Daryl D. Silva

BSCpE-4B2

Laboratory Activity No. 2:

Topic belongs to: Software Design and Database Systems

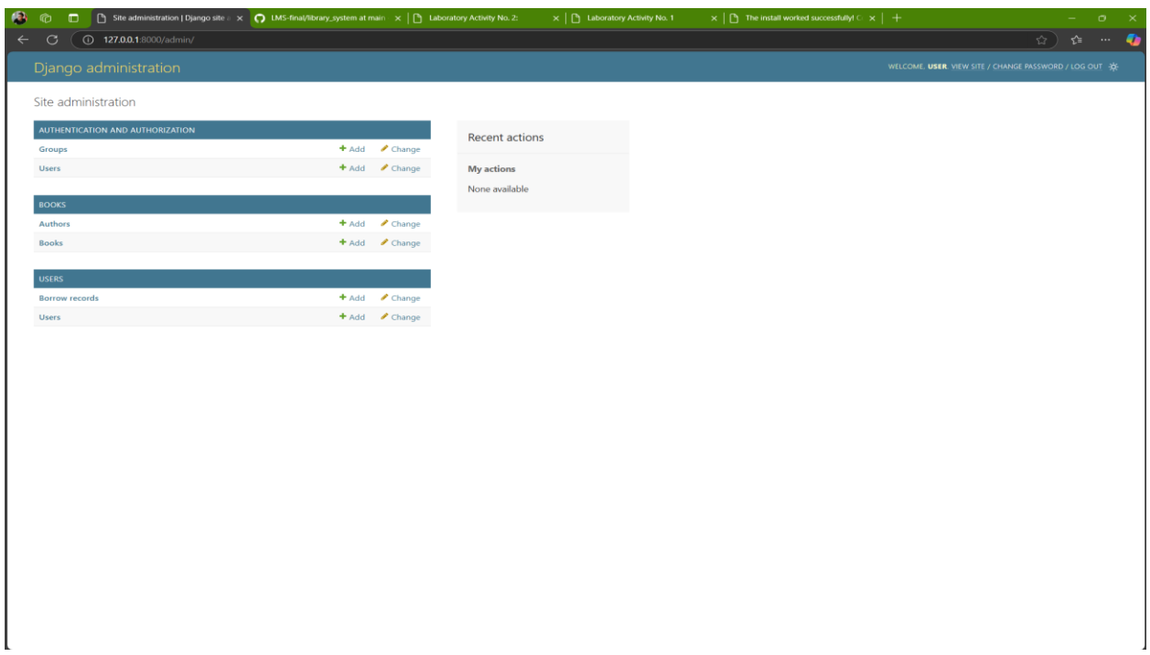
Title: *Designing the Database Schema for the Library Management System*

1. Django Program or Code: Write down the summary of the code for models that has been provided in this activity.
   Answer: The Django models define the database schema for the Library Management System. These models represent different entities such as Books, Authors, Users, and Borrow Records.

   - **Book Model** – Stores book details, including title, author, ISBN, publication date, and availability status.
   - **Author Model** – Maintains author information such as name and biography.
   - **User Model** – Manages library users, extending Django's built-in User model to store additional details like membership type.
   - **Borrow Record Model** – Tracks book borrowing history, linking users to borrowed books along with issue and return dates.

   These models utilize Django's Object-Relational Mapping (ORM) to facilitate easy database interactions. Proper relationships such as One-to-Many (Author to Books) and Many-to-Many (Users to Books) ensure efficient data management

2. 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)



Link: [LMS-final/library_system at main · Darylsilva500/LMS-final]

Follow-Up Questions:

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

Answer: A ForeignKey in Django models establishes a one-to-many relationship between two tables. It connects records in one model to multiple records in another, ensuring data integrity and enforcing referential constraints. For example, in a Library Management System, a Book model can have a ForeignKey linking to an Author model, meaning each book is associated with a single author, but an author can have multiple books.

2. How does Django's ORM simplify database interaction?
   Answer: Django's Object-Relational Mapping (ORM) allows developers to interact with the database using Python code instead of raw SQL queries. It simplifies tasks like:

   - Creating, retrieving, updating, and deleting records using Python objects.
   - Automatic schema management with migrations.
   - Efficient query handling through filters and model relationships.
   - Database abstraction, allowing easy switching between different database backends.

Findings:

- A well-structured database schema is crucial for efficient data storage and retrieval in a Library Management System.
- Key entities in the system include Books, Authors, Users, and Borrow Records, each represented as a model in Django.
- Relationships such as One-to-Many (Author to Books) and Many-to-Many (Users to Books) ensure data consistency and organization.
- The use of ForeignKey fields establishes links between models, enforcing referential integrity.
- Django's ORM simplifies database interactions, allowing seamless management of data without requiring raw SQL queries.

Summary:

The database schema for the Library Management System is designed to efficiently store and manage information about books, users, and borrowing records. The schema uses Django models with appropriate relationships to ensure data integrity and optimized queries. With Django's ORM, developers can easily interact with the database, reducing complexity and improving maintainability.

Conclusion:

A well-designed database schema is essential for the smooth operation of a Library Management System. Proper use of Django models and relationships ensures scalable and organized data management. By leveraging Django's ORM, developers can efficiently implement and maintain the system, ensuring a seamless experience for users.