System Requirements

Specification Index

For

Django API for Restaurant Ordering System

(Topic:- Django Best Practices and Design Patterns)

Version 1.0

Scenario:

You have joined an e-commerce startup as a Django developer. The company is launching a restaurant ordering system, where customers can place orders for different menu items. The front-end team is using a single-page application (SPA) that interacts with the Django API to manage menu items and customer orders. Your task is to design the Django API for handling the restaurant's menu and customer orders.

You need to implement API endpoints to:

- **List all menu items.**
- **Create orders for customers.**
- **Update the order status (e.g., pending, completed).**
- **View the details of a specific order.**

Additionally, you will need to write test cases that call these API endpoints and verify that the system works correctly.

Problem Statement

Your task is to:

Design API endpoints for the MenuItem and Order models.

Write test cases using Django's APIClient to interact with the API endpoints and verify the correct functionality of the API.

The endpoints will be as follows:

- **MenuItem API: GET /api/menu/ - Fetch all menu items.**
- **Order API: POST /api/orders/ - Create a new order.**
- **Order Status Update API: PATCH /api/orders/{id}/update_status/ - Update the status of an existing order.**
- **Order Details API: GET /api/orders/{id}/ - View details of a specific order.**

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.