Sarvida, Jhay-r C.

BSCPE 2B1

Laboratory Activity No. 3:

Topic belongs to: Programming Constructs and Paradigms
Title: Implementing CRUD Operations for Books and Users in the Library Management System
Introduction:
This activity involves implementing CRUD (Create, Read, Update, Delete) operations for books and users in the Library Management System. You will create views and forms to handle these operations via a web interface.

Objectives:

- Create views for CRUD operations.
- Use Django forms to manage user input.
- Learn about Django's URL routing for view handling.

Theory and Detailed Discussion:
CRUD operations are the fundamental actions in any database-driven application. Django's views and forms make it easy to manage the interaction between the user and the database. Using Django's URL routing system, we can map different actions (like creating or updating data) to specific views.

Materials, Software, and Libraries:

- Django Framework
- SQLite (or any database you are using)

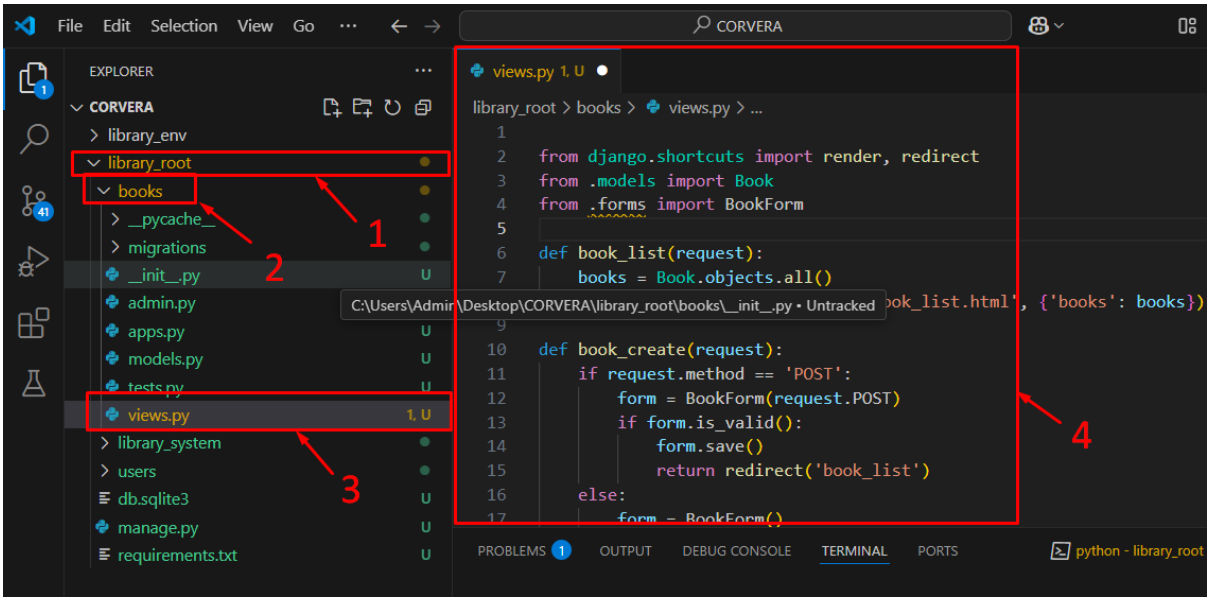
Time Frame:
2 hours

Step by Step Procedure:

1. Create views for handling CRUD operations:
    - Open books/views.py and add the following views:

```
from django.shortcuts import render, redirect

from .models import Book

from .forms import BookForm


def book_list(request):

  books = Book.objects.all()
```

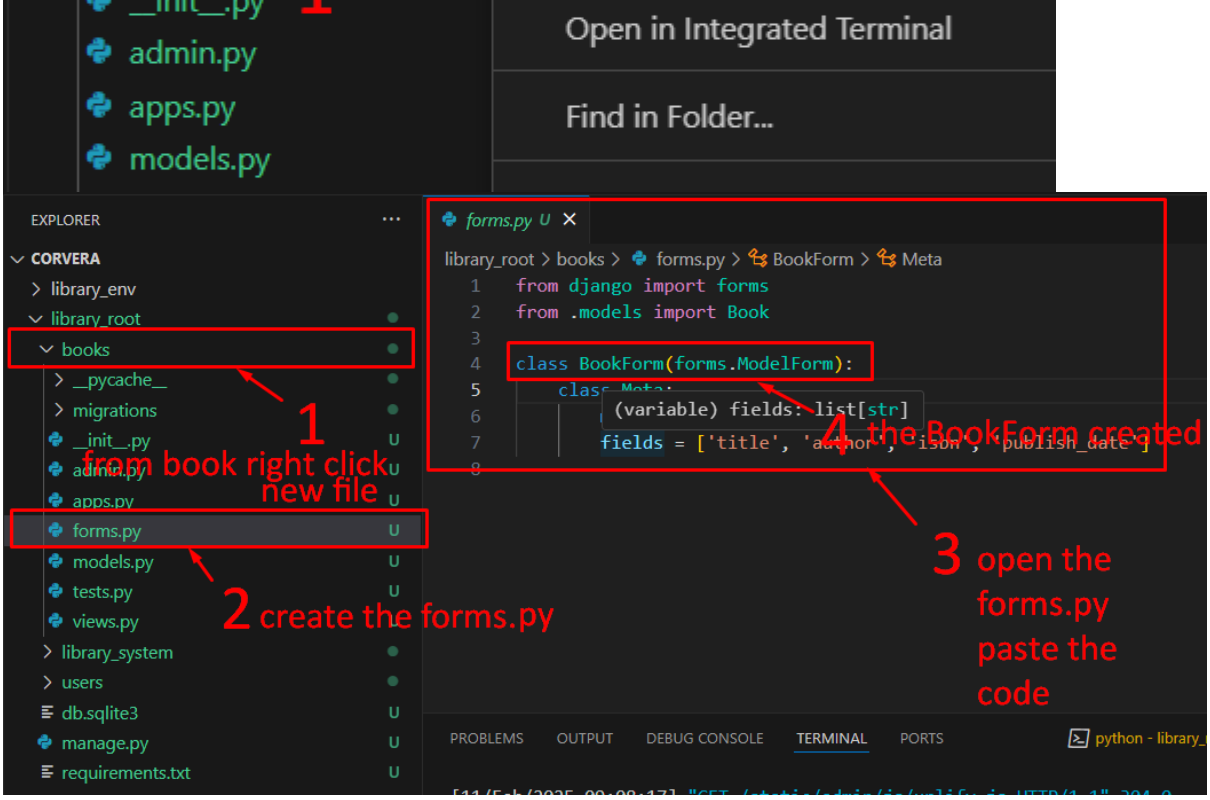
```
    return render(request, 'books/book_list.html', {'books': books})

def book_create(request):
    if request.method == 'POST':
        form = BookForm(request.POST)
        if form.is_valid():
            form.save()
            return redirect('book_list')
    else:
        form = BookForm()
    return render(request, 'books/book_form.html', {'form': form})

def book_update(request, pk):
    book = Book.objects.get(pk=pk)
    if request.method == 'POST':
        form = BookForm(request.POST, instance=book)
        if form.is_valid():
            form.save()
            return redirect('book_list')
    else:
        form = BookForm(instance=book)
    return render(request, 'books/book_form.html', {'form': form})

def book_delete(request, pk):
    book = Book.objects.get(pk=pk)
    if request.method == 'POST':
        book.delete()
        return redirect('book_list')
    return render(request, 'books/book_confirm_delete.html', {'book': book})
```



2. Create a BookForm in books/forms.py:

   Create first the forms.py then inside of it past the code.

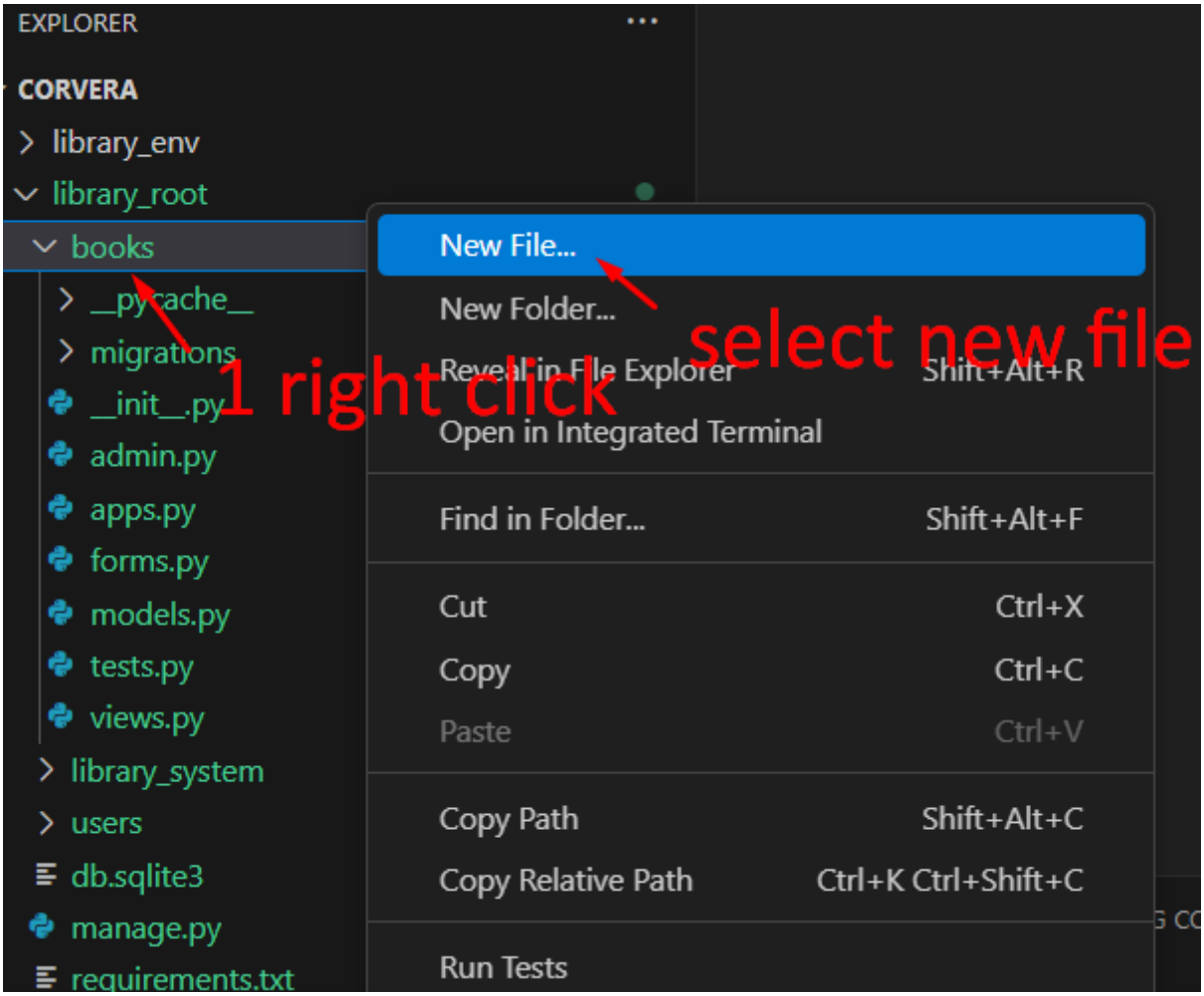
```
from django import forms
from .models import Book

class BookForm(forms.ModelForm):
  class Meta:
    model = Book
    fields = ['title', 'author', 'isbn', 'publish_date']
```

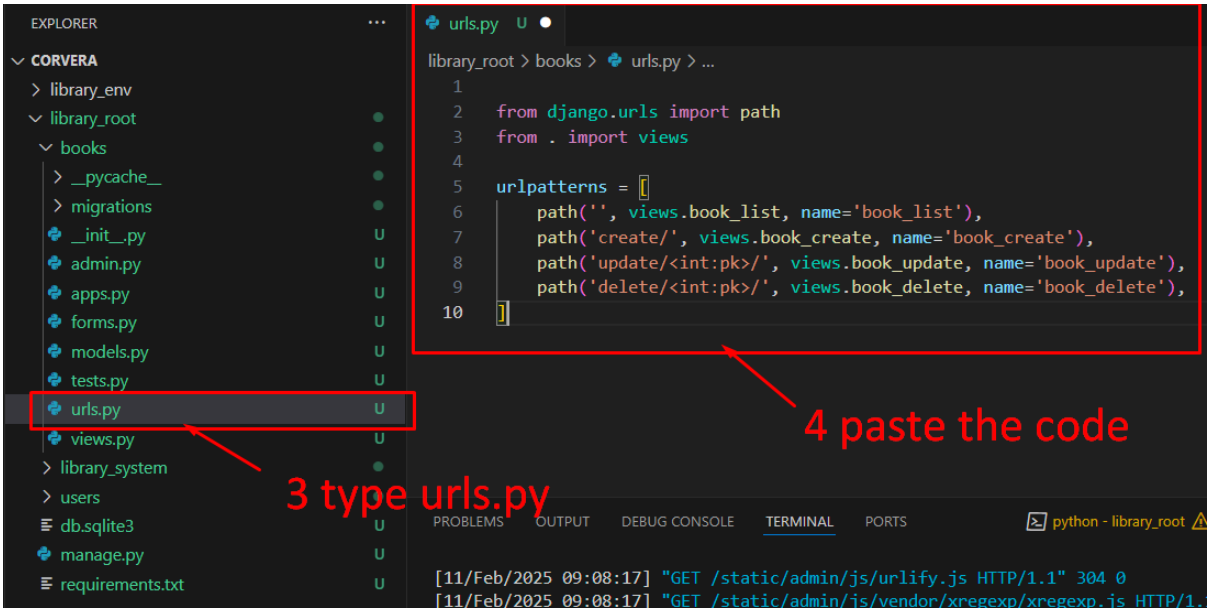


1. Define the URLs for the views in books/urls.py:

   Under the urls.py

   ```
   from django.urls import path
   from . import views

   urlpatterns = [
     path('', views.book_list, name='book_list'),
     path('create/', views.book_create, name='book_create'),
     path('update/<int:pk>/', views.book_update, name='book_update'),
     path('delete/<int:pk>/', views.book_delete, name='book_delete'),
   ]
   ```





1. Register the URLs in library_system/urls.py:

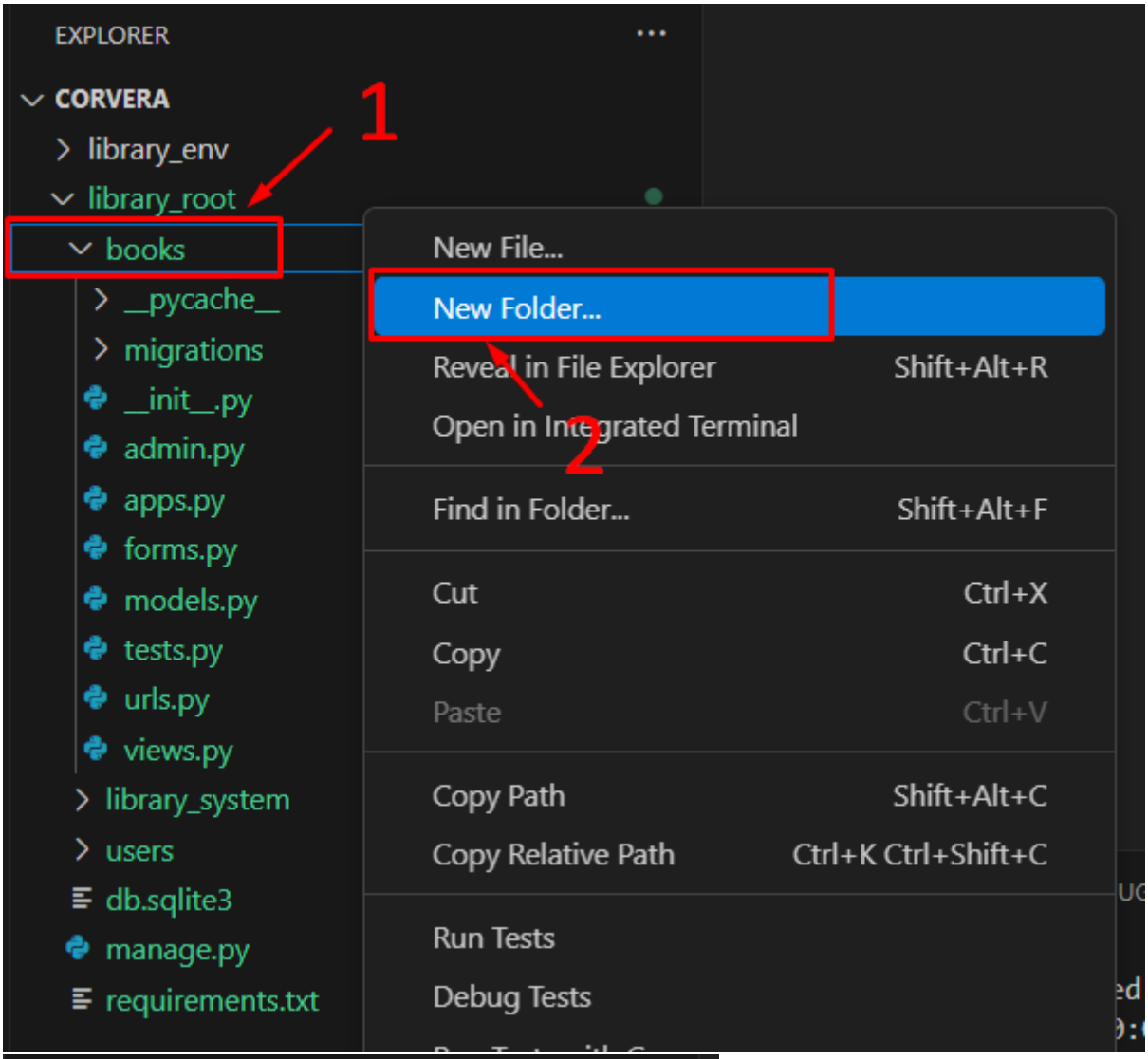
   from django.contrib import admin

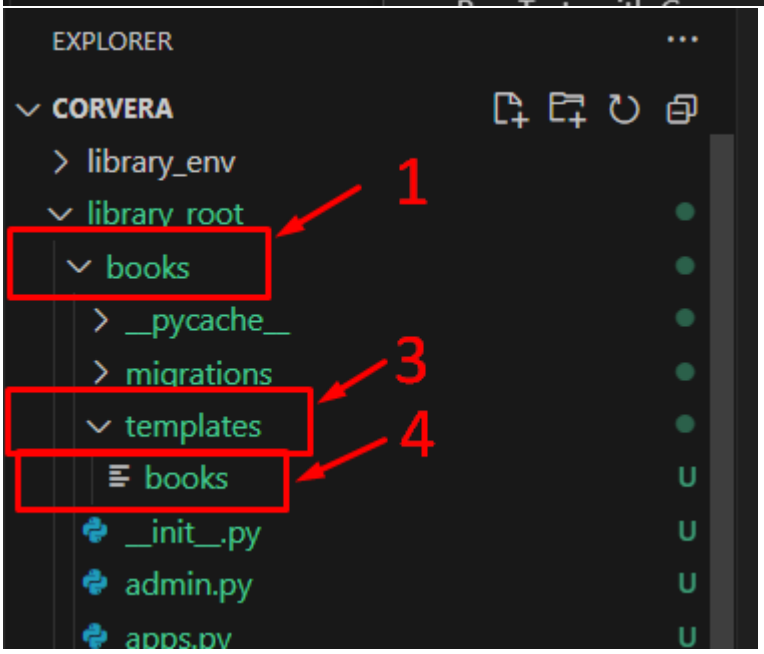
   from django.urls import path, include

```
urlpatterns = [
  path('admin/', admin.site.urls),
  path('books/', include('books.urls')),
]
```

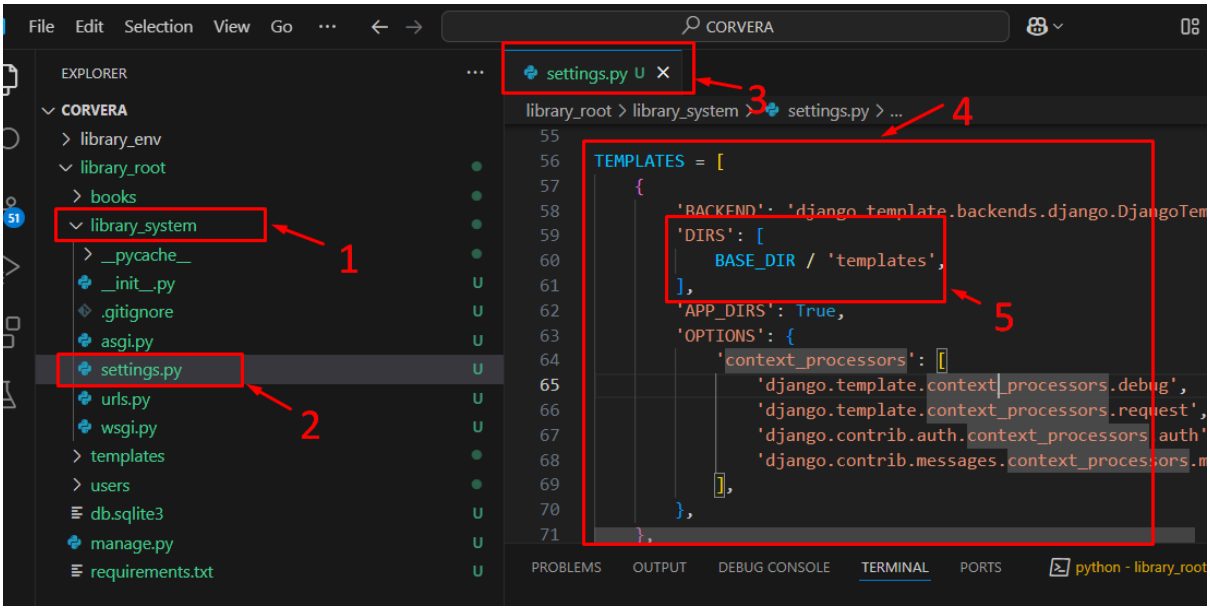
1. Create the HTML templates (book_list.html, book_form.html, book_confirm_delete.html) to render the CRUD forms.

   Inside the books (1) create new folder named templates (3). Inside the templates(3) create a folder books (4).


EXPLORER
CORVERA
library_env
library_root
books
__pycache__
migrations
__init__.py
admin.py
apps.py
forms.py
models.py
tests.py
urls.py
views.py
library_system
users
db.sqlite3
manage.py
requirements.txt
1
2
New File...
New Folder...
Reveal in File Explorer Shift+Alt+R
Open in Integrated Terminal
Find in Folder... Shift+Alt+F
Cut Ctrl+X
Copy Ctrl+C
Paste Ctrl+V
Copy Path Shift+Alt+C
Copy Relative Path Ctrl+K Ctrl+Shift+C
Run Tests
Debug Tests


EXPLORER
CORVERA
library_env
library_root
books
__pycache__
migrations
templates
books
__init__.py
admin.py
apps.py
1
3
4
U

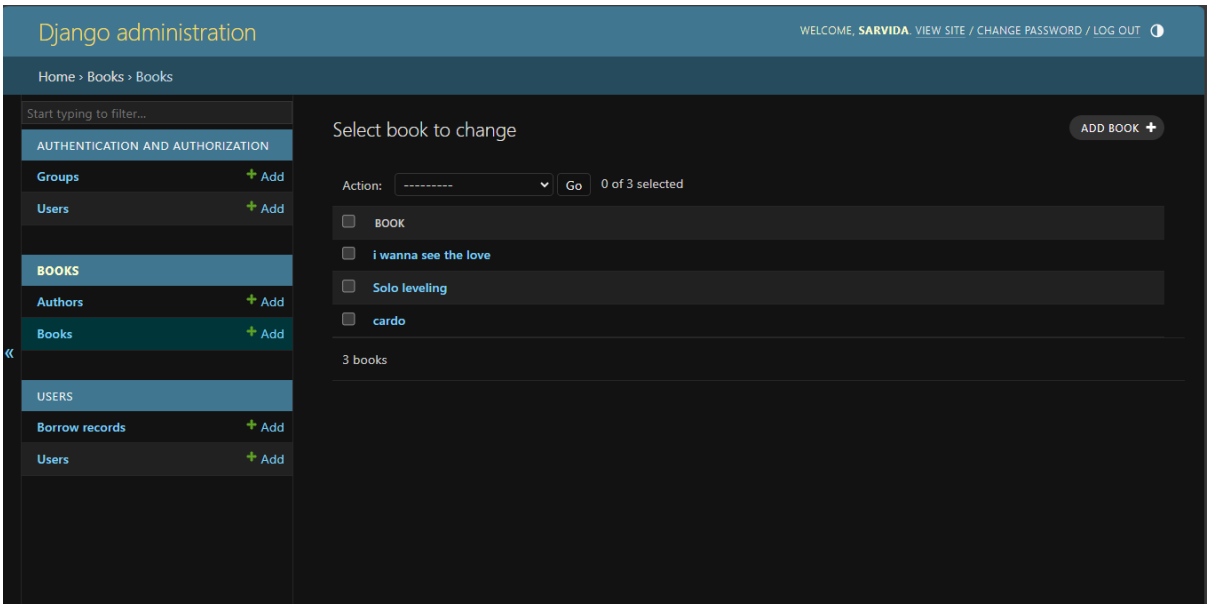
6. Register the templates directory to the Settings template directory



Django Program or Code:
Code for views, forms, and URL patterns has been provided above.

Results:
The user can now view the list of books, add new books, update existing books, and delete books.

The result:



My Github Link: https://github.com/sarvida20/My-Project-2025/tree/99101723ff519b2e3b5623c9ba1ca663ceab3fd7/library_system

Follow-Up Questions:

1. **How do Django forms work with models for CRUD operations?**
Answer: Django forms work with models using **ModelForms**, which automatically create form fields based on the model's attributes. This simplifies handling user input and validation. When a user submits a form, Django validates the data, and if it's valid, it saves it directly to the database. In CRUD operations:

Create: A form is displayed to enter new data, and upon submission, it creates a new record.

Read: Forms can be used to filter or search data.

Update: Forms can be pre-filled with existing data and updated upon submission.

Delete: Forms may confirm deletion before removing an entry.

2. **What is the role of the redirect() function in Django views?**
Answer: The redirect() function is used to send the user to another URL after an action is performed. In CRUD operations, after creating, updating, or deleting an item, redirect('book_list') ensures the user is taken back to the main book list page instead of staying on the form submission page. This improves user experience and prevents duplicate form submissions when refreshing the page.

Findings:
Students can perform CRUD operations for books and users.

Summary:
This activity allows students to implement and interact with the core database of the Library Management System.

Conclusion:
CRUD functionality is essential for any web application, and students now have the ability to manage books in the system.