**Laboratory Activity No. 2:**

**Laboratory Activity No. 2:**

**Topic belongs to**: **Software Design and Database Systems**

**Title**: *Designing the Database Schema for the Library Management System*

**Introduction**: In this activity, you will design the database schema for the Library Management System. The database will include tables for books, authors, users, and borrowing records. You will also learn how to use Django’s ORM (Object-Relational Mapping) to define the models.

**Objectives**:

Design the database schema for the Library Management System.

Create Django models to represent the schema.

Use Django’s ORM to interact with the database.

**Theory and Detailed Discussion**: Django uses an ORM (Object-Relational Mapping) system to map Python objects to database tables. By defining models in Python code, Django automatically creates the corresponding database tables. We will start by designing the database schema with the necessary relationships between entities like books, authors, and users.

**Materials, Software, and Libraries**:

**Django** framework

**SQLite** database (default in Django) **Time Frame**: 2 Hours

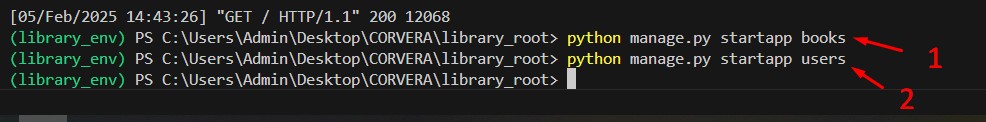
**Procedure**:

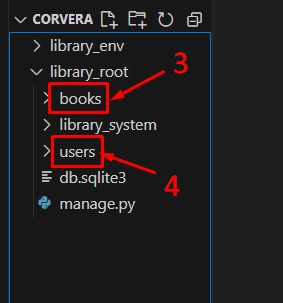
1. **Create Django Apps**:

In Django, an app is a module that handles a specific functionality. To keep things modular, we will create two apps: one for managing books and another for managing users.

Make sure that your are inside the *library\_root* directory

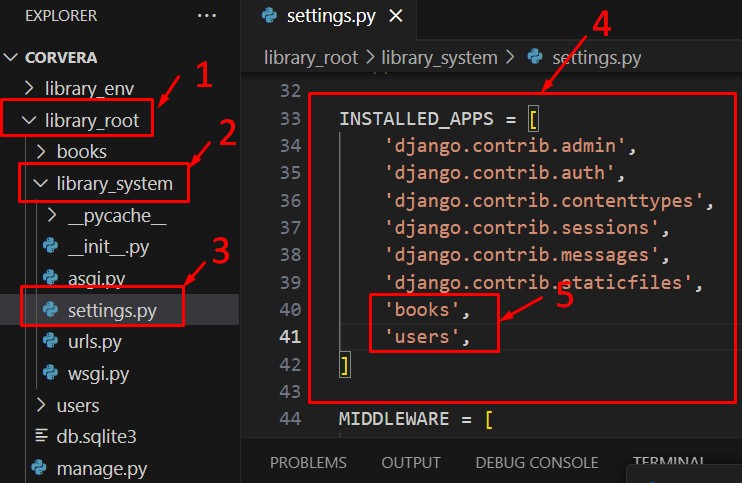
python manage.py startapp books python manage.py startapp users





1. **Register the Apps in Settings.py**

Under library\_root directory open library\_system>settings.py then add the books and users application under the INSTALLED\_APPS



1. **Define Models for the Books App**:

Open the books/models.py file and define the following models:

from django.db import models

class Author(models.Model):

name = models.CharField(max\_length=100) birth\_date = models.DateField()

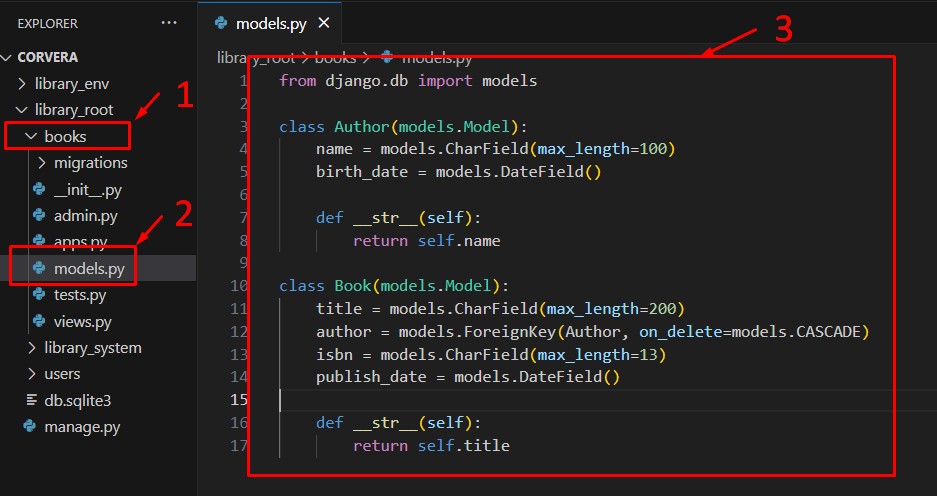
def \_\_str\_\_(self): return self.name

class Book(models.Model):

title = models.CharField(max\_length=200)

author = models.ForeignKey(Author, on\_delete=models.CASCADE) isbn = models.CharField(max\_length=13) publish\_date = models.DateField()

def \_\_str\_\_(self): return self.title



2. **Define Models for the Users App**:

Open the users/models.py file and define the following models:

from django.db import models from books.models import Book

class User(models.Model):

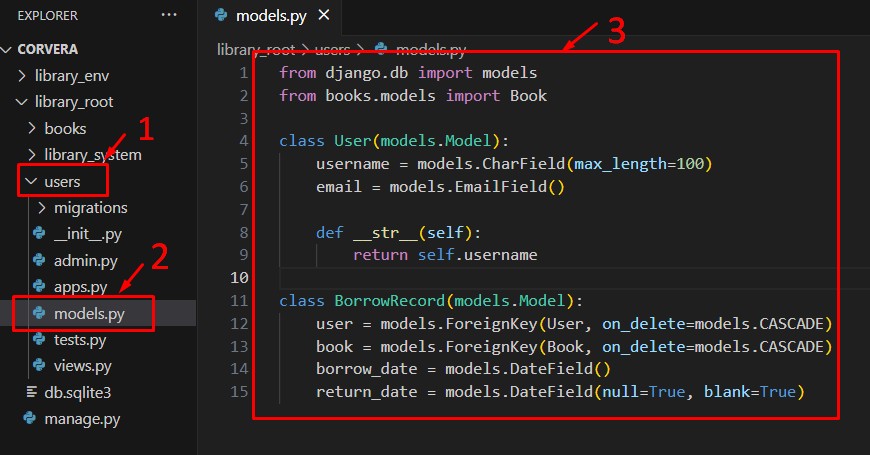
username = models.CharField(max\_length=100) email = models.EmailField()

def \_\_str\_\_(self): return self.username

class BorrowRecord(models.Model):

user = models.ForeignKey(User, on\_delete=models.CASCADE)

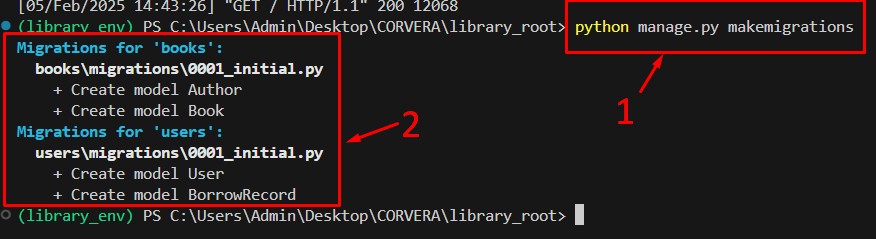
book = models.ForeignKey(Book, on\_delete=models.CASCADE) borrow\_date = models.DateField() return\_date = models.DateField(null=True, blank=True)



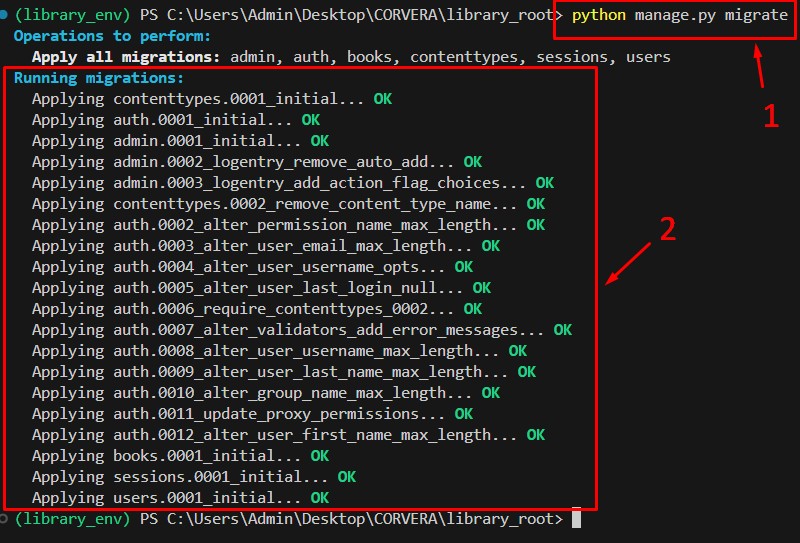
1. **Apply Migrations**:

To create the database tables based on the models, run the following commands:

python manage.py makemigrations

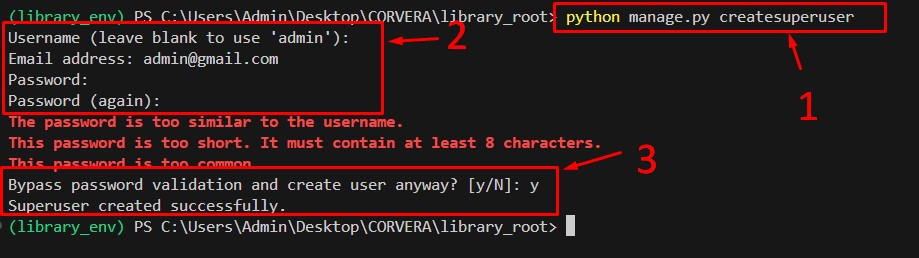


python manage.py migrate



1. **Create Superuser for Admin Panel**:

Create a superuser to access the Django admin panel: python manage.py createsuperuser Note: The password won’t show when you type it.

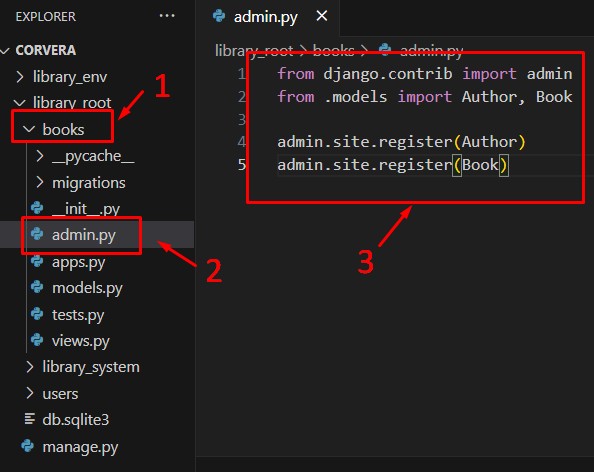


1. **Register Models in Admin Panel**:

In books/admin.py, register the Author and Book models:

from django.contrib import admin from .models import Author, Book

admin.site.register(Author) admin.site.register(Book)

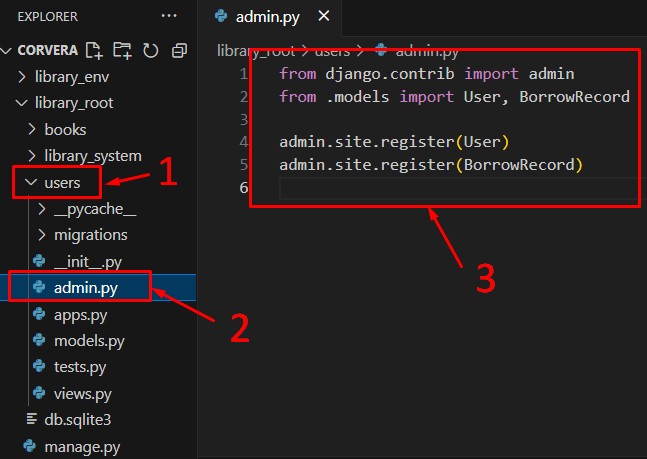


In users/admin.py, register the User and BorrowRecord models:

from django.contrib import admin

from .models import User, BorrowRecord

admin.site.register(User) admin.site.register(BorrowRecord)



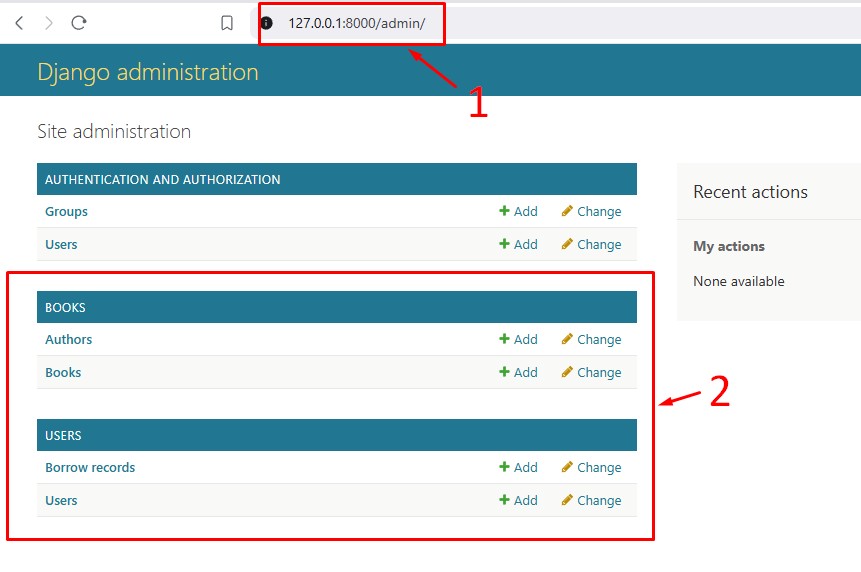
1. **Run the Development Server**:

Start the server again to access the Django admin panel:

python manage.py runserver

1. **Access Admin Panel**:

Open a browser and go to http://127.0.0.1:8000/admin and log in using the superuser credentials. You should see the Author, Book, User, and BorrowRecord models.



**Django Program or Code**: Write down the summary of the code for models that has been provided in this activity.

**Results**: By the end of this activity, you will have successfully defined the database schema using Django models, created the corresponding database tables, and registered the models in the admin panel. (print screen the result and provide the github link of your work)

**Follow-Up Questions**:

1. What is the purpose of using ForeignKey in Django models?

1. How does Django’s ORM simplify database interaction?

**Findings**:

**Summary**:

**Conclusion**: