(en_password)
65
66              current = self.logins.head
67              while current:
68                  if "user_id" in current.data and "password" in current.data:
69                      if current.data["user_id"] == user_id:
70                          if current.data["password"] == password:
71                              print(f"Login successful! Welcome {current.data.get
72                              return current.data.get("role", None), current.data
73                          else:
74                              print("Incorrect password. Please try again.")
75                              return None, None, None
76
77                  current = current.next
78
79              print("Username not found. Please try again.")
80              return None, None, None
81
82          except Exception as e:
83              print(f"An error occurred: {e}. Please try again.")
84              return None, None, None
85
```

```
PS C:\Users\rishx\Downloads\SJSU DATA 200\RishJain-Lab1> & C:/Users/ris
hx/AppData/Local/Programs/Python/Python312/python.exe "c:/Users/rishx/D
ownloads/SJSU DATA 200/RishJain-Lab1/RishJain-Lab1.py"

Welcome to the Check My Grade Main Menu!:
1. Login
2. Signup
3. Exit
Enter your choice: 1
Enter email_address: amber.richardson594@university.edu
Enter password:
Login successful! Welcome student!

What would you like to do?
1. Display records
2. Check my grades
3. Check my marks
4. Display courses
5. Change password
6. Exit
Enter your choice:
```

Signup: This function allows a user to put data in the login.csv file which would contain email address, the password (the function stores the encrypted password in the file), and the role which must be either student or professor.

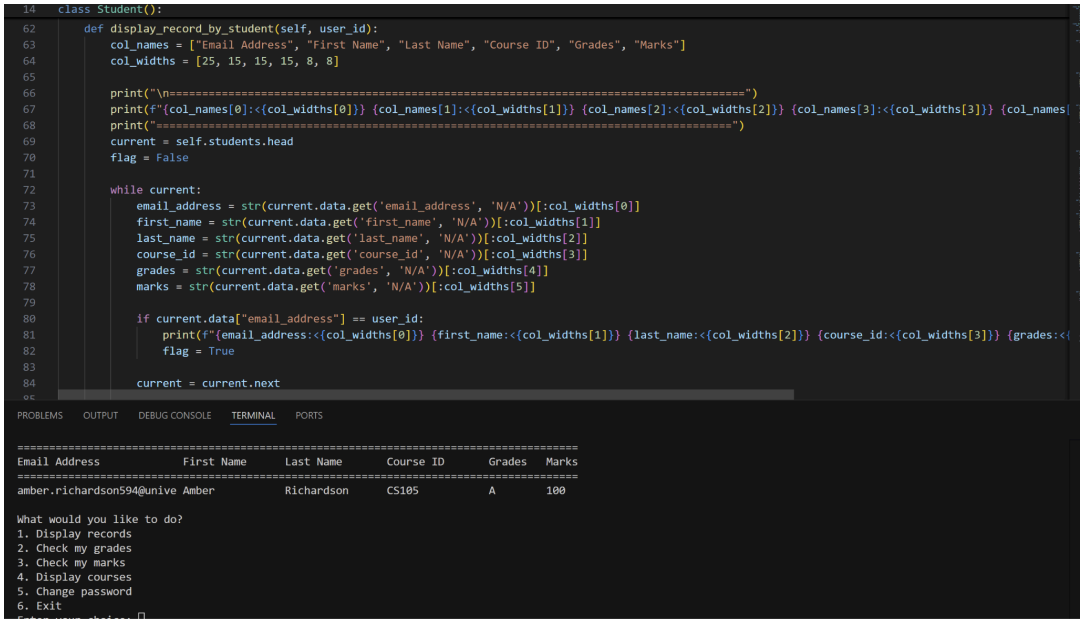
```
34  class LoginUser():
86      def signup(self):
87          print("\nSignup screen:")
88          while True:
89              user_id = input("Enter email address/user id: ")
90
91              current = self.logins.head
92              flag = False
93
94              while current:
95                  if current.data["user_id"] == user_id:
96                      flag = True
97                      break
98                  current = current.next
99
100             if flag:
101                 print("\nEmail/user id already exists. Please choose another.")
102             else:
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

Welcome to the Check My Grade Main Menu!:
1. Login
2. Signup
3. Exit
Enter your choice: 2

Signup screen:
Enter email address/user id: testprof@mycsu.edu
Enter password:
Enter role: professor
Signup successful! You can now log in.

Welcome to the Check My Grade Main Menu!:
1. Login
2. Signup
3. Exit
Enter your choice: 1
Enter email_address: testprof@mycsu.edu
Enter password:
Login successful! Welcome professor!
```

```
34  class LoginUser():
86      def signup(self):
102             else:
103                 break
104
105         password = getpass.getpass("Enter password: ")
106         coded = self.encrypt_password(password)
107
108         role = input("Enter role: ")
109         while role not in ["student", "professor"]:
110             role = input("Invalid choice. Enter role student or professor: ")
111
112         self.logins.add_last(user_id=user_id, password=coded, role=role)
113
114         self.update_login_csv()
115
116         print("Signup successful! You can now log in.")
117
```

Student: The student role should only be able to view their own records ie grades, marks, email, course, etc. The student should also be able to change their own password and view all courses and their descriptions.

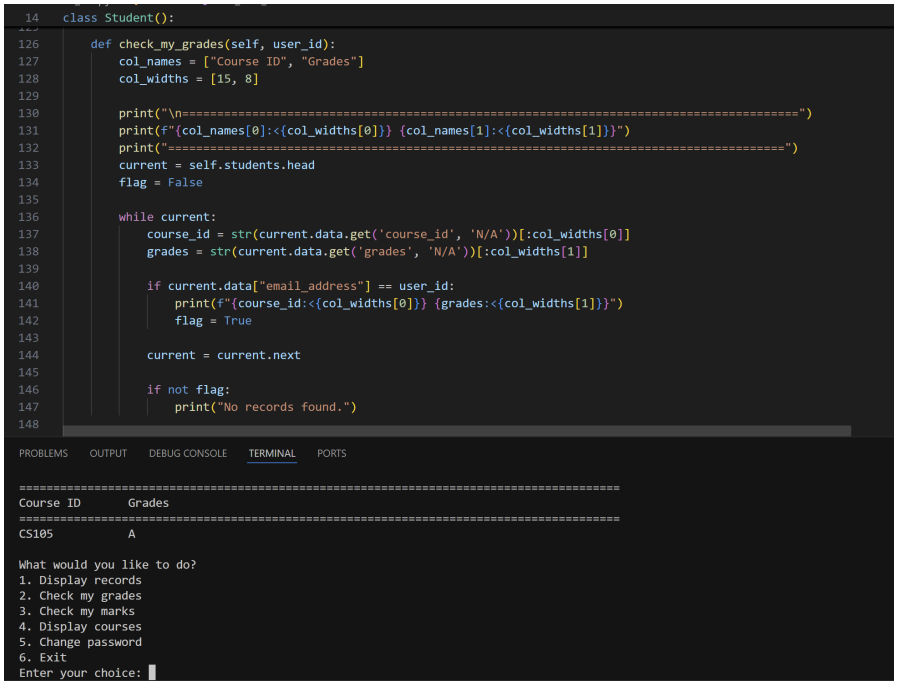
Display own records: This function displays the student's own records from the student.csv based on searching for the email address logged in.



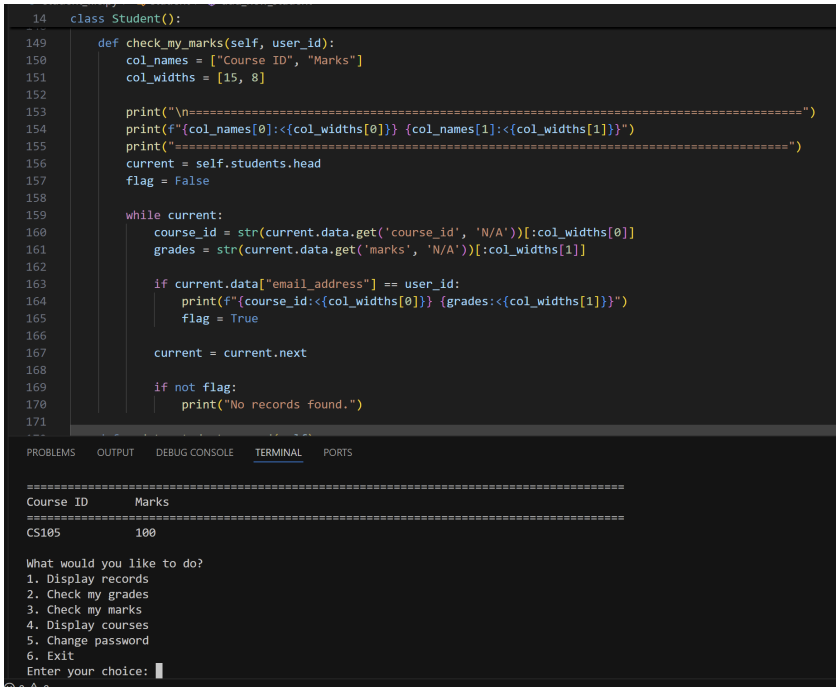
```
class Student():
    def display_record_by_student(self, user_id):
        col_names = ["Email Address", "First Name", "Last Name", "Course ID", "Grades", "Marks"]
        col_widths = [25, 15, 15, 15, 8, 8]

        print("\n" + "=" * 100)
        print(f"{col_names[0]:<{col_widths[0]}} {col_names[1]:<{col_widths[1]}} {col_names[2]:<{col_widths[2]}} {col_names[3]:<{col_widths[3]}} {col_names[
        print("=" * 100)
        current = self.students.head
        flag = False

        while current:
            email_address = str(current.data.get('email_address', 'N/A'))[:col_widths[0]]
            first_name = str(current.data.get('first_name', 'N/A'))[:col_widths[1]]
            last_name = str(current.data.get('last_name', 'N/A'))[:col_widths[2]]
            course_id = str(current.data.get('course_id', 'N/A'))[:col_widths[3]]
            grades = str(current.data.get('grades', 'N/A'))[:col_widths[4]]
            marks = str(current.data.get('marks', 'N/A'))[:col_widths[5]]

            if current.data["email_address"] == user_id:
                print(f"{email_address:<{col_widths[0]}} {first_name:<{col_widths[1]}} {last_name:<{col_widths[2]}} {course_id:<{col_widths[3]}} {grades:<{
                flag = True

            current = current.next
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

====================================================================================
Email Address             First Name      Last Name       Course ID       Grades   Marks
====================================================================================
amber.richardson594@unive Amber           Richardson      CS105           A        100

What would you like to do?
1. Display records
2. Check my grades
3. Check my marks
4. Display courses
5. Change password
6. Exit
```

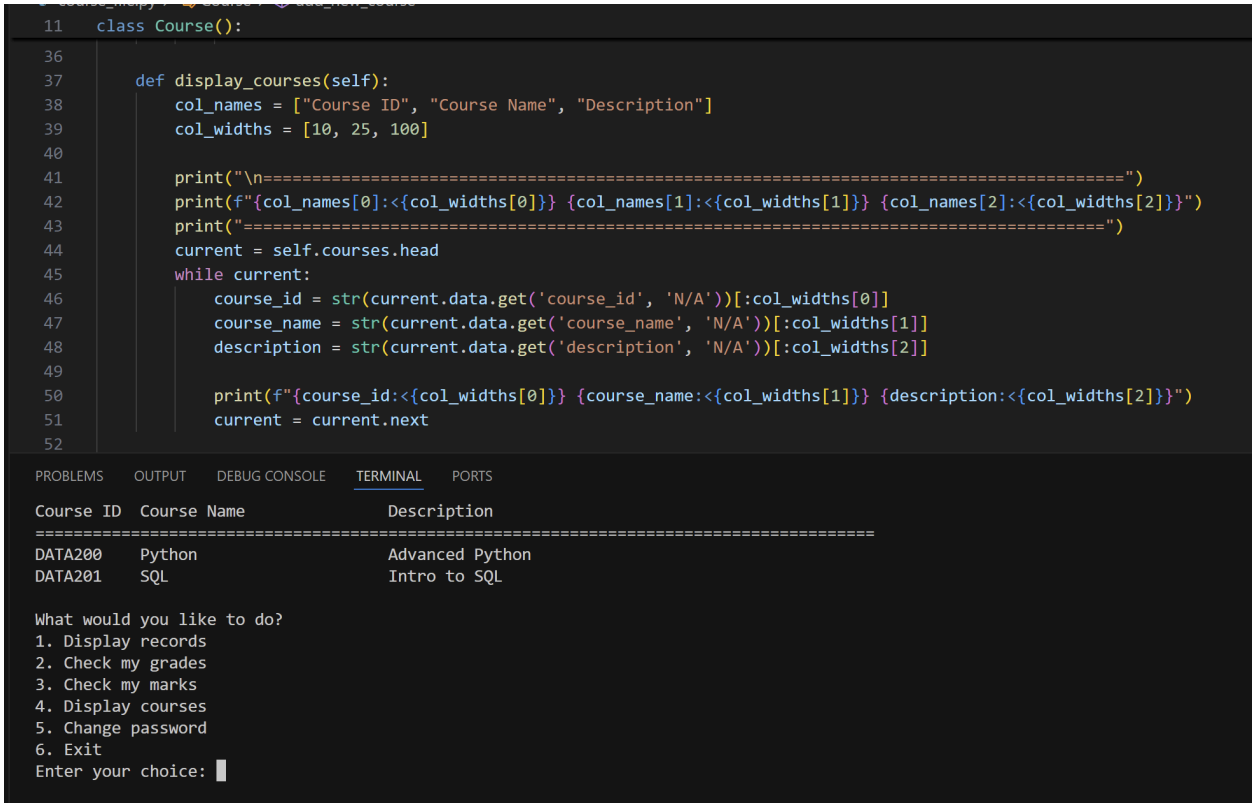
Display own grades: This function displays the student's own grades from student.csv based on searching for the email address logged in.



Display own marks: This function displays the student's own marks from student.csv based on searching for the email address logged in.



Display courses: This function displays all the courses in the course.csv file using the CourseLinkedList.



```
class Course():

    def display_courses(self):
        col_names = ["Course ID", "Course Name", "Description"]
        col_widths = [10, 25, 100]

        print("\n" + "=" * 140)
        print(f"{col_names[0]:<{col_widths[0]}} {col_names[1]:<{col_widths[1]}} {col_names[2]:<{col_widths[2]}}")
        print("=" * 140)
        current = self.courses.head
        while current:
            course_id = str(current.data.get('course_id', 'N/A'))[:col_widths[0]]
            course_name = str(current.data.get('course_name', 'N/A'))[:col_widths[1]]
            description = str(current.data.get('description', 'N/A'))[:col_widths[2]]

            print(f"{course_id:<{col_widths[0]}} {course_name:<{col_widths[1]}} {description:<{col_widths[2]}}")
            current = current.next
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

Course ID  Course Name               Description
============================================================================================
DATA200    Python                    Advanced Python
DATA201    SQL                       Intro to SQL

What would you like to do?
1. Display records
2. Check my grades
3. Check my marks
4. Display courses
5. Change password
6. Exit
Enter your choice:
```

Change password: This function changes the password in the login.csv file of the user id/email that is logged in. It takes the password entered and stores the encrypted version in the file.

```
 34  class LoginUser():
117
118      def change_password(self, user_id):
119          new_pass = input("Please enter the new password: ")
120
121          current = self.logins.head
122          while current:
123              if "user_id" in current.data:
124                  if current.data["user_id"] == user_id:
125                      coded = self.encrypt_password(new_pass)
126                      current.data["password"] = coded
127                      print("Password changed successfully.")
128                      break
129
130              current = current.next
131
132          self.update_login_csv()
133
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
DATA201    SQL                          Intro to SQL

What would you like to do?
1. Display records
2. Check my grades
3. Check my marks
4. Display courses
5. Change password
6. Exit
Enter your choice: 5
Please enter the new password: welcome1
Password changed successfully.

What would you like to do?
1. Display records
2. Check my grades
3. Check my marks
4. Display courses
5. Change password
6. Exit
Enter your choice:
```

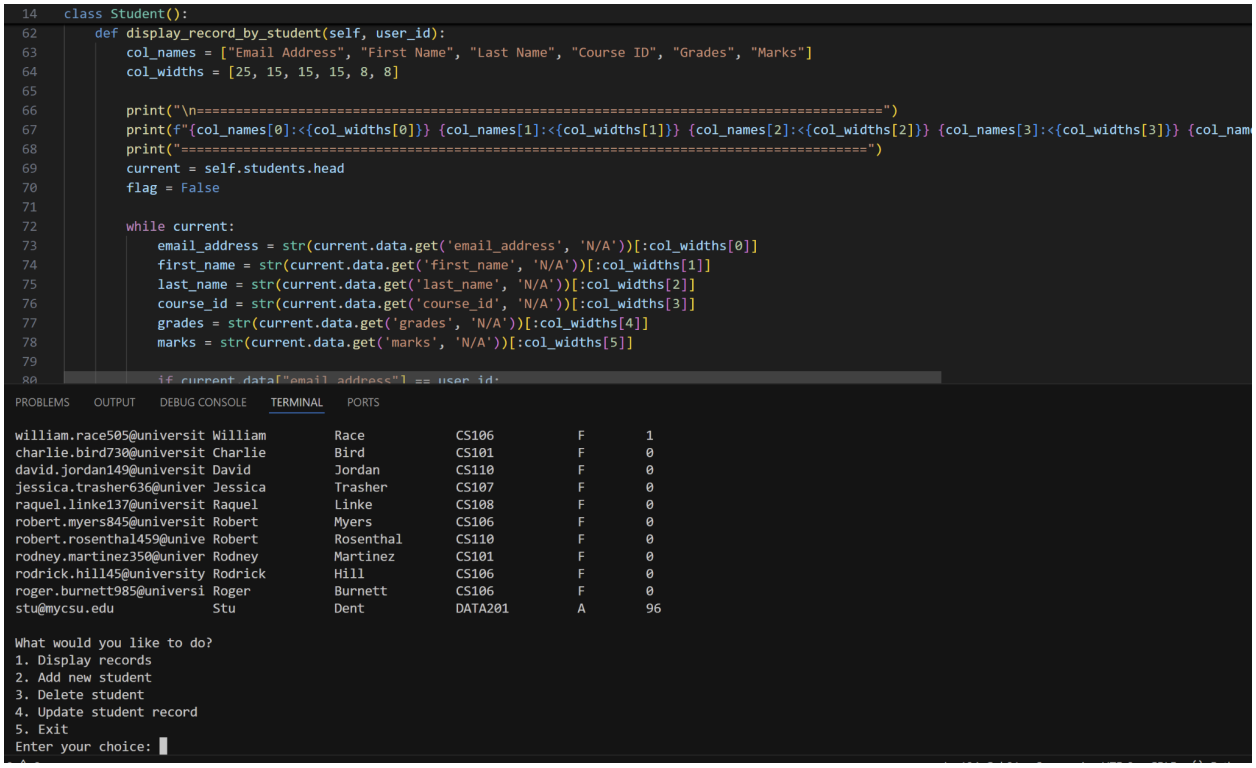
Professor: The professor should be able to have access to everything ie change grades, marks, courses, professors, etc.

Add student: This function asks for a new email address and first checks if that student already exists. If it does, then nothing will happen. If it doesn't, then the user will be prompted to enter first name, last name, course id, and marks. The grade A-F will be set based on the marks which has to be an integer.

```
14  class Student():
89      def add_new_student(self):
90          new_email_address = input("Please enter the new email address: ")
91          current = self.students.head
92
93          while current:
94              if current.data["email_address"] == new_email_address:
95                  print("Student already exists")
96                  return
97              current = current.next
98
99          first_name = input("Please enter the new first name: ")
100         last_name = input("Please enter the new last name: ")
101         course_id = input("Please enter new course id: ")
102         while True:
103             try:
104                 marks = int(input("Please enter the new marks (0-100): "))
105                 if 0 <= marks <= 100:
106                     break
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

6. CHECK MY GRADE APP
7. Exit
Enter your choice: 1

What would you like to do?
1. Display records
2. Add new student
3. Delete student
4. Update student record
5. Exit
Enter your choice: 2
Please enter the new email address: stu@mycsu.edu
Please enter the new first name: Stu
Please enter the new last name: Dent
Please enter new course id: DATA201
Please enter the new marks (0-100): r
Invalid input
Please enter the new marks (0-100): 96
```

Display all records: This function displays all records of all students in the student.csv file in an organized manner with a table.

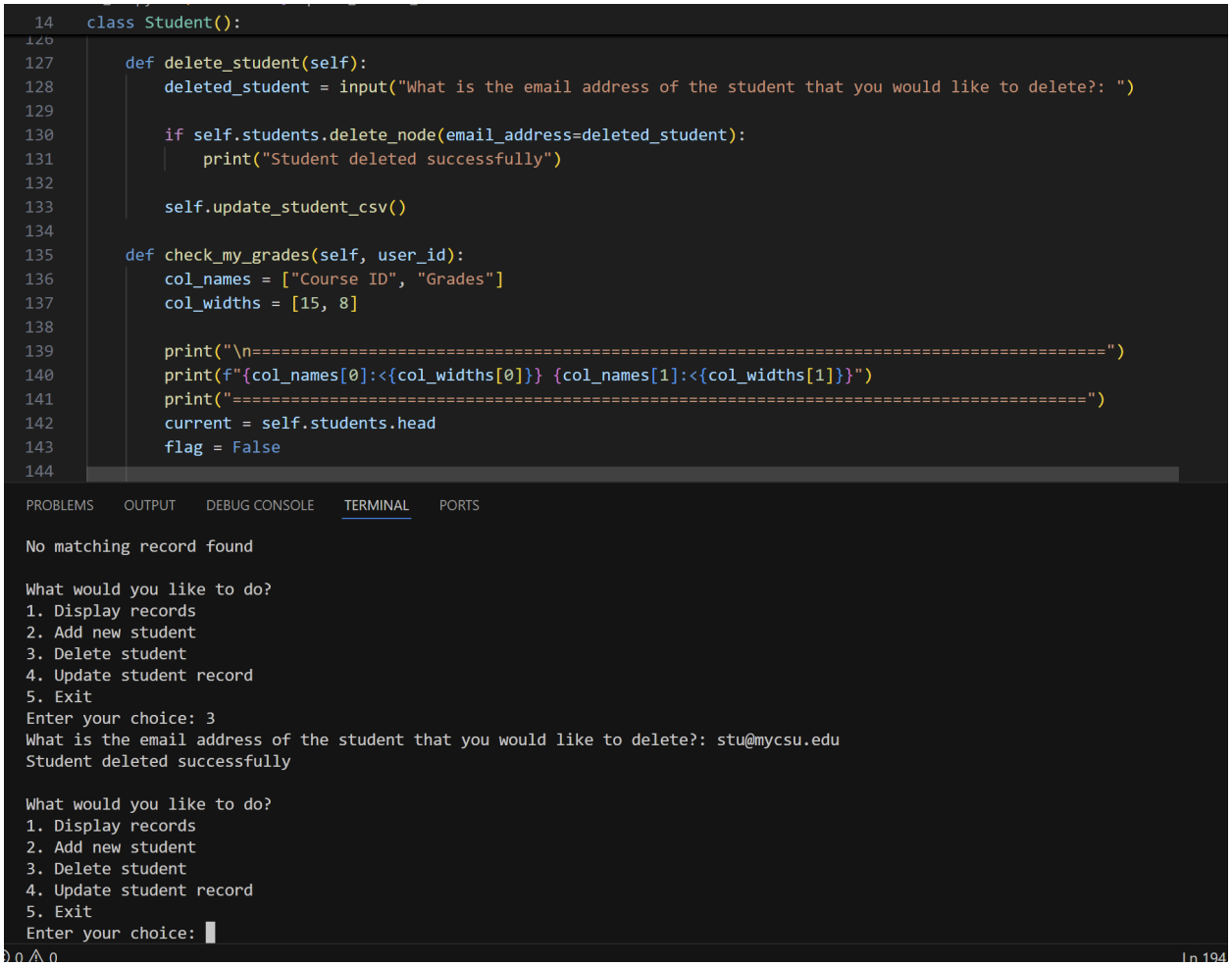


```
class Student():
    def display_record_by_student(self, user_id):
        col_names = ["Email Address", "First Name", "Last Name", "Course ID", "Grades", "Marks"]
        col_widths = [25, 15, 15, 15, 8, 8]

        print("\n================================================================================")
        print(f"{col_names[0]:<{col_widths[0]}} {col_names[1]:<{col_widths[1]}} {col_names[2]:<{col_widths[2]}} {col_names[3]:<{col_widths[3]}} {col_nam
        print("================================================================================")
        current = self.students.head
        flag = False

        while current:
            email_address = str(current.data.get('email_address', 'N/A'))[:col_widths[0]]
            first_name = str(current.data.get('first_name', 'N/A'))[:col_widths[1]]
            last_name = str(current.data.get('last_name', 'N/A'))[:col_widths[2]]
            course_id = str(current.data.get('course_id', 'N/A'))[:col_widths[3]]
            grades = str(current.data.get('grades', 'N/A'))[:col_widths[4]]
            marks = str(current.data.get('marks', 'N/A'))[:col_widths[5]]

            if current.data["email_address"] == user_id:
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
william.race505@universit William        Race          CS106         F       1
charlie.bird730@universit Charlie         Bird          CS101         F       0
david.jordan149@universit David           Jordan        CS110         F       0
jessica.trasher636@univer Jessica         Trasher       CS107         F       0
raquel.linke137@universit Raquel          Linke         CS108         F       0
robert.myers845@universit Robert          Myers         CS106         F       0
robert.rosenthal459@unive Robert          Rosenthal     CS110         F       0
rodney.martinez350@univer Rodney          Martinez      CS101         F       0
rodrick.hill45@university Rodrick         Hill          CS106         F       0
roger.burnett985@universi Roger           Burnett       CS106         F       0
stu@mycsu.edu             Stu             Dent          DATA201       A       96

What would you like to do?
1. Display records
2. Add new student
3. Delete student
4. Update student record
5. Exit
Enter your choice:
```

Delete student: This function asks for the email address and deletes the email address if found from both the linked list and csv.



```
14  class Student():
126
127     def delete_student(self):
128         deleted_student = input("What is the email address of the student that you would like to delete?: ")
129
130         if self.students.delete_node(email_address=deleted_student):
131             print("Student deleted successfully")
132
133         self.update_student_csv()
134
135     def check_my_grades(self, user_id):
136         col_names = ["Course ID", "Grades"]
137         col_widths = [15, 8]
138
139         print("\n================================================================================")
140         print(f"{col_names[0]:<{col_widths[0]}} {col_names[1]:<{col_widths[1]}}")
141         print("================================================================================")
142         current = self.students.head
143         flag = False
144
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
No matching record found

What would you like to do?
1. Display records
2. Add new student
3. Delete student
4. Update student record
5. Exit
Enter your choice: 3
What is the email address of the student that you would like to delete?: stu@mycsu.edu
Student deleted successfully

What would you like to do?
1. Display records
2. Add new student
3. Delete student
4. Update student record
5. Exit
Enter your choice:
```

0 0 ⚠ 0 Ln 194

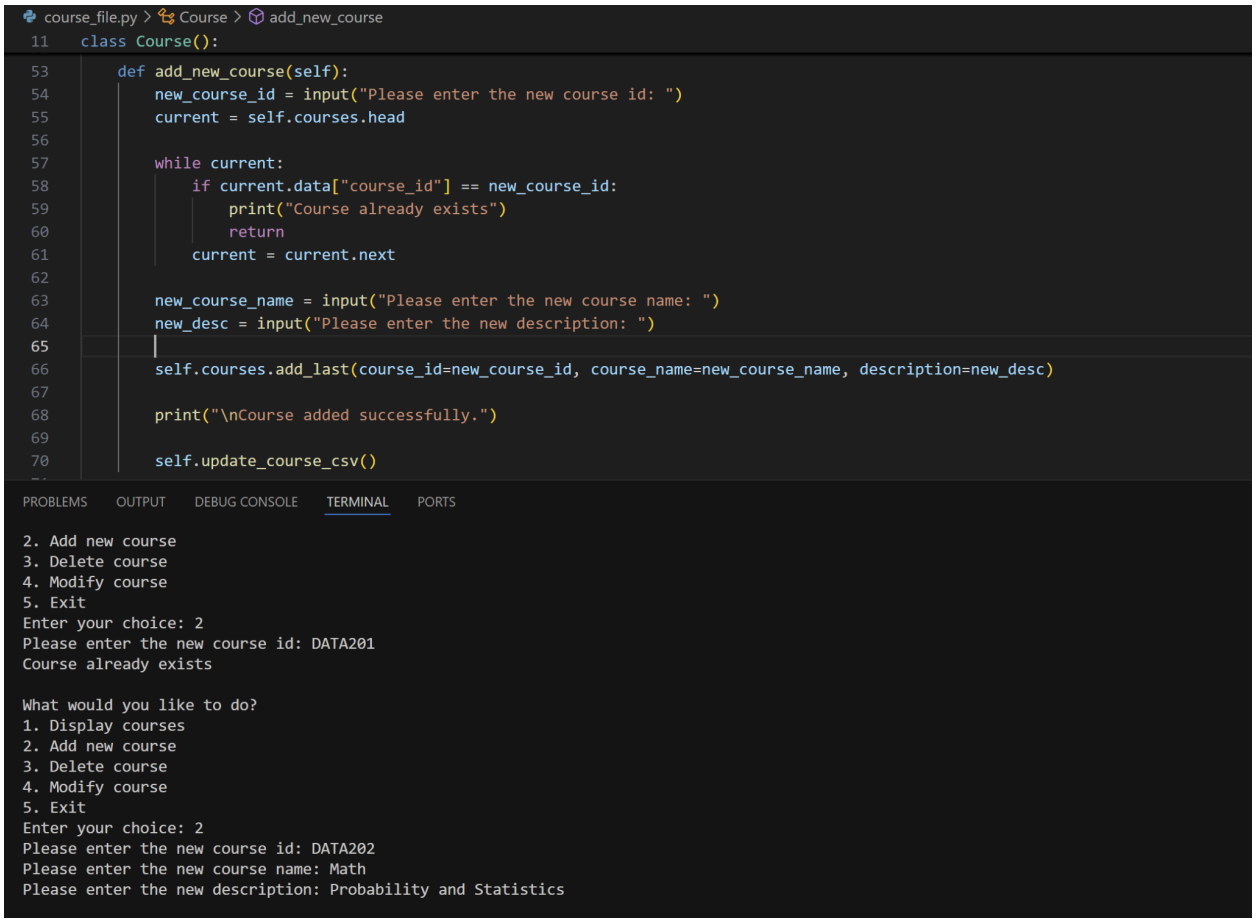
Update student record: This function asks for the email address of the student. If it isn't found, nothing happens but if the email is found then the user can update everything including the email and the linked list + student.csv file gets updated.

```
14  class Student():
181     def update_student_record(self):
182         user_id = input("Please enter the email address of the student that you want to update: ")
183         flag = False
184
185         current = self.students.head
186         while current:
187             if "email_address" in current.data and current.data["email_address"] == user_id:
188                     current.data["email_address"] = input("Please enter the updated email address: ")
189                     current.data["first_name"] = input("Please enter the updated first name: ")
190                     current.data["last_name"] = input("Please enter the updated last name: ")
191                     current.data["course_id"] = input("Please enter the updated course id: ")
192                     while True:
193                         try:
194                             current.data["marks"] = int(input("Please enter the new marks (0-100): "))
195                             if 0 <= current.data["marks"] <= 100:
196                                 break
197                             else:
198                                 print("Marks must be between 0 and 100")
199                         except ValueError:
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
What is the email address of the student that you would like to delete?: stu@mycsu.edu
Student deleted successfully

What would you like to do?
1. Display records
2. Add new student
3. Delete student
4. Update student record
5. Exit
Enter your choice: 4
Please enter the email address of the student that you want to update: amber.richardson594@university.edu
Please enter the updated email address: amber.richardson@csu.edu
Please enter the updated first name: Amber
Please enter the updated last name: Richardson
Please enter the updated course id: DATA202
Please enter the new marks (0-100): i
Invalid input
Please enter the new marks (0-100): 100
```

Add new course: This function asks for a new course id to add in the course.csv file. If the course id already exists then nothing happens. If the course id doesn't exist then the user is prompted to enter a course name and description which gets added to the CourseLinkedList then the course.csv file.

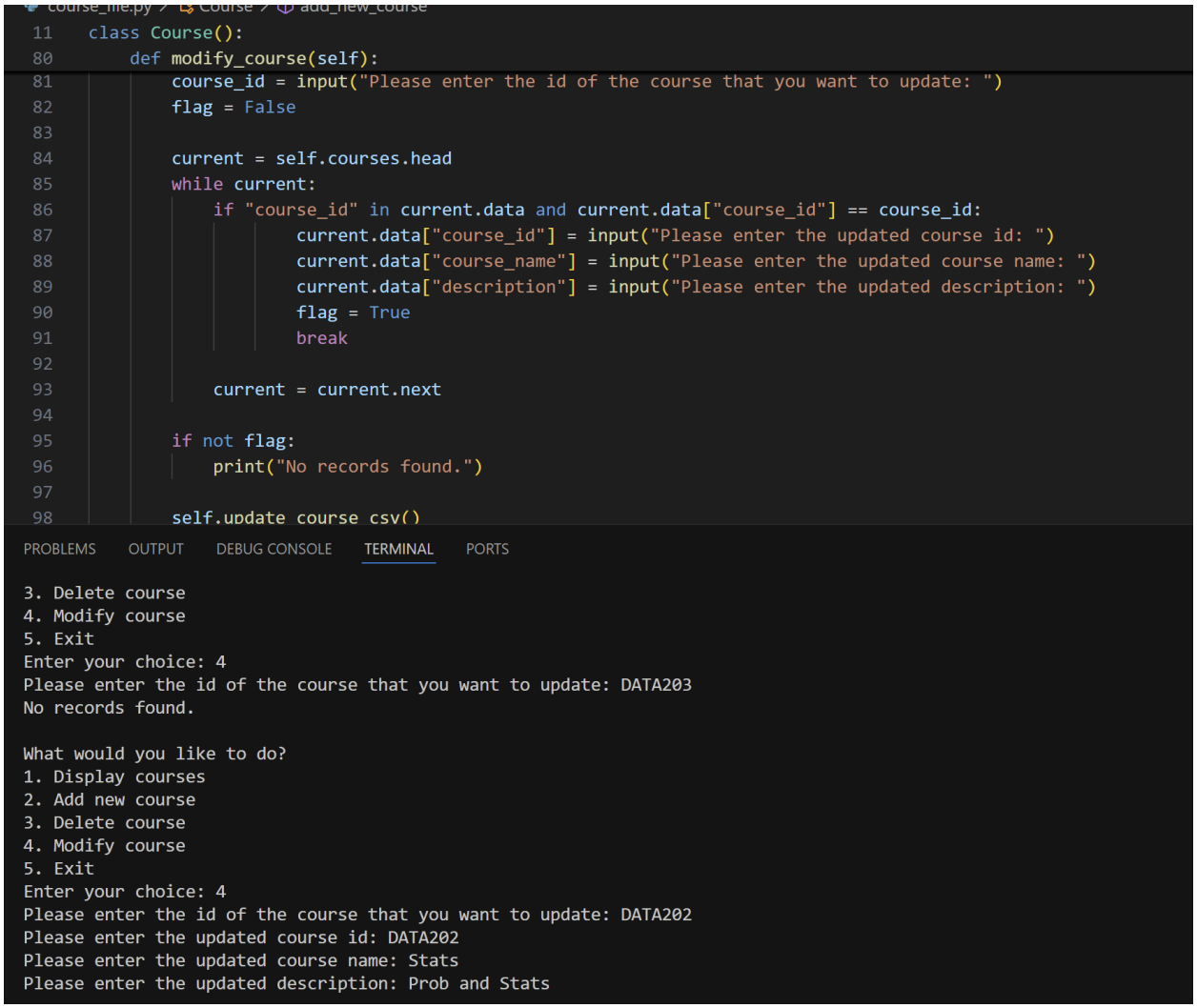


```
course_file.py > Course > add_new_course
11  class Course():
53      def add_new_course(self):
54          new_course_id = input("Please enter the new course id: ")
55          current = self.courses.head
56
57          while current:
58              if current.data["course_id"] == new_course_id:
59                  print("Course already exists")
60                  return
61              current = current.next
62
63          new_course_name = input("Please enter the new course name: ")
64          new_desc = input("Please enter the new description: ")
65
66          self.courses.add_last(course_id=new_course_id, course_name=new_course_name, description=new_desc)
67
68          print("\nCourse added successfully.")
69
70          self.update_course_csv()
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
2. Add new course
3. Delete course
4. Modify course
5. Exit
Enter your choice: 2
Please enter the new course id: DATA201
Course already exists

What would you like to do?
1. Display courses
2. Add new course
3. Delete course
4. Modify course
5. Exit
Enter your choice: 2
Please enter the new course id: DATA202
Please enter the new course name: Math
Please enter the new description: Probability and Statistics
```

Modify course: This function asks the user for the course id that they want to update. If the course isn't found, then nothing happens but if the course is found the course id, course name, and description can be updated in both the CourseLinkedList and the course.csv file.



```
course_file.py > Course > add_new_course
11  class Course():
80      def modify_course(self):
81          course_id = input("Please enter the id of the course that you want to update: ")
82          flag = False
83
84          current = self.courses.head
85          while current:
86              if "course_id" in current.data and current.data["course_id"] == course_id:
87                      current.data["course_id"] = input("Please enter the updated course id: ")
88                      current.data["course_name"] = input("Please enter the updated course name: ")
89                      current.data["description"] = input("Please enter the updated description: ")
90                      flag = True
91                      break
92
93              current = current.next
94
95          if not flag:
96              print("No records found.")
97
98          self.update_course_csv()
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
3. Delete course
4. Modify course
5. Exit
Enter your choice: 4
Please enter the id of the course that you want to update: DATA203
No records found.

What would you like to do?
1. Display courses
2. Add new course
3. Delete course
4. Modify course
5. Exit
Enter your choice: 4
Please enter the id of the course that you want to update: DATA202
Please enter the updated course id: DATA202
Please enter the updated course name: Stats
Please enter the updated description: Prob and Stats
```

Delete course: This function asks the user what course id they want to delete. If it's not found, then nothing happens but if it is found then that course id row is deleted from the linked list then the course.csv file.

```
11  class Course():
72      def delete_course(self):
73          deleted_course = input("What is the course id of the course that you would like to delete?: ")
74
75          if self.courses.delete_node(course_id=deleted_course):
76              print("Course deleted successfully")
77
78          self.update_course_csv()
79
80      def modify_course(self):
81          course_id = input("Please enter the id of the course that you want to update: ")
82          flag = False
83
84          current = self.courses.head
85          while current:
86              if "course_id" in current.data and current.data["course_id"] == course_id:
87                      current.data["course_id"] = input("Please enter the updated course id: ")
88                      current.data["course_name"] = input("Please enter the updated course name: ")
89                      current.data["description"] = input("Please enter the updated description: ")
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
1. Display courses
2. Add new course
3. Delete course
4. Modify course
5. Exit
Enter your choice: 3
What is the course id of the course that you would like to delete?: DATA203
No matching record found

What would you like to do?
1. Display courses
2. Add new course
3. Delete course
4. Modify course
5. Exit
Enter your choice: 3
What is the course id of the course that you would like to delete?: DATA202
Course deleted successfully
```

Add professor: This function asks for a new professor id/email to add. If that email is found in the ProfessorLinkedList then nothing happens. If that email isn't found, then the user is prompted to add the professor name, rank, and course which they teach.

```
13  class Professor():
57      def add_new_professor(self):
58          new_prof_id = input("Please enter the new professor id: ")
59          current = self.professors.head
60
61          while current:
62              if current.data["professor_id"] == new_prof_id:
63                  print("Professor already exists")
64                  return
65              current = current.next
66
67          new_prof_name = input("Please enter the new professor name: ")
68          new_rank = input("Please enter the new professor rank: ")
69          new_course_id = input("Please enter the new course id: ")
70
71          self.professors.add_last(professor_id=new_prof_id, professor_name=new_prof_name, rank=new_rank, course_id=new_course_id)
72
73          self.update_professor_csv()
74
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
3. Professors
4. Grades
5. Password
6. CHECK MY GRADE APP
7. Exit
Enter your choice: 3

What would you like to do?
1. Display professor details
2. Add new professor
3. Delete professor
4. Modify professor details
5. Show course details by professor
6. Exit
Enter your choice: 2
Please enter the new professor id: mike@mycsu.edu
Please enter the new professor name: Mike
Please enter the new professor rank: Assistant Professor
Please enter the new course id: DATA202
```

Display professor details: This function displays all the professors in the professor.csv file/ProfessorLinkedList.

```python
class Professor():


    def professors_details(self):
        col_names = ["Professor ID", "Name", "Rank", "Course ID"]
        col_widths = [25, 25, 25, 10]

        print("\n================================================================================")
        print(f"{col_names[0]:<{col_widths[0]}} {col_names[1]:<{col_widths[1]}} {col_names[2]:<{col_widths[2]}} {col_names[3]:<{col_widths[3]}}")
        print("================================================================================")
        current = self.professors.head
        while current:
            professor_id = str(current.data.get('professor_id', 'N/A'))[:col_widths[0]]
            professor_name = str(current.data.get('professor_name', 'N/A'))[:col_widths[1]]
            rank = str(current.data.get('rank', 'N/A'))[:col_widths[2]]
            course_id = str(current.data.get('course_id', 'N/A'))[:col_widths[3]]

            print(f"{professor_id:<{col_widths[0]}} {professor_name:<{col_widths[1]}} {rank:<{col_widths[2]}} {course_id:<{col_widths[3]}}")
            current = current.next

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
Please enter the new professor name: Mike
Please enter the new professor rank: Assistant Professor
Please enter the new course id: DATA202

What would you like to do?
1. Display professor details
2. Add new professor
3. Delete professor
4. Modify professor details
5. Show course details by professor
6. Exit
Enter your choice: 1

================================================================================
Professor ID              Name                      Rank                      Course ID
================================================================================
michael@mycsu.edu         Michael                   Senior Professor          DATA201
mike@mycsu.edu            Mike                      Assistant Professor       DATA202
```

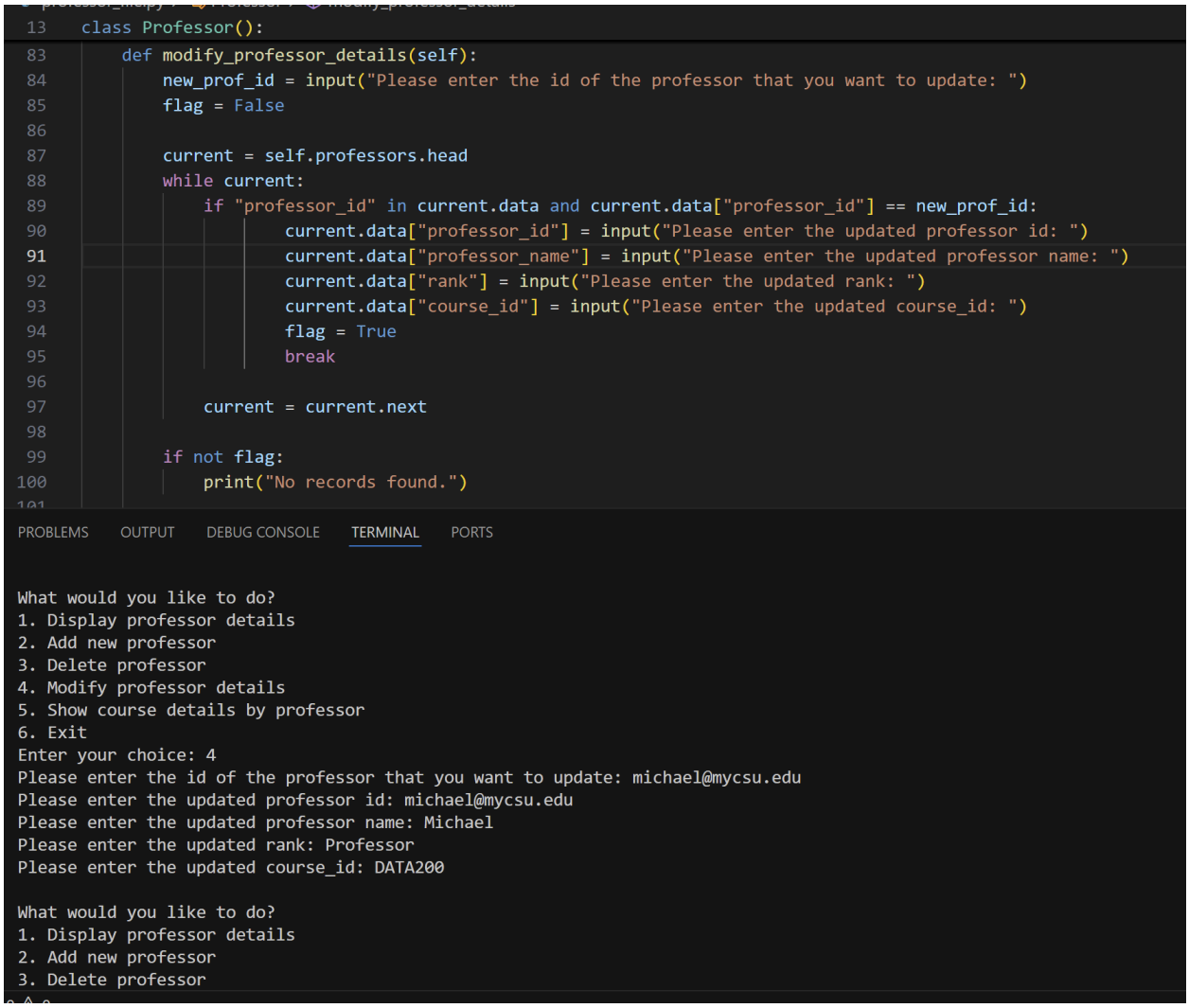
Show course details by professor: This function takes the course id in the ProfessorLinkedList of the professor id/email entered if found. Then it goes through the CourseLinkedList and displays the records for that course id.

```
 13   class Professor():
104       def show_course_details_by_professor(self):
105           self.c1 = Course()
106           self.c1.read_into_courses()
107
108           self.p1 = Professor()
109           self.p1.read_into_professors()
110
111           prof1 = input("What is the professor that you would like to get course details on?: ")
112
113           current = self.p1.professors.head
114           prof_found = False
115
116           while current:
117               if current.data["professor_id"] == prof1:
118                   prof_found = True
119                   course1 = current.data["course_id"]
120                   print(f"\nProfessor {prof1} is a {current.data["rank"]} of course {course1}")
121                   break
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
What would you like to do?
1. Display professor details
2. Add new professor
3. Delete professor
4. Modify professor details
5. Show course details by professor
6. Exit
Enter your choice: 5
What is the professor that you would like to get course details on?: michael@mycsu.edu

Professor michael@mycsu.edu is a Senior Professor of course DATA201

================================================================================
Course ID  Course Name              Description
================================================================================
DATA201    SQL                      Intro to SQL

What would you like to do?
```

Modify professor details: This function ask the user to modify the professor id/email, name, rank, and course id in the ProfessorLinkedList if the professor id entered is found. If not found, then nothing happens. The data gets updated in the csv file too.



```
class Professor():
    def modify_professor_details(self):
        new_prof_id = input("Please enter the id of the professor that you want to update: ")
        flag = False

        current = self.professors.head
        while current:
            if "professor_id" in current.data and current.data["professor_id"] == new_prof_id:
                current.data["professor_id"] = input("Please enter the updated professor id: ")
                current.data["professor_name"] = input("Please enter the updated professor name: ")
                current.data["rank"] = input("Please enter the updated rank: ")
                current.data["course_id"] = input("Please enter the updated course_id: ")
                flag = True
                break

            current = current.next

        if not flag:
            print("No records found.")
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

What would you like to do?
1. Display professor details
2. Add new professor
3. Delete professor
4. Modify professor details
5. Show course details by professor
6. Exit
Enter your choice: 4
Please enter the id of the professor that you want to update: michael@mycsu.edu
Please enter the updated professor id: michael@mycsu.edu
Please enter the updated professor name: Michael
Please enter the updated rank: Professor
Please enter the updated course_id: DATA200

What would you like to do?
1. Display professor details
2. Add new professor
3. Delete professor
```

Delete professor: This function asks for the professor id that the user wants to delete. If it is found, then that node gets deleted from the ProfessorLinkedList and then the professor.csv file gets updated. If it isn't found, then nothing happens.

```
13  class Professor():
75      def delete_professor(self):
76          deleted_professor = input("What is the email address of the professor that you would like to delete?: ")
77
78          if self.professors.delete_node(professor_id=deleted_professor):
79              print("Professor deleted successfully")
80
81          self.update_professor_csv()
82
83      def modify_professor_details(self):
84          new_prof_id = input("Please enter the id of the professor that you want to update: ")
85          flag = False
86
87          current = self.professors.head
88          while current:
89              if "professor_id" in current.data and current.data["professor_id"] == new_prof_id:
90                  current.data["professor_id"] = input("Please enter the updated professor id: ")
91                  current.data["professor_name"] = input("Please enter the updated professor name: ")
92                  current.data["rank"] = input("Please enter the updated rank: ")
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
Enter your choice: 3
What is the email address of the professor that you would like to delete?: mike@mycsu.edu
Professor deleted successfully

What would you like to do?
1. Display professor details
2. Add new professor
3. Delete professor
4. Modify professor details
5. Show course details by professor
6. Exit
Enter your choice: 1

================================================================================
Professor ID              Name                    Rank                    Course ID
================================================================================
michael@mycsu.edu         Michael                 Professor               DATA200

What would you like to do?
```

Add/modify grade: This function asks what student that the user would like to add a grade for. If the grade isn't N/A and the marks aren't 0 (no grade), then there is a grade. The user is then asked if they want to modify the grade. If so, they enter the new marks and based on the marks range the grade from A-F will be populated.

```
12  class Grade():
47      def add_grade(self):
48          email = input("What student would you like to add a grade for?: ")
49
50          current = self.grades.head
51          student_found = False
52
53          while current:
54              course_id = current.data["course_id"]
55              if current.data["email_address"] == email:
56                  student_found = True
57
58                  if current.data["grades"] != "N/A":
59                      grade_edit = input("A grade already exists. Would you like to edit the grade? (y/n): ")
60                      while grade_edit not in ["y", "n"]:
61                          grade_edit = input("Invalid input. Would you like to edit the grade? (y/n): ")
62                      if grade_edit == "n":
63                          return
64
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
What would you like to do?
1. Add/modify grade
2. Delete grade
3. Exit
Enter your choice: 1
What student would you like to add a grade for?: amber.richardson@csu.edu

Please enter the new marks (integer value): r
Invalid input. Please enter an integer

Please enter the new marks (integer value): 96
Grade and marks modified successfully

What would you like to do?
1. Add/modify grade
2. Delete grade
3. Exit
Enter your choice:
```

```
12  class Grade():
47      def add_grade(self):
65                  while True:
66                      try:
67                          new_marks = int(input("\nPlease enter the new marks (integer value): "))
68                          if 0 <= new_marks <= 100:
69                              break
70                          else:
71                              print("Marks must be between 0 and 100. Please try again")
72                          break
73                      except ValueError:
74                          print("Invalid input. Please enter an integer")
75
76                  current.data["marks"] = new_marks
77
78                  if new_marks >= 90:
79                      new_grade = "A"
80                  elif new_marks >= 80:
81                      new_grade = "B"
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
Please enter the new marks (integer value): 96
Grade and marks modified successfully

What would you like to do?
1. Add/modify grade
2. Delete grade
3. Exit
Enter your choice: 1
What student would you like to add a grade for?: amber.richardson@csu.edu
A grade already exists. Would you like to edit the grade? (y/n): y

Please enter the new marks (integer value): 87
Grade and marks modified successfully

What would you like to do?
1. Add/modify grade
2. Delete grade
3. Exit
Enter your choice:
```

Delete grade: This function asks what the email address is of the student that they would like to delete a grade for. If a record is found, the grade becomes N/A and the marks become 0. For add/modify/delete the GradeLinkedList (student id, course id, grades, and marks)gets updated then it goes into the student.csv file.

```
12  class Grade():
101     def delete_grade(self):
102         del_email = input("What is the email address of the grade that you would like to delete?: ")
103
104         updated = False
105
106         current = self.grades.head
107         while current:
108             del_course = current.data["course_id"]
109             if current.data["email_address"] == del_email:
110                 new_grade = current.data["grades"] = "N/A"
111                 new_marks = current.data["marks"] = 0
112                 updated = True
113                 break
114             current = current.next
115
116         if updated:
117             print("Grade and marks deleted successfully")
118             self.update_grade_csv(del_email, del_course, new_grade, new_marks)
119         else:
120             print("No record found.")
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
1. Students
2. Courses
3. Professors
4. Grades
5. Password
6. CHECK MY GRADE APP
7. Exit
Enter your choice: 4

What would you like to do?
1. Add/modify grade
2. Delete grade
3. Exit
Enter your choice: 2
What is the email address of the grade that you would like to delete?: amber.richardson@csu.edu
Grade and marks deleted successfully
```

Decrypt password: This function displays both the encrypted password in the database and the decrypted password using the decrypt password function in the login_file.

```
  3   def check_my_grade_menu():
121                     elif answer == "5":
122                         while True:
123                             print("\nWhat would you like to do?")
124                             print("1. Decrypt password")
125                             print("2. Change password")
126                             print("3. Exit")
127                             ans = input("Enter your choice: ")
128                             if ans == "1":
129                                 print(f"Your encrypted password in the database is {password}")
130                                 decoded = login.decrypt_password(password)
131                                 print(f"Your decrypted password is {decoded}")
132                             elif ans == "2":
133                                 login.change_password(user_id)
134                                 login.read_into_logins()
135
136                                 current = login.logins.head
137                                 while current:
138                                     if current.data["user_id"] == user_id:
139                                         password = current.data["password"]
140                                         break
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
6. CHECK MY GRADE APP
7. Exit
Enter your choice: 5

What would you like to do?
1. Decrypt password
2. Change password
3. Exit
Enter your choice: 1
Your encrypted password in the database is aipgsqi
Your decrypted password is welcome

What would you like to do?
1. Decrypt password
2. Change password
3. Exit
Enter your choice:
```

Grade stats: This function first asks what course that the user would like to get the median and mean score of. This then searches the GradeLinkedList and if the course matches the entered input, the marks gets added to the total, the count is tracked for the mean score, and a list is created for the marks. The marks are then sorted and the middle value is found/calculated. The grade of A-F are displayed based on the mean and median marks.

```
11  class check_my_grade_app():
28      def grade_stats(self):
29          total_marks = 0
30          count = 0
31          course1 = input("What is the course you would like to get grade statistics on?: ")
32
33          found = False
34
35          current = self.grade.grades.head
36          while current:
37              if current.data["course_id"] == course1:
38                  curr = self.grade.grades.head
39                  grade_found = False
40
41                  temp_list = []
42                  while curr:
43                      if curr.data["course_id"] == course1:
44                          marks = int(curr.data.get("marks", 0))
45                          total_marks += marks
46                          count += 1
47                          temp_list.append(marks)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
6. CHECK MY GRADE APP
7. Exit
Enter your choice: 6

What would you like to do?
1. Search records
2. Sort records
3. Get grade statistics of course
4. Display student grade report
5. Display course grade report
6. Display professor grade report
7. Exit
Enter your choice: 3
What is the course you would like to get grade statistics on?: CS103
The average marks of course CS103 was 54.01 to come out to an average grade of F
The median marks of course CS103 was 56.0 to come out to an average grade of F
```

Search student: This function asks whether the user wants to search the StudentLinkedList by student email, first name, last name, course id, or grades. It then displays all the records in the linked list based on the entered input if matching for that column.

```
 11  class check_my_grade_app():
226      def search(self):
253          elif entered == "first":
254              first = input("What is the student first name you would like to find?: ")
255
256              start = time.time()
257
258              found = False
259              curr = self.student.students.head
260
261              while curr:
262                  if curr.data["first_name"] == first:
263                      found = True
264                      break
265                  curr = curr.next
266
267              end = time.time()
268
269              if not found:
270                  print(f"\nNo records found for student first name: {first}")
271                  return
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
What would you like to do?
1. Search records
2. Sort records
3. Get grade statistics of course
4. Display student grade report
5. Display course grade report
6. Display professor grade report
7. Exit
Enter your choice: 1
Do you want to search by student email (email), first name (first), last name (last), course, or grade?: first
What is the student first name you would like to find?: John

================================================================================
Email Address            First Name      Last Name       Course ID       Grade    Marks
================================================================================
john.hardy416@university. John           Hardy           CS107           A        96
```

```
What is the student first name you would like to find?: John

================================================================================================
Email Address             First Name      Last Name       Course ID       Grade    Marks
================================================================================================
john.hardy416@university. John           Hardy           CS107           A        96
john.adams194@university. John           Adams           CS105           A        95
john.clark539@university. John           Clark           CS110           A        95
john.wallis526@university John           Wallis          CS107           A        90
new_student@csu.edu       John           Doe             CS101           B        89
john.oliver346@university John           Oliver          CS107           B        81
john.smith886@university. John           Smith           CS107           B        81
john.manis688@university. John           Manis           CS109           C        73
john.oloughlin410@univers John           Oloughlin       CS101           D        66
john.young274@university. John           Young           CS105           F        58
john.chavez768@university John           Chavez          CS109           F        51
john.dahl561@university.e John           Dahl            CS102           F        45
```

Sort student: This function asks whether the user wants to sort the student.csv file by email or marks in ascending or descending order.

```
 11  class check_my_grade_app():
359      def sorting(self):
360          ll = self.student.students
361
362          entered = input("Do you want to sort by student email or marks?: ")
363
364          if entered == "email":
365              choice = input("Do you want to sort in desc order (True or False): ")
366              if choice not in["True", "False", "true", "false"]:
367                  print("Invalid choice")
368                  return
369
370              choice = choice.strip().lower() == 'true'
371
372              with open('student.csv', mode='r') as file:
373                  reader = csv.DictReader(file)
374                  data = list(reader)
375
376              start = time.time()
377
378              sorted_data = sorted(data, key=lambda x: x['email_address'], reverse = choice)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
Search completed in 0.000000000000000 seconds.

What would you like to do?
1. Search records
2. Sort records
3. Get grade statistics of course
4. Display student grade report
5. Display course grade report
6. Display professor grade report
7. Exit
Enter your choice: 2
Do you want to sort by student email or marks?: email
Do you want to sort in desc order (True or False): False

Sorting completed in 0.000000000000000 seconds
Sorted by email successfully
```

```
email_address,first_name,last_name,course_id,grades,marks
aaron.perkins477@university.edu,Aaron,Perkins,CS108,F,54
adam.pauley241@university.edu,Adam,Pauley,CS102,F,43
adele.bobo243@university.edu,Adele,Bobo,CS104,F,31
alan.fraser101@university.edu,Alan,Fraser,CS102,C,73
alice.bradley646@university.edu,Alice,Bradley,CS107,F,30
alice.welch537@university.edu,Alice,Welch,CS108,F,36
alisa.galvez840@university.edu,Alisa,Galvez,CS101,B,84
alison.patterson813@university.edu,Alison,Patterson,CS104,A,95
allen.binder733@university.edu,Allen,Binder,CS104,A,93
alvin.crespino977@university.edu,Alvin,Crespino,CS108,F,58
amanda.davis691@university.edu,Amanda,Davis,CS102,F,13
amanda.ellison533@university.edu,Amanda,Ellison,CS104,F,52
amanda.leblond244@university.edu,Amanda,Leblond,CS105,D,63
amber.lopez908@university.edu,Amber,Lopez,CS109,D,69
amber.richardson@csu.edu,Amber,Richardson,DATA202,B,87
amelia.mills463@university.edu,Amelia,Mills,CS104,D,66
amy.galarza116@university.edu,Amy,Galarza,CS107,F,3
amy.gouge12@university.edu,Amy,Gouge,CS103,A,100
amy.hall82@university.edu,Amy,Hall,CS103,F,41
amy.mueller734@university.edu,Amy,Mueller,CS105,F,32
amy.seneker514@university.edu,Amy,Seneker,CS105,C,70
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
Search completed in 0.000000000000000 seconds.

What would you like to do?
1. Search records
2. Sort records
3. Get grade statistics of course
4. Display student grade report
5. Display course grade report
6. Display professor grade report
7. Exit
Enter your choice: 2
Do you want to sort by student email or marks?: email
Do you want to sort in desc order (True or False): False

Sorting completed in 0.000000000000000 seconds
Sorted by email successfully
```

Display student grade report: This function displays the grade report (GradeLinkedList) of the email entered if it matches.

```
11   class check_my_grade_app():
169      def display_student_grade_report(self):
170          student1 = input("What is the student you would like to get a grade report on?: ")
171
172          student_found = False
173          curr = self.grade.grades.head
174
175          while curr:
176              if curr.data["email_address"] == student1:
177                  student_found = True
178                  break
179              curr = curr.next
180
181          if not student_found:
182              print(f"\nNo grades found for student: {student1}")
183              return
184
185          col_names = ["Email Address", "Course ID", "Grade", "Marks"]
186          col_widths = [25, 15, 10, 10]
187
188          print("\n==========================================================================================
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
What would you like to do?
1. Search records
2. Sort records
3. Get grade statistics of course
4. Display student grade report
5. Display course grade report
6. Display professor grade report
7. Exit
Enter your choice: 4
What is the student you would like to get a grade report on?: amber.richardson@csu.edu

==========================================================================================
Email Address             Course ID       Grade      Marks
==========================================================================================
amber.richardson@csu.edu  DATA202         B          87
```

Display course grade report: This function displays the grade report (GradeLinkedList) for all matching courses if the entered course id is found.

```
11  class check_my_grade_app():
97      def display_course_grade_report(self):
98          course1 = input("What is the course you would like to get a grade report on?: ")
99
100         grade_found = False
101         curr = self.grade.grades.head
102
103         while curr:
104             if curr.data["course_id"] == course1:
105                 grade_found = True
106                 break
107             curr = curr.next
108
109         if not grade_found:
110             print(f"\nNo grades found for course {course1}")
111             return
112
113         col_names = ["Email Address", "Course ID", "Grade", "Marks"]
114         col_widths = [25, 15, 10, 10]
115
116         print("\n==========================================================================================
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

What is the student you would like to get a grade report on?: amber.richardson@csu.edu

================================================================================
Email Address             Course ID       Grade      Marks
================================================================================
amber.richardson@csu.edu  DATA202         B          87

What would you like to do?
1. Search records
2. Sort records
3. Get grade statistics of course
4. Display student grade report
5. Display course grade report
6. Display professor grade report
7. Exit
Enter your choice: 5
What is the course you would like to get a grade report on?: CS103
```

```
What is the course you would like to get a grade report on?: CS103

================================================================================================
Email Address             Course ID       Grade      Marks
================================================================================================
amy.gouge12@university.ed CS103           A          100
evelyn.wicker782@universi CS103           A          100
debra.olson682@university CS103           A          99
delores.dennison512@unive CS103           A          99
eric.vazquez902@universit CS103           A          99
linda.ruiz942@university. CS103           A          99
duane.ash552@university.e CS103           A          98
edward.jackson232@univers CS103           A          98
george.conner32@universit CS103           A          97
jessica.hunt72@university CS103           A          96
kurt.brinlee962@universit CS103           A          95
lora.pierce572@university CS103           A          95
```

Display professor grade report: This function gets the course id that the entered professor teaches from the ProfessorLinkedList. From there it goes through the GradeLinkedList and displays all matching records with that course.

```
11  class check_my_grade_app():
132     def display_professor_grade_report(self):
133         prof1 = input("What is the professor that you would like to get a grade report on?: ")
134
135         current = self.professor.professors.head
136         prof_found = False
137
138         while current:
139             if current.data["professor_id"] == prof1:
140                 prof_found = True
141                 course1 = current.data["course_id"]
142                 print(f"\nProfessor {prof1} is a {current.data["rank"]} of course {course1}")
143                 break
144             current = current.next
145
146         if not prof_found:
147             print(f"\nNo grades found for professor {prof1}")
148             return
149
150         col_names = ["Email Address", "Course ID", "Grade", "Marks"]
151         col_widths = [25, 15, 10, 10]
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

What is the professor that you would like to get a grade report on?: michael@mycsu.edu

Professor michael@mycsu.edu is a Professor of course CS103

================================================================================
Email Address             Course ID       Grade      Marks
================================================================================
amy.gouge12@university.ed CS103           A          100
amy.hall82@university.edu CS103           F          41
ana.hopkins532@university CS103           A          91
andrew.cooksey832@univers CS103           F          41
andy.gaffney752@universit CS103           C          78
barbara.gischer802@univer CS103           D          65
barbara.smith792@universi CS103           F          47
bernard.ulrich292@univers CS103           F          58
bernice.lowe312@universit CS103           B          80
blair.gelsinger422@univer CS103           A          90
```