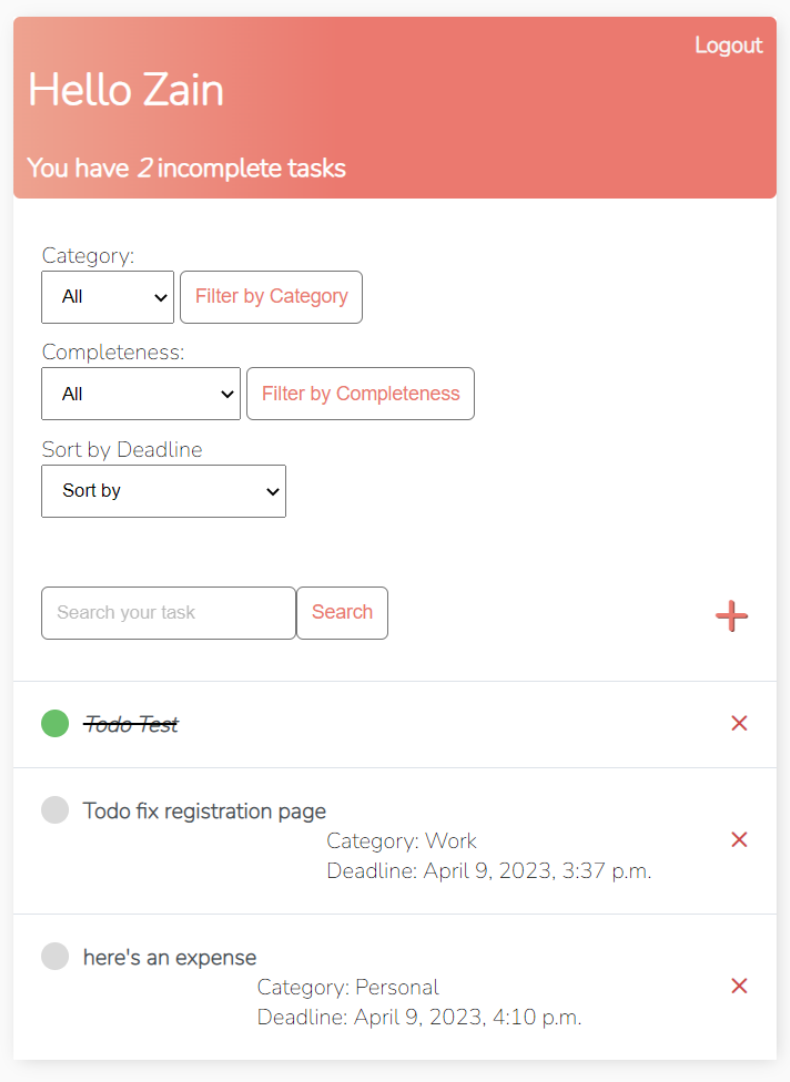
Django Todo List Documentation

Omema Rizvi (or06360), Zain Usmani (zu06777)

The to do list app is based on the following prompt:
"***Todo List Website:***
A personal website to manage a to-do list for users. This is a multi-user website allowing users to sign-up. After singing-in, a user can a to-do list by giving the task, category, and deadline. The user must be able to filter these tasks with respect to deadlines, categories, etc. These tasks should not be accessible/visible to any other users or guest users (users without credentials)."
That is:

Feature/Requirement	Description
To-do list	A list that allows users to add tasks, categories, and deadlines.
Multi-user functionality	A website that allows multiple users to sign up and access their own personalized to-do lists.
User authentication	Users must be authenticated in order to access their to-do list.
Task filtering	Users must be able to filter their tasks by category, deadline, and other relevant criteria.
User data isolation	Each user's to-do list data should not be accessible by any other users or guests.

We created a website with the above functionalities and CRUD for the to-do list entries. The website is a response web app which lets each logged in user create and manage their own todo list.



Greeting page and task list.

Views

Most of the views have been developed based on django's built-in functionality. The login, logout and registration pages are also largely handled by Django's **Class-based Views**. Some classes have been modified, e.g. login view and registration page form modified to achieve user authentication and data isolation.

Modified UserCreationForm to change the view of the registration page.

Privacy is handled by one of django's Login Mixins - which prevents a user from accessing another user's data.

CSS

The visual design was kept from DennisIvy's sample project with minor modifications.

Models

We created a model for Task that stores necessary details such as deadline, category, completion status etc. Each user has multiple tasks and each task has these attributes. The category attribute comes from another class. Creating a category model enables us to allow each user to create custom categories, associate them with different tasks and filter on their basis.

Further improvements

- The design for some of the pages could be improved.
    - The datetime picker on the add expense list is not very intuitive to use. Some libraries could be imported to provide widgets for that.
    - The filter button on the homepage could be more compact.
- Sorting and filtering don't work simultaneously.
- User could be allowed to add their own categories. Currently, only two categories exist created by the admin (Work/Personal).

Attributions

The website was created by following along a tutorial from DennisIvy..