1. INTRODUCTION TO FRONT END

* Html
* CSS
* Javascript
* Bootstrap

2. INTRODUCTION TO DJANGO

* Features of Django
* Django web server
* Understanding Django environment
* A simple ‘Hello world’ application

3. DISPLAYING HYPERLINKS - PROJECT

* Django architecture
* MVC and MTV
* Starting a project
* Django apps
* Activating our first app
* A view that displays a hyperlink
* Mapping the views to URLs
* Running our first app
* Improving the views using templates
* The improved templates
* Template inheritance
* Sending data from url to view
* Sending data from view to template

4. CREATING A WEBSITE - PROJECT

* Starting a project
* Creating an app inside the project
* Activating the app
* Creating model for our site
* Converting the model into a table
* Examples for Fields in Models
* Basic data access using Django shell
* Saving objects into database
* Retrieving objects from database
* Modifying objects of database
* Sorting objects
* Filtering objects
* Deleting objects
* Making changes in the data model

5. CREATING ADMINISTRATION PANEL

* Using the admin interface
* Customizing the admin interface
* Adding users
* Data access and modification using admin panel
* Giving permissions to users

6. CREATING FIRST PAGE OF OUR SITE

* The Django template system
* Template Inheritance
* Improving the website
* Adding background color for web pages
* Adding banner to the web site
* Adding background image in the web pages
* Storing and displaying images
* Adding users to our site
* Uploading the information by user
* Changing the administration interface title

7. DJANGO FORMS CREATION

* Forms basics
* Creating Contact Us form
* Form field examples

8. DJANGO's EMAIL FUNCTIONALITY

* Configuring email settings
* Sending emails with Django

9. DJANGO TEMPLATE LANGUAGE

* Django template tags
* If/else
* If/elif/else
* For
* Comments
* Filters
* Using templates to display data in the site

10. INTEGRATING BOOTSTRAP INTO DJANGO

* Introduction to Bootstrap
* Creating tables
* Creating grids
* Creating carousels

11. SESSIONS AND COOKIES

* Difference between session and cookie
* Creating sessions and cookies in Django

12. USING OTHER DATABASES IN DJANGO

* Using SQLite
* Configuring MySQL database
* Working with MySQL in Django
* Configuring Oracle database
* Working with Oracle in Django

13. DJANGO RESTful API

* CRUD operations

14. LIVE PROJECT IMPLEMENTATION

* Project life cycle
* Creating a functional website in Django

1. **Introduction to Django:**
   * Understanding what Django is and its features.
   * Setting up a development environment.
2. **Django Models:**
   * Creating models to define the database schema.
   * Defining relationships between models (OneToOne, ForeignKey, ManyToMany).
   * Working with model fields and options.
3. **Django Admin Interface:**
   * Utilizing Django's built-in admin interface for CRUD (Create, Read, Update, Delete) operations on models.
4. **Views and URL Mapping:**
   * Creating views to handle HTTP requests.
   * Mapping URLs to views using URL patterns.
5. **Templates:**
   * Designing HTML templates using Django's template language.
   * Passing data from views to templates.
6. **Static Files and Media Handling:**
   * Managing static files (CSS, JavaScript) and media files (images, videos).
   * Using Django's static and media handling capabilities.
7. **Forms and Validation:**
   * Building and processing HTML forms using Django's Form class.
   * Implementing form validation and error handling.
8. **User Authentication:**
   * Implementing user registration and login functionality.
   * Using Django's authentication system.
9. **Django ORM (Object-Relational Mapping):**
   * Querying the database using Django's QuerySet API.
   * Performing CRUD operations on database records.
10. **Class-Based Views:**
    * Understanding and using class-based views for more organized code.
11. **Middleware:**
    * Working with middleware components for request and response processing.
12. **URL Namespaces and Includes:**
    * Organizing URL patterns using namespaces and includes.
13. **Django Forms:**
    * Working with Django forms in detail, including ModelForms.
14. **Django Templates:**
    * Advanced template features like template inheritance and custom template tags.
15. **Security and Best Practices:**
    * Implementing security measures in Django applications.
    * Following best practices for Django development.
16. **Deployment:**
    * Preparing a Django project for deployment on production servers.
17. **Additional Django Features:**
    * Introduction to additional features like signals, caching, internationalization, etc.

**Class 1: Introduction to Django**

* Introduction to web frameworks and Django
* Setting up a Django development environment
* Creating a new Django project and understanding its structure
* Running the development server and accessing the default Django admin interface

**Class 2: Django Models**

* Defining models and database schema
* Field types and options for models
* Creating relationships between models (OneToOne, ForeignKey, ManyToMany)
* Registering models in the admin interface for CRUD operations

**Class 3: Views and URLs**

* Understanding views and HTTP methods (GET, POST)
* Mapping URLs to views using URL patterns
* Rendering templates and passing data from views to templates

**Class 4: Templates**

* Introduction to Django's template language
* Template inheritance and extending base templates
* Working with template tags and filters

**Class 5: Forms and Validation**

* Creating HTML forms using Django's Form class
* Handling form submissions and form validation
* Displaying form errors and messages

**Class 6: User Authentication**

* Implementing user registration and login functionality
* Using Django's built-in authentication system
* Handling user sessions and authentication-related views

**Class 7: Django ORM (Object-Relational Mapping)**

* Querying the database using Django's QuerySet API
* Performing CRUD operations on database records
* Understanding database migrations

**Class 8: Class-Based Views**

* Introduction to class-based views (CBVs)
* Using built-in CBVs for common tasks
* Creating custom CBVs

**Class 9: Static Files and Media Handling**

* Managing static files (CSS, JavaScript) in Django
* Handling media files (images, videos) in Django

**Class 10: Middleware**

* Understanding Django middleware and its role in request/response processing
* Creating custom middleware components

**Class 11: URL Namespaces and Includes**

* Organizing URL patterns using namespaces and includes

**Class 12: Django Forms**

* Working with Django forms in detail, including ModelForms

**Class 13: Django Templates (Advanced)**

* Advanced template features like template inheritance and custom template tags

**Class 14: Security and Best Practices**

* Implementing security measures in Django applications
* Following best practices for Django development

**Class 15: Deployment**

* Preparing a Django project for deployment on production servers

**Class 16: Additional Django Features**

* Introduction to additional features like signals, caching, internationalization, etc.

**Class 17: Final Project**

* Students work on a real-world Django project from start to finish, applying the concepts learned throughout the course.

Roadmap for Django

1. **Python Basics:**
   * If you're new to Python, start with learning the basics of the language, including variables, data types, loops, functions, etc. Django is a Python web framework, so having a solid foundation in Python is essential.
2. **HTML, CSS, and JavaScript:**
   * Familiarize yourself with front-end technologies like HTML for markup, CSS for styling, and JavaScript for interactivity. Django templates heavily use HTML for rendering content.
3. **Version Control with Git:**
   * Learn how to use Git for version control. Understanding version control is crucial for collaborating on projects and managing code changes effectively.
4. **Introduction to Django:**
   * Understand what Django is and how it works as a web framework.
   * Set up a development environment with Django installed.
5. **Django Basics:**
   * Create a new Django project and understand its directory structure.
   * Define and work with Django apps within a project.
   * Run the development server and view the default admin interface.
6. **Django Models and Admin:**
   * Define models to create database tables.
   * Use the Django admin interface for CRUD (Create, Read, Update, Delete) operations on models.
7. **Views and URL Patterns:**
   * Create views to handle HTTP requests.
   * Map URLs to views using URL patterns.
8. **Django Templates:**
   * Design HTML templates using Django's template language.
   * Pass data from views to templates for dynamic content rendering.
9. **Static Files and Media Handling:**
   * Manage static files (CSS, JavaScript) and media files (images, uploads) in Django.
10. **User Authentication and Authorization:**
    * Implement user registration and login functionality.
    * Use Django's authentication system to handle user sessions and permissions.
11. **Forms and Validation:**
    * Build and process HTML forms using Django's Form class.
    * Implement form validation and error handling.
12. **Class-Based Views (CBVs):**
    * Understand class-based views and their benefits over function-based views.
    * Use built-in CBVs for common tasks and create custom CBVs.
13. **Django ORM (Object-Relational Mapping):**
    * Query the database using Django's QuerySet API.
    * Perform CRUD operations on database records.
    * Learn about database migrations to manage schema changes.
14. **Middleware:**
    * Work with middleware components for request and response processing.
15. **Advanced Topics:**
    * Add pagination to handle large datasets.
    * Implement custom template tags and filters.
    * Handle file uploads and file processing.
    * Use Django's built-in caching mechanism for performance optimization.
16. **Deployment:**
    * Prepare a Django project for deployment on production servers.
    * Learn about hosting options, server setup, and database configurations.
17. **Testing:**
    * Write unit tests and integration tests for Django applications.
18. **Security and Best Practices:**
    * Implement security measures to protect your Django application.
    * Follow best practices in Django development.
19. **RESTful APIs with Django:**
    * Learn to build RESTful APIs using Django REST framework.
20. **Continuous Learning and Projects:**
    * Keep exploring Django's documentation and advanced topics.
    * Work on real-world projects to apply your knowledge and gain practical experience.