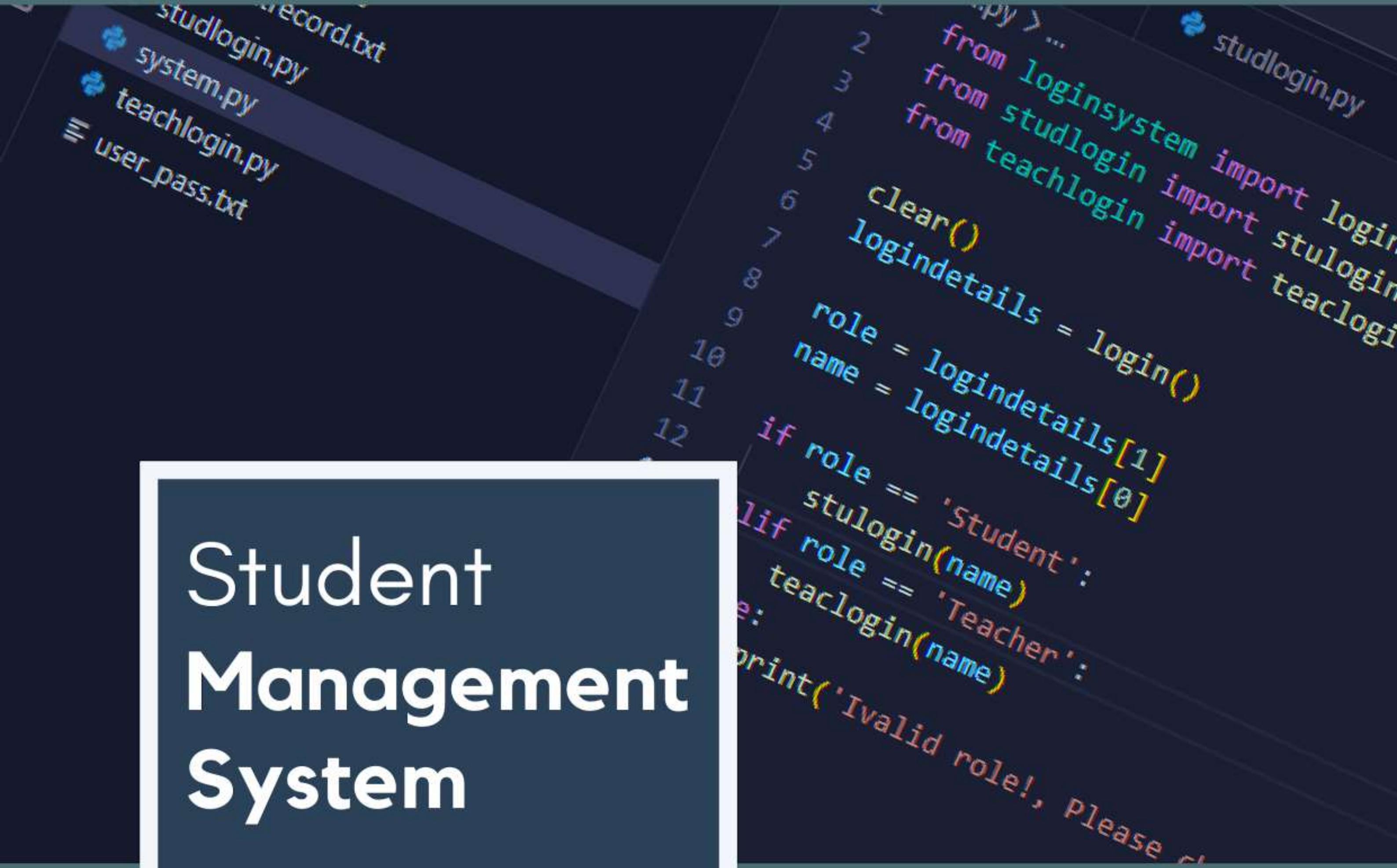
Submitted on: 19.06.2023

Team - 8
mini PROJECT



```
.py > ...
from loginsystem import login
from studlogin import stulogin
from teachlogin import teaclogi

clear()
logindetails = login()

role = logindetails[1]
name = logindetails[0]

if role == 'Student':
    stulogin(name)
lif role == 'Teacher':
    teaclogin(name)
e:
    print('Invalid role!, Please c
```

Student **Management System**

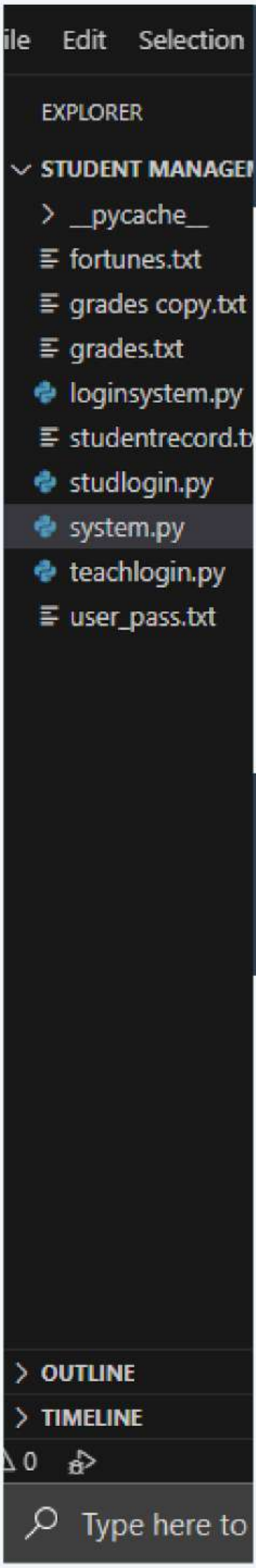
Python

Prepared By:
Nitheesh Kani [921722115039]
collaboration with:
P T Mohamed Anas [921722115032]
B Bharath [921722115008]

ABOUT THE PROJECT

The Student Management System is a Python-based software application designed to facilitate efficient management of student records and enhance communication between teachers and students in an educational institution. This report provides an overview of the project, outlining its objectives, key features, and significance in the context of educational administration.

The Student Management System is a software application developed to streamline administrative tasks and facilitate efficient communication between teachers and students in an educational institution. This report presents an overview of the system, including its purpose, objectives, and key features



VISION

Our vision is to create a comprehensive and user-friendly Student Management System that empowers educational institutions to efficiently manage student records and facilitate effective communication between teachers and students. We aim to streamline administrative processes, enhance data accuracy, and provide a platform that promotes collaboration and engagement within the education community.

MISSION

1. Streamline Administrative Processes: Our system aims to automate and streamline time-consuming administrative tasks, such as record keeping and grade management. By eliminating manual processes, we aim to save time and improve overall efficiency.
2. Continuously Improve and Innovate: We are dedicated to continuously improving our Student Management System based on user feedback and emerging needs in the education sector.

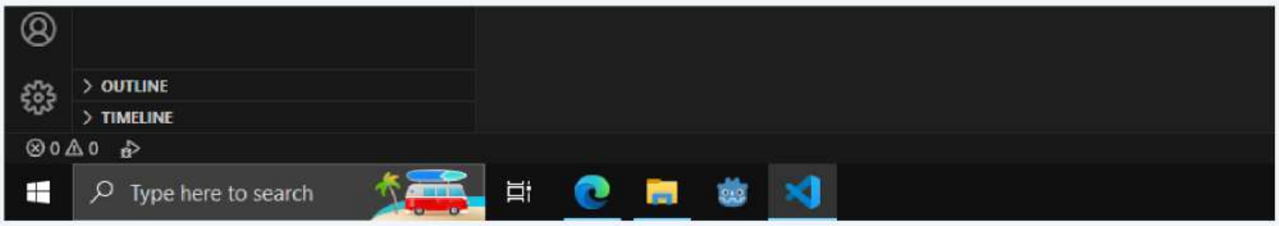
ABSTRACT

The Student Management System is a Python-based software application designed to streamline administrative tasks and enhance communication between teachers and students in an educational institution. The system incorporates a login mechanism to authenticate users and provides different functionalities based on their roles. Teachers can access and modify student records and grades, while students have the ability to view their records and a personalized fortune. The system utilizes a text file for storing user credentials and student data.

This project report presents a comprehensive overview of the Student Management System, including its design, implementation, and testing. The report highlights the system's architecture, the programming language and libraries used, and the mechanisms employed for user authentication and data storage. It also discusses the user interface design, testing procedures, and evaluation of the system's performance.

Through the development of this Student Management System, the project aims to achieve increased efficiency, improved data management, and enhanced user experience within the educational institution. The report concludes by discussing the project's successes, potential areas for improvement, and the overall impact of the system on the institution's administrative processes.

The Student Management System project holds immense potential to revolutionize student record management and communication, ultimately benefiting both teachers and students by providing a streamlined and user-friendly platform for administrative tasks.



THE REQUIEMENTS

Student Information Management

The system allows administrators to store and manage comprehensive student profiles, including personal details, contact information, enrollment details, and emergency contacts.

Academic Performance Tracking

It facilitates the recording and management of student grades, assessments, assignments, and examinations. It enables teachers to generate progress reports and analyze student performance.

Communication and Collaboration

It provides a platform for effective communication between teachers, students, and parents. It may include features such as messaging, notifications, announcements, and online discussion forums.

THE SOLUTIONS

The Login System

The provided login system is a basic implementation that allows users to authenticate by entering their username and password.
The user information (role, name, and code) is stored in the user_pass.txt file. Each line in the file represents a user's credentials, with the role, name, and code separated by commas. The system reads this file to verify the user's credentials.

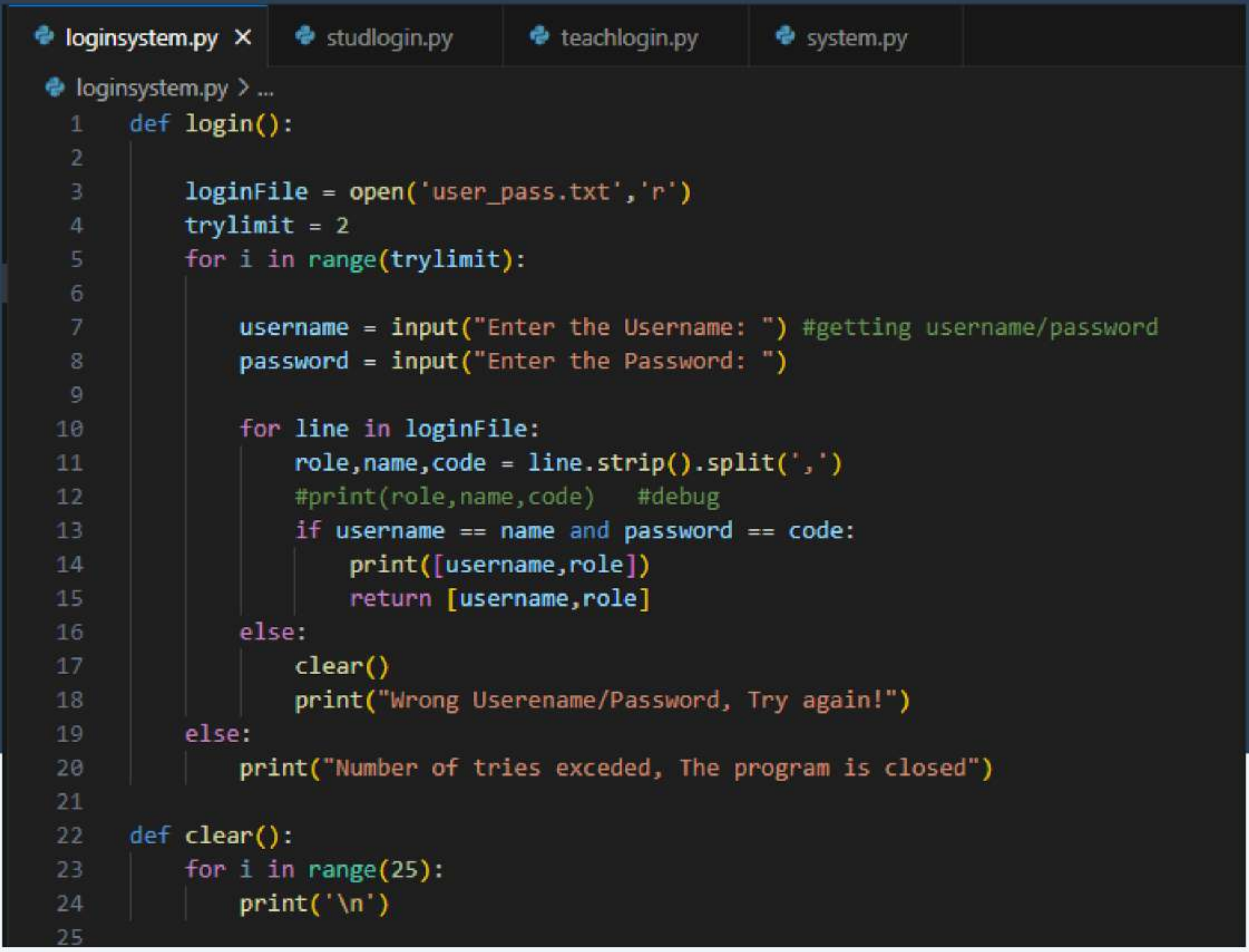
The Student Information System

the student information system provides basic functionality for managing and displaying student records. Users can view all student records, search for specific student records based on various criteria, and access their own personal record. The system employs a menu-driven approach to facilitate user interaction and navigation through different features.

The Grading System

, the grading system provides functionality for managing and displaying student records, storing and presenting student grades, and offering additional features such as displaying random fortunes. It offers a menu-driven approach for users to interact with the system and access different functionalities based on their needs.

THE LOGIN SYSTEM



```python
def login():

    loginFile = open('user_pass.txt','r')
    trylimit = 2
    for i in range(trylimit):

        username = input("Enter the Username: ") #getting username/password
        password = input("Enter the Password: ")

        for line in loginFile:
            role,name,code = line.strip().split(',')
            #print(role,name,code)   #debug
            if username == name and password == code:
                print([username,role])
                return [username,role]
        else:
            clear()
            print("Wrong Userename/Password, Try again!")
    else:
        print("Number of tries exceded, The program is closed")

def clear():
    for i in range(25):
        print('\n')

```

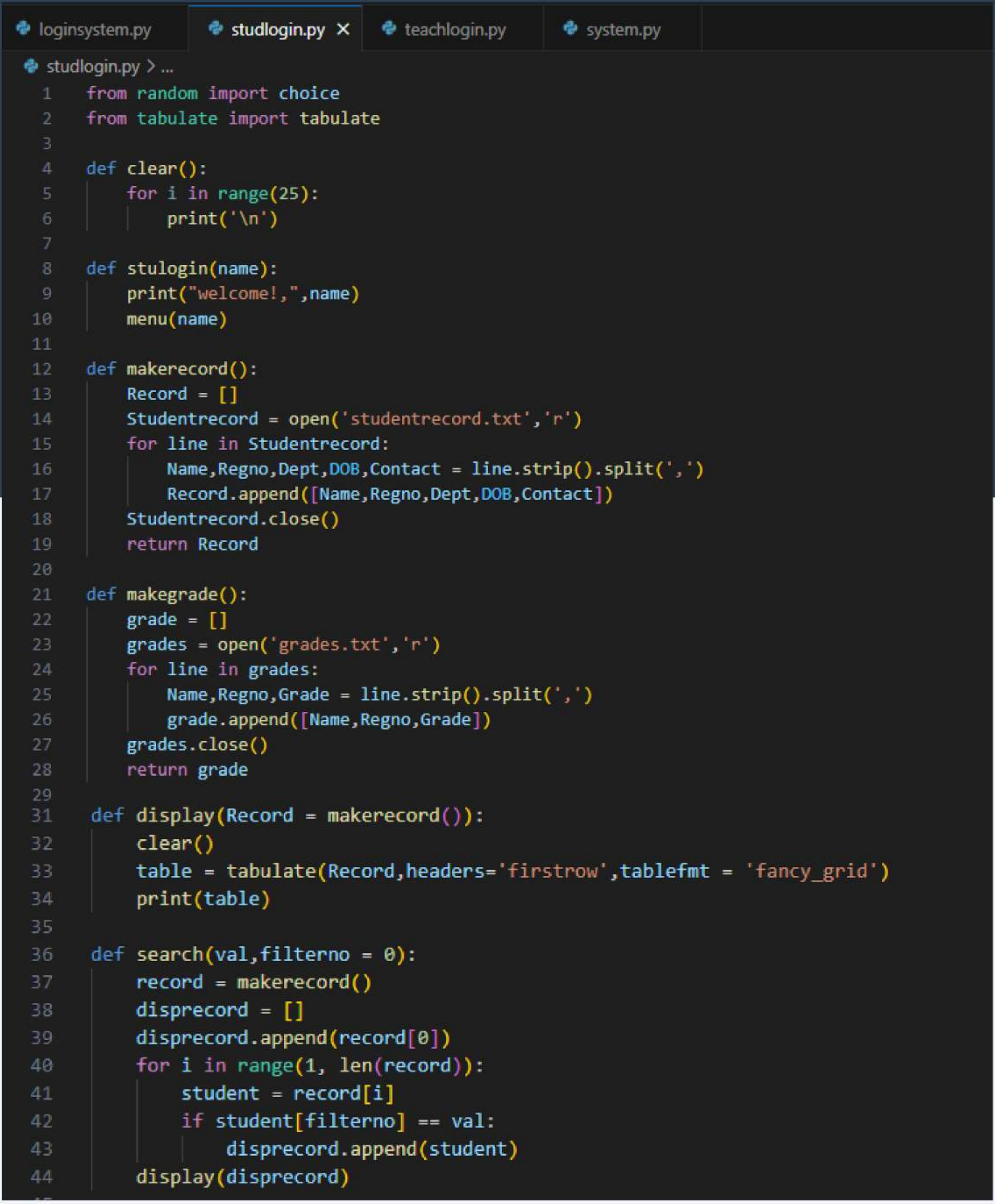
The provided login system is a basic implementation that allows users to authenticate by entering their username and password. Here are some key points about the system:

1. Authentication: The login system checks the entered username and password against the information stored in a file (**user_pass.txt**). It compares the entered credentials with each line in the file until a match is found or the maximum number of login attempts is reached.
2. User Information Storage: The user information (role, name, and code) is stored in the **user_pass.txt** file. Each line in the file represents a user's credentials, with the role, name, and code separated by commas. The system reads this file to verify the user's credentials.

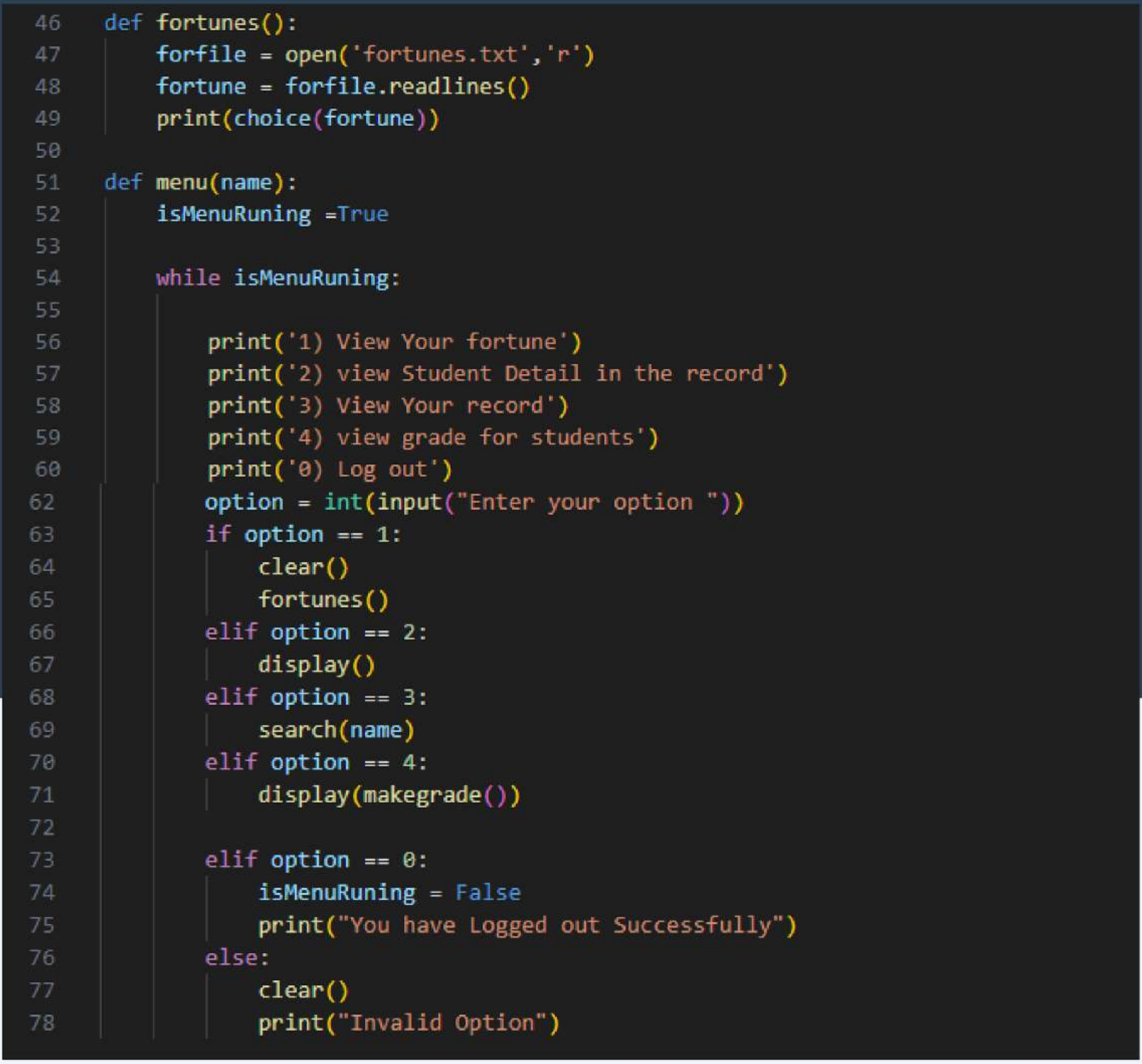
1. Limited Login Attempts: The system limits the number of login attempts to a specified value (**trylimit**). If the user fails to provide valid credentials within the allowed attempts, the program terminates.
2. User Roles: The system associates each user with a role. When a user successfully logs in, the system retrieves their role from the file and returns it along with the username. This can be useful for implementing different levels of access or privileges based on the user's role.
3. Clearing the Screen: The system includes a custom **clear** function that attempts to clear the screen by printing newline characters. This feature aims to enhance user experience by removing previous input or output from the display.

The login system implemented in the code provides a basic authentication mechanism for users. It prompts the user to enter their username and password, compares the input against the information stored in a file, and grants access if the credentials match. The system allows for a limited number of login attempts before terminating the program to prevent unauthorized access.
The user information, including the role, name, and code, is stored in a file named 'user_pass.txt'. Each line in the file represents a user's credentials, with the fields separated by commas. Upon successful login, the system retrieves the user's role and displays a welcome message along with their name.
The login system also includes a custom 'clear' function that attempts to clear the screen by printing newline characters. This feature enhances the user experience by removing previous input or output from the display.
While the login system provides a basic level of authentication, there are areas for improvement. For instance, implementing more robust security measures like password hashing and salting could enhance the system's security. Additionally, incorporating error handling and input validation would make the system more resilient to invalid or malicious inputs. Nonetheless, the existing login system serves as a starting point and can be further enhanced to meet specific security requirements.

THE STUDENT INFORMATION SYSTEM



```python
from random import choice
from tabulate import tabulate

def clear():
    for i in range(25):
        print('\n')

def stulogin(name):
    print("welcome!,",name)
    menu(name)

def makerecord():
    Record = []
    Studentrecord = open('studentrecord.txt','r')
    for line in Studentrecord:
        Name,Regno,Dept,DOB,Contact = line.strip().split(',')
        Record.append([Name,Regno,Dept,DOB,Contact])
    Studentrecord.close()
    return Record

def makegrade():
    grade = []
    grades = open('grades.txt','r')
    for line in grades:
        Name,Regno,Grade = line.strip().split(',')
        grade.append([Name,Regno,Grade])
    grades.close()
    return grade

def display(Record = makerecord()):
    clear()
    table = tabulate(Record,headers='firstrow',tablefmt = 'fancy_grid')
    print(table)

def search(val,filterno = 0):
    record = makerecord()
    disprecord = []
    disprecord.append(record[0])
    for i in range(1, len(record)):
        student = record[i]
        if student[filterno] == val:
            disprecord.append(student)
    display(disprecord)
```



```python
def fortunes():
    forfile = open('fortunes.txt','r')
    fortune = forfile.readlines()
    print(choice(fortune))

def menu(name):
    isMenuRuning =True

    while isMenuRuning:

        print('1) View Your fortune')
        print('2) view Student Detail in the record')
        print('3) View Your record')
        print('4) view grade for students')
        print('0) Log out')
        option = int(input("Enter your option "))
        if option == 1:
            clear()
            fortunes()
        elif option == 2:
            display()
        elif option == 3:
            search(name)
        elif option == 4:
            display(makegrade())

        elif option == 0:
            isMenuRuning = False
            print("You have Logged out Successfully")
        else:
            clear()
            print("Invalid Option")
```

The provided code implements a student information system that allows users to view and search for student records. Here are some key points about the student information system:

1. Student Record Storage: The system reads student information from a file named `studentrecord.txt`. Each line in the file represents a student's information, including their name, registration number, department, date of birth, and contact. The `makerecord` function reads this file and creates a list called `Record` to store the student records.

2. Displaying Student Records: The `display` function uses the `tabulate` module to present the student records in a tabular format. It retrieves the records from the `Record` list (created by `makerecord`) and prints them with appropriate headers.

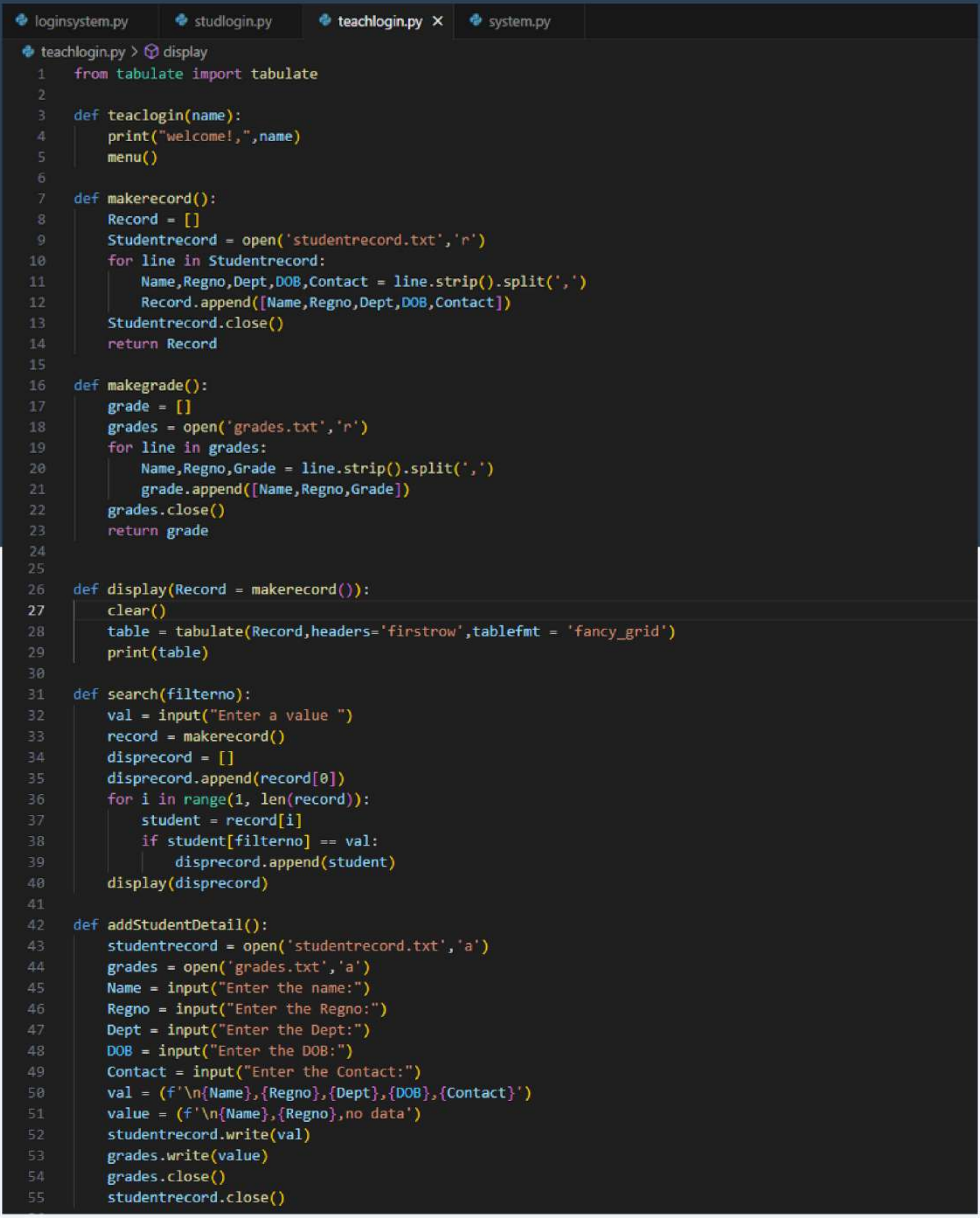
3. Searching Student Records: The `search` function allows users to search for specific student records based on a provided value and an optional filter number. It retrieves the student records using `makerecord`, applies the filter based on the provided value and filter number, and displays the filtered records using the `display` function.

4. Menu Functionality: The `menu` function represents the main menu of the student information system. It offers various options to the user, including viewing student records, searching for student records, and viewing personal records. The menu function also provides the option to log out and exit the system.

5. Additional Features: The student information system includes a `fortunes` function that reads fortunes from a file named `fortunes.txt` and randomly selects and prints one of them. This feature adds an extra element of user engagement to the system.

Overall, the student information system provides basic functionality for managing and displaying student records. Users can view all student records, search for specific student records based on various criteria, and access their own personal record. The system employs a menu-driven approach to facilitate user interaction and navigation through different features.

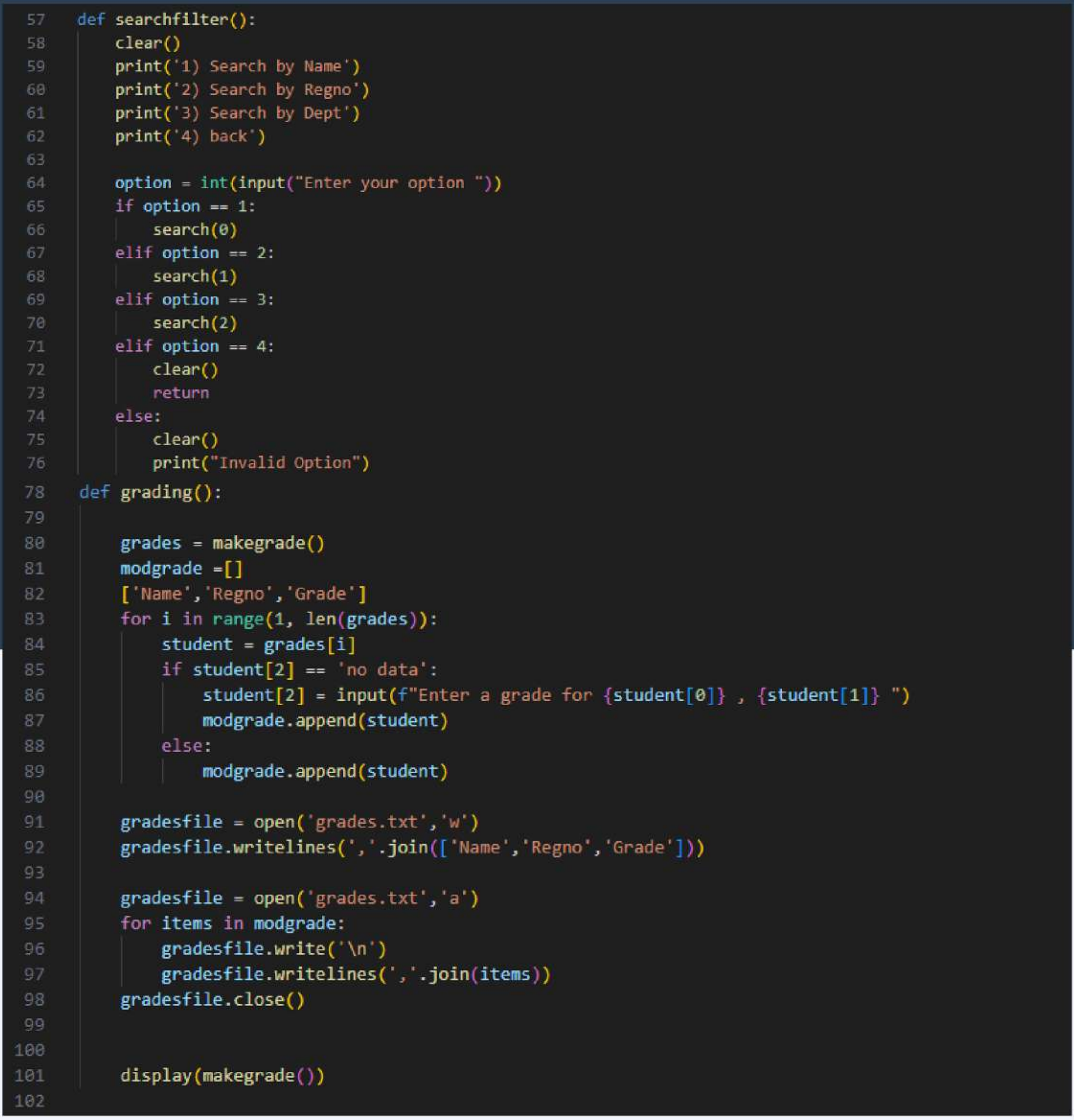
THE GRADING SYSTEM



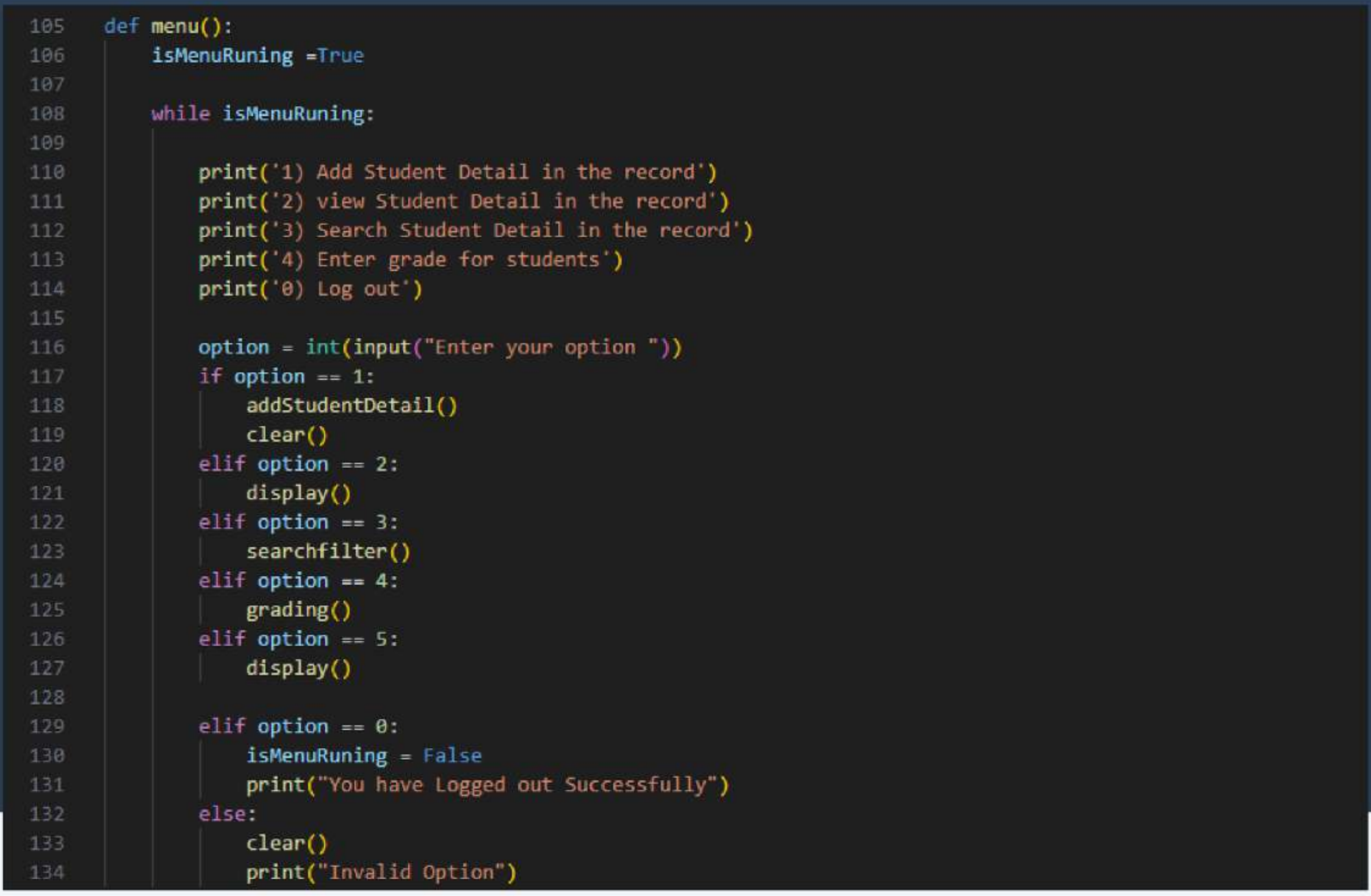
loginsystem.py | studlogin.py | teachlogin.py × | system.py

teachlogin.py > display

```python
from tabulate import tabulate

def teaclogin(name):
    print("welcome!,",name)
    menu()

def makerecord():
    Record = []
    Studentrecord = open('studentrecord.txt','r')
    for line in Studentrecord:
        Name,Regno,Dept,DOB,Contact = line.strip().split(',')
        Record.append([Name,Regno,Dept,DOB,Contact])
    Studentrecord.close()
    return Record

def makegrade():
    grade = []
    grades = open('grades.txt','r')
    for line in grades:
        Name,Regno,Grade = line.strip().split(',')
        grade.append([Name,Regno,Grade])
    grades.close()
    return grade


def display(Record = makerecord()):
    clear()
    table = tabulate(Record,headers='firstrow',tablefmt = 'fancy_grid')
    print(table)

def search(filterno):
    val = input("Enter a value ")
    record = makerecord()
    disprecord = []
    disprecord.append(record[0])
    for i in range(1, len(record)):
        student = record[i]
        if student[filterno] == val:
            disprecord.append(student)
    display(disprecord)

def addStudentDetail():
    studentrecord = open('studentrecord.txt','a')
    grades = open('grades.txt','a')
    Name = input("Enter the name:")
    Regno = input("Enter the Regno:")
    Dept = input("Enter the Dept:")
    DOB = input("Enter the DOB:")
    Contact = input("Enter the Contact:")
    val = (f'\n{Name},{Regno},{Dept},{DOB},{Contact}')
    value = (f'\n{Name},{Regno},no data')
    studentrecord.write(val)
    grades.write(value)
    grades.close()
    studentrecord.close()
```



```python
def searchfilter():
    clear()
    print('1) Search by Name')
    print('2) Search by Regno')
    print('3) Search by Dept')
    print('4) back')

    option = int(input("Enter your option "))
    if option == 1:
        search(0)
    elif option == 2:
        search(1)
    elif option == 3:
        search(2)
    elif option == 4:
        clear()
        return
    else:
        clear()
        print("Invalid Option")
def grading():

    grades = makegrade()
    modgrade =[]
    ['Name','Regno','Grade']
    for i in range(1, len(grades)):
        student = grades[i]
        if student[2] == 'no data':
            student[2] = input(f"Enter a grade for {student[0]} , {student[1]} ")
            modgrade.append(student)
        else:
            modgrade.append(student)

    gradesfile = open('grades.txt','w')
    gradesfile.writelines(','.join(['Name','Regno','Grade']))

    gradesfile = open('grades.txt','a')
    for items in modgrade:
        gradesfile.write('\n')
        gradesfile.writelines(','.join(items))
    gradesfile.close()


    display(makegrade())

```



```python
def menu():
    isMenuRuning =True

    while isMenuRuning:

        print('1) Add Student Detail in the record')
        print('2) view Student Detail in the record')
        print('3) Search Student Detail in the record')
        print('4) Enter grade for students')
        print('0) Log out')

        option = int(input("Enter your option "))
        if option == 1:
            addStudentDetail()
            clear()
        elif option == 2:
            display()
        elif option == 3:
            searchfilter()
        elif option == 4:
            grading()
        elif option == 5:
            display()

        elif option == 0:
            isMenuRuning = False
            print("You have Logged out Successfully")
        else:
            clear()
            print("Invalid Option")
```

The updated code provides an expanded student information system with additional functionality for teachers. Here are the notable features of the system:

1. Teacher Login: The `teaclogin` function is called after a successful login for a teacher. It displays a welcome message and then calls the `menu` function to present the teacher-specific options.

2. Adding Student Details: The `addStudentDetail` function allows teachers to add student information to the record. Teachers are prompted to input the student's name, registration number, department, date of birth, and contact details. The information is then appended to the `studentrecord.txt` file as a new line. Additionally, a default grade of "no data" is added to the `grades.txt` file for the newly added student.

3. Displaying Student Details: The `display` function is used to show the student details stored in the `studentrecord.txt` file. It retrieves the records using the `makerecord` function and displays them in a tabular format using the `tabulate` module.

4. Searching Student Details: The `search` function enables teachers to search for specific student records based on various filter options, including name, registration number, and department. Teachers are prompted to enter a value to search for, and the matching records are displayed using the `display` function.

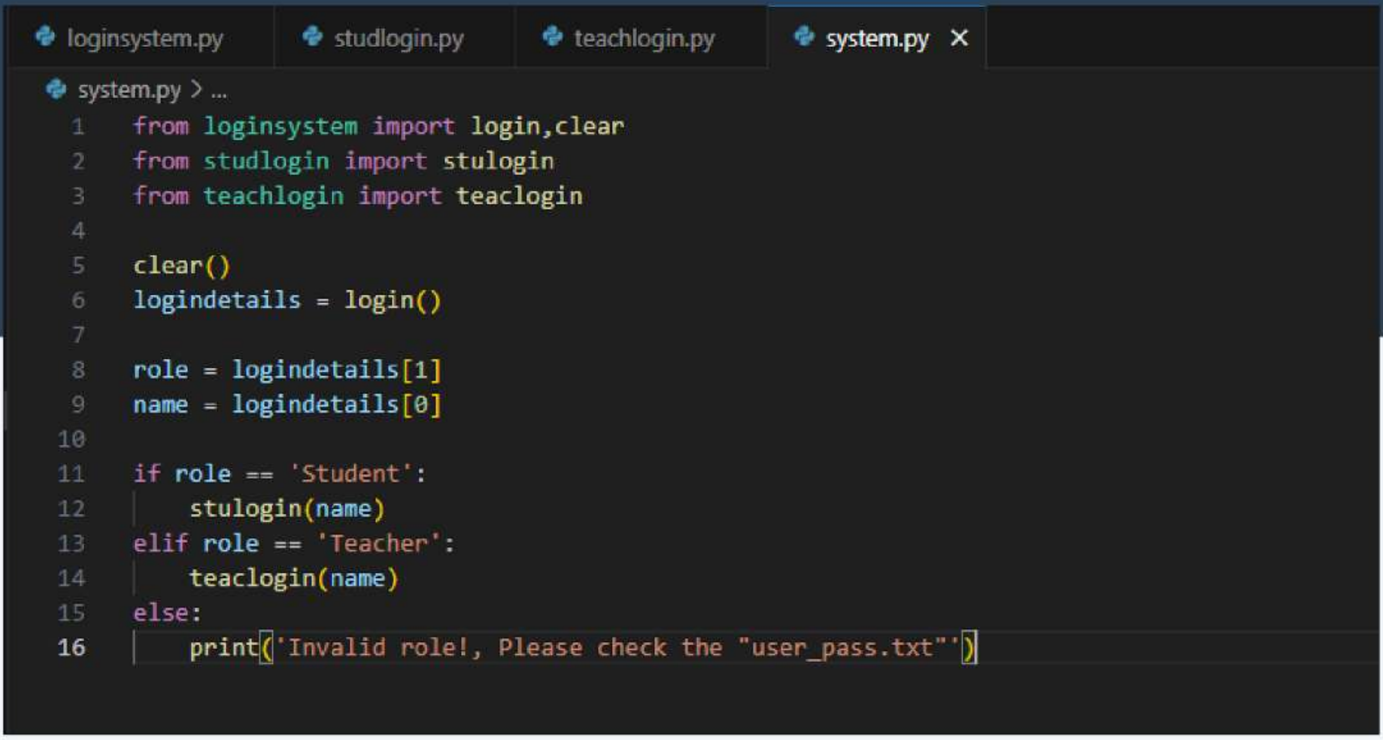
5. Grade Entry for Students: The `grading` function allows teachers to enter grades for students. It retrieves the student grades using the `makegrade` function, and teachers are prompted to input the grades for students who have a default grade of "no data." The modified grades are then updated in the `grades.txt` file, and the updated grades are displayed using the `display` function.

6. Menu Functionality: The `menu` function presents a menu of options specifically for teachers. It includes options to add student details, view student details, search student details, enter grades for students, and log out.

7. Additional Features: The `searchfilter` function offers a sub-menu for teachers to select the filter option for searching student details. It provides options to search by name, registration number, or department. Furthermore, the `clear` function is provided to clear the screen by printing newline characters.

Overall, this updated information system extends the functionality to include teacher-specific features. Teachers can add student details, view and search student records, enter grades for students, and navigate through the system using the menu-based interface.

THE LINK



```python
from loginsystem import login,clear
from studlogin import stulogin
from teachlogin import teaclogin

clear()
logindetails = login()

role = logindetails[1]
name = logindetails[0]

if role == 'Student':
    stulogin(name)
elif role == 'Teacher':
    teaclogin(name)
else:
    print('Invalid role!, Please check the "user_pass.txt"')
```

I The main code you provided integrates these modules to create a login system and route users based on their roles. Here's an overview of how the code works:

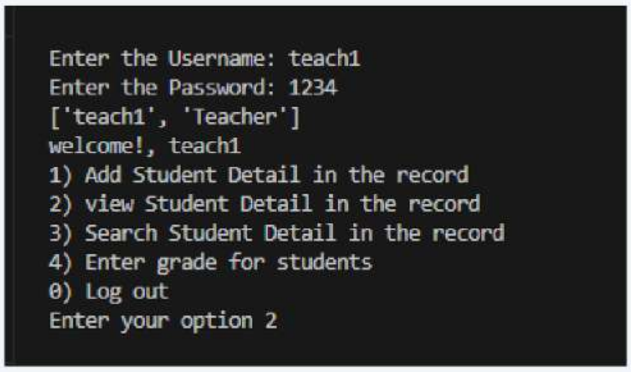
1. The `login` function from the `loginsystem` module is imported, along with the `clear` function.
2. The `clear` function is called to clear the screen.
3. The `login` function is called, which prompts the user to enter their username and password. It validates the credentials by reading from the `user_pass.txt` file and returns the username and role.
4. The role and name from the login details are assigned to variables.
5. Based on the role, the code branches into different paths:
   - If the role is "Student," the `stulogin` function from the `studlogin` module is called, passing the name as an argument.
   - If the role is "Teacher," the `teaclogin` function from the `teachlogin` module is called, passing the name as an argument.
   - If the role is neither "Student" nor "Teacher," an error message is printed.
6. The appropriate login function is responsible for further interactions and menu options based on the user's role.

It appears that the `stulogin` and `teaclogin` functions are responsible for displaying the menu and handling user interactions based on their roles.

Please note that the code assumes the existence of the `user_pass.txt` file, which should contain valid username, password, and role combinations. Make sure the file is correctly formatted and accessible in the expected location.

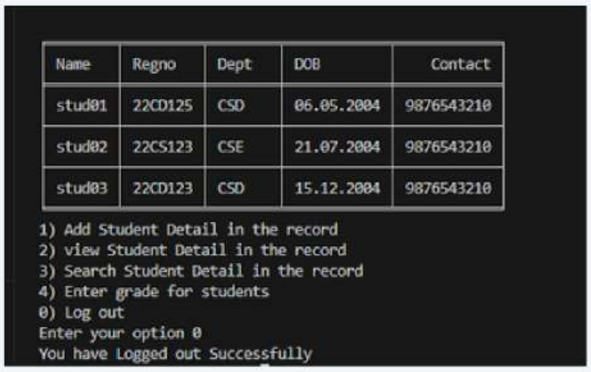
OUTPUT

LOGIN AS TEACHER



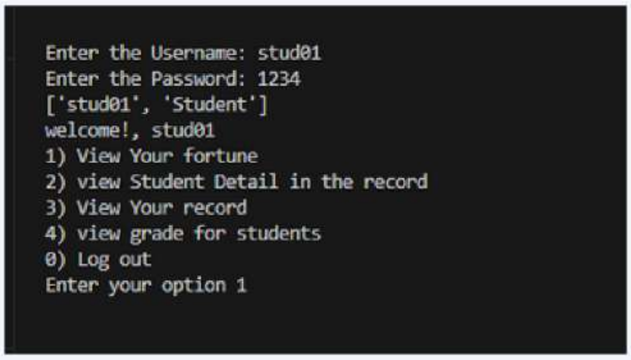
```
Enter the Username: teach1
Enter the Password: 1234
['teach1', 'Teacher']
welcome!, teach1
1) Add Student Detail in the record
2) view Student Detail in the record
3) Search Student Detail in the record
4) Enter grade for students
0) Log out
Enter your option 2
```

VIEWING RECORD



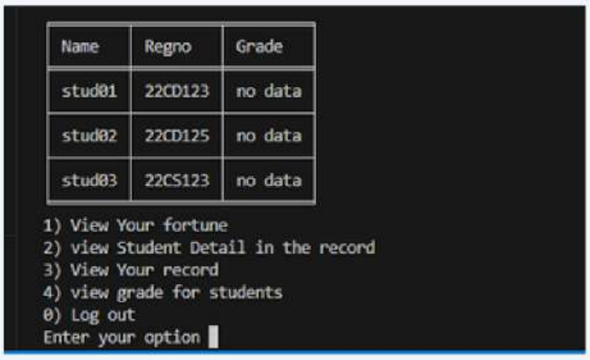
```
| Name   | Regno   | Dept | DOB        |    Contact |
| stud01 | 22CD125 | CSD  | 06.05.2004 | 9876543210 |
| stud02 | 22CS123 | CSE  | 21.07.2004 | 9876543210 |
| stud03 | 22CD123 | CSD  | 15.12.2004 | 9876543210 |
1) Add Student Detail in the record
2) view Student Detail in the record
3) Search Student Detail in the record
4) Enter grade for students
0) Log out
Enter your option 0
You have Logged out Successfully
```

LOGIN AS STUDENT



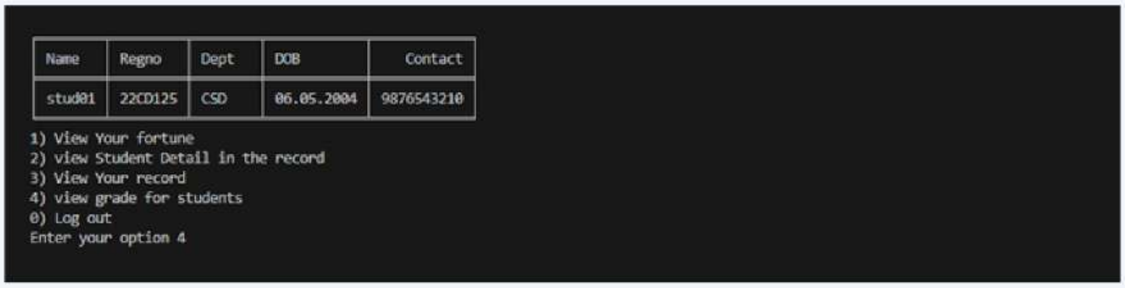
```
Enter the Username: stud01
Enter the Password: 1234
['stud01', 'Student']
welcome!, stud01
1) View Your fortune
2) view Student Detail in the record
3) View Your record
4) view grade for students
0) Log out
Enter your option 1
```

VIEWING GRADE



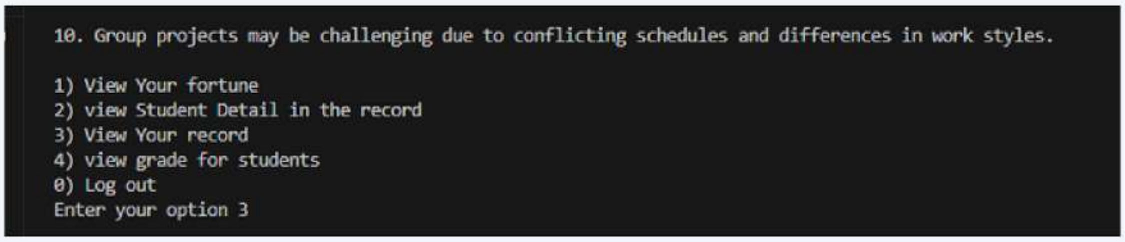
```
| Name   | Regno   | Grade   |
| stud01 | 22CD123 | no data |
| stud02 | 22CD125 | no data |
| stud03 | 22CS123 | no data |
1) View Your fortune
2) view Student Detail in the record
3) View Your record
4) view grade for students
0) Log out
Enter your option
```

VIEWING STUDENT



```
| Name   | Regno   | Dept | DOB        |    Contact |
| stud01 | 22CD125 | CSD  | 06.05.2004 | 9876543210 |
1) View Your fortune
2) view Student Detail in the record
3) View Your record
4) view grade for students
0) Log out
Enter your option 4
```

THE FORTUNE



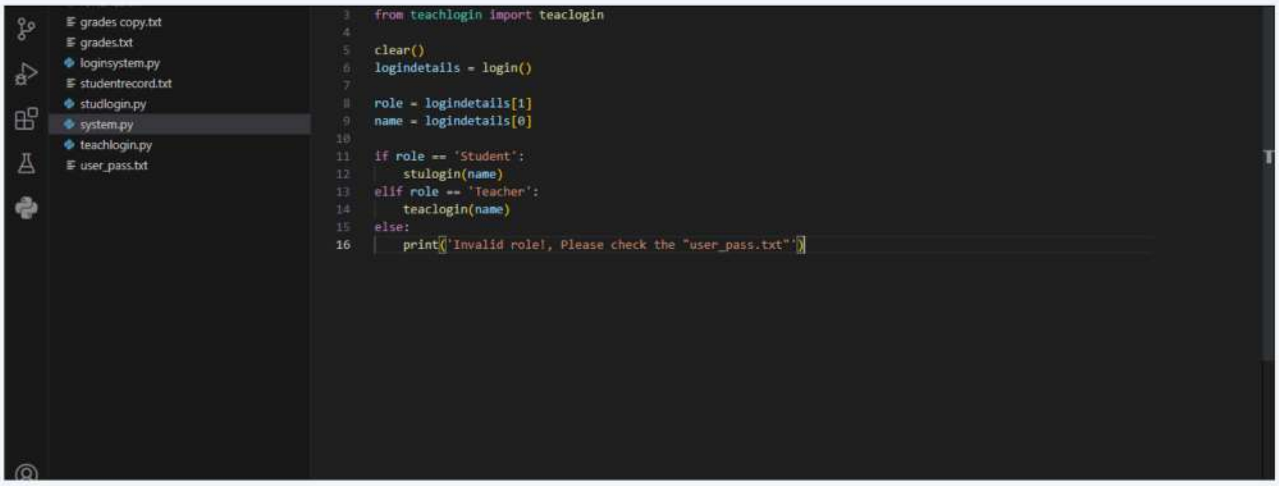
```
10. Group projects may be challenging due to conflicting schedules and differences in work styles.

1) View Your fortune
2) view Student Detail in the record
3) View Your record
4) view grade for students
0) Log out
Enter your option 3
```

PROJECT REVIEW

the Student Management System is a commendable product that successfully addresses the challenges of student record management and communication in educational institutions. With its user-friendly interface, secure login system, and valuable features, it offers significant benefits for both teachers and students.

The system's performance and usability contribute to its overall value, making it a worthwhile investment for institutions seeking to streamline administrative processes and enhance communication within their educational ecosystem.

The ChatGPT has given the program a Rating: 4.5/5



```python
from teachlogin import teaclogin

clear()
logindetails = login()

role = logindetails[1]
name = logindetails[0]

if role == 'Student':
    stulogin(name)
elif role == 'Teacher':
    teaclogin(name)
else:
    print('Invalid role!, Please check the "user_pass.txt"')
```

OPPORTUNITIES FOR IMPROVEMENT

Password Security: Implementing stronger password security measures is crucial. Currently, the system compares passwords directly as plaintext. Consider incorporating password hashing algorithms (e.g., bcrypt, Argon2) to securely store and verify passwords. This adds an extra layer of protection against potential data breaches.

The Student Management System provides significant value to educational institutions. By automating administrative tasks and centralizing student records, it simplifies data management processes and saves time for teachers and staff. The system promotes accurate record keeping and communication, improving overall efficiency and effectiveness in educational administration. The ability to customize the system based on institution-specific requirements adds further value, making it adaptable to varying needs.

While the Student Management System offers valuable features, there are areas for potential improvement. Integration with other educational systems, such as grading platforms or timetable management tools, could further streamline processes. Additionally, the inclusion of additional functionalities like attendance tracking or communication tools for students and parents would enhance the system's capabilities.

CONCLUSION

Despite these areas for improvement, the project has served as a valuable learning experience, providing insights into user authentication, file handling, and modular code organization. It has also laid the groundwork for further enhancements and customization based on specific needs and requirements.
Overall, the completion of this project showcases the ability to create a functional login system and highlights the potential for future growth and development. The knowledge gained through this project can be applied to various applications and systems that require user authentication and role-based access control.

"*We would like to extend my heartfelt thanks to all of my teachers who have played a significant role in guiding and supporting me throughout the development of this project. Your expertise, patience, and dedication have been instrumental in my learning and growth. I am truly grateful for your valuable insights, encouragement, and assistance. Your contributions have made a lasting impact on my journey, and I appreciate all that you have done to help me succeed. Thank you for being outstanding teachers and for your unwavering support.*"