

Intelligent Agro-Based Retail Monitoring System: A Systematic Literature Review and System Design

P. Anlet Pamila Suhi¹, Punith², Manasa² & Deepa²

¹Assistant Professor, Department of ISE

Sri Sairam College of Engineering, Anekal, Bangalore, Karnataka

²Final Year, BE (ISE), Sri Sai Ram College of Engineering, Anekal, Bangalore

DOI: doi.org/10.34293/iejcsa.v4i5.172

Abstract — Agricultural retail forms the last operational link between producers and consumers, and its efficiency has a direct bearing on food security and on how much edible produce is lost before it ever reaches a shopping cart. For decades, this link has been managed through periodic, manual shelf inspection—an approach that is labor-intensive, inconsistent across staff, and structurally reactive, since spoilage is normally noticed only after a batch has already degraded. This review surveys eight recent software- and hardware-oriented approaches to agro-retail and agricultural monitoring, spanning automated freshness tracking, database-driven decay modeling, food-spoilage classification research, web-based supply-chain dashboards, dynamic markdown pricing, multi-tenant retail platforms, spatial floor-mapping systems, and cloud-based crop-cultivation frameworks. Read together, these works show a consistent shift away from static, spreadsheet-driven inventory logs and toward continuous, predictive, and increasingly visual monitoring. At the same time, nearly every reviewed approach depends on the completeness and accuracy of manual data entry at the point of receiving, and only one line of work attempts to translate predictive output into an intuitive, colour-coded physical layout of the store. Building on these findings, this paper presents the design of an Intelligent Agro-Based Retail Monitoring System: a three-tier, software-only platform that combines a Python analytics engine for automated data cleaning and shelf-life prediction, a conversational orchestration layer capable of interpreting natural-language and voice queries, and a React.js dashboard that renders inventory health onto an interactive, colour-coded storefront map. The resulting architecture is intended to move agro-retail operations from speculative, labor-bound management toward real-time, evidence-based decision-making.

Keywords: Agro-retail monitoring, computer vision, produce freshness classification, predictive inventory analytics, shelf-life estimation, conversational AI agents, retail floor mapping, food-waste reduction.

INTRODUCTION

The retail distribution of fresh agricultural produce is shaped by a constraint that most other retail categories do not face: the product itself is decaying from the moment it is harvested. In conventional grocery and agro-retail settings, keeping track of that decay still falls mainly to floor staff who walk the aisles and judge freshness by eye. This works, up to a point, but it is slow, subjective, and—critically—reactive. A batch is usually flagged only once visible spoilage has already set in, by which time the retailer has already absorbed most of the loss. Compounding this, demand for fresh produce is volatile and difficult to

forecast from static records alone, so stockouts and overstocking tend to occur side by side within the same store.

A parallel shift is underway in how such systems are designed to be used. Rather than menu-driven dashboards that assume a technically fluent operator, newer platforms increasingly expose their functionality through natural-language and voice interaction, so that a shop owner with no formal technical training can ask a plain-language question and receive a chart, a summary, or a flow diagram in response. This review draws on eight recent works spanning both the retail-analytics side of this problem (freshness tracking, decay modeling, dashboarding, and dynamic pricing) and the underlying infrastructure side (multi-tenant platforms, floor-mapping systems, and cloud-based agricultural monitoring), in order to ground the design of an Intelligent Agro-Based Retail Monitoring System capable of addressing both dimensions at once.

These two sides of the problem have historically been treated as separate research tracks. Retail-analytics work focuses on what happens once a batch has arrived at the store—how it is tracked, priced, and displayed—while field-level monitoring work focuses on the conditions under which a crop was grown and transported. Few of the works reviewed attempt to connect the two, even though the freshness a shelf-monitoring system observes is, in large part, a downstream consequence of the storage and handling conditions a field-level system already measures. The economic stakes are also not trivial: fresh produce carries some of the thinnest margins in general retail, and any inefficiency in tracking or pricing stock is absorbed almost immediately as write-off cost. Even a modest improvement in how early a degrading batch is detected—measured in hours rather than days—can meaningfully change how much stock is sold at full price, discounted in time, or simply thrown away.

A. Objectives of This Review

This review pursues four goals. First, to organize the reviewed approaches by their primary design emphasis—freshness prediction, inventory-decay modeling, dashboard visualization, dynamic pricing, platform scalability, spatial mapping, or field-level agricultural monitoring. Second, to compare the advantages and structural limitations reported for each approach. Third, to identify the gaps that most directly constrain a deployable, store-level monitoring platform. Fourth, to translate those gaps into concrete architectural decisions for the proposed system.

B. Contributions

- A thematic synthesis of eight recent works, grouped by design emphasis and compared along predictive depth, physical actionability, accessibility, and deployment cost.
- A set of five research gaps, ordered by how foundational they are to a deployable store-level platform, each tied to a specific design response.

- The design of a three-tier, software-only Intelligent Agro-Based Retail Monitoring System that combines self-cleaning data ingestion, shelf-life prediction, a conversational and voice-driven interface, and a colour-coded storefront map.
- A staged evaluation plan describing how the proposed design can be validated against real store data in future work.

C. Organization of the Paper

This paper is organized as follows. Section II reviews the literature, covering the operational background, the review methodology, a thematic synthesis of the eight reviewed sources, and their comparative analysis; Section III distills the resulting research gaps; Section IV states the problem addressed; Section V introduces the proposed system, Section VI describes its architecture, and Section VII sets out its methodology and workflow; Section VIII covers implementation requirements, the evaluation plan, and limitations; and Section IX concludes.

LITERATURE REVIEW

A. Background and Theoretical Foundations

1) Operational Challenges in Fresh-Produce Retail

Three recurring obstacles surface across the literature. The first is the labor intensity of manual inspection: someone must physically walk each aisle, on a schedule frequent enough to catch early degradation, which is difficult to sustain across a large store or many branches. The second is the purely reactive nature of visual inspection—by design, it identifies problems only after they are visible, leaving little room for preemptive action such as early markdown or shelf reorganization. The third is the disconnect between raw inventory data and an actionable, physical view of the store: even where digital records exist, they are typically presented as rows in a table rather than as a layout a floor worker can act on directly.

These obstacles compound rather than operate independently. A store that relies on infrequent manual inspection is also relying on staff to mentally log what they see rather than recording it durably, so the same batch may be silently re-inspected by different staff on different shifts without anyone building a consistent picture of how quickly it is deteriorating. Even where receiving and sales records are digitized, a manager reviewing a spreadsheet of expiry dates has no real-time way to tell a stocker which specific shelf to check first. Any credible automated system must therefore address labor-intensive detection, reactive timing, and a data layer disconnected from physical action together.

2) Technical Evolution of Agro-Retail Monitoring

The approaches reviewed here fall loosely into three generations. The earliest digitizes inventory records but leaves the interpretation of freshness entirely to staff—essentially a spreadsheet replacing a paper ledger. A second generation introduces

mathematical decay modeling and time-series forecasting on top of digitized receiving and point-of-sale data, so that expiration windows can be estimated without a person inspecting each batch. The most recent generation, which this review emphasizes, layers computer vision, machine-learned classification, and—increasingly—large language model (LLM) driven orchestration on top of that predictive core, so that store operations can be queried conversationally and visualized as an interactive floor plan.

Each generation inherits the weaknesses of the one before it unless those weaknesses are deliberately addressed. Digitized ledgers still suffer from entry errors; decay models still depend on those ledgers; and conversational layers are only as trustworthy as the data and tools beneath them. The proposed system is therefore designed so that each layer defends the one above it: validation protects the predictor, the predictor feeds the map, and a constrained tool interface prevents the language model from acting on anything other than verified data.

3) Perishability and Shelf-Life Modeling

Shelf life is not a fixed property of a crop but a function of its condition at harvest, the time elapsed since harvest, and the temperature, humidity, and handling it has experienced since. Classical approaches reduce this to a printed best-before date; more refined approaches model quality as a decaying quantity, commonly with first-order or exponential forms in which the rate of quality loss depends on storage conditions. The survey summarized in [3] catalogues such formulas alongside classification methods that score freshness from observable variables. For a retail platform the practical lesson is that a single static label is a poor unit of analysis: two batches of the same crop with the same arrival date may reach the same quality threshold days apart if one is stored in a cooler and the other on an open display.

4) Motivation for Combining Retail and Field-Level Approaches

Although cultivation and retail shelf management are usually studied separately, they share a common object: a perishable batch whose condition changes continuously over time. A cultivation-side system that tracks soil, weather, and handling conditions generates the earliest data points in a decay curve that a retail-side system later has to estimate from receiving date and storage conditions alone. Bringing insight from field-level monitoring into a retail platform helps clarify which storage and handling variables are most predictive of shelf-life, and reinforces the case for treating decay prediction, rather than static shelf-life labels, as the correct unit of analysis.

5) Conversational Agents and LLM Orchestration in Retail

Large language models trained to follow instructions [9], built on the transformer architecture [11], have made it practical to interpret free-form questions and map them onto structured operations. In supply-chain settings, recent work explores multi-agent LLM

systems for inventory management [12], LLM-based extraction of unstructured data in agricultural logistics [13], and text-to-SQL interfaces that let non-programmers query databases in plain language [14]. A broader systematic review of artificial intelligence and big-data analytics in supply chains is given in [10]. Together these studies suggest that the language model is best used as an intent router and explainer rather than as a source of facts: the facts should come from the database and from deterministic analytics, with the model deciding which function to call and how to phrase the result.

This division of labor is central to the proposed design. A language model that is allowed to invent numbers is unacceptable in an inventory setting, whereas a model restricted to selecting among a small set of verified tools—and to summarizing what those tools return—can add accessibility without adding unverifiable claims.

6) *The Shop Owner as the Central End User*

Earlier tools were largely built for a technically literate administrator—someone comfortable configuring a database schema, reading SKU-level statistics, or setting up a rules engine for markdown thresholds. The small or mid-sized agro-retail owner who most needs such a tool is rarely that person: their day is split between receiving deliveries, serving customers, and managing staff. Designing for this owner as the primary user has direct architectural consequences, favoring conversational and voice-driven interaction, visual summaries over raw tables, and a floor-plan view over a spreadsheet, all of which are reflected in the system proposed in Sections V to VII.

The owner's questions are also narrow and repetitive: what is expiring today, which section needs attention, how much was sold this week, which product contributes the most margin. Because the question space is small, a tool-based conversational design can cover most real needs with a modest set of well-tested functions, rather than requiring open-ended reasoning over arbitrary data.

B. Review Methodology

1) *Corpus and Search Scope*

Eight studies addressing agro-retail freshness tracking, inventory decay modeling, food-spoilage classification, retail dashboarding, dynamic pricing, multi-tenant retail platforms, spatial floor mapping, and cloud-based agricultural monitoring were selected for detailed review. Each was assessed for its stated architecture, its reported advantages, and the limitations the original authors themselves acknowledged. Eight additional works [9]–[16] are used as supporting context for language-model orchestration, supply-chain analytics, and perishable-food monitoring, but are not part of the core comparison.

2) *Inclusion Criteria*

A source was included if its primary focus was an automated or semi-automated approach to tracking, predicting, or visualizing the condition of fresh or perishable retail

inventory, or a closely related field-level agricultural monitoring problem with a directly transferable architecture (e.g., sensor- and vision-based crop monitoring). Works that addressed only logistics routing, or only consumer-facing e-commerce, were excluded because their central object is not the condition of the perishable batch.

3) Extraction Framework

For each source, four fields were extracted: (i) the core technical approach; (ii) its principal operational advantages; (iii) its acknowledged disadvantages or limitations; and (iv) how directly its architecture maps onto the design of a store-level, software-only monitoring platform. Where a source reported a limitation only implicitly, for example by describing a manual setup step without calling it a burden, the limitation was recorded as an inferred one and is described in neutral terms in Section II-C.

4) Synthesis Approach

Extracted fields were grouped into four themes: predictive freshness and decay modeling; dashboarding, pricing, and visualization; platform scalability and spatial mapping; and field-level agricultural monitoring. Within each theme the sources were compared qualitatively, and across themes they were compared along four axes: predictive depth, physical actionability, accessibility to non-technical operators, and deployment cost. No quantitative meta-analysis was attempted, since the reviewed works report heterogeneous outcomes on different datasets.

5) Limitations of the Review

The corpus is small and was selected for architectural relevance rather than exhaustiveness. Several sources are cited as unpublished surveys, so their claims could be assessed only as described and not independently reproduced. The conclusions drawn in later sections should therefore be read as design motivation rather than as a statistical finding about the field as a whole.

C. Thematic Synthesis

The eight sources fall into four thematic groups: predictive freshness and decay modeling; dashboarding, pricing, and visualization; platform scalability and spatial mapping; and field-level agricultural monitoring. Reading across these groups makes it easier to see where capabilities overlap and where a real deployment would need to combine two or more of them to be genuinely useful to a store operator.

1) Predictive Freshness and Decay-Modeling Approaches

An automated freshness-tracking framework described in [1] replaces manual shelf inspection with a purely software pipeline: electronic receiving records and point-of-sale history are fed into time-series algorithms that estimate shelf life and project deterioration dates for each batch. Its principal strength is consistency—standardizing the evaluation step

removes the subjective bias inherent in visual audits and considerably increases inventory-logging throughput. Its central weakness is a strict dependency on clean data at intake: an incorrect timestamp or a mismatched crop-category code entered during receiving propagates directly into a corrupted downstream prediction. Because the framework describes no validation step at the point of entry, an error introduced in seconds during a busy receiving shift can quietly persist through every later prediction.

A related, procurement-oriented approach in [2] centers the same problem on a relational database that correlates incoming supply manifests against historical sales curves to recommend optimal stock volumes, reducing both overpurchasing and shrink. It aligns ordering with real consumption patterns and gives managers a centralized tabular view across departments, but its static trend calculations struggle when seasonal demand spikes or shifting consumer behavior break the assumed pattern, and it offers no visual layout of the store. The work is also acknowledged by its authors to be largely theoretical, without a deployed codebase.

A broader survey in [3] catalogues the mathematical degradation formulas and classification algorithms available for scoring produce freshness from variables such as crop type, harvest date, and storage duration. Its contribution is foundational rather than applied: it maps out which model families suit low-latency, on-device computation, but does not show how these formulas would be wired into a working multi-tier system, nor does it address the null values and inconsistent spreadsheet formats that any real ingestion pipeline must handle.

Taken together, these three works establish that shelf life can be estimated from records rather than from inspection, that estimates can be linked to purchasing, and that a toolbox of formulas exists for scoring freshness. What they leave open is the connective tissue: how raw, imperfect store data becomes a trustworthy input, and how the output reaches a person standing on the shop floor.

2) Dashboarding, Pricing, and Visualization Systems

A cloud-hosted dashboard platform in [4] compiles warehouse logs, shipping manifests, and retail stock metrics into a single browser-accessible interface, replacing dense spreadsheets with charts and status indicators readable from a phone, tablet, or terminal without local installation. This lowers the barrier to cross-department coordination, but the design is purely descriptive—it visualizes what has already happened and offers no mechanism for forecasting when a batch will spoil—and its usefulness collapses without a stable, high-bandwidth connection.

A complementary pricing-side approach in [5] continuously compares a batch's age against a predefined shelf-life threshold and, once crossed, automatically pushes a calculated markdown to point-of-sale terminals, converting stock that would otherwise be discarded into recovered revenue while removing manual retagging. The tradeoff is that

decay-rate and margin parameters must be tuned carefully in advance; a miscalibrated threshold either erodes margin through premature discounts or confuses shoppers with prices that shift visibly through the day.

These two works illustrate a useful contrast. The dashboard in [4] is broad but passive: it informs. The pricing engine in [5] is narrow but active: it acts. A store-level platform needs both qualities, informing the owner broadly while offering narrow, well-bounded actions such as a suggested markdown that the owner can accept or decline.

3) Platform Scalability and Spatial Mapping

AgroTrack, described in [6], addresses how a single platform can serve many stores at once. It isolates data by store through separate schemas on shared infrastructure, giving corporate staff, branch managers, and warehouse operators role-appropriate dashboards without cross-contamination, and it centralizes compliance history for fast auditing across a chain. Its cost grows with scale, however, and because branches share the same platform, a bug or a corrupted bulk import in one location can in principle disrupt others.

A floor-mapping system in [7] binds inventory database rows directly to coordinates on a digital storefront layout, so a status change (say, a batch moving from stable to near-expiry) triggers an instant color change on the map without a page refresh, letting staff be dispatched straight to the aisle that needs attention. The cost is a substantial one-time setup: every shelf, cooler, and display bin must be manually mapped to a coordinate, and any physical layout change forces the mapping to be redone. Simultaneous updates from multiple staff can also introduce brief inconsistencies between terminals.

The two works pull in different directions. [6] optimizes for breadth across many stores and pays for it in infrastructure and shared-failure risk; [7] optimizes for depth within one store and pays for it in setup effort. A design aimed at small and mid-sized retailers should borrow the role-based views of [6] in lightweight form and the data-to-map binding of [7], while reducing the manual mapping burden through a simple, editable layout definition.

4) Field-Level Agricultural Monitoring

Outside the retail floor, [8] describes a cloud framework for crop cultivation in which soil-moisture probes, NPK sensors, weather stations, and drone cameras feed continuous environmental data into a central database, where deep-learning models such as CNN-LSTM networks and vision-based detectors (e.g., YOLO-family models) guide irrigation and flag early signs of disease from leaf-pattern changes. This precision-agriculture approach reduces water and fertilizer waste and can catch disease before it spreads, but it depends on ongoing investment in physical sensor hardware, and its architecture does not transfer directly to an indoor retail setting where the object of interest is a shelf rather than a field. Even so, the underlying pattern—continuous sensing feeding a predictive model that

triggers an early, targeted intervention—is directly analogous to what a retail-side freshness system needs to do.

5) Supporting Work on Perishable Storage and LLM-Based Analytics

Beyond the eight core sources, two supporting strands inform the design. IntelliStore [16] combines IoT sensing and artificial intelligence to monitor storage conditions for perishable food, reinforcing the point that storage conditions are a primary driver of shelf life. On the language-model side, [12] and [15] explore agentic and natural-language approaches to inventory and data querying, and [14] surveys text-to-SQL workflows, which highlights both the convenience and the risk of letting a model generate database queries freely. The proposed system adopts the convenience while limiting the risk by routing requests through a fixed set of domain-specific functions.

D. Comparative Analysis

Table I summarizes the primary approach and the key reported outcome of each reviewed study. Table II then compares the studies along four qualitative axes introduced in Section II-B.

Table 1 Summary of Reviewed Studies

Ref	Primary Approach	Key Reported Outcome
[1]	Time-series freshness prediction from receiving + POS data	Eliminates manual audit bias; brittle to bad intake data
[2]	DB-driven decay modeling & stock-volume recommendation	Reduces overpurchasing; no visual layout, weak to demand shifts
[3]	Survey of spoilage-classification algorithms	Strong theoretical grounding; no working implementation
[4]	Cloud dashboard for supply-chain metrics	High accessibility; descriptive only, connectivity-dependent
[5]	Automated markdown pricing engine	Recovers revenue from aging stock; needs careful calibration
[6]	Multi-tenant chain-scale retail platform	Scales across branches; shared-infrastructure risk
[7]	Coordinate-bound retail floor mapping	Instant visual status; high manual setup overhead
[8]	IoT + deep learning crop cultivation platform	Precision irrigation/disease detection; hardware-heavy, field-only

Two clear axes separate the reviewed works. The first is predictive depth: [1] and [3] attempt genuine forecasting of decay, while [2] and [4] largely describe or visualize inventory state after the fact. The second is physical actionability: only [7] connects a database state directly to a spatial, walkable representation of the store, and only [5] closes the loop into an automatic operational action (a price change) rather than stopping at an alert. No single reviewed system combines predictive decay modeling, dynamic pricing, and spatial floor visualization within one architecture.

A third axis is accessibility: most reviewed systems assume an operator comfortable navigating a conventional web dashboard, reading a data table, or configuring a rules-based pricing threshold. None treats conversational or voice access as a first-class requirement. This suggests that accessibility, rather than raw predictive accuracy, may be the more immediate barrier separating current agro-retail research from adoption by small and mid-sized store owners. A fourth pattern is that cost and deployment scope move in opposite directions from predictive sophistication: [1] and [3] are described purely in software terms with little physical infrastructure, whereas [6] and [7] reach furthest into store operations at the price of rising cloud costs or substantial manual setup.

Table 2 Qualitative Comparison Along Four Axes

Ref	Predictive depth	Physical actionability	Accessibility	Deployment cost
[1]	High	Low	Moderate	Low (software)
[2]	Moderate	Low	Moderate	Low
[3]	High (theory)	None	Low	Low
[4]	Low	Low	High	Moderate (cloud)
[5]	Moderate	High (price)	Moderate	Moderate
[6]	Low	Moderate	Moderate	High (scale)
[7]	Low	High (spatial)	Moderate	High (setup)
[8]	High (field)	Moderate	Low	High (hardware)

Ratings in Table II are the authors' qualitative reading of how each work is described, not measurements reported by the original studies. They are intended to make the pattern visible: no row is strong on all four axes at once, and the rows that score highest on actionability tend to score highest on cost.

Taken as a whole, the field has developed strong, individually validated components—decay forecasting, dashboarding, dynamic pricing, multi-branch scaling, and spatial mapping—without yet producing an architecture that assembles them behind a single, low-overhead interface. This is less a criticism of any individual work than an observation about the corpus: the next meaningful advance is more likely to come from integration than from further improving any one component in isolation.

RESEARCH GAPS

Five gaps recur across the reviewed literature, in rough order of how foundational they are.

A. Data-Quality Fragility

Nearly every predictive approach—most explicitly [1]—assumes clean, correctly coded data at receiving, with no mechanism to catch or correct anomalies before they reach the prediction model. Receiving-stage entry is done under time pressure by staff rarely

trained in data hygiene, so mistyped timestamps, duplicated batch codes, and inconsistent unit labels are common; a single bad entry can silently corrupt every downstream forecast.

B. Absence of Combined Predictive-and-Spatial Visualization

Decay prediction ([1], [2]) and spatial floor mapping ([7]) are demonstrated separately; none render predicted—as opposed to already-observed—freshness onto a physical store layout. A manager must still mentally translate a table of expiry dates into a walking route, reintroducing the manual, error-prone step that predictive modeling was meant to remove.

C. Limited Adaptability to Demand Volatility

The trend-based models in [2] struggle with seasonal or sudden shifts in demand. A model trained on a typical week has no built-in way to anticipate a festival-driven spike or a local shortage, and none of the reviewed systems describe a retraining or online-adaptation strategy.

D. High Manual Configuration Overhead for Spatial Systems

The approach in [7] requires every shelf and display bin to be manually coordinate-mapped and re-mapped on any layout change—tolerable for one flagship store, but a poor fit for multi-branch retailers that periodically reset their floor plan.

E. No Natural-Language or Voice Access Layer

None of the retail-side systems ([1], [2], [4]–[7]) exposes functionality through conversational or voice interaction, quietly excluding small-shop owners with limited comfort around software, even though the underlying queries—stock levels, expiring batches, today’s revenue—are narrow and repetitive.

Table 3 Mapping of Research Gaps to Design Responses

Gap	Design response in the proposed system
Data-quality fragility	Automated validation and cleaning stage in the Python analytics engine before any prediction is made
No predictive + spatial view	Predicted freshness state per batch rendered on a colour-coded React-Leaflet storefront map
Demand volatility	Prediction parameters stored as editable data; periodic re-fitting against recent sales history
Spatial setup overhead	Section-level layout defined in a single editable configuration rather than per-item coordinates
No natural-language access	LangChain agent over Gemini 1.5 Flash with typed and voice input and a fixed set of domain tools

PROBLEM STATEMENT

Small and mid-sized agro-retail stores typically operate with three weaknesses at once. Their receiving data is entered by hand under time pressure; their freshness

judgments are made by whichever staff member happens to be on the floor; and their reporting tools, where they exist, present rows and columns rather than a picture of the shop. The consequence is a chain of delays: a batch begins to degrade, nobody notices until it is visibly poor, the owner learns of it from a write-off report days later, and the lesson is never fed back into purchasing or pricing. The problem addressed in this paper is how to shorten that chain using only software that a small retailer can realistically deploy and operate.

PROPOSED SYSTEM

A. Design Rationale

The identified gaps motivate a platform built around three coupled capabilities that no single reviewed system combines: automated, self-cleaning data ingestion; predictive shelf-life modeling; and a store layout that visualizes that prediction directly, all accessible through natural-language or voice interaction rather than a conventional dashboard alone. Four principles guide the design. The platform must be software-only, so a small retailer does not need to buy sensors. It must be tolerant of imperfect data, since receiving records are entered under pressure. It must be explainable, so every prediction can be traced to the records behind it. And it must be conversational, so the owner can reach any function by asking a question.

B. Expected Impact

The automated ingestion pipeline, predictive analytics core, and conversational floor-map interface are intended to compress the time between a batch beginning to degrade and the owner acting on that information. Where manual inspection might catch early spoilage only once a day or once per shift, a continuously updated prediction on a live map can flag a batch the moment it crosses into an aging or critical state, narrowing the window for intervention through restocking, repositioning, or a markdown. Over time, this shift is expected to reduce both the produce discarded after visible spoilage and the labor hours spent on aisle-by-aisle inspection, while giving owners a single conversational entry point for ad hoc questions—what is expiring today, which section needs attention, how revenue compares to last week. These benefits are design expectations; their magnitude has not yet been measured.

SYSTEM ARCHITECTURE

The platform follows a three-tier architecture. A React.js frontend provides a chat-style interface accepting typed text and voice input, transcribed client-side through the browser's Web Speech API, alongside animated charts (Recharts) and process flow diagrams (Mermaid.js). A FastAPI backend hosts an agent-orchestration layer, implemented with LangChain, that maintains conversational memory and delegates intent interpretation to a large language model (Google Gemini 1.5 Flash) accessed through its API. Rather than routing requests through hardcoded command paths, the LLM determines which of a small

set of domain-specific Python functions—reading the database schema, executing a query, retrieving expiring stock, generating a chart, or producing an explanatory summary—best matches the owner’s spoken or typed intent. Those functions operate against a local SQLite database (`agro_shop.db`) organized around four domains: product catalog, inventory and batch expiry, sales and revenue, and cost tracking.

TIER 1 — PRESENTATION (React.js)			
Chat UI (text + voice)	Recharts animated charts	Mermaid.js flow diagrams	React-Leaflet colour-coded map
TIER 2 — ORCHESTRATION (FastAPI + LangChain)			
Agent executor + memory	Gemini 1.5 Flash (intent routing)	Domain tools (query, expiry, chart)	Python analytics (clean + predict)
TIER 3 — DATA (SQLite: <code>agro_shop.db</code>)			
Product catalog	Inventory & batch expiry	Sales & revenue	Cost tracking

Figure 1 Three-Tier Architecture of the Proposed System

METHODOLOGY AND WORKFLOW

A. Automated Data Ingestion and Cleaning

The Python analytics engine is the first line of defense against the data-quality fragility identified in Section III. Inbound records, whether uploaded as spreadsheets or entered through a form, pass through a fixed sequence of checks before they are written to the database:

- **Schema normalization.** Column names, date formats, and unit labels are mapped to a canonical form, so that “kg”, “Kgs”, and “kilograms” are treated as one unit.
- **Null handling.** Missing quantities or dates are flagged for correction rather than silently defaulted; missing optional fields receive documented defaults.
- **Duplicate detection.** Repeated batch codes, or rows that match an existing batch on product, supplier, and arrival date, are held for confirmation.
- **Plausibility checks.** Arrival dates in the future, expiry dates earlier than arrival, and quantities far outside a product’s normal range are rejected or queried.
- **Category reconciliation.** Product names are matched against the catalog so that a crop-category code cannot drift from the product it describes.

Rejected or ambiguous rows are returned to the owner in plain language—for example, “this batch has an expiry date earlier than its arrival date; did you mean the following year?”—so that correction happens at intake rather than being discovered later as an unexplained forecast error.

B. Shelf-Life Prediction

For each batch, the engine estimates remaining shelf life from the product’s baseline shelf life, the time since arrival, and the storage condition recorded for its location (for example, chilled cooler versus ambient display). As an illustrative formulation, freshness can be expressed as a score that decays with elapsed time, $F(t) = F_0 \cdot e^{(-k \cdot t)}$, in which the decay

constant k depends on the product and on its storage class. The score is then compared against two configurable thresholds to assign one of three states: stable, aging, or critical. Baseline shelf lives and decay constants are stored as editable data rather than hard-coded, so that they can be adjusted per product and re-fitted against recent sales and write-off history.

Two properties of this design are deliberate. First, it is transparent: the owner can ask why a batch is marked aging and receive the arrival date, storage class, and threshold that produced the state. Second, it is adaptable: because parameters are data, a seasonal change in demand or a shift in supplier quality can be reflected by updating them, addressing in part the demand-volatility gap in Section III. The exact functional form and parameter values are to be calibrated against real store data in future work and are not claimed here to be optimal.

C. Conversational Orchestration Layer

The conversational layer converts a typed or spoken question into a verified answer. The LangChain agent executor receives the transcribed query along with recent conversation history, and Gemini 1.5 Flash selects among the available tools. Because the tools are deterministic Python functions over the database, the numbers that appear in an answer come from queries rather than from the model's own generation; the model's role is to choose the tool, supply its arguments, and phrase the result.

Table 4 Domain Tools Exposed to the Language Model

Tool	Purpose	Typical owner question
read_schema	Return table and column descriptions	"What data do you have?"
run_query	Execute a validated, read-only query	"How much tomato did I sell this week?"
get_expiring_stock	List batches by predicted state or date	"What is expiring today?"
make_chart	Return a chart specification for Recharts	"Show revenue by category."
explain_summary	Produce a plain-language summary or flow diagram	"Explain how receiving works."

Several safeguards follow from this design. Queries are executed read-only by default; any write operation, such as updating a batch or accepting a markdown, requires explicit confirmation. Tool arguments are validated before execution. Conversational memory allows follow-up questions such as "and last week?" to inherit context without restating it. Voice input is transcribed in the browser, so spoken queries are handled by the same path as typed ones.

D. Colour-Coded Storefront Map

The predicted state of each batch is rolled up to its store section and rendered through an interactive React-Leaflet floor-plan map, with sections coloured by condition:

bright green for stable stock, warm amber for aging stock approaching its threshold, and vibrant red for stock in critical condition. Rather than binding every item to a coordinate, as in [7], the layout is defined once at section level—for example, leafy greens, root vegetables, fruit, and dairy—in an editable configuration. A layout change therefore requires editing a section definition rather than re-mapping every shelf, which reduces the setup overhead identified in Section III.

Selecting a section opens the batches behind its colour, ordered by urgency, so that a floor worker can walk directly to the shelf that needs attention. This closes the gap between predictive modeling and spatial visualization: rather than reading a table of expiry dates, a floor worker acts on a walkable map immediately.

E. Shelf-Imagery and Visual Freshness Modules

On top of this conversational core, the platform scans shelf imagery to track stock levels and trigger restocking alerts when items fall below a defined threshold, and applies visual freshness-assessment models to produce imagery that flags early degradation. These modules are positioned as complements to the record-based predictor rather than replacements for it: record-based prediction provides an estimate for every batch, while imagery provides a check on those batches that can be photographed. Disagreement between the two—a batch the record-based model calls stable but the image model flags as degraded—is itself a useful signal that the baseline parameters need review.

F. Operational Flow

A typical interaction begins when the shop owner submits a natural-language or spoken query. The React application forwards a structured request to the FastAPI backend, which invokes the LangChain agent executor. The agent, in consultation with Gemini 1.5 Flash, selects and executes the appropriate backend tool against the SQLite database, and the resulting rows are translated into a natural-language explanation, a chart payload, or both. The frontend renders this as text, an animated chart, or an update to the color-coded floor map, closing the loop from a single conversational query to a concrete, visual operational insight.

Voice / text query	React frontend (structured request)	FastAPI + LangChain agent	Gemini selects tool	SQLite query executed	Text / chart / map update
→	→	→	→	→	✓

Figure 2 Operational Flow of a Conversational Query

G. Database Design

The SQLite database, `agro_shop.db`, is intentionally small and organized around four domains that mirror the owner’s everyday questions. Table V lists the principal entities. Keeping the schema compact keeps the schema-reading tool simple and lets the language model reason reliably about which table answers which question.

Table 5 Principal Database Entities

Domain	Typical contents
Product catalog	Product identifier, name, category, unit, baseline shelf life, storage class
Inventory and batch expiry	Batch code, product, supplier, arrival date, expiry date, quantity, store section, predicted state
Sales and revenue	Transaction date, product, quantity sold, selling price, discounts applied
Cost tracking	Purchase cost per batch, overhead categories, write-off quantity and value

IMPLEMENTATION AND EVALUATION / RESULTS

A. System Requirements

1) Software Requirements

The frontend is built with HTML5, CSS3, and client-side JavaScript, accessed through standard browsers such as Chrome, Firefox, and Edge: HTML5 provides the structural skeleton, CSS3 governs styling and responsiveness across screen resolutions, and JavaScript injects the dynamic behavior needed for real-time interaction, voice capture, and live chart updates. The backend runs on Node.js and FastAPI (Python); Node.js manages API integration and session handling, while the Python-based FastAPI layer hosts the LangChain agent executor and coordinates calls to Gemini 1.5 Flash. Development takes place in Visual Studio Code, while jsPDF supports downloadable, formatted report generation directly from the dashboard. The platform targets Windows 10 or later, Windows 11, Ubuntu 20.04 or later, and macOS, and requires an active, continuous internet connection for LLM API access, database synchronization, and voice-transcription services.

2) Hardware Requirements

A minimum configuration of an Intel Core i3 or AMD Ryzen 3 dual-core processor at 2.0 GHz, 4 GB of RAM, and 20 GB of free HDD storage is sufficient to run the platform on a small scale, though performance may be slower during heavy chart rendering or concurrent database queries. A recommended configuration—an Intel Core i5/i7 or AMD Ryzen 5/7 quad-core processor, 8 to 16 GB of RAM, and 50 GB or more of free SSD storage—is advised to run the local server, database engine, development environment, and browser simultaneously without degradation. Solid-state storage is preferred because it accelerates database reads and writes and local code execution, directly affecting how responsive the conversational interface feels.

Table 6 Summary of Hardware Configurations

Component	Minimum	Recommended
Processor	Intel Core i3 / AMD Ryzen 3, dual-core, 2.0 GHz	Intel Core i5/i7 / AMD Ryzen 5/7, quad-core
Memory	4 GB RAM	8–16 GB RAM
Storage	20 GB free (HDD)	50 GB or more free (SSD)
Network	Continuous internet connection	Continuous, stable internet connection

B. Security, Privacy, and Reliability Considerations

Because the platform sends the owner's queries to an external language-model API, the design limits what leaves the store: the model receives the question, the tool descriptions, and the results of tool calls needed to phrase an answer, and it does not receive the full database. API credentials are held on the backend and never exposed to the browser. Operations that modify data are confirmed by the user. Because the system depends on internet access for the language model and voice transcription, core read operations on the local database and the colour-coded map should degrade gracefully, remaining available from cached data when the connection is interrupted, even though conversational features would be unavailable.

C. Proposed Evaluation Plan

Validation of the design is planned in stages. In the first stage, the ingestion engine is tested on deliberately corrupted sample data to measure the share of injected errors it catches. In the second, the shelf-life predictor is compared against dated manual inspections over a trial period, with agreement measured by how many hours earlier or later the predicted state change occurs than the inspector's judgment. In the third, the conversational layer is assessed by whether a fixed set of owner questions returns correct answers, and by how often the agent selects the correct tool. In the fourth, a pilot in a working store compares write-off volume and inspection hours before and after adoption across a full seasonal cycle. Only the last stage can support claims about waste reduction, and no such claims are made in this paper.

D. Discussion and Limitations

The central argument of this paper is that integration, not any single component, is the open problem in agro-retail monitoring. The proposed design is an attempt to demonstrate that integration is feasible with modest, software-only means. Several limitations should nonetheless be stated plainly.

- **No empirical results.** The system is presented as a design. Claims about reduced waste, reduced labor, or prediction accuracy are expectations, to be tested in the staged evaluation of Section VIII-C.

- **Dependence on connectivity.** Conversational features rely on an external language-model API and on browser voice transcription, so they require a continuous internet connection and incur per-request cost.
- **Language-model reliability.** Although tool-based orchestration limits hallucination, the model can still select the wrong tool or misread an ambiguous question. Confirmation prompts for writes and display of the underlying query mitigate, but do not eliminate, this risk.
- **Prediction fidelity.** A record-based decay model cannot see conditions the records do not capture, such as a refrigeration fault overnight. The imagery module and, in future, optional temperature inputs are intended to narrow this gap.
- **Scope of validation.** The corpus reviewed is small and partly composed of unpublished surveys, so the gaps identified are design motivation rather than a statistically established finding.
- **Language coverage.** The initial design targets a single interface language; regional-language support is deferred.

A further open question is how far the platform should go in acting on its own predictions. The review in Section II-C shows that automated markdown pricing [5] can recover revenue but is sensitive to miscalibration. The proposed system therefore stops at recommending actions and asks the owner to confirm them, trading some automation for trust during early adoption.

CONCLUSION

The literature reviewed here shows steady, independent progress on the individual pieces of agro-retail monitoring—predictive decay modeling, dashboarding, dynamic pricing, chain-scale platforms, and spatial floor mapping—but no single existing system combines automated data cleaning, predictive freshness modeling, natural-language access, and spatial visualization into one coherent platform. Each capability has been demonstrated in isolation, by a different team, against a different dataset, which makes it difficult for a retailer to adopt more than one at a time without stitching together unrelated tools.

The proposed Intelligent Agro-Based Retail Monitoring System is designed to close that combination gap. A Python analytics core handles ingestion and shelf-life prediction, automatically cleaning and standardizing inbound records so that forecasts are not derailed by data-entry errors. A LangChain-orchestrated conversational layer built on Gemini 1.5 Flash makes that intelligence accessible to a non-technical shop owner through natural language or voice. A React-Leaflet floor map turns the resulting predictions into an immediately actionable, color-coded view of the store. In doing so, the platform aims to shift agro-retail operations away from reactive, labor-bound inspection and toward proactive, evidence-based management—reducing food waste, lowering operational overhead, and giving retailers of any size a scalable route to data-driven decision-making.

A. Future Work

Future extensions could incorporate computer-vision-based freshness scoring directly from shelf imagery, broaden the conversational layer to additional regional languages, and validate predictive accuracy against real store data collected over a full seasonal cycle, further narrowing the gap between the proposed architecture and a fully field-tested deployment. Further directions include optional low-cost temperature and humidity inputs to sharpen the decay model, an offline mode for core read operations, multi-branch support using lightweight role-based views in the spirit of [6], and recommendation of markdown actions informed by the pricing logic of [5].

REFERENCES

- [1] Anonymous, "Automated Agro-Retail Freshness Tracking System using Predictive Analytics," unpublished survey.
- [2] Anonymous, "Data-Driven Inventory Management and Decay Modeling for Fresh Commodities," unpublished survey.
- [3] Anonymous, "A Survey Paper on Automated Food Spoilage Classification and Machine Learning Methods," unpublished survey.
- [4] Anonymous, "Web-Based Interactive Dashboards for Fresh Produce Supply Coordination," unpublished survey.
- [5] Anonymous, "An Intelligent Software Framework for Dynamic Markdown Pricing of Perishable Goods," unpublished survey.
- [6] Anonymous, "AgroTrack: A Multi-Tenant Software Platform for Scale-Out Grocery Chains," unpublished survey.
- [7] Anonymous, "Design and Development of an Intelligent Software System for Retail Floor Mapping," unpublished survey.
- [8] Anonymous, "Intelligent Agro-Cloud Framework for Efficient Crop Cultivation," unpublished survey.
- [9] L. Ouyang et al., "Training language models to follow instructions with human feedback," 2022.
- [10] E. D. Zamani et al., "Artificial intelligence and big data analytics for supply chain resilience: A systematic literature review," 2022.
- [11] A. Vaswani et al., "Attention is all you need," 2017.
- [12] Y. Quan and Z. Liu, "InvAgent: A large language model based multi-agent system for inventory management in supply chains," 2025.
- [13] A. Knott et al., "Unstructured data extraction in agricultural logistics using frontier large language models," 2023.

- [14] T. Unal and A. Bashar, "Next-generation database interfaces: A survey of LLM-based text-to-SQL workflows," 2025.
- [15] X. Wu et al., "AgentNLQ," 2026.
- [16] S. Singha, D. Saha, M. Brahma, and P. K. Singh, "IntelliStore: IoT and AI-Based Intelligent Storage Monitoring for Perishable Food," *Internet Technology Letters*, vol. 6, no. 3, 2023, doi: 10.1002/itl2.402.