

Task 3: Django Models, ORM, and Database Relationships

Objective

This task focuses on learning how Django stores and retrieves data using **models** and the **ORM (Object Relational Mapper)**. You will understand how to define models, apply migrations, and query data. By the end of this task, you should be able to build a small Django app that creates, stores, and displays data from a database.

Questions to Explore

Database:

- **What is Relational Database? (for the course we do need to know sql syntax just the whole idea how does relational database work)**

Models and ORM

1. What is a model in Django, and how is it connected to the database?
2. What is the role of migrations in Django?
3. How do you create a model class in Django? Provide an example.
4. What is the difference between makemigrations and migrate?
5. What is the Django ORM, and how does it differ from writing raw SQL queries?
6. How do you perform basic queries with the ORM? For example:
 - o Create a new object and save it.
 - o Retrieve all objects.
 - o Filter objects by a condition.
 - o Update an object.
 - o Delete an object.

Relationships

7. What are the three types of relationships in Django models?
 - o One-to-One (why not just use one model?)
 - o One-to-Many (ForeignKey)

- Many-to-Many (what happened in behind)
- 8. Provide an example of when you would use each relationship.
- 9. How do you define relationships in Django models? Show code snippets.

Admin and Data Management

- 10. What is Django Admin?
- 11. How do you register models in the Django admin site?
- 12. How do you create a superuser and log in to the Django admin?

Forms

- 9. What is a Django Form, and how does it differ from raw HTML forms?
- 10. How do you create a forms.py file and connect it to a view?
- 11. What are the advantages of Django forms (e.g., validation, security, reusability)?
- 12. What's the difference between **ModelForm** and **Form** in Django? When would you use each?

Deliverables

You will implement a small Django app that manages **Developers** and **Projects**.

Requirements

- 1. **Model Setup**
 - Create a Developer model with fields: first_name, last_name, email, and age.
 - Create a Project model with fields: title, description.
 - Create skill model with field: title, description
 - Define a **Many-to-Many** relationship between developers and projects
 - Define a **Many-to-One** relationship between skills and developer
- 2. **Database & Admin**
 - Run makemigrations and migrate to create database tables.

- Register both models in the Django admin and verify you can add students and courses.

3. Views & Templates

- Create a view `/developers/` that lists all developers and their skill tags.
- Create a view `/projects/` that lists all with name of their developers.
- Create New Developer using Form
- Create Project form which should assign to one or more developer using form

References

- Django Models: <https://docs.djangoproject.com/en/5.0/topics/db/models/>
- Django ORM Queries: <https://docs.djangoproject.com/en/5.0/topics/db/queries/>
- Django Admin: <https://docs.djangoproject.com/en/5.0/ref/contrib/admin/>
- W3Schools Models: https://www.w3schools.com/django/django_models.php
- Previous Course you chose for Django (ORM section)