01-QA Governance Protocol

SVS (Oh-Dish Backoffice) – Startup Simulation

Document Type: Governance / Quality Protocol

Prepared By: QA– SH (Portfolio Simulation)

Date: February 14, 2026

Document Status: Representative Governance Artifact (Simulation)

Note: This document represents a structured QA governance framework developed as part of QA Internship training to simulate enterprise-level controls in an early-stage SaaS startup environment.

Table of Contents

1. Purpose & Scope

This protocol defines the quality governance structure for the SVS (Oh-Dish Backoffice) startup simulation. It establishes operational standards for sprint execution, defect management, automation controls, and release readiness. The objective is to simulate enterprise-level QA governance practices within a startup team model (3 Developers + 1 QA).

2. Organizational Context

SVS is modeled as a restaurant backoffice SaaS system including the following modules:

- Identity & Access (Authentication, Users, Roles, RBAC)
- POS Transaction Engine (Cart, Tax, Discounts)
- Payments & Reconciliation
- Orders Lifecycle Management
- Reporting & Profit/Loss Dashboards

Architecture assumptions: MySQL database, microservices-oriented domain separation, CI/CD deployment pipeline, Manual + TestCafe Studio (JavaScript) automation, environments: Dev → QA → Staging → Production.

3. Roles & Responsibilities

Role	Primary Responsibility	Quality Accountability
Product Owner	Defines backlog & priorities	Clarifies acceptance criteria
Developers (3)	Feature implementation & unit testing	Code quality & contract stability
QA Intern (SH)	System validation, regression, automation, governance tracking	Quality evidence, risk reporting, release gate validation
Stakeholders	Business validation (UAT)	Release acceptance sign-off

4. Quality Operating Principles

- Build Quality In: Validation begins at backlog refinement with testability review.
- Risk-Based Testing: Payments, RBAC, and financial reports treated as high-risk domains.
- Traceability: Every story must map to test cases and evidence.
- Automation Growth: Critical flows prioritized for TestCafe automation.
- Release Integrity: No open Critical defects allowed at production release.

5. Definition of Ready (DoR)

- Acceptance criteria clearly defined and testable.
- Dependencies identified (internal or external).
- Environment readiness confirmed.
- Security/compliance considerations documented (especially for payments/PII).

6. Definition of Done (DoD)

- All acceptance criteria validated (positive & negative cases).
- Regression impact assessed and executed.
- Automation updated for stable flows where applicable.
- No Critical defects open; Major defects documented with decision.
- Evidence stored (logs, screenshots, reports).

7. Traceability & Evidence Standards

Traceability model: Requirement → Epic → User Story → Test Case (Manual/TestCafe) → Defect → Release Evidence.

Mandatory Evidence Includes:

- Test execution logs
- Automation reports
- Defect screenshots & reproduction steps
- Reconciliation logs for payment-related flows
- Sprint review quality summary

8. Severity Model & Escalation

Severity	Definition	Release Rule
Critical	Revenue/security/compliance impact (e.g., payment charged but order missing)	Release Blocked
Major	High functional impact with workaround	Decision Required
Minor	Low-risk cosmetic/edge issue	Can Defer

9. Automation Policy (Manual + TestCafe)

Automation Prioritization Order:

1. Login & RBAC enforcement
2. POS cart calculations
3. Payment processing (success/failure)
4. Order lifecycle transitions
5. Core reporting views

Target: Achieve minimum 60% regression automation coverage by end of 6-sprint Program Increment.

10. Governance Review Cycle

- Sprint-level quality report generated at end of each sprint.
- Risk index evaluated before staging deployment.
- Release Go/No-Go meeting conducted before production deployment.
- Retrospective includes defect leakage and automation gap review.