**Next-step Review**

If given more time to build the solution, my priorities would have been focused on enhancing scalability, maintainability, and functionality to support future growth and ease of use. Here are the main areas I would prioritize:

**1. Enhanced Search and Filtering Capabilities**:

To improve usability for end-users, I would add advanced search and filtering features, allowing users to query SKUs based on specific fields (e.g., dose, presentation, unit). This would enable buyers and sellers to locate relevant SKUs faster, especially in a growing database.

**2. Database Transition and Optimization**:

While I used SQLite for initial development and local testing, I would migrate to PostgreSQL for production due to its scalability and support for advanced features like indexing, which would enhance performance for larger datasets.

**3. Role-Based Access Control (RBAC)**:

I would implement role-based access to secure the API, allowing different access levels (e.g., admin, editor, viewer) based on the user’s role. This would provide better control over CRUD operations, enhancing data integrity and security.

**4. Comprehensive Validation and Error Handling**:

To improve data quality, I would expand validation mechanisms, especially for bulk uploads, ensuring that each entry complies with unique constraints. Detailed error handling would give users meaningful feedback on why certain entries may have failed during bulk uploads.

**5. Automated Testing with Code Coverage**:

I would extend test coverage with more extensive unit and integration tests to catch edge cases, ensuring the reliability and stability of the system. Additionally, I’d consider adding test coverage metrics to monitor code quality over time.

**6. CI/CD Pipeline for Automated Deployment**:

Setting up a CI/CD pipeline with automated testing and deployment steps (using Docker and GitHub Actions) would streamline the development process and allow for quicker iterations and quality control in production environments.

**7. Bulk Operation Performance Optimization**:

To handle large-scale bulk uploads, I’d explore optimizations for batch processing, leveraging Django’s bulk operations, and fine-tuning database configurations for more efficient inserts, which would improve response times for large data uploads.