Basics of Dynamic Web Pages

- Dynamic web pages are pages that can change content based on user interactions or other variables. Unlike static web pages, which are fixed and display the same content to every user, dynamic web pages can display different content to different users or at different times.
- In Django, dynamic web pages are created using templates, which are HTML files that can include placeholders for dynamic content. These placeholders are filled in with data from the server before the page is sent to the client's browser.

Dynamic Content:

- Dynamic content in Django is typically generated using the Django template language (DTL). DTL allows you to embed Python-like code in your HTML templates to dynamically generate content.
- Dynamic content can also be generated using Django's views, which are Python functions that process requests and return responses. Views can query databases, interact with other web services, or perform other tasks to generate dynamic content.

Mapping URLs to views:

- URL mapping in Django is done using the `urls.py` file in each app. This file contains a list of URL patterns and their corresponding view functions.
- Each URL pattern is defined using a regular expression and is associated with a specific view function. When a request is made to a URL, Django matches the URL to a pattern in the `urls.py` file and calls the corresponding view function to generate the response.

Request Processing by Django:

- When a request is made to a Django web application, Django follows a predefined process to handle the request and generate a response.
- The request passes through **middleware**, which can modify or intercept the request before it reaches the view function.
- The URL dispatcher then matches the request to a view function based on the requested URL.
- The view function processes the request, typically by querying a database or performing some other action to generate the response.
- Finally, the response is sent back to the client's browser.

An Overview of the Settings File in Django:

- The settings file in Django (`settings.py`) contains configuration settings for the Django project. These settings include database configuration, static files configuration, middleware settings, and more.

- The settings file is used to customize the behavior of the Django project and to store sensitive information such as secret keys and database passwords.

Pretty Error Pages:

- Django provides a built-in mechanism for customizing error pages, such as the 404 (Page Not Found) and 500 (Server Error) pages.
- You can create custom templates for these error pages and configure Django to use them by defining `handler404` and `handler500` views in your `urls.py` file.