1. User Class

A class represents a user in the system with basic attributes and methods for accessing these attributes.

```
class User:
    def __init__(self, name, user_id, username, password,
role):
        self.__name = name
        self.__user_id = user_id
        self.__username = username
        self.__password = password
        self.__role = role

    def get_username(self):
        return self.__username

    def get_password(self):
        return self.__password

    def get_name(self):
        return self.__name

    def get_user_id(self):
        return self.__user_id

    def get_role(self):
        return self.__role

    def __repr__(self):
        return f'User({self.__name}, {self.__user_id}, {self.__username}, {self.__role})'
```

Attributes:

- `name`: The name of the user.
- `user_id`: A unique identifier for the user.
- `username`: The username for logging in.
- `password`: The password for logging in.
- `role`: The role of the user, e.g., "student" or "doctor".

Methods:

- `get_username()`: Returns the username.
- `get_password()`: Returns the password.
- `get_name()`: Returns the name.
- `get_user_id()`: Returns the user ID.
- `get_role()`: Returns the role of the user.
- `__repr__()`: Returns a string representation of the user.

2. Student Class

This class represents a student, inheriting from the `User` class.

python
Copy code
```python
class Student(User):
    def __init__(self, name, user_id, username, password):
        super().__init__(name, user_id, username, password, "student")
        self.courses = []
        self.assignments = []
```

Attributes:

- `courses`: **A list of courses the student is enrolled in.**
- `assignments`: **A list of assignments the student has submitted.**

3. Doctor Class

This class represents a doctor, inheriting from the `User` class.

python
Copy code

```
class Doctor(User):
        def  __init__(self,  name,  user_id,  username,
password):
                super().__init__(name,  user_id,  username,
password, "doctor")
```

4. Course Class

This class represents a course with attributes like name, code, doctor, and assignments.

python
Copy code

```
class Course:
    def __init__(self, name, code, doctor, assignments):
        self.__name = name
        self.__code = code
        self.__doctor = doctor
        self.__assignments = assignments
        self.students = []

    def get_name(self):
        return self.__name

    def get_code(self):
        return self.__code

    def get_doctor(self):
        return self.__doctor

    def get_assignments(self):
        return self.__assignments

    def __repr__(self):
            return f'Course({self.__name}, {self.__code},
{self.__doctor})'
```

Attributes:

- `name`: **The name of the course.**
- `code`: **The course code.**
- `doctor`: **The doctor teaching the course.**
- `assignments`: **The number of assignments in the course.**
- `students`: **A list of students enrolled in the course.**

Methods:

- `get_name()`: **Returns the course name.**
- `get_code()`: **Returns the course code.**
- `get_doctor()`: **Returns the doctor's name.**

- `get_assignments()`: Returns the number of assignments.
- `__repr__()`: Returns a string representation of the course.

5. Dummy Data Creation

A function to create dummy data for testing the system.

python
Copy code
```
def create_dummy_data():
    # Creating doctor instances
    doctors = [
        Doctor("Ali", "d001", "d001", "d001"),
        Doctor("Mostafa", "d002", "d002", "d002"),
        # ... other doctors
    ]

    # Creating course instances
    courses = [
        Course("Prog 1", "CS111", "Samy", 3),
        Course("Prog 2", "CS112", "Morad", 3),
        # ... other courses
    ]

    # Creating student instances
    students = [
         Student("Hussien Samy", "00102345", "s00102345",
"s00102345"),
         Student("Ashraf Sayed", "00204690", "s00204690",
"s00204690"),
        # ... other students
    ]

    # Enrolling students in courses
    course_enrollments = {
         "00102345": ["CS111", "CS112", "CS333", "CS136",
"CS240", "CS350"],
        # ... other enrollments
    }

    for student in students:
                                         student_courses     =
course_enrollments.get(student.get_user_id(), [])
        for course_code in student_courses:
            for course in courses:
                if course.get_code() == course_code:
                    student.courses.append(course)
                    course.students.append(student)

    users = students + doctors
    return users, courses
```

6. User Login Function

A function to handle user login.

python
Copy code

```
def login(users):
    username = input("Enter username: ")
    password = input("Enter password: ")
    for user in users:
```

```
                if user.get_username() == username and
user.get_password() == password:
                print(f"Welcome, {user.get_name()}. You are
logged in.")
            return user
    print("Invalid username or password.")
    return None
```

7. Course Listing Function

A function to list all courses a student is enrolled in.

python
Copy code

```
def list_courses(user):
    if user.courses:
        print("My Courses list:")
             for idx, course in enumerate(user.courses,
start=1):
                print(f"{idx}) Course {course.get_name()} -
Code {course.get_code()}")
    else:
        print("You are not registered in any courses.")
```

8. View Course Details Function

A function to view details of a specific course.

python
Copy code

```
def view_course(user):
    list_courses(user)
```

```
     choice = int(input("Which course to view? [1 - {}]:
".format(len(user.courses))))
    if 1 <= choice <= len(user.courses):
        course = user.courses[choice - 1]
            print(f"Course {course.get_name()} with Code
{course.get_code()}      -      taught      by      Doctor
{course.get_doctor()}")
            print(f"Course has {course.get_assignments()}
assignments")
        for i in range(course.get_assignments()):
                        submitted = "submitted" if i <
len(user.assignments) else "NOT submitted"
              print(f"Assignment {i + 1} {submitted} NA /
{random.randint(20, 50)}")
    else:
        print("Invalid course selection.")
```

9. Submit Assignment Function

A function to submit an assignment for a course.

python
Copy code

```
def submit_assignment(course, user):
    view_course(user)
      assignment_number = int(input("Which assignment to
submit? [1 - {}]: ".format(course.get_assignments())))
                  if     1     <=     assignment_number     <=
course.get_assignments():
         solution = input("Enter the solution (no space):
")
```

```
                user.assignments.append((course.get_code(),
assignment_number, solution))
        print("Assignment submitted successfully!")
    else:
        print("Invalid assignment number.")
```

10. View Grades Report Function

A function to view the grades report of a student.

python
Copy code

```
def view_grades_report(user):
    print("Grades Report:")
    for course in user.courses:
        total_assignments = course.get_assignments()
        submitted_assignments = len([assignment for
assignment in user.assignments if assignment[0] ==
course.get_code()])
        print(f"Course code {course.get_code()}")
        print(f"Total submitted {submitted_assignments}
assignments")
                print(f"Grade {random.randint(0, 25)} /
{random.randint(50, 100)}")
```

11. Logged-In Menu Function

A function to display the menu after a user logs in.

python
Copy code

```
def logged_in_menu(user):
    while True:
        print("\nPlease make a choice:")
        print("1 - Register in Course")
        print("2 - List My Courses")
        print("3 - View Course")
        print("4 - Grades Report")
        print("5 - Log out")

        choice = input("Enter Choice: ")

        if choice == '1':
            pass  # Functionality to be added
        elif choice == '2':
            list_courses(user)
        elif choice == '3':
            view_course(user)
            while True:
                print("\nPlease make a choice:")
                print("1 - Unregister from Course")
                print("2 - Submit solution")
                print("3 - Back")
                choice = input("Enter Choice: ")
                if choice == '1':
                    pass  # Functionality to be added
                elif choice == '2':
                    submit_assignment(user.courses[0],
user)  # Fix this to select the correct course
                elif choice == '3':
                    break
        elif choice == '4':
            view_grades_report(user)
        elif choice == '5':
            print("Logged out successfully.")
            break
```

12. User Sign-Up Function

A function to sign up a new student.

python
Copy code

```
def sign_up(users):
    name = input("Enter your name: ")
    user_id = input("Enter your ID: ")
    username = input("Enter your username: ")
    password = input("Enter your password: ")
    role = "student"

    for user in users:
            if user.get_username() == username or
user.get_user_id() == user_id:
            print("Username or ID already exists. Please
try again.")
            return None

    new_user = Student(name, user_id, username, password)
    users.append(new_user)
    print("Sign-up successful! You can now log in.")
    return new_user
```

13. Main Function

The main function that starts the program.

python
Copy code
```
def main():
    users, courses = create_dummy_data()
    while True:
        print("\nPlease make a choice:")
        print("1 - Login")
        print("2 - Sign up")
        print("3 - Shutdown system")

        choice = input("Enter Choice: ")

        if choice == '1':
            user = login(users)
            if user:
                logged_in_menu(user)
        elif choice == '2':
            sign_up(users)
        elif choice == '3':
            print("System shutdown. Goodbye!")
            break

if __name__ == "__main__":
    main()
```