**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

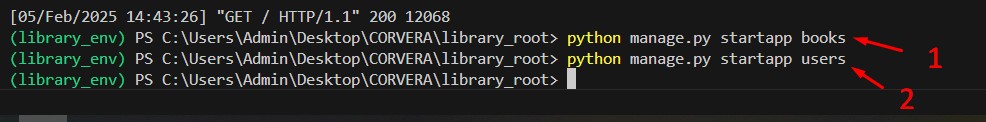
**Procedure**:

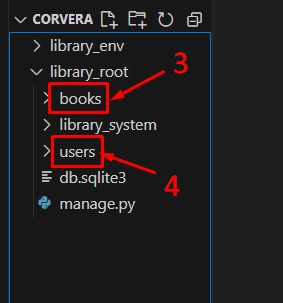
1. **Create Django Apps**:

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

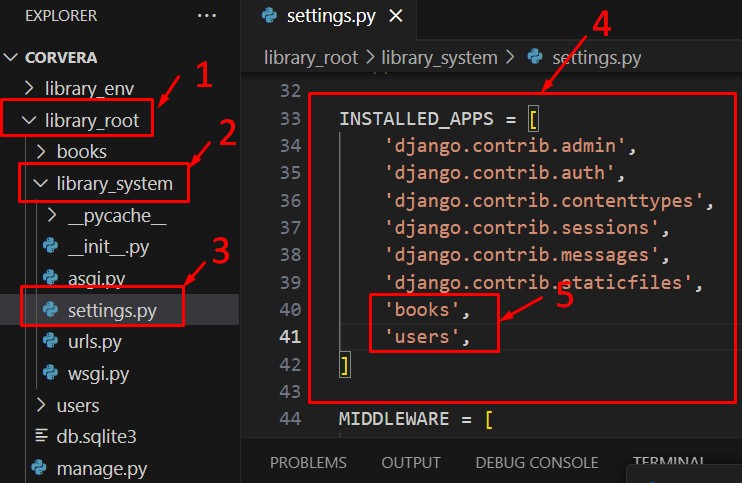
python manage.py startapp books python manage.py startapp users





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()

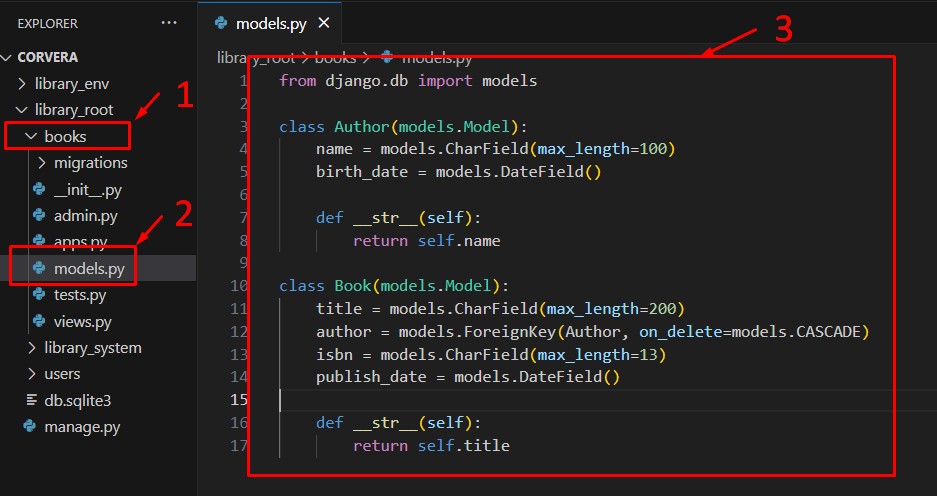
def \_\_str\_\_(self): return self.name

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):

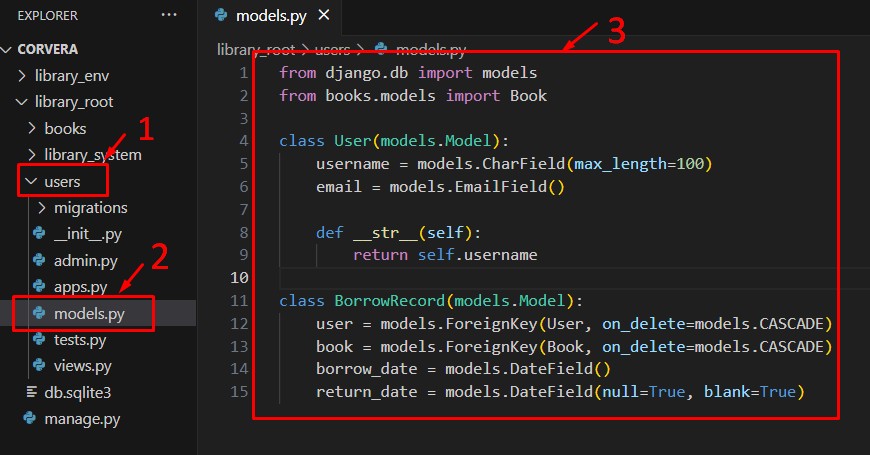
username = models.CharField(max\_length=100) email = models.EmailField()

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

class BorrowRecord(models.Model):

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

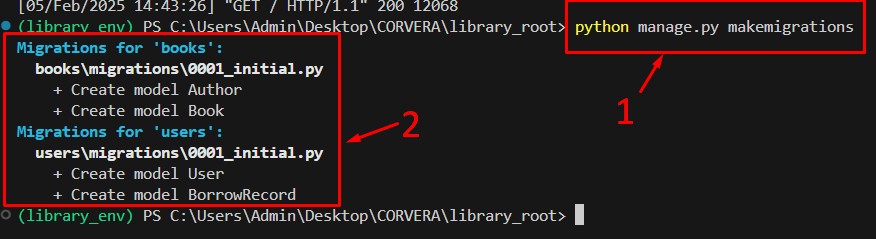
book = models.ForeignKey(Book, on\_delete=models.CASCADE) borrow\_date = models.DateField() return\_date = models.DateField(null=True, blank=True)



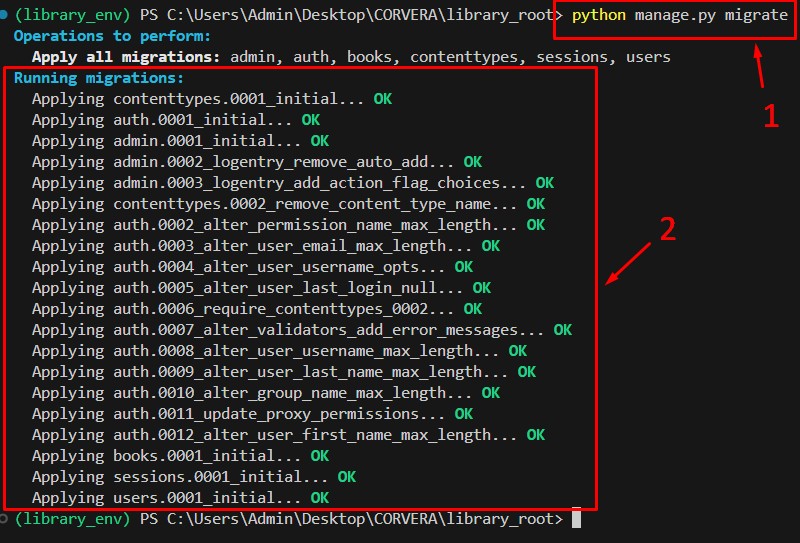
1. **Apply Migrations**:

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

python manage.py makemigrations

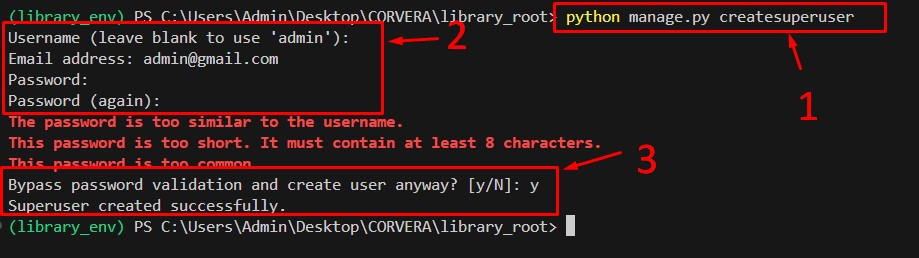


python manage.py migrate



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.

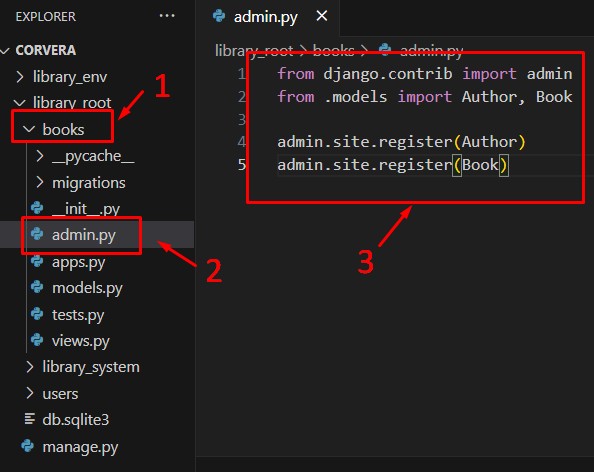


1. **Register Models in Admin Panel**:

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)

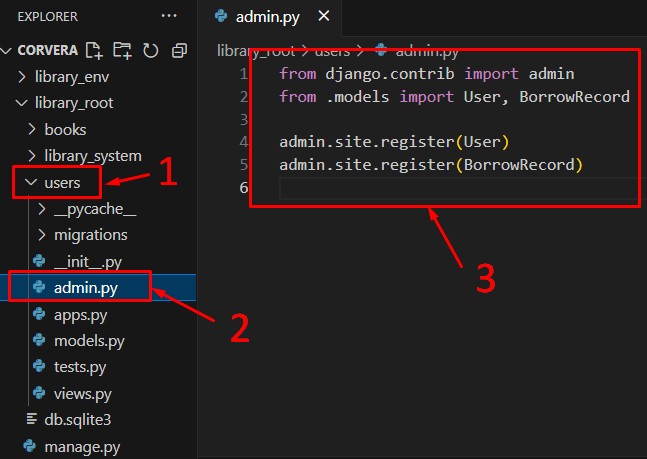


In users/admin.py, register the User and BorrowRecord models:

from django.contrib import admin

from .models import User, BorrowRecord

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



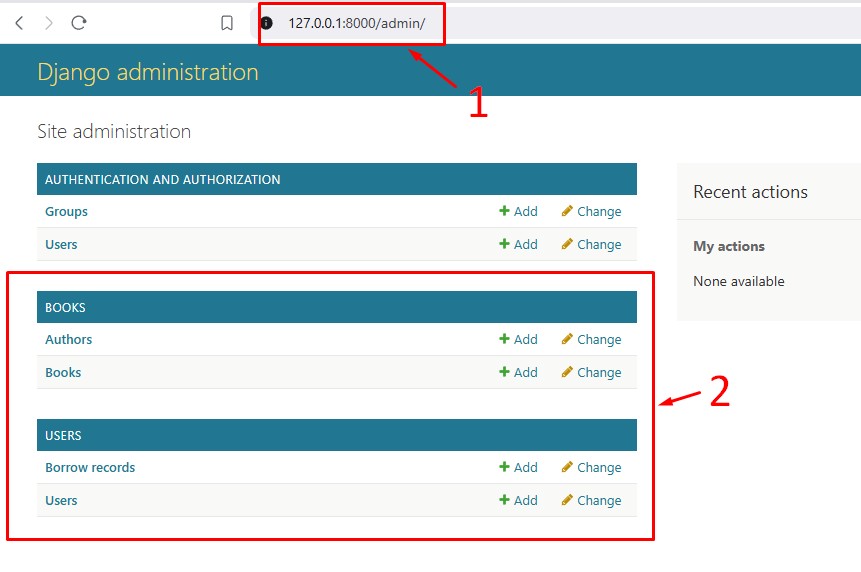
1. **Run the Development Server**:

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.

Summary of the code for models is that they act as classes and defined functions to create objects that are used as templates for various recorded blocks of information. Example books, authors, and users.

Then they create an extra class which is either used to attach one class’ information to another like how author class’ information are attached to book classes to allow the author of a book to be independently searched to show what other books they written, as well as records for users, allowing for a log-in function to be essentially made.

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?

A foreign key is essentially used when you want to connect one information to many other sources instead of one to one sources, which would use the OneToOne function.

An example of this in the laboratory assignment is that it allows you to connect the class author to a plentiful amount of book classes. So that you are capable of searching the author, and seeing the various books they have written.

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

Essentially, it allows you to create databases in native Python without going through a third party mediator like SQLDatabase to fix stuff up. Making it super easy to use as you essentially use object classes of python to create database entries in the first place.

Findings:

Generally, I have found out the server manager/admin side of creating websites with databases. From the general construction to databases, as well as creating debug functions like super users with default passwords and admin panels to allow for easier access to the website and its database in order to properly test it so you may find errors or general bug checking.

I also found out that migration is used to add changes from the model classes into the database properly.

Summary:

The summary of the code is that we have progressed from simply creating the framework with Django to allow a website to appear, and now we created the database itself and have it appear in a standard format within the website. One that can be easily edited on the UI and website itself, instead of directly in the terminal or code.

Said database uses object classes called Models to store blocks of information. Independent information that can use other functions like ForeignKeys to connect this information together. Like how you can connect account login information to Users, or Authors to their Books.

Conclusion:

In conclusion, we figured out how to create the basic database in Django without even any SQLDatabase stuff, using simply the model object classes and so on, while connecting them together in information. However, things like not being able to have OneToOne functions in order to ensure that Users can only be accessed with one set of proper account log in models, and various possible features seemingly absent. Means that it is likely we will figure out how to do more things as time passes on.