System Requirements

Specification Index

For

Library Management System

(Topic:- Django Models: Defining models in Django, Model fields (CharField, IntegerField, DateTimeField, etc.)

Version 1.0

Problem Statement: Library Management System

Scenario:

You have been hired as a Django developer at a **digital library startup** that aims to modernize book borrowing and tracking. The system requires a **Book model** to manage book records efficiently and an **Author model** to store author details.

The **Book** model must be linked to the **Author** model via a **ForeignKey**, ensuring that each book belongs to one author but an author can have multiple books.

This structure will help in **efficient querying, author-wise book retrieval, and scalability** as the library collection expands.

Problem Statement:

Your task is to **define a Django model** for a **Library System**, ensuring proper database relationships and constraints.

Each **Author** should include:

- **name** → The full name of the author (**CharField**, max length: 100).
- **birth_date** → The birth date of the author (**DateField**, optional).

Each **Book** should include:

- **title** → A unique title of the book (**CharField**, max length: 200).
- **author** → A ForeignKey linking to **Author** (on delete, books should also be deleted).
- **published_date** → The publication date of the book (**DateField**).
- **isbn** → Unique ISBN code (**CharField**, max length: 13, unique=True).
- **available_copies** → Number of copies available in the library (**IntegerField**, default=1).

Your Task:

1. Define **Author** and **Book** models with the fields mentioned above.
2. Ensure that the models follow Django's **ORM standards** and **best practices**.
3. Implement **ForeignKey constraints** between **Book** and **Author**.

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.