Exercise 2.6: User Authentication in Django

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

Overall, incorporating authentication into an application is essential for maintaining the security, privacy, and accountability of an application, as well as for providing a customized user experience.

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

Set up the authentication system: Django comes with a built-in authentication system that can be easily added to your app. To set up authentication, you'll need to add 'django.contrib.auth' to the INSTALLED_APPS list in your project's settings.py file.

Define a User model: Django's authentication system uses a User model to represent users. You can customize the default User model or create a new model that inherits from it. To define a custom User model, create a new model class in your app's models.py file and set AUTH_USER_MODEL in your project's settings.py file.

Create a login view: Create a view that displays a login form and handles login requests. You can use Django's built-in LoginView or create a custom view that inherits from it.

Create a logout view: Create a view that logs the user out of the application. You can use Django's built-in LogoutView or create a custom view that inherits from it.

Protect views with authentication: To ensure that only authenticated users can access certain views, use Django's @login_required decorator or the LoginRequiredMixin mixin.

Include login and logout URLs in your app's URLs: Add URLs for the login and logout views to your app's URLs using Django's path() function.

Create templates: Create templates for the login and logout views using Django's templating system.

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()	It takes user credentials as parameters and checks if the user is valid against the backend data. If the user is valid, it returns a user object. If the user is not valid, it returns ‘none’.
redirect()	It takes the URL of the page you want a user to be directed to and it returns the view of that page, which in turn the view displays the corresponding template.
include()	It is used to add URLs from the apps directory to the main urls.py file in the project directory.