TED UNIVERSITY

CMPE 492

Test Plan Report

Team Members :

- **Dilara Yargıcı** - 20290006896
- **Berna Tanrıverdi** - 63295214002
- **Aslı Dölek** - 67915217862
- **Yaşar Mert Dirican** – 13630073160

Supervisor :

- **Prof. Dr. Gökçe Nur Yılmaz**

Jury Members :

- **Dr. Öğr. Üyesi Mehmet Evren Coşkun**
- **Dr. Öğr. Üyesi Elif Kurtaran Özbudak**

Table of Contents

1. Introduction

This Test Plan Report outlines the strategic framework for verifying and validating the **PriceLink** system, an innovative IoT-based smart price tag solution designed for modern retail environments. The primary objective is to ensure that the integration between the E-paper display hardware, the mobile management application, and the centralized cloud server operates with 100% data integrity and minimal latency.

The scope of this testing process encompasses the entire ecosystem of PriceLink, including the firmware performance of the ESP32-based tags, the responsiveness of the Flutter-based mobile interface, and the security of the RESTful API communications. By identifying potential failure points early in the development lifecycle, this plan ensures that price updates are reflected accurately on the shelves, battery life is optimized through efficient deep-sleep cycles, and the user experience remains intuitive for retail staff.

2. Features to be Tested

This section identifies the functional features of the PriceLink system that are subject to testing. In order to ensure traceability and systematic coverage, the features are grouped into modules and assigned unique identifiers. The list below focuses only on what is to be tested within the system. The identified test items were derived from the system's functional requirements, administrator scenarios, use cases, and subsystem/service definitions presented in the project analysis and high-level design documents.

2.1 User Authentication and Session Management

AUTH-001: Secure administrator login

AUTH-002: Verification of valid username and password credentials

AUTH-003: Rejection of invalid login credentials

AUTH-004: Authorized access to the management panel after successful login

AUTH-005: Administrator logout and session termination

AUTH-006: Redirection to the login page after logout

AUTH-007: Restriction of unauthorized access to administrative functions

2.2 Dashboard and System Overview

DSH-001: Display of the main dashboard after administrator login

DSH-002: Retrieval of current tag status information from the database

DSH-003: Visualization of tag connection status on the dashboard

DSH-004: Visualization of battery status on the dashboard

DSH-005: Display of warning and status indicators for problematic tags

DSH-006: Real-time reflection of tag updates on the management panel

2.3 Product Management

PRD-001: Adding a new product to the system

PRD-002: Storing product information including name, category, SKU/barcode, price, and store location

PRD-003: Saving newly added product records in the database

PRD-004: Viewing product details from the product list

PRD-005: Editing existing product information

PRD-006: Updating product information in the database

PRD-007: Deleting a product from the system

PRD-008: Removing the product–tag relationship when a product is deleted

2.4 Price Management

PRC-001: Updating the price of a selected product

PRC-002: Recording the new product price in the database

PRC-003: Transmitting updated price information to the related electronic tag

PRC-004: Displaying the updated price on the E-Ink tag

PRC-005: Instant price synchronization between management panel, backend, gateway, and tag

PRC-006: Verification that current active price information is correctly stored for the product

PRC-007: Preservation of original price information for rollback purposes

2.5 Bulk Campaign Price Update

CMP-001: Access to the bulk price update function from the management panel

CMP-002: Selection of products by category for campaign updates

CMP-003: Selection of campaign type such as discount rate or fixed new price

CMP-004: Display of preview information before applying bulk price updates

CMP-005: Entry of campaign start and end time information

CMP-006: Validation of campaign parameters before execution

CMP-007: Application of bulk price updates to all relevant products

CMP-008: Broadcasting updated campaign prices to associated tags

CMP-009: Display of campaign prices on reachable tags

CMP-010: Automatic reversion to original prices after campaign expiration

CMP-011: Handling of invalid or expired campaign definitions

CMP-012: Retry option for tags that fail to receive campaign updates

2.6 Product–Tag Pairing and Reassignment

APP-001: Initial pairing of a physical electronic tag with a product

APP-002: Reading tag identity through QR code scanning

APP-003: Verification of tag ID and product information during pairing

APP-004: Storage of product–tag pairing data in the database

APP-005: Successful association of the correct product with the correct tag

APP-006: Reassignment of an existing tag to a new product

APP-007: Display of current product information linked to a selected tag

APP-008: Comparison of old product and new product before reassignment

APP-009: Updating tag–product mapping after reassignment

APP-010: Transmission of new product details and current price to the reassigned tag

APP-011: Automatic application of pending reassignment when an offline tag reconnects

2.7 Tag Monitoring and Status Tracking

MON-001: Monitoring of tag battery level

MON-002: Monitoring of tag connection status

MON-003: Display of tag states such as Connected, Update Received, Disconnected, and Low Battery

MON-004: Generation of low-battery warning when battery level falls below the defined threshold

MON-005: Detection of disconnected tags

MON-006: Indication of pending update status for offline tags

MON-007: Tracking whether a tag has successfully received an update

MON-008: Dashboard presentation of hardware status and warning notifications

2.8 Logging and Traceability

LOG-001: Logging of administrator login activity

LOG-002: Logging of product creation operations

LOG-003: Logging of product editing operations

LOG-004: Logging of product deletion operations

LOG-005: Logging of individual price update operations

LOG-006: Logging of bulk campaign update operations

LOG-007: Logging of product–tag reassignment operations

LOG-008: Recording username, timestamp, and previous/new state information for each action

LOG-009: Storage of operation history for traceability and auditing purposes

LOG-010: Logging of failed validation and update failure events

2.9 Data Synchronization and Update Propagation

SYN-001: Writing updated data to the cloud database

SYN-002: Propagation of database updates to the gateway service

SYN-003: Propagation of gateway updates to the target smart tag

SYN-004: Acknowledgment flow from smart tag back to the system

SYN-005: Real-time update of interface status after acknowledgment

SYN-006: Consistent synchronization between cloud, gateway, dashboard, and smart tag layers

2.10 Offline Handling and Operational Continuity

OFF-001: Preservation of previous price when update delivery fails

OFF-002: Queueing of pending updates during connectivity loss

OFF-003: Synchronization of queued operations after connectivity restoration

OFF-004: Automatic delivery of pending updates to reconnected tags

OFF-005: Retention of the last valid price on the E-Ink display during power loss

OFF-006: Prevention of blank or misleading display after battery depletion or power interruption

2.11 Customer-Facing Information Display

CUS-001: Display of product name or short description on the PriceLink tag

CUS-002: Display of current price on the PriceLink tag

CUS-003: Display of campaign price, when applicable

CUS-004: Display of campaign duration information

CUS-005: Consistency between tag price and POS/system price

CUS-006: Readable presentation of product information to the customer

2.12 Summary of Test Modules

In summary, the features to be tested in PriceLink are grouped under authentication, dashboard operations, product management, price management, bulk campaign handling, product–tag pairing, tag monitoring, logging, synchronization, offline resilience, and customer-facing information display. These modules collectively represent the core functional behavior of the system and provide the basis for systematic functional testing.

3. Test Methodology

To guarantee hardware dependability, smooth cloud synchronization, and a positive user experience for store staff, PriceLink's testing process uses a multi-layered approach. The following tactics will be used to confirm the functionality and performance of the system.

3.1 Unit Testing

Unit testing aims to confirm that discrete components and isolated code blocks operate as expected before system integration.

- **Hardware Component Testing:** Certain features of the TagController and DisplayManager classes will be tested using mock data.
    - **Test Case ID: UT-01 (Screen Refresh):** Confirming that updateEInkDisplay() produces an accurate double-type price value free of artifacts.
    - **Test Case ID: UT-02 (Power Management):** Ensuring that deepSleepMode() appropriately turns off the CPU while maintaining the RTC's ability to initiate wake-up cycles.
- **Backend Logic Testing:** Verifying that the PriceService and AuthManager modules properly handle API calls and verify employee credentials.
- **Database Interface Testing:** Confirming that FirebaseConnector obtains product document fields from Firestore and appropriately toggles the is_pending flag.

3.2 Integration Testing

Integration testing assesses how well the various PriceLink ecosystem levels communicate and exchange data.

- **IoT-to-Cloud Communication:** Testing the entire line of synchronization between Firebase Cloud and the Smart Tag.
    - **Test Case ID: IT-01 (Heartbeat Sync):** Confirming that reportStatus() accurately modifies the battery_level and last_sync timestamp fields in the database.
    - **Test Case ID: IT-02 (Message Delivery):** Ensuring that "PRICE_UPDATE" JSON payloads are successfully forwarded to the subscribed Gateway topic by the MQTT broker.
- **Real-time Dashboard Updates:** Verifying whether Firestore modifications are instantly reflected on the DashboardUI via WebSocket listeners without the need for manual page refreshes.

3.3 Performance Testing

Performance tests measure how fast and efficient the technology is in real-world scenarios.

- **Response Time & Latency:** Calculating the total time from the moment an administrator clicks "Update Price" on the dashboard till the actual E-ink screen refreshes.
    - **Test Case ID: PT-01 (System Latency):** Ensuring that, in spite of the "Deep Sleep" cycles, the entire update flow (Sequence Flows 1–10) finishes within reasonable time constraints.
- **Power Efficiency:** Verifying that the asynchronous event-driven paradigm satisfies the ESP32-based tags' battery life target of three to four years.
- **Scalability Testing:** To make sure the MQTT broker manages concurrent broadcasts to several tags without data loss, bulk campaign updates are simulated.

3.4 User Acceptance Testing (UAT)

UAT guarantees that the PriceLink Management Panel offers a dependable user experience and satisfies the practical requirements of store employees.

- **Functional Usability:** Testing the ProductTagPairingUI and ProductManagementUI to make sure employees can quickly connect new products to physical tags using QR codes.
    - **Test Case ID: UAT-01 (Admin Workflow):** To verify that administrators can successfully filter transaction logs and receive clear visual warnings for low battery situations thanks to the showLowBatteryWarning() function.
- **Operational Transparency:** Verifying that when a tag finishes its refresh cycle, the system displays a clear "Update Received" (ACK) status on the dashboard

4. Test Environment

4.1 System Architecture

The PriceLink system is designed as a distributed, event-driven IoT architecture consisting of three primary layers: the presentation layer, application layer, and edge layer.

At the presentation layer, the **Management Dashboard (Frontend)** provides an interface for administrators to manage products, update prices, monitor tag statuses, and control campaign operations. This layer communicates with the backend using secure HTTP and WebSocket protocols to ensure real-time data synchronization.

The **Application Layer (Backend)** is implemented using a cloud-based infrastructure and is responsible for handling core business logic. It processes price update requests, manages product and tag data, maintains transaction logs, and coordinates communication between system components. A real-time database is used to propagate updates instantly across all connected components.

The **Edge Layer** consists of the IoT Gateway and Smart Tags. The gateway acts as a bridge between the cloud backend and physical devices. It receives update messages from the backend and transmits them to Smart Tags using lightweight wireless communication protocols. Smart Tags display product prices using low-power E-Ink technology and send acknowledgment (ACK) messages back to the system to confirm successful updates.

The system follows an **event-driven communication model**, where updates are triggered and propagated instantly without continuous polling. This architecture minimizes latency and reduces unnecessary network overhead.

The overall system architecture is illustrated in **Figure 1**.

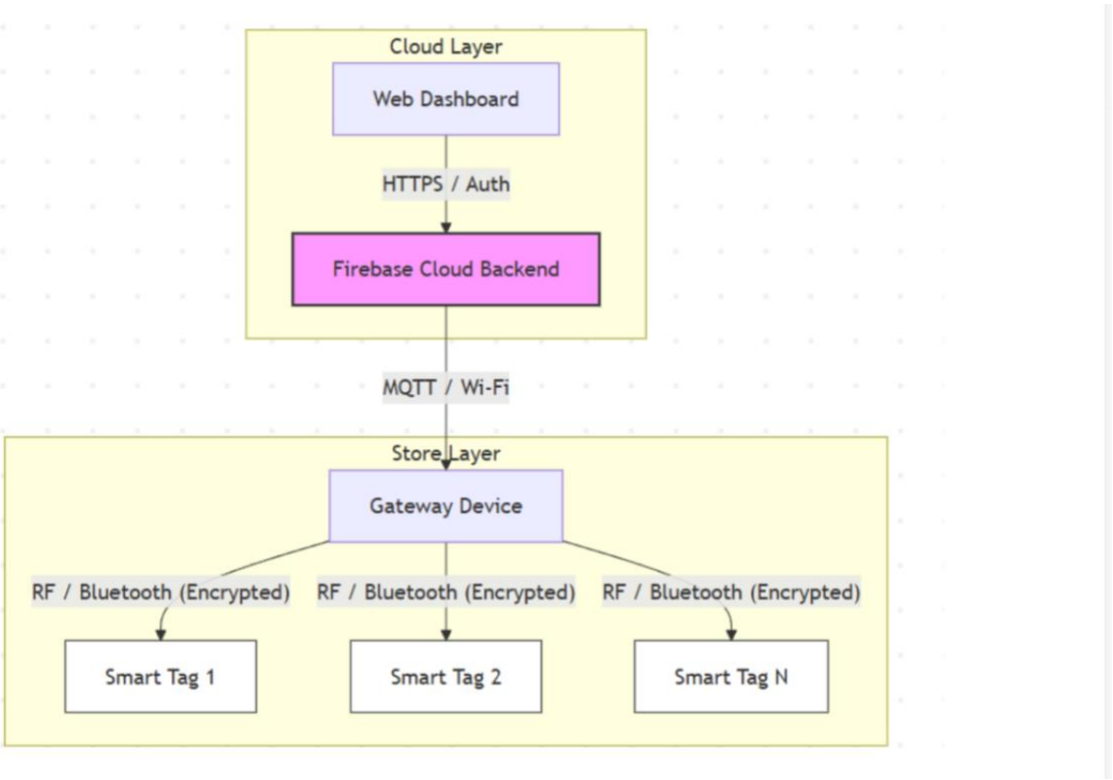

Figure 1: Overall System Architecture of the PriceLink System

The testing environment is designed to closely simulate real-world deployment conditions. Both cloud-based and edge components are included to ensure that system behavior can be validated under realistic operational scenarios. While the frontend and backend components are tested in a live cloud environment, IoT components such as Smart Tags are tested using both physical ESP32 devices and simulated MQTT clients when necessary.

4.2 Software Components

The PriceLink system is developed using a combination of modern web technologies, cloud services, and embedded system components.

Frontend :

- **React.js** is used to develop the Management Dashboard.
- Provides a responsive and interactive user interface.
- Communicates with the backend through REST APIs and WebSocket connections.

Backend :

- **Firebase (Google Cloud Platform)** is used as the primary backend infrastructure.
    - **Cloud Firestore**: NoSQL database for storing product, tag, and transaction data.
    - **Firebase Authentication**: Handles secure login and user access control.
    - **Firebase Cloud Functions**: Executes server-side logic and processes events.

Communication Technologies :

- **MQTT Protocol**:
    - Used for backend-to-gateway and gateway-to-tag communication.
    - Enables lightweight, publish/subscribe-based messaging suitable for IoT systems.
- **WebSocket**:
    - Used for real-time communication between backend and frontend.
    - Ensures immediate synchronization of dashboard data.

Embedded System (Hardware Layer) :

- **ESP32 Microcontroller**:
    - Used in both gateway and smart tag devices.
- **E-Ink Display Modules**:
    - Used for energy-efficient and persistent display of product prices.
- Firmware is developed using **C/C++ with the Arduino framework**.

Data Format :

- **JSON** is used as the standard data format for communication between system components, ensuring consistency and interoperability.

Test Data :

Test data includes sample product records, pricing scenarios (such as bulk updates and discount campaigns), and simulated tag identifiers. Both valid and invalid inputs are used to verify system robustness, input validation mechanisms, and error handling capabilities.All components are tested in a controlled and consistent environment to ensure reproducibility of test results.

4.3 Testing Tools & Frameworks

A variety of tools and frameworks are used to support testing activities across different layers of the PriceLink system.

Development Environments :

- **Visual Studio Code (VS Code)**:
  - Used for frontend and backend development.
  - Provides debugging and code management capabilities.
- **Arduino IDE**:
  - Used for programming and testing ESP32-based devices.

API and Backend Testing :

- **Postman**:
  - Used to test REST API endpoints.
  - Validates request-response structures and backend functionality.
- **Firebase Console**:
  - Used to monitor real-time database updates.
  - Verifies correct data propagation and backend-triggered events.

Real-Time Communication Testing :

- **MQTT Testing Tools (e.g., MQTT Explorer / Mosquitto)**:
  - Used to publish and subscribe to MQTT topics.
  - Verifies message delivery, topic structure, and acknowledgment (ACK) responses.
- **Browser Developer Tools (Chrome DevTools)**:
  - Used to monitor WebSocket connections and network traffic in real time.

Hardware Testing :

- **Serial Monitor (Arduino IDE)**:
    - Used to observe device logs and debug communication between components.
- **ESP32 Debugging Tools**:
    - Used to analyze connectivity, signal strength, and device behavior.

Version Control :

- **Git & GitHub**:
    - Used for version control and collaborative development.
    - Ensures traceability of changes throughout the testing process.

Network Environment :

The system operates over standard internet connectivity using secure HTTPS and WebSocket protocols. Local communication between gateway and smart tags is performed over Wi-Fi using MQTT-based messaging, ensuring efficient and low-latency data exchange.

5. Test Schedule

To maintain a rigorous quality assurance standard, the PriceLink testing phase is divided into five distinct stages, aligned with our project milestones.

Phase	Methodology	Objectives	Duration
Stage 1	Unit Testing	Verification of individual functions in the backend and sensor readings in the firmware.	Week 9 - 10
Stage 2	Integration Testing	Testing the end-to-end data flow from the Mobile App to the Database and then to the IoT tags via Wi-Fi/MQTT.	Week 11
Stage 3	Performance Testing	Stress testing the server with concurrent price update requests and measuring tag wake-up latency.	Week 12
Stage 4	UAT & Beta	On-site simulation of retail scenarios (e.g., rapid price changes, bulk product syncing) to validate user flow.	Week 13
Stage 5	Final Validation	Regression testing to ensure bug fixes haven't introduced new issues; final documentation of results.	Week 14 - Final

7. Roles and Responsibilities

The PriceLink project is managed under a structured division of labor where each member leads a specific architectural layer. Technical accountability is maintained through designated documentation and diagramming tasks.

1. **Dilara Yargıcı (Project Manager & Database Lead**)

- Project Management: Responsible for overall project coordination, test schedule maintenance, risk mitigation strategies, and final report integrity.

- Technical Responsibility: Database Architecture and System Integration.

- Diagrammatic Output: Entity Relationship (ER) Diagram, mapping the relationships between products, smart tags, and transaction logs within Firebase.

- Written Deliverables: Introduction, System Trade-offs (Technical justification for protocol selection over REST), and Glossary.

2. **Yaşar Mert Dirican (Embedded Systems & Hardware Lead)**

- Technical Responsibility: IoT Logic and Display Management.

- Diagrammatic Output: Hardware State Machine Flowchart, illustrating the tag's lifecycle: deep-sleep cycles, wake-up triggers, and data fetching.

- Written Deliverables: Documentation of hardware libraries (GxEPD2, WiFi.h) and technical specifications for TagController class methods.

3. **Aslı Dölek (Communication & Backend Logic Lead)**

- Technical Responsibility: Communication Protocols and Server-side Logic.

- Diagrammatic Output: System Architecture (Package) Diagram, visualizing the positioning of the chosen communication protocol (e.g., MQTT Broker) within the ecosystem.

- Written Deliverables: Interface Guidelines (Data formats and rules for the chosen protocol) and Engineering Standards.

4. **Berna Tanrıverdi (Frontend & User Flow Lead)**

- Technical Responsibility: User Interface and System Process Flow.

- Diagrammatic Output: Sequence Diagram, detailing the end-to-end path of a price update starting from the dashboard to the E-ink tag.

- Written Deliverables: Documentation of DashboardUI classes and the comprehensive Attributes (Variables) table.

8. Risks and Contingency Plans

The PriceLink team has identified the following potential risks and established mitigation strategies:

- **R1: Network Latency & Connectivity Drops:** Retail environments often have crowded Wi-Fi networks which may cause "Packet Loss" during price updates.
    - *Contingency:* Implementation of an Acknowledgement (ACK) protocol where the tag sends a confirmation signal back to the server. If no ACK is received, the system will retry the update automatically.
- **R2: Hardware Power Constraints:** Intensive testing might drain the E-paper tag batteries faster than predicted in production.
    - *Contingency:* Power-consumption profiling will be prioritized in Stage 3, and deep-sleep optimizations will be adjusted if battery life falls below the 1-year target.
- **R3: Data Desynchronization:** A mismatch between the price displayed on the mobile app and the physical tag.
    - *Contingency:* A "System Audit" feature will be developed to allow the QA Lead to ping all tags and verify their current display state against the database records.
- **R4: Scope Creep during Testing:** New feature requests from jury feedback might delay the testing schedule.
    - *Contingency:* A strict freeze on new features will be implemented starting from Week 11 to focus solely on stabilizing existing functions.

9. References

[1] IEEE Std 1016-2009, “IEEE Standard for Software Design Descriptions.” This standard was utilized to define the structure and organization of the Package, Class, and Sequence Diagrams.

[2] Google, “Firebase Documentation,” [Online]. Available: https://firebase.google.com/docs. (Technical reference for Firestore NoSQL structures and real-time listeners).

[3] OASIS, “MQTT Version 5.0 Specification,” 2019. [Online]. Available: https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html. (Primary protocol for asynchronous communication with Smart Tags).

[4] I. Fette and A. Melnikov, “The WebSocket Protocol (RFC 6455),” 2011. [Online]. Available: https://datatracker.ietf.org/doc/html/rfc6455. (Standard for full-duplex communication between Backend and Dashboard).

[5] Espressif Systems, “ESP32 Technical Reference Manual,” [Online]. Available: https://www.espressif.com/en/support/documents/technical-documents.

[6] Meta, “React – A JavaScript library for building user interfaces,” [Online]. Available: https://react.dev.

[7] OWASP Foundation, “OWASP Security Best Practices,” [Online]. Available: https://owasp.org. (Applied for TLS/WSS encryption and secure data transmission).

[8] ACM, “Code of Ethics and Professional Conduct,” [Online]. Available: https://www.acm.org/code-of-ethics. (Followed for responsible data handling and user privacy).

[9] PostgreSQL Global Development Group, “PostgreSQL Documentation,” [Online]. Available: https://www.postgresql.org/docs/. (Reference for relational logic and transactional audit logging).

[10] Arduino, “Arduino Documentation and Libraries (GxEPD2, WiFi.h),” [Online]. Available: https://docs.arduino.cc.

[11] Postman Inc., “Postman API Platform Documentation,” [Online]. Available: https://learning.postman.com.

[12] Microsoft, “Visual Studio Code Documentation,” [Online]. Available: https://code.visualstudio.com/docs.