EE202: Object-Oriented Computer Programming (Fall-2025) -Term Project ECE Department Course Registration System

Project Overview

The **ECE Department Course Registration System** is a software solution designed to streamline the course registration process for students within the ECE department. The system enables students to plan and register for their courses each semester by automatically validating their choices against a comprehensive set of academic and logistical rules. It checks for prerequisite completion, adherence to the college's course plan for specific programs (Computer, Communications, Power, Biomedical), course availability, classroom/lab capacity, and faculty scheduling. The system aims to prevent scheduling conflicts, ensure students meet graduation requirements on time, and optimize resource allocation.

Key objectives include:

- Facilitating an efficient, error-free course registration experience for students.
- Automating the validation of student course selections based on **program plans**, **prerequisites**, **course caps**, and **resource availability**.
- Ensuring data integrity, scalability, and usability through robust database design and an intuitive graphical interface.

Technical Specifications

The system shall adhere to the following technical standards:

- **Programming Language:** Python 3.x (latest stable release).
- **GUI Framework:** PyQt5 or PyQt6 for interface implementation.
- **Database:** SQLite3 for persistent data storage.
- **Design Paradigm:** Object-Oriented Programming (OOP) with modular, reusable classes.

Functional Requirements

1. Course & Curriculum Management

1.1 Core Functionality

- **Primary Purpose:** Enable administrators to define and manage the department's course offerings, program plans, and prerequisites.
- **Input Sources:**
  - Course data: Course code, name, credits, lecture/lab hours, prerequisites (list of course codes), maximum capacity.
  - Program plan: Mapping of courses to specific programs (Computer, Comm, Power, Biomedical) and levels (e.g., Level 3, Semester 1).
- **Output Formats:**
  - Validated entries stored in courses, program_plans, and prerequisites tables in SQLite.
- **Data Processing:**
  - Validate credit hours > 0; enforce unique course codes; check that prerequisite courses exist in the system.

1.2 Input/Output Handling

- **Validation:**
  - All fields mandatory; credits must be positive; course code must be unique.
- **Output Standards:**
  - Database entries include all course attributes and structured program-plan relationships.

1.3 Error Handling

- **Exceptions:**
  - Duplicate course code, invalid prerequisite, negative credits.
- **User Feedback:**
  - Display errors like "Course code 'COE310' already exists" or "Prerequisite 'COE200' is not a valid course."

1.4 Additional Features

- **Bulk Import:** Load course and program plan data from CSV/Excel files.

---

2. Student Profile & Academic History Management

2.1 Core Functionality

- **Primary Purpose:** Allow students to register and maintain their profiles, including their completed courses and current program.
- **Input Sources:**
  - Student-provided fields: Name, Student ID (unique), Email, Program (dropdown: Computer, Comm, Power, Biomedical), Current Level.
  - System-managed: Transcript (list of completed courses with grades).
- **Output Formats:**
  - Validated entries stored in students and transcripts tables.
- **Data Processing:**
  - Validate unique Student ID; ensure program is from the defined list.

2.2 Input/Output Handling

- **Validation:**
  - Reject duplicate Student IDs; enforce valid program selection.
- **Output Standards:**
  - Database entries include student_id, program, level, and linked transcript.

2.3 Error Handling

- **Exceptions:**
  - Duplicate Student ID, invalid program.
- **User Feedback:**
  - Display "Student ID already exists" or "Please select a valid program."

2.4 Additional Features

- **Transcript View:** Allow students to view their academic history.

---

3. Registration Validation & Timetable Builder

3.1 Core Functionality

- **Primary Purpose:** The core module that allows students to select courses and automatically validates their schedule against all constraints.
- **Input Sources:**
  - Student's selected courses from a list of available offerings.
  - Student's profile (program, level, transcript).
  - Course data (prerequisites, capacities, schedule).
- **Output Formats:**
  - A visual timetable (weekly schedule view).
  - A list of validation messages (errors/warnings).
  - Finalized registration stored in registrations table upon successful validation.
- **Data Processing:**
  1. Check total credit hours are within min/max limits (e.g., 12 to 18 credits).
  2. Verify all selected course prerequisites are met (exist in student's transcript).
  3. Ensure courses are part of the student's program plan for their level.
  4. Check for schedule conflicts (time overlaps between lectures/labs).
  5. Verify course capacity has not been exceeded.

3.2 Input/Output Handling

- **Validation:**
  - Prevent registration if any hard constraints are violated (prerequisites, conflicts).
- **Output Standards:**
  - Timetable displayed in a clear, color-coded weekly calendar format in the GUI.
  - Real-time validation feedback as courses are added/removed.

3.3 Error Handling

- **Exceptions:**
  - Prerequisite not met, schedule conflict, course full, credit limit exceeded.
- **User Feedback:**
  - Display specific, actionable messages like "Cannot register for COE310: Prerequisite COE200 not completed" or "Schedule conflict: COE310 Lab overlaps with COE320 Lecture."

3.4 Additional Features

- **Waitlist System:** Automatically add students to a waitlist for full courses and notify them if a spot opens.

---

4. User Authentication & Role-Based Access Control

4.1 Core Functionality

- **Primary Purpose:** Secure system access through role-based permissions (Student, Admin).
- **Input Sources:**
  - User credentials (Student ID/Email, password) via GUI forms.
- **Output Formats:**
  - Session tokens; access logs in SQLite.
- **Data Processing:**
  - Encrypt passwords with **bcrypt**; assign roles.

4.2 Input/Output Handling

- **Validation:**
  - Enforce strong passwords.
- **Output Standards:**
  - Redirect students to the registration dashboard and admins to the course management dashboard post-login.

4.3 Error Handling

- **Exceptions:**
  - Invalid credentials, duplicate email/ID during sign-up.
- **User Feedback:**
  - Display "Invalid Student ID or password."

4.4 Additional Features

- **Password Recovery:** Email-based reset using **smtplib**.

Additional Bonus Functional Requirements

Your team can select two or more from the following functional requirements to implement for additional bonus marks.

5. Reporting & Analytics Dashboard

5.1 Core Functionality

- **Primary Purpose:** Provide administrators with insights into course demand, registration statistics, and resource utilization.
- **Input Sources:**
    - Aggregated data from registrations, courses, and students' tables.
- **Output Formats:**
    - Interactive dashboard with charts (e.g., course enrollment bar charts, program-wise registration heatmaps).
    - Exportable reports (PDF/CSV).
- **Data Processing:**
    - Calculate metrics like course fill rate, average load per student, etc., via SQL queries.
    - Generate visualizations using **matplotlib** or **Plotly**.

5.2 Justification

"Data-driven dashboards empower the department to make informed decisions about course offerings and resource allocation."

---

6. Faculty Module & Availability Management

6.1 Core Functionality

- **Primary Purpose:** Allow faculty to indicate their course preferences and availability, and admins to assign courses to faculty.
- **Input Sources:**
    - Faculty profiles: Name, ID, preferred courses, time availability.
    - Admin assignments: Assigning specific courses to faculty members.
- **Output Formats:**
    - Faculty assignments stored in the database.

    - Visual schedule for each faculty member.
- **Data Processing:**
    - Check for faculty scheduling conflicts during assignment.

6.2 Justification

"Streamlines the teaching assignment process and ensures faculty are assigned to courses they are qualified and available to teach."

---

7. Advanced Conflict Checking & "What-If" Scenarios

7.1 Core Functionality

- **Primary Purpose:** Enhance the validation system to check for softer conflicts (e.g., back-to-back classes on opposite sides of campus) and allow students to simulate "what-if" scenarios for different programs or levels.
- **Input Sources:**
    - Student's simulated program/level change.
    - Classroom locations (for proximity checks).
- **Output Formats:**
    - Warnings for potential soft conflicts.
    - A simulated timetable and validation report for the "what-if" scenario.

7.2 Justification

"Provides a more robust and student-centric planning tool, mirroring features in modern university ERP systems."

Data Management:

Implement a robust data management system that securely stores student information, course catalogs, program plans, and registration records using an SQLite database. This approach will effectively manage relational data within the application, ensuring referential integrity and efficient querying.

Error Handling

a. The system should handle invalid inputs gracefully (e.g., non-numeric data in credit fields, malformed course codes).

b. Appropriate, informative error messages should be displayed to the user to guide corrective action.

Class Structure

The Application should be structured into different classes, for example:

- **MainApp (PyQt QApplication):**
    - The entry point of the application, responsible for initializing the main window and starting the event loop.
- **MainWindow (PyQt QMainWindow):**
    - Manages the overall graphical interface, including login screen and role-specific dashboards.
- **LoginDialog (PyQt QDialog):**
    - Handles user authentication.
- **StudentDashboard / AdminDashboard (PyQt QWidget):**
    - Role-specific central interfaces for the application's functionalities.
- **Course Class:**
    - Attributes: course_code, name, credits, lecture_hours, lab_hours, max_capacity, prerequisites (list).
    - Methods: check_prerequisites(student_transcript), is_full()
- **Student Class:**
    - Attributes: student_id, name, email, program, level, transcript (list of completed courses).
    - Methods: get_completed_credits(), add_to_transcript(course)
- **RegistrationSystem Class:**
    - Attributes: SQLite database connection and cursor.
    - Methods: validate_schedule(student, selected_courses), register_student(student, course_list), add_course(course), get_available_courses(program, level)

Documentation:

You need to submit a program written in Python. The code should be well-documented with comments. In addition, you need to write a detailed report explaining your design choices and decisions. The report should show your hidden efforts in producing a good GUI Application. The report should be word-processed according to the presentation of the technical work checklist (PTWC).

Each team member must contribute significantly to the programming. You should discuss each concept and implementation in meetings and combine your efforts optimally.

The report must provide detailed documentation covering the design, implementation, and testing of the ECE Course Registration System. Include explanations of key features, code snippets, screenshots of the system, and any challenges encountered during development. A user manual must be provided to explain how to use the system, including screenshots of the GUI. Moreover, the report should include a document describing the test cases used to validate the system.

The report should also include a table that certifies whether a team member was able to complete their programming tasks on their own (Yes/NO) and summarizes the work distribution.

Table-1: Example of the work distribution table

Tasks / Members	M1	M2	M3	M4	M5
Database & Core Classes	25%	15%	10%	30%	20%
GUI Design (PyQt)	10%	30%	25%	15%	20%
Registration Logic	20%	20%	30%	20%	10%
User Authentication	15%	15%	20%	10%	40%
Bonus Feature #1	30%	20%	20%	15%	15%
Testing & Documentation	20%	20%	15%	25%	20%
Total	**20%**	**20%**	**20%**	**19%**	**21%**

Presentation:

Finally, you need to prepare a short PowerPoint presentation (10-15 minutes) to summarize your work. During the presentation, you must demonstrate your application with a live demo. Highlight the application's features, functionality, user interface, and discuss potential future enhancements.

Project Rubric

Application Functionality and User Experience (12)		Code Quality and Documentation (8)		Presentation (5)	
Criteria	**Points**	**Criteria**	**Points**	**Criteria**	**Points**
Course & Student Classes & DB Setup	2	No runtime exceptions	2	Presentation Skills & Demo	5
Registration Validation Logic	4	Clean Code & Clear Comments	2		
GUI Design & Usability	4	Meaningful Variable Names	2		
Error Handling	2	Comprehensive Documentation & Test Cases	2		
Total	**12**	**Total**	**8**	**Total**	**5**

Submission

All deliverables (Source Code, Project Report, User Manual, Test Cases) should be submitted electronically by **Saturday, 06 December 2025**. Project demonstrations and discussions will commence on **Sunday, 07 December 2025**.