Coding Round

Python

Duration: 1 Hour

Objective

This test evaluates the candidate's ability to:

1. Work with **Django Rest Framework (DRF)** to create APIs.
2. Handle **large datasets** efficiently and perform complex queries.
3. Optimize API performance using **caching** and **database indexing**.

Instructions

- Use Django as the web framework and Django Rest Framework (DRF) to build the APIs.
- Write clean, modular, and reusable code with comments wherever necessary.
- Handle edge cases and validate the input data properly.
- Use caching and database indexing for performance optimization.
- Ensure the project can handle large datasets efficiently.

Coding Criteria

Skill	Criteria	Rating
Dataset Handling	Efficient handling of large dataset uploads	10
	Validation and error handling for dataset	5
API Development	Correct implementation of DRF API	5
	Filters and query parameters work as expected	5
	Aggregation logic is correct	5
Performance	Caching implemented correctly	5
	Database queries optimized with indexing	10
Coding Quality	Code is modular and reusable	5
	Proper variable naming conventions used	5
Documentation	Comments and explanations for critical sections	5
Total		**60**

Tasks

Task 1: Import Large Dataset

Create a Django management command to import the dataset (large_dataset.csv) into the Product model.

Requirements:

1. **Efficient Processing**:
   - Use bulk inserts to handle large datasets.
2. **Validation**:
   - Ensure price and stock are non-negative.
   - Handle invalid or missing data gracefully.

Dataset CSV - https://drive.google.com/file/d/1QVonkcBUawYLzHoNEAZh4XQyGmdVHarR/view?usp=sharing

Task 2: Optimized API for Data Retrieval

Create an API endpoint /api/products/analytics/ that provides analytics on the Product model.

Requirements:

1. **Filtering**:
   - Accept query parameters:
     - category (case-insensitive).
     - min_price and max_price.
2. **Aggregation**:
   - Return the following statistics for the filtered data:
     - Total number of products (total_products).
     - Average price of the filtered products (average_price).
     - Total stock value (total_stock_value = stock * price).

```
GET
/api/products/analytics/?category=electronics&min_price
=10&max_price=100


{
  "total_products": 1200,
  "average_price": 45.67,
  "total_stock_value": 150000.00
}
```

Task 3: Caching and Optimization

Optimize the /api/products/analytics/ endpoint for performance.

Requirements:

1. **Caching**:
   - Cache results for 5 minutes.
   - Invalidate the cache when query parameters change.
2. **Indexing**:
   - Add database indexing for fields used in filtering (category, price).