CS236 – LAB ASSIGNMENT #5

BANNER WEB – STUDENT COURSE REGISTRATION SYSTEM

Project Documentation: Banner Web

Group Members:

- Chinyemba
- Lleyton
- Gideon
- Jonathan

Project Overview

This documentation outlines the design, functionality, and development process of Banner Web, a desktop course registration system created using Python and the Tkinter library. The project simulates a university student portal where authenticated students can browse available courses, enroll or drop classes, view their current schedule, and print a formatted weekly timetable. The application is architected using Object-Oriented Programming (OOP) principles, with responsibilities cleanly divided into distinct classes for authentication, domain models, business logic, and user interfaces.

The application is a graphical user interface (GUI) divided into two primary sections:

1. **My Courses Tab:** Displays the student's currently enrolled classes as modern cards, with details such as instructor, schedule, and location; allows dropping a course and printing the schedule.
2. **Browse Courses Tab:** Lists all available courses with capacity and enrollment status; supports one-click enrollment with validation (capacity and schedule conflicts).

Team Contributions & Features Implemented

The project was developed collaboratively, with each team member owning core components of the system.

Chinyemba: Core Enrollment Logic & Course Model

- **Course Class:** Implemented the course representation, including capacity tracking using a Set for O(1) lookups, schedule fields (days, time, location), and utilities like conflict detection and available seat calculations.
- **Enrollment System Class:** Built the business logic for registering students, adding courses, enrolling and dropping students, CSV persistence (courses, students, enrollments), and generating a printable weekly schedule view.

Lleyton: Main Student GUI

- **Registration System GUI Class:** Created the modern, card-based Tkinter interface with two tabs (My Courses, Browse Courses), polished visual design, status badges, and interactive actions (Enroll, Drop, Print Schedule) with automatic UI refresh.
- **UI Design:** Added UI polish: color scheme, badges, limits (e.g., 6-course cap), live refresh after actions.

Gideon: Student Model & Application Orchestration

- **Student Class:** Implemented the student representation with a Set-backed collection for registered courses (O(1) membership and deduplication) and supportive utilities.
- **Application Entry (main.py):** Wired the login flow to the main GUI, set up window lifecycle, and provided demo credentials for quick start.

Jonathan: Authentication & Login/Registration UI

- **Authentication System & User Model:** Implemented secure credential management using SHA-256 hashing, role handling, unique Student ID validation, and CSV-based persistence.
- **Login Window:** Built the unified login/registration interface with form validation and smooth toggling between modes, assigning the Student ID to the authenticated user on first login.

Challenges Faced

- **State Synchronization in the GUI:** Ensuring the two tabs (My Courses, Browse Courses) always reflected the latest enrollment state required carefully placed refresh calls and clear separation between data and presentation.

- **CSV Data Persistence:** Reliable read/write of multiple CSVs (courses, students, enrollments, users) demanded strict header management, encoding handling, and consistency between in-memory Sets and serialized formats.
- **Authentication Edge Cases:** Enforcing unique Student IDs across accounts and handling first-time Student ID assignment after login required additional validation and persistence updates.
- **Schedule Conflict Detection:** Implementing practical conflict checks (shared days and overlapping time ranges) necessitated robust parsing and guardrails for missing schedule fields.
- **Capacity & Limits:** Coordinating course capacity checks with per-student enrollment limits (e.g., max of 6 courses) and surfacing user-friendly feedback in the GUI.

Conclusion

The Banner Web project successfully delivers a functional, user-friendly desktop application for student course registration. It demonstrates strong application of CS 236 concepts, including OOP design, efficient data structures (Dictionaries and Sets for O(1) operations), file-based data persistence with CSV, and interactive GUI development in Tkinter. The modular architecture—spanning authentication, business logic, and polished UI—supports maintainability and future enhancements such as admin tools, richer scheduling, and improved conflict detection.