**Django API Documentation**

**Overview**

This documentation outlines the setup and functionality of a Django RESTful API designed for managing books and authors. The API includes user authentication, search functionality, and a recommendation system. This guide explains the design of the models, the similarity algorithm used for recommendations, and testing response times of the endpoints. It also includes related commands for setup and maintenance.

**1. Models Design and Logic**

**Models**

**Author Model**

class Author(models.Model):

name = models.CharField(max\_length=255)

about = models.TextField(blank=True, null=True)

def \_\_str\_\_(self):

return self.name

* **name**: Represents the author's name.
* **about**: An optional text field for additional information about the author.

The \_\_str\_\_ method returns the author's name, which is useful for displaying instances in the Django admin interface and in string representations.

**Book Model**

class Book(models.Model):

title = models.CharField(max\_length=255)

description = models.TextField()

author = models.ForeignKey(Author, related\_name='books', on\_delete=models.CASCADE)

publication\_date = models.DateField()

isbn = models.CharField(max\_length=13, unique=True)

num\_pages = models.IntegerField()

genre = models.CharField(max\_length=255, blank=True, null=True) # Added genre field

def \_\_str\_\_(self):

return self.title

* **title**: The title of the book.
* **description**: A text field describing the book.
* **author**: A foreign key linking to the Author model.
* **publication\_date**: The date when the book was published.
* **isbn**: The ISBN of the book (unique).
* **num\_pages**: Number of pages in the book.
* **genre**: A text field for the genre of the book (optional).

The \_\_str\_\_ method returns the book title, which is useful for display purposes.

**Favorite Model**

class Favorite(models.Model):

user = models.ForeignKey(User, related\_name='favorites', on\_delete=models.CASCADE)

book = models.ForeignKey(Book, related\_name='favorited\_by', on\_delete=models.CASCADE)

class Meta:

unique\_together = ('user', 'book')

* **user**: A foreign key linking to the Django User model.
* **book**: A foreign key linking to the Book model.

The unique\_together constraint ensures that each user can only have one favorite entry per book.

**2. Similarity Algorithm for Recommendations**

**Similarity Algorithm**

The recommendation logic is implemented in the get\_recommendations method:

def get\_recommendations(self, user):

favorite\_books = Favorite.objects.filter(user=user).values\_list('book\_id', flat=True)

if not favorite\_books:

return []

# Example: Recommend books by the same genre as user's favorites

favorite\_genres = Book.objects.filter(id\_\_in=favorite\_books).values\_list('genre', flat=True).distinct()

recommended\_books = Book.objects.filter(genre\_\_in=favorite\_genres).exclude(id\_\_in=favorite\_books).distinct()[:5]

return BookSerializer(recommended\_books, many=True).data

**Explanation:**

1. **Retrieve Favorite Books**: Get a list of book IDs that the user has favorited.
2. **Find Favorite Genres**: Get the genres of these favorite books.
3. **Recommend Books**: Find books that share these genres, excluding those already favorited by the user. Limit the recommendations to 5 books.

**3. API Endpoints**

**Endpoints**

**User Authentication**

* **Register**: POST /api/register/
  + **Request Body**: { "username": "", "password": "", "email": "" }
  + **Response**: JWT tokens (access and refresh).
* **Login**: POST /api/login/
  + **Request Body**: { "username": "", "password": "" }
  + **Response**: JWT tokens (access and refresh).

**Books**

* **List/Create Books**: GET/POST /api/books/
  + **GET**: List all books, optionally filter by search query.
  + **POST**: Create a new book.
* **Retrieve/Update/Delete Book**: GET/PUT/DELETE /api/books/<int:pk>/
  + **GET**: Retrieve a specific book by ID.
  + **PUT**: Update a book by ID.
  + **DELETE**: Delete a book by ID.

**Authors**

* **List/Create Authors**: GET/POST /api/authors/
  + **GET**: List all authors.
  + **POST**: Create a new author.
* **Retrieve/Update/Delete Author**: GET/PUT/DELETE /api/authors/<int:pk>/
  + **GET**: Retrieve a specific author by ID.
  + **PUT**: Update an author by ID.
  + **DELETE**: Delete an author by ID.

**Favorites**

* **List/Create Favorites**: GET/POST /api/favorites/
  + **GET**: List all favorite books for the authenticated user.
  + **POST**: Add a book to favorites (up to 20 books).
* **Delete Favorite**: DELETE /api/favorites/
  + **Request Body**: { "book\_id": <int> }
  + **Response**: Confirmation of removal or error if not found.

**4. Related Commands**

**Setting Up the Project**

* **Install Required Packages:**

pip install djangorestframework djangorestframework-simplejwt

**Creating Database Schema**

* **Make Migrations:**

python manage.py makemigrations

* **Apply Migrations:**

python manage.py migrate

**Running the Development Server**

* **Start the Django Development Server:**

python manage.py runserver

**Creating a Superuser**

* **Create Superuser for Admin Interface:**

python manage.py createsuperuser

**Testing the API with Postman**

* **Register a New User:**

**Request Type**: POST  
**URL**: http://127.0.0.1:8000/api/register/  
**Body**: { "username": "testuser", "password": "testpassword", "email": "test@example.com" }

* **Login:**

**Request Type**: POST  
**URL**: http://127.0.0.1:8000/api/login/  
**Body**: { "username": "testuser", "password": "testpassword" }

* **Search Books:**

**Request Type**: GET  
**URL**: http://127.0.0.1:8000/api/books/?search=Harry  
**Headers**: Authorization: Bearer <access\_token>

* **Add a Favorite Book:**

**Request Type**: POST  
**URL**: http://127.0.0.1:8000/api/favorites/  
**Body**: { "book\_id": 1 }  
**Headers**: Authorization: Bearer <access\_token>

* **Remove a Favorite Book:**

**Request Type**: DELETE  
**URL**: http://127.0.0.1:8000/api/favorites/  
**Body**: { "book\_id": 1 }  
**Headers**: Authorization: Bearer <access\_token>

**Summary**

This documentation covers the design of the Django models, the recommendation algorithm used for suggesting books, and the API endpoints for managing books, authors, and user favorites. It also includes commands for setting up the project, creating and applying migrations, running the server, and testing the API using Postman.