Backend Developer Technical Assessment

Creating a comprehensive RESTful API for managing pharmacy products using Django Rest Framework (DRF) involves several steps. This project will demonstrate skills in DRF, JWT authentication, model creation, CRUD operations, endpoint creation, data validation, and file handling. Here's a detailed guide:

1. JWT Authentication

- Use `djangorestframework-simplejwt` for JWT authentication.
- Implement token authentication for secure access to the API.
- Create endpoints for token generation, refresh, and verification.

2. Django Model for "Product"

- Define a **`Product`** model in Django
- Implement model methods if needed for custom behavior.

3. CRUD Operations

- Utilize DRF's viewsets or APIViews to implement CRUD operations for the **`Product`** model.
- Ensure the API handles Create, Read, Update, and Delete operations efficiently and securely.

4. RESTful API Endpoints

- GET to list all products, POST to create a new product.
- GET to retrieve a single product, PUT/PATCH to update a product, DELETE to remove a product.

5. Validation Rules

- Use Django's validation rules to validate data.

6. File Handling

- Support for image (JPEG, PNG) and PDF file uploads.
- Configure Django to handle media files and static files correctly.
- Implement a method to handle PDF files, perhaps as downloadable product information.

Others Features

- Implement API versioning.
- Add pagination and filtering capabilities to the product list endpoint.
- Include unit and integration tests to ensure API reliability and stability.
- Implement rate limiting to protect against abuse.
- Add a caching mechanism for frequently accessed data.
- Dockerize the application for easy deployment.
- Add API documentation using tools like Swagger.

Considerations for Deployment and Maintenance

- Ensure the application is scalable and follows best practices.
- Maintain clear documentation and code comments for future maintainability.
- Consider security practices like throttling, permission classes, and data validation.

Code Structure

- Organize the project into multiple apps if needed for scalability.
- Use a modular approach for views, serializers, models, and tests.
- Implement error handling and custom exceptions for clarity in response messages.