Exercise 2.6: User Authentication in Django

Learning Goals

- Create authentication for your web application
- Use GET and POST methods
- Password protect your web application's views

Reflection Questions

- In your own words, write down the importance of incorporating authentication into an application. You can take an example application to explain your answer.

Incorporating authentication into an application is essential for ensuring security, protecting user data, and enabling personalized experiences. For example, in a social media application like Instagram, authentication verifies user identity, preventing unauthorized access to personal profiles, messages, and photos. This safeguards sensitive information from malicious users and allows for tailored content recommendations based on user preferences. Additionally, authentication enables users to manage their accounts securely, updating their information and privacy settings while enforcing access controls to restrict certain features to logged-in users.

- In your own words, explain the steps you should take to create a login for your Django web application.

- Set up a login view in the Django views file that handles user authentication.
- Next, create a login template (HTML file) to design the login form where users can enter their credentials.
- Register this view in the urls.py file to map the URL http://127.0.0.1:8000/login/ to the login view.

- Finally, add a clickable link or button on the homepage that directs users to the login page, ensuring they can access the authentication form easily.

- Look up the following three Django functions on Django's official documentation and/or other trusted sources and write a brief description of each.

Function	Description
authenticate()	
redirect()	
include()	

`authenticate()`: This function verifies a user's credentials (username and password) against the authentication backend. It returns a `User` object if successful; otherwise, it returns `None`. It is typically used in views to handle user login.

`redirect()`: This function returns an HTTP response that redirects the user to a different URL. It can take a URL path or a view name as an argument and is commonly used after form submissions to navigate users to the appropriate page.

`include()`: This function allows you to include other URLconf modules in the main URL configuration, promoting modular and organized URL routing. It takes a URLconf module path and appends its patterns to the main urlpatterns, making the code more maintainable.