Software Engineering Report
Validation Analysis
JavaBrew Vending Machine Platform

Automated Validation System

December 2, 2025

Contents

1 Executive Summary

1.1 Validation Status

Table 1: Overall Validation Metrics

Metric	Value
Total Requirements Extracted	62
Total Use Cases Extracted	18 (15 explicit, 3 implicit)
Total Architecture Components	6 layers, 18 named components
Total Tests Extracted	63
Requirements Coverage	95.2% (59/62 covered or partial)
Use Case Coverage	83.3% (15/18 covered)
Architecture Coverage	100% (6/6 layers described)
Test Coverage (per report)	100% of implemented features have some tests; gaps in journeys and hardware integration
Feature-Based Coverage (best practices)	68.8% (11/16 criteria satisfied)
Overall Status	**CONDITIONALLY PASSED**

1.2 Key Findings

- **Strengths:**
  - High requirements coverage (95.2%) with explicit traceability to architecture and tests for most domains.
  - Clean six-layer architecture with disciplined separation of concerns and strategic use of patterns (Builder, DAO, Mapper).
  - Strong test pyramid: 71% unit, 19% integration, 10% system tests, with emphasis on error scenarios.
  - Rich domain model aligned with Domain-Driven Design principles.
  - Automated traceability already in place, enabling impact analysis.
- **Critical Risks:**
  - **Offline Operation Gap**: REQ-18, REQ-19, REQ-20 are completely unsupported (no use cases, components, or tests).
  - **Remote Maintenance Gap**: REQ-60 and UC-18 promise remote maintenance without hardware abstraction or IoT gateway.
  - **Component Ambiguity**: Several components (especially DAOs and services) have vague responsibilities, risking architectural erosion.
- **Areas for Improvement:**
  - Clarify vague requirements (REQ-8, REQ-10, REQ-21, REQ-58, REQ-60) with measurable criteria.
  - Introduce offline-capable architecture (local storage, sync protocol, offline auth) and corresponding tests.
  - Define domain events and aggregate roots to enforce invariants and improve extensibility.
  - Add end-to-end user journey tests and hardware integration tests.

2 Content Extraction Results

2.1 Table of Contents and Sections

2.1.1 Extracted Table of Contents

Table 2: Extracted Table of Contents with Page Ranges

Section	Start Page	End Page
1 Executive Summary	3	4
1.1 Assessment Overview	3	3
1.2 Critical Findings	3	4
1.3 Report Quality Validation	4	4
2 Functional Domain Analysis	5	17
2.1 Authentication & Authorization	5	6
2.2 Transaction & Payment Management	7	8
2.3 Offline Operation & Resilience	9	10
2.4 Inventory & Product Management	10	11
2.5 Maintenance & Worker Operations	12	13
2.6 Architectural Overview	13	14
2.7 Architectural Strengths	14	15
2.8 Critical Weaknesses & How to Improve	14	16
2.9 Testing Quality	16	16
2.10 Testing Gaps	17	17
3 Cross-Cutting Concerns	18	18
3.1 Error Handling & Validation	18	18
3.2 Requirements Quality Issues	18	18
4 Conclusions	19	20
4.1 Overall Assessment	19	20
4.2 Critical Gaps	19	20
4.3 Final Verdict	19	20
Appendix: Complete Requirements Inventory	21	23

2.2 Section Classification

Table 3: Section Classification by Content Type

Section	Labels
1 Executive Summary	[]
1.1 Assessment Overview	[]
1.2 Critical Findings	[Requirements, Architecture, Test] (high-level gaps)

Continued on next page

Table 3 – continued from previous page

Section	Labels
1.3 Report Quality Validation	[Test] (meta-validation of report quality)
2 Functional Domain Analysis	[]
2.1 Authentication & Authorization	[Requirements, Architecture, Test]
2.2 Transaction & Payment Management	[Requirements, Architecture, Test]
2.3 Offline Operation & Resilience	[Requirements, Architecture, Test]
2.4 Inventory & Product Management	[Requirements, Architecture, Test]
2.5 Maintenance & Worker Operations	[Requirements, Architecture, Test]
2.6 Architectural Overview	[Architecture]
2.7 Architectural Strengths	[Architecture]
2.8 Critical Weaknesses & How to Improve	[Architecture, Requirements]
2.9 Testing Quality	[Test]
2.10 Testing Gaps	[Test]
3 Cross-Cutting Concerns	[]
3.1 Error Handling & Validation	[Requirements, Test]
3.2 Requirements Quality Issues	[Requirements]
4 Conclusions	[]
4.1 Overall Assessment	[]
4.2 Critical Gaps	[Requirements, Architecture, Test]
4.3 Final Verdict	[]
Appendix: Complete Requirements Inventory	[Requirements]

2.3 Requirements Extraction

A total of **62 requirements** were extracted from the Appendix and domain sections.

2.3.1 Requirements Quality Distribution

Table 4: Requirements Quality Assessment

Quality Level	Count	Percentage
Well-defined	49	79.0%
Needs Detail	6	9.7%
Vague/Unquantified	7	11.3%
Total	**62**	**100%**

2.3.2 Requirements by Type

Table 5: Requirements Classification

Type	**Count**	**Percentage**
Functional	42	67.7%
Non-Functional	9	14.5%
Constraint	5	8.1%
Goal/Background	6	9.7%
Total	**62**	**100%**

2.3.3 Sample of Extracted Requirements

Due to space, this table lists a representative subset; all 62 are included in the internal JSON inventory.

Table 6: Sample Extracted Requirements

ID	**Type**	**Description**	**Quality**
REQ-1	Functional	The system must support user authentication with email and password.	Well-defined
REQ-2	Functional	The system must allow user registration with role assignment.	Needs Detail (role model not fully specified)
REQ-3	Functional	The system must generate QR codes for machine access.	Well-defined
REQ-4	Functional	The system must provide product inventory management capabilities.	Needs Detail (operations and constraints not fully enumerated)
REQ-5	Functional	The system must support real-time inventory tracking.	Needs Detail (latency and consistency not quantified)
REQ-6	Functional	The system must manage wallet balances for customers.	Well-defined
REQ-7	Functional	The system must provide balance recharge functionality.	Well-defined
REQ-8	Functional	The system must support digital payment methods.	Vague/Unquantified (no providers or standards specified)

Continued on next page

Table 6 – continued from previous page

ID	Type	Description	Quality
REQ-9	Functional	The system must track transaction history.	Well-defined
REQ-10	Goal/background	The system should improve user experience.	Vague/Unquantified (no usability metrics)
REQ-18	Functional	The system must track local transactions while offline.	Needs Detail (storage and conflict rules unspecified)
REQ-19	Functional	The system must synchronize offline and online transactions.	Needs Detail (sync strategy and conflict resolution missing)
REQ-20	Functional	The system must provide an anonymous cash transaction fallback.	Needs Detail (limits and reconciliation rules missing)
REQ-21	Goal/background	The system should improve operational efficiency.	Vague/Unquantified
REQ-34	Functional	The system must provide structured authentication error responses.	Needs Detail (schema not defined)
REQ-35	Functional	The system must provide structured transaction error responses.	Needs Detail
REQ-45	Functional	The system must provide structured validation error responses.	Needs Detail
REQ-46	Non-functional	The system must validate input to prevent invalid or malicious data.	Well-defined
REQ-50	Constraint	The system must implement service layer orchestration.	Well-defined
REQ-53	Constraint	The system must follow a layered architecture separation.	Well-defined
REQ-58	Non-functional	The system should define usability metrics.	Vague/Unquantified
REQ-60	Functional	The system must provide remote maintenance capabilities.	Vague/Unquantified (scope unclear)
REQ-61	Functional	The system must manage machine connections.	Well-defined

Continued on next page

Table 6 – continued from previous page

ID	Type	Description	Quality
REQ-62	Functional	The system must support multiple user roles.	Well-defined

2.4 Use Case Extraction

The report explicitly names several use cases and implies others via coverage tables.

2.4.1 Use Case Summary

Table 7: Extracted Use Cases (Summary)

ID	Name	Type	Actors
UC-1	User Login	Explicit	User, System
UC-2	User Registration	Explicit	User, System, Admin
UC-3	Purchase Item	Explicit	Customer, System, Wallet, Payment Provider
UC-4	Recharge Wallet	Explicit	Customer, System, Payment Provider
UC-6	View Transaction History	Explicit	Customer, System
UC-8	Complete Maintenance Task	Explicit	Worker, System
UC-12	Update Item	Explicit	Admin, System
UC-13	Delete Item	Explicit	Admin, System
UC-14	Add Item	Explicit	Admin, System
UC-15	View Items	Explicit	Admin, System
UC-18	Remote Maintenance	Explicit	Worker, System, Vending Machine Hardware
UC-20	Register Machine	Implicit	Admin, System
UC-21	View Analytics Dashboard	Implicit	Admin, System
UC-22	Monitor Machine Status	Implicit	Admin, System

2.4.2 Detailed Use Case Descriptions (Selected)

UC-1: User Login

- **Actors:** User, System
- **Type:** Explicit
- **Main Flow (inferred from REQ-1 and tests):**
  1. User submits email and password to the login endpoint.

2. System validates credentials against stored user data.
3. On success, system establishes an authenticated session or returns a token.

- **Alternative Flows:**
  - Invalid credentials: System returns a structured authentication error (REQ-34).
  - Missing fields: System returns validation error (REQ-45).

UC-3: Purchase Item

- **Actors:** Customer, System, Wallet, Payment Provider, Vending Machine
- **Type:** Explicit
- **Main Flow (from Section 2.2):**
  1. Customer selects a product (REQ-12).
  2. System verifies wallet balance (REQ-6, REQ-7).
  3. System processes payment using digital methods (REQ-8).
  4. System records transaction (REQ-9, REQ-11, REQ-16).
  5. System triggers item dispensing (REQ-17).
- **Alternative Flows:**
  - Insufficient balance: System handles error and prevents purchase (REQ-14).
  - Out-of-stock item: System handles error and prevents purchase (REQ-15).
  - Payment gateway failure: System rolls back transaction and maintains data integrity.

UC-18: Remote Maintenance

- **Actors:** Worker, System, Vending Machine Hardware
- **Type:** Explicit
- **Main Flow (promised, but not fully supported):**
  1. Worker initiates remote maintenance session.
  2. System should communicate with machine hardware to perform diagnostics or control actions.
- **Alternative Flows:**
  - Hardware unreachable: System should report connection failure (REQ-36).
- **Architectural Status:** Not implementable with current architecture (no hardware abstraction, no IoT gateway).

2.5 Architecture Extraction

2.5.1 Architectural Pattern

The report describes a **six-layer layered architecture** with clear separation of concerns:

- Presentation
- Controller
- Service
- DAO
- Persistence
- Domain Model

2.5.2 Architecture Components

Table 8: Architecture Components Analysis

Component	Responsibility	Design Notes
Presentation Layer	UI components and user interaction (Web UI, mobile mockups, user interfaces)	Clear UI focus; technology-independent description.
Controller Layer (UserController, MachineController, TransactionController)	HTTP routing and input validation for respective domains	Good separation from business logic; no layer skipping observed.
Service Layer (CustomerService, AdminService, WorkerService)	Business logic orchestration for customer, admin, and worker operations	Responsibilities somewhat broad; recommendation to split by bounded context (PurchaseOrchestrationService, InventoryManagementService, PricingService).
DAO Layer (UserDao, TransactionDao, ItemDao, MachineDao)	Data access abstraction for users, transactions, items, and machines	Uses DAO pattern; current descriptions are generic ("manages data access") and should be refined per DAO.
Persistence Layer (JPA/Hibernate, DBManager, connection pools)	ORM mapping and database connection management	Clear infrastructure concern; supports PostgreSQL in production and H2 for tests.

Continued on next page

Table 8 – continued from previous page

Component	**Responsibility**	**Design Notes**
Domain Model (ConcreteVending-Machine, Transaction, Inventory, Wallet, MachineStatus)	Business entities and value objects encapsulating domain behavior	Rich domain model with behavior; aligns with Domain-Driven Design. Aggregate root enforcement currently weak.
Payment Gateway Integration	External payment provider integration for digital payments	Mentioned in tests and recommendations; provider and protocol unspecified due to vague REQ-8.
d		
Domain Events Infrastructure	Publish domain events such as ProductPurchased, BalanceRecharged, MaintenanceTaskCreated	Not yet implemented; recommended to improve extensibility & decoupling.

2.5.3 Architecture Analysis

Summary: The architecture follows a disciplined layered pattern with strong separation of concerns and a rich domain model. The main weaknesses are vague component boundaries, missing offline and hardware integration components, and lack of domain events and aggregate root enforcement.

Strengths:

- Clean layer separation; no layer-skipping violations.
- Strategic use of Builder, DAO, and Mapper patterns.
- Technology independence: database and API technologies can be swapped with limited impact.
- Rich domain entities encapsulating behavior.

Weaknesses:

- Vague DAO and service responsibilities risk monolithic classes.
- No components for offline storage, synchronization, or offline authentication.
- No hardware abstraction or IoT gateway for remote maintenance.
- No domain events infrastructure; aggregate root enforcement is weak.

2.6 Test Extraction

2.6.1 Test Type Distribution

Table 9: Test Distribution by Type

Test Type	Count	Percentage
Unit	45	71%
Integration	12	19%
System	6	10%
Total	**63**	**100%**

2.6.2 Test Coverage by Domain

Table 10: Domain-Level Test Coverage

Domain	Tests	Coverage	Notes
Authentication & Authorization	16	High	Covers valid/invalid credentials, input validation, error handling, business rules, security, authorization.
Transaction & Payment	18	High	Covers purchase flow, wallet operations, error handling, data integrity, payment integration.
Offline Operation	0	None	No tests for offline scenarios.
Inventory & Product Management	13	High	Covers CRUD, validation, error handling.
Maintenance & Worker Operations	5	Partial	Tests for UC-8 only; no tests for UC-18 (remote maintenance).
Cross-Cutting Error Handling	Several	Partial	Tests verify errors occur but not standardized response formats.

2.6.3 Representative Tests

Table 11: Representative Extracted Tests

ID	Type	Artifact	Coverage Hint
TEST-1	Unit	Authentication tests (valid/invalid credentials)	User Login success and failure scenarios (UC-1).
TEST-2	Unit	Registration tests (duplicate email)	User Registration business rules (UC-2).
TEST-10	Integration	Transaction rollback tests	Purchase Item error handling and data integrity (UC-3).
TEST-20	Unit	Inventory CRUD tests	Add/Update/Delete/View Items (UC-12–UC-15).
TEST-30	Unit	Maintenance task completion tests	Complete Maintenance Task (UC-8).
TEST-40	System	End-to-end purchase flow	Purchase Item main flow (UC-3).
TEST-50	System	Navigation tests	*Missing* according to Section 2.10; navigation flows not covered.
TEST-60	System	Hardware integration tests	*Missing*; no remote control or device communication tests.

3 Traceability Matrix

3.1 Coverage Summary

Table 12: Traceability Coverage Summary

Artifact Type	Total	Covered	Uncovered	Coverage %
Requirements	62	59	3	95.2%
Use Cases	18	15	3	83.3%
Components (layers)	6	6	0	100%
Tests	63	55 (mapped to UCs)	8 (orphan / infra-only)	87.3%

3.2 Requirements to Use Cases Mapping

Table 13: Requirements to Use Cases Traceability (Selected)

Req ID	Use Cases	Status	Rationale
REQ-1	UC-1	Covered	UC-1 (User Login) directly implements email/password authentication.
REQ-2	UC-2	Covered	UC-2 (User Registration) covers registration with role assignment.

Continued on next page

Table 13 – continued from previous page

Req ID	Use Cases	Status	Rationale
REQ-3	UC-3	Covered	QR code generation is part of purchase access flow.
REQ-4	UC-12, UC-13, UC-14, UC-15	Covered	Inventory CRUD use cases implement inventory management.
REQ-5	UC-3, UC-12–UC-15	Partial	Real-time tracking implied by purchase and inventory flows; latency not specified.
REQ-6	UC-3, UC-4	Covered	Wallet balance used in purchase and recharge flows.
REQ-8	UC-3, UC-4	Partial	Digital payments used, but providers/compliance unspecified.
REQ-9	UC-3, UC-6	Covered	Transaction history recorded and viewed.
REQ-10	UC-1–UC-4, UC-6	Partial	"Improved user experience" is a goal across flows; no metrics.
REQ-18	–	UNSUPPORTED	No offline use cases defined.
REQ-19	–	UNSUPPORTED	No sync use cases defined.
REQ-20	–	UNSUPPORTED	No anonymous cash fallback use case.
REQ-22	UC-8	Covered	Worker task assignment realized in maintenance task flows.
REQ-24	UC-8	Covered	Maintenance notifications tied to task lifecycle.
REQ-34	UC-1	Partial	Authentication errors occur and are tested, but format unspecified.
REQ-35	UC-3	Partial	Transaction errors occur and are tested, but format unspecified.
REQ-45	UC-1–UC-4	Partial	Validation errors tested, but no standard schema.
REQ-46–REQ-49	Multiple	Covered	Validation requirements realized across authentication, transaction, and inventory flows.
REQ-50–REQ-53	All	Covered	Architectural requirements realized by six-layer design.
REQ-60	UC-18	Partial	Remote maintenance use case exists but is not implementable with current architecture.

3.3 Use Cases to Architecture Mapping

Table 14: Use Cases to Architecture Traceability (Selected)

UC ID	UC Name	Components	Status	Rationale
UC-1	User Login	Presentation, UserController, CustomerService/AdminService/WorkerService, UserDao, Domain Model (app_user)	Covered	Standard layered flow from UI to domain and persistence.
UC-2	User Registration	Presentation, UserController, Services, UserDao, Domain Model	Covered	Similar to UC-1 with additional role assignment logic.
UC-3	Purchase Item	Presentation, TransactionController, CustomerService, TransactionDao, TransactionItemDao, Wallet, Inventory, ConcreteVendingMachine	Covered	Purchase orchestration, persistence, and vending operations are mapped to components.
UC-4	Recharge Wallet	Presentation, TransactionController, CustomerService, Wallet, Payment Gateway, TransactionDao	Covered	Wallet operations and payment integration realized.
UC-6	View Transaction History	Presentation, TransactionController, TransactionDao	Covered	Read-only access to transaction history.
UC-8	Complete Maintenance Task	Presentation, WorkerService, TaskMapper, Domain Model (Worker, Task)	Covered	Task lifecycle managed in service and domain layers.
UC-12–UC-15	Inventory CRUD	Presentation, AdminService, ItemDao, MachineDao, InventoryMapper, Inventory	Covered	CRUD operations mapped to admin service and DAOs.
UC-18	Remote Maintenance	Presentation, WorkerService	Not Implementable	No hardware abstraction, IoT gateway, or device protocol components.

3.4 Use Cases to Tests Mapping

Table 15: Use Cases to Tests Traceability (Selected)

UC ID	UC Name	Main Flow	Alt Flows	Status & Details
UC-1	User Login	Tested	Partial	Main flow and invalid credentials tested. Missing standardized error response format tests (REQ-34).
UC-2	User Registration	Tested	Tested	Registration success, duplicate email, and validation errors tested.
UC-3	Purchase Item	Tested	Tested	Main purchase flow, insufficient balance, out-of-stock, and rollback scenarios tested.
UC-4	Recharge Wallet	Tested	Partial	Recharge success and some error scenarios tested; payment security tests recommended.
UC-6	View Transaction History	Tested	N/A	Read-only flow tested; no alternative flows required.
UC-8	Complete Maintenance Task	Tested	Partial	Task completion and error scenarios tested; no tests for remote aspects.
UC-12–UC-15	Inventory CRUD	Tested	Tested	CRUD operations and validation/error scenarios comprehensively tested.
UC-18	Remote Maintenance	Not Tested	Not Tested	No tests exist for remote maintenance or hardware communication.
Multi-UC Journeys	Register → Recharge → Purchase → History	Not Tested	Not Tested	Section 2.10 notes missing end-to-end workflows and navigation tests.

3.5 Orphan Artifacts

3.5.1 Orphan Requirements

- **REQ-18:** Local transaction tracking during offline – no use cases, components, or tests.
- **REQ-19:** Offline-online synchronization – no use cases, components, or tests.
- **REQ-20:** Anonymous cash transactions fallback – no use cases, components, or tests.

3.5.2 Orphan Use Cases

- **UC-18:** Remote Maintenance – mapped to REQ-60 but not to any hardware-related components or tests.
- **Implicit navigation flows:** Role-based routing and multi-step journeys are not explicitly modeled as use cases.

3.5.3 Orphan Tests

- Some infrastructure tests (e.g., DB migration, connection pooling) are not clearly mapped to specific use cases but support non-functional requirements.
- No tests exist for offline behavior or hardware integration, leaving REQ-18–REQ-20 and REQ-60 effectively untested.

4 Feature-Based Validation

4.1 Overview

The report states that it was validated against 16 software engineering documentation standards, satisfying 11 of them (68.8% coherence). This section aligns those findings with universal best practices.

Table 16: Feature-Based Validation Summary

Metric	**Value**
Total Knowledge Base Features (assumed)	16
Features Covered in Report	11
Features Not Covered	5
Coverage Percentage	**68.8%**

4.2 Covered Features (Selected)

Table 17: Covered Universal Features (Examples)

ID	**Feature**	**Category**	**Evidence from Report**
F-1	Clear architectural pattern	Architecture	"The system implements a six-layer architecture following classic separation of concerns principles." (Section 2.6)
F-2	Layer responsibilities defined	Architecture	Table in Section 2.6 lists each layer with responsibilities and key components.
F-3	Traceability from requirements to tests	Documentation	"Automated Traceability: Complete automated traceability from requirements through tests." (Architectural Strengths)
F-4	Test pyramid adherence	Testing	"Unit Tests (71%), Integration (19%), System (10%)... The distribution follows the test pyramid pattern." (Section 2.9)
F-5	Error handling strategy identified	Cross-Cutting	Section 3.1 discusses inconsistent error response formats and recommends a standardized JSON schema.
F-6	Requirements inventory	Documentation	Appendix lists all 62 requirements with category and coverage status.

Continued on next page

Table 17 – continued from previous page

ID	Feature	Category	Evidence from Report
F-7	Identification of vague requirements	Requirements	Section 3.2 lists REQ-10, REQ-21, REQ-58, REQ-8, REQ-60 as vague and recommends measurable criteria.
F-8	Use of design patterns	Architecture	Section 2.6 and 2.7 describe Builder, DAO, and Mapper patterns and their rationale.
F-9	Domain-driven design elements	Architecture	"The domain model follows Domain-Driven Design principles with rich entities... and clear compositional relationships." (Section 2.6)
F-10	Risk identification	Project Management	Executive summary and Section 4.2 list three critical gaps and their impact.
F-11	Methodology validation	Documentation	"This report was validated against 16 established software engineering documentation standards, achieving 68.8% coherence." (Section 1.3)

4.3 Uncovered / Weak Features

Table 18: Uncovered or Weak Universal Features (Inferred)

ID	Feature	Category	Description
UF-1	Performance requirements	Performance	No explicit performance metrics (e.g., response times, throughput) are specified.
UF-2	Security threat analysis	Security	No systematic threat model or mitigation strategy is documented.
UF-3	Deployment architecture	Architecture	No deployment topology (nodes, networks, environments) is described.
UF-4	Monitoring and observability	Maintainability	No logging/monitoring/alerting strategy beyond basic error logging.
UF-5	Formal non-functional requirements	Requirements	Usability and efficiency goals are vague and lack measurable acceptance criteria.

4.4 Checklist Compliance Example

4.4.1 Feature: Clear Architectural Pattern

Category: Architecture

Description: The architecture pattern is explicitly identified, justified, and consistently applied.

Table 19: Checklist Compliance for "Clear Architectural Pattern"

ID	Checklist Item	Status	Explanation
1.1	Architecture pattern is identified	True	Section 2.6 explicitly states a six-layer architecture with separation of concerns.
1.2	Pattern is justified	True	Benefits such as testability, technology substitution, and no layer skipping are described.
1.3	Layer responsibilities are clear	Partial	High-level responsibilities are clear, but individual DAO and service responsibilities are still vague.
1.4	Pattern is consistently applied	True	No layer-skipping violations are reported; controllers delegate to services, services to DAOs.

4.4.2 Feature: Requirements Quality Management

Category: Requirements

Description: Requirements are complete, non-ambiguous, and testable, with explicit handling of vague items.

Table 20: Checklist Compliance for "Requirements Quality Management"

ID	Checklist Item	Status	Explanation
2.1	Requirements are uniquely identified	True	Appendix lists REQ-1 to REQ-62 with unique IDs.
2.2	Requirements are categorized	True	Each requirement has a category (Authentication, Inventory, etc.).
2.3	Vague requirements are flagged	True	Section 3.2 explicitly lists vague requirements and their issues.
2.4	All requirements are testable	False	Several requirements (REQ-10, REQ-21, REQ-58, REQ-8, REQ-60) lack measurable criteria, making them not fully testable.

5 Detailed Analysis

5.1 Requirements Quality Assessment

5.1.1 Vague/Unquantified Requirements

Vague/Unquantified Requirements (7 total):

- **REQ-10:** "Improved user experience" – No usability metrics (e.g., task completion time, SUS score).
  Recommendation: Define measurable UX goals such as "80% of users complete a purchase in under 60 seconds".
- **REQ-21:** "Operational efficiency improvements" – No KPIs (e.g., refill frequency, downtime).
  Recommendation: Specify metrics like "reduce refill visits by 20%" or "99.5% machine uptime".
- **REQ-58:** "Usability metrics" – Mentioned but not defined.
  Recommendation: Provide concrete metrics and target values.
- **REQ-8:** "Digital payment methods support" – No providers, standards, or flows.
  Recommendation: Name providers (e.g., Stripe), standards (PCI-DSS), and integration model.
- **REQ-60:** "Remote maintenance capabilities" – Scope (diagnostics vs. full control) unclear.
  Recommendation: Clarify scope and associated safety constraints.
- **REQ-10, REQ-21 (repeated in Section 3.2):** Highlighted as lacking quantifiable targets.

5.1.2 Requirements Needing Detail

- **REQ-18, REQ-19, REQ-20:** Offline behavior requirements lack details on storage technology, sync frequency, conflict resolution, and security.
  Recommendation: Define offline storage (e.g., SQLite), sync protocol, and conflict rules.
- **REQ-34, REQ-35, REQ-45:** Error response requirements lack JSON schema.
  Recommendation: Define a standard error object with code, message, details, timestamp, and request ID.

5.2 Use Case Completeness Analysis

5.2.1 Explicit vs. Implicit Use Cases

Table 21: Use Case Categorization

Category	Count	Percentage
Explicit (named in sections)	15	83.3%
Implicit (inferred from requirements)	3	16.7%
Total	**18**	**100%**

Observations:

- Core flows (authentication, purchase, inventory, maintenance tasks) are explicitly modeled.

- Offline and anonymous cash flows are not modeled as use cases.
- Navigation and multi-step user journeys are not formalized as use cases.

5.2.2 Alternative Flow Coverage

- **Strong:** UC-1, UC-2, UC-3, UC-4, UC-12–UC-15 include error scenarios (invalid credentials, insufficient balance, out-of-stock, validation errors).
- **Weak:** UC-18 (Remote Maintenance) lacks any detailed flows; only high-level promise.
- **Missing:** Offline-related alternative flows (e.g., purchase during network outage) are absent.

5.3 Architecture Quality Evaluation

5.3.1 Design Principles Assessment

Table 22: Architecture Quality Metrics

Principle	Status	Notes
Separation of Concerns	Good	Six-layer architecture with no layer skipping.
Loose Coupling	Good	Controllers, services, and DAOs are decoupled via interfaces.
High Cohesion	Moderate	Some services and DAOs have broad responsibilities.
Single Responsibility	Moderate	Component boundaries need clearer documentation.
Extensibility	Moderate	Lack of domain events and aggregate roots limits extensibility.
Resilience	Weak	No offline or hardware resilience components.

5.3.2 Key Architectural Gaps

- **Offline Operation:** No local storage, sync protocol, or offline auth components.
- **Remote Maintenance:** No hardware abstraction layer, IoT gateway, or device protocol (e.g., MQTT).
- **Aggregate Root Enforcement:** Inventory can be modified directly, bypassing machine capacity rules.
- **Domain Events:** No event infrastructure for decoupled notifications, analytics, or audit logging.

5.4 Test Coverage Analysis

5.4.1 Strengths

- Strong adherence to test pyramid with majority unit tests.
- Error-first testing: many failure scenarios are explicitly tested.
- Use of H2 test database enables fast feedback loops.

5.4.2 Gaps

- No tests for offline behavior (REQ-18–REQ-20).
- No tests for remote maintenance or hardware communication (REQ-60, UC-18).
- No end-to-end multi-use-case journeys (register $\rightarrow$ recharge $\rightarrow$ purchase $\rightarrow$ history).
- No navigation tests for role-based routing.

6 Recommendations

6.1 Priority 1: Critical Items

1. **Implement Offline Operation Support**
    - Address REQ-18, REQ-19, REQ-20.
    - **Action:** Design local storage (e.g., SQLite), define sync protocol and conflict resolution, and model offline use cases and tests.
2. **Make Remote Maintenance Implementable**
    - Address REQ-60 and UC-18.
    - **Action:** Clarify scope (diagnostics vs. full control), design hardware abstraction, IoT gateway, and communication protocol; add integration tests.
3. **Clarify Component Responsibilities**
    - Reduce risk of architectural erosion.
    - **Action:** Document precise responsibilities for each DAO and service; split services by bounded context where needed.

6.2 Priority 2: Important Improvements

1. **Standardize Error Response Formats**
    - Address REQ-34, REQ-35, REQ-45.
    - **Action:** Define JSON error schema and update tests to assert structure.
2. **Introduce Domain Events and Aggregate Roots**
    - Improve extensibility and data consistency.
    - **Action:** Implement DomainEvent interface, event publisher, and aggregate root enforcement for ConcreteVendingMachine and Inventory.
3. **Quantify Vague Requirements**
    - Address REQ-8, REQ-10, REQ-21, REQ-58, REQ-60.
    - **Action:** Add measurable acceptance criteria and update traceability.

6.3 Priority 3: Nice-to-Have Enhancements

1. Add end-to-end user journey tests and navigation tests.
2. Document deployment architecture and monitoring strategy.
3. Extend security analysis with a basic threat model and mitigations.

6.4 Summary Checklist

Table 23: Action Item Summary

Priority	**Items**	**Indicative Effort**
Critical (P1)	3	5–10 days
Important (P2)	3	5–8 days
Nice-to-Have (P3)	3	5–7 days
Total	**9**	**15–25 days**