Jerick C. Jualo BSCpE – 2A
Laboratory Activity No. 1

Laboratory Activity No. 1:

Topic: **Introduction to Software Design, History, and Overview**

Title: *Setting Up the Development Environment for Django Project*

Introduction: This activity will guide you through the process of setting up your development environment to start building the Library Management System (LMS) in Django. The process involves installing necessary software, setting up Python and Django, and verifying the installation.

Objectives:

- Install Python and Django on your system.
- Create a virtual environment to manage dependencies.
- Verify the installation by running a simple Django project.

Theory and Detailed Discussion: To develop the Library Management System, we will use the Django framework. Django is a high-level Python web framework that allows developers to create robust web applications quickly and efficiently. Before we can start developing, we need to set up the development environment.

Materials, Software, and Libraries:

- **Python** (version 3.8 or above)
- **Django** (version 4.0 or above)
- **pip** (Python package manager)
- **Text Editor** (Visual Studio Code or PyCharm)
- **Database** (SQLite – comes with Django by default)

Time Frame: 1 Hour

Procedure:

1. **Install Python**:
   - Go to python.org and download the latest version of Python.
   - Install Python by following the installation instructions for your operating system.
2. **Install pip** (Python package installer):
   - Open a terminal and type the following command:

```
python -m ensurepip --upgrade
```

3. **Install Virtual Environment**:
   - Create a virtual environment for our project to avoid conflicts with global packages.

```
pip install virtualenv
```

   - Create a new virtual environment:

```
python -m venv library_env
```

   - Activate the virtual environment:
   - On Windows:

```
.\library_env\Scripts\activate
```

- On Mac/Linux:

```
source library_env/bin/activate
```

1. **Install Django**:
   - After activating the virtual environment, install Django by running:

```
pip install django
```

2. **Verify the Django Installation**:
   - Run the following command to verify if Django is installed:

```
django-admin --version
```

3. **Create a New Django Project**:
   - Create a new Django project called "library_system":

```
django-admin startproject library_system
```

   - Navigate into the project directory:

```
cd library_system
```

4. **Run the Django Development Server**:

- Start the development server to verify everything is working:

```
python manage.py runserver
```

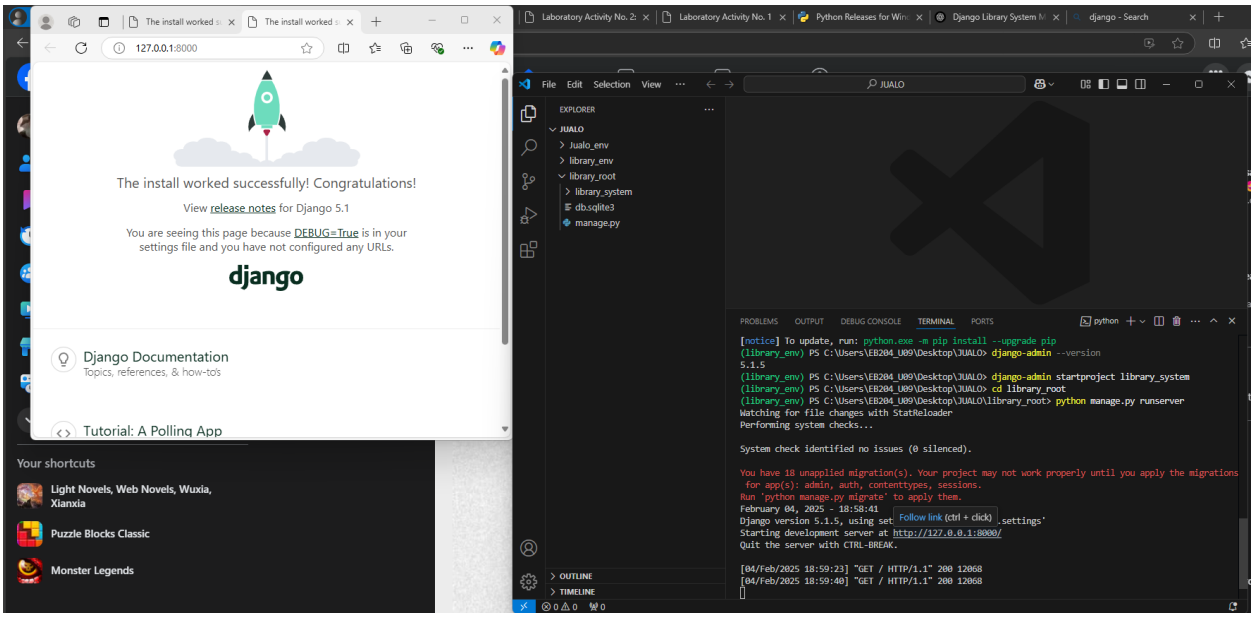
- Open a browser and go to `http://127.0.0.1:8000/`. You should see the Django welcome page.

Program/Code: The code here is focused on setting up the environment. The following commands should be run in the terminal:

```
python -m venv library_env
source library_env/bin/activate  # or
.\library_env\Scripts\activate on Windows
pip install django
django-admin startproject library_system
cd library_system
python manage.py runserver
```

Results: (print screen the result and provide the github link of your work)

RESULT:



Follow-Up Questions:

1. What is the role of a virtual environment in Django development?

   ANSWER:
   A virtual environment in Django development is an isolated workspace that allows creators to manage projects without interfering with other system.

1. What are the advantages of using Django for web development over other frameworks?

   ANSWER:
   Django allows users to build securable, clean and maintainable websites. It distinguishes itself among other frameworks because it is fast, includes tons of packages, it is secure, scalable as well as versatile.

Findings:

1. Successfully installed Python, pip, and Django on the system.
2. Created and activated a virtual environment to manage deficiencies efficiently.
3. Verified Django installation and successfully set up a new Django project.
4. The Development server ran without issues, confirming a proper setup.

Summary:

This activity focused on setting up a development environment for Django. It involved installing Python and Django, configuring a virtual environment, and verifying the setup by running a basic Django project. The environment setup is crucial for ensuring that the Library Management System project can be developed in a structured and organized manner.

Conclusion:

The activity successfully established the foundation for Django development. Setting up the environment correctly ensures smooth project execution and dependency management. Using a virtual environment provides isolation, preventing conflicts between packages. The successful installation and initial project setup indicate readiness for further development.

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

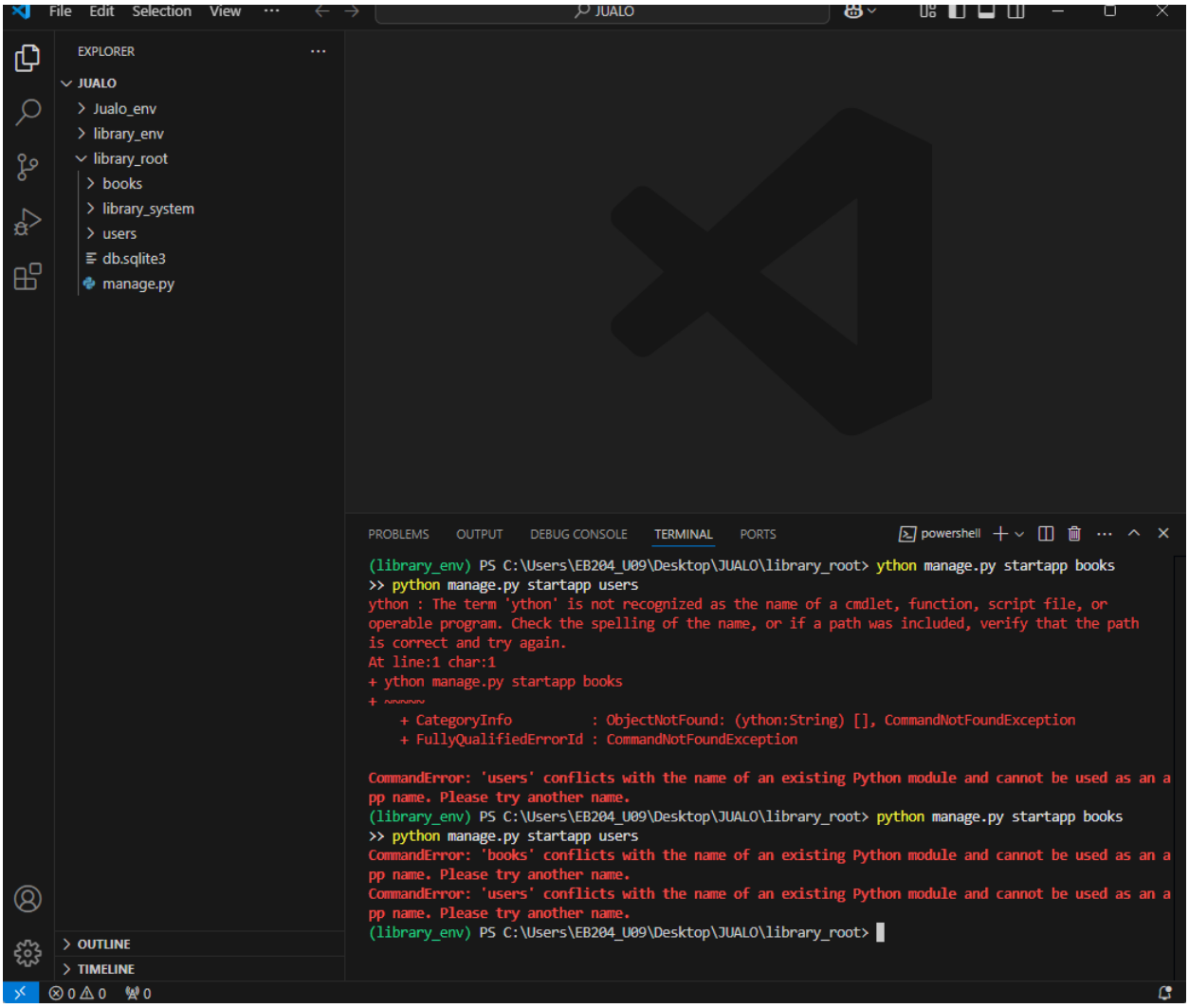
Procedure:

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.

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

RESULT:



2. **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)
```

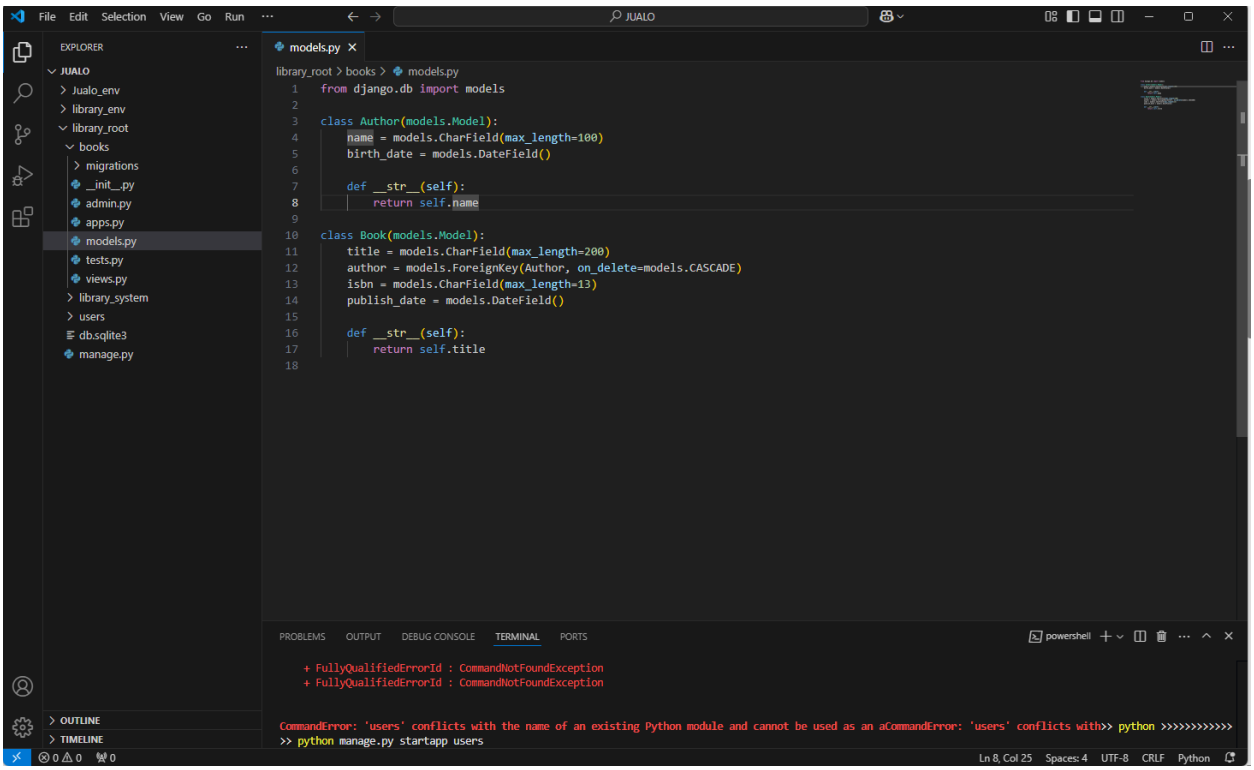
```
    birth_date = models.DateField()

    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
```

RESULT:



3. **Define Models for the Users App**:
   - Open the `users/models.py` file and define the following models:

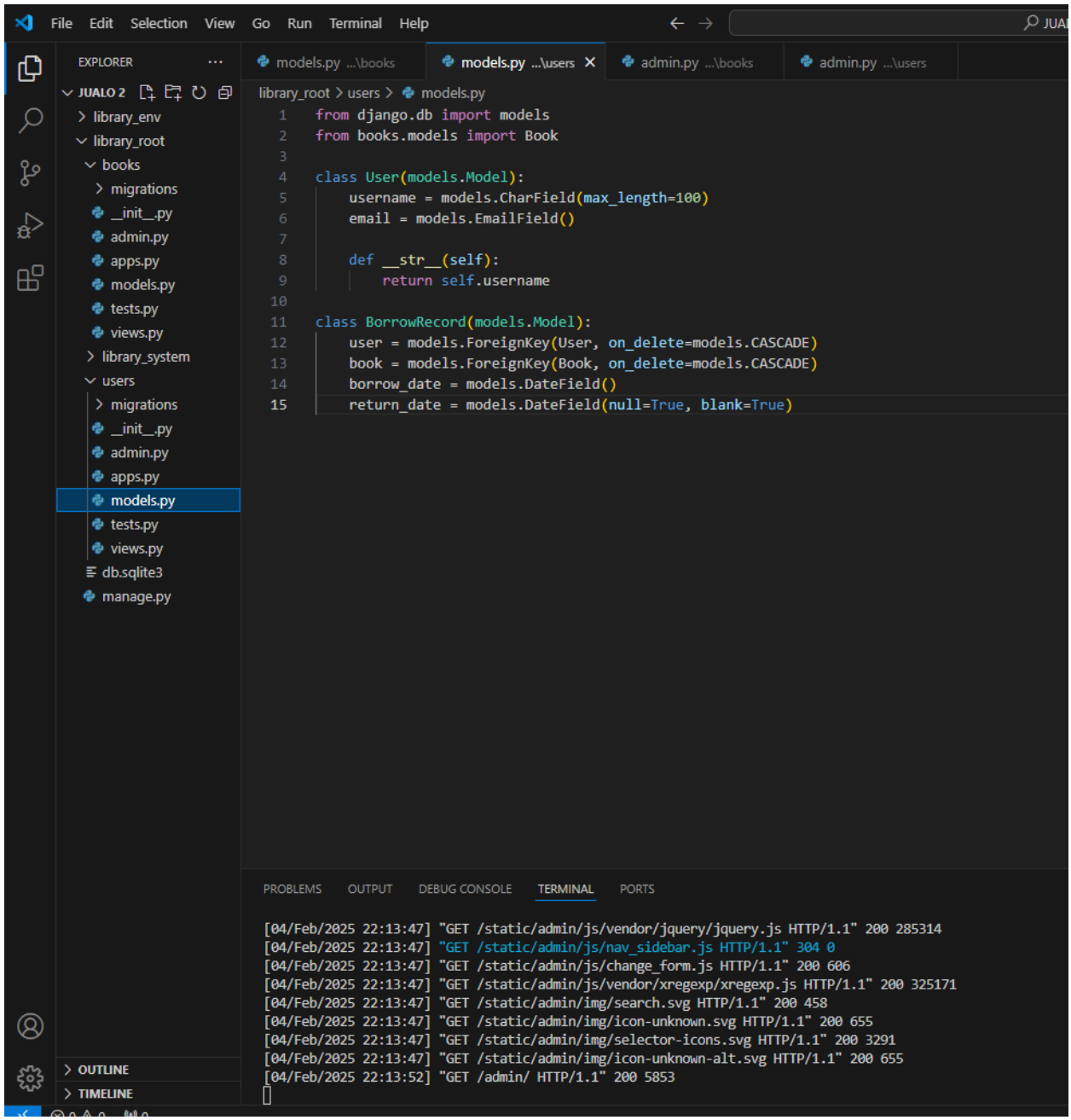
```
from django.db import models
from books.models import Book

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)
```

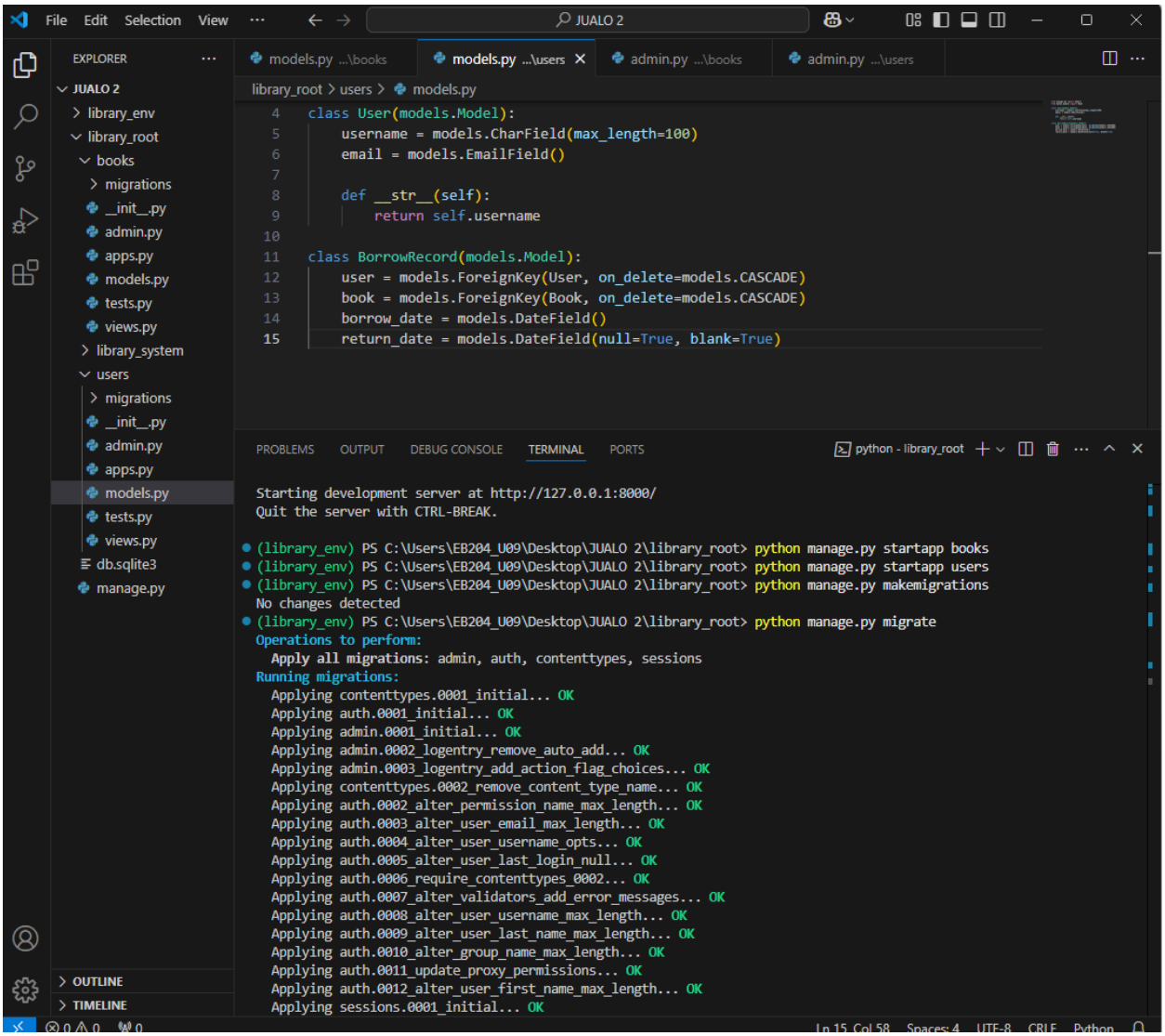
RESULT:



4. **Apply Migrations**:
   - To create the database tables based on the models, run the following commands:

```
python manage.py makemigrations
python manage.py migrate
```

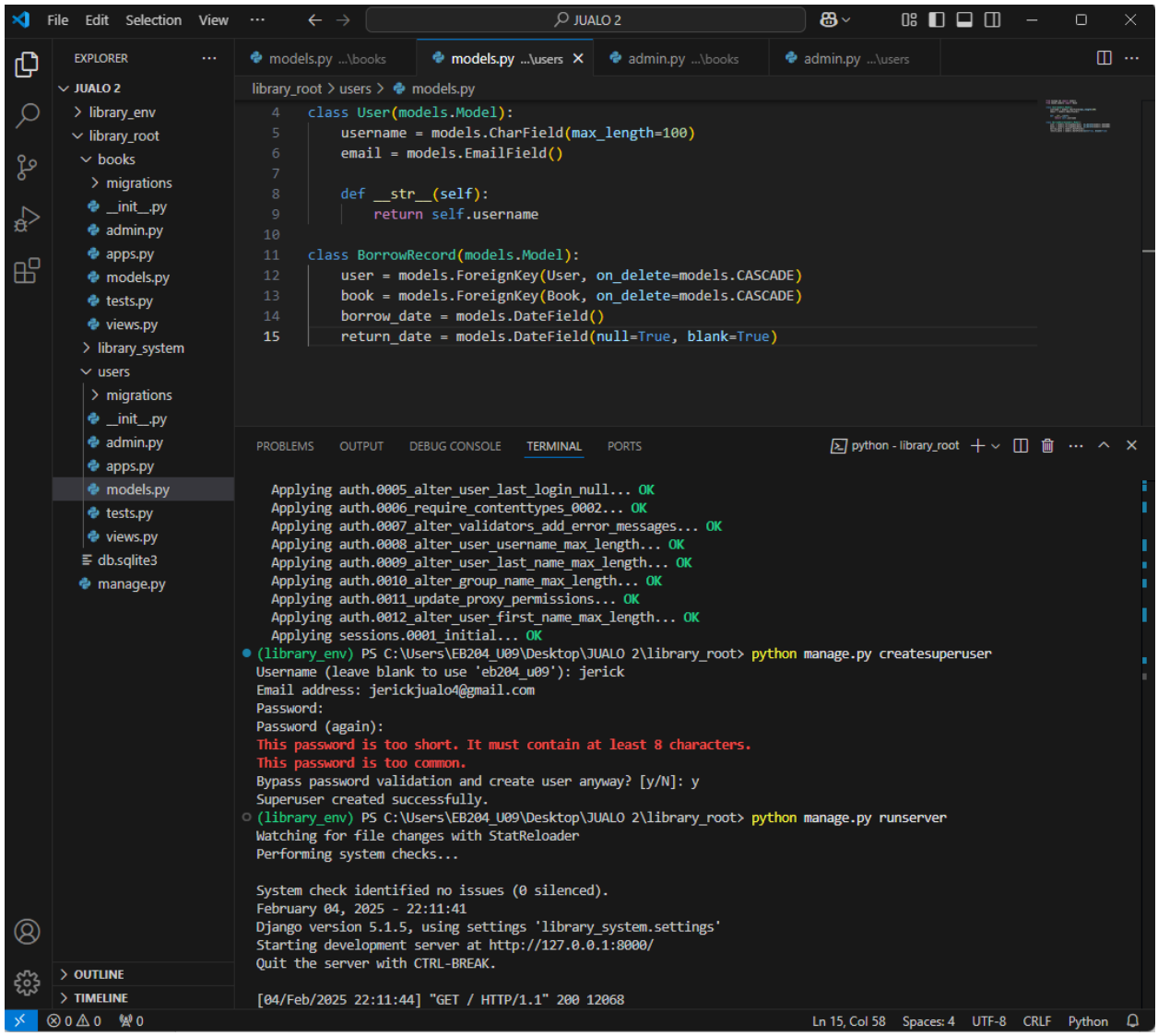
RESULT:



5. **Create Superuser for Admin Panel**:
   - Create a superuser to access the Django admin panel:

```
python manage.py createsuperuser
```

RESULT:



6.

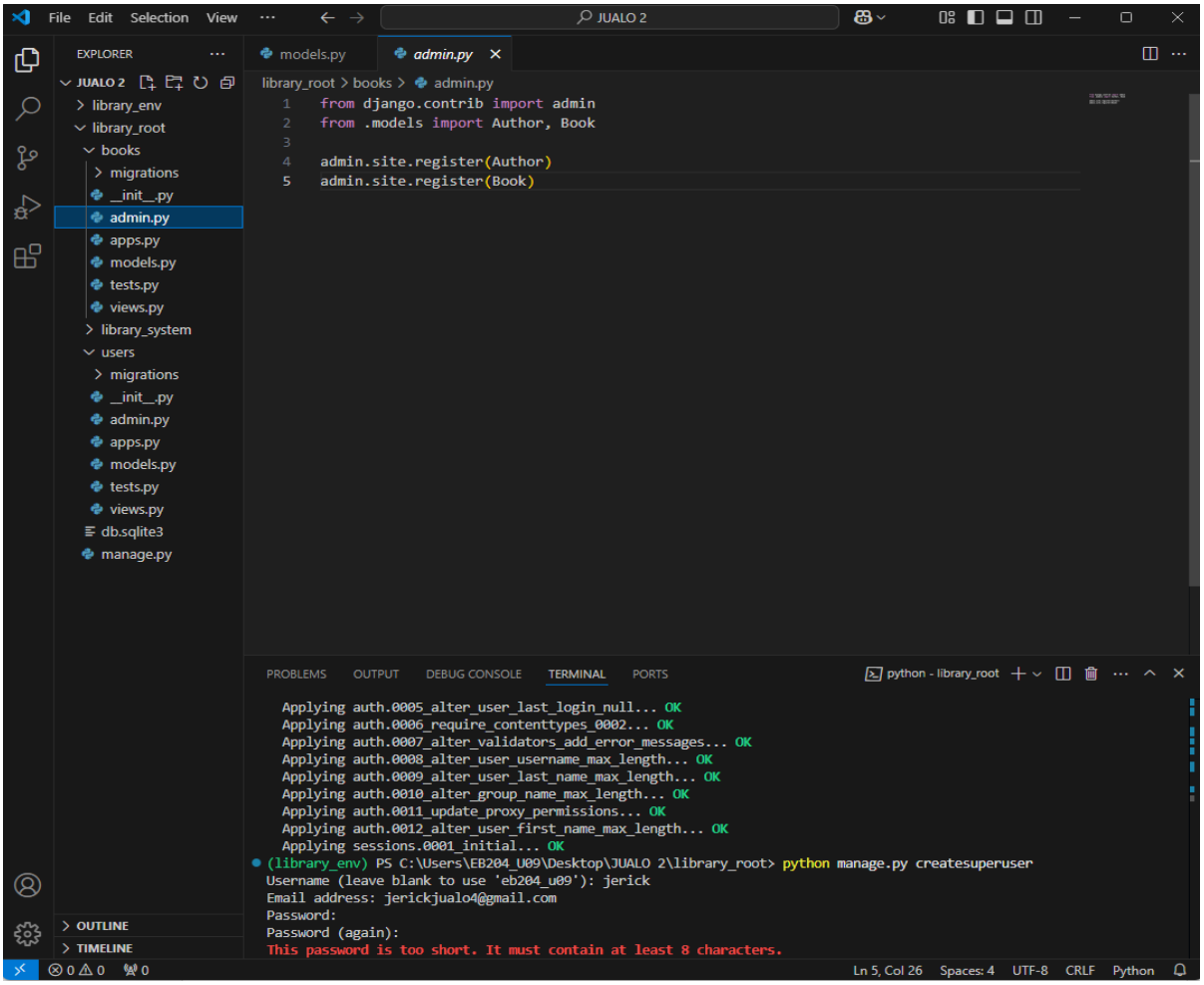
Register Models in Admin Panel:

- 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)
```

RESULT:

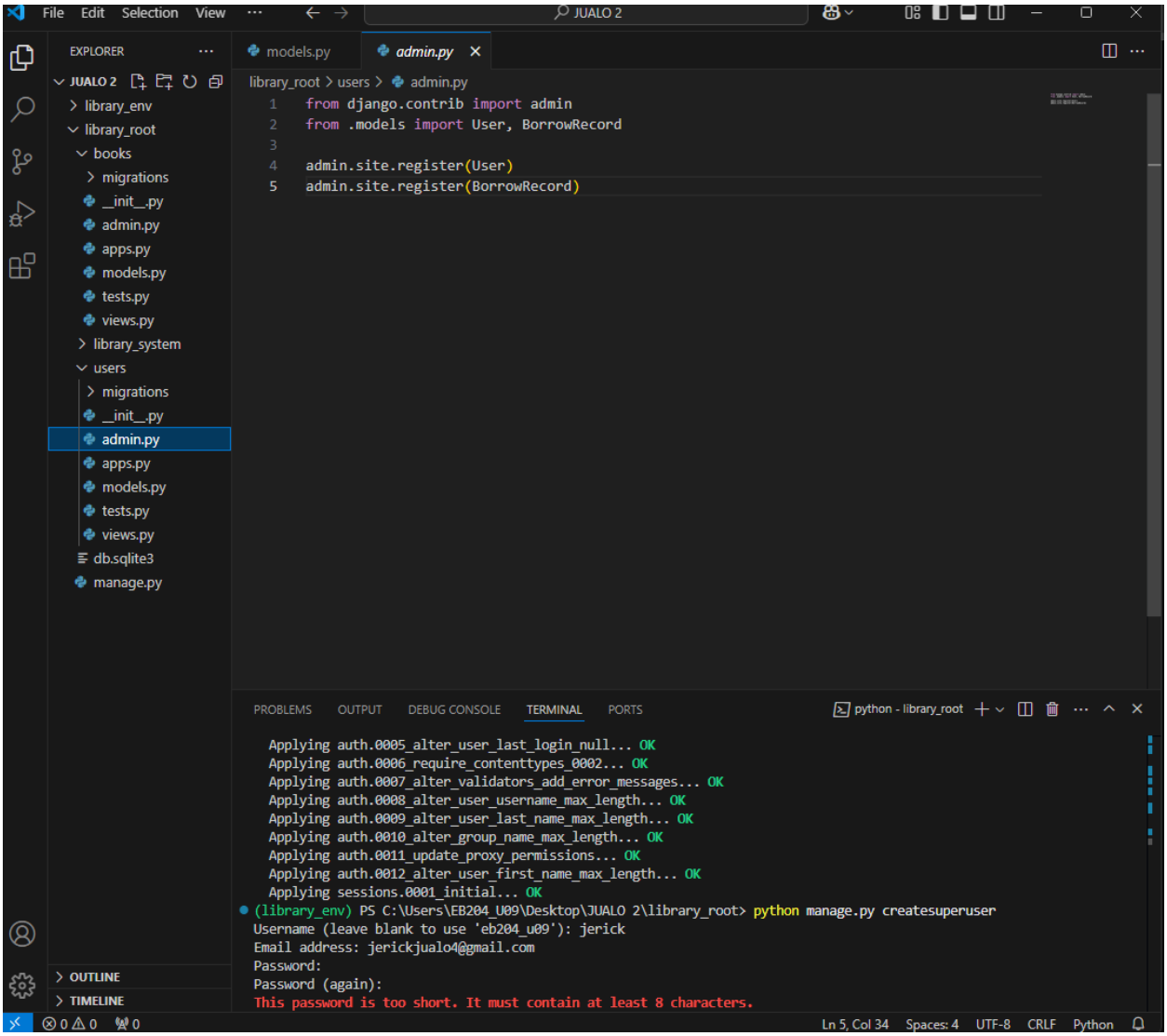


- In `users/admin.py`, register the `User` and `BorrowRecord` models:

```
from django.contrib import admin
from .models import User, BorrowRecord

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

RESULT:



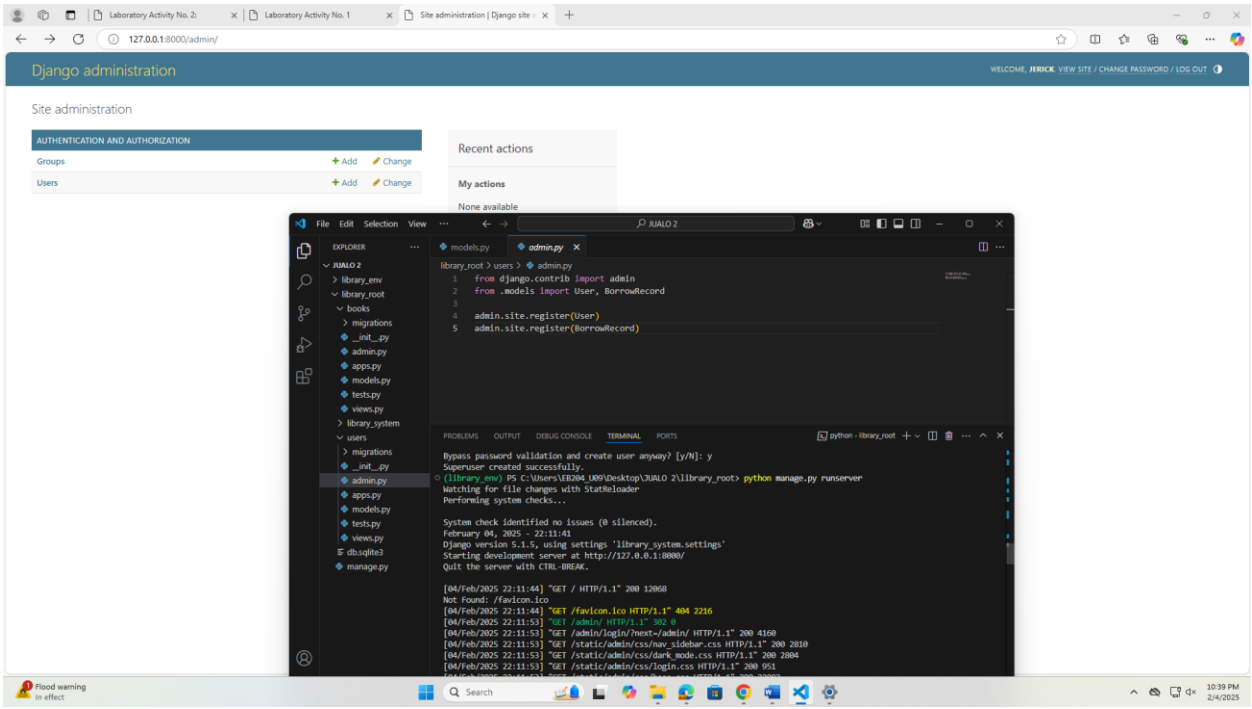
7. **Run the Development Server**:
   - Start the server again to access the Django admin panel:

```
python manage.py runserver
```

8. **Access Admin Panel**:

- 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.

RESULT FOR 7. AND 8. :



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

The Library Management System is designed using Django's ORM, consisting of two main apps: books and users.

Books App Models (books/models.py):

```
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
```

Users App Models (users/models.py):

```
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)
```

Applying Migrations:

To apply the database changes, use:

```
python manage.py makemigrations

python manage.py migrate
```

Registering Models in Admin Panel:

In books/admin.py:

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

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

In user/admin.py:

```
from django.contrib import admin
from .models import User, BorrowRecord

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

Run the Development Server:

```
python manage.py runserver
```

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?

   ANSWER:

   In Django models, a ForeignKey is used to establish a one-to-many relationship between two database tables.

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

   ANSWER:

   Django's ORM meaning Object-Relational Mapping means it simplifies database interaction by allowing developers to work with databases using Python objects instead of writing raw SQL queries.

Findings:

1. Designed a database schema with appropriate relationships using Django's ORM.
2. Defined models for books , authors, users, and borrowing records.
3. Created and migrated database tables based on the defined models.
4. Successfully registered models in the Django admin panel.
5. The Development server ran without issues, allowing access to the admin panel.

Summary:

This activity involved designing and implementing the database schema for the Library Management System using Django's ORM. The models for books, authors, users, and borrow records were defined, and relationships were established using ForeignKey. The database schema was then migrated and tested in the Django admin panel.

Conclusion:

The activity demonstrated the effectiveness of Django's ORM in simplifying database interactions. By defining models in Python, complex SQL queries were avoided. The database schema was successfully implemented, ensuring a well-structured foundation for managing books, users, and borrowing records in the Library Management System.