URL Shortener Project using Flask

Introduction:

This is a Python Flask web application that creates shortened URLs for long URLs using the pyshorteners library. Users need to register and login to use the URL shortening service.

The application uses SQLAlchemy to create and manage the database. There are two models: User, ShortURL, ShortURLPair. User stores user information, including username, hashed password, and name. ShortURL stores the user's shortened URLs. ShortURLPair stores Original and Shortened URLs by users.

The application also uses Flask-Login to handle user authentication. The load_user function is used to load the user from the database based on the user ID stored in the session. The login_user function is used to log the user in, and the login_required decorator is used to restrict access to the /shorten page.

The project is a URL shortener web application that allows users to register, log in, and shorten long URLs. The application uses Flask, a Python web framework, to create the backend and SQLAlchemy to handle database interactions. The frontend is created using HTML.

Features:

The application has the following routes:

/ - the home page, which contains a link to the registration page and a login form.

/register - the registration page, where users can create a new account. If the username is not taken and length of username should be in between 5 and 9 characters, and the passwords match, the account is created, and the user is redirected to the login page. Else, particular message is displayed to user. Eg: If username is taken by others, it displays “Username taken, try another”.

/login - the login page. If the user provides valid credentials, they are logged in and redirected to the home page.

/shorten - the URL shortening page. Users must be logged in to access this page. If the Users try to access this route without logging in, Login route will be displayed first, if the credentials available and match then user will be redirected to this page. Users can enter a long URL, which will be shortened using the pyshorteners library. The pyshorteners library is used to shorten the URLs. The pyshorteners.Shortener() function is used to create a shortener object, and the short() function is used to shorten the URL. The shortened URL is then associated with the user's account by creating a ShortURLPair object and adding it to the user's ShortURL object. If a user enters a URL that has already been short-end by him/her, it displays message “This URL is already short-end by you, Kindly check history”.

/shortend_urls – this is used to display all the URLs short-end by respective user after logging in. This shows both Original and short URLs. This can be viewed only if user access **/shorten** route.

The application has password hashing and user authentication features to ensure that user information is secure. When a user registers, their password is hashed using the werkzeug.security library. When a user logs in, their credentials are authenticated using the check_password_hash function from the same library. With this there is no leakage of passwords of users to the members working at server side.

The application has several features that allow users to manage their URLs effectively. Users can register and log in to the application, and once logged in, they can shorten long URLs. The application uses the pyshorteners library to generate short URLs. When a user enters a long URL, the application generates a short URL using the pyshorteners library. The short URL is then saved in the database along with the original long URL. This shortened URLs are saved in the database, and users can view their shortened URLs by clicking on the "History" link. Particular user can only view the URLs shortened by him/her. If the URL has already been shortened, the application shows to check history, in order to save the storage space.

The application also uses the Flask-Login library to manage user sessions. When a user logs in, a session is created, and the user is redirected to the home page. If the user tries to access a protected page without logging in, they are redirected to the login page.

Conclusion:

In conclusion, the URL shortener web application is a simple but useful tool for managing long URLs. The application uses Flask and SQLAlchemy to handle backend interactions and provides several features to ensure that user information is secure. The application also uses pyshorteners to generate short URLs and Flask-Login to manage user sessions. Overall, the application is a great example of how Flask can be used to create a simple yet effective web application.