System Requirements

Specification Index

For

Django Static Files Configuration for Production

(Topic:- Django Deployment to Production)

Version 1.0

Problem Statement Description

Scenario: You are a Django developer working for a company that is deploying its project to production. The company needs to ensure that static files (such as CSS, JavaScript, and image files) are served correctly in the production environment. Currently, Django's default static file handling is set for development, and it needs to be optimized for production.

Your task is to configure Django to handle static files correctly in production, ensuring they are collected, served, and cached properly.

Problem Statement:

You need to:

- **Configure Django to handle static files in the production environment.**
- **Use django.contrib.staticfiles to manage and serve static files in production.**
- **Set up proper caching headers for static files to ensure they are cached in the browser and CDN.**
- **After configuring, ensure that when the app is deployed to production, static files are collected and served correctly.**

Your Task:

- **Configure Django to serve static files correctly in production.**
- **Set up STATIC_URL, STATIC_ROOT, and other necessary settings for production.**
- **Implement proper caching headers for static files.**
- **Run the necessary Django commands (collectstatic) to collect static files.**

Execution Steps to Follow:

1. All actions like build, compile, running application, running test cases will be through Command Terminal.
2. To open the command terminal the test takers, need to go to

   Application menu(Three horizontal lines at left top)->Terminal->NewTerminal.

3. The editor Auto Saves the code.
4. If you want to exit (logout) and to continue the coding later anytime(using Save & Exit option on Assessment LandingPage) then you need to use CTRL+Shift+B command compulsorily on code IDE. This will push or save the updated contents in the internal git/repository. Else the code will not be available in the next login.
5. These are time bound assessments the timer would stop if you logout and while

   logging in back using the same credentials the timer would resume from the same time it was stopped from the previous logout.

6. To test any Restful application, the last option on the left panel of IDE, you can find

   ThunderClient, which is the lightweight equivalent of POSTMAN.

7. To test any UI based application the second last option on the left panel of IDE, you can

   find Browser Preview, where you can launch the application.

8. Install ‘djangorestframework’ module before running the code. For this use the following command.

   pip install djangorestframework

9. Use the following command to run the server

   python3 manage.py runserver

10. Mandatory: Before final submission run the following commands to execute testcases

    python3 manage.py test library.test.test_functional

    python3 manage.py test library.test.test_exceptional

    python3 manage.py test library.test.test_boundary

11. To test rest end points

    Click on ‘Thunder Client’ or use Ctrl+Shift+R->Click on ‘New Request’ (at left side of IDE)

12. Once you are done with development and ready with submission, you may navigate to the previous tab and submit the workspace. It is mandatory to click on “Submit Assessment” after you are done with code.

13. You need to use CTRL+Shift+B - command compulsorily on code IDE, before final submission as well. This will push or save the updated contents in the internal git/repository, and will be used to evaluate the code quality.