CIN – U52100HR2019PTC081059
EPVI New Technological Epoch Private Limited
1800 2030 276 hi@enlog.co.in
Registered Office: House No. 86 Block No. 4, Gali SDO Kitab Singh, Shiv Nagar, Fatehabad, Haryana 125050

**Assignment: Advanced E-commerce API with Caching and Notifications**

You are tasked with creating a **Django REST API** for a small e-commerce system that allows users to:

1. **Register, login, and manage their profiles**.
2. **Browse products, categories, and place orders**.
3. **Leverage caching and pagination for optimization**.

**Requirements:**

1. **User Authentication (JWT-based)**

- Use **Django Rest Framework SimpleJWT** for token-based authentication.
- Allow users to:
  - Register with email and password.
  - Log in to obtain an access token and refresh token.
  - Manage their profile (name, address, phone, etc.).
  - View their order history and order details.

2. **Product Management**

- Create **Product** and **Category** models with the following fields:
  - **Category**: name, description
  - **Product**: name, description, price, stock, category (foreign key)
- **Admin users** can:
  - Create, update, and delete categories.
  - Add, update, and delete products.
  - Manage product stock (when a product is ordered, its stock decreases).

3. **Order System**

- Users can:
  - Add products to their **cart**.
  - Place an order from the cart.
  - Get notifications (via an API call) when the order status changes (using WebSockets or Django Channels for real-time updates).
- **Order Model**:
  - Order: user, product(s), total price, status (pending, shipped, delivered), created_at, updated_at
- Implement a flow where orders go through these statuses:
  - **Pending** (default).
  - **Shipped**.
  - **Delivered**.

CIN – U52100HR2019PTC081059
EPVI New Technological Epoch Private Limited
1800 2030 276 hi@enlog.co.in
Registered Office: House No. 86 Block No. 4, Gali SDO
Kitab Singh, Shiv Nagar, Fatehabad, Haryana 125050

**4. Caching & Performance Optimizations**

- Use Redis as a caching layer to store products and categories for quick access.
- Implement a caching mechanism to store expensive database queries (e.g., fetching product lists).
  - Set a timeout for the cached data (e.g., 1 hour).
  - When a product's stock or details change, invalidate the cache.
- Optimize queries using Django's select related and prefetch related for complex relations (product and category data).

**5. Pagination & Filtering**

- Implement pagination for the product listing (limit 10 products per page).
- Allow filtering products by:
  - Category.
  - Price range.
  - Stock availability (in stock or out of stock).
- Ensure that the API can handle large amounts of data efficiently with paginated responses.

**6. Real-Time Notifications**

- Use Django Channels or WebSockets to notify users of their order status updates.
  - When the order status changes (e.g., from pending to shipped), notify the user in real time.

**Instructions:**

1. **Setup:** Use Django and Django REST Framework to build the API. Use PostgreSQL as the database and Redis for caching.
2. **Authentication:** Implement token-based authentication using JWT.
3. **Caching:** Use Redis for caching product and category data.

**Submission:**

- Create a GitHub repository and upload your project code and share Url.
- Write clear instructions in the `README.md` for setting up and running the project.