Exercise 2.3: Django Models

Learning Goals

- Discuss Django models, the “M” part of Django’s MVT architecture
- Create apps and models representing different parts of your web application
- Write and run automated tests

Reflection Questions

- Do some research on Django models. In your own words, write down how Django models work and what their benefits are.

In Django, a model is a Python class that defines the structure of your data, with each model corresponding to a database table. Models have fields that specify the data types, such as CharField for text and IntegerField for numbers. When you create or modify a model, you use Django's migration system to apply those changes to the database, keeping the schema in sync. Django's powerful ORM allows you to interact with the database using Python code, making it easy to perform CRUD operations. Additionally, models include built-in validation to ensure data conforms to specified formats before being saved.

The benefits of using Django models include abstraction, which simplifies database management and allows developers to focus on application logic. Models promote reusability across the application, following the DRY (Don't Repeat Yourself) principle, and come with an automatic admin interface for easy management. They integrate seamlessly with other Django features, enhancing web application development, and help maintain data integrity through built-in validation and constraints that prevent invalid data from being saved.

- In your own words, explain why it is crucial to write test cases from the beginning of a project. You can take an example project to explain your answer.

It ensures code quality by verifying features like the shopping cart, allowing developers to identify and fix bugs early. This practice facilitates safe refactoring, promotes better design, and reduces debugging time by quickly pinpointing issues. Tests also serve as documentation, helping team members understand expected code behavior, and enable collaboration by allowing multiple developers to work on features simultaneously without conflicts. Ultimately, thorough testing enhances user experience, leading to a more stable and reliable product that keeps customers returning. Prioritizing testing from the start enables teams to deliver robust applications.