# 

# Python Project Proposal

Project name

# “E-Learning platform”

# Distribution of work

|  |  |
| --- | --- |
| Abdukholik Tukhtamishev, 474886 | Otajon Yuldashev, 473457 |
| -Introduction | -Description of SQL Queries |
| -Data base schema | -Pictures of Designed Application |
| -Description of SQL Queries | -External Python Packages |
| -External Python Packages | -User Dashboard |
| -Posts | -Searching tutorial |
| -Calendar | -Testing and debugging |
| - Connect the database to Python using a library like sqlite3 | -Logout |

# Introduction

The Educational Platform is a web application designed to streamline interactions between students, teachers, and administrators in an educational environment. Built using Python and Streamlit, the platform provides an intuitive interface for managing user roles, posts, comments, grades, and class schedules. The application incorporates a robust SQLite database for handling data efficiently and ensuring the integrity of user information, content, and academic records. Teachers can create posts with file attachments, manage grades, and schedule classes, while students can engage by viewing posts, leaving comments, and accessing their grades with detailed feedback. The platform also includes secure user authentication, role-based access control, and integrated search functionalities for exploring resources on platforms like YouTube, Google, Wikipedia, and Stack Overflow. By combining modern web development tools and efficient database management, the Educational Platform delivers a comprehensive solution for fostering collaboration and learning in an organized and interactive manner.

# Database Schema

# 

# Description of SQL Queries

The **"CREATE TABLE IF NOT EXISTS users"** query creates a table called users. It defines the structure for storing user information such as a unique identifier (user\_id), the username, the password for login, and the role (which can be "admin," "teacher," or "student"). The user\_id is set to auto-increment, meaning it will automatically assign a unique value for each new user. The username is defined as unique, meaning no two users can have the same username.

The **"CREATE TABLE IF NOT EXISTS posts"** query creates a table called posts. This table is used to store posts created by users. It includes columns for a unique post\_id, a user\_id that links the post to a user, the title and content of the post, as well as a file\_path for any attached file and the file\_type of the file. The date\_posted column stores the date and time when the post was created. The user\_id references the user\_id from the users table, establishing a relationship between the post and the user who created it.

The **"CREATE TABLE IF NOT EXISTS comments"** query creates a table called comments, which stores user comments on posts. The table includes a unique comment\_id for each comment, a post\_id to link the comment to the relevant post, a user\_id to identify which user made the comment, the comment\_text itself, and the date\_commented column to store the timestamp of when the comment was made. Both post\_id and user\_id are foreign keys, meaning they reference the respective post\_id from the posts table and user\_id from the users table, ensuring that the comment is associated with a specific post and user.

The **"CREATE TABLE IF NOT EXISTS classes"** query creates a table called classes to store information about classes. This includes a unique class\_id, the user\_id of the instructor who created the class, the class\_title (name of the class), and the class\_date when the class will take place. The user\_id is a foreign key referencing the users table, linking each class to a specific instructor.

The **"CREATE TABLE IF NOT EXISTS grades"** query creates a table called grades, which stores grade information for students in various classes. The table includes a unique grade\_id, the user\_id of the student who received the grade, the class\_id of the class for which the grade was awarded, the grade value itself, and the date\_assigned to record when the grade was given. Additionally, there is a comments field where instructors can leave feedback on the student's performance. The user\_id references the users table, and the class\_id references the classes table, establishing relationships with both the student and the class.

The **"INSERT INTO users"** query is used to add a new record to the users table. It takes three values: the username, password, and role of the user. This query is used when a new user registers. If a user with the same username already exists, an error will occur due to the unique constraint on the username column.

The **"INSERT INTO grades"** query is used to assign a grade to a student. It takes the user\_id of the student, the class\_id of the class, the grade value, the date\_assigned, and optional comments. This query inserts a new grade record into the grades table.

The **"INSERT INTO posts"** query adds a new post to the posts table. It requires the user\_id of the user creating the post, the title and content of the post, the file\_path and file\_type (if a file is attached), and the date\_posted, which is the timestamp when the post was created. This query is used when a user (typically a teacher) creates a new post.

The **"INSERT INTO comments"** query is used to add a comment to a post. It requires the post\_id to specify which post the comment belongs to, the user\_id of the user making the comment, the comment\_text itself, and the date\_commented, which stores when the comment was made. This query is executed when a user adds a comment to a post.

The **"INSERT INTO classes"** query is used to add a new class to the classes table. It requires the user\_id of the instructor, the class\_title (name of the class), and the class\_date (date and time of the class). This query is executed when an instructor adds a class to the system.

The **"SELECT user\_id, role FROM users WHERE username = ? AND password = ?"** query is used to authenticate a user. It retrieves the user\_id and role of the user if their username and password match the values provided. This query is executed during login to verify the user's credentials.

The **"SELECT classes.class\_title, grades.grade, grades.date\_assigned, grades.comments FROM grades JOIN classes ON grades.class\_id = classes.class\_id WHERE grades.user\_id = ?"** query retrieves the grades for a specific user (student). It joins the grades and classes tables to get the class\_title, grade, date\_assigned, and comments for each grade the student has received. This query is used when fetching a student's grades.

The **"SELECT posts.post\_id, users.username, posts.title, posts.content, posts.file\_path, posts.file\_type, posts.date\_posted FROM posts JOIN users ON posts.user\_id = users.user\_id ORDER BY posts.date\_posted DESC"** query retrieves all posts in the system, ordered by the date they were posted. It joins the posts and users tables to display information about each post, including the post\_id, the username of the author, the title, content, file\_path, file\_type, and the date\_posted. This query is used to fetch all posts for display.

The **"SELECT users.username, comments.comment\_text, comments.date\_commented FROM comments JOIN users ON comments.user\_id = users.user\_id WHERE post\_id = ? ORDER BY comments.date\_commented ASC"** query retrieves the comments for a specific post. It joins the comments and users tables to get the username of the user who made the comment, the comment\_text, and the date\_commented. This query is used when displaying comments for a post.

The **"SELECT users.username, classes.class\_title, classes.class\_date FROM classes JOIN users ON classes.user\_id = users.user\_id ORDER BY classes.class\_date ASC"** query retrieves all classes, ordered by the class\_date. It joins the classes and users tables to get the username of the instructor, the class\_title, and the class\_date. This query is used to display the list of all classes.

The **"DELETE FROM posts WHERE post\_id = ?"** query is used to delete a post from the posts table. It removes the post with the specified post\_id. This query is executed when an admin or teacher deletes a post from the system.

Each of these queries performs a specific function, whether it's creating tables, inserting records, fetching data, or deleting records from the database, ensuring that the application works smoothly and maintains data integrity.

# Description of Python libraries

1. Streamlit

Purpose: Streamlit was used for creating the user interface (UI) of the application. It allows developers to build interactive web apps easily using Python.

Usage:

- Designed the navigation menu and role-specific pages (e.g., posts, grades, calendar).

- Provided input fields for login, registration, and form submissions.

- Used widgets like st.text\_input, st.text\_area, st.file\_uploader, st.expander, and st.download\_button for interaction.

- Enabled dynamic content rendering based on user roles and actions.

2. SQLite3

Purpose: SQLite was used as the database for storing and managing data, such as user accounts, posts, comments, grades, and classes.

Usage:

- Created and managed tables (users, posts, comments, grades, classes).

- Performed SQL queries to handle data (e.g., inserting new posts, fetching comments, managing grades).

- Ensured data consistency with relational integrity using FOREIGN KEY constraints.

3. Datetime

Purpose: The datetime module was used for capturing and formatting timestamps for various events, such as posting content, adding comments, assigning grades, and scheduling classes.

Usage:

- Generated timestamps using datetime.now() and formatted them with strftime for consistent storage and display.

- Used to track when posts were created, comments were added, and grades were assigned.

4. OS

Purpose: The os module was used for file and directory management to support file uploads and downloads.

Usage:

- Ensured the existence of the uploads directory for storing user-uploaded files using os.makedirs().

- Managed file paths to save and retrieve uploaded files.

5. Pathlib

Purpose: Pathlib was used for handling file paths in a more modern and platform-independent way.

Usage:

- Constructed file paths for uploaded files using Path.

- Provided cleaner and safer file path handling compared to traditional os.path methods.

6. Web browser

Purpose: The web browser module was used to enable users to perform external searches directly from the platform.

Usage:

- Opened search queries on platforms like YouTube, Google, Wikipedia, and Stack Overflow.

- Constructed search URLs dynamically and launched them in the user’s default browser.

Summary

These libraries were carefully selected to handle different aspects of the platform:

- Streamlit for UI development.

- SQLite3 for data storage and management.

- Datetime for managing timestamps.

- OS and Pathlib for file handling.

- Webbrowser for integrating external search functionality.

**Login and registration page**

A screenshot of a computer

Description automatically generated

**Login and Registration Page Overview**

The illustration above depicts the login and registration page. To begin, new members must register by providing a unique **Username**, **Password**, and selecting their **Role** (Student, Teacher, or Admin).

For added security, a **privacy feature** allows users to hide their passwords during both registration and login. Once all required fields are completed, users can click the **Register** button. If the registration is successful, a green notification will appear, confirming that the user has been registered.

If a user attempts to register with a username that already exists in the database, they will receive a red notification informing them of the issue. Similarly, incorrect or missing details during login or registration will also trigger a red notification, ensuring users are clearly informed about any errors.

After successful registration, users can log in using the **Login** section located just below the registration area. Here, users need to enter their credentials and press the **Login** button. If the credentials are correct, the system will grant access seamlessly.

This streamlined design ensures a secure and user-friendly experience for all roles.

**Dashboard and Posts Page Overview**

Once a teacher, student, or admin logs in, they are greeted with a **Dashboard** on the left-hand side of the screen. The Dashboard provides quick access to key pages, including **Posts**, **Calendar**, **Grades**, **Search Tutorials**, **Statistics**, and the option to **Log Out**.

A screenshot of a computer

Description automatically generated

The current page displayed in the image is the **Posts** section. This page allows teachers to create and share posts with students. Teachers can:

* Add a **Post Title** to provide a clear summary of the content.
* Enter detailed information or instructions in the **Post Content** section.
* Optionally, upload files (e.g., PDFs, images, or documents) through the **Upload File** feature, which supports file sizes up to 200MB.

After completing the necessary fields, teachers can click the **Post** button to share the post. Once submitted, the post will appear in the **All-Posts** section, as shown at the bottom of the image.

The posts created by teachers are automatically displayed in the students' accounts under the **Posts** section, ensuring easy access to shared resources and announcements. This feature enhances communication and streamlines the sharing of educational materials.

A screenshot of a computer

Description automatically generated

**Calendar page**

**A screenshot of a computer

Description automatically generated**

The **Calendar Page** enables teachers to efficiently manage class schedules by adding important updates or rescheduling information.

In the first image, the **Add Class to Calendar** feature is displayed. Teachers can:

* Enter the **Class Title** to specify the update (e.g., “Statistics class is postponed to the next Friday”).
* Set the **Class Date** using the date picker field.
* Click the **Add Class** button to save the class or update.

When a class is successfully added to the calendar, a green notification ("Class added to calendar!") confirms the action. The added class is then displayed in the **Class Calendar** section, visible to both the teacher and students.

In the image below, the **Class Calendar** page for a student is shown. The newly added class update ("Statistics class is postponed to the next Friday by teacher on 2025-01-31") appears under the calendar section, ensuring that students are kept informed about any changes or updates.

This streamlined feature ensures that all teachers and students efficiently track and manage class schedules in real time.

**A screenshot of a computer

Description automatically generated**

**Grading page**

A screenshot of a computer

Description automatically generated

The **Grading Page** is a dedicated feature for teachers, allowing them to assign grades to students based on their performance. This page is designed to simplify the grading process by including the following steps:

* **Enter Student Username**: Teachers input the unique username of the student to ensure accurate identification.
* **Enter Class ID**: Teachers provide the specific class ID to associate the grade with the correct course.
* **Enter Grade**: Teachers assign a grade (e.g., numerical score or letter grade) based on the student's performance.
* **Add Comments (Optional)**: Teachers can include personalized comments regarding the student’s performance, such as encouragement or constructive feedback (e.g., "You did a very good job! Keep it up!").

Once all fields are completed, teachers click the **Assign Grade** button to finalize the grading process. This ensures that the grade and any accompanying comments are saved and associated with the student.

This streamlined functionality enables teachers to efficiently manage and provide meaningful feedback on student performance, fostering better communication and academic growth.

**Search resources**

A screenshot of a computer

Description automatically generated

**Search Resources Overview**

The **Search Resources** page is available for both students and teachers, providing a convenient way to find relevant information on any topic of interest. This feature supports quick access to four major platforms: **YouTube**, **Google**, **Wikipedia**, and **Stack Overflow**.

To use this feature:

1. Enter a topic in the **Enter Topic** search field.
2. Select one of the four options:
   * **Search on YouTube**: Find video tutorials or lectures on the topic.
   * **Search on Google**: Explore a wide range of web resources and articles.
   * **Search on Wikipedia**: Access detailed summaries and explanations.
   * **Search on Stack Overflow**: Get coding-related help and solutions for technical questions.

This tool is beneficial for students to explore additional information or gain a deeper understanding of their subject. For teachers, it offers a quick way to refresh their knowledge or find supplemental material before a class. The intuitive design ensures that both students and teachers can easily find resources tailored to their needs with just a few clicks.

**Logout**

**A screenshot of a computer

Description automatically generated**

**Logout Page Overview**

The **Logout Page** is the final step in the navigation menu, allowing users to securely log out of their accounts. Upon selecting the **Logout** option, users are redirected to a confirmation screen, ensuring that their session has ended successfully.

The interface also displays a message, reminding users that they need to log in again to access features such as creating posts, adding calendar entries, or managing other functionalities.

This feature ensures account security by preventing unauthorized access after a session ends, providing users with peace of mind while using the system.