TED UNIVERSITY

CMPE 491

SENIOR PROJECT

High Level Design Report

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

Table of Contents

1. Introduction

1.1 Purpose of the System

The PriceLink project has recently placed a strong emphasis on the excessive expenses and environmental waste brought on by the retail industry's extensive usage of paper labels. Additionally, store staff' workload is greatly increased by the continuous manual altering of product labels. This issue is resolved by the PriceLink project, which makes it possible to execute all label pricing activities correctly from a single panel. By avoiding disparities between shelf and checkout prices, it also seeks to increase customer satisfaction.

1.2 Design Goals

The system architecture is based on the following:

- **Power Efficiency:** Labels made using e-ink technology are targeted to have an average battery life of 3 years.
- **Consistency & Reliability:** In the event of unpredictable internet and power outages, the current state of the system and data must be securely preserved.
- **Scalability:** In the event of unpredictable internet and power outages, the current state of the system and data must be securely preserved.

1.3 Acronyms and Abbreviations

Abbreviation	Description
ESL	Electronic Shelf Label
HLD	High-Level Design
MQTT	Message Queuing Telemetry Transport
API	Application Programming Interface

1.4 Overview

The High-Level Design study provides a detailed explanation of the whole system architecture and operational principles of the proposed PriceLink system.

- **Part 1**, This study explains the PriceLink project's overall goal, effectiveness, and scalability. Additionally, it explains the acronyms for the technical phrases that are used throughout the paper.
- **Part 2**, This part displays a software synopsis of the retail industry's current labeling system.

- **Part 3**, The software architecture is elaborated in this section. The system's primary elements—the Web Panel, Firebase Backend, Gateway, and Smart Tags—are described in detail. It also involves security procedures and data management.
- **Part 4**, It describes the characteristics and functional requirements of the subsystems contained within the system.
- **Part 5**, This section includes explanations of technical terms such as MQTT, ESP32, and E-Ink so that the report can be easily understood by everyone who reads it.
- **Part 6 (References)**, This section contains a list of sources used in preparing the report.

2. Current Software Architecture

Currently unfortunately there is no integrated digital software architecture governing the shelf labeling process in the main target retail environments. The existing workflow is predominantly analog and old school manual, creating a significant disconnect between the digital inventory systems and the physical store shelves. This architectural gap results in several critical bottlenecks like:

- Data Silos and Inconsistency: While product prices are centrally managed in POS or ERP systems, this data does not automatically propagate to the shelf edge. Physical tags act as isolated as Data Silos completely disconnected from the real-time database. This leads to pricing discrepancies where the price displayed on the shelf differs from the price charged at the register.
- Synchronization Latency: The propagation of a price update involves a high-latency manual workflow: printing, cutting, and physically replacing paper tags.
  This Synchronization Latency prevents retailers from implementing dynamic pricing strategies or reacting swiftly to market fluctuations due to Türkiye’s economic conditions.
- Lack of State Verification: In the current manual architecture, there is no digital feedback loop. Central management lacks visibility into the actual state of the shelf, meaning there is no system-level verification (ACK signal) to confirm that a price change has been successfully applied.

3. Proposed Software Architecture

3.1 Overview

Three primary, synchronous components make up the PriceLink system. The Web Dashboard, which is located at the top layer, allows the administrator to view and modify

product details. All system data is kept in the Firebase Cloud Backend (Firestore), which is the intermediary layer. There are two parts to the bottom layer: 1. Cloud data is transformed into signals by gateway devices; 2. Smart Tags (E-Ink) employ these signals to visualize prices.

3.2 Subsystem Decomposition

The PriceLink system consists of 4 main subsystems:

1. **Web Dashboard:** This is the user interface where the admin can log in using their username and password to make price updates and perform other operations.
2. **Firebase Cloud Backend:** It is a cloud-based central database containing all data (e.g., transaction logs, product and user information).
3. **Gateway Device:** This gear facilitates Wi-Fi and RF/Bluetooth communication and connectivity between tags and a cloud-based database server.
4. **Smart Tag (Smart Label):** It is an e-ink display that shows product prices and is a low-power end-user device.

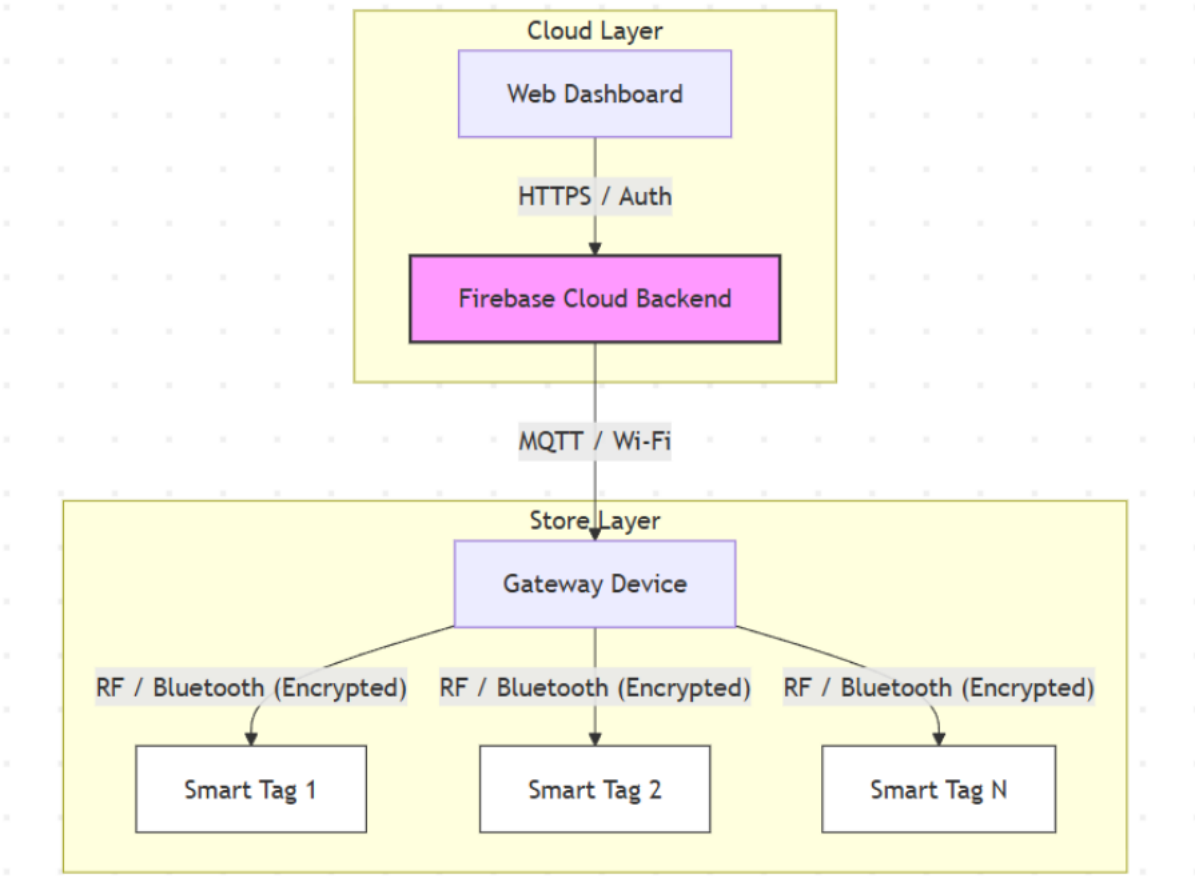

Figure 1: Subsystem Diagram

3.3 Hardware/Software Mapping

To address the limitations of the legacy system, PriceLink provide a modern, three-tier IoT architecture. The system adopts a Cloud-Native and Serverless approach, prioritizing scalability and energy efficiency.

1. Cloud Backend (Google Firebase Platform): Instead of traditional monolithic SQL servers, the system leverages Google Firebase as its scalable backend.

- Data Management (NoSQL): We will utilize Cloud Firestore, a NoSQL document database, to store product details, tag configurations, and transaction logs. Its flexible document-oriented structure allows for the management of dynamic product attributes without rigid schema constraints.
- Hosting & Logic: The management dashboard will be deployed with Firebase Hosting, while background processes (such as bulk updates) are going to handle serverlessly by Cloud Functions.

2. IoT Gateway (Edge Layer):The Gateway serves as the communication bridge between the cloud and the physical tags.

- Hardware: An ESP32 microcontroller or Raspberry Pi acts as the local store coordinator.
- Function: Unlike passive bridges, the Gateway maintains a persistent, bi-directional WebSocket connectionwith Firestore. It listens for real-time changes in the cloud and translates these digital commands into local RF signals to wake up specific Smart Tags.

3. Smart Tag (End Node): The end-user device is designed with a "power-first" philosophy.

- Hardware: A custom PCB featuring an ESP32 microcontroller and an integrated E-Ink Display Driver.
- Power Efficiency: The firmware is optimized for Deep Sleep cycles. The device remains dormant and only wakes up upon receiving a trigger signal from the Gateway, ensuring a multi-year battery life.
- Display Technology: We utilize E-Ink technology for its bistable nature; the image remains visible on the screen even when power is completely cut off, ensuring fail-safe pricing visibility.

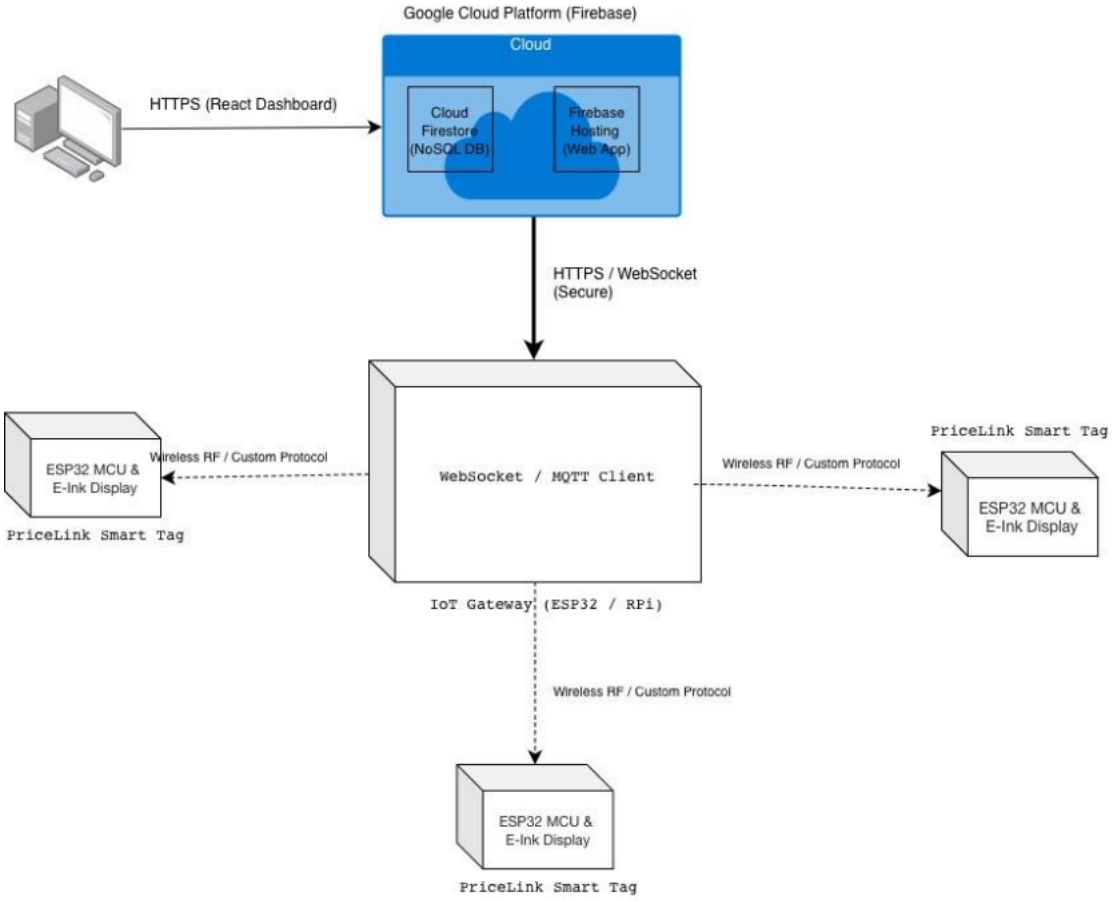

Figure 2: Deployment diagram illustrating the cloud-native architecture of PriceLink, utilizing Google Firebase for backend services and ESP32-based gateways for local tag communication.

3.4 Persistent Data Management

The PriceLink system adopts **Google Firebase Firestore** as its primary persistent data management solution. Firestore is a **cloud-native, NoSQL document-oriented database** that offers real-time synchronization, high availability, and seamless scalability, which are critical requirements for a distributed IoT system operating across multiple retail stores. Firestore is selected due to its strong integration with Firebase services, real-time data listeners, offline persistence capabilities, and its suitability for event-driven architectures commonly used in IoT systems.

Data Model Overview

The database is structured using **collections and documents**, enabling flexible schema evolution as system requirements grow.

Core Collection: Products

Each document under the Products collection represents a unique retail product and contains the following key attributes:

- **product_id**: Unique identifier for the product (Firestore document ID).
- **name**: Human-readable product name.
- **price**: Current active price displayed on the tag.
- **original_price**: Base price used for campaign rollback.
- **linked_tag_id**: Identifier of the associated PriceLink E-Ink tag.
- **category_id**: Reference to the product category.
- **store_location**: Physical location of the product in the store.
- **last_updated**: Timestamp of the most recent price modification.

Supporting Collections

- **Tags**: Stores tag-level metadata such as battery level, connection status, MAC address, and last update timestamp.
- **Categories**: Enables hierarchical grouping for bulk operations.
- **TransactionLogs**: Maintains immutable records of all administrative actions for traceability and auditing.

Consistency and Synchronization

Firestore's real-time listeners ensure that any price update committed by the backend is instantly propagated to:

- The Management Panel dashboard
- The Gateway service
- All subscribed PriceLink Tags

This guarantees **eventual consistency** across cloud, gateway, and edge devices while maintaining system responsiveness.

3.5 Access Control & Security

Security in PriceLink is designed using a **defense-in-depth** approach, addressing both user authentication and fine-grained authorization.

Authentication

User authentication is implemented using **Firebase Authentication**. Only registered administrators and authorized store personnel can access the PriceLink Management Panel.

- Secure login via email/password credentials

- Encrypted session handling
- Token-based authentication for backend communication

This mechanism ensures that all system interactions are performed by verified identities.

Authorization

Authorization rules are enforced at both **application** and **database** levels.

- Only users with **Admin** or **Authorized Staff** roles can:
    - Modify product prices
    - Perform bulk campaign updates
    - Reassign tags
- Read-only access is enforced for monitoring dashboards where appropriate.

Firestore **Security Rules** ensure that:

- Unauthorized users cannot write to sensitive collections such as Products or TransactionLogs
- Each operation is validated against user roles before execution

Data Security

- All communication between Management Panel, Backend, Gateway, and Tags is encrypted using secure protocols (HTTPS, WPA2).
- Sensitive credentials are never stored in plaintext.
- All price changes are logged to ensure non-repudiation and accountability.

3.6 Global Software Control

The global control flow of PriceLink is designed as an Event-Driven Architecture, fundamentally differing from traditional "Polling" based systems.

- Real-time Listeners over Polling: In standard web systems, clients typically poll the server periodically for updates. PriceLink utilizes Firebase’s Real-time Listeners. The Gateway subscribes to specific database collections, ensuring the system remains reactive and does not generate unnecessary network traffic until a change occurs.
- Propagation Flow:
    1. Trigger: When an administrator updates a price on the Dashboard, a "Write" event occurs in the Firestore database.
    2. Push: This event instantly triggers a data push to the connected Gateway via the active socket, reducing synchronization latency to milliseconds.

3. Broadcast: The Gateway processes the payload and broadcasts the update via the local RF protocol to the target Smart Tag.

- Asynchronous Acknowledgment: The system operates asynchronously to ensure a non-blocking user experience. The Dashboard does not freeze during an update; instead, it waits for an Acknowledgment (ACK)signal that travels back from the Tag → Gateway → Cloud, updating the UI status to "Green" only when the physical screen has confirmed the refresh.

3.7 Boundary Conditions

Boundary conditions are critical in IoT-based systems due to environmental uncertainty, connectivity variability, and power limitations. Instead of classical shutdown scenarios, PriceLink focuses on **offline resilience and power-loss tolerance**.

Initialization

When a PriceLink Tag or Gateway device boots:

- It establishes a secure connection with Firebase.
- Retrieves the **most recent valid product data** from the cloud.
- Synchronizes its local state before entering active operation.

This guarantees consistency even after unexpected restarts.

Connectivity Loss (Offline Mode)

If internet connectivity is lost:

- The **Gateway continues operating** using Firestore's offline persistence.
- Previously cached price data remains accessible.
- Pending updates are queued locally.

Once connectivity is restored:

- Queued operations are automatically synchronized with the cloud.
- Tags marked as *Pending Update* receive their latest data.

This design ensures uninterrupted store operations during network outages.

Power Loss

PriceLink Tags utilize **E-Ink bistable display technology**, which retains the last rendered image **without requiring power**.

- If the battery is depleted:
    - The last valid price remains visible on the screen.

- No blank or misleading display occurs.

- This prevents customer confusion and legal risks caused by missing prices.

The system therefore achieves **fail-safe visibility**, a critical requirement in retail environments.

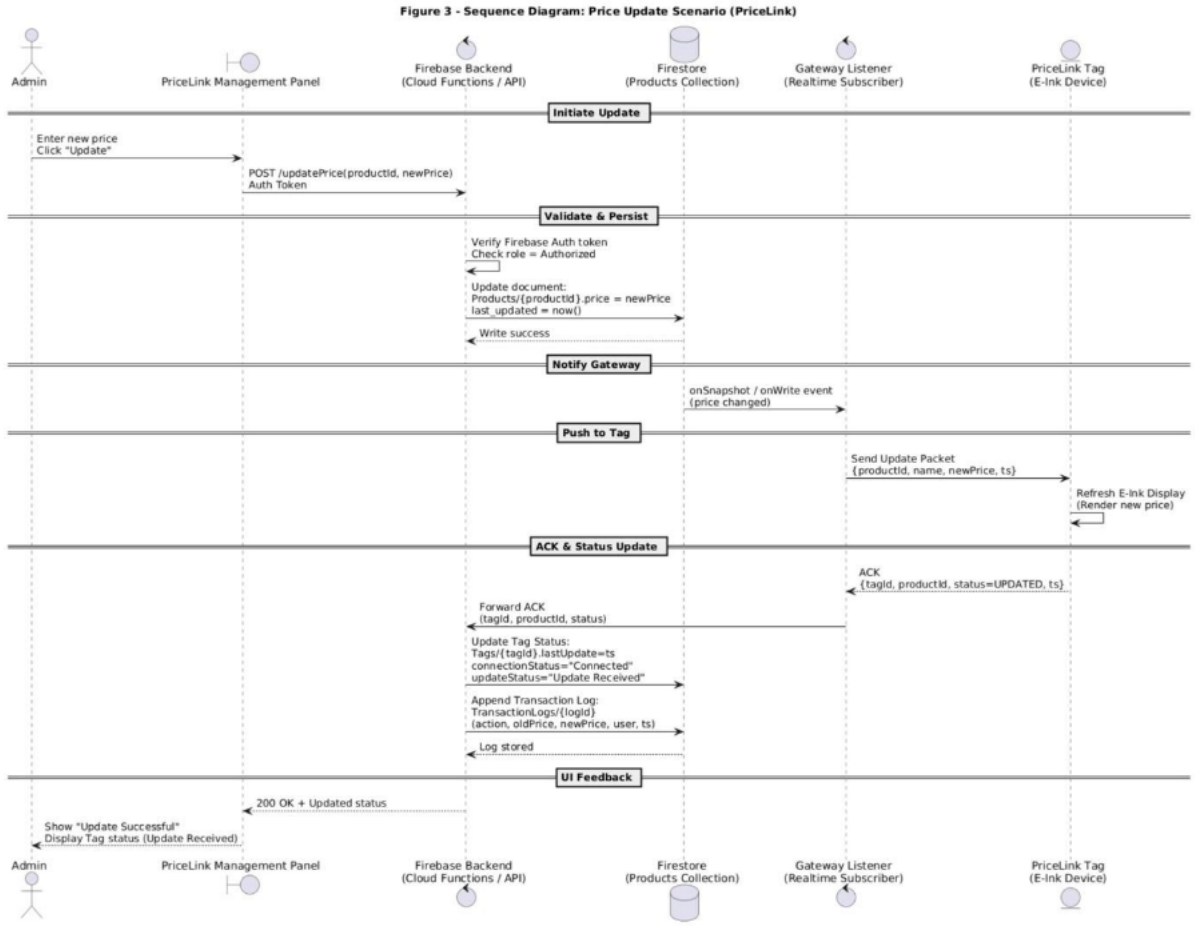

Figure 3: Sequence Diagram: Price Update Scenario (PriceLink)

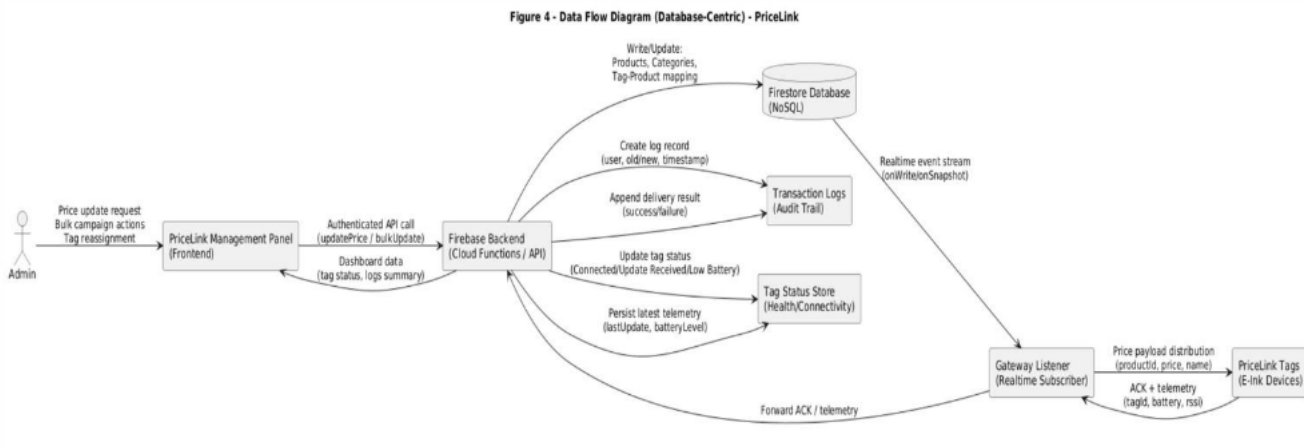

Figure 4: Data Flow Diagram(Database - Centric) – PriceLink

4. Description

This section describes the core functionalities of the PriceLink system using the Input → Process → Output approach. The system enables real-time data flow between electronic shelf labels used in the store environment and a centralized cloud platform, supporting price management, hardware monitoring, and product–tag pairing operations.

4.1 Price Management Service

The Price Management Service allows store staff to update the price of a product by scanning its barcode or selecting it from a product list. The newly entered price information is transmitted to the relevant electronic shelf label in real timevia the centralized system and displayed on the e-ink screen.

Input

- Product information (barcode or product ID)
- New price value

Process

- Verification of the selected product in the system
- Storing the new price information in the database
- Sending the updated price information to the corresponding electronic tag in real time

Output

- Display of the updated price on the electronic shelf label

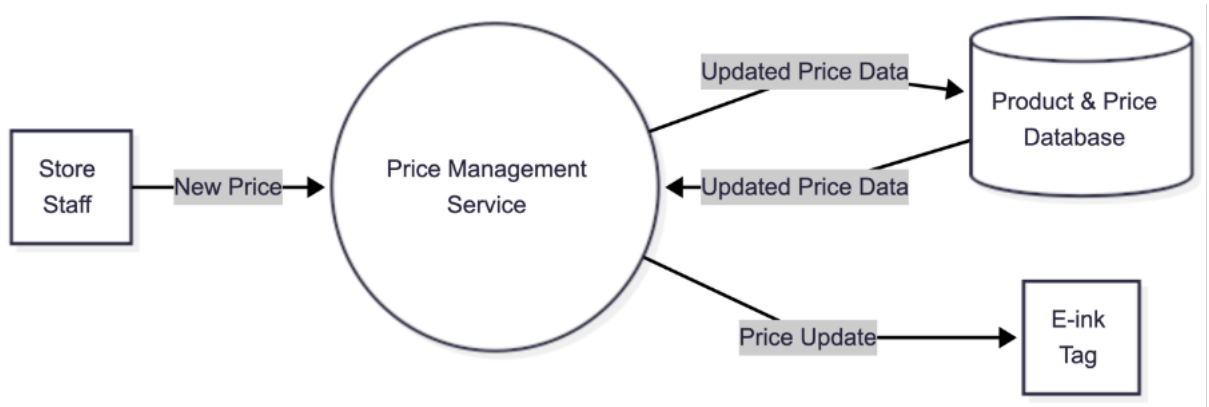

Figure 5: Data Flow Diagram (Price Management Service)

4.2 Tag Monitoring Service

Description

The Tag Monitoring Service continuously monitors the hardware status of electronic shelf labels used in the store, including battery level and connection status. This service enables real-time tracking of tag health conditions.

Feature

- When the battery level of a tag falls below 20%, a warning message is displayed on the management dashboard.

Input

- Battery level data received from electronic tags
- Connection status information

Process

- Analysis of hardware status data by the system
- Checking predefined critical thresholds

Output

- Display of hardware status and warning notifications on the dashboard

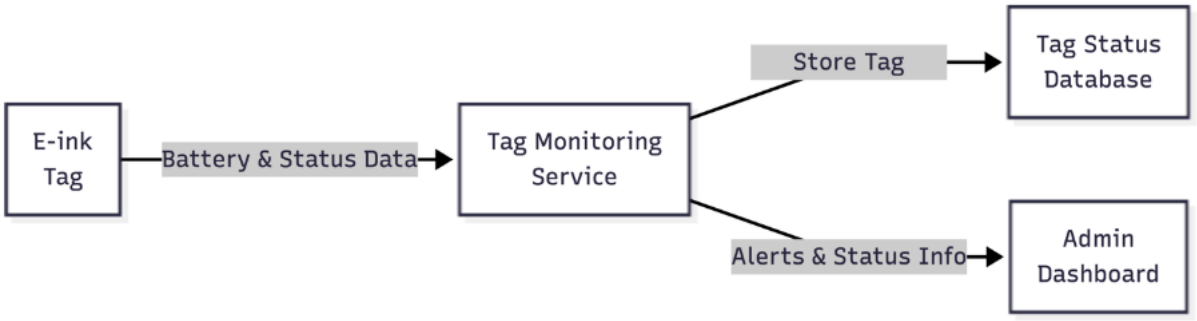

Figure 6: Data Flow Diagram (Tag Monitoring Service)

4.3 Product Pairing Service

Description

The Product Pairing Service enables the association of a physical electronic shelf label with a product stored in the database by scanning a QR code. This pairing ensures that correct product information is assigned to the correct electronic tag.

Input

- Tag ID obtained from QR code scanning
- Product information (product ID)

Process

- Verification of tag and product information
- Storing the product–tag pairing information in the database

Output

- Successful pairing of the product with the corresponding electronic shelf label

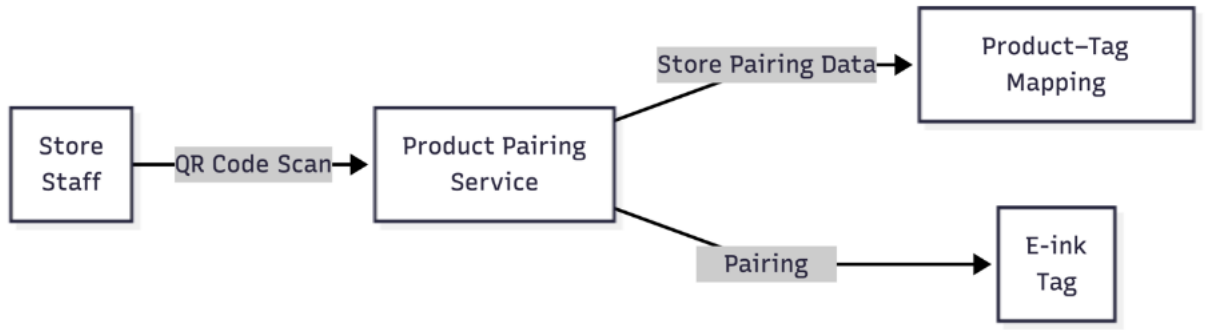

Figure 7: Data Flow Diagram (Product Pairing Service)

5. Glossary

- **MQTT:** Machine-to-machine communication is facilitated by the standards-based messaging protocol or set of rules known as MQTT.
- **Bistable Display:** E-Ink is a screen technology that only uses energy while the image is changing; it doesn't use energy when the image is still.
- **ESP32:** It is a microcontroller used in gateway or tag controllers that has Bluetooth and Wi-Fi capabilities.
- **Firebase Firestore:** Google provides a NoSQL cloud database that uses documents that resemble JSON to store data.

- **WPA2 Encryption:** It is an encryption standard that guarantees the security of wireless networks.

6. References

[1] Amazon Web Services. (2023). *Real-time data processing architectures*. https://aws.amazon.com

[2] E Ink Holdings Inc. (2023). *E-ink display technology overview*. https://www.eink.com

[3] Google. (2025). *Gemini* [Large language model]. https://gemini.google.com

[4] GS1. (2023). *Barcode standards and QR code specifications*. https://www.gs1.org

[5] Kendall, K. E., & Kendall, J. E. (2019). *Systems analysis and design* (10th ed.). Pearson Education.

[6] Pressman, R. S., & Maxim, B. R. (2020). *Software engineering: A practitioner's approach* (9th ed.). McGraw-Hill Education.

[7] Yourdon, E. (1989). *Modern structured analysis*. Prentice Hall.