EDGES

OUR EDGE IS YOUR SUCCESS

Student Database Management System with Chatbot and User Authentication

1. Project Description:

This project is designed to create a Student Database Management System with a Chatbot Interface and User Authentication (both Admin and normal users). The app will allow users to perform various tasks such as:

- **Login/Registration: Admins can log in to manage the student database, while normal users can register and log in to interact with the chatbot.**
- **Admin Dashboard: Admins can add, update, delete, and view students, as well as handle bulk uploads via CSV files.**
- **User Dashboard: Normal users can interact with the chatbot to ask questions about student data stored in the database.**
- **JSON-based Credentials: Admin credentials are fetched from a credentials.json file, while user credentials are stored in a users.json file.**
- **Chatbot Functionality: The chatbot responds with predefined queries based on student data, simulating how a more advanced AI chatbot might behave in the future.**

---

2. Tasks to Complete the Project:

1. **Set up the MySQL Database:**
    - **Create a database (student_db) and a students table.**
    - **Include fields like id, name, age, and grade.**
2. **Create JSON files for Admin and User Credentials:**
    - **Create a credentials.json file for admin credentials.**
    - **Create a users.json file for storing user registrations.**
3. **Develop Backend in Python (OOP):**
    - **Create Student, Database, and Chatbot classes to handle the business logic of managing students and handling chatbot queries.**
4. **Develop the Frontend with Streamlit:**
    - **Create a login page for both admins and users.**
    - **Implement a registration page for new users.**
    - **Design a Home Page with functionalities based on user type (Admin or Normal User).**
    - **Build a Chatbot Interface for user interaction.**
5. **Integrate the Backend and Frontend:**
    - **Ensure the frontend (Streamlit) is connected to the backend Python classes and MySQL database.**

6. **Enhance Security:**
    - **Use SHA-256 hashing for secure password storage.**
    - **Ensure passwords are not stored in plain text.**
7. **Prepare for Future Enhancements:**
    - **Prepare the system to be easily extendable for integrating advanced AI-driven chatbots.**
8. **Final Testing and Debugging:**
    - **Test the login, registration, and student management functionalities.**
    - **Ensure proper error handling and security measures are in place.**

---

3. Detailed Description of Each Task:

Task 1: Set up the MySQL Database

- **Objective: Create a student_db MySQL database and a students table to store student data.**
- **Steps:**
    - **Install MySQL or use a local development environment like XAMPP.**
    - **Connect to MySQL and create the database student_db.**
    - **Create the students table with the following fields:**
        - **id INT AUTO_INCREMENT PRIMARY KEY**
        - **name VARCHAR(255)**
        - **age INT**
        - **grade VARCHAR(50)**
    - **Example SQL to create the table:**

```
CREATE DATABASE student_db;
USE student_db;

CREATE TABLE students (
   id INT AUTO_INCREMENT PRIMARY KEY,
   name VARCHAR(255),
   age INT,
   grade VARCHAR(50)
);
```

Task 2: Create JSON Files for Admin and User Credentials

- **Objective: Store admin and user credentials in JSON files for easy management.**
- **Steps:**
    - **credentials.json: Store admin credentials (username and password).**
    - **users.json: Store user credentials, which will be dynamically added during registration.**

Sample content of credentials.json:

```
{
  "admin": {
    "username": "admin",
    "password": "admin123"
  }
}
```

Sample content of users.json:

```
{
  "admin": {
    "username": "admin",
    "password": "admin123"
  }
}
```

Task 3: Develop Backend in Python (OOP)

- **Objective: Develop Python classes using Object-Oriented Programming (OOP) to manage the core logic of the application.**
    - **Student Class:**
        - **Models student data with attributes like name, age, and grade.**
        - **Includes methods for updating student information.**
    - **Database Class:**
        - **Manages database interactions, including CRUD operations (Create, Read, Update, Delete).**
        - **Methods to interact with the students table (insert, fetch, delete).**
    - **Chatbot Class:**

        - **Handles predefined queries related to student information, simulating an AI-driven chatbot.**
        - **Fetches data from the database and responds to user inputs.**

Example of the Student class:

```
class Student:
    def __init__(self, student_id=None, name=None, age=None,
grade=None):
        self.student_id = student_id
        self.name = name
        self.age = age
        self.grade = grade

    def update(self, name=None, age=None, grade=None):
        if name:
            self.name = name
        if age:
            self.age = age
        if grade:
            self.grade = grade
```

Task 4: Develop the Frontend with Streamlit

- **Objective: Create a user-friendly interface using Streamlit that allows users to interact with the system.**
    - **Login Page:**
        - **Implement a login form where users can log in as either Admin or User.**
        - **If the user is an admin, show the Admin Dashboard with student management options.**
        - **If the user is a normal user, show the User Dashboard with chatbot interaction.**
    - **Registration Page:**
        - **Allow new users to register by entering a username and password. Store their credentials in users.json.**
    - **Home Page:**
        - **Based on the login type (Admin or User), show the appropriate dashboard.**
    - **Chatbot Interface:**
        - **Display a text box where users can ask questions, and the chatbot will respond.**

Task 5: Integrate the Backend and Frontend

- **Objective: Link the backend Python classes with the frontend Streamlit UI.**
  - **Make sure the login page correctly checks credentials from the JSON files.**
  - **Ensure the Home Page reflects the functionalities based on whether the user is an Admin or a User.**
  - **Connect the chatbot with the Chatbot class so that it fetches relevant information from the MySQL database.**

Task 6: Enhance Security- BONUS

- **Objective: Implement secure password storage using SHA-256 hashing.**
  - **Passwords entered during registration are hashed and stored securely.**

Example of password hashing:

```
from hashlib import sha256

hashed_password = sha256(password.encode()).hexdigest()
```

Task 7: Prepare for Future Enhancements

- **Objective: Ensure the system is extensible for adding more advanced features in the future.**
  - **Design the system so it can later integrate with an LLM-powered chatbot for advanced interactions.**
  - **Ensure the database is scalable and can handle a larger amount of data.**

Task 8: Final Testing and Debugging

- **Objective: Ensure all features work as expected and that the system is free from bugs.**
  - **Test login, registration, student management, and chatbot functionality.**
  - **Handle any potential errors like invalid input or database connection failures.**
  - **Ensure the UI is responsive and user-friendly.**

---

4. Guidance to Make it OOP:

To make the project follow Object-Oriented Programming (OOP) principles:

1. **Encapsulation:**
   - **Group related data and functionality into classes (e.g., Student, Database, Chatbot).**
   - **Use class methods to perform actions on the data (e.g., update() method in the Student class).**

2. **Abstraction:**
    - **Abstract complex functionality like database operations into methods (e.g., insert_student(), fetch_students()).**
    - **Provide simple interfaces for the user (e.g., methods for adding or fetching student data without exposing the database logic).**
3. **Inheritance:**
    - **While inheritance is not necessary for this simple project, you could extend the Chatbot class in the future if different chatbot models or behaviors are added.**
4. **Polymorphism:**
    - **You can define generic methods that can be overridden if needed. For example, the chatbot could have different responses based on the user type.**

---

5. Deliverables:

1. Recorded Demo of the Running Project:

- **A video recording that demonstrates:**
    - **The registration process.**
    - **User login.**
    - **Admin dashboard functionalities (Add, Update, Delete, View Students).**
    - **Chatbot interaction for both Admin and Normal Users.**
    - **Any security measures (like login and registration with hashed passwords).**
    - **Handling of errors and successful operations.**

2. ZIP Folder with Source Code:

- **A zipped folder containing:**
    - **Python source code for the backend (OOP classes and Streamlit app).**
    - **credentials.json and users.json files (to store credentials).**
    - **A readme file detailing how to run the project and dependencies.**

Folder structure example:

```
student_management_system/
|
├── app.py (Streamlit app)
├── student.py (Student Class)
├── database.py (Database Class)
├── chatbot.py (Chatbot Class)
├── credentials.json
├── users.json
└── README.md
```

Thank You

Edges For Training Team