**University Course Registration System - System Design**

**Objective:**

To build a University Course Registration System using Python that supports account creation, course enrollment, class capacity management, conflict detection, and CSV-based data storage.

**Core Features:**

1. **Create Student Accounts**
2. **Student Login**
3. **View Available Courses**
4. **Enroll in Courses** (with schedule conflict and capacity check)
5. **Drop Courses**
6. **Track Courses per Student & Students per Course**
7. **Store and retrieve data using CSV files**

**Class Definitions:**

**1. Student Class**

class Student:

def \_\_init\_\_(self, student\_id, name, password):

self.student\_id = student\_id

self.name = name

self.password = password

self.registered\_courses = [] # [ [CourseID, Days, Time, CreditHours] ]

**2. Course Class**

class Course:

def \_\_init\_\_(self, course\_id, name, instructor, schedule, max\_students=30):

self.course\_id = course\_id

self.name = name

self.instructor = instructor

self.schedule = schedule # List of [Days, Time, CreditHours]

self.enrolled\_students = [] # List of student\_ids

self.max\_students = max\_students

**CSV File Structures:**

| **File Name** | **Description** |
| --- | --- |
| students.csv | Stores student ID, full name, and password |
| courses.csv | Stores course ID, name, instructor, and schedule |
| enrollments.csv | Stores student ID and course ID mappings |
| class\_<course\_id>.csv | Stores list of student IDs registered in the course |

**Available Courses Representation:**

available\_courses = {

"CS120": {

"name": "Intro to Programming",

"instructor": "Prof. X",

"schedule": [

["MWF", "1000-1120", 3],

["TR", "0800-0930", 3]

]

},

# Add more courses as needed

}

**Note:** Each unique time offering should have its own Course ID (e.g., CS120-A, CS120-B).

**Utility Functions and Integration Workflow:**

To ensure smooth integration and clarity for developers, here’s how all components will work together:

1. **User Registration & Authentication**
   * add\_student\_account(): Adds a new student to students.csv.
   * login\_student(): Verifies credentials from students.csv.
2. **Viewing Courses**
   * load\_courses\_from\_csv(): Reads courses from courses.csv.
   * display\_courses(): Displays courses in a user-friendly format.
3. **Enrollment Process**
   * check\_schedule\_conflict(): Ensures the student doesn’t enroll in overlapping classes.
   * check\_credit\_limit(): Verifies students do not exceed max credit hours.
   * check\_course\_capacity(): Ensures the course hasn’t exceeded 30 students.
   * enroll\_student\_in\_course(): Finalizes enrollment by:
     + Adding course to the student’s registered\_courses list.
     + Writing enrollment details to enrollments.csv.
     + Updating class\_<course\_id>.csv to track enrolled students.
4. **Dropping a Course**
   * drop\_course(): Removes a student from a course by:
     + Removing course from registered\_courses.
     + Updating enrollments.csv.
     + Removing student from class\_<course\_id>.csv.
5. **Fetching Enrollment Data**
   * get\_enrolled\_students(): Reads class\_<course\_id>.csv to get the list of students.
6. **Overall System Flow:**
   * A student **registers an account**.
   * The student **logs in**.
   * The student **views available courses**.
   * If they wish to enroll, the system **checks for conflicts, credit limits, and course capacity**.
   * If all checks pass, the student is **enrolled and records are updated**.
   * If a student **drops a course**, the records are adjusted accordingly.
   * Admin or students can **view enrollments** at any time.