

## Project Timelines

#	Milestones	SLA-Fanatics	SLA - In Real-time	Owner	Communication Channel
1.	Ambiguity Review & Alignment	20 mins	1 day	QA, Dev, Business	Online Meeting, Slack Channel
2.	Test Plan Design & Drive Alignment	2.30 hours	1 day	QA	Offline Review/ Online Meeting
3.	Test Execution Manual (Smoke Test, Critical business path)	-	½ day	QA	Slack Channel
4.	Automation Strategy Documentation	30 mins	1 to 2 days	QA	Offline Review/ Online Meeting
5.	Develop Automation	2 hours	3 to 4 days depending on the requirements	QA	Focus Time
6.	Test Reporting and Bug Triage	-	1 day	QA	Online Meeting
7.	Bug Bash & Launch Readiness - Retesting and No P0 bugs	-	½ day	QA	Online Meeting
8.	Prod Deployment	-		QA, Dev	
9.	Monitoring	-		QA, Dev	
10.	Sprint Demo	-	1 hour	Team	

## Ambiguity Review

The ambiguities are raised in a thought process of real-time environment assuming BRD is prepared by Business and each microservice or interface is owned by Dev teams.

### 1. Payload & Input Validation

- Are all fields mandatory (order\_id, customer\_id, item\_id, order\_ts)?
- Can items be an empty array?
- Is there a maximum number of items allowed per order?
- Are duplicate SKUs allowed in the same order?
- Is quantity = 0 allowed and negative quantity allowed?
- What about duplicate SKU?
- Are extra/unrecognized fields allowed in the payload?
- Should invalid timestamps result in 400 or auto-correction?
- Is SKU case-sensitive?

#### Constraints

- Request must be JSON
- Quantity must be a positive integer

### 2. Catalog Enrichment Ambiguities

- What happens if a SKU does not exist in the catalog?
- Is enrichment synchronous or asynchronous?
- What fields are added during enrichment?
- Can partial enrichment succeed?
- What is the timeout threshold for catalog lookup?
- Should missing SKU fail the entire request?
- Is it heavily dependent on real-time data?
- Should catalog failures return 502 or 503?

#### Constraints

- Catalog service is an external dependency
- Schema of enriched data must be consistent

### **3. Queue Publishing Ambiguities**

#### **Ambiguities**

- Is message delivery guaranteed (at-least-once vs exactly-once)?
- What happens if SQS publish fails?
- Is message ordering important?
- Is message deduplication required?
- Is DLQ configured?
- Should API return success if SQS publish fails?
- Should retries be automatic?

#### **Constraints**

- Messages must be published before returning 200
- Queue must handle high throughput
- Failure handling must be deterministic

### **4. Error Handling Ambiguities**

#### **Ambiguities**

- Are retries expected from client or server?
- Should error responses include correlation IDs?
- Should partial failures be retried automatically?
- Should client receive retry hints?

#### **Constraints**

- Errors must be consistent and predictable
- No partial success responses allowed

### **5. Idempotency & Duplicate Handling**

#### **Ambiguities**

- What happens if the same order is submitted twice?
- Is order\_id guaranteed unique?
- Are retries expected from clients?
- Should duplicate requests be ignored or rejected?
- Is idempotency required across retries?

#### **Constraints**

- Duplicate submissions must not create duplicate orders

### **6. Performance & Scalability Ambiguities**

#### **Ambiguities**

- Expected TPS or QPS?
- Peak traffic expectations?
- Payload size limits?
- SLA for API response time?

#### **Constraints**

- Must support horizontal scaling
- Must not block on slow dependencies

### **7. Security & Compliance Ambiguities**

#### **Ambiguities**

- Is authentication required?
- Are authorization checks needed?
- Is this an internal or public API?

#### **Constraints**

- Must prevent unauthorized access