JAMES CARL E. BITUIN BSCPE_2A

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.

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

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

- 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)
```

- python manage.py makemigrations
- python manage.py migrate
- python manage.py createsuperuser
- from django.contrib import admin from .models import Author, Book

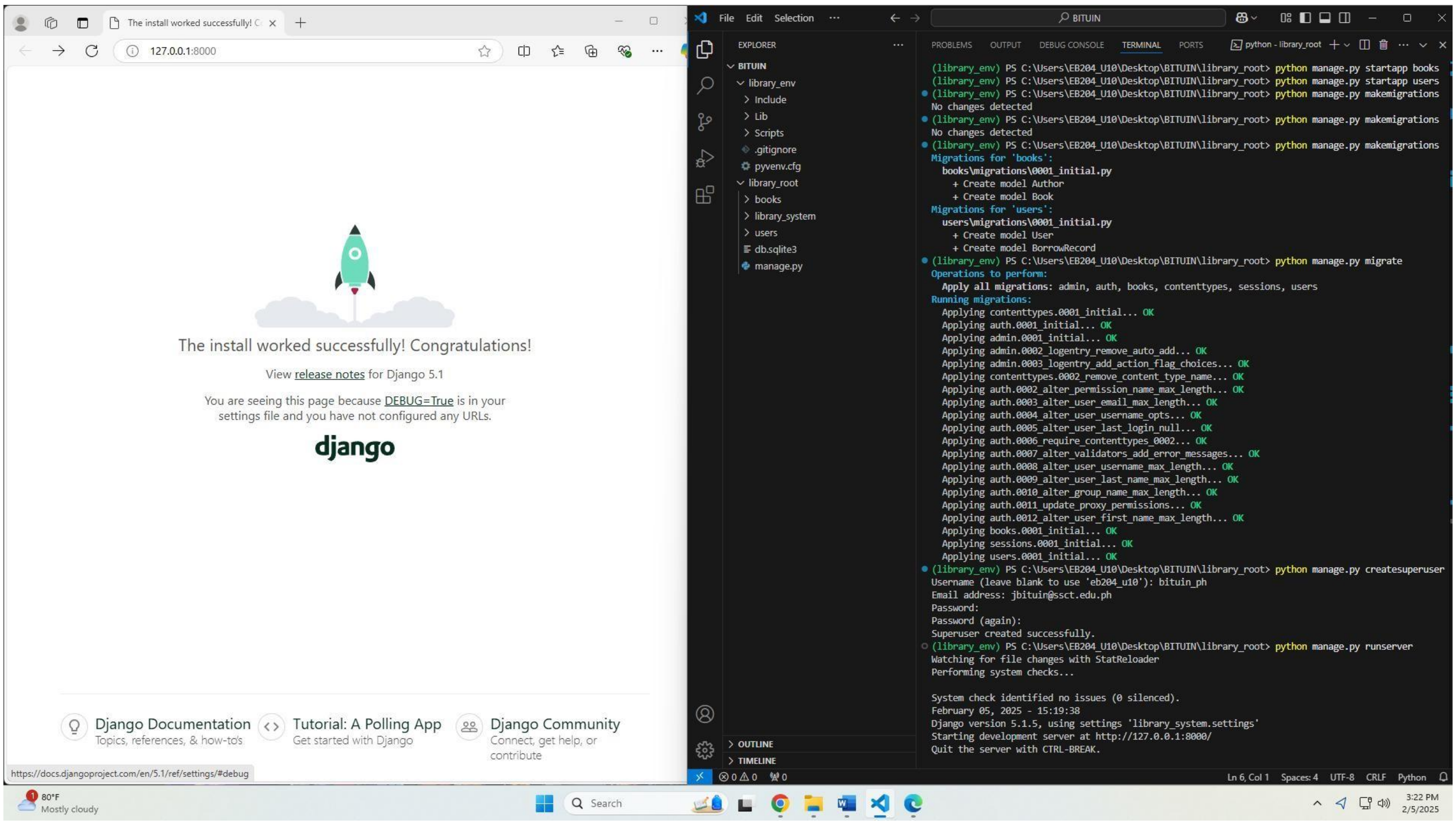
```
admin.site.register(Author) admin.site.register(Book)
```

- from django.contrib import admin

```
from .models import User, BorrowRecord
```

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

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)


The install worked successfully! C
127.0.0.1:8000
The install worked successfully! Congratulations!
View release notes for Django 5.1
You are seeing this page because DEBUG=True is in your settings file and you have not configured any URLs.
django
Django Documentation
Topics, references, & how-to's
Tutorial: A Polling App
Get started with Django
Django Community
Connect, get help, or contribute
https://docs.djangoproject.com/en/5.1/ref/settings/#debug
File Edit Selection
BITUIN
EXPLORER
BITUIN
library_env
Include
Lib
Scripts
.gitignore
pyvenv.cfg
library_root
books
library_system
users
db.sqlite3
manage.py
OUTLINE
TIMELINE
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
python - library_root
Ln 6, Col 1 Spaces: 4 UTF-8 CRLF Python
80°F
Mostly cloudy
Search
3:22 PM
2/5/2025

```
(library_env) PS C:\Users\EB204_U10\Desktop\BITUIN\library_root> python manage.py startapp books
(library_env) PS C:\Users\EB204_U10\Desktop\BITUIN\library_root> python manage.py startapp users
(library_env) PS C:\Users\EB204_U10\Desktop\BITUIN\library_root> python manage.py makemigrations
No changes detected
(library_env) PS C:\Users\EB204_U10\Desktop\BITUIN\library_root> python manage.py makemigrations
No changes detected
(library_env) PS C:\Users\EB204_U10\Desktop\BITUIN\library_root> python manage.py makemigrations
Migrations for 'books':
  books\migrations\0001_initial.py
    + Create model Author
    + Create model Book
Migrations for 'users':
  users\migrations\0001_initial.py
    + Create model User
    + Create model BorrowRecord
(library_env) PS C:\Users\EB204_U10\Desktop\BITUIN\library_root> python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, books, contenttypes, sessions, users
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying admin.0002_logentry_remove_auto_add... OK
  Applying admin.0003_logentry_add_action_flag_choices... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0002_alter_permission_name_max_length... OK
  Applying auth.0003_alter_user_email_max_length... OK
  Applying auth.0004_alter_user_username_opts... OK
  Applying auth.0005_alter_user_last_login_null... OK
  Applying auth.0006_require_contenttypes_0002... OK
  Applying auth.0007_alter_validators_add_error_messages... OK
  Applying auth.0008_alter_user_username_max_length... OK
  Applying auth.0009_alter_user_last_name_max_length... OK
  Applying auth.0010_alter_group_name_max_length... OK
  Applying auth.0011_update_proxy_permissions... OK
  Applying auth.0012_alter_user_first_name_max_length... OK
  Applying books.0001_initial... OK
  Applying sessions.0001_initial... OK
  Applying users.0001_initial... OK
(library_env) PS C:\Users\EB204_U10\Desktop\BITUIN\library_root> python manage.py createsuperuser
Username (leave blank to use 'eb204_u10'): bituin_ph
Email address: jbituin@ssct.edu.ph
Password:
Password (again):
Superuser created successfully.
(library_env) PS C:\Users\EB204_U10\Desktop\BITUIN\library_root> python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
February 05, 2025 - 15:19:38
Django version 5.1.5, using settings 'library_system.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

Follow-Up Questions:

Follow-Up Questions:

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

Answer:

A ForeignKey in Django models is used to create a **one-to-many relationship** between two models. It allows one model to reference another, establishing a link between the two. For example, if you have a Book model and an Author model, each book can have one author, but an author can have many books. The ForeignKey field is placed in the Book model, linking each book to a specific author.

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

Answer:

Django's ORM (Object-Relational Mapping) allows developers to interact with the database using **Python code** instead of writing raw SQL queries. It translates Python objects into database tables and vice versa, making it easier to:

- **Query**: Retrieve data using Python objects (Book.objects.all() instead of SELECT * FROM books).
- **Create, Update, Delete**: Easily create and modify records (e.g., book.save() instead of INSERT INTO books).
- **Maintain Relationships**: Manage relationships between models (like ForeignKey) with simple attributes rather than complex SQL joins.