RPS™
learn..evolve

WIPRO NGA Program – Python

Capstone Project Presentation –4th and 5th Sept

Project Title Here - Student Management System

Presented by - Ansh Pandey

Introduction (Streamlining Student Data Management)

- ❖ The Student Management System is a lightweight, web-based application designed to efficiently organize and manage student records. Developed with Flask, Python's micro-framework, it offers a practical solution for educational institutions or individual educators to maintain student information with ease.

- ❖ This system addresses common challenges associated with manual or disparate data management methods. It provides a centralized, user-friendly interface for critical operations, ensuring data integrity and accessibility.

Features:

- ➢ **Centralized Data -** Consolidates student information into a single, accessible database.

- ➢ **Efficient Operations -** Enables seamless creation, reading, updating, and deletion (CRUD) of records.

- ➢ **User-Friendly -** Provides an intuitive web interface for easy navigation and data entry.

Technology Overview

- **Flask Framework**
  A lightweight and flexible Python web framework, ideal for building small to medium-sized applications. Its minimalist approach allows developers to choose their own tools and libraries.

- **SQLite Database**
  A self-contained, serverless, zero-configuration, transactional SQL database engine. Perfect for simple, local data storage without the need for a separate database server.

- **HTML & CSS**
  Standard technologies for structuring content and styling the web application, ensuring a clean and responsive user interface across different devices.

- **Jinja2 Templating**
  A powerful and modern templating engine for Python, used to dynamically generate HTML pages by injecting data from the Flask application into predefined templates.

Key Features (Comprehensive Student Management)

The system offers a robust set of features designed to cover essential student data management needs.

- **Add New Student Records -** A dedicated form allows for easy input of new student details, including name, email, and other relevant information.
- **View All Students -** Access a comprehensive list of all registered students, displayed in an organized and sortable table format.
- **Search & Filter Students -** Quickly locate specific student records using search functionality by name or email, enhancing data retrieval efficiency.
- **Update & Delete Records -** Maintain data accuracy by easily modifying existing student information or removing outdated entries.
- **Informational Pages -** Includes standard "Contact" and "About Us" pages for system information and user interaction.

System Architecture (User to Database Workflow)

The system follows a clear, modular architecture, similar to the Model-View-Controller (MVC) pattern, to ensure scalability and maintainability.

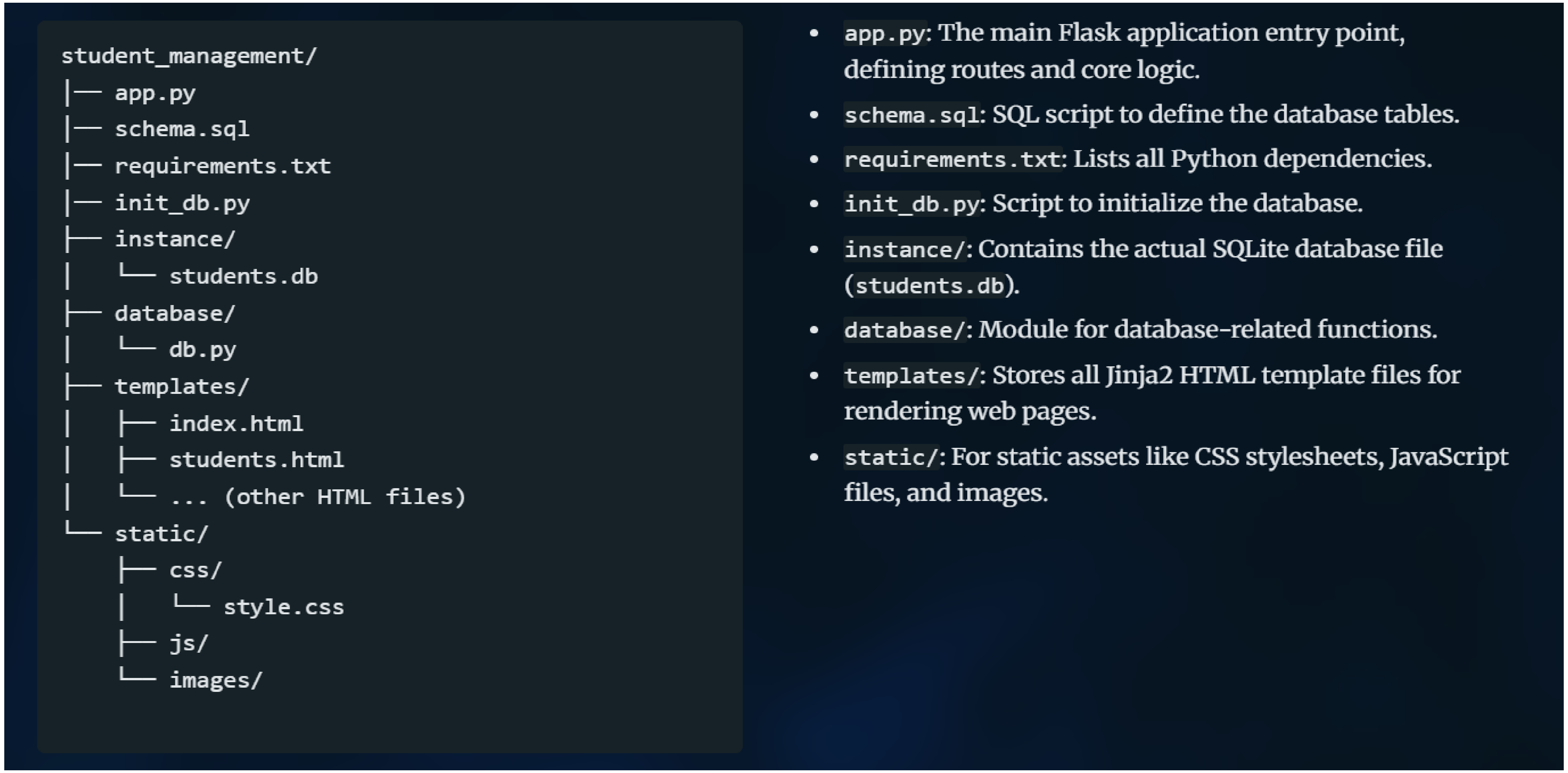
- **Routes & Logic -** Defined in the Flask application, these handle incoming user requests and execute the necessary business logic. In Flask, **routes** are URLs that the user visits (like /students, /about, /contact). Each route has a **function (view function)** in app.py that decides **what to do when the route is accessed**.
  ✅ **Example:** adding students, fetching students, handling forms.

- **Database Interaction -** The Flask application communicates with the SQLite database to perform CRUD operations, ensuring data persistence. Flask interacts with the DB through SQL queries to perform **CRUD operations**: **C**reate → Add a new student record ; **R**ead → Fetch existing records ;
  Update → Modify student details ; **D**elete → Remove a record

- **Templates (View) -** Jinja2 templates serve as the "View" layer, rendering dynamic content received from the Flask application into HTML for the user. Instead of hardcoding HTML, we use placeholders like {{ student.FULL_NAME }} to render actual data.
  ✅ **Example:** If there are 3 students in the database, this template will automatically list all 3 without manually writing HTML for each one.

Project Structure(Organizing the CodeBase)

❖ A well-defined folder structure promotes clear separation of concerns, making the project easy to navigate, understand, and maintain.

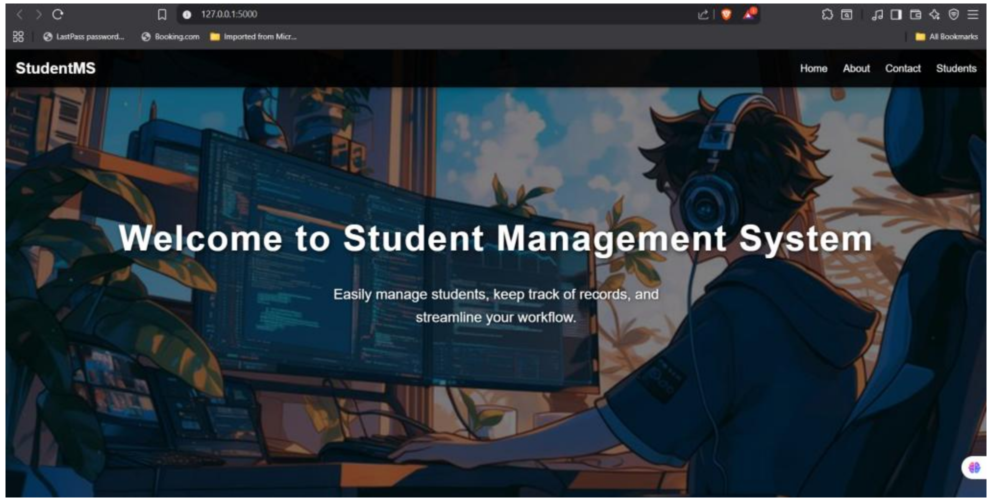


```
student_management/
|── app.py
|── schema.sql
|── requirements.txt
|── init_db.py
├── instance/
│   └── students.db
├── database/
│   └── db.py
├── templates/
│   ├── index.html
│   ├── students.html
│   └── ... (other HTML files)
└── static/
    ├── css/
    │   └── style.css
    ├── js/
    └── images/
```

- `app.py`: The main Flask application entry point, defining routes and core logic.
- `schema.sql`: SQL script to define the database tables.
- `requirements.txt`: Lists all Python dependencies.
- `init_db.py`: Script to initialize the database.
- `instance/`: Contains the actual SQLite database file (`students.db`).
- `database/`: Module for database-related functions.
- `templates/`: Stores all Jinja2 HTML template files for rendering web pages.
- `static/`: For static assets like CSS stylesheets, JavaScript files, and images.

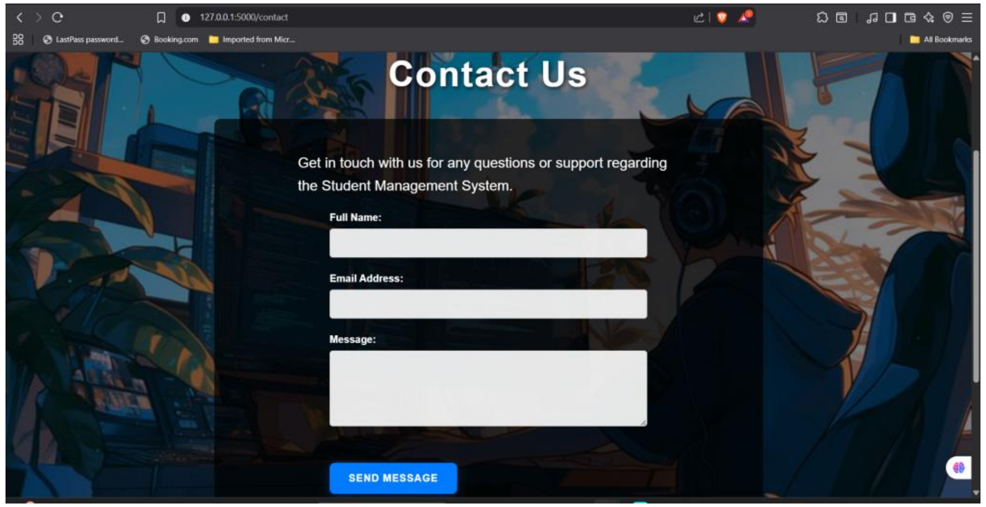
UI Preview (A Glimpse of User Interface)

- **Fig.1:** Homepage, **Fig. 2:** Contact Us, **Fig. 3:** About Us, **Fig. 4 :** Student Management Page



Welcome to Student Management System

Easily manage students, keep track of records, and streamline your workflow.



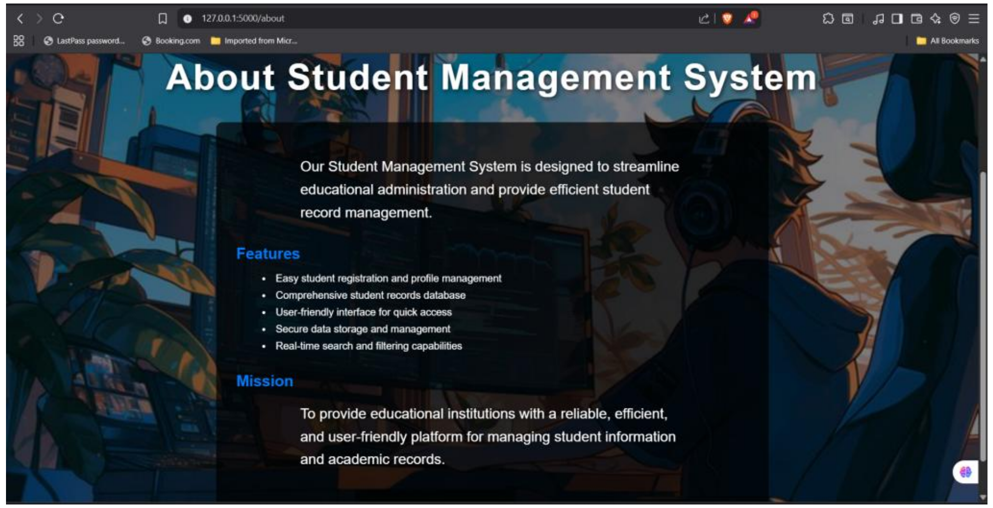
Contact Us

Get in touch with us for any questions or support regarding the Student Management System.

Full Name:

Email Address:

Message:



About Student Management System

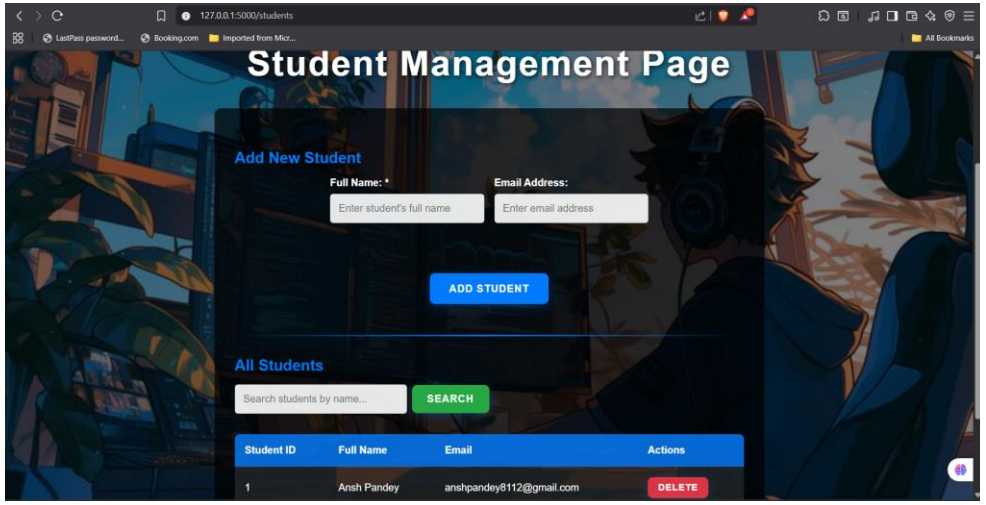
Our Student Management System is designed to streamline educational administration and provide efficient student record management.

Features

- Easy student registration and profile management
- Comprehensive student records database
- User-friendly interface for quick access
- Secure data storage and management
- Real-time search and filtering capabilities

Mission

To provide educational institutions with a reliable, efficient, and user-friendly platform for managing student information and academic records.



Student Management Page

Add New Student

Full Name: * Enter student's full name

Email Address: Enter email address

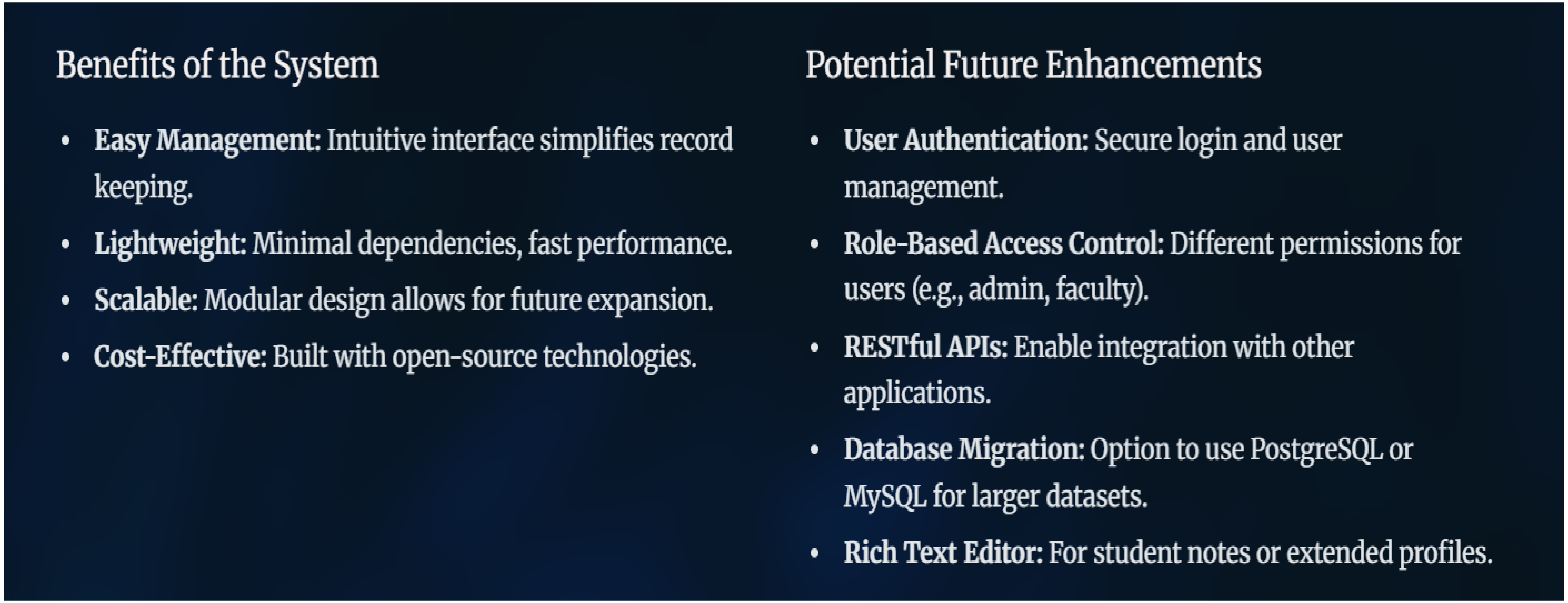
All Students

Search students by name...

Student ID	Full Name	Email	Actions
1	Ansh Pandey	anshpandey8112@gmail.com	DELETE

Conclusion & Future Scope

❖ The Flask Student Management System provides a solid foundation for efficient data handling, but there's always room for growth.



Benefits of the System

- **Easy Management:** Intuitive interface simplifies record keeping.
- **Lightweight:** Minimal dependencies, fast performance.
- **Scalable:** Modular design allows for future expansion.
- **Cost-Effective:** Built with open-source technologies.

Potential Future Enhancements

- **User Authentication:** Secure login and user management.
- **Role-Based Access Control:** Different permissions for users (e.g., admin, faculty).
- **RESTful APIs:** Enable integration with other applications.
- **Database Migration:** Option to use PostgreSQL or MySQL for larger datasets.
- **Rich Text Editor:** For student notes or extended profiles.

THANK YOU