CheckMyGrade Screenshots

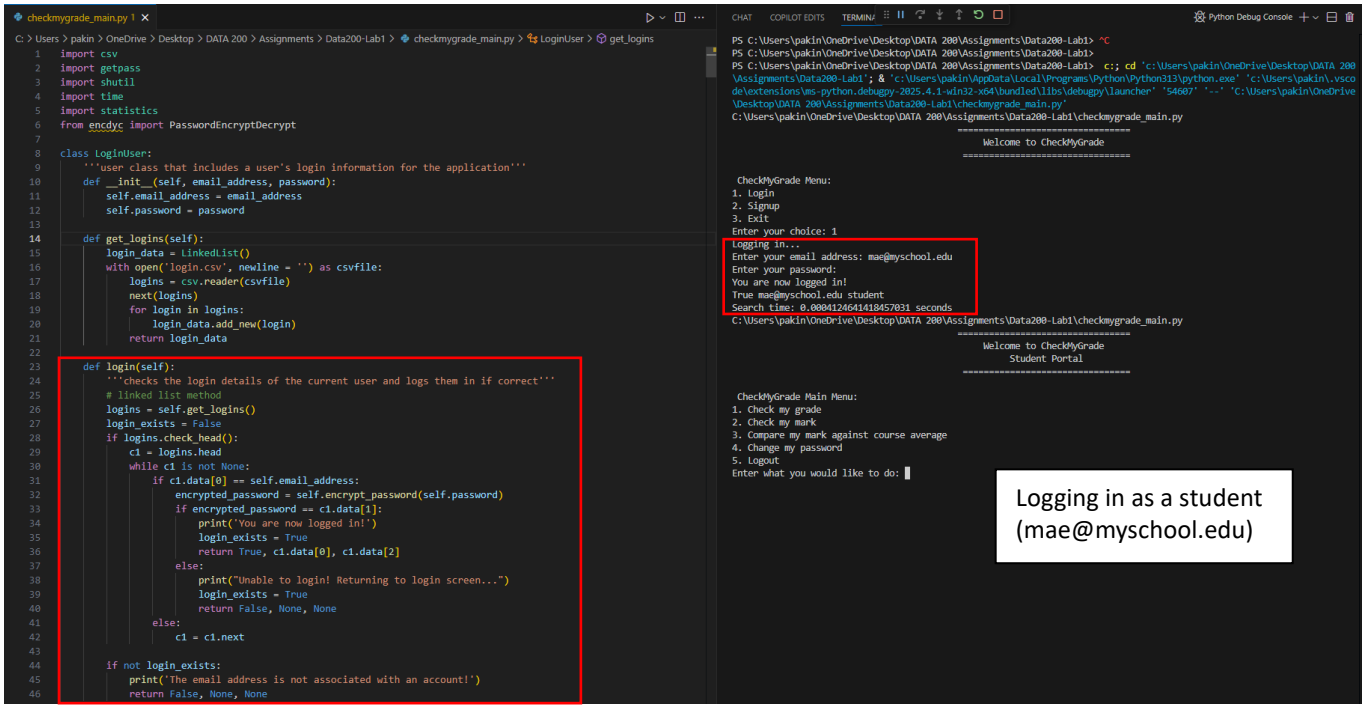
For the CheckMyGrade application, the main screen that a user sees will depend on their permission. There are three types of permissions: **student**, **professor**, or **admin**.

Scenario 1: Student

I will log on using the login details of student Samantha Mae.

Her login credentials are **mae@myshool.edu** and **student1.**

Another student that can be used to test is **smart@myschool.edu** and **student2**.



A successful login with a student's credentials will bring you to the 'Student Portal' screen specifically for students using the application.

The 'Student Portal' gives students the option of checking their letter grades and marks. It also gives them the ability to check their current grades against the course average as a way of evaluating their own performance.

```
            student[5] = self.marks = new_mark
        write_to_file_array('student.csv', header, student_list)
        print(f'The grade and mark for {self.first_name} {self.last_name} has been updated to {new_grade} ({new_ma

    def check_my_marks(self):
        '''lets student check their own marks'''
        # array method
        student_list = get_data('student.csv')[1]
        for student in student_list:
            if student[0] == self.email_address:
                course_list = self.course_ids.split('|')
                if self.course_ids:
                    print(f'{self.email_address} is currently taking {len(course_list)} courses.')
                    print('Showing marks...')
                    marks_list = self.marks.split('|')
                    for idx, course in enumerate(course_list):
                        print(f"{course}: {marks_list[idx]}%")
                else:
                    print(f'{self.email_address} currently has no marks.')

    @staticmethod
    def sort_students(by, order):
        '''sorts students either by email or grade'''
        start_time = time.time()
        student_list = get_data('student.csv')[1]
        grades_list = Grades.get_grades()
        sorted_list = sorted(student_list, key = lambda x: x[0]) if by == 'email' else sorted(grades_list, key = l
        if order == 'reverse':
            sorted_list = sorted_list[::-1]
        for data in sorted_list:
            print(data)
```

```
Showing grades...
DATA200: B+
DATA201: A-
C:\Users\pakin\OneDrive\Desktop\DATA 200\Assignments\Data200-Lab1\checkmygrade_main.py
==============================
Welcome to CheckMyGrade
Student Portal
==============================

 CheckMyGrade Main Menu:
1. Check my grade
2. Check my mark
3. Compare my mark against course average
4. Change my password
5. Logout
Enter what you would like to do: 2
Checking your mark...
mae@myschool.edu is currently taking 2 courses.
Showing marks...
DATA200: 88%
DATA201: 92%
C:\Users\pakin\OneDrive\Desktop\DATA 200\Assignments\Data200-Lab1\checkmygrade_main.py
==============================
Welcome to CheckMyGrade
Student Portal
==============================

 CheckMyGrade Main Menu:
1. Check my grade
2. Check my mark
3. Compare my mark against course average
4. Change my password
5. Logout
Enter what you would like to do:
```

2. Check my marks (checks the current student's marks)

```
checkmygrade_main.py 1
C: > Users > pakin > OneDrive > Desktop > DATA 200 > Assignments > Data200-Lab1 > checkmygrade_main.py > LoginUser > get_logins
345 class Grades:
363
364     @staticmethod
365     def display_grade_report():
366         '''displays a report of all the grades for students'''
367         grades = Grades.get_grades()
368         for grade in grades:
369             grade_id = grade[0]
370             grade_id = Grades(grade[0], grade[1], grade[2], grade[3])
371             print(f'{grade_id.grade_id}: {grade_id.grade} / {grade_id.mark}% / {grade[3]}')
372
373     @staticmethod
374     def display_grades_by_course(course):
375         start_time = time.time()
376         grades = Grades.get_grades()
377         course_grades = []
378         for grade in grades:
379             if grade[3] == course:
380                 course_grades.append(grade[2])
381                 print(f'{grade[0]}: {grade[1]} ({grade[2]}%)')
382
383         print(f'There are {len(course_grades)} students in {course}.\n')
384         print(f'Search time: {time.time() - start_time} seconds')
385
386     @staticmethod
387     def get_course_average(course):
388         course_grades = []
389         grades = Grades.get_grades()
390         for grade in grades:
391                 if grade[3] == course:
392                     course_grades.append(int(grade[2]))
393         class_average = sum(course_grades) / len(course_grades)
394         return class_average
395
396     @staticmethod
397     def get_course_median(course):
398         course_grades = []
399         grades = Grades.get_grades()
400         for grade in grades:
401                 if grade[3] == course:
402                     course_grades.append(int(grade[2]))
403         class_median = statistics.median(course_grades)
404         return class_median
405
406 class Node:
407     def __init__(self, data):
408         self.data = data
409         self.next = None
```

```
DATA200: B+
DATA201: A-
C:\Users\pakin\OneDrive\Desktop\DATA 200\Assignments\Data200-Lab1\checkmygrade_main.py
==============================
Welcome to CheckMyGrade
Student Portal
==============================

 CheckMyGrade Main Menu:
1. Check my grade
2. Check my mark
3. Compare my mark against course average
4. Change my password
5. Logout
Enter what you would like to do: 2
Checking your mark...
mae@myschool.edu is currently taking 2 courses.
Showing marks...
DATA200: 88%
DATA201: 92%
C:\Users\pakin\OneDrive\Desktop\DATA 200\Assignments\Data200-Lab1\checkmygrade_main.py
==============================
Welcome to CheckMyGrade
Student Portal
==============================

 CheckMyGrade Main Menu:
1. Check my grade
2. Check my mark
3. Compare my mark against course average
4. Change my password
5. Logout
Enter what you would like to do: 3
Getting information for comparison...
Class 1: DATA200
Your mark: 88%
Average: 86.000000%
Median: 88.0%
Class 2: DATA201
Your mark: 92%
Average: 84.600000%
Median: 90%
C:\Users\pakin\OneDrive\Desktop\DATA 200\Assignments\Data200-Lab1\checkmygrade_main.py
==============================
Welcome to CheckMyGrade
Student Portal
==============================

 CheckMyGrade Main Menu:
1. Check my grade
2. Check my mark
3. Compare my mark against course average
4. Change my password
5. Logout
Enter what you would like to do:
```

3. Compares student's mark to that of the course

```
    def logout(self):
        '''logs the current user out of the system'''
        return False

    def change_password(self):
        '''changes a user's password'''
        logins_ll, login_list, header = get_data('login.csv')
        for login in login_list:
            if login[0] == self.email_address:
                new_password = getpass.getpass("Enter your new password: ")
                encrypted_new_password = self.encrypt_password(new_password)
                login[1] = self.password = encrypted_new_password
        write_to_file_array('login.csv', header, login_list)
        print(f'Your password has been updated! Please try logging in with your new password...')
        return False

    def encrypt_password(self, password):
        '''encrypts a user's password'''
        preencrypted_password = PasswordEncryptDecrypt(4)
        encrypted_password = preencrypted_password.encrypt(password)
        return encrypted_password

    def decrypt_password(self, password):
        '''decrypts a user's password'''
        predecrypted_password = PasswordEncryptDecrypt(4)
        decrypted_password = predecrypted_password.decrypt(password)
        return decrypted_password

class Student:
    '''student class that includes information regarding a student's name, email, grades/marks'''
    def __init__(self, first_name, last_name, email_address, course_ids, grades, marks):
        self.first_name = first_name
        self.last_name = last_name
```

```
Getting information for comparison...
Class 1: DATA200
Your mark: 88%
Average: 86.000000%
Median: 88.0%
Class 2: DATA201
Your mark: 92%
Average: 84.600000%
Median: 90%
C:\Users\pakin\OneDrive\Desktop\DATA 200\Assignments\Data200-Lab1\checkmygrade_main.py
==============================

==============================

 CheckMyGrade Main Menu:
1. Check my grade
2. Check my mark
3. Compare my mark against course average
4. Change my password
5. Logout
Enter what you would like to do: 4
Changing your password...
Enter your new password:
Your password has been updated! Please try logging in with your new password...
C:\Users\pakin\OneDrive\Desktop\DATA 200\Assignments\Data200-Lab1\checkmygrade_main.py
==============================
Welcome to CheckMyGrade
Student Portal
==============================

 CheckMyGrade Main Menu:
1. Check my grade
2. Check my mark
3. Compare my mark against course average
4. Change my password
5. Logout
Enter what you would like to do:
```

4. Change Password (requires encrypting the new password)

checkmygrade_main.py 1 | login.csv

C: > Users > pakin > OneDrive > Desktop > DATA 200 > Assignments > Data200-Lab1 > login.csv

```
1  user_id,password,role
2  admin@myschool.edu,tewwasvh,admin
3  smith@myschool.edu,tvsjiwwsvi,professor
4  mae@myschool.edu,wxyhirxi,student
5  smart@myschool.edu,wxyhirxj,student
6
```

4. Change Password (the new encrypted password gets stored back into login.csv)

```
            students_ll.add_new(new_student)
            add_to_file('student.csv', new_student)
        else:
            new_professor = [email, first_name + '' + last_name,  None, None]
            professors_ll = get_data('professor.csv')[0]
            professors_ll.add_new(new_professor)
            add_to_file('professor.csv', new_professor)
        print('You have successfully signed up! Please try logging in.')

    def logout(self):
        '''logs the current user out of the system'''
        return False

    def change_password(self):
        '''changes a user's password'''
        logins_ll, login_list, header = get_data('login.csv')
        for login in login_list:
            if login[0] == self.email_address:
                new_password = getpass.getpass("Enter your new password: ")
                encrypted_new_password = self.encrypt_password(new_password)
                login[1] = self.password = encrypted_new_password
        write_to_file_array('login.csv', header, login_list)
        print(f'Your password has been updated! Please try logging in with your new password...')
        return False
```

```
Changing your password...
Enter your new password:
Your password has been updated! Please try logging in with your new password...
C:\Users\pakin\OneDrive\Desktop\DATA 200\Assignments\Data200-Lab1\checkmygrade_main.py
                                        ==============================
                                              Welcome to CheckMyGrade
                                                 Student Portal
                                        ==============================

 CheckMyGrade Main Menu:
1. Check my grade
2. Check my mark
3. Compare my mark against course average
4. Change my password
5. Logout
Enter what you would like to do: 5
Logging out...
C:\Users\pakin\OneDrive\Desktop\DATA 200\Assignments\Data200-Lab1\checkmygrade_main.py
                                        ==============================
                                              Welcome to CheckMyGrade
                                        ==============================

 CheckMyGrade Menu:
1. Login
2. Signup
3. Exit
Enter your choice:
```

5. Logging out (returns to login screen)

Scenario 2: Professor

I will log on using the login details of professor John Smith.

His login credentials are **smith@myshool.edu** and **professor1.**

```
def login(self):
    '''checks the login details of the current user and logs them in if correct'''
    # linked list method
    logins = self.get_logins()
    login_exists = False
    if logins.check_head():
        c1 = logins.head
        while c1 is not None:
            if c1.data[0] == self.email_address:
                encrypted_password = self.encrypt_password(self.password)
                if encrypted_password == c1.data[1]:
                    print('You are now logged in!')
                    login_exists = True
                    return True, c1.data[0], c1.data[2]
                else:
                    print("Unable to login! Returning to login screen...")
                    login_exists = True
                    return False, None, None
            else:
                c1 = c1.next

    if not login_exists:
        print('The email address is not associated with an account!')
        return False, None, None

def signup(self, email, password, first_name, last_name, role):
```

```
Logging out...
C:\Users\pakin\OneDrive\Desktop\DATA 200\Assignments\Data200-Lab1\checkmygrade_main.py
                                        ==============================
                                              Welcome to CheckMyGrade
                                        ==============================

 CheckMyGrade Menu:
1. Login
2. Signup
3. Exit
Enter your choice: 1
Logging in...
Enter your email address: smith@myschool.edu
Enter your password:
You are now logged in!
True smith@myschool.edu professor
Search time: 0.0006113052368164062 seconds
C:\Users\pakin\OneDrive\Desktop\DATA 200\Assignments\Data200-Lab1\checkmygrade_main.py
                                        ==============================
                                              Welcome to CheckMyGrade
                                                 Professor Portal
                                        ==============================

 CheckMyGrade Main Menu:
1. Check my course
2. Get grade report for my course
3. Change student grades/marks
4. Change my password
5. Logout
Enter what you would like to do:
```

Logging in as a professor (smith@myschool.edu)

A successful login with a professor's credentials will bring you to the 'Professor Portal' screen specifically for professors using the application.

The 'Professor Portal' gives professors the ability to get their course details. It also allows them to generate a grade report for their course, where they can see the grades of each of the students taking their course. Professors also have the ability to update grades for their students.

```python
    def modify_professor_details(self, new_rank):
        '''modify a professor in the system using their email address'''
        professors_ll, professor_list, header = get_data('professor.csv')
        for professor in professor_list:
            if professor[0] == self.email_address:
                professor[2] = self.rank = new_rank
        write_to_file_array('professor.csv', header, professor_list)
        print(f'The rank for {self.name} has been updated to {self.rank}')

    def show_course_details_by_professor(self):
        '''show a professor's course using their email address'''
        start_time = time.time()
        professor_list = get_data('professor.csv')[1]
        course_list = get_data('course.csv')[1]
        if self.email_address:
            for professor in professor_list:
                if professor[0] == self.email_address:
                    for course in course_list:
                        if professor[3] == course[0]:
                            print(f'''
                                Showing course details for {professor[1]}:
                                ID: {course[0]}
                                Name: {course[1]}
                                Credits: {course[2]}
                                Description: {course[3]}
                                ''')
        print(f'Search time: {time.time() - start_time} seconds')

class Grades:
    '''grades class that includes information regarding a specific grade for a student'''
    def __init__(self, grade_id, grade, mark, course_id = None):
```

```
True smith@myschool.edu professor
Search time: 0.0006113052368164062 seconds
C:\Users\pakin\OneDrive\Desktop\DATA 200\Assignments\Data200-Lab1\checkmygrade_main.py
                                        ==============================
                                              Welcome to CheckMyGrade

                                        =======

 CheckMyGrade Main Menu:
1. Check my course
2. Get grade report for my course
3. Change student grades/marks
4. Change my password
5. Logout
Enter what you would like to do: 1
Retrieving your course details...

                                        Showing course details for John Smith:
                                        ID: DATA200
                                        Name: Intro to Data Science
                                        Credits: 3
                                        Description: Utilizing Python in Data Science

Search time: 0.0021963119506835938 seconds
C:\Users\pakin\OneDrive\Desktop\DATA 200\Assignments\Data200-Lab1\checkmygrade_main.py
                                        ==============================
                                              Welcome to CheckMyGrade
                                                Professor Portal
                                        ==============================

 CheckMyGrade Main Menu:
1. Check my course
2. Get grade report for my course
3. Change student grades/marks
4. Change my password
5. Logout
Enter what you would like to do:
```

1. Check my course (gets the details of the current professor's course)

```python
    @staticmethod
    def display_grade_report():
        '''displays a report of all the grades for students'''
        grades = Grades.get_grades()
        for grade in grades:
            grade_id = grade[0]
            grade_id = Grades(grade[0], grade[1], grade[2], grade[3])
            print(f'{grade_id.grade_id}: {grade_id.grade} / {grade_id.mark}% / {grade[3]}')

    @staticmethod
    def display_grades_by_course(course):
        start_time = time.time()
        grades = Grades.get_grades()
        course_grades = []
        for grade in grades:
            if grade[3] == course:
                course_grades.append(grade[2])
                print(f'{grade[0]}: {grade[1]} ({grade[2]}%)')

        print(f'There are {len(course_grades)} students in {course}.\n')
        print(f'Search time: {time.time() - start_time} seconds')

    @staticmethod
    def get_course_average(course):
        course_grades = []
        grades = Grades.get_grades()
        for grade in grades:
                if grade[3] == course:
                    course_grades.append(int(grade[2]))
        class_average = sum(course_grades) / len(course_grades)
        return class_average

    @staticmethod
    def get_course_median(course):
```

```
C:\Users\pakin\OneDrive\Desktop\DATA 200\Assignments\Data200-Lab1\checkmygrade_main.py
                                        ==============================
                                              Welcome to CheckMyGrade
                                                Professor Portal
                                        ==============================

 CheckMyGrade Main Menu:
1. Check my course
2. Get grade report for my course
3. Change student grades/marks
4. Change my password
5. Logout
Enter what you would like to do: 2
Getting grade report for your course (DATA200)...
mae@myschool.edu: B+ (88%)
russell@myschool.edu: A- (90%)
smart@myschool.edu: B+ (88%)
brown@myschool.edu: B (86%)
moore@myschool.edu: A (94%)
johnson@myschool.edu: C- (70%)
There are 6 students in DATA200.

Search time: 0.0022432804107666016 seconds
Course average: 86.0%
Course median: 88.0
C:\Users\pakin\OneDrive\Desktop\DATA 200\Assignments\Data200-Lab1\checkmygrade_main.py
                                        ==============================
                                              Welcome to CheckMyGrade
                                                Professor Portal
                                        ==============================

 CheckMyGrade Main Menu:
1. Check my course
2. Get grade report for my course
3. Change student grades/marks
4. Change my password
5. Logout
Enter what you would like to do:
```

2. Get grade report (gets all grades of the students enrolled in the professor's course, average, and median)

```python
@staticmethod
def display_grades_by_course(course):
    start_time = time.time()
    grades = Grades.get_grades()
    course_grades = []
    for grade in grades:
        if grade[3] == course:
            course_grades.append(grade[2])
            print(f'{grade[0]}: {grade[1]} ({grade[2]}%)')

    print(f'There are {len(course_grades)} students in {course}.\n')
    print(f'Search time: {time.time() - start_time} seconds')

@staticmethod
def get_course_average(course):
    course_grades = []
    grades = Grades.get_grades()
    for grade in grades:
            if grade[3] == course:
                course_grades.append(int(grade[2]))
    class_average = sum(course_grades) / len(course_grades)
    return class_average
```

```
Search time: 0.0022432804107666016 seconds
Course average: 86.0%
Course median: 88.0
C:\Users\pakin\OneDrive\Desktop\DATA 200\Assignments\Data200-Lab1\checkmygrade_main.py
                                        ==============================
                                              Welcome to CheckMyGrade
                                                Professor Portal
                                        ==============================

 CheckMyGrade Main Menu:
1. Check my course
2. Get grade report for my course
3. Change student grades/marks
4. Change my password
5. Logout
Enter what you would like to do: 3
Showing list of students in your course...
mae@myschool.edu: B+ (88%)
russell@myschool.edu: A- (90%)
smart@myschool.edu: B+ (88%)
brown@myschool.edu: B (86%)
moore@myschool.edu: A (94%)
johnson@myschool.edu: C- (70%)
There are 6 students in DATA200.

Search time: 0.002649068832397461 seconds
Enter the email of the student you would like to edit:
```

3. Change student grades/marks – First gives a professor a list student grades for them to select from

```
        print(f'Error deleting student: {e}!')

    def check_my_grades(self):
        '''lets student check their own grades'''
        # array method
        student_list = get_data('student.csv')[1]
        for student in student_list:
            if student[0] == self.email_address:
                course_list = self.course_ids.split('|')
                if self.course_ids:
                    print(f'{self.email_address} is currently taking {len(course_list)} courses.')
                    print('Showing grades...')
                    grades_list = self.grades.split('|')
                    for idx, course in enumerate(course_list):
                        print(f"{course}: {grades_list[idx]}")
                else:
                    print(f'{self.email_address} currently has no grades.')

    def update_student_record(self, new_grade, new_mark):
        '''updates a student's record by using their email address'''
        # array method
        students_ll, student_list, header = get_data('student.csv')
        for student in student_list:
            if student[0] == self.email_address:
                student[4] = self.grades = new_grade
                student[5] = self.marks = new_mark
        write_to_file_array('student.csv', header, student_list)
        print(f'The grade and mark for {self.first_name} {self.last_name} has been updated to {new_grade} ({new_mark

    def check_my_marks(self):
        '''lets student check their own marks'''
        # array method
        student_list = get_data('student.csv')[1]
        for student in student_list:
            if student[0] == self.email_address:
                course_list = self.course_ids.split('|')
                if self.course_ids:
                    print(f'{self.email_address} is currently taking {len(course_list)} courses.')
                    print('Showing marks...')
                    marks_list = self.marks.split('|')
                    for idx, course in enumerate(course_list):
                        print(f"{course}: {marks_list[idx]}%")
                else:
                    print(f'{self.email_address} currently has no marks.')

    @staticmethod
    def sort_students(by, order):
```

```
CheckMyGrade Main Menu:
1. Check my course
2. Get grade report for my course
3. Change student grades/marks
4. Change my password
5. Logout
Enter what you would like to do: 3
Showing list of students in your course...
mae@myschool.edu: B+ (88%)
russell@myschool.edu: A- (90%)
smart@myschool.edu: B+ (88%)
brown@myschool.edu: B (86%)
moore@myschool.edu: A (94%)
johnson@myschool.edu: C- (70%)
There are 6 students in DATA200.

Search time: 0.0026490688323974610 seconds
Enter the email of the student you would like to edit: johnson@myschool.edu
Search time: 0.0006127357482910156 seconds
Enter the new mark for Jack Johnson: 75
Enter the new grade for Jack Johnson: C
The grade and mark for Jack Johnson has been updated to C (75%)
C:\Users\pakin\OneDrive\Desktop\DATA 200\Assignments\Data200-Lab1\checkmygrade_main.py
==============================
Welcome to CheckMyGrade
Professor Portal
==============================

CheckMyGrade Main Menu:
1. Check my course
2. Get grade report for my course
3. Change student grades/marks
4. Change my password
5. Logout
Enter what you would like to do: 2
Getting grade report for your course (DATA200)...
mae@myschool.edu: B+ (88%)
russell@myschool.edu: A- (90%)
smart@myschool.edu: B+ (88%)
brown@myschool.edu: B (86%)
moore@myschool.edu: A (94%)
johnson@myschool.edu: C (75%)
There are 6 students in DATA200.

Search time: 0.003100872039794922 seconds
Course average: 86.83333333333333%
Course median: 88.0
C:\Users\pakin\OneDrive\Desktop\DATA 200\Assignments\Data200-Lab1\checkmygrade_main.py
==============================
Welcome to CheckMyGrade
Professor Portal
```

3. Change student grades/marks – johnson@myschool.edu is selected and given a new mark of 75 and a new grade of C

Getting a new grade report shows the student with the new updated grade

checkmygrade_main.py 1 | student.csv | login.csv

C: > Users > pakin > OneDrive > Desktop > DATA 200 > Assignments > Data200-Lab1 > student.csv

```
1  student_id,first_name,last_name,course_ids,grades,marks
2  mae@myschool.edu,Samantha,Mae,DATA200|DATA201,B+|A-,88|92
3  russell@myschool.edu,Sally,Russell,DATA200,A-,90
4  doe@myschool.edu,John,Doe,DATA201|DATA203,A|B+,94|89
5  smart@myschool.edu,Jane,Smart,DATA200|DATA201|DATA202,B+|A-|A,88|90|96
6  jones@myschool.edu,Emily,Jones,DATA203,A-,92
7  brown@myschool.edu,Michael,Brown,DATA200|DATA203,B|C+,86|78
8  davis@myschool.edu,Sarah,Davis,DATA202|DATA203,A|B+,97|88
9  miller@myschool.edu,William,Miller,DATA201|DATA202,C-|D+,72|68
10 wilson@myschool.edu,Lucy,Wilson,DATA202,B-,80
11 moore@myschool.edu,Charles,Moore,DATA200,A,94
12 taylor@myschool.edu,Olivia,Taylor,DATA204,B+,88
13 anderson@myschool.edu,Benjamin,Anderson,DATA201,C,75
14 johnson@myschool.edu,Jack,Johnson,DATA200,C,75
15
```

The student.csv file reflects the new update

```
                professors_ll.add_new(new_professor)
                add_to_file('professor.csv', new_professor)
            print('You have successfully signed up! Please try logging in.')

    def logout(self):
        '''logs the current user out of the system'''
        return False

    def change_password(self):
        '''changes a user's password'''
        logins_ll, login_list, header = get_data('login.csv')
        for login in login_list:
            if login[0] == self.email_address:
                new_password = getpass.getpass("Enter your new password: ")
                encrypted_new_password = self.encrypt_password(new_password)
                login[1] = self.password = encrypted_new_password
        write_to_file_array('login.csv', header, login_list)
        print(f'Your password has been updated! Please try logging in with your new password...')
        return False

    def encrypt_password(self, password):
        '''encrypts a user's password'''
        preencrypted_password = PasswordEncryptDecrypt(4)
        encrypted_password = preencrypted_password.encrypt(password)
        return encrypted_password
```

```
C:\Users\pakin\OneDrive\Desktop\DATA 200\Assignments\Data200-Lab1\checkmygrade_main.py
==============================
Welcome to CheckMyGrade
==============================

CheckMyGrade Main Menu:
1. Check my course
2. Get grade report for my course
3. Change student grades/marks
4. Change my password
5. Logout
Enter what you would like to do: 4
Changing your password...
Enter your new password:
Your password has been updated! Please try logging in with your new password...
C:\Users\pakin\OneDrive\Desktop\DATA 200\Assignments\Data200-Lab1\checkmygrade_main.py
==============================
Welcome to CheckMyGrade
Professor Portal
==============================

CheckMyGrade Main Menu:
1. Check my course
2. Get grade report for my course
3. Change student grades/marks
4. Change my password
5. Logout
Enter what you would like to do:
```

4. Change Password (requires encrypting the new password) – rewrites to login.csv (as seen above)

```
def logout(self):
    '''logs the current user out of the system'''
    return False

def change_password(self):
    '''changes a user's password'''
    logins_ll, login_list, header = get_data('login.csv')
    for login in login_list:
        if login[0] == self.email_address:
            new_password = getpass.getpass("Enter your new password: ")
            encrypted_new_password = self.encrypt_password(new_password)
            login[1] = self.password = encrypted_new_password
    write_to_file_array('login.csv', header, login_list)
    print(f'Your password has been updated! Please try logging in with your new password...')
    return False

def encrypt_password(self, password):
    '''encrypts a user's password'''
    preencrypted_password = PasswordEncryptDecrypt(4)
    encrypted_password = preencrypted_password.encrypt(password)
    return encrypted_password
```

```
Enter your new password:
Your password has been updated! Please try logging in with your new password...
C:\Users\pakin\OneDrive\Desktop\DATA 200\Assignments\Data200-Lab1\checkmygrade_main.py
                                        ==============================
                                              Welcome to CheckMyGrade
                                                Professor Portal
                                        ==============================

 CheckMyGrade Main Menu:
1. Check my course
2. Get grade report for my course
3. Change student grades/marks
4. Change my password
5. Logout
Enter what you would like to do: 5
Logging out...
C:\Users\pakin\OneDrive\Desktop\DATA 200\Assignments\Data200-Lab1\checkmygrade_main.py
                                        ==============================
                                              Welcome to CheckMyGrade
                                        ==============================


 CheckMyGrade Menu:
1. Login
2. Signup
3. Exit
Enter your choice:
```

5. Logging out (returns to login screen)

Scenario 3: Admin

I will log on using the login details of the admin.

The admin credentials are **admin@myschool.edu** and **admin1.**

```
def login(self):
    '''checks the login details of the current user and logs them in if correct'''
    # linked list method
    logins = self.get_logins()
    login_exists = False
    if logins.check_head():
        c1 = logins.head
        while c1 is not None:
            if c1.data[0] == self.email_address:
                encrypted_password = self.encrypt_password(self.password)
                if encrypted_password == c1.data[1]:
                    print('You are now logged in!')
                    login_exists = True
                    return True, c1.data[0], c1.data[2]
                else:
                    print("Unable to login! Returning to login screen...")
                    login_exists = True
                    return False, None, None
            else:
                c1 = c1.next

    if not login_exists:
        print('The email address is not associated with an account!')
        return False, None, None

def signup(self, email, password, first_name, last_name, role):
```

```
2. Get grade report for my course
3. Change student grades/marks
4. Change my password
5. Logout
Enter what you would like to do: 5
Logging out...
C:\Users\pakin\OneDrive\Desktop\DATA 200\Assignments\Data200-Lab1\checkmygrade_main.py
                                        ==============================
                                              Welcome to CheckMyGrade
                                        ==============================


 CheckMyGrade Menu:
1. Login
2. Signup
3. Exit
Enter your choice: 1
Logging in...
Enter your email address: admin@myschool.edu
Enter your password:
You are now logged in!
True admin@myschool.edu admin
C:\Users\pakin\OneDrive\Desktop\DATA 200\Assignments\Data200-Lab1\checkmygrade_main.py
                                        ==============================
                                              Welcome to CheckMyGrade
                                                Admin Portal
                                        ==============================

1. Show student options
2. Show professor options
3. Show course options
4. Logout
Enter what you would like to do:
```

Logging in as the admin (admin@myschool.edu)

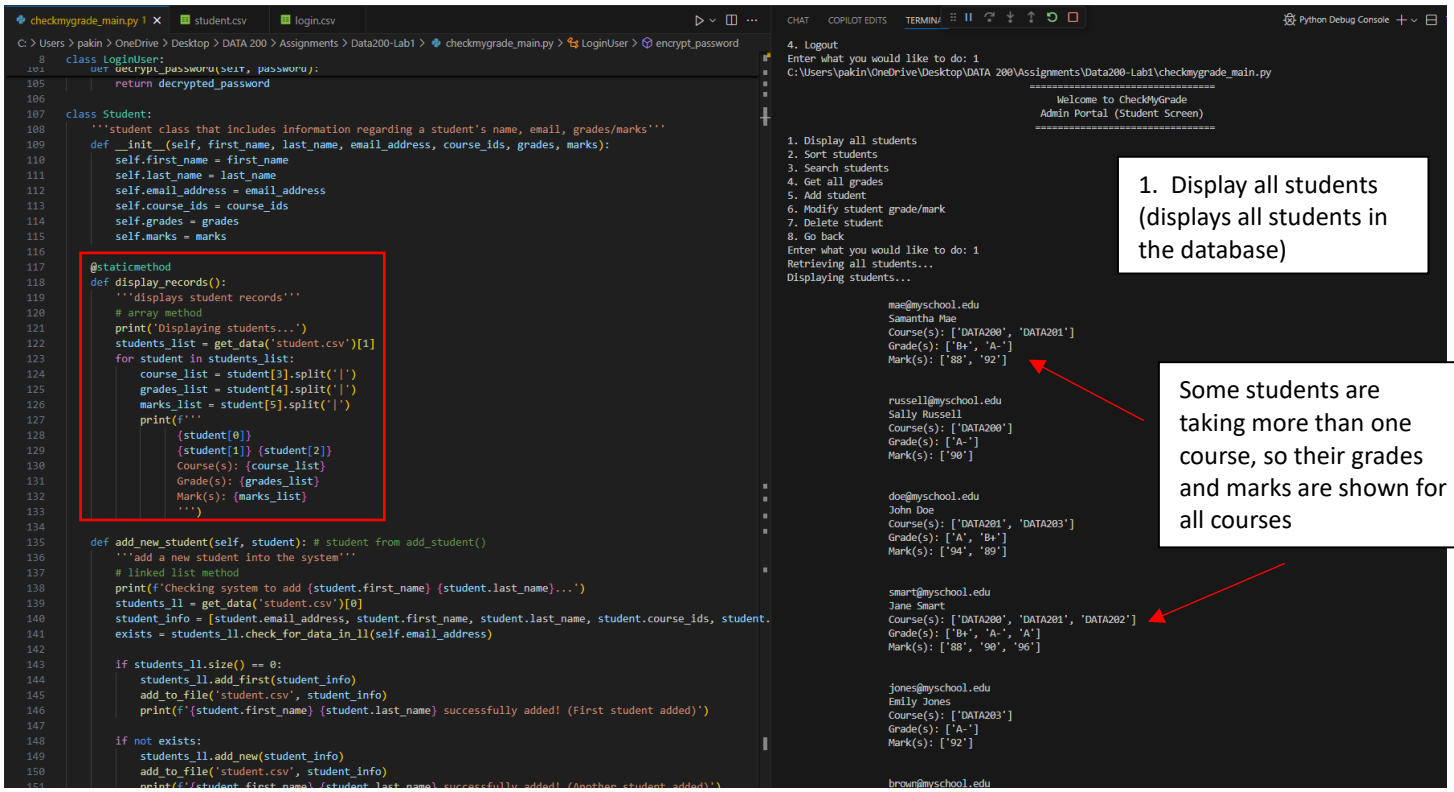
A successful login with the admin credentials will bring you to the ‘Admin Portal’ screen specifically for the admin of the application.

The ‘Admin Portal’ gives the admin the ability to work with student, professor, and course data. The portal is split into three separate screens for student, professor, and course options. The admin has access to more modification and evaluation functions than either students or professors. They have the ability to add and delete students, professors, and courses. Admins can also sort students, modify both students and professors, and create grade reports based on professors or courses.

```
    if choice == '1':
        student_screen = True
        while student_screen:
            print(__file__)
            print('================================'.center(columns))
            msg='''Welcome to CheckMyGrade'''
            print(msg.center(columns))
            msg2='''Admin Portal (Student Screen)'''
            print(msg2.center(columns))
            print('================================'.center(columns))
            print('1. Display all students')
            print('2. Sort students')
            print('3. Search students')
            print('4. Get all grades')
            print('5. Add student')
            print('6. Modify student grade/mark')
            print('7. Delete student')
            print('8. Go back')
            student_choice = input('Enter what you would like to do: ')
            if student_choice == '1':
                print('Retrieving all students...')
                Student.display_records()
            elif student_choice == '2':
```

```
You are now logged in!
True admin@myschool.edu admin
C:\Users\pakin\OneDrive\Desktop\DATA 200\Assignments\Data200-Lab1\checkmygrade_main.py
                                        ==============================
                                              Welcome to CheckMyGrade
                                                Admin Portal
                                        ==============================

1. Show student options
2. Show professor options
3. Show course options
4. Logout
Enter what you would like to do: 1
C:\Users\pakin\OneDrive\Desktop\DATA 200\Assignments\Data200-Lab1\checkmygrade_main.py
                                        ==============================
                                              Welcome to CheckMyGrade
                                          Admin Portal (Student Screen)
                                        ==============================

1. Display all students
2. Sort students
3. Search students
4. Get all grades
5. Add student
6. Modify student grade/mark
7. Delete student
8. Go back
Enter what you would like to do:
```

Opting to view the student screen

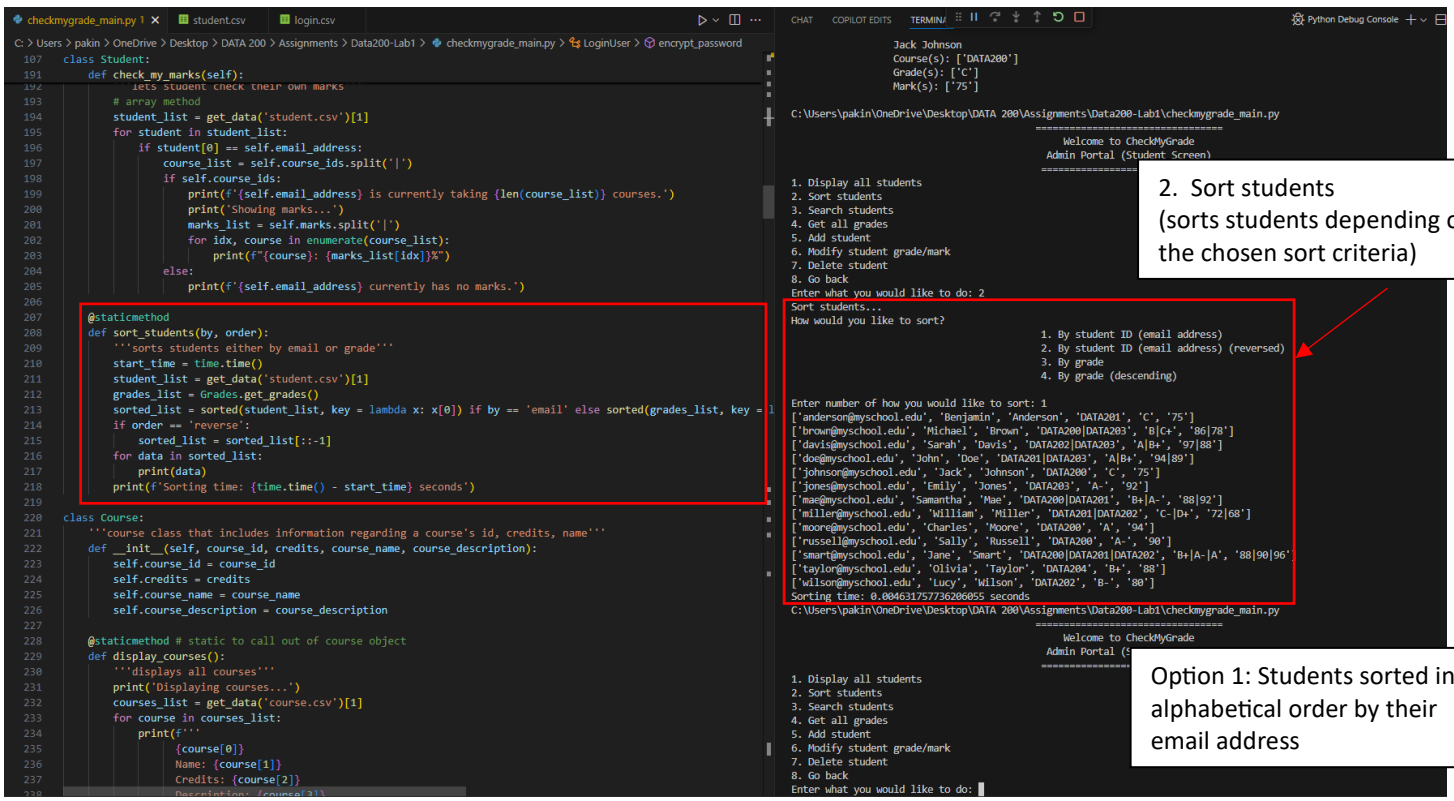


```
class LoginUser:
    def decrypt_password(self, password):
        return decrypted_password

class Student:
    '''student class that includes information regarding a student's name, email, grades/marks'''
    def __init__(self, first_name, last_name, email_address, course_ids, grades, marks):
        self.first_name = first_name
        self.last_name = last_name
        self.email_address = email_address
        self.course_ids = course_ids
        self.grades = grades
        self.marks = marks

    @staticmethod
    def display_records():
        '''displays student records'''
        # array method
        print('Displaying students...')
        students_list = get_data('student.csv')[1]
        for student in students_list:
            course_list = student[3].split('|')
            grades_list = student[4].split('|')
            marks_list = student[5].split('|')
            print(f'''
                {student[0]}
                {student[1]} {student[2]}
                Course(s): {course_list}
                Grade(s): {grades_list}
                Mark(s): {marks_list}
                ''')

    def add_new_student(self, student): # student from add_student()
        '''add a new student into the system'''
        # linked list method
        print(f'Checking system to add {student.first_name} {student.last_name}...')
        students_ll = get_data('student.csv')[0]
        student_info = [student.email_address, student.first_name, student.last_name, student.course_ids, student.
        exists = students_ll.check_for_data_in_ll(self.email_address)

        if students_ll.size() == 0:
            students_ll.add_first(student_info)
            add_to_file('student.csv', student_info)
            print(f'{student.first_name} {student.last_name} successfully added! (First student added)')

        if not exists:
            students_ll.add_new(student_info)
            add_to_file('student.csv', student_info)
```

```
4. Logout
Enter what you would like to do: 1
C:\Users\pakin\OneDrive\Desktop\DATA 200\Assignments\Data200-Lab1\checkmygrade_main.py
                                        ================================
                                            Welcome to CheckMyGrade
                                            Admin Portal (Student Screen)
                                        ================================

1. Display all students
2. Sort students
3. Search students
4. Get all grades
5. Add student
6. Modify student grade/mark
7. Delete student
8. Go back
Enter what you would like to do: 1
Retrieving all students...
Displaying students...

                mae@myschool.edu
                Samantha Mae
                Course(s): ['DATA200', 'DATA201']
                Grade(s): ['B+', 'A-']
                Mark(s): ['88', '92']


                russell@myschool.edu
                Sally Russell
                Course(s): ['DATA200']
                Grade(s): ['A-']
                Mark(s): ['90']


                doe@myschool.edu
                John Doe
                Course(s): ['DATA201', 'DATA203']
                Grade(s): ['A', 'B+']
                Mark(s): ['94', '89']


                smart@myschool.edu
                Jane Smart
                Course(s): ['DATA200', 'DATA201', 'DATA202']
                Grade(s): ['B+', 'A-', 'A']
                Mark(s): ['88', '90', '96']


                jones@myschool.edu
                Emily Jones
                Course(s): ['DATA203']
                Grade(s): ['A-']
                Mark(s): ['92']


                brown@myschool.edu
```

1. Display all students (displays all students in the database)

Some students are taking more than one course, so their grades and marks are shown for all courses



```
    def check_my_marks(self):
        '''lets student check their own marks'''
        # array method
        student_list = get_data('student.csv')[1]
        for student in student_list:
            if student[0] == self.email_address:
                course_list = self.course_ids.split('|')
                if self.course_ids:
                    print(f'{self.email_address} is currently taking {len(course_list)} courses.')
                    print('Showing marks...')
                    marks_list = self.marks.split('|')
                    for idx, course in enumerate(course_list):
                        print(f"{course}: {marks_list[idx]}%")
                else:
                    print(f'{self.email_address} currently has no marks.')

    @staticmethod
    def sort_students(by, order):
        '''sorts students either by email or grade'''
        start_time = time.time()
        student_list = get_data('student.csv')[1]
        grades_list = Grades.get_grades()
        sorted_list = sorted(student_list, key = lambda x: x[0]) if by == 'email' else sorted(grades_list, key = l
        if order == 'reverse':
            sorted_list = sorted_list[::-1]
        for data in sorted_list:
            print(data)
        print(f'Sorting time: {time.time() - start_time} seconds')

class Course:
    '''course class that includes information regarding a course's id, credits, name'''
    def __init__(self, course_id, credits, course_name, course_description):
        self.course_id = course_id
        self.credits = credits
        self.course_name = course_name
        self.course_description = course_description

    @staticmethod # static to call out of course object
    def display_courses():
        '''displays all courses'''
        print('Displaying courses...')
        courses_list = get_data('course.csv')[1]
        for course in courses_list:
            print(f'''
                {course[0]}
                Name: {course[1]}
                Credits: {course[2]}
                Description: {course[3]}
```

```
                Jack Johnson
                Course(s): ['DATA200']
                Grade(s): ['C']
                Mark(s): ['75']

C:\Users\pakin\OneDrive\Desktop\DATA 200\Assignments\Data200-Lab1\checkmygrade_main.py
                                        ================================
                                            Welcome to CheckMyGrade
                                            Admin Portal (Student Screen)
                                        ================================

1. Display all students
2. Sort students
3. Search students
4. Get all grades
5. Add student
6. Modify student grade/mark
7. Delete student
8. Go back
Enter what you would like to do: 2
Sort students...
How would you like to sort?
                                            1. By student ID (email address)
                                            2. By student ID (email address) (reversed)
                                            3. By grade
                                            4. By grade (descending)

Enter number of how you would like to sort: 1
['anderson@myschool.edu', 'Benjamin', 'Anderson', 'DATA201', 'C', '75']
['brown@myschool.edu', 'Michael', 'Brown', 'DATA200|DATA203', 'B|C+', '86|78']
['davis@myschool.edu', 'Sarah', 'Davis', 'DATA202|DATA203', 'A|B+', '97|88']
['doe@myschool.edu', 'John', 'Doe', 'DATA201|DATA203', 'A|B+', '94|89']
['johnson@myschool.edu', 'Jack', 'Johnson', 'DATA200', 'C', '75']
['jones@myschool.edu', 'Emily', 'Jones', 'DATA203', 'A-', '92']
['mae@myschool.edu', 'Samantha', 'Mae', 'DATA200|DATA201', 'B+|A-', '88|92']
['miller@myschool.edu', 'William', 'Miller', 'DATA201|DATA202', 'C-|D+', '72|68']
['moore@myschool.edu', 'Charles', 'Moore', 'DATA200', 'A', '94']
['russell@myschool.edu', 'Sally', 'Russell', 'DATA200', 'A-', '90']
['smart@myschool.edu', 'Jane', 'Smart', 'DATA200|DATA201|DATA202', 'B+|A-|A', '88|90|96']
['taylor@myschool.edu', 'Olivia', 'Taylor', 'DATA204', 'B+', '88']
['wilson@myschool.edu', 'Lucy', 'Wilson', 'DATA202', 'B-', '80']
Sorting time: 0.00463175773620605 seconds
C:\Users\pakin\OneDrive\Desktop\DATA 200\Assignments\Data200-Lab1\checkmygrade_main.py
                                        ================================
                                            Welcome to CheckMyGrade
                                            Admin Portal (S
                                        ================================

1. Display all students
2. Sort students
3. Search students
4. Get all grades
5. Add student
6. Modify student grade/mark
7. Delete student
8. Go back
Enter what you would like to do:
```

2. Sort students (sorts students depending on the chosen sort criteria)

Option 1: Students sorted in alphabetical order by their email address

```
Enter number of how you would like to sort: 2
['wilson@myschool.edu', 'Lucy', 'Wilson', 'DATA202', 'B-', '80']
['taylor@myschool.edu', 'Olivia', 'Taylor', 'DATA204', 'B+', '88']
['smart@myschool.edu', 'Jane', 'Smart', 'DATA200|DATA201|DATA202', 'B+|A-|A', '88|90|96']
['russell@myschool.edu', 'Sally', 'Russell', 'DATA200', 'A-', '90']
['moore@myschool.edu', 'Charles', 'Moore', 'DATA200', 'A', '94']
['miller@myschool.edu', 'William', 'Miller', 'DATA201|DATA202', 'C-|D+', '72|68']
['mae@myschool.edu', 'Samantha', 'Mae', 'DATA200|DATA201', 'B+|A-', '88|92']
['jones@myschool.edu', 'Emily', 'Jones', 'DATA203', 'A-', '92']
['johnson@myschool.edu', 'Jack', 'Johnson', 'DATA200', 'C', '75']
['doe@myschool.edu', 'John', 'Doe', 'DATA201|DATA203', 'A|B+', '94|89']
['davis@myschool.edu', 'Sarah', 'Davis', 'DATA202|DATA203', 'A|B+', '97|88']
['brown@myschool.edu', 'Michael', 'Brown', 'DATA200|DATA203', 'B|C+', '86|78']
['anderson@myschool.edu', 'Benjamin', 'Anderson', 'DATA201', 'C', '75']
Sorting time: 0.004913806915283203 seconds
```

Option 2: Students sorted in descending alphabetical order

Option 3: Students sorted in ascending grade order

```
Enter number of how you would like to sort: 3
['miller@myschool.edu', 'D+', '68', 'DATA202']
['miller@myschool.edu', 'C-', '72', 'DATA201']
['anderson@myschool.edu', 'C', '75', 'DATA201']
['johnson@myschool.edu', 'C', '75', 'DATA200']
['brown@myschool.edu', 'C+', '78', 'DATA203']
['wilson@myschool.edu', 'B-', '80', 'DATA202']
['brown@myschool.edu', 'B', '86', 'DATA200']
['mae@myschool.edu', 'B+', '88', 'DATA200']
['smart@myschool.edu', 'B+', '88', 'DATA200']
['davis@myschool.edu', 'B+', '88', 'DATA203']
['taylor@myschool.edu', 'B+', '88', 'DATA204']
['doe@myschool.edu', 'B+', '89', 'DATA203']
['russell@myschool.edu', 'A-', '90', 'DATA200']
['smart@myschool.edu', 'A-', '90', 'DATA201']
['mae@myschool.edu', 'A-', '92', 'DATA201']
['jones@myschool.edu', 'A-', '92', 'DATA203']
['doe@myschool.edu', 'A', '94', 'DATA201']
['moore@myschool.edu', 'A', '94', 'DATA200']
['smart@myschool.edu', 'A', '96', 'DATA202']
['davis@myschool.edu', 'A', '97', 'DATA202']
Sorting time: 0.004064083099365234 seconds
```

```
Enter number of how you would like to sort: 4
['davis@myschool.edu', 'A', '97', 'DATA202']
['smart@myschool.edu', 'A', '96', 'DATA202']
['moore@myschool.edu', 'A', '94', 'DATA200']
['doe@myschool.edu', 'A', '94', 'DATA201']
['jones@myschool.edu', 'A-', '92', 'DATA203']
['mae@myschool.edu', 'A-', '92', 'DATA201']
['smart@myschool.edu', 'A-', '90', 'DATA201']
['russell@myschool.edu', 'A-', '90', 'DATA200']
['doe@myschool.edu', 'B+', '89', 'DATA203']
['taylor@myschool.edu', 'B+', '88', 'DATA204']
['davis@myschool.edu', 'B+', '88', 'DATA203']
['smart@myschool.edu', 'B+', '88', 'DATA200']
['mae@myschool.edu', 'B+', '88', 'DATA200']
['brown@myschool.edu', 'B', '86', 'DATA200']
['wilson@myschool.edu', 'B-', '80', 'DATA202']
['brown@myschool.edu', 'C+', '78', 'DATA203']
['johnson@myschool.edu', 'C', '75', 'DATA200']
['anderson@myschool.edu', 'C', '75', 'DATA201']
['miller@myschool.edu', 'C-', '72', 'DATA201']
['miller@myschool.edu', 'D+', '68', 'DATA202']
Sorting time: 0.005667448043823242 seconds
```

Option 4: Students sorted in descending grade order

```python
def add_student():
    '''gets student details for add_new_student()'''
    first_name = input('Enter first name of student: ')
    last_name = input('Enter last name of student: ')
    email_address = input('Enter email of student: ')
    course_ids = input('Enter the course that the student is enrolled in: ')
    grades = input('Enter the grade the student has in the course: ')
    marks = input('Enter the mark in integer form of the student: ')
    return first_name.strip(), last_name.strip(), email_address.strip(), course_ids.strip(), grades.strip(), marks
```

```python
def get_student(email_address):
    '''get one student's details'''
    start_time = time.time()
    student_list = get_data('student.csv')[1]
    for student in student_list:
        if student[0] == email_address:
            current_student = Student(student[1], student[2], student[0], student[3], student[4], student[5])
            print(f'Search time: {time.time() - start_time} seconds')
            return current_student
    else:
        print('No student found with the email!')
        return None
```

```python
def get_professor(email_address):
    '''get one professor's details'''
    start_time = time.time()
    professor_list = get_data('professor.csv')[1]
    for professor in professor_list:
        if professor[0] == email_address:
            current_professor = Professor(professor[0], professor[1], professor[2], professor[3])
            print(f'Search time: {time.time() - start_time} seconds')
            return current_professor
    else:
        print('No professor found with the email!')
        return None
```

```
C:\Users\pakin\OneDrive\Desktop\DATA 200\Assignments\Data200-Lab1\checkmygrade_main.py
==================================
Welcome to CheckMyGrade
Admin Portal (Student Screen)
==================================

1. Display all students
2. Sort students
3. Search students
4. Get all grades
5. Add student
6. Modify student grade/mark
7. Delete student
8. Go back
Enter what you would like to do: 3
Search for student...
Enter student email: mae@myschool.edu
Search time: 0.0006573200225830078 seconds
Displaying information for Samantha Mae
mae@myschool.edu is currently taking 2 courses.
DATA200
B+ (88%)
Above average (86.83333333333333%)
DATA201
A- (92%)
Above average (84.6%)
C:\Users\pakin\OneDrive\Desktop\DATA 200\Assignments\Data200-Lab1\checkmygrade_main.py
==================================
Welcome to CheckMyGrade
Admin Portal (Student Screen)
==================================

1. Display all students
2. Sort students
3. Search students
4. Get all grades
5. Add student
6. Modify student grade/mark
7. Delete student
8. Go back
Enter what you would like to do:
```

3. Search students (returns a searched student's current marks and evaluates whether they are above/below the course average)

In this instance, mae@myschool.edu is used

```python
class Grades:
    '''grades class that includes information regarding a specific grade for a student'''
    def __init__(self, grade_id, grade, mark, course_id = None):
        self.grade_id = grade_id # email address to match with the student it is assigned to
        self.grade = grade
        self.mark = mark

    @staticmethod
    def get_grades():
        students = get_data('student.csv')[1]
        grade_data = []
        for student in students:
            student_courses = student[3].split('|')
            student_grades = student[4].split('|')
            student_marks = student[5].split('|')
            for idx, course in enumerate(student_courses):
                grade_data.append([student[0], student_grades[idx], student_marks[idx], course])
        return grade_data

    @staticmethod
    def display_grade_report():
        '''displays a report of all the grades for students'''
        grades = Grades.get_grades()
        for grade in grades:
            grade_id = grade[0]
            grade_id = Grades(grade[0], grade[1], grade[2], grade[3])
            print(f'{grade_id.grade_id}: {grade_id.grade} / {grade_id.mark}% / {grade[3]}')

    @staticmethod
    def display_grades_by_course(course):
        start_time = time.time()
        grades = Grades.get_grades()
        course_grades = []
        for grade in grades:
            if grade[3] == course:
                course_grades.append(grade[2])
                print(f'{grade[0]}: {grade[1]} ({grade[2]}%)')

        print(f'There are {len(course_grades)} students in {course}.\n')
        print(f'Search time: {time.time() - start_time} seconds')

    @staticmethod
    def get_course_average(course):
        course_grades = []
        grades = Grades.get_grades()
        for grade in grades:
```

```
DATA200
B+ (88%)
Above average (86.83333333333333%)
DATA201
A- (92%)
Above average (84.6%)
C:\Users\pakin\OneDrive\Desktop\DATA 200\Assignments\Data200-Lab1\checkmygrade_main.py
==================================
Welcome to CheckMyGrade
Admin Portal (Student Screen)
==================================

1. Display all students
2. Sort students
3. Search students
4. Get all grades
5. Add student
6. Modify student grade/mark
7. Delete student
8. Go back
Enter what you would like to do: 4
Displaying all grades...
mae@myschool.edu: B+ / 88% / DATA200
mae@myschool.edu: A- / 92% / DATA201
russell@myschool.edu: A- / 90% / DATA200
doe@myschool.edu: A / 94% / DATA201
doe@myschool.edu: B+ / 89% / DATA203
smart@myschool.edu: B+ / 88% / DATA200
smart@myschool.edu: A- / 90% / DATA201
smart@myschool.edu: A / 96% / DATA202
jones@myschool.edu: A- / 92% / DATA203
brown@myschool.edu: B / 86% / DATA200
brown@myschool.edu: C+ / 78% / DATA203
davis@myschool.edu: A / 97% / DATA202
davis@myschool.edu: B+ / 88% / DATA203
miller@myschool.edu: C- / 72% / DATA201
miller@myschool.edu: D+ / 68% / DATA202
wilson@myschool.edu: B- / 80% / DATA202
moore@myschool.edu: A / 94% / DATA200
taylor@myschool.edu: B+ / 88% / DATA204
anderson@myschool.edu: C / 75% / DATA201
johnson@myschool.edu: C / 75% / DATA200
C:\Users\pakin\OneDrive\Desktop\DATA 200\Assignments\Data200-Lab1\checkmygrade_main.py
==================================
Welcome to CheckMyGrade
Admin Portal (Student Screen)
==================================

1. Display all students
2. Sort students
3. Search students
4. Get all grades
5. Add student
6. Modify student grade/mark
7. Delete student
8. Go back
Enter what you would like to do:
```

4. Get all grades (displays all grades in the system)

Some student emails are listed more than once, because that student takes more than one course

```
def display_records():
        marks_list = student[5].split('|')
        print(f'''
            {student[0]}
            {student[1]} {student[2]}
            Course(s): {course_list}
            Grade(s): {grades_list}
            Mark(s): {marks_list}
            ''')

def add_new_student(self, student): # student from add_student()
    '''add a new student into the system'''
    # linked list method
    print(f'Checking system to add {student.first_name} {student.last_name}...')
    students_ll = get_data('student.csv')[0]
    student_info = [student.email_address, student.first_name, student.last_name, student.course_ids, student.
    exists = students_ll.check_for_data_in_ll(self.email_address)

    if students_ll.size() == 0:
        students_ll.add_first(student_info)
        add_to_file('student.csv', student_info)
        print(f'{student.first_name} {student.last_name} successfully added! (First student added)')

    if not exists:
        students_ll.add_new(student_info)
        add_to_file('student.csv', student_info)
        print(f'{student.first_name} {student.last_name} successfully added! (Another student added)')

def delete_student(self, email_address):
    '''delete a student in the system using their email address'''
    # linked list method
    students_ll, student_list, header = get_data('student.csv')
    try:
        print(f'Checking system to delete student associated with email {email_address}...')
        students_ll.delete_node(email_address)
        write_to_file_ll('student.csv', header, students_ll)
    except Exception as e:
        print(f'Error deleting student: {e}!')

def check_my_grades(self):
    '''lets student check their own grades'''
    # array method
    student_list = get_data('student.csv')[1]
    for student in student_list:
        if student[0] == self.email_address:
            course_list = self.course_ids.split('|')
            if self.course_ids:
```

```
wilson@myschool.edu: B- / 80% / DATA202
moore@myschool.edu: A / 94% / DATA200
taylor@myschool.edu: B+ / 88% / DATA204
anderson@myschool.edu: C / 75% / DATA201
johnson@myschool.edu: C / 75% / DATA200
C:\Users\pakin\OneDrive\Desktop\DATA 200\Assignments\Data200-Lab1\checkmygrade_main.py
==========
Welcome to CheckMyGrade
Admin Portal (Student Screen)
==========
1. Display all students
2. Sort students
3. Search students
4. Get all grades
5. Add student
6. Modify student grade/mark
7. Delete student
8. Go back
Enter what you would like to do: 5
Adding student...
Enter first name of student: Liana
Enter last name of student: Pak
Enter email of student: pak@myschool.edu
Enter the course that the student is enrolled in: DATA201
Enter the grade the student has in the course: A
Enter the mark in integer form of the student: 96
Checking system to add Liana Pak...
Liana Pak successfully added! (Another student added)
C:\Users\pakin\OneDrive\Desktop\DATA 200\Assignments\Data200-Lab1\checkmygrade_main.py
==========
Welcome to CheckMyGrade
Admin Portal (Student Screen)
==========
1. Display all students
2. Sort students
3. Search students
4. Get all grades
5. Add student
6. Modify student grade/mark
7. Delete student
8. Go back
Enter what you would like to do: 3
Search for student...
Enter student email: pak@myschool.edu
Search time: 0.0007464885711669922 seconds
Displaying information for Liana Pak
pak@myschool.edu is currently taking 1 courses.
DATA201
A (96%)
Above average (86.5%)
C:\Users\pakin\OneDrive\Desktop\DATA 200\Assignments\Data200-Lab1\checkmygrade_main.py
==========
Welcome to CheckMyGrade
Admin Portal (Student Screen)
==========
```

5. Add student (adds a new student into the system – linked list)

Checking to see if new student Liana was added successfully by searching

From student.csv

```
13  anderson@myschool.edu,Benjamin,Anderson,DATA2
14  johnson@myschool.edu,Jack,Johnson,DATA200,C,7
15  pak@myschool.edu,Liana,Pak,DATA201,A,96
16
```

```
def check_my_grades(self):
    '''lets student check their own grades'''
    # array method
    student_list = get_data('student.csv')[1]
    for student in student_list:
        if student[0] == self.email_address:
            course_list = self.course_ids.split('|')
            if self.course_ids:
                print(f'{self.email_address} is currently taking {len(course_list)} courses.')
                print('Showing grades...')
                grades_list = self.grades.split('|')
                for idx, course in enumerate(course_list):
                    print(f"{course}: {grades_list[idx]}")
            else:
                print(f'{self.email_address} currently has no grades.')

def update_student_record(self, new_grade, new_mark):
    '''updates a student's record by using their email address'''
    # array method
    students_ll, student_list, header = get_data('student.csv')
    for student in student_list:
        if student[0] == self.email_address:
            student[4] = self.grades = new_grade
            student[5] = self.marks = new_mark
    write_to_file_array('student.csv', header, student_list)
    print(f'The grade and mark for {self.first_name} {self.last_name} has been updated to {new_grade} ({new_ma

def check_my_marks(self):
    '''lets student check their own marks'''
    # array method
    student_list = get_data('student.csv')[1]
    for student in student_list:
        if student[0] == self.email_address:
            course_list = self.course_ids.split('|')
            if self.course_ids:
                print(f'{self.email_address} is currently taking {len(course_list)} courses.')
                print('Showing marks...')
                marks_list = self.marks.split('|')
                for idx, course in enumerate(course_list):
                    print(f"{course}: {marks_list[idx]}%")
            else:
                print(f'{self.email_address} currently has no marks.')

@staticmethod
def sort_students(by, order):
```

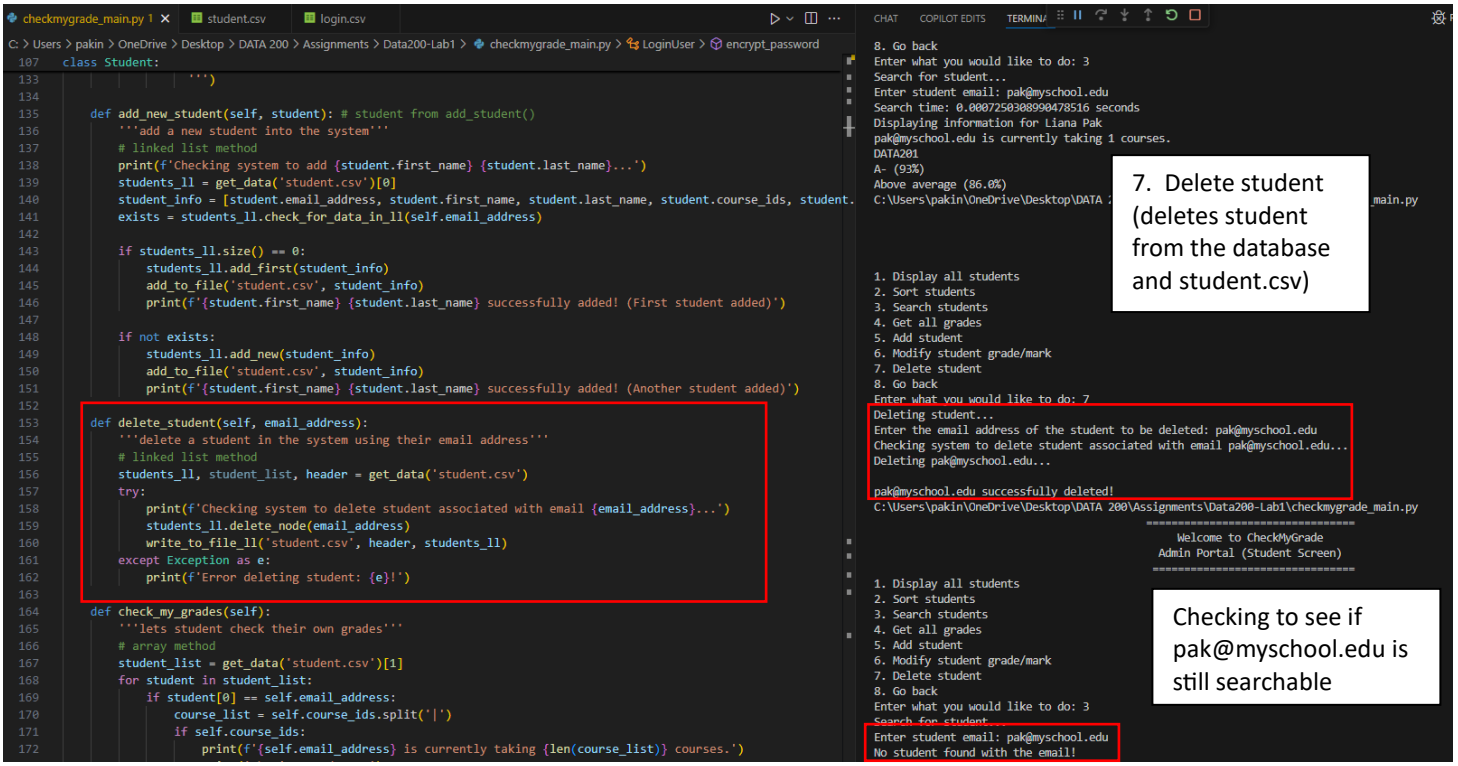
```
Enter student email: pak@myschool.edu
Search time: 0.0007464885711669922 seconds
Displaying information for Liana Pak
pak@myschool.edu is currently taking 1 courses.
DATA201
A (96%)
Above average (86.5%)
C:\Users\pakin\OneDrive\Desktop\DATA 200\Assig
==
A
==
1. Display all students
2. Sort students
3. Search students
4. Get all grades
5. Add student
6. Modify student grade/mark
7. Delete student
8. Go back
Enter what you would like to do: 6
Modifying student...
Enter the email of the student you would like to edit: pak@myschool.edu
Search time: 0.0004749288095703125 seconds
The current grade and mark for Liana Pak is A (96%)
Enter the new mark for Liana Pak: 93
Enter the new grade for Liana Pak: A-
The grade and mark for Liana Pak has been updated to A- (93%)
C:\Users\pakin\OneDrive\Desktop\DATA 200\Assignments\Data200-Lab1\checkmygrade_main.py
==========
Welcome to CheckMyGrade
Admin Portal (Student Screen)
==========
1. Display all students
2. Sort students
3. Search students
4. Get all grades
5. Add student
6. Modify student grade/mark
7. Delete student
8. Go back
Enter what you would like to do: 3
Search for student...
Enter student email: pak@myschool.edu
Search time: 0.0007250308990478516 seconds
Displaying information for Liana Pak
pak@myschool.edu is currently taking 1 courses.
DATA201
A- (93%)
Above average (86.0%)
C:\Users\pakin\OneDrive\Desktop\DATA 200\Assignments\Data200-Lab1\checkmygrade_main.py
==========
Welcome to CheckMyGrade
Admin Portal (Student Screen)
```

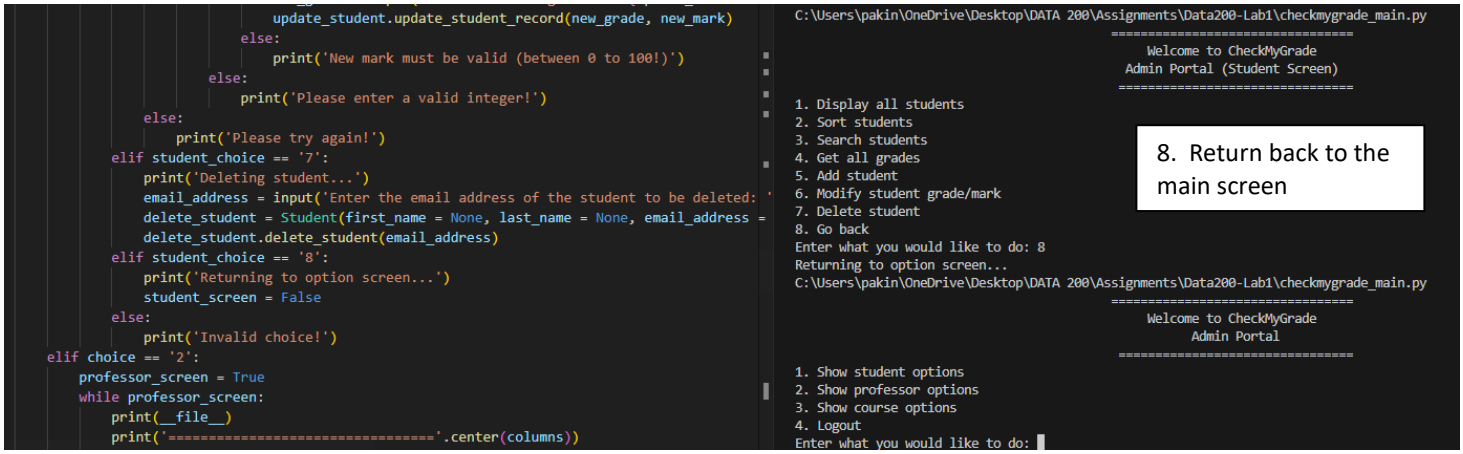
6. Modify student (modifies student record with new mark and grade)

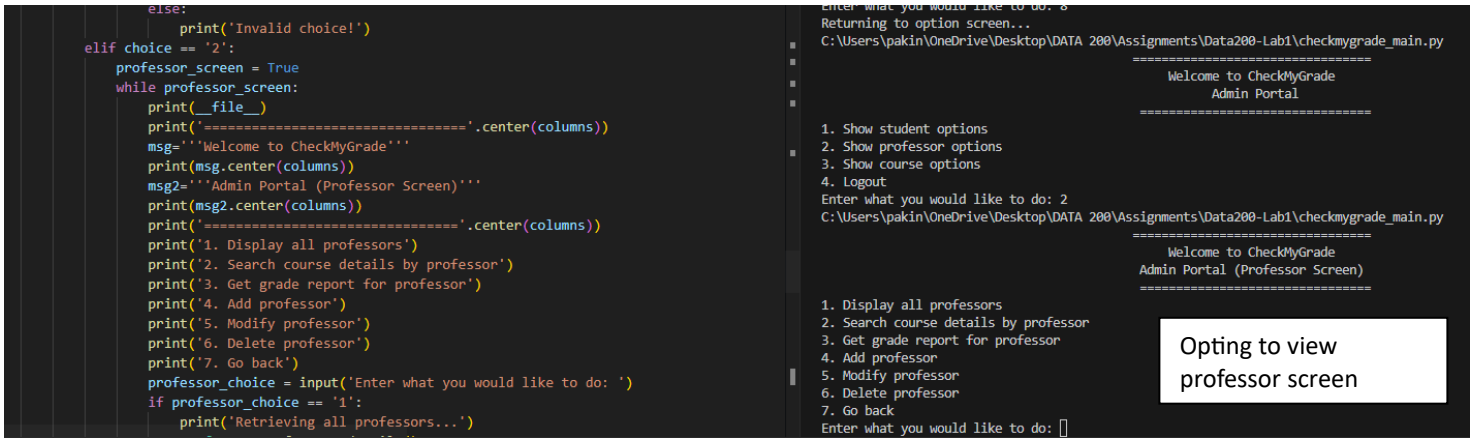
Checking to see if Liana's mark and grade were changed

```
14  johnson@myschool.edu,Jack,Johnson,DATA200,C,75
15  pak@myschool.edu,Liana,Pak,DATA201,A-,93
16
```

From student.csv - the grade and mark updated


checkmygrade_main.py 1
student.csv
login.csv
C: > Users > pakin > OneDrive > Desktop > DATA 200 > Assignments > Data200-Lab1 > checkmygrade_main.py > LoginUser > encrypt_password
class Student:
def add_new_student(self, student): # student from add_student()
'''add a new student into the system'''
# linked list method
print(f'Checking system to add {student.first_name} {student.last_name}...')
students_ll = get_data('student.csv')[0]
student_info = [student.email_address, student.first_name, student.last_name, student.course_ids, student.
exists = students_ll.check_for_data_in_ll(self.email_address)
if students_ll.size() == 0:
students_ll.add_first(student_info)
add_to_file('student.csv', student_info)
print(f'{student.first_name} {student.last_name} successfully added! (First student added)')
if not exists:
students_ll.add_new(student_info)
add_to_file('student.csv', student_info)
print(f'{student.first_name} {student.last_name} successfully added! (Another student added)')
def delete_student(self, email_address):
'''delete a student in the system using their email address'''
linked list method
students_ll, student_list, header = get_data('student.csv')
try:
print(f'Checking system to delete student associated with email {email_address}...')
students_ll.delete_node(email_address)
write_to_file_ll('student.csv', header, students_ll)
except Exception as e:
print(f'Error deleting student: {e}!')
def check_my_grades(self):
'''lets student check their own grades'''
# array method
student_list = get_data('student.csv')[1]
for student in student_list:
if student[0] == self.email_address:
course_list = self.course_ids.split('|')
if self.course_ids:
print(f'{self.email_address} is currently taking {len(course_list)} courses.')
CHAT
COPILOT EDITS
TERMINAL
8. Go back
Enter what you would like to do: 3
Search for student...
Enter student email: pak@myschool.edu
Search time: 0.0007250308990478516 seconds
Displaying information for Liana Pak
pak@myschool.edu is currently taking 1 courses.
DATA201
A- (93%)
Above average (86.0%)
C:\Users\pakin\OneDrive\Desktop\DATA 200\Assignments\Data200-Lab1\checkmygrade_main.py
7. Delete student (deletes student from the database and student.csv)
1. Display all students
2. Sort students
3. Search students
4. Get all grades
5. Add student
6. Modify student grade/mark
7. Delete student
8. Go back
Enter what you would like to do: 7
Deleting student...
Enter the email address of the student to be deleted: pak@myschool.edu
Checking system to delete student associated with email pak@myschool.edu...
Deleting pak@myschool.edu...
pak@myschool.edu successfully deleted!
C:\Users\pakin\OneDrive\Desktop\DATA 200\Assignments\Data200-Lab1\checkmygrade_main.py
Welcome to CheckMyGrade
Admin Portal (Student Screen)
1. Display all students
2. Sort students
3. Search students
4. Get all grades
5. Add student
6. Modify student grade/mark
7. Delete student
8. Go back
Enter what you would like to do: 3
Enter student email: pak@myschool.edu
No student found with the email!
Checking to see if pak@myschool.edu is still searchable


update_student.update_student_record(new_grade, new_mark)
print('New mark must be valid (between 0 to 100!)')
print('Please enter a valid integer!')
print('Please try again!')
elif student_choice == '7':
print('Deleting student...')
email_address = input('Enter the email address of the student to be deleted:
delete_student = Student(first_name = None, last_name = None, email_address =
delete_student.delete_student(email_address)
elif student_choice == '8':
print('Returning to option screen...')
student_screen = False
print('Invalid choice!')
elif choice == '2':
professor_screen = True
while professor_screen:
print(__file__)
print('================================'.center(columns))
C:\Users\pakin\OneDrive\Desktop\DATA 200\Assignments\Data200-Lab1\checkmygrade_main.py
Welcome to CheckMyGrade
Admin Portal (Student Screen)
1. Display all students
2. Sort students
3. Search students
4. Get all grades
5. Add student
6. Modify student grade/mark
7. Delete student
8. Go back
Enter what you would like to do: 8
Returning to option screen...
C:\Users\pakin\OneDrive\Desktop\DATA 200\Assignments\Data200-Lab1\checkmygrade_main.py
Welcome to CheckMyGrade
Admin Portal
8. Return back to the main screen
1. Show student options
2. Show professor options
3. Show course options
4. Logout
Enter what you would like to do:


print('Invalid choice!')
elif choice == '2':
professor_screen = True
while professor_screen:
print(__file__)
print('================================'.center(columns))
msg='''Welcome to CheckMyGrade'''
print(msg.center(columns))
msg2='''Admin Portal (Professor Screen)'''
print(msg2.center(columns))
print('================================'.center(columns))
print('1. Display all professors')
print('2. Search course details by professor')
print('3. Get grade report for professor')
print('4. Add professor')
print('5. Modify professor')
print('6. Delete professor')
print('7. Go back')
professor_choice = input('Enter what you would like to do: ')
if professor_choice == '1':
print('Retrieving all professors...')
Returning to option screen...
C:\Users\pakin\OneDrive\Desktop\DATA 200\Assignments\Data200-Lab1\checkmygrade_main.py
Welcome to CheckMyGrade
Admin Portal
1. Show student options
2. Show professor options
3. Show course options
4. Logout
Enter what you would like to do: 2
C:\Users\pakin\OneDrive\Desktop\DATA 200\Assignments\Data200-Lab1\checkmygrade_main.py
Welcome to CheckMyGrade
Admin Portal (Professor Screen)
1. Display all professors
2. Search course details by professor
3. Get grade report for professor
4. Add professor
5. Modify professor
6. Delete professor
7. Go back
Enter what you would like to do:
Opting to view professor screen

```
checkmygrade_main.py 1 | student.csv | login.csv
C: > Users > pakin > OneDrive > Desktop > DATA 200 > Assignments > Data200-Lab1 > checkmygrade_main.py > LoginUser > encrypt_password
220   class Course:
258       def delete_course(self, id):
264               write_to_file_ll('course.csv', header, courses_ll)
265           except Exception as e:
266               print(f'Error deleting course: {e}!')
267
268   class Professor:
269       '''professor class that includes information regarding a professor's name, email, rank'''
270       def __init__(self, email_address, name, rank, course_id):
271           self.name = name
272           self.email_address = email_address
273           self.rank = rank
274           self.course_id = course_id
275
276       @staticmethod
277       def professors_details():
278           '''displays all professors'''
279           print('Displaying professor details...')
280           professors_list = get_data('professor.csv')[1]
281           for professor in professors_list:
282               print(f'''
283                     {professor[0]}
284                     {professor[1]}
285                     Rank: {professor[2]}
286                     Currently Teaching: {professor[3]}
287                     ''')
288
289       def add_new_professor(self, professor): # professor from add_professor()
290           '''add a new professor into the system'''
291           print(f'Checking system to add {professor.name}...')
292           professors_ll = get_data('professor.csv')[0]
293           professor_info = [professor.email_address, professor.name, professor.rank, professor.course_id]
294           exists = professors_ll.check_for_data_in_ll(self.email_address)
295
296           if professors_ll.size() == 0:
297               professors_ll.add_first(professor_info)
298               add_to_file('professor.csv', professor_info)
299               print(f'{professor.name} successfully added! (First professor added)')
300
301           if not exists:
302               professors_ll.add_new(professor_info)
303               add_to_file('professor.csv', professor_info)
304               print(f'{professor.name} successfully added! (Another professor added)')
305
306       def delete_professor(self, email_address):
307           '''delete a professor using their email address'''
308           professors_ll, professor_list, header = get_data('professor.csv')
309           try:
```

```
CHAT  COPILOT EDITS  TERMINA
3. Show course options
4. Logout
Enter what you would like to do: 2
C:\Users\pakin\OneDrive\Desktop\DATA 200\Assignments\Data200-Lab1\checkmygrade_main.py
                                        ================================
                                              Welcome to CheckMyGrade
                                           Admin Portal (Professor Screen)
                                        ================================

1. Display all professors
2. Search course details by professor
3. Get grade report for professor
4. Add professor
5. Modify professor
6. Delete professor
7. Go back
Enter what you would like to do: 1
Retrieving all professors...
Displaying professor details...

                    smith@myschool.edu
                    John Smith
                    Rank: Tenured Scholar
                    Currently Teaching: DATA200


                    douglas@myschool.edu
                    Ken Douglas
                    Rank: Visiting Lecturer
                    Currently Teaching: DATA201


                    roberts@myschool.edu
                    Dave Roberts
                    Rank: Visiting Scholar
                    Currently Teaching: DATA203


                    gomez@myschool.edu
                    Kerry Gomez
                    Rank: Tenured Lecturer
                    Currently Teaching: DATA202

C:\Users\pakin\OneDrive\Desktop\DATA 200\Assignments\Data200-Lab1\checkmygrade_main.py
                                        ================================
                                              Welcome to CheckMyGrade
                                           Admin Portal (Professor Screen)
                                        ================================

1. Display all professors
2. Search course details by professor
3. Get grade report for professor
4. Add professor
5. Modify professor
6. Delete professor
7. Go back
Enter what you would like to do:
```

1. Display all professors (displays all professors in database)

```
    for student in student_list:
        if student[0] == email_address:
            current_student = Student(student[1], student[2], student[0], student[3], student[4], student[5])
            print(f'Search time: {time.time() - start_time} seconds')
            return current_student
    else:
        print('No student found with the email!')
        return None

def get_professor(email_address):
    '''get one professor's details'''
    start_time = time.time()
    professor_list = get_data('professor.csv')[1]
    for professor in professor_list:
        if professor[0] == email_address:
            current_professor = Professor(professor[0], professor[1], professor[2], professor[3])
            print(f'Search time: {time.time() - start_time} seconds')
            return current_professor
    else:
        print('No professor found with the email!')
        return None

def add_course():
    '''gets course details for add_new_course()'''
    course_id = input('Enter id of the course: ') # i.e. DATA200
    str_credits = input('Enter the number of credits of the course: ')
    course_name = input('Enter the name of the course: ')
    course_description = input('Enter the description of the course: ')
    try:
        credits = int(str_credits)
        if credits > 0:
            return course_id.strip(), credits, course_name.strip(), course_description.strip()
        else:
```

```
C:\Users\pakin\OneDrive\Desktop\DATA 200\Assignments\Data200-Lab1\checkmygrade_main.py
                                        ================================

                                        Adm
                                        ====

1. Display all professors
2. Search course details by professor
3. Get grade report for professor
4. Add professor
5. Modify professor
6. Delete professor
7. Go back
Enter what you would like to do: 2
Enter email of the professor to get their course details: smith@myschool.edu
Search time: 0.0005903244018554688 seconds

                                Showing course details for John Smith:
                                ID: DATA200
                                Name: Intro to Data Science
                                Credits: 3
                                Description: Utilizing Python in Data Science

Search time: 0.0017418861389160156 seconds
C:\Users\pakin\OneDrive\Desktop\DATA 200\Assignments\Data200-Lab1\checkmygrade_main.py
                                        ================================

                                        Admi
                                        ====

1. Display all professors
2. Search course details by professor
3. Get grade report for professor
4. Add professor
5. Modify professor
6. Delete professor
7. Go back
Enter what you would like to do:
```

2. Search course details by professor (returns course details of professor)

Getting course details for smith@myschool.edu

checkmygrade_main.py 1 | student.csv | login.csv

C: > Users > pakin > OneDrive > Desktop > DATA 200 > Assignments > Data200-Lab1 > checkmygrade_main.py > LoginUser > encrypt_password

```
345  class Grades:
362          return grade_data
363
364      @staticmethod
365      def display_grade_report():
366          '''displays a report of all the grades for students'''
367          grades = Grades.get_grades()
368          for grade in grades:
369              grade_id = grade[0]
370              grade_id = Grades(grade[0], grade[1], grade[2], grade[3])
371              print(f'{grade_id.grade_id}: {grade_id.grade} / {grade_id.mark}% / {grade[3]}')
372
373      @staticmethod
374      def display_grades_by_course(course):
375          start_time = time.time()
376          grades = Grades.get_grades()
377          course_grades = []
378          for grade in grades:
379              if grade[3] == course:
380                  course_grades.append(grade[2])
381                  print(f'{grade[0]}: {grade[1]} ({grade[2]}%)')
382
383          print(f'There are {len(course_grades)} students in {course}.\n')
384          print(f'Search time: {time.time() - start_time} seconds')
385
386      @staticmethod
387      def get_course_average(course):
388          course_grades = []
389          grades = Grades.get_grades()
390          for grade in grades:
391                  if grade[3] == course:
392                      course_grades.append(int(grade[2]))
393          class_average = sum(course_grades) / len(course_grades)
394          return class_average
395
396      @staticmethod
397      def get_course_median(course):
398          course_grades = []
399          grades = Grades.get_grades()
400          for grade in grades:
401                  if grade[3] == course:
402                      course_grades.append(int(grade[2]))
403          class_median = statistics.median(course_grades)
404          return class_median
405
406  class Node:
407      def __init__(self, data):
408          self.data = data
```

CHAT | COPILOT EDITS | TERMINAL

```
2. Search course details by professor
3. Get grade report for professor
4. Add professor
5. Modify professor
6. Delete professor
7. Go back
Enter what you would like to do: 2
Enter email of the professor to get their course details: smith@myschool.edu
Search time: 0.0005903244018554688 seconds

                                    Showing course details for John Smith:
                                    ID: DATA200
                                    Name: Intro to Data Science
                                    Credits: 3
                                    Description: Uti

Search time: 0.0017418861389160156 seconds
C:\Users\pakin\OneDrive\Desktop\DATA 200\Assignm
                                    ====
                                    Admi
                                    ====
1. Display all professors
2. Search course details by professor
3. Get grade report for professor
4. Add professor
5. Modify professor
6. Delete professor
7. Go back
Enter what you would like to do: 3
Enter email of the professor to get their course grade report: smith@myschool.edu
Search time: 0.0008499622344970703 seconds
mae@myschool.edu: B+ (88%)
russell@myschool.edu: A- (90%)
smart@myschool.edu: B+ (88%)
brown@myschool.edu: B (86%)
moore@myschool.edu: A (94%)
johnson@myschool.edu: C (75%)
There are 6 students in DATA200.

Search time: 0.0026488304138183594 seconds
The average for John Smith's course is 86.83333333333333%
C:\Users\pakin\OneDrive\Desktop\DATA 200\Assignments\Data200-Lab1\checkmygrade_main.py
                                    ==============================
                                    Welcome to CheckMyGrade
                                    Admin Portal (Professor Screen)
                                    ==============================
1. Display all professors
2. Search course details by professor
3. Get grade report for professor
4. Add professor
5. Modify professor
6. Delete professor
7. Go back
Enter what you would like to do:
```

3. Get grade report for professor (returns a report of all student grades and the professor's course's average)

```
            Currently Teaching: {professor[3]}
            ''')

def add_new_professor(self, professor): # professor from add_professor()
    '''add a new professor into the system'''
    print(f'Checking system to add {professor.name}...')
    professors_ll = get_data('professor.csv')[0]
    professor_info = [professor.email_address, professor.name, professor.rank, professor.course_id]
    exists = professors_ll.check_for_data_in_ll(self.email_address)

    if professors_ll.size() == 0:
        professors_ll.add_first(professor_info)
        add_to_file('professor.csv', professor_info)
        print(f'{professor.name} successfully added! (First professor added)')

    if not exists:
        professors_ll.add_new(professor_info)
        add_to_file('professor.csv', professor_info)
        print(f'{professor.name} successfully added! (Another professor added)')

def delete_professor(self, email_address):
    '''delete a professor using their email address'''
    professors_ll, professor_list, header = get_data('professor.csv')
    try:
        print(f'Checking system to delete professor associated with the email {email_address}...')
        professors_ll.delete_node(email_address)
        write_to_file_ll('professor.csv', header, professors_ll)
    except Exception as e:
        print(f'Error deleting professor: {e}!')
```

```
C:\Users\pakin\OneDrive\Desktop\DATA 200\Assignments\Data200-Lab1\checkmygrade_main.py
                                    ==============================
                                    Welcome to CheckMyGrade
                                    Admin Portal (Prof
                                    ==============
1. Display all professors
2. Search course details by professor
3. Get grade report for professor
4. Add professor
5. Modify professor
6. Delete professor
7. Go back
Enter what you would like to do: 4
Adding professor...
Enter the name of the professor: Jack Sparrow
Enter the email of the professor: sparrow@myschool.edu
Enter the rank of the professor: Amateur Lecturer
Enter the course id of the course the professor teaches: DATA200
Checking system to add Jack Sparrow...
Jack Sparrow successfully added! (Another professor added)
C:\Users\pakin\OneDrive\Desktop\DATA 200\Assignments\Data200-Lab1\checkmygrade_main.py
                                    ==============================
                                    Welcome to CheckMyGrade
                                    Admin Portal (Professor Screen)
                                    ==============================
1. Display all professors
2. Search course details by professor
3. Get grade report for professor
4. Add professor
5. Modify professor
6. Delete professor
7. Go back
```

4. Add professor (add a new professor into the system – linked list)

checkmygrade_main.py 1 | professor.csv | student.csv | login.csv

C: > Users > pakin > OneDrive > Desktop > DATA 200 > Assignments > Data200-Lab1 > professor.csv

```
1  professor_id,professor_name,rank,course_id
2  smith@myschool.edu,John Smith,Tenured Scholar,DATA200
3  douglas@myschool.edu,Ken Douglas,Visiting Lecturer,DATA201
4  roberts@myschool.edu,Dave Roberts,Visiting Scholar,DATA203
5  gomez@myschool.edu,Kerry Gomez,Tenured Lecturer,DATA202
6  sparrow@myschool.edu,Jack Sparrow,Amateur Lecturer,DATA200
7
```

Checking professor.csv for the newly added professor

```
        print(f'{professor.name} successfully added! (Another professor added)')

    def delete_professor(self, email_address):
        '''delete a professor using their email address'''
        professors_ll, professor_list, header = get_data('professor.csv')
        try:
            print(f'Checking system to delete professor associated with the email {email_address}...')
            professors_ll.delete_node(email_address)
            write_to_file_ll('professor.csv', header, professors_ll)
        except Exception as e:
            print(f'Error deleting professor: {e}!')

    def modify_professor_details(self, new_rank):
        '''modify a professor in the system using their email address'''
        professors_ll, professor_list, header = get_data('professor.csv')
        for professor in professor_list:
            if professor[0] == self.email_address:
                professor[2] = self.rank = new_rank
        write_to_file_array('professor.csv', header, professor_list)
        print(f'The rank for {self.name} has been updated to {self.rank}')

    def show_course_details_by_professor(self):
        '''show a professor's course using their email address'''
        start_time = time.time()
        professor_list = get_data('professor.csv')[1]
        course_list = get_data('course.csv')[1]
        if self.email_address:
            for professor in professor_list:
```

```
C:\Users\pakin\OneDrive\Desktop\DATA 200\Assignments\Data200-Lab1\checkmygrade_main.py
================================
      Welcome to CheckMyGrade
Admin Portal (Professor Screen)
================================

1. Display all professors
2. Search course details by professor
3. Get grade report for professor
4. Add professor
5. Modify professor
6. Delete professor
7. Go back
Enter what you would like to do: 5
Update professor details...
Enter the email address of the professor you wish to modify: sparrow@myschool.edu
Search time: 0.00100874900878711 seconds
Enter the new rank of Jack Sparrow: Tenured Lecturer
The rank for Jack Sparrow has been updated to Tenured Lecturer
C:\Users\pakin\OneDrive\Desktop\DATA 200\Assignments\Data200-Lab1\checkmygrade_main.py
================================
      Welcome to CheckMyGrade
Admin Portal (Professor Screen)
================================

1. Display all professors
2. Search course details by professor
3. Get grade report for professor
4. Add professor
5. Modify professor
6. Delete professor
7. Go back
Enter what you would like to do:
```

5. Modify professor (updates the rank of the professor)

checkmygrade_main.py 1 | professor.csv | student.csv | login.csv

C: > Users > pakin > OneDrive > Desktop > DATA 200 > Assignments > Data200-Lab1 > professor.csv

```
1  professor_id,professor_name,rank,course_id
2  smith@myschool.edu,John Smith,Tenured Scholar,DATA200
3  douglas@myschool.edu,Ken Douglas,Visiting Lecturer,DATA201
4  roberts@myschool.edu,Dave Roberts,Visiting Scholar,DATA203
5  gomez@myschool.edu,Kerry Gomez,Tenured Lecturer,DATA202
6  sparrow@myschool.edu,Jack Sparrow,Tenured Lecturer,DATA200
7
```

Checking professor.csv for the updated rank

```
    professors_ll = get_data('professor.csv')[0]
    professor_info = [professor.email_address, professor.name, professor.rank, professor.course_id]
    exists = professors_ll.check_for_data_in_ll(self.email_address)

    if professors_ll.size() == 0:
        professors_ll.add_first(professor_info)
        add_to_file('professor.csv', professor_info)
        print(f'{professor.name} successfully added! (First professor added)')

    if not exists:
        professors_ll.add_new(professor_info)
        add_to_file('professor.csv', professor_info)
        print(f'{professor.name} successfully added! (Another professor added)')

def delete_professor(self, email_address):
    '''delete a professor using their email address'''
    professors_ll, professor_list, header = get_data('professor.csv')
    try:
        print(f'Checking system to delete professor associated with the email {email_address}...')
        professors_ll.delete_node(email_address)
        write_to_file_ll('professor.csv', header, professors_ll)
    except Exception as e:
        print(f'Error deleting professor: {e}!')

def modify_professor_details(self, new_rank):
    '''modify a professor in the system using their email address'''
    professors_ll, professor_list, header = get_data('professor.csv')
    for professor in professor_list:
        if professor[0] == self.email_address:
            professor[2] = self.rank = new_rank
    write_to_file_array('professor.csv', header, professor_list)
    print(f'The rank for {self.name} has been updated to {self.rank}')

def show_course_details_by_professor(self):
    '''show a professor's course using their email address'''
    start_time = time.time()
    professor_list = get_data('professor.csv')[1]
    course_list = get_data('course.csv')[1]
    if self.email_address:
        for professor in professor_list:
```

```
The rank for Jack Sparrow has been updated to Tenured Lecturer
C:\Users\pakin\OneDrive\Desktop\DATA 200\Assignments\Data200-Lab1\checkmygrade_main.py
================================
      We
Admin
================================

1. Display all professors
2. Search course details by professor
3. Get grade report for professor
4. Add professor
5. Modify professor
6. Delete professor
7. Go back
Enter what you would like to do: 6
Deleting professor...
Enter the email address of the professor to be deleted: sparrow@myschool.edu
Checking system to delete professor associated with the email sparrow@myschool.edu...
Deleting sparrow@myschool.edu...

sparrow@myschool.edu successfully deleted!
C:\Users\pakin\OneDrive\Desktop\DATA 200\Assignments\Data200-Lab1\checkmygrade_main.py
================================
      Welcome to CheckMyGrade
Admin Portal (Professor Screen)
================================

1. Display all professors
2. Search course details by professor
3. Get grade report for professor
4. Add professor
5. Modify professor
6. Delete professor
7. Go back
Enter what you would like to do: 2
Enter email of the professor to get their course details: sparrow@myschool.edu
No professor found with the email!
C:\Users\pakin\OneDrive\Desktop\DATA 200\Assignments\Data200-Lab1\checkmygrade_main.py
================================
      W
Admi
================================

1. Display all professors
2. Search course details by professor
3. Get grade report for professor
4. Add professor
5. Modify professor
```

6. Delete professor (deletes professor from the database and professor.csv)

Checking if sparrow@myschool.edu was deleted

```
                update_professor = get_professor(email_address)
                if update_professor:
                    new_rank = input(f'Enter the new rank of {update_professor.name}: ')
                    update_professor.modify_professor_details(new_rank)
                else:
                    print('Professor not found! Make sure you entered their email correctly.')
            elif professor_choice == '6':
                print('Deleting professor...')
                email_address = input('Enter the email address of the professor to be deleted:
                delete_professor = Professor(email_address = None, name = None, rank = None, c
                delete_professor.delete_professor(email_address)
            elif professor_choice == '7':
                print('Returning to option screen...')
                professor_screen = False
            else:
                print('Invalid choice!')
        elif choice == '3':
            course_screen = True
            while course_screen:
                print(__file__)
                print('================================'.center(columns))
                msg='''Welcome to CheckMyGrade'''
```

```
C:\Users\pakin\OneDrive\Desktop\DATA 200\Assignments\Data200-Lab1\checkmygrade_main.py
================================
      Welcome to CheckMyGrade
Admin Portal (Professor Screen)
================================

1. Display all professors
2. Search course details by professor
3. Get grade report for professor
4. Add professor
5. Modify professor
6. Delete professor
7. Go back
Enter what you would like to do: 7
Returning to option screen...
C:\Users\pakin\OneDrive\Desktop\DATA 200\Assignments\Data200-Lab1\checkmygrade_main.py
================================
      Welcome to CheckMyGrade
            Admin Portal
================================

1. Show student options
2. Show professor options
3. Show course options
4. Logout
Enter what you would like to do:
```

7. Return back to the main screen

```
                    print('Returning to option screen...')
                    professor_screen = False
                else:
                    print('Invalid choice!')
            elif choice == '3':
                course_screen = True
                while course_screen:
                    print(__file__)
                    print('================================'.center(columns))
                    msg='''Welcome to CheckMyGrade'''
                    print(msg.center(columns))
                    msg2='''Admin Portal (Course Screen)'''
                    print(msg2.center(columns))
                    print('================================'.center(columns))
                    print('1. Display all courses')
                    print('2. Get grade report for course')
                    print('3. Add course')
                    print('4. Delete course')
                    print('5. Go back')
```

```
Returning to option screen...
C:\Users\pakin\OneDrive\Desktop\DATA 200\Assignments\Data200-Lab1\checkmygrade_main.py
                                ================================
                                    Welcome to CheckMyGrade
                                        Admin Portal
                                =

1. Show student options
2. Show professor options
3. Show course options
4. Logout
Enter what you would like to do: 3
C:\Users\pakin\OneDrive\Desktop\DATA 200\Assignments\Data200-Lab1\checkmygrade_main.py
                                ================================
                                    Welcome to CheckMyGrade
                                  Admin Portal (Course Screen)
                                ================================

1. Display all courses
2. Get grade report for course
3. Add course
4. Delete course
5. Go back
Enter what you would like to do:
```

Opting to view course screen

```
        sorted_list = sorted_list[::-1]
    for data in sorted_list:
        print(data)
    print(f'Sorting time: {time.time() - start_time} seconds')

class Course:
    '''course class that includes information regarding a course's id, credits, name'''
    def __init__(self, course_id, credits, course_name, course_description):
        self.course_id = course_id
        self.credits = credits
        self.course_name = course_name
        self.course_description = course_description

    @staticmethod # static to call out of course object
    def display_courses():
        '''displays all courses'''
        print('Displaying courses...')
        courses_list = get_data('course.csv')[1]
        for course in courses_list:
            print(f'''
                  {course[0]}
                  Name: {course[1]}
                  Credits: {course[2]}
                  Description: {course[3]}
                  ''')

    def add_new_course(self, course): # course from add_course()
        '''add a new course'''
        print(f'Checking system to add {course.course_id}...')
        courses_ll = get_data('course.csv')[0]
        course_info = [course.course_id, course.credits, course.course_name, course.course_description]
        exists = courses_ll.check_for_data_in_ll(self.course_id)

        if courses_ll.size() == 0:
            courses_ll.add_first(course_info)
            add_to_file('course.csv', course_info)
            print(f'{course.course_id} successfully added! (First course added)')

        if not exists:
            courses_ll.add_new(course_info)
            add_to_file('course.csv', course_info)
            print(f'{course.course_id} successfully added! (Another course added)')

    def delete_course(self, id):
        '''delete a course using its id'''
```

```
4. Logout
Enter what you would like to do: 3
C:\Users\pakin\OneDrive\Desktop\DATA 200\Assignments\Data200-Lab1\checkmygrade_main.py
                                ================================
                                    Welcome to CheckMyGrade
                                  Admin Portal (Course Screen)
                                ================================

1. Display all courses
2. Get grade report for course
3. Add course
4. Delete course
5. Go back
Enter what you would like to do: 1
Retrieving all courses...
Displaying courses...

                DATA200
                Name: Intro to Data Science
                Credits: 3
                Description: Utilizing Python in Data Science


                DATA201
                Name: Intro to Databases
                Credits: 3
                Description: Utilizing SQL in Data Science


                DATA202
                Name: Advanced Data Science
                Credits: 4
                Description: Involved data science algorithms and calculations


                DATA203
                Name: Advanced Database Techniques
                Credits: 3
                Description: Learn about advanced SQL queries and processing pipelines

C:\Users\pakin\OneDrive\Desktop\DATA 200\Assignments\Data200-Lab1\checkmygrade_main.py
                                ================================
                                    Welcome to CheckMyGrade
                                  Admin Portal (Course Screen)
                                ================================

1. Display all courses
2. Get grade report for course
3. Add course
4. Delete course
5. Go back
Enter what you would like to do:
```

1. Display all courses (shows all courses in the database)

```
        grade_id = Grades(grade[0], grade[1], grade[2], grade[3])
        print(f'{grade_id.grade_id}: {grade_id.grade} / {grade_id.mark}% / {grade[3]}')

    @staticmethod
    def display_grades_by_course(course):
        start_time = time.time()
        grades = Grades.get_grades()
        course_grades = []
        for grade in grades:
            if grade[3] == course:
                course_grades.append(grade[2])
                print(f'{grade[0]}: {grade[1]} ({grade[2]}%)')

        print(f'There are {len(course_grades)} students in {course}.\n')
        print(f'Search time: {time.time() - start_time} seconds')

    @staticmethod
    def get_course_average(course):
        course_grades = []
        grades = Grades.get_grades()
        for grade in grades:
            if grade[3] == course:
                course_grades.append(int(grade[2]))
        class_average = sum(course_grades) / len(course_grades)
        return class_average

    @staticmethod
    def get_course_median(course):
        course_grades = []
```

```
C:\Users\pakin\OneDrive\Desktop\DATA 200\Assignments\Data200-Lab1\checkmygrade_main.py
                                ================================


1. Display all courses
2. Get grade report for course
3. Add course
4. Delete course
5. Go back
Enter what you would like to do: 2
Enter the course id of the course you would like to get a grade report for: DATA200
mae@myschool.edu: B+ (88%)
russell@myschool.edu: A- (90%)
smart@myschool.edu: B+ (88%)
brown@myschool.edu: B (86%)
moore@myschool.edu: A (94%)
johnson@myschool.edu: C (75%)
There are 6 students in DATA200.

Search time: 0.0029206275939941406 seconds
C:\Users\pakin\OneDrive\Desktop\DATA 200\Assignments\Data200-Lab1\checkmygrade_main.py
                                ================================
                                    Welcome to CheckMyGrade
                                  Admin Portal (Course Screen)
                                ================================

1. Display all courses
2. Get grade report for course
3. Add course
4. Delete course
5. Go back
Enter what you would like to do:
```

2. Get grade report for course (gets grades of all students in the course)

```
        print(f'''
            {course[0]}
            Name: {course[1]}
            Credits: {course[2]}
            Description: {course[3]}
            ''')

def add_new_course(self, course): # course from add_course()
    '''add a new course'''
    print(f'Checking system to add {course.course_id}...')
    courses_ll = get_data('course.csv')[0]
    course_info = [course.course_id, course.credits, course.course_name, course.course_description]
    exists = courses_ll.check_for_data_in_ll(self.course_id)

    if courses_ll.size() == 0:
        courses_ll.add_first(course_info)
        add_to_file('course.csv', course_info)
        print(f'{course.course_id} successfully added! (First course added)')

    if not exists:
        courses_ll.add_new(course_info)
        add_to_file('course.csv', course_info)
        print(f'{course.course_id} successfully added! (Another course added)')

def delete_course(self, id):
    '''delete a course using its id'''
```

```
Search time: 0.0029206275939941406 seconds
C:\Users\pakin\OneDrive\Desktop\DATA 200\Assignments\Data200-Lab1\checkmygrade_main.py
==========
Welcome to CheckMyGrade
Admi
==========

1. Display all courses
2. Get grade report for course
3. Add course
4. Delete course
5. Go back
Enter what you would like to do: 3
Adding course...
Enter id of the course: DATA204
Enter the number of credits of the course: 3
Enter the name of the course: Data Techniques for Data Science
Enter the description of the course: SQL, Python advanced techniques
Checking system to add DATA204...
DATA204 successfully added! (Another course added)
C:\Users\pakin\OneDrive\Desktop\DATA 200\Assignments\Data200-Lab1\checkmygrade_main.py
==========
Welcome to CheckMyGrade
Admin Portal (Course Screen)
==========

1. Display all courses
2. Get grade report for course
3. Add course
4. Delete course
5. Go back
```

4. Add course (add a new course into the system – linked list)

Adding DATA204

checkmygrade_main.py 1 | course.csv | professor.csv | student.csv | login.csv

C: > Users > pakin > OneDrive > Desktop > DATA 200 > Assignments > Data200-Lab1 > course.csv

```
1  course_id,course_name,credits,description
2  DATA200,Intro to Data Science,3,Utilizing Python in Data Science
3  DATA201,Intro to Databases,3,Utilizing SQL in Data Science
4  DATA202,Advanced Data Science,4,Involved data science algorithms and calculations
5  DATA203,Advanced Database Techniques,3,Learn about advanced SQL queries and processing pipelines
6  DATA204,Data Techniques for Data Science,3,"SQL, Python advanced techniques"
7
```

Checking course.csv for the newly added course

```
            add_to_file('course.csv', course_info)
            print(f'{course.course_id} successfully added! (First course added)')

        if not exists:
            courses_ll.add_new(course_info)
            add_to_file('course.csv', course_info)
            print(f'{course.course_id} successfully added! (Another course added)')

    def delete_course(self, id):
        '''delete a course using its id'''
        courses_ll, course_list, header = get_data('course.csv')
        try:
            print(f'Checking system to delete course associated with the course id {id}...')
            courses_ll.delete_node(id)
            write_to_file_ll('course.csv', header, courses_ll)
        except Exception as e:
            print(f'Error deleting course: {e}!')

class Professor:
    '''professor class that includes information regarding a professor's name, email, rank'''
    def __init__(self, email_address, name, rank, course_id):
        self.name = name
        self.email_address = email_address
        self.rank = rank
        self.course_id = course_id
```

```
C:\Users\pakin\OneDrive\Desktop\DATA 200\Assignments\Data200-Lab1\checkmygrade_main.py
==========
Welcome
Admin Por
==========

1. Display all courses
2. Get grade report for course
3. Add course
4. Delete course
5. Go back
Enter what you would like to do: 4
Deleting course...
Enter the course id to be deleted: DATA204
Checking system to delete course associated with the course id DATA204...
Deleting DATA204...

DATA204 successfully deleted!
C:\Users\pakin\OneDrive\Desktop\DATA 200\Assignments\Data200-Lab1\checkmygrade_main.py
==========
Welcome to CheckMyGrade
Admin Portal (Course Screen)
==========

1. Display all courses
2. Get grade report for course
3. Add course
4. Delete course
5. Go back
Enter what you would like to do:
```

4. Delete course (delete course from the database and course.csv)

Removing DATA204

```
                        new_course = Course(course_id, course_credits, course_name, course_description
                        if new_course:
                            new_course.add_new_course(new_course)
                        else:
                            print('Issue adding course! Make sure you are entering the course details
                    elif course_choice == '4':
                        print('Deleting course...')
                        course_id = input('Enter the course id to be deleted: ')
                        delete_course = Course(course_id = None, credits = None, course_name = None, c
                        delete_course.delete_course(course_id)
                    elif course_choice == '5':
                        print('Returning to option screen...')
                        course_screen = False
                    else:
                        print('Invalid choice!')
                elif choice == '4':
                    print('Logging out...')
                    current_user.logout()
                    break
                else:
                    print('Enter a valid choice!')
elif choice == '2':
    print('Signing up...')
    first_name = input('Enter your first name: ')
    last_name = input('Enter your last name: ')
    email_address = input('Enter you email address: ').strip()
    password = getpass.getpass('Enter your password: ').strip()
    print('''
        Are you a...
```

```
C:\Users\pakin\OneDrive\Desktop\DATA 200\Assignments\Data200-Lab1\checkmygrade_main.py
==========
Welcome to CheckMyGrade
Admin Portal (Course Screen)
==========

1. Display all courses
2. Get grade report for course
3. Add course
4. Delete course
5. Go back
Enter what you would like to do: 5
Returning to option screen...
C:\Users\pakin\OneDrive\Desktop\DATA 200\Assignments\Data200-Lab1\checkmygrade_main.py
==========
Welcome to CheckMyGrade
Admin Portal
==========

1. Show student options
2. Show professor options
3. Show course options
4. Logout
Enter what you would like to do: 4
Logging out...
C:\Users\pakin\OneDrive\Desktop\DATA 200\Assignments\Data200-Lab1\checkmygrade_main.py
==========
Welcome to CheckMyGrade
==========

 CheckMyGrade Menu:
1. Login
2. Signup
3. Exit
Enter your choice:
```

5. Returning to main screen

Logging out of admin

Scenario 4: New User

CheckMyGrade allows for signing up of new users. A new user will input their first name, last name, email address, and password. They will then answer whether they are a student or a professor, which will give them access to whichever functionalities of the app are relevant to their needs. Currently, new users are more like placeholders, and will have more viability in later updates of the app where courses can be added to students and professors.

Scenario 1: New Student

```
3. Show course options
4. Logout
Enter what you would like to do: 4
Logging out...
C:\Users\pakin\OneDrive\Desktop\DATA 200\Assignments\Data200-Lab1\checkmygrade_main.py
                                   ==============================
                                          Welcome to CheckMyGrade
                                   ==============================


 CheckMyGrade Menu:
1. Login
2. Signup
3. Exit
Enter your choice: 2
Signing up...
Enter your first name: Liana
Enter your last name: P
Enter you email address: pak@myschool.edu
Enter your password:

                Are you a...
                1. Student
                2. Professor

Enter number of your status: 1
You have successfully signed up! Please try logging in
C:\Users\pakin\OneDrive\Desktop\DATA 200\Assignments\Data200-Lab1\checkmygrade_main.py
                                   ==============================
                                          Welcome to CheckMyGrade
                                   ==============================


 CheckMyGrade Menu:
1. Login
2. Signup
3. Exit
Enter your choice: 1
Logging in...
Enter your email address: pak@myschool.edu
Enter your password:
You are now logged in!
True pak@myschool.edu student
Search time: 0.0005021095275878906 seconds
C:\Users\pakin\OneDrive\Desktop\DATA 200\Assignments\Data200-Lab1\checkmygrade_main.py
                                   ==============================
                                          Welcome to CheckMyGrade
                                             Student Portal
                                   ==============================


 CheckMyGrade Main Menu:
1. Check my grade
2. Check my mark
3. Compare my mark against course average
4. Change my password
5. Logout
Enter what you would like to do:
```

The new student enters their information and selects '1', which denotes that they are a student

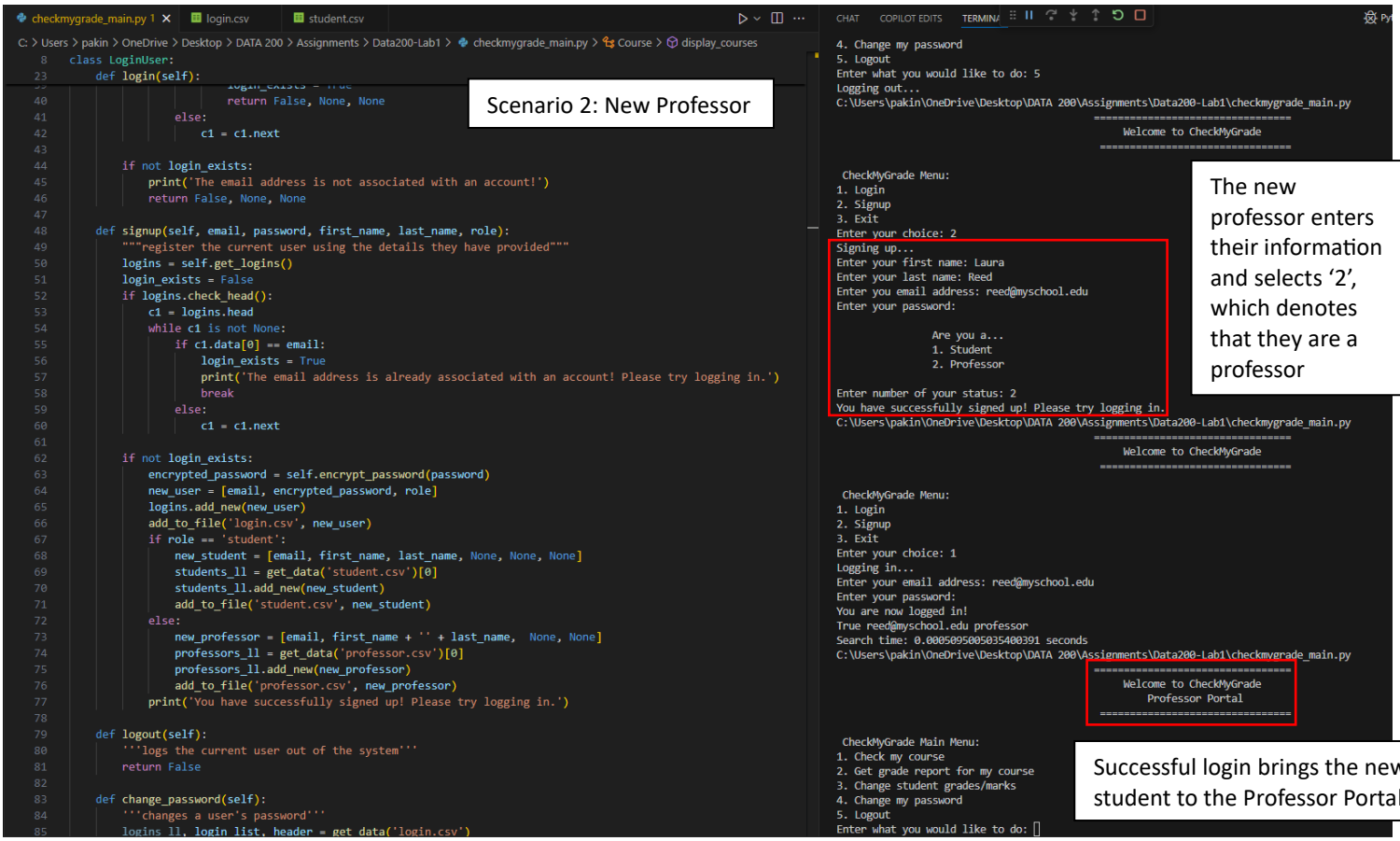
Successful login brings the new student to the Student Portal

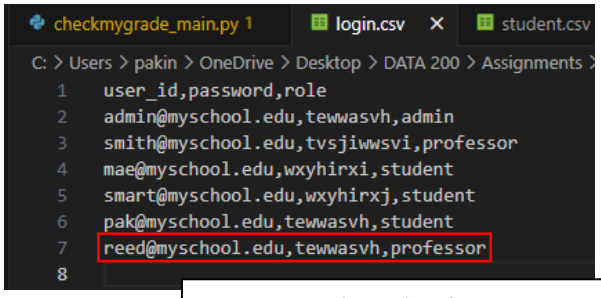
```
user_id,password,role
admin@myschool.edu,tewwasvh,admin
smith@myschool.edu,tvsjiwwsvi,professor
mae@myschool.edu,wxyhirxi,student
smart@myschool.edu,wxyhirxj,student
pak@myschool.edu,tewwasvh,student
```

The new student's information is added to both the login.csv and student.csv

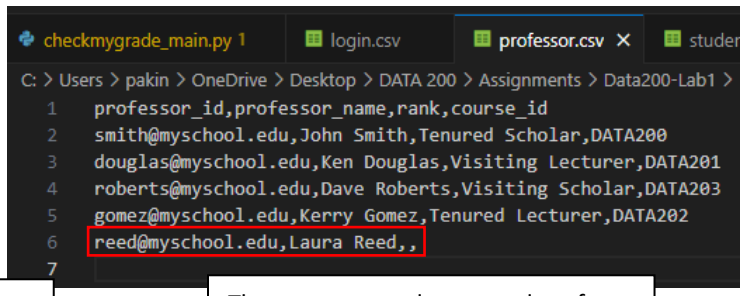
```
student_id,first_name,last_name,course_ids,grades,marks
mae@myschool.edu,Samantha,Mae,DATA200|DATA201,B+|A-,88|92
russell@myschool.edu,Sally,Russell,DATA200,A-,90
doe@myschool.edu,John,Doe,DATA201|DATA203,A|B+,94|89
smart@myschool.edu,Jane,Smart,DATA200|DATA201|DATA202,B+|A-|A,88|90|96
jones@myschool.edu,Emily,Jones,DATA203,A-,92
brown@myschool.edu,Michael,Brown,DATA200|DATA203,B|C+,86|78
davis@myschool.edu,Sarah,Davis,DATA202|DATA203,A|B+,97|88
miller@myschool.edu,William,Miller,DATA201|DATA202,C-|D+,72|68
wilson@myschool.edu,Lucy,Wilson,DATA202,B-,80
moore@myschool.edu,Charles,Moore,DATA200,A,94
taylor@myschool.edu,Olivia,Taylor,DATA203,B+,88
anderson@myschool.edu,Benjamin,Anderson,DATA201,C,75
johnson@myschool.edu,Jack,Johnson,DATA200,C,75
pak@myschool.edu,Liana,P,,,
```

There are currently none values for the new student's courses, grades, and marks



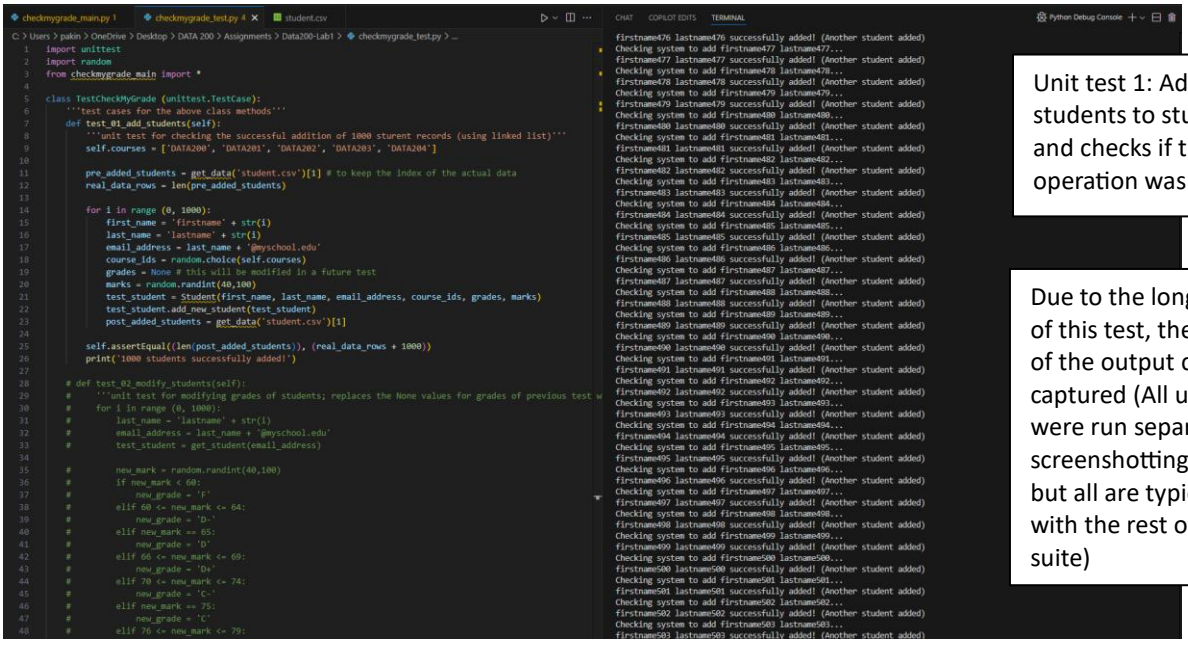


The new professor's information is added to both the login.csv and professor.csv



There are currently none values for the new professor's rank and course

Unit Testing



Unit test 1: Adds 1000 students to student.csv and checks if the operation was successful

Due to the long output of this test, the entirety of the output cannot be captured (All unit tests were run separately for screenshotting purposes, but all are typically run with the rest of the test suite)

```python
class TestCheckMyGrade (unittest.TestCase):
    '''test cases for the above class methods'''
    def test_01_add_students(self):
        '''unit test for checking the successful addition of 1000 sturent records (using linked list)'''
        self.courses = ['DATA200', 'DATA201', 'DATA202', 'DATA203', 'DATA204']

        pre_added_students = get_data('student.csv')[1] # to keep the index of the actual data
        real_data_rows = len(pre_added_students)

        for i in range (0, 1000):
            first_name = 'firstname' + str(i)
            last_name = 'lastname' + str(i)
            email_address = last_name + '@myschool.edu'
            course_ids = random.choice(self.courses)
            grades = None # this will be modified in a future test
            marks = random.randint(40,100)
            test_student = Student(first_name, last_name, email_address, course_ids, grades, marks)
            test_student.add_new_student(test_student)
            post_added_students = get_data('student.csv')[1]

        self.assertEqual((len(post_added_students)), (real_data_rows + 1000))
        print('1000 students successfully added!')

    # def test_02_modify_students(self):
    #     '''unit test for modifying grades of students; replaces the None values for grades of previous test w
    #     for i in range (0, 1000):
    #         last_name = 'lastname' + str(i)
    #         email_address = last_name + '@myschool.edu'
    #         test_student = get_student(email_address)

    #         new_mark = random.randint(40,100)
    #         if new_mark < 60:
    #             new_grade = 'F'
    #         elif 60 <= new_mark <= 64:
    #             new_grade = 'D-'
    #         elif new_mark == 65:
    #             new_grade = 'D'
    #         elif 66 <= new_mark <= 69:
    #             new_grade = 'D+'
    #         elif 70 <= new_mark <= 74:
    #             new_grade = 'C-'
    #         elif new_mark == 75:
    #             new_grade = 'C'
    #         elif 76 <= new_mark <= 79:
```

```
Checking system to add firstname979 lastname979...
firstname979 lastname979 successfully added! (Another student added)
Checking system to add firstname980 lastname980...
firstname980 lastname980 successfully added! (Another student added)
Checking system to add firstname981 lastname981...
firstname981 lastname981 successfully added! (Another student added)
Checking system to add firstname982 lastname982...
lly added! (Another student added)
3 lastname983...
lly added! (Another student added)
4 lastname984...
lly added! (Another student added)
5 lastname985...
lly added! (Another student added)
6 lastname986...
lly added! (Another student added)
7 lastname987...
lly added! (Another student added)
8 lastname988...
lly added! (Another student added)
9 lastname989...
firstname989 lastname989 successfully added! (Another student added)
Checking system to add firstname990 lastname990...
firstname990 lastname990 successfully added! (Another student added)
Checking system to add firstname991 lastname991...
firstname991 lastname991 successfully added! (Another student added)
Checking system to add firstname992 lastname992...
firstname992 lastname992 successfully added! (Another student added)
Checking system to add firstname993 lastname993...
firstname993 lastname993 successfully added! (Another student added)
Checking system to add firstname994 lastname994...
firstname994 lastname994 successfully added! (Another student added)
Checking system to add firstname995 lastname995...
firstname995 lastname995 successfully added! (Another student added)
Checking system to add firstname996 lastname996...
firstname996 lastname996 successfully added! (Another student added)
Checking system to add firstname997 lastname997...
firstname997 lastname997 successfully added! (Another student added)
Checking system to add firstname998 lastname998...
firstname998 lastname998 successfully added! (Another student added)
Checking system to add firstname999 lastname999...
firstname999 lastname999 successfully added! (Another student added)
1000 students successfully added!
ok

----------------------------------------------------------------------
Ran 1 test in 12.277s

OK
PS C:\Users\pakin\OneDrive\Desktop\DATA 200\Assignments\Data200-Lab1>
```

The test succeeds if the length after the insertion of 1000 students is equal to the length of the original file (with the actual students) + 1000

checkmygrade_main.py 1 | checkmygrade_test.py 2 | student.csv

C: > Users > pakin > OneDrive > Desktop > DATA 200 > Assignments > Data200-Lab1 > checkmygrade_test.py > ...

```python
class TestCheckMyGrade (unittest.TestCase):
    #     print('1000 students successfully added!')

    def test_02_modify_students(self):
        '''unit test for modifying grades of students; replaces the None values for grades of previous test wit
        for i in range (0, 1000):
            last_name = 'lastname' + str(i)
            email_address = last_name + '@myschool.edu'
            test_student = get_student(email_address)

            new_mark = random.randint(40,100)
            if new_mark < 60:
                new_grade = 'F'
            elif 60 <= new_mark <= 64:
                new_grade = 'D-'
            elif new_mark == 65:
                new_grade = 'D'
            elif 66 <= new_mark <= 69:
                new_grade = 'D+'
            elif 70 <= new_mark <= 74:
                new_grade = 'C-'
            elif new_mark == 75:
                new_grade = 'C'
            elif 76 <= new_mark <= 79:
                new_grade = 'C+'
            elif 80 <= new_mark <= 84:
                new_grade = 'B-'
            elif new_mark == 85:
                new_grade = 'B'
            elif 86 <= new_mark <= 89:
                new_grade = 'B+'
            elif 90 <= new_mark <= 94:
                new_grade = 'A-'
            elif new_mark == 95:
                new_grade = 'A'
            elif 96 <= new_mark <= 100:
                new_grade = 'A+'
            test_student.update_student_record(new_grade, new_mark)

        self.assertIsNotNone(test_student.grades) # checks to see if the student grades are no longer None
        print('Student grades were modified!')

    # def test_03_delete_students(self):
    #     '''unit test for deleting the 1000 sturent records (using linked list)'''
    #     pre_deleted_students = get_data('student.csv')[1]
    #     pre_deleted_students_length = len(pre_deleted_students)

    #     for i in range (0, 1000):
```

CHAT | COPILOT EDITS | TERMINAL | Python Debug Console

```
PS C:\Users\pakin\OneDrive\Desktop\DATA 200\Assignments\Data200-Lab1> ^C
PS C:\Users\pakin\OneDrive\Desktop\DATA 200\Assignments\Data200-Lab1>
PS C:\Users\pakin\OneDrive\Desktop\DATA 200\Assignments\Data200-Lab1>  c:; cd 'c:\Users\pakin\OneDrive\Desktop\DATA 200\Assi
gnments\Data200-Lab1'; & 'c:\Users\pakin\AppData\Local\Programs\Python\Python313\python.exe' 'c:\Users\pakin\.vscode\extensi
ons\ms-python.debugpy-2025.4.1-win32-x64\bundled\libs\debugpy\launcher' '63848' '--' 'C:\Users\pakin\OneDrive\Desktop\DATA 2
00\Assignments\Data200-Lab1\checkmygrade_test.py'
test_02_modify_students (__main__.TestCheckMyGrade.test_02_modify_students)
unit test for modifying grades of students; replaces the None values for grades of previous test with actual grades ... Sear
ch time: 0.013820409774780273 seconds
The grade and mark for firstname0 lastname0 has been updated to C+ (78%)
Search time: 0.011208772659301758 seconds
The grade and mark for firstname1 lastname1 has been updated to C- (72%)
Search time: 0.013284051599121094 seconds
The grade and mark for firstname2 lastname2 has been updated to A+ (98%)
Search time: 0.011330366134643555 seconds
The grade and mark for firstname3 lastname3 has been updated to F (55%)
Search time: 0.012778043746948242 seconds
The grade and mark for firstname4 lastname4 has been updated to F (43%)
Search time: 0.014619588851928711 seconds
The grade and mark for firstname5 lastname5 has been updated to F (49%)
Search time: 0.01596975326538086 seconds
The grade and mark for firstname6 lastname6 has been updated to F (51%)
Search time: 0.015400409698486328 seconds
The grade and mark for firstname7 lastname7 has been updated to B- (82%)
Search time: 0.015519380569458008 seconds
The grade and mark for firstname8 lastname8 has been updated to C+ (76%)
Search time: 0.012742757797241211 seconds
The grade and mark for firstname9 lastname9 has been updated to D+ (66%)
Search time: 0.013720989227294922 seconds
The grade and mark for firstname10 lastname10 has been updated to A+ (99%)
Search time: 0.015157699584960938 seconds
The grade and mark for firstname11 lastname11 has been updated to B- (82%)
Search time: 0.012419223785400391 seconds
The grade and mark for firstname12 lastname12 has been updated to A+ (97%)
Search time: 0.013502836227416992 seconds
The grade and mark for firstname13 lastname13 has been updated to A+ (96%)
Search time: 0.015069007873535156 seconds
The grade and mark for firstname14 lastname14 has been updated to C+ (79%)
Search time: 0.013628244400024414 seconds
The grade and mark for firstname15 lastname15 has been updated to B- (84%)
Search time: 0.014990091323852539 seconds
The grade and mark for firstname16 lastname16 has been updated to F (40%)
Search time: 0.015700340270996094 seconds
The grade and mark for firstname17 lastname17 has been updated to F (53%)
Search time: 0.012405633926391602 seconds
The grade and mark for firstname18 lastname18 has been updated to D+ (68%)
Search time: 0.011895179748535156 seconds
The grade and mark for firstname19 lastname19 has been updated to B+ (88%)
Search time: 0.011680603027343750 seconds
The grade and mark for firstname20 lastname20 has been updated to C- (73%)
Search time: 0.011520862579345703 seconds
The grade and mark for firstname21 lastname21 has been updated to D- (61%)
Search time: 0.012590408325195312 seconds
The grade and mark for firstname22 lastname22 has been updated to D- (60%)
Search time: 0.012044191360473633 seconds
```

Unit test 2: Modifies 1000 student records to give each student a letter grade based on their mark

Each modification returns the output of the student's updated grade along with the time it took to retrieve the student's information

checkmygrade_main.py 1 | checkmygrade_test.py 2 | student.csv

C: > Users > pakin > OneDrive > Desktop > DATA 200 > Assignments > Data200-Lab1 > checkmygrade_test.py > ...

```
class TestCheckMyGrade (unittest.TestCase):
    #         marks = random.randint(40,100)
    #         test_student = Student(first_name, last_name, email_address, course_ids, grades, marks)
    #         test_student.add_new_student(test_student)
    #         post_added_students = get_data('student.csv')[1]

    #     self.assertEqual((len(post_added_students)), (real_data_rows + 1000))
    #     print('1000 students successfully added!')

    def test_02_modify_students(self):
        '''unit test for modifying grades of students; replaces the None values for grades of previous test wit
        for i in range (0, 1000):
            last_name = 'lastname' + str(i)
            email_address = last_name + '@myschool.edu'
            test_student = get_student(email_address)

            new_mark = random.randint(40,100)
            if new_mark < 60:
                new_grade = 'F'
            elif 60 <= new_mark <= 64:
                new_grade = 'D-'
            elif new_mark == 65:
                new_grade = 'D'
            elif 66 <= new_mark <= 69:
                new_grade = 'D+'
            elif 70 <= new_mark <= 74:
                new_grade = 'C-'
            elif new_mark == 75:
                new_grade = 'C'
            elif 76 <= new_mark <= 79:
                new_grade = 'C+'
            elif 80 <= new_mark <= 84:
                new_grade = 'B-'
            elif new_mark == 85:
                new_grade = 'B'
            elif 86 <= new_mark <= 89:
                new_grade = 'B+'
            elif 90 <= new_mark <= 94:
                new_grade = 'A-'
            elif new_mark == 95:
                new_grade = 'A'
            elif 96 <= new_mark <= 100:
                new_grade = 'A+'
            test_student.update_student_record(new_grade, new_mark)

        self.assertIsNotNone(test_student.grades) # checks to see if the student grades are no longer None
        print('Student grades were modified!')

    # def test_03_delete_students(self):
```

The test succeeds if the grade of the student is no longer None, which it was when it was first inserted into the database

CHAT | COPILOT EDITS | TERMINAL

```
The grade and mark for firstname976 lastname976 has been updated to F (54%)
Search time: 0.012262821197509766 seconds
The grade and mark for firstname977 lastname977 has been updated to C- (70%)
Search time: 0.015490055084228516 seconds
The grade and mark for firstname978 lastname978 has been updated to F (48%)
Search time: 0.014477252960205078 seconds
The grade and mark for firstname979 lastname979 has been updated to F (54%)
Search time: 0.012951135635375977 seconds
The grade and mark for firstname980 lastname980 has been updated to F (50%)
Search time: 0.012653827667236328 seconds
The grade and mark for firstname981 lastname981 has been updated to A- (91%)
Search time: 0.012491226196289062 seconds
The grade and mark for firstname982 lastname982 has been updated to B (85%)
Search time: 0.01183629035949707 seconds
The grade and mark for firstname983 lastname983 has been updated to F (43%)
Search time: 0.01654672622680664 seconds
The grade and mark for firstname984 lastname984 has been updated to F (59%)
Search time: 0.014436960220336914 seconds
The grade and mark for firstname985 lastname985 has been updated to B (85%)
Search time: 0.013510465621948242 seconds
The grade and mark for firstname986 lastname986 has been updated to D (65%)
Search time: 0.051900386810302734 seconds
The grade and mark for firstname987 lastname987 has been updated to C+ (79%)
Search time: 0.012314081192016602 seconds
The grade and mark for firstname988 lastname988 has been updated to D- (63%)
Search time: 0.01708984375 seconds
The grade and mark for firstname989 lastname989 has been updated to B- (83%)
Search time: 0.015577077865600586 seconds
The grade and mark for firstname990 lastname990 has been updated to F (50%)
Search time: 0.013770580291748047 seconds
The grade and mark for firstname991 lastname991 has been updated to F (53%)
Search time: 0.011567354202270508 seconds
The grade and mark for firstname992 lastname992 has been updated to F (49%)
Search time: 0.012549400329589844 seconds
The grade and mark for firstname993 lastname993 has been updated to F (46%)
Search time: 0.01606893539428711 seconds
The grade and mark for firstname994 lastname994 has been updated to D- (64%)
Search time: 0.014562368392944336 seconds
The grade and mark for firstname995 lastname995 has been updated to C+ (79%)
Search time: 0.010529518127441406 seconds
The grade and mark for firstname996 lastname996 has been updated to C- (70%)
Search time: 0.014367580413818336 seconds
The grade and mark for firstname997 lastname997 has been updated to F (43%)
Search time: 0.011324644088745117 seconds
The grade and mark for firstname998 lastname998 has been updated to A+ (97%)
Search time: 0.013955116271972656 seconds
The grade and mark for firstname999 lastname999 has been updated to B- (83%)
Student grades were modified!
ok

----------------------------------------------------------------------
Ran 1 test in 32.761s

OK
PS C:\Users\pakin\OneDrive\Desktop\DATA 200\Assignments\Data200-Lab1>
```

checkmygrade_main.py 1 | checkmygrade_test.py 4 | student.csv

C: > Users > pakin > OneDrive > Desktop > DATA 200 > Assignments > Data200-Lab1 > checkmygrade_test.py > ...

```
class TestCheckMyGrade (unittest.TestCase):
    #             new_grade = 'B'
    #         elif 86 <= new_mark <= 89:
    #             new_grade = 'B+'
    #         elif 90 <= new_mark <= 94:
    #             new_grade = 'A-'
    #         elif new_mark == 95:
    #             new_grade = 'A'
    #         elif 96 <= new_mark <= 100:
    #             new_grade = 'A+'
    #         test_student.update_student_record(new_grade, new_mark)

    #     self.assertIsNotNone(test_student.grades) # checks to see if the student grades are no longer None
    #     print('Student grades were modified!')

    def test_03_delete_students(self):
        '''unit test for deleting the 1000 sturent records (using linked list)'''
        pre_deleted_students = get_data('student.csv')[1]
        pre_deleted_students_length = len(pre_deleted_students)

        for i in range (0, 1000):
            last_name = 'lastname' + str(i)
            email_address = last_name + '@myschool.edu'
            test_student = Student(first_name = None, last_name = None, email_address = None, course_ids = None
            test_student.delete_student(email_address)
            post_deleted_students = get_data('student.csv')[1]

        self.assertEqual((len(post_deleted_students)), (pre_deleted_students_length - 1000), 'After 1000 studen
        print('1000 students were deleted!')

    # def test_04_load_file(self):
    #     '''unit test for checking to see if students from
    #     students_from_file = get_data('student.csv')[0]
    #     self.assertGreater(students_from_file.size(), 0)
    #     print('Student loaded from file successfully')

    # def test_05_search_from_loaded_file(self):
    #     '''unit test for timing searches of certain studen
    #     print('Searching for students...')
    #     starting_time_1 = time.time()
    #     searched_student_1 = get_student('mae@myschool.edu
    #     print(f'Amount of time it took to find {searched_s

    #     starting_time_2 = time.time()
    #     searched_student_2 = get_student('smart@myschool.e
    #     print(f'Amount of time it took to find {searched_student_2.first_name} {searched_student_2.last_name}

    # def test_06_sort_students(self):
```

Unit test 3: Deletes 1000 student records from student.csv

Students that were added in the previous test are deleted, with the system checking and matching the email to the stored student

CHAT | COPILOT EDITS | TERMINAL

```
Deleting lastname204@myschool.edu...

lastname204@myschool.edu successfully deleted!
Checking system to delete student associated with email lastname205@myschool.edu...
Deleting lastname205@myschool.edu...

lastname205@myschool.edu successfully deleted!
Checking system to delete student associated with email lastname206@myschool.edu...
Deleting lastname206@myschool.edu...

lastname206@myschool.edu successfully deleted!
Checking system to delete student associated with email lastname207@myschool.edu...
Deleting lastname207@myschool.edu...

lastname207@myschool.edu successfully deleted!
Checking system to delete student associated with email lastname208@myschool.edu...
Deleting lastname208@myschool.edu...

lastname208@myschool.edu successfully deleted!
Checking system to delete student associated with email lastname209@myschool.edu...
Deleting lastname209@myschool.edu...

lastname209@myschool.edu successfully deleted!
Checking system to delete student associated with email lastname210@myschool.edu...
Deleting lastname210@myschool.edu...

lastname210@myschool.edu successfully deleted!
Checking system to delete student associated with email lastname211@myschool.edu...
Deleting lastname211@myschool.edu...

lastname211@myschool.edu successfully deleted!
Checking system to delete student associated with email lastname212@myschool.edu...
Deleting lastname212@myschool.edu...

lastname212@myschool.edu successfully deleted!
Checking system to delete student associated with email lastname213@myschool.edu...
Deleting lastname213@myschool.edu...

lastname213@myschool.edu successfully deleted!
Checking system to delete student associated with email lastname214@myschool.edu...
Deleting lastname214@myschool.edu...

lastname214@myschool.edu successfully deleted!
Checking system to delete student associated with email lastname215@myschool.edu...
Deleting lastname215@myschool.edu...

lastname215@myschool.edu successfully deleted!
Checking system to delete student associated with email lastname216@myschool.edu...
Deleting lastname216@myschool.edu...

lastname216@myschool.edu successfully deleted!
Checking system to delete student associated with email lastname217@myschool.edu...
Deleting lastname217@myschool.edu...

lastname217@myschool.edu successfully deleted!
```

```python
#     print('Student grades were modified!')

def test_03_delete_students(self):
    '''unit test for deleting the 1000 sturent records (using
    pre_deleted_students = get_data('student.csv')[1]
    pre_deleted_students_length = len(pre_deleted_students)

    for i in range (0, 1000):
        last_name = 'lastname' + str(i)
        email_address = last_name + '@myschool.edu'
        test_student = Student(first_name = None, last_name =
        test_student.delete_student(email_address)
        post_deleted_students = get_data('student.csv')[1]

    self.assertEqual((len(post_deleted_students)), (pre_deleted_students_length - 1000), 'After 1000 students
    print('1000 students were deleted!')

# def test_04_load_file(self):
#     '''unit test for checking to see if students from student.csv file were successfully loaded'''
#     students_from_file = get_data('student.csv')[0]
#     self.assertGreater(students_from_file.size(), 0)
#     print('Student loaded from file successfully')

# def test_05_search_from_loaded_file(self):
#     '''unit test for timing searches of certain students'''
#     print('Searching for students...')
#     starting_time_1 = time.time()
```

The test succeeds if the number of students in student.csv post-deletion is 1000 less than the number of student pre-deletion

```
lastname994@myschool.edu successfully deleted!
Checking system to delete student associated with email lastname995@myschool.edu...
Deleting lastname995@myschool.edu...

lastname995@myschool.edu successfully deleted!
Checking system to delete student associated with email lastname996@myschool.edu...
Deleting lastname996@myschool.edu...

lastname996@myschool.edu successfully deleted!
Checking system to delete student associated with email lastname997@myschool.edu...
Deleting lastname997@myschool.edu...

lastname997@myschool.edu successfully deleted!
Checking system to delete student associated with email lastname998@myschool.edu...
Deleting lastname998@myschool.edu...

lastname998@myschool.edu successfully deleted!
Checking system to delete student associated with email lastname999@myschool.edu...
Deleting lastname999@myschool.edu...

lastname999@myschool.edu successfully deleted!
1000 students were deleted!
ok

----------------------------------------------------------------------
Ran 1 test in 13.239s

OK
PS C:\Users\pakin\OneDrive\Desktop\DATA 200\Assignments\Data200-Lab1>
```

```python
#     self.assertEqual((len(post_deleted_students)), (pre_deleted_students_length - 1000), 'After 1000 stud
#     print('1000 students were deleted!')

def test_04_load_file(self):
    '''unit test for checking to see if students from student.csv file were successfully loaded'''
    students_from_file = get_data('student.csv')[0]
    self.assertGreater(students_from_file.size(), 0)
    print('Student loaded from file successfully')

# def test_05_search_from_loaded_file(self):
#     '''unit test for timing searches of certain students'''
```

```
gnments\Data200-Lab1'; & 'C:\Users\pakin\AppData\Local\Programs\Python\Python313\python.exe' 'c:\Users\pakin\.vscode\extensi
ons\ms-python.debugpy-2025.4.1-win32-x64\bundled\libs\debugpy\launcher' '63946' '--' 'C:\Users\pakin\OneDrive\Desktop\DATA 2
00\Assignments\Data200-Lab1\checkmygrade_test.py'
test_04_load_file (__main__.TestCheckMyGrade.test_04_load_file)
unit test for checking to see if students from student.csv file were successfully loaded ... Student loaded from file succes
sfully
ok

----------------------------------------------------------------------
Ran 1 test in 0.002s

OK
PS C:\Users\pakin\OneDrive\Desktop\DATA 200\Assignments\Data200-Lab1>
```

Unit test 4: Checking if data loads from student.csv

```python
#     self.assertEqual((len(post_deleted_students)), (pre_deleted_students_length - 1000), 'After 1000 student
#     print('1000 students were deleted!')

def test_04_load_file(self):
    '''unit test for checking to see if students from student.csv file were successfully loaded'''
    students_from_file = get_data('student.csv')[0]
    self.assertGreater(students_from_file.size(), 0)
    print('Student loaded from file successfully')

def test_05_search_from_loaded_file(self):
    '''unit test for timing searches of certain students'''
    print('Searching for students...')
    starting_time_1 = time.time()
    searched_student_1 = get_student('mae@myschool.edu') # get_student inherently calls loads the file (uses
    print(f'Amount of time it took to find {searched_student_1.first_name} {searched_student_1.last_name}: {(t

    starting_time_2 = time.time()
    searched_student_2 = get_student('smart@myschool.edu')
    print(f'Amount of time it took to find {searched_student_2.first_name} {searched_student_2.last_name}: {(t

# def test_06_sort_students(self):
```

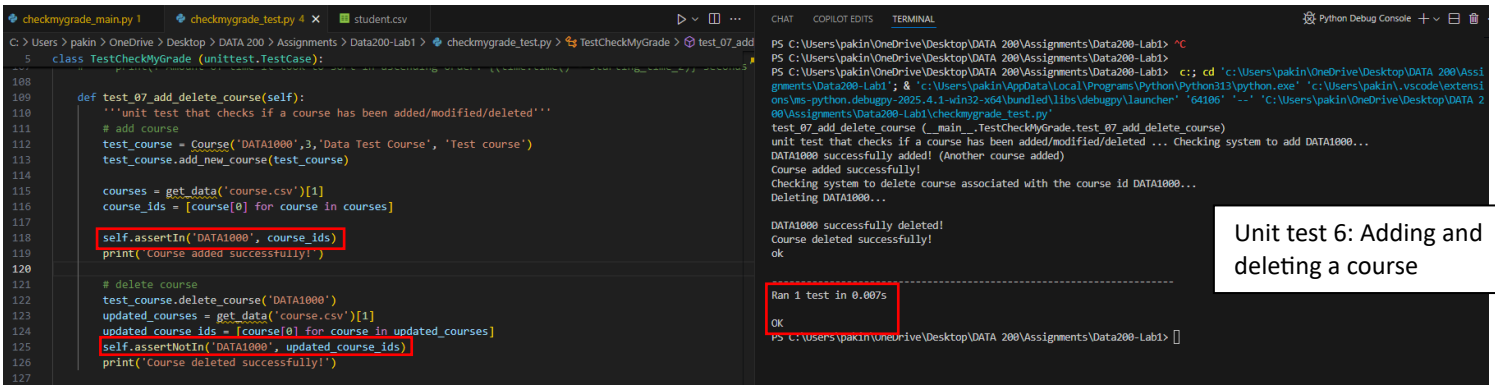
```
ok
PS C:\Users\pakin\OneDrive\Desktop\DATA 200\Assignments\Data200-Lab1> ^C
PS C:\Users\pakin\OneDrive\Desktop\DATA 200\Assignments\Data200-Lab1>
PS C:\Users\pakin\OneDrive\Desktop\DATA 200\Assignments\Data200-Lab1>  c:; cd 'c:\Users\pakin\OneDrive\Desktop\DATA 200\Assi
gnments\Data200-Lab1'; & 'c:\Users\pakin\AppData\Local\Programs\Python\Python313\python.exe' 'c:\Users\pakin\.vscode\extensi
ons\ms-python.debugpy-2025.4.1-win32-x64\bundled\libs\debugpy\launcher' '63990' '--' 'C:\Users\pakin\OneDrive\Desktop\DATA 2
00\Assignments\Data200-Lab1\checkmygrade_test.py'
test_04_load_file (__main__.TestCheckMyGrade.test_04_load_file)
unit test for checking to see if students from student.csv file were successfully loaded ... Student loaded from file succes
sfully
ok
test_05_search_from_loaded_file (__main__.TestCheckMyGrade.test_05_search_from_loaded_file)
unit test for timing searches of certain students ... Searching for students...
Search time: 0.0005695819854736328 seconds
Amount of time it took to find Samantha Mae: 0.0008208751678466797 seconds
Search time: 0.0003678798675537109 seconds
Amount of time it took to find Jane Smart: 0.0007505416870117188 seconds
ok

----------------------------------------------------------------------
Ran 2 tests in 0.006s

OK
PS C:\Users\pakin\OneDrive\Desktop\DATA 200\Assignments\Data200-Lab1>
```

Unit test 5: Checking how long it takes to search for students

```python
# def test_05_search_from_loaded_file(self):
#     '''unit test for timing searches of certain students'''
#     print('Searching for students...')
#     starting_time_1 = time.time()
#     searched_student_1 = get_student('mae@myschool.edu') # get_student inherently calls loads the file (uses
#     print(f'Amount of time it took to find {searched_student_1.first_name} {searched_student_1.last_name}: {(

#     starting_time_2 = time.time()
#     searched_student_2 = get_student('smart@myschool.edu')
#     print(f'Amount of time it took to find {searched_student_2.first_name} {searched_student_2.last_name}: {(

def test_06_sort_students(self):
    '''unit test for sorting students by email (asc and desc order)'''
    starting_time_1 = time.time()
    Student.sort_students('email', 'default') # ascending/default
    print(f'Amount of time it took to sort in ascending order: {(time.time() - starting_time_1)} seconds')

    starting_time_2 = time.time()
    Student.sort_students('email', 'reverse') # descending
    print(f'Amount of time it took to sort in descending order: {(time.time() - starting_time_2)} seconds')

# def test_07_add_delete_course(self):
#     '''unit test that checks if a course has been added/modified/deleted'''
#     # add course
#     test_course = Course('DATA1000',3,'Data Test Course', 'Test course')
#     test_course.add_new_course(test_course)

#     courses = get_data('course.csv')[1]
#     course_ids = [course[0] for course in courses]

#     self.assertIn('DATA1000', course_ids)
#     print('Course added successfully!')

#     # delete course
#     test_course.delete_course('DATA1000')
#     updated_courses = get_data('course.csv')[1]
#     updated_course_ids = [course[0] for course in updated_courses]
#     self.assertNotIn('DATA1000', updated_course_ids)
```

```
gnments\Data200-Lab1'; & 'c:\Users\pakin\AppData\Local\Programs\Python\Python313\python.exe' 'c:\Users\pakin\.vscode\extensi
ons\ms-python.debugpy-2025.4.1-win32-x64\bundled\libs\debugpy\launcher' '64072' '--' 'C:\Users\pakin\OneDrive\Desktop\DATA 2
00\Assignments\Data200-Lab1\checkmygrade_test.py'
test_06_sort_students (__main__.TestCheckMyGrade.test_06_sort_students)
unit test for sorting students by email (asc and desc order) ... ['anderson@myschool.edu', 'Benjamin', 'Anderson', 'DATA201'
, 'C', '75']
['brown@myschool.edu', 'Michael', 'Brown', 'DATA200|DATA203', 'B|C+', '86|78']
['davis@myschool.edu', 'Sarah', 'Davis', 'DATA202|DATA203', 'A|B+', '97|88']
['doe@myschool.edu', 'John', 'Doe', 'DATA201|DATA203', 'A|B+', '94|89']
['johnson@myschool.edu', 'Jack', 'Johnson', 'DATA200', 'C', '75']
['jones@myschool.edu', 'Emily', 'Jones', 'DATA203', 'A-', '92']
['mae@myschool.edu', 'Samantha', 'Mae', 'DATA200|DATA201', 'B+|A-', '88|92']
['miller@myschool.edu', 'William', 'Miller', 'DATA201|DATA202', 'C-|D+', '72|68']
['moore@myschool.edu', 'Charles', 'Moore', 'DATA200', 'A', '94']
['russell@myschool.edu', 'Sally', 'Russell', 'DATA200', 'A-', '90']
['smart@myschool.edu', 'Jane', 'Smart', 'DATA200|DATA201|DATA202', 'B+|A-|A', '88|90|96']
['taylor@myschool.edu', 'Olivia', 'Taylor', 'DATA203', 'B+', '88']
['wilson@myschool.edu', 'Lucy', 'Wilson', 'DATA202', 'B-', '80']
Sorting time: 0.0022428035736083984 seconds
Amount of time it took to sort in ascending order: 0.0023553371429443364 seconds
['wilson@myschool.edu', 'Lucy', 'Wilson', 'DATA202', 'B-', '80']
['taylor@myschool.edu', 'Olivia', 'Taylor', 'DATA203', 'B+', '88']
['smart@myschool.edu', 'Jane', 'Smart', 'DATA200|DATA201|DATA202', 'B+|A-|A', '88|90|96']
['russell@myschool.edu', 'Sally', 'Russell', 'DATA200', 'A-', '90']
['moore@myschool.edu', 'Charles', 'Moore', 'DATA200', 'A', '94']
['miller@myschool.edu', 'William', 'Miller', 'DATA201|DATA202', 'C-|D+', '72|68']
['mae@myschool.edu', 'Samantha', 'Mae', 'DATA200|DATA201', 'B+|A-', '88|92']
['jones@myschool.edu', 'Emily', 'Jones', 'DATA203', 'A-', '92']
['johnson@myschool.edu', 'Jack', 'Johnson', 'DATA200', 'C', '75']
['doe@myschool.edu', 'John', 'Doe', 'DATA201|DATA203', 'A|B+', '94|89']
['davis@myschool.edu', 'Sarah', 'Davis', 'DATA202|DATA203', 'A|B+', '97|88']
['brown@myschool.edu', 'Michael', 'Brown', 'DATA200|DATA203', 'B|C+', '86|78']
['anderson@myschool.edu', 'Benjamin', 'Anderson', 'DATA201', 'C', '75']
Sorting time: 0.0012981891632080078 seconds
Amount of time it took to sort in descending order: 0.0013692378997802734 seconds
ok

----------------------------------------------------------------------
Ran 1 test in 0.004s

OK
PS C:\Users\pakin\OneDrive\Desktop\DATA 200\Assignments\Data200-Lab1>
```

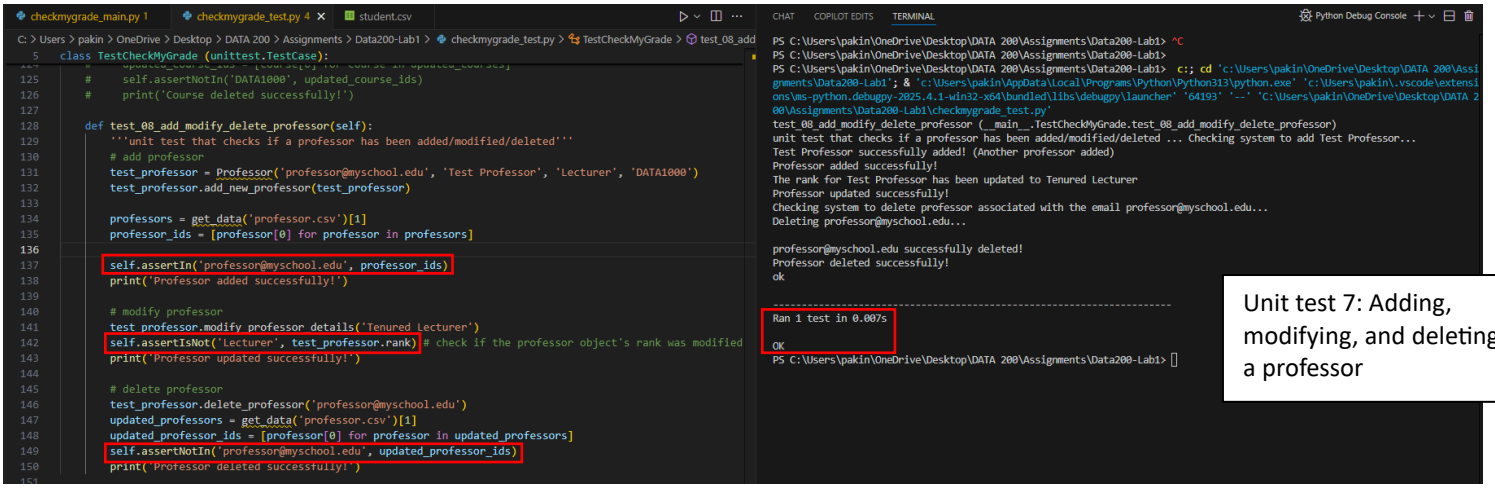
Unit test 6: Checking how long it takes to sort the students alphabetically/in reverse by email



```
def test_07_add_delete_course(self):
    '''unit test that checks if a course has been added/modified/deleted'''
    # add course
    test_course = Course('DATA1000',3,'Data Test Course', 'Test course')
    test_course.add_new_course(test_course)

    courses = get_data('course.csv')[1]
    course_ids = [course[0] for course in courses]

    self.assertIn('DATA1000', course_ids)
    print('Course added successfully!')

    # delete course
    test_course.delete_course('DATA1000')
    updated_courses = get_data('course.csv')[1]
    updated_course_ids = [course[0] for course in updated_courses]
    self.assertNotIn('DATA1000', updated_course_ids)
    print('Course deleted successfully!')
```

```
PS C:\Users\pakin\OneDrive\Desktop\DATA 200\Assignments\Data200-Lab1> ^C
PS C:\Users\pakin\OneDrive\Desktop\DATA 200\Assignments\Data200-Lab1>
PS C:\Users\pakin\OneDrive\Desktop\DATA 200\Assignments\Data200-Lab1>  c:; cd 'c:\Users\pakin\OneDrive\Desktop\DATA 200\Assignments\Data200-Lab1'; & 'c:\Users\pakin\AppData\Local\Programs\Python\Python313\python.exe' 'c:\Users\pakin\.vscode\extensions\ms-python.debugpy-2025.4.1-win32-x64\bundled\libs\debugpy\launcher' '64106' '--' 'C:\Users\pakin\OneDrive\Desktop\DATA 200\Assignments\Data200-Lab1\checkmygrade_test.py'
test_07_add_delete_course (__main__.TestCheckMyGrade.test_07_add_delete_course)
unit test that checks if a course has been added/modified/deleted ... Checking system to add DATA1000...
DATA1000 successfully added! (Another course added)
Course added successfully!
Checking system to delete course associated with the course id DATA1000...
Deleting DATA1000...

DATA1000 successfully deleted!
Course deleted successfully!
ok

----------------------------------------------------------------------
Ran 1 test in 0.007s

OK
PS C:\Users\pakin\OneDrive\Desktop\DATA 200\Assignments\Data200-Lab1>
```

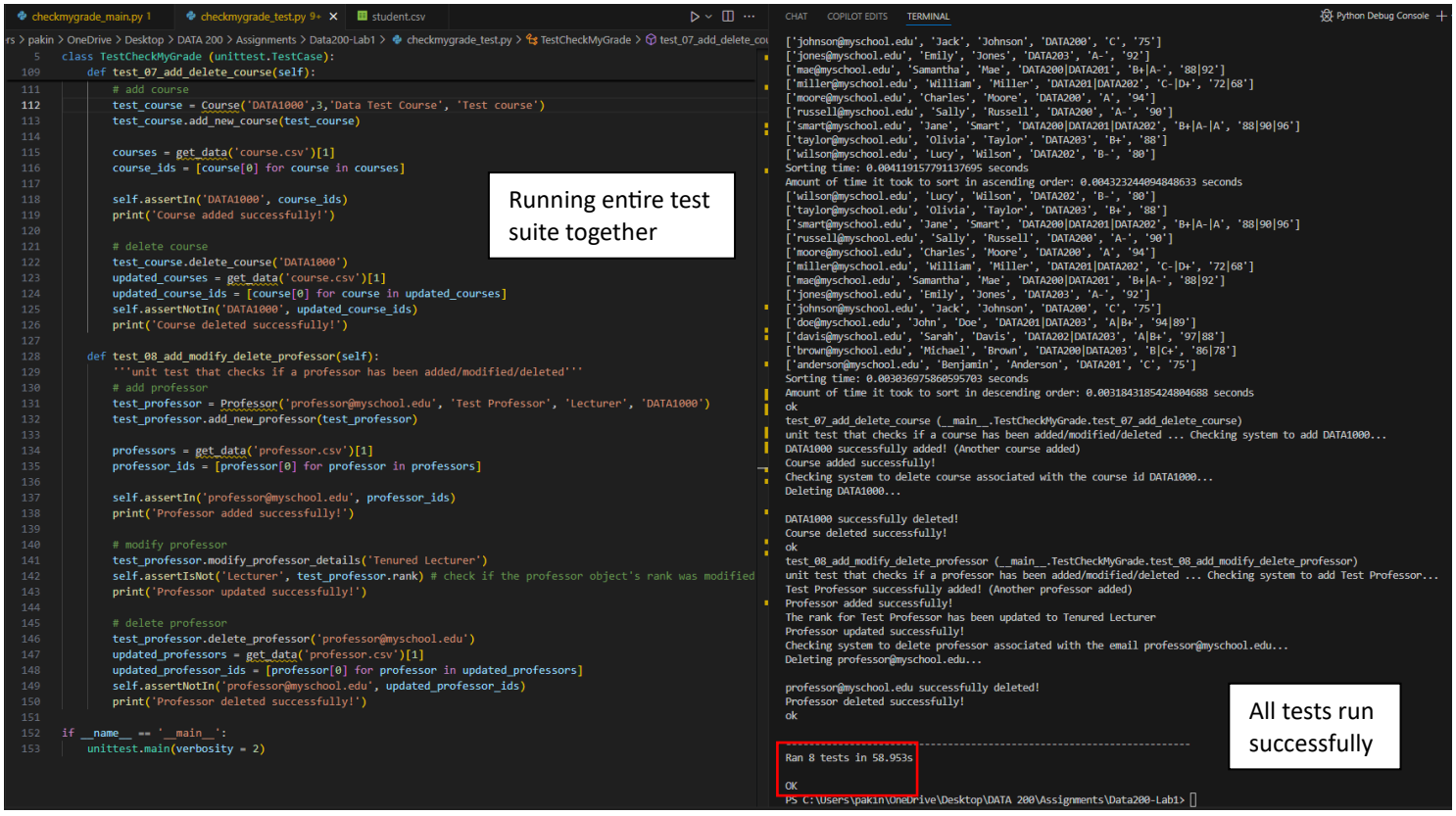
Unit test 6: Adding and deleting a course



```
def test_08_add_modify_delete_professor(self):
    '''unit test that checks if a professor has been added/modified/deleted'''
    # add professor
    test_professor = Professor('professor@myschool.edu', 'Test Professor', 'Lecturer', 'DATA1000')
    test_professor.add_new_professor(test_professor)

    professors = get_data('professor.csv')[1]
    professor_ids = [professor[0] for professor in professors]

    self.assertIn('professor@myschool.edu', professor_ids)
    print('Professor added successfully!')

    # modify professor
    test_professor.modify_professor_details('Tenured Lecturer')
    self.assertIsNot('Lecturer', test_professor.rank) # check if the professor object's rank was modified
    print('Professor updated successfully!')

    # delete professor
    test_professor.delete_professor('professor@myschool.edu')
    updated_professors = get_data('professor.csv')[1]
    updated_professor_ids = [professor[0] for professor in updated_professors]
    self.assertNotIn('professor@myschool.edu', updated_professor_ids)
    print('Professor deleted successfully!')
```

```
PS C:\Users\pakin\OneDrive\Desktop\DATA 200\Assignments\Data200-Lab1> ^C
PS C:\Users\pakin\OneDrive\Desktop\DATA 200\Assignments\Data200-Lab1>
PS C:\Users\pakin\OneDrive\Desktop\DATA 200\Assignments\Data200-Lab1>  c:; cd 'c:\Users\pakin\OneDrive\Desktop\DATA 200\Assignments\Data200-Lab1'; & 'c:\Users\pakin\AppData\Local\Programs\Python\Python313\python.exe' 'c:\Users\pakin\.vscode\extensions\ms-python.debugpy-2025.4.1-win32-x64\bundled\libs\debugpy\launcher' '64193' '--' 'C:\Users\pakin\OneDrive\Desktop\DATA 200\Assignments\Data200-Lab1\checkmygrade_test.py'
test_08_add_modify_delete_professor (__main__.TestCheckMyGrade.test_08_add_modify_delete_professor)
unit test that checks if a professor has been added/modified/deleted ... Checking system to add Test Professor...
Test Professor successfully added! (Another professor added)
Professor added successfully!
The rank for Test Professor has been updated to Tenured Lecturer
Professor updated successfully!
Checking system to delete professor associated with the email professor@myschool.edu...
Deleting professor@myschool.edu...

professor@myschool.edu successfully deleted!
Professor deleted successfully!
ok

----------------------------------------------------------------------
Ran 1 test in 0.007s

OK
PS C:\Users\pakin\OneDrive\Desktop\DATA 200\Assignments\Data200-Lab1>
```

Unit test 7: Adding, modifying, and deleting a professor

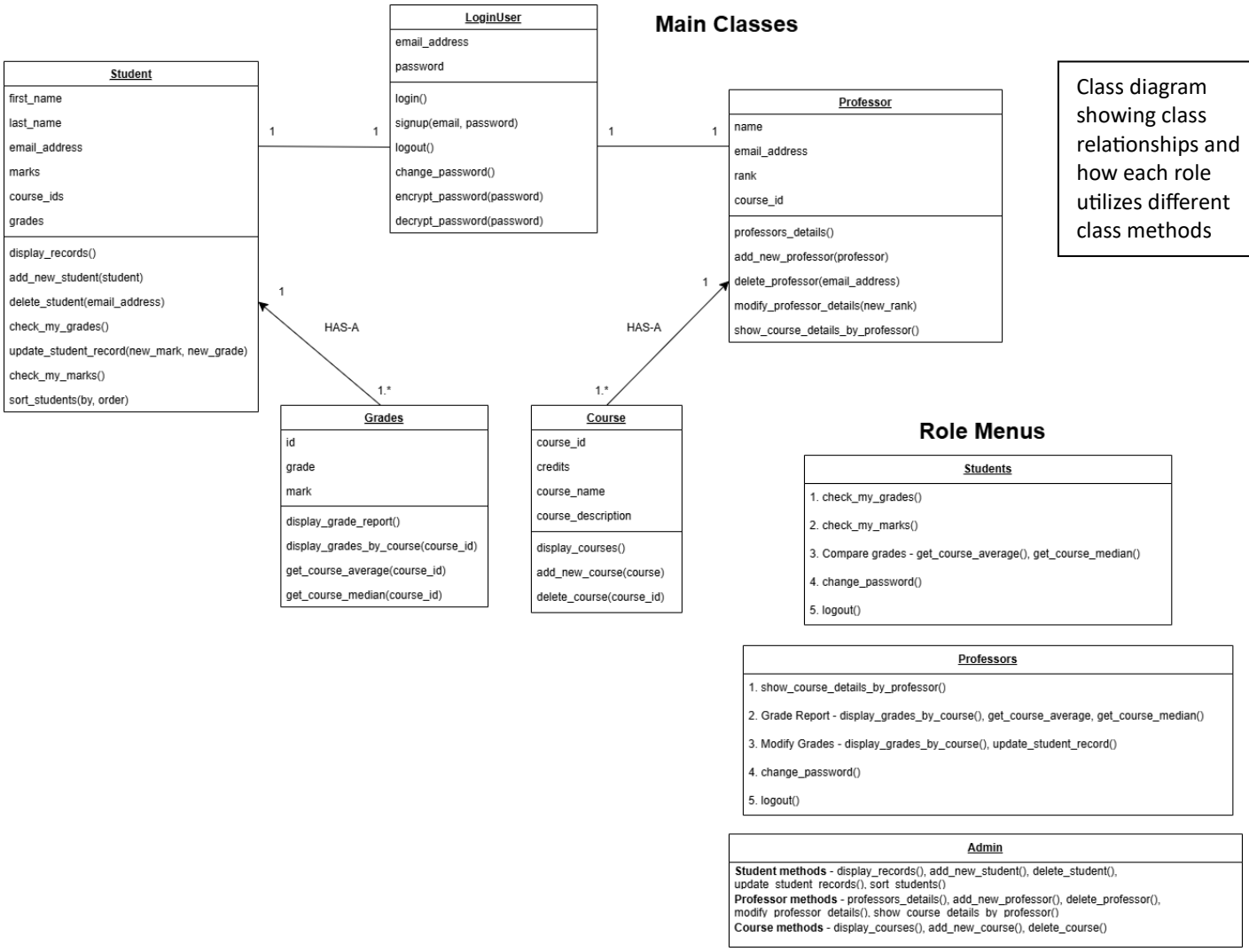


```
class TestCheckMyGrade (unittest.TestCase):
    def test_07_add_delete_course(self):
        # add course
        test_course = Course('DATA1000',3,'Data Test Course', 'Test course')
        test_course.add_new_course(test_course)

        courses = get_data('course.csv')[1]
        course_ids = [course[0] for course in courses]

        self.assertIn('DATA1000', course_ids)
        print('Course added successfully!')

        # delete course
        test_course.delete_course('DATA1000')
        updated_courses = get_data('course.csv')[1]
        updated_course_ids = [course[0] for course in updated_courses]
        self.assertNotIn('DATA1000', updated_course_ids)
        print('Course deleted successfully!')

    def test_08_add_modify_delete_professor(self):
        '''unit test that checks if a professor has been added/modified/deleted'''
        # add professor
        test_professor = Professor('professor@myschool.edu', 'Test Professor', 'Lecturer', 'DATA1000')
        test_professor.add_new_professor(test_professor)

        professors = get_data('professor.csv')[1]
        professor_ids = [professor[0] for professor in professors]

        self.assertIn('professor@myschool.edu', professor_ids)
        print('Professor added successfully!')

        # modify professor
        test_professor.modify_professor_details('Tenured Lecturer')
        self.assertIsNot('Lecturer', test_professor.rank) # check if the professor object's rank was modified
        print('Professor updated successfully!')

        # delete professor
        test_professor.delete_professor('professor@myschool.edu')
        updated_professors = get_data('professor.csv')[1]
        updated_professor_ids = [professor[0] for professor in updated_professors]
        self.assertNotIn('professor@myschool.edu', updated_professor_ids)
        print('Professor deleted successfully!')

if __name__ == '__main__':
    unittest.main(verbosity = 2)
```

Running entire test suite together

```
['johnson@myschool.edu', 'Jack', 'Johnson', 'DATA200', 'C', '75']
['jones@myschool.edu', 'Emily', 'Jones', 'DATA203', 'A-', '92']
['mae@myschool.edu', 'Samantha', 'Mae', 'DATA200|DATA201', 'B+|A-', '88|92']
['miller@myschool.edu', 'William', 'Miller', 'DATA201|DATA202', 'C-|D+', '72|68']
['moore@myschool.edu', 'Charles', 'Moore', 'DATA200', 'A', '94']
['russell@myschool.edu', 'Sally', 'Russell', 'DATA200', 'A-', '90']
['smart@myschool.edu', 'Jane', 'Smart', 'DATA200|DATA201|DATA202', 'B+|A-|A', '88|90|96']
['taylor@myschool.edu', 'Olivia', 'Taylor', 'DATA203', 'B+', '88']
['wilson@myschool.edu', 'Lucy', 'Wilson', 'DATA202', 'B-', '80']
Sorting time: 0.00411915779137695 seconds
Amount of time it took to sort in ascending order: 0.004323244094848633 seconds
['wilson@myschool.edu', 'Lucy', 'Wilson', 'DATA202', 'B-', '80']
['taylor@myschool.edu', 'Olivia', 'Taylor', 'DATA203', 'B+', '88']
['smart@myschool.edu', 'Jane', 'Smart', 'DATA200|DATA201|DATA202', 'B+|A-|A', '88|90|96']
['russell@myschool.edu', 'Sally', 'Russell', 'DATA200', 'A-', '90']
['moore@myschool.edu', 'Charles', 'Moore', 'DATA200', 'A', '94']
['miller@myschool.edu', 'William', 'Miller', 'DATA201|DATA202', 'C-|D+', '72|68']
['mae@myschool.edu', 'Samantha', 'Mae', 'DATA200|DATA201', 'B+|A-', '88|92']
['jones@myschool.edu', 'Emily', 'Jones', 'DATA203', 'A-', '92']
['johnson@myschool.edu', 'Jack', 'Johnson', 'DATA200', 'C', '75']
['doe@myschool.edu', 'John', 'Doe', 'DATA201|DATA203', 'A|B+', '94|89']
['davis@myschool.edu', 'Sarah', 'Davis', 'DATA202|DATA203', 'A|B+', '97|88']
['brown@myschool.edu', 'Michael', 'Brown', 'DATA200|DATA203', 'B|C+', '86|78']
['anderson@myschool.edu', 'Benjamin', 'Anderson', 'DATA201', 'C', '75']
Sorting time: 0.0030369758605957031 seconds
Amount of time it took to sort in descending order: 0.0031843185424804688 seconds
ok
test_07_add_delete_course (__main__.TestCheckMyGrade.test_07_add_delete_course)
unit test that checks if a course has been added/modified/deleted ... Checking system to add DATA1000...
DATA1000 successfully added! (Another course added)
Course added successfully!
Checking system to delete course associated with the course id DATA1000...
Deleting DATA1000...

DATA1000 successfully deleted!
Course deleted successfully!
ok
test_08_add_modify_delete_professor (__main__.TestCheckMyGrade.test_08_add_modify_delete_professor)
unit test that checks if a professor has been added/modified/deleted ... Checking system to add Test Professor...
Test Professor successfully added! (Another professor added)
Professor added successfully!
The rank for Test Professor has been updated to Tenured Lecturer
Professor updated successfully!
Checking system to delete professor associated with the email professor@myschool.edu...
Deleting professor@myschool.edu...

professor@myschool.edu successfully deleted!
Professor deleted successfully!
ok

----------------------------------------------------------------------
Ran 8 tests in 58.953s

OK
PS C:\Users\pakin\OneDrive\Desktop\DATA 200\Assignments\Data200-Lab1>
```

All tests run successfully



Class diagram showing class relationships and how each role utilizes different class methods

GitHub repository: https://github.com/lpakingan/Data200-Lab1