. Key Architectural Decisions

- **SpringBoot microservices backend**
  - **Order Service**
    – Handles customer order creation and order history management
  - **Warehouse Service**
    – Manages inventory, package operations, product tracking, and integrates directly with the UPS API
  - **World Connector**
    – Abstracts synchronous/asynchronous communication with the simulated World environment
  - **(Future) UPS Service**
    – A standalone microservice dedicated to handling all UPS third-party logistics requests
- **Unified API Gateway**
  - Single external entry point (e.g., port 8080), simplifying frontend integration and deployment
  - Centralized management of routing, CORS, JWT authentication, rate limiting, and circuit breaking

---

2. Advantages of Our Implementation

1. **Service Isolation**
   - Failure in a single service does not affect the entire system:

     When the World Connector fails during truck loading, the Warehouse Service can still continue processing new orders and packing.

2. **Technology Flexibility**
   - **Warehouse Service**: Uses Spring Data JPA with transaction management to ensure consistent package states
   - **World Connector**: Uses asynchronous messaging to improve interaction with the World simulator

3. **Inter-Service Communication**
    - Synchronous REST using RestTemplate with internal DTOs as the communication contract
4. **Unified API Gateway Endpoint**
    - **CORS**: Only allows http://localhost:3000
    - **Clean routing structure** (all under http://localhost:8080):

      /api/orders/** → Order Service
      /api/warehouse/** → Warehouse Service
      /internal/world/** → Warehouse Callback
      /api/ups/** → UPS Service(etc)

5. **Amazon-Style React Frontend**
    - Mimics Amazon UI
    - Built with React and TypeScript
6. **Observability & Monitoring**
    - **Granular Logging**: SLF4J + Logback with INFO/WARN/ERROR levels for application monitoring
7. **Extensible AckManager (Factory Pattern)**
    - World Connector dynamically generates ACK handlers based on events (created, matched, arrived)
    - Fully follows the Open/Closed Principle, allowing new event types to be added without changing existing code

---

These differentiation points ensure **high availability**, **maintainability**, and **scalability**,