University_System Project.

Presented by:

Abdallah Ahmed - Jana Hassan - Mohammed Adel

Project Overview

A simple system that allows students to:

- Log in using their ID and name
- Register for courses based on level requirements
- View their grades

Key Features:

- Data validation
- Level-based course restrictions
- Modular design

Defining the Project

System Architecture

main.py

Entry point; handles interface and menu navigation.

register.py

Handles course registration logic and validation.

STUDENT.PY/ PROFESSR

Contains Student/Professer class with attributes and methods

Data.json (Data base)

Contains predefined student/Professr and course data.

Student Authentication

- Student must enter a valid ID and name.
- If credentials match, a Student object is created with:
- ID, Name, College, Level
- then it let the student to register to course , view grades and view attendance

```python
    def __init__(self, student_id, name, level, college, department):
        self.student_id = student_id
        self.name = name
        self.level = level
        self.college = college
        self.department = department
        self.courses = {}
        self.registration = Registration()
        self.attendance = Attendance()
        self.grading = Grading()

    def register_course(self, course):
        return self.registration.register_course(self, course)

    def unregister_course(self, course_id):
        return self.registration.unregister_course(self, course_id)
```

```python
    def get_student_info(self):
        return {
            'student_id': self.student_id,
            'name': self.name,
            'department': self.department,
            'college': self.college,
            'level': self.level,
            'courses': self.courses
        }

    def print_courses(self):
        if not self.courses:
            print(f"\nStudent {self.name} is not enrolled in any courses.")
            return

        print(f"\nCourses for {self.name} (ID: {self.student_id}):")
        print("-" * 80)
        print(f"{'Course ID':<10} {'Course Name':<30} {'Grade':<8}")
        print("-" * 80)
```

Course Registration

- register_course(course_id) funcation :
- Checks if course ID exists then Compares student level with course level.
- list all the available courses.
- Prevents duplicate course registration.
- Enforces a 7-course maximum.
- allow Admin to remove student or professer from a course

```python
def add_student(self, student_id, student_name, student_level, student_college):
    self.is_enrolled = False
    if student_id in self.students:
        print(f"Student {student_name} is already enrolled in this course.")
        return

    if student_level != self.level:
        print(f"Student level ({student_level}) does not match course level.")
        return

    if student_college != self.college:
        print(f"Student college does not match course college ({self.college}).")
        return

    self.students[student_id] = {
        'name': student_name,
        'grade': None,
        'attendance': []
    }
    self.is_enrolled = True
```

```python
def remove_student(self, student_id):
    self.is_enrolled = False
    if student_id not in self.students:
        print(f"Student with ID {student_id} is not enrolled in this course.")
        return

    self.students.pop(student_id)
    self.is_enrolled = True

#assign professor to course in Admin mode
def assign_professor(self, professor):
    self.professor_assigned = False
    for p in self.professors:
        if p.professor_id == professor.professor_id:
            print(f"Professor {professor.name} is already assigned to this course.")
            return

    self.professors.append(professor)
    self.professor_assigned = True
```

Data Storage

- Stores all students, courses, and professors information.
- Keeps grades organized under each student.
- Ensures data is persistent across sessions.
- Uses JSON format for readability and easy access.
- Supports future upgrades like web interfaces or database migration.

```python
class Student:  4 usages
    def __init__(self, student_id, name, level, college, de
        self.student_id = student_id
        self.name = name
        self.level = level
        self.college = college
        self.department = department
        self.courses = {}
        self.registration = Registration()
        self.attendance = Attendance()
        self.grading = Grading()
```

```python
class Course:  4 usages
    def __init__(self, course_id, name, department, level, college, credit_hours):
        self.course_id = course_id
        self.name = name
        self.department = department
        self.level = level
        self.college = college
        self.credit_hours = credit_hours
        self.professors = []
        self.students = {}
class Professor:  5 usages
    def __init__(self, professor_id, name, department, college):
        self.professor_id = professor_id
        self.name = name
        self.department = department
        self.college = college
        self.courses = {}
```

How It Works

START
↓
Student / Professor
↓ ↓
Student Login Professor Login
↓ ↓
Student Enters ID & Name Professor Enters ID & Name
↓
Check Validity (from data.json)
↓
If Valid → opens Menu
↓ ↓
[Student Menu] [Professer Menu]
1. Register Course → register_course() 1. Set Grades → update data.json
2. View Grades → view_grades() 2. Assign Attendance → mark in data.json
3. Exit
↓
END

Features Summary

1. **Simple CLI interface**
2. **Course registration with validation**
3. **Grade viewing for students**
4. **Clean modular design**
5. **Professor login and access**
6. **Set student grades**
7. **Assign attendance**

Thank You