University Course Management & AI Assistant System

CS5200: Database Theory and Applications – Final Project Report

CRN / Group: 11475/ Group 4

Student Name:

700792797 – Swaathi Shelvapulle

700774061 – Junaid Khan

700794792 – Prabhath Bellamkonda

700787150 – Puppala Charan Sai

1. Project Description

This project implements a university course management system with an integrated AI assistant for database-aware question answering. The application supports three primary roles—Admin, Teacher, and Student—and provides a full workflow for user registration and approval, course and section management, and student enrollment and drop operations. The system also exposes a natural-language interface, powered by a Large Language Model (Google Gemini), that allows users to ask questions about the university data without manually writing SQL queries.

The backend is implemented using FastAPI and exposes RESTful endpoints for authentication, role-based operations, and AI-assisted querying. A Streamlit-based frontend consumes these APIs to provide role-specific dashboards and forms. All persistent data is stored in a relational database (SQLite), which follows a normalized schema based on a standard university database design.

2. Database Description

The database is implemented in SQLite and stored as a single file (university.db). The schema models the core entities and relationships of a university environment, with an emphasis on referential integrity and support for typical academic operations. The design is normalized to approximately Third Normal Form (3NF), minimizing redundancy while maintaining query performance for transactional workloads.

Key design goals of the database include:

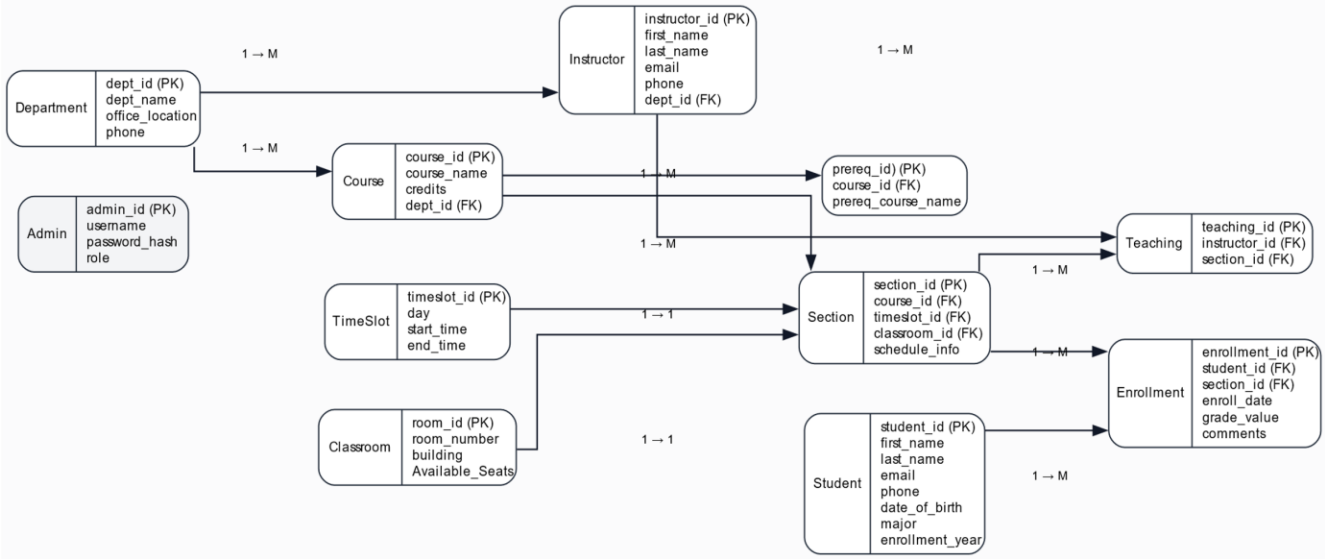
- - Support course catalog management across multiple departments.
- - Represent instructors, students, and their departmental affiliations.
- - Model course offerings by semester and year as sections, with room, capacity, and time slot information.

- - Track which instructor teaches which section.
- - Track which student takes which section and store grades when assigned.
- - Enforce business rules such as section capacity, credit limits, and schedule conflicts through a combination of constraints and application logic.
- - Provide a secure, role-aware foundation for AI-generated SQL queries.

3. Database Diagram (ERD Description)

The Entity–Relationship Diagram (ERD) for this project can be summarized as follows:

- - DEPARTMENT(dept_name, building, budget) is a central entity that represents academic departments.
- - COURSE(course_id, title, dept_name, credits) belongs to exactly one DEPARTMENT via dept_name (1:N relationship).
- - INSTRUCTOR(ID, name, dept_name, salary) works in one DEPARTMENT; a department can have many instructors (1:N).
- - STUDENT(ID, name, dept_name, tot_cred) majors in one DEPARTMENT; a department can have many students (1:N).
- - SECTION(course_id, sec_id, semester, year, building, room_number, time_slot_id, capacity) represents an offering of a course in a specific term.
- - TEACHES(ID, course_id, sec_id, semester, year) links INSTRUCTOR to SECTION (M:N resolved via this associative entity).
- - TAKES(ID, course_id, sec_id, semester, year, grade) links STUDENT to SECTION (M:N resolved via this associative entity).
- - TIME_SLOT(time_slot_id, day, start_hr, start_min, end_hr, end_min) defines meeting times; SECTION references TIME_SLOT.
- - PREREQ(course_id, prereq_id) defines prerequisite relationships between courses (self-referencing COURSE).
- - CLASSROOM(building, room_number, capacity) stores physical classrooms referenced by SECTION.
- - ADVISOR(s_ID, i_ID) connects STUDENT to INSTRUCTOR in an advising relationship.
- - LOGIN_CREDENTIALS(username, password, role, approved) provides authentication and authorization metadata for all users.



In a graphical ERD, these entities are shown as rectangles with primary keys underlined. Foreign key relationships are depicted using crow's foot notation to indicate cardinality (1:N between department and course, M:N between student and section via takes, etc.).

4. Data Dictionary

The following tables summarize the main entities, their attributes, and data types. (PK = Primary Key, FK = Foreign Key).

DEPARTMENT

Field Name	Data Type	Key / Constraint	Description
dept_name	VARCHAR(20)	PK	Unique name of the department.
building	VARCHAR(15)		Building where the department is located.
budget	NUMERIC(12,2)		Annual budget allocated to the department.

COURSE

Field Name	Data Type	Key / Constraint	Description
course_id	VARCHAR(8)	PK	Unique course identifier.
title	VARCHAR(50)		Course title.
dept_name	VARCHAR(20)	FK → DEPARTMENT.dept_name	Owning department.
credits	INTEGER	CHECK (credits > 0)	Number of credits earned by completing the course.

STUDENT

Field Name	Data Type	Key / Constraint	Description
ID	VARCHAR(5)	PK	Unique student identifier.
name	VARCHAR(20)		Student full name.
dept_name	VARCHAR(20)	FK → DEPARTMENT.dept_name	Student's major department.
tot_cred	INTEGER		Total credits earned by the student.

INSTRUCTOR

Field Name	Data Type	Key / Constraint	Description
ID	VARCHAR(5)	PK	Unique instructor identifier.
name	VARCHAR(20)		Instructor full name.
dept_name	VARCHAR(20)	FK → DEPARTMENT.dept_name	Instructor's home department.

salary	NUMERIC(8,2)		Annual salary of the instructor.

SECTION

Field Name	Data Type	Key / Constraint	Description
course_id	VARCHAR(8)	PK, FK → COURSE.course_id	Course being offered.
sec_id	VARCHAR(8)	PK	Section number for this course.
semester	VARCHAR(6)	PK	Semester (e.g., 'Fall').
year	NUMERIC(4,0)	PK	Year (e.g., 2025).
building	VARCHAR(15)		Building where the class meets.
room_number	VARCHAR(7)		Room number for the class.
time_slot	VARCHAR(4)	FK → TIME_SLOT.time_slot	Time slot when the section meets.
capacity	INTEGER		Maximum number of students allowed to enroll.

TAKES

Field Name	Data Type	Key / Constraint	Description
ID	VARCHAR(5)	PK, FK → STUDENT.ID	Student identifier.
course_id	VARCHAR(8)	PK, FK → SECTION.course_id	Course identifier.
sec_id	VARCHAR(8)	PK, FK → SECTION.sec_id	Section identifier.
semester	VARCHAR(6)	PK, FK → SECTION.semester	Semester.

year	NUMERIC(4,0)	PK, FK SECTION.year	Year.
grade	VARCHAR(2)		Assigned grade, if available.

LOGIN_CREDENTIALS

Field Name	Data Type	Key / Constraint	Description
username	VARCHAR(50)	PK	Login name for the user.
password	VARCHAR(255)		Password (plaintext in current prototype; should be hashed in production).
role	VARCHAR(10)	CHECK (role IN ('admin','student','teacher'))	Role of the user in the system.
approved	INTEGER	CHECK (approved IN (0,1))	Approval status; 1 = active account.

5. Sample Data

The database is initialized with several departments, courses, instructors, students, and sections to allow immediate testing. A small subset of illustrative sample data is shown below.

Sample STUDENT rows

Field Name	Data Type	Description
S001	ID	Comp. Sci. major, 9 credits
S002	ID	Finance major, 12 credits
S003	ID	History major, 6 credits

Sample COURSE rows

Field Name	Data Type	Description
CS-101	course_id	Intro to Computer Science, 3 credits
CS-201	course_id	Data Structures, 3 credits
FIN-101	course_id	Principles of Finance, 3 credits

Actual initialization scripts (setup.py) insert consistent and relationally valid data so that enrollments, schedule views, and AI queries can be demonstrated without manual seeding.

6. User Interfaces and Forms

The frontend is implemented in Streamlit and provides separate navigation paths for each role:

- - Login / Register: Authentication screen where users can log in or register as students or teachers. Admin accounts are seeded.
- - Admin Dashboard: Tabs for summary statistics, approvals, course and section management, student and instructor management, user listings, AI assistant, and password update.
- - Student Dashboard: Views for personal profile, enrolled courses and weekly schedule, enrollment and drop forms, AI assistant for database queries, and password update.
- - Teacher Dashboard: Summary of sections taught, student counts, AI assistant for extracting insights from teaching data, and password update.

Each logical operation (e.g., adding a course, approving a user, enrolling in a section) is backed by a FastAPI endpoint and triggered through a simple HTML-like form rendered by Streamlit.

7. Source Code and Architecture

The source code is organized into three main components:
- main.py: FastAPI backend with endpoints for authentication, admin operations, enrollment logic, and the AI query interface.
- frontend.py: Streamlit application implementing the UI and calling backend APIs using the requests library.
- setup.py: Database creation and sample data loading script.

A simplified example of a FastAPI endpoint for login is shown below:

```
from fastapi import FastAPI, HTTPException
import sqlite3

app = FastAPI()

@app.post('/login')
def login(username: str, password: str):
  conn = sqlite3.connect('university.db')
  cur = conn.cursor()
  cur.execute('SELECT password, role, approved FROM
login_credentials WHERE username = ?', (username,))
  row = cur.fetchone()
  conn.close()
  if not row or row[0] != password:
    raise HTTPException(status_code=401, detail='Invalid username
or password')
  if row[2] != 1:
    raise HTTPException(status_code=403, detail='User not
approved')
  return {'status': 'success', 'role': row[1]}
```

---

8. SQL Commands and Examples

This section shows representative SQL commands used in the project, including data definition (DDL) and data manipulation (DML).

```
-- Example: Create COURSE table
CREATE TABLE IF NOT EXISTS course (
  course_id   VARCHAR(8) PRIMARY KEY,
  title      VARCHAR(50) NOT NULL,
  dept_name   VARCHAR(20) NOT NULL,
  credits     INTEGER NOT NULL,
  FOREIGN KEY (dept_name) REFERENCES department (dept_name)
);

-- Example: Enroll a student in a section
INSERT INTO takes (ID, course_id, sec_id, semester, year)
VALUES ('S001', 'CS-101', '1', 'Fall', 2025);

-- Example: Query all courses a student is taking
SELECT s.ID, s.name, c.course_id, c.title, t.semester, t.year
FROM student s
JOIN takes t ON s.ID = t.ID
```

```
JOIN course c ON t.course_id = c.course_id
WHERE s.ID = 'S001';
```

9. LLM / AI Component

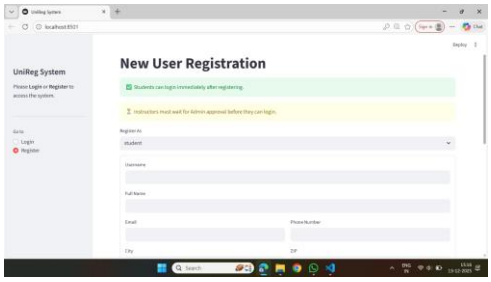
The AI component leverages Google Gemini to translate natural language questions into safe SQL queries. The backend feeds a sanitized schema description and the user's role into the model and instructs it to generate only SELECT statements. All generated SQL is post-validated in code to ensure that it does not contain any destructive commands such as INSERT, UPDATE, DELETE, DROP, or ALTER and that it does not access sensitive tables such as LOGIN_CREDENTIALS.

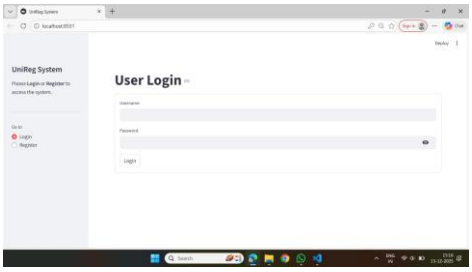
By combining the relational database with an LLM, the system allows non-technical users to explore complex data while ensuring that data integrity and security are preserved.

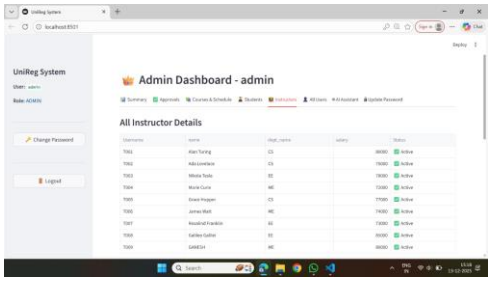
10. Results and Discussion

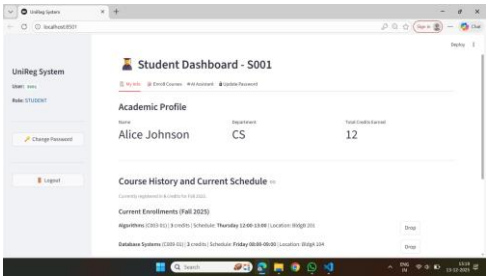
The implemented system successfully demonstrates how a well-designed relational database can support a realistic university course management workflow. The normalized schema and enforced constraints ensure that the data remains consistent under typical operations such as enrollment and schedule modifications.

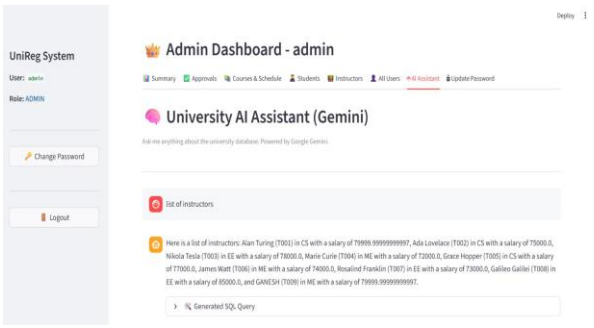
In addition, the AI assistant significantly improves usability. Users can ask questions like "Which courses am I taking this semester?" or "How many students are enrolled in CS-101?" and receive answers without writing SQL. The combination of schema-aware prompts and strong server-side validation ensures that the integration of AI does not compromise security or integrity.

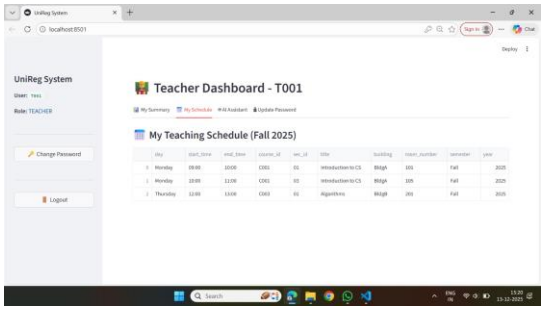












11. Future Work

- - Migrate from SQLite to PostgreSQL or MySQL for concurrent, multi-user deployments.
- - Implement password hashing and full token-based authentication (e.g., JWT) for production-grade security.
- - Extend the ERD and schema to cover additional university processes such as degree audits, waitlists, and billing.
- - Introduce analytical queries and dashboards (e.g., course popularity trends, GPA distributions) on top of the existing schema.
- - Enhance the AI assistant to support parameterized templates and user-friendly error recovery when queries fail.