System Requirements

Specification Index

For

Building a REST API for Task Management

(Topic:- Django REST Framework)

Version 1.0

Scenario:

You have joined a project management software startup as a Django developer. The company is building a new feature to manage tasks within teams. Tasks can have the following details:

- **Title**
- **Description**
- **Due Date**
- **Status (To Do, In Progress, Done)**
- **Priority (Low, Medium, High)**

Your task is to design a Django REST API that allows CRUD operations (Create, Read, Update, Delete) on the Task model using Django REST Framework (DRF). This will enable team members to create, update, delete, and view tasks, helping them track the progress of projects efficiently.

Problem Statement:

Your task is to create the following for a Task model:

- **A Task model with the fields mentioned above.**
- **A REST API that allows users to perform CRUD operations on the Task model.**
- **Proper serialization and validation for the Task model using Django REST Framework.**
- **Include basic validation such as ensuring due dates are valid and the status field is constrained to a fixed set of choices.**

Your Task:

Define a Django model named Task with the following fields:

- **title (CharField, max_length=200)**
- **description (TextField)**
- **due_date (DateTimeField)**
- **status (CharField, choices: "To Do", "In Progress", "Done")**
- **priority (CharField, choices: "Low", "Medium", "High")**

Create a Django REST API for the following operations:

- **POST: Create a new task**
- **GET: Get a list of tasks or details of a specific task**
- **PUT/PATCH: Update an existing task**
- **DELETE: Delete a task**

Implement the required serializers and views.

Create appropriate URL routing for the API.

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.