Shangri-La Petition Platform (SLPP)

Overview

The Shangri-La Petition Platform (SLPP) is a web application that allows citizens to create, manage, and sign petitions. It also includes an admin interface for overseeing petition activities, including setting thresholds and responding to petitions.

Requirements

To successfully run this project, ensure you have the following:

- Python 3.8 or higher
- Flask framework
- SQLite for database management
- Postman (optional)
- The following Python packages installed:
    - flask
    - flask_sqlalchemy
    - flask_bcrypt
    - flask_login
    - requests

Tools

- **Backend:**
- Flask: Python-based web framework for routing and backend logic.
- SQLAlchemy: ORM for database modeling and management.
- SQLite: Lightweight database for storing data.
- Flask-Bcrypt: Secure password hashing.
- Flask-Login: User session management.
- Frontend: HTML/CSS, Bootstrap, JavaScript
- Database: SQLite
- Other Tools: Postman(optional) for API Testing

Database Setup (if required)

```
flask shell
>>> from app import db
>>> db.create_all()
>>> exit()
```

To Run the Application

- python app.py
- Open your web browser and go to:
- http://127.0.0.1:5000/

Features

Petitioner

1. **User Registration**
   a. Users can register using their email, name, date of birth, password, and a valid BioID.
   b. Valid BioIDs are predefined and verified during registration.
2. **Login**
   a. Secure login using registered credentials.
3. **Petition Management**
   a. Create petitions with a title and content.
   b. View all petitions on the platform.
   c. Sign petitions created by other users.

Admin Features

1. **Admin Login**
   a. Secure login with pre-configured admin credentials.
2. **Petition Oversight**
   a. View all petitions and their details.
   b. Respond to petitions and close them.
   c. Set signature thresholds for petition responses.
3. **Signature Threshold Management**
   a. Change the minimum number of signatures required for a petition to be considered for a response.

APIs

- RESTful APIs for petition listing, details, and signature retrieval.
- Open data API endpoints for public consumption.

File Structure

- app.py: Main application logic and route definitions.
- test_register.py: Automated tests for user registration, login, petition creation, and admin functionality.

- database.db: SQLite database file storing user and petition data.
- templates/: HTML templates for rendering web pages.
- static/: Static files (CSS, JS).

Testing

- To run tests, execute the test_register.py script: **python test_register.py**
- Make sure that the main file i.e. app.py should be kept running while running the test_register.py.
- Testing using Postman:
  1. http://127.0.0.1:5000/slpp/petitions?status=open (Method GET)
  2. http://127.0.0.1:5000/slpp/petitions (Method GET)
  3. http://127.0.0.1:5000/api/petitions/5/signatures (Method GET)

And many others.

Known Issues

- Ensure the VALID_BIO_IDS list is up to date.
- Only unique email addresses and BioIDs are allowed.

Key Points

1. Remember that the port no. 5000 should be free and open in your system to work easily.
2. All the necessary libraries should be installed before running the program.