🎓 E-Learning Platform (LearnVerse) Mini-Project

This is a full-stack mini-project developed for a Database Management System (DBMS) course, demonstrating the core principles of relational database design, API development, and client-side interaction.

The application simulates a simplified online course platform where users can browse courses, sign in (mock authentication), and enroll in (purchase) courses via a transactional API endpoint.

✨ Features

- **DBMS Focused:** Implements a relational database schema for **Users, Courses, Lessons, and Enrollments**.
- **Mock Authentication:** Simple sign-in/sign-out process to simulate a logged-in student (MOCK_USER_ID=4).
- **Course Browsing:** Fetches course details and instructor names using **SQL JOIN queries**.
- **Enrollment Logic:** Dynamically checks a student's enrollment status and controls the **"Buy/View Course"** button state.
- **Transactional API:** A secure API endpoint (/api/enroll) handles the transaction of adding a record to the Enrollments table.
- **Modern UI:** Responsive, dark-themed interface built using HTML, JavaScript, and Tailwind CSS.
- **Payment Simulation:** A modal simulates a payment gateway before committing the enrollment transaction to the database.

🛠️ Technology Stack

Layer	**Technology**	**Purpose**
Data Tier (DBMS)	**SQLite**	Lightweight, file-based SQL database for persistent storage.
Application Tier	**Python 3.x**	Core language for the backend logic.
Web Framework	**Flask**	Micro-framework used to create the RESTful API

		server.
Client-side	**HTML5, JavaScript**	Structure and client-side logic (fetching data, UI updates).
Styling	**Tailwind CSS**	Utility-first CSS framework for a responsive, dark-themed aesthetic.

💾 Database Schema (DBMS Design)

The project relies on four interconnected tables to manage the relationships (Entity-Relationship Model).

Table Name	**Primary Key**	**Foreign Key Relationship**	**Description**
Users	id	-	Stores user profiles (Instructors and Students).
Courses	id	instructor_id **(FK to Users)**	Stores course metadata (title, description).
Lessons	id	course_id **(FK to Courses)**	Stores the content belonging to a specific course.
Enrollments	Composite (user_id, course_id)	**FK to Users & Courses**	**Crucial M:M table** that tracks which student is enrolled in which course.

🚀 Setup and Run Instructions

Follow these steps to get the project running on your local machine.

1. Prerequisites

You must have **Python 3.7+** installed.

2. Project Setup

1. **Save Files:** Place app.py and index.html in the same folder.
2. **Install Dependencies:** Open your terminal or command prompt in the project folder and run:
   ```
   pip install Flask Flask-CORS
   ```

3. Start the Backend Server

The Python file must run constantly to serve data.

1. In your terminal, start the Flask application:
   ```
   python app.py
   ```
2. You will see output indicating the server is running on http://127.0.0.1:5000/. **KEEP THIS TERMINAL WINDOW OPEN.**

4. Access the Frontend

1. Navigate to your project folder.
2. **Double-click the index.html file.**
3. The web application will open in your default browser and automatically connect to the running Python server to fetch course data.

Testing Enrollment

1. **Sign In:** Use the Sign In button (Mock Student ID: 4).
2. **Enroll:** Click a course card, then click **"Buy/Enroll Now"**.
3. **Payment:** Complete the Mock Payment modal.
4. The button will change to **"View Course"**, confirming a record was successfully inserted into the Enrollments table in the database.