**EA Foods – Pre-Order QA Strategy**

**1. Introduction**

This document defines the QA strategy for validating EA Foods’ **Pre-Order Model**. The goal is to ensure the system is **functionally correct, scalable, resilient, and compliant** with all business constraints.

The QA scope includes functional, negative, and edge-case testing of pre-orders, inventory-driven rules, delivery slots, and cut-off enforcement. Additionally, automation coverage will ensure repeatable regression validation.

**2. Objectives**

* Validate **core business constraints**: inventory enforcement, delivery slots, 6PM cut-off, cancellation restores stock, and stock updates by Operation managers.
* Ensure **data integrity** during simultaneous or bulk order scenarios.
* Verify **usability and reliability** of flows from customer perspective.
* Provide **observability** through logs, screenshots, and structured bug reporting.
* Document assumptions, risks, and trade-offs clearly for transparency.
* Track QA metrics: automation coverage %, defect leakage %, execution pass rate.

**3. Test Scope**

**In Scope**

* Functional testing: order placement, cancellation, stock updates.
* Negative testing: stock limits, invalid slots, invalid quantities.
* Edge testing: boundary cut-off conditions, zero/negative stock, concurrent orders.
* Automation: Covering critical flows through Postman (RestAssuredAPI) and validating customer-facing workflows via UI automation (Selenium).
* Performance/Load testing → JMeter.
* Strategy lacks how automation fits into CI/CD pipelines (Jenkins, GitHub Actions).
* Security testing (basic) → JWT token expiry, unauthorized access.
* Database validation (stock consistency across orders).
* API contract/schema validation.

**Out of Scope**

* Non-functional requirements like security penetration testing.

**4. Test Approach**

**Manual Testing**

* Exploratory testing for UI and edge-case validation.
* Verification of slot boundaries, stock updates, and unusual input conditions.

**Automation Testing**

* Core regression suite implemented in **Postman (API workflows)**.
* Automated checks include:
  + Order placed within stock → success.
  + Order exceeding stock → rejected.
  + Order after cut-off → scheduled +2 days.
  + Cancel order → stock restored.
  + Ops stock update → reflected in availability.

**Test Data Management**

* Seed local DB/JSON mocks with **5–10 products** of varying stock (including edge case: 0 stock).
* Use **fixed delivery slots**: Morning (8–11), Afternoon (12–3), Evening (4–7).
* Cut-off time set to **6:00 PM local time**.

**Test Environment**

* Automated tests run locally with clear setup instructions in README.

**5. Deliverables**

### ****Test Planning****

* Defined QA scope: functional, negative, edge case, and automation testing.
* Selected tools: **Postman** (API regression) and **manual exploratory testing** for edge cases.
* Designed **test data**: 5–10 products seeded with varying stock levels (including 0 stock).
* QA metrics: coverage %, defect density, leakage rate.

### ****Test Design****

* Prepared **test cases** covering:
  + Positive flows (valid orders, cancellations, stock updates).
  + Negative flows (exceeding stock, invalid slots, orders after cut-off).
  + Edge cases (exact 6:00 PM order, simultaneous orders, and zero stock).
* Prioritized **critical business scenarios** (stock enforcement, cut-off handling).

### ****Test Execution Reports****

* Conducted manual testing of functional flows:
  + Placing valid orders
  + Cancelling orders & verifying stock restoration
  + Ops stock updates (8AM/6PM)
* Executed negative and boundary scenarios (e.g., order at 6:00:01 PM).
* Automated **5 regression flows** in Postman:
  + Valid order before cut-off
  + Rejected order after cut-off
  + Rejected order exceeding stock
  + Cancel restores stock
  + Ops stock update reflects in system

**6. Assumptions**

* Orders placed **after 6:00 PM** are automatically pushed to **+2 days**.
* Slots are **strictly limited** to  (Morning 8–11, Afternoon 12–3, Evening 4–7).
* Cancelling an order **immediately restores stock**.
* Operation managers update times are **exactly 8AM and 6PM**.
* All services (inventory, slot management) available during test cycles
* All dependent services (inventory, slot mgmt.) remain stable during testing.

**7. Risks & Mitigation**

| **Risk** | **Impact** | **Mitigation** |
| --- | --- | --- |
| Concurrent orders causing stock mismatch | High | Simulate concurrency in automation |
| Boundary case failures at cut-off (6:00 PM) | Medium | Test exactly at 6:00 PM and 6:01 PM |
| Manual misses on high-volume orders | Medium | Automate bulk-order flows |

**8. QA Execution Plan**

1. **Unit Testing**: Validate core logic (stock decrement, cutoff calculation, slot validation).
2. **Integration Testing**: Validate workflows (place → cancel → reorder → stock update).
3. **Negative/Edge Testing**: Invalid slots, invalid products, simultaneous orders.
4. **Automation**: Smoke → Regression → Exploratory → Performance → Security.
5. **Reporting**: Capture defects, screenshots and test run logs..

**9. Expected Outcome**

* Robust validation of **inventory-driven pre-order model**.
* Automation ensures **repeatability and regression readiness**.
* Clear defect reporting highlights real-world failure risks.
* Documentation ensures transparency and reviewer confidence.
* Faster feedback cycles through automation in CI/CD.