System Requirements Specification Index

For

Managing Orders in an E-Commerce Platform

(Topic:- Django Admin Interface)

Version 1.0

Problem Statement Description

Scenario:

You have joined an e-commerce startup as a Django developer. The company is expanding its operations and needs an efficient order management system. The management team wants to customize the Django Admin panel to make order tracking easier.

Orders should be categorized based on their status (e.g., "Pending", "Shipped", "Delivered", "Cancelled"). The admin panel should allow filtering orders by their status and display key details like order_number, customer_name, status, and total_price.

Problem Statement

Your task is to customize the Django Admin panel for the Order model. The model should be structured to enable efficient querying, filtering, and management.

Each order should include the following details: order_number → Unique order identifier (CharField, max length: 20).

- **customer_name → Name of the customer (CharField, max length: 100).**
- **status → Current status of the order (Choices: Pending, Shipped, Delivered, Cancelled).**
- **total_price → Total cost of the order (DecimalField, max_digits=10, decimal_places=2).**
- **created_at → Timestamp when the order was placed (DateTimeField, auto-generated).**

Your Task

1 Define a Django model named Order with the fields mentioned above.

2. Implement status as a choice field for easy categorization.

3 Customize the Django Admin panel to:

- **Display key order details (order_number, customer_name, status, total_price, created_at).**
- **Allow filtering orders by status in the admin panel.**
- **Enable searching orders by order_number and customer_name.**

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.