Technical Evaluation of an Autonomous Business Operations Engine

Haardik Garg
Sapien Institute
University of Waterloo
h9garg@uwaterloo.ca

Jeevan Sandhu
Sapien Institute
University of Toronto
js.sandhu@mail.utoronto.ca

Abstract

Tally is an autonomous operations engine for retail businesses. We evaluate it in a 365-day closed-loop simulation of an ice cream shop with three suppliers, twelve inventory items, four employees, and 26,369 synthetic sales transactions driven by real historical weather. The engine independently forecasts demand, maintains inventory state, generates and fulfills purchase orders, produces weekly staff schedules, detects operational anomalies, adapts its parameters through bounded correction factors, and qualifies for per-supplier ordering autonomy through a deterministic promotion protocol. Over the full year, Tally placed 124 autonomous purchase orders totaling $17,856, maintained 96.3–116.5% restock coverage across all items, published 52 weekly schedules with zero coverage gaps, and detected 409 anomalies, all of which received operator feedback. All three suppliers were granted autonomous ordering authority within the first five months. The forecasting engine received only observable sales and weather inputs without access to the demand-generation process. This evaluation demonstrates that Tally maintains continuous closed-loop operation across the configured operational domains, including bounded autonomous ordering after supplier-specific authority was earned. Performance against live point-of-sale data and real supplier behavior remains a deployment validation question.

1 Introduction

A small retail business produces a stream of daily decisions: what to order, how much, from which supplier, when to schedule staff, whether today's sales are normal, and whether the system making those decisions can be trusted. These decisions interact. Ordering depends on forecasts; forecasts depend on sales data; staffing depends on demand profiles; anomaly detection depends on forecast residuals; and autonomous action depends on accumulated trust. A system that handles only one domain in isolation misses the operational reality that these loops run simultaneously and continuously.

Tally is an integrated autonomous operations engine that closes all of these loops. It is not a forecasting model, a scheduling optimizer, or an anomaly detector used independently. It is a single system where each component produces outputs consumed by others, running against a shared clock, shared state, and shared learning infrastructure.

This document evaluates Tally through a 365-day closed-loop simulation. The simulation is not a demonstration of selected capabilities. It is a continuous run in which the engine processes every sale, updates every inventory position, evaluates every ordering decision, proposes every schedule, flags every anomaly, and adapts every correction factor for an entire simulated year. A virtual operator interacts with the system daily, providing imperfect and stochastic feedback that the engine must absorb.

The evaluation is structured to be transparent about what was tested and what was not. Section 2 describes the simulated business, demand generation, virtual operator, and supplier behavior in full. Section 3 defines what the engine was and was not allowed to do. Section 4 reports evaluation criteria and measured results across each operational domain. Section 5 describes engineering failures discovered and corrected during iterative simulation. Section 6 summarizes headline metrics. Section 7 addresses threats to validity, including the demand-generator alignment problem. Section 8 discusses reproducibility constraints.

The simulation is designed to answer five questions: Can Tally maintain consistent operational state over a full simulated year? Can specialized decision systems operate as a coupled closed loop? Can the system safely transition from recommendation to bounded autonomy? Can feedback and adaptation improve behavior without unbounded drift? Can the system detect and recover from implementation failures exposed through long-running simulation? The simulation does not attempt to establish real-world business performance, generalization across business types, or superiority of any particular forecasting model.

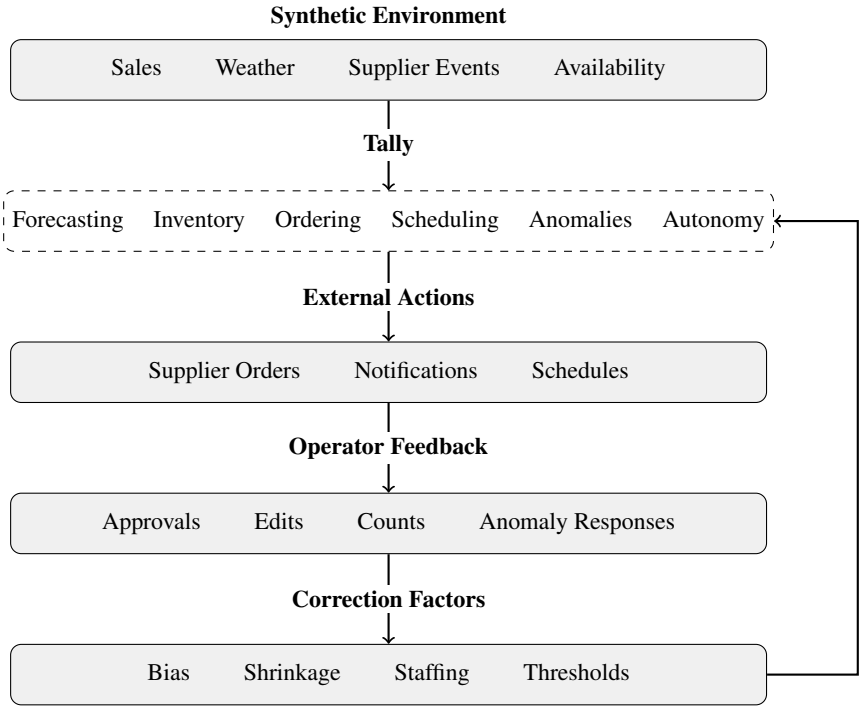

Figure 1: System boundary and feedback structure. Correction factors feed back into Tally on the next daily cycle.

2 Experimental environment

2.1 Business configuration

The simulated business is a single-location ice cream shop in Vancouver, BC, operating every day for one calendar year (January 1–December 31, 2019). Table 1 summarizes the configuration.

The three suppliers carry four items each, with minimum order values between $40 and $75, and review periods of 3–7 days. Every inventory item has at least one supplier. The four employees have heterogeneous constraints: weekly hour caps of 20–40 hours, shift lengths of 4–8 hours, and availability Monday through Saturday. Sundays are owner-operated: the owner staffs the shop directly while performing weekly inventory counts and reviewing the system's operational output. This is the assumed management day for businesses at this staffing scale. The scheduling engine covers the six employee-staffed days.

Table 1: Simulation configuration.

Parameter	Value
Business type	Ice cream shop
Location	Vancouver, BC (49.28° N, 123.12° W)
Simulation period	365 days (Jan 1–Dec 31, 2019)
Catalog items	3 (Single Scoop, Double Scoop, Sundae)
Modifiers	10 (3 flavors, 2 cones, 5 toppings)
Inventory items tracked	12
Bill-of-materials lines	32
Suppliers	3 (lead times: 2, 3, 5 days)
Employees	4 (2 keyholders)
Weather source	Open-Meteo historical archive
Time resolution	Hourly sales, daily decisions

2.2 Synthetic demand generation

Sales are generated from a parameterized stochastic process incorporating calendar effects, temperature sensitivity, and random variation. The simulation does not use a fixed random seed, so different runs produce quantitatively different outputs with the same structural properties.

Daily volume is the product of a base rate, a day-of-week multiplier, a temperature factor, a noise term, and a mild upward trend. Day-of-week multipliers range from 0.7 (Tuesday) to 1.4 (Sunday), reflecting the weekend demand pattern typical of retail food service. The temperature factor is a linear function of the day's maximum temperature relative to 20° C, clamped to the interval $[0.5, 1.8]$. The noise term is drawn uniformly from $[0.85, 1.15]$. The trend compounds at 0.15% per week, yielding approximately 8% annual growth. These parameters produce daily volumes ranging from 16 to 242 transactions, with mean 72.2 and standard deviation 24.8.

Six demand shocks are injected at fixed dates to test the engine's response to atypical demand. Four are positive shocks on public holidays (1.5–2.5× the baseline multiplier) and two are negative shocks representing a mid-January deep freeze (0.4×) and Remembrance Day (0.3×). The engine has no prior knowledge of these dates.

Product mix shifts with temperature. Warmer days produce proportionally more sundaes and more strawberry relative to chocolate. Colder days produce more single scoops and more chocolate. Toppings are applied to all sundaes and 30% of scoops. Each transaction depletes inventory through a bill-of-materials explosion: a double scoop with two flavors and a cone produces five inventory transactions (two flavor depletions, one cone, one spoon, and one topping if selected).

Hourly distribution follows a fixed intraday profile peaking at 2–3pm (15% of daily volume per hour) and tapering to 3% at opening and closing.

Table 2: Realized demand by month.

Month	Sales	Revenue	Days	Avg daily
January	1,748	$9,436.50	31	56.4
February	1,536	$8,188.50	28	54.9
March	2,039	$11,062.50	31	65.8
April	2,014	$11,081.00	30	67.1
May	2,565	$14,251.50	31	82.7
June	2,393	$13,312.50	30	79.8
July	2,667	$15,030.50	31	86.0
August	2,803	$15,914.50	31	90.4
September	2,494	$14,006.50	30	83.1
October	2,134	$11,688.50	31	68.8
November	2,003	$10,913.00	30	66.8
December	1,973	$10,451.50	31	63.6
Total	**26,369**	**$145,337**	**365**	**72.2**

2.3 Weather integration

Both the demand generator and the forecasting engine use real weather data from the Open-Meteo historical (Open-Meteo, 2024) archive for Vancouver, 2019. The actual daily maximum temperature ranged from $-1.1°\mathrm{C}$ to $27.6°\mathrm{C}$ over the year. To simulate forecast error, a separate forecast weather series was generated by adding uniform noise from $-3°\mathrm{C}$ to $+3°\mathrm{C}$ to each day's actual temperature. Historical weather archives record observed conditions, not the forecasts that were issued at the time. No public source preserves the 14-day forecast that was available on a given date in 2019, so the forecast series could not be reconstructed from real forecast snapshots. The uniform perturbation is a simplification; real forecast error is correlated across days, larger at longer horizons, and varies by weather regime. The consequence is that the prediction intervals in this simulation absorb synthetic forecast error that may understate or mischaracterize the structure of real forecast uncertainty.

The engine trains its models using historical weather observations available through each forecast origin and generates predictions using the noisy forecast series, so the empirical calibration incorporates the resulting synthetic weather-forecast error into the prediction intervals.

2.4 Virtual operator

A virtual operator provides imperfect, stochastic feedback to the engine, simulating a small-business owner who engages daily but does not follow instructions perfectly.

Purchase order review. Of proposals presented for approval, 75% are approved without modification, 20% are approved with edits to 1–2 randomly selected line items ($\pm$10–25% quantity adjustment, rounded to pack size), and 5% are cancelled outright. Cancellations and edits are random and content-independent: the operator does not evaluate whether a proposal is reasonable. This means the engine faces noise in operator feedback rather than learnable signal, which tests whether the adaptation mechanisms remain stable under uninformative feedback.

Delivery handling. Confirmed orders due on the current day are received with 75% probability; the remainder slip to the next day. Orders due tomorrow arrive early with 5% probability. Overdue orders are always received. Of received deliveries, 10% arrive short (85–95% of ordered quantity). Supplier behavior is otherwise deterministic: prices do not change, suppliers do not stock out, and communication does not fail.

Anomaly feedback. Inventory underflow anomalies are always acknowledged and trigger a correction count (resetting the item's position). All other anomaly types are acknowledged with 75% probability and dismissed with 25% probability. Feedback drives the threshold calibration loop.

Autonomy grants. When the engine proposes a supplier promotion (having met the evidence threshold), the operator waits 3 days then grants with 90% probability.

Inventory counts. Physical cycle counts occur on any given day with 15% probability (approximately every 6–7 days). Counts introduce measurement noise: $\pm$0.5 kg for weight-measured items and $\pm$5 units for count-measured items. Discrepancies feed the shrinkage learning loop.

Schedule review. Each Monday, the operator selects one random shift and adjusts its end time by $\pm$60 minutes with 50% probability, then publishes the schedule. Edits feed the staffing-ratio correction factor.

2.5 Simulation loop

Each simulated day proceeds in fixed order: the virtual clock advances to opening; all sales for the day are generated and published as events; consumers process sales through BOM explosion and inventory updates; the clock advances to 2pm for order-timeout checks and proposal generation; the clock advances to closing for outbox sweep and autonomy evaluation; on Mondays, schedule calibration and model polling run. After all engine jobs complete, the virtual operator handles proposals, deliveries, anomalies, autonomy grants, cycle counts, and schedule edits. The engine uses

the same consumer code, event bus, and database as a production deployment; only the clock and event source differ.

3 System capabilities and action boundaries

Tally's autonomous actions are bounded by design. This section defines what the engine was allowed to do, what required operator involvement, and what structural constraints limit downside exposure.

3.1 Autonomous actions

The engine autonomously generates demand forecasts with prediction intervals; maintains perpetual inventory state through BOM-exploded sales depletions; generates purchase order proposals with full provenance (forecast values, quantile grids, correction factors, bounds checked); places qualifying orders directly after earning per-supplier autonomy; produces weekly staff schedules via constraint solver; issues operational anomalies with templated evidence sentences; and adapts forecasting bias, order-edit bias, shrinkage rates, anomaly thresholds, and staffing ratios through bounded correction factors.

3.2 Actions requiring the operator

The operator must grant per-supplier autonomous ordering authority (the engine proposes, the operator confirms). Configuration changes, physical inventory counts, and actions outside configured autonomy bounds require operator involvement. Before autonomy is granted for a given supplier, every order to that supplier requires explicit operator approval.

3.3 Structural safety bounds

The system holds no payment credentials. Its only real-world actuator is an email requesting goods from a supplier. Every autonomous order is bounded by a per-supplier maximum order value, a cumulative spend ledger with daily and weekly caps, and a novelty gate that escalates any order significantly different from the trailing pattern. Bounds are validated independently at the proposer (ordering engine) and at the executor (email notifier): a bug or bad state in the ordering engine cannot cause the notifier to send an out-of-bounds order. The worst reachable state is bounded excess inventory within the configured caps.

An email outbox with idempotency keys prevents duplicate sends across crash recovery. A global dry-run flag (default on) prevents any real supplier communication until explicitly disabled at launch.

4 Evaluation methodology and results

Table 3: Evaluation metrics by operational domain.

Domain	Primary metric	Secondary metrics
Forecasting	MAE	Skill, interval coverage
Inventory	Shrinkage rate	Underflow count, count discrepancies
Ordering	Restock coverage	Autonomous spend, order count
Scheduling	Coverage gaps	Shifts per week, edits
Anomalies	Feedback rate	Suppression rate, type distribution
Autonomy	Time to authority	Proposals, approval rate
Adaptation	Factor bounds compliance	Clamp saturation, history depth

4.1 Forecasting

The engine runs four concurrent forecasting models: a trailing 7-day mean (baseline), a seasonal naive (same-weekday average), a Poisson GLM with weather and calendar features (Pedregosa et al., 2011), and a LightGBM gradient-boosted model (Ke et al., 2017). The champion model is selected through automated rolling-origin backtesting (Tashman, 2000): a challenger is promoted only when

it beats the incumbent on both skill (WAPE improvement) and prediction-interval coverage. All four models forecast daily total demand. Per-item forecasts are derived by decomposing the total through learned day-of-week-keyed share vectors (Gross and Sohl, 1990; Hyndman and Athanasopoulos, 2021), which pool statistical strength across items rather than fitting sparse per-item series directly.

Fifty-seven backtests ran over the simulation year. The two trained models traded the champion position repeatedly: LightGBM challenged Poisson GLM (6 passes out of 35 tests), and Poisson GLM challenged back (5 passes out of 22 tests). LightGBM was first promoted on February 26 and held the final champion position with a skill score of 0.026, where skill $= 1 - \text{WAPE}_{\text{challenger}}/\text{WAPE}_{\text{incumbent}}$, and interval coverage of 93.2% against a 90% nominal target.

Table 4: Full-year forecast accuracy by model (MAE in units). Observation counts differ because models generate forecasts only while active as champion or challenger. These figures represent observed production-path accuracy, not a controlled common-origin comparison.

Model	Avg MAE	Observations
Seasonal naive	32.49	9,898
Poisson GLM	33.13	5,852
LightGBM	37.59	4,130
Trailing 7-day mean	53.32	9,982

LightGBM held the final champion position despite higher aggregate production-path MAE than seasonal naive. This is not evidence that LightGBM is a superior point forecaster in this environment. The champion gate evaluates online WAPE improvement jointly with interval coverage over the active evaluation window, and the models are evaluated over different active periods. This result demonstrates adaptive model-selection infrastructure, not forecasting-model superiority.

The seasonal naive was competitive with both trained models on aggregate MAE. This is expected: the synthetic demand's dominant signals (day-of-week and seasonal level) are exactly what naive captures. The trained models incorporate temperature-conditional effects, and the prediction intervals provide the uncertainty estimates consumed by the ordering policy. The ordering engine consumes quantile grids, not point forecasts, so interval quality matters more than point-forecast MAE for operational decisions. Prediction intervals are constructed from empirical forecast-error quantiles over a rolling calibration window; the window length and calibration procedure are proprietary. All 4,464 item-level demand forecasts produced non-null quantile grids. This count includes 168 forecasts (14×12) whose target dates extend into January 2020, reflecting the engine's 14-day forecast horizon operating normally through the simulation boundary.

Section 7 addresses the demand-generator alignment question directly.

4.2 Inventory

The engine maintains perpetual inventory through BOM-exploded sales depletions, delivery receipts, shrinkage estimates, and physical count corrections. Over the year, 97,046 usage transactions, 450 restock transactions, and 4,015 shrinkage transactions were recorded.

208 physical counts were performed: 58 regular operator-initiated counts and 150 automated correction counts. Of 178 underflow anomalies, 150 required position corrections; the remaining 28 resolved naturally through intervening deliveries before the operator responded. Of 846 individual count lines, 811 recorded nonzero discrepancies, reflecting both portioning variance (hand-scooped servings vary) and the measurement noise introduced by the virtual operator ($\pm$0.5 kg for weight-measured items, $\pm$5 units for count-measured items). These perturbations are consistent with the configured measurement noise, not systematic inventory errors.

The aggregate shrinkage rate was 2.28% of usage volume. Four per-category shrinkage correction factors settled within the range 0.014–0.077, with 180 total evidence observations.

178 inventory underflow anomalies occurred over the year. The monthly distribution reveals a three-phase pattern: 24 in January (cold-start on template defaults), declining to 5 in April and June as ordering calibrates, spiking to 27–29 during July and August, the peak demand season (daily volumes 65% above winter levels with higher variance), and declining to 2 in December as the system stabilized through a full seasonal cycle.

For perishable items (ice cream, whipped cream), the ordering policy uses a newsvendor-style critical ratio (Arrow et al., 1951): the service level is set where the marginal cost of a stockout equals the marginal cost of spoilage. This policy deliberately accepts stockout risk when over-ordering would produce spoilage losses exceeding the cost of lost sales. The observed underflows on perishable items are consistent with the stockout-spoilage tradeoff permitted by the configured newsvendor-style policy and are not, by themselves, evidence of a system failure.

4.3 Ordering

Of 366 purchase orders generated, 353 were received ($26,602 total value) and 13 were cancelled by the virtual operator (3.6% of all generated purchase orders). All cancellations occurred among operator-reviewed proposals, for which the configured cancellation probability was 5%. All orders were initially created by the system: before autonomy, as proposals requiring approval; after autonomy, as direct orders within bounds.

Per-supplier promotion was evaluated through a deterministic promotion protocol. The engine tracked proposal count, approval rate, edit magnitude, rejection streaks, and critical failures per supplier. When gates passed (at least 12 proposals over at least 3 weeks, at least 90% approval rate, zero critical failures), the system proposed its own promotion with the evidence attached.

Table 5: Per-supplier ordering and autonomy (received orders only).

Supplier	Orders	Autonomous	Spend	Auto spend	Granted
Dairy Farms Co-op	128	100	$17,723	$15,748	Feb 18
Sunrise Bakery	160	19	$7,322	$1,868	Feb 6
Sweet Toppings	65	5	$1,557	$240	May 11
Total	**353**	**124**	**$26,602**	**$17,856**	—

Sunrise Bakery was granted autonomy first (February 6), followed by Dairy Farms Co-op (February 18) and Sweet Toppings (May 11). The slower timeline for Sweet Toppings reflects its 5-day lead time and 7-day review period, which produces fewer ordering opportunities and therefore requires more calendar time to accumulate the 12-proposal evidence threshold.

124 orders were placed autonomously after supplier grants, totaling $17,856 through the spend ledger. Restock coverage ranged from 96.3% to 116.5% across items. Restock coverage measures cumulative restocked volume as a percentage of cumulative usage volume; it does not directly measure fill rate or service level.

Order quantities follow an order-up-to policy (Scarf, 1960): for each item, the target stock level is derived from forecast demand quantiles over the supplier's delivery horizon, net of current inventory position and on-order quantities, rounded to pack size.

The decision log recorded 470 ordering decisions: 451 in forecast mode and 19 in par mode (static reorder-point defaults, used before forecast calibration completed in January). A single purchase order may contain multiple item-level decisions; the decision log therefore contains more entries than the 366 purchase orders generated. Every decision stored a full provenance snapshot: the forecast values, quantile grid, correction factors applied, bounds checked, and calibration state at the time of the decision.

4.4 Scheduling

The scheduling engine uses OR-Tools CP-SAT (Google LLC, 2024) to produce weekly staff schedules. Hard constraints enforce employee availability, maximum weekly hours, minimum and maximum shift lengths, and keyholder presence during all operating hours. Coverage requirements are soft constraints with high penalties; unmet coverage appears as reported shortfalls.

52 weekly schedules were published over the year, covering every operating week. 1,009 individual shifts were assigned, averaging 19.4 shifts per week and 110.3 total scheduled hours per week. Zero coverage gaps were recorded. A coverage gap is defined as any scheduled operating hour in which assigned staff fell below the minimum staffing requirement; none occurred in any published schedule. 26 schedule edits were applied by the virtual operator (one per week, all shift-end-time modifications).

These edits fed the staffing-ratio correction factor. Six staffing-ratio factors reached the lower clamp of 0.70, indicating that repeated operator feedback drove these factors toward the configured lower bound relative to the initial template staffing ratios.

4.5 Anomaly detection

The anomaly engine combines forecast-residual checks with deterministic threshold rules. 409 anomalies were raised over the year. Of these, 268 were included in the daily summary email and 141 were suppressed by deduplication or cooldown rules. Suppression controls summary inclusion only; the operator received and responded to all 409 anomalies.

Table 6: Anomaly distribution and operator feedback.

Type	Count	Acknowledged	Dismissed
Inventory underflow	178	178	0
Forecast residual	129	110	19
Overdue delivery	84	63	21
Count discrepancy	18	15	3
Total	**409**	**366**	**43**

Over 26,369 sales, this corresponds to 6.8 underflow events and 4.9 forecast-residual alerts per 1,000 transactions.

Each anomaly includes a templated evidence sentence generated from the values that triggered it, requiring no language model.

4.6 Learning and adaptation

All parameter adaptation uses a single mechanism: bounded correction factors with exponential decay, clamped to a configured range, with full change history. 23 correction factors were active across five categories, with 954 total history entries recording every parameter change.

Table 7: Correction factor summary.

Kind	Factors	Avg value	Range	Evidence
Forecast bias	4	1.053	1.038–1.073	706
Order-edit bias	8	1.052	0.891–1.250	38
Shrinkage rate	4	0.033	0.014–0.077	180
Staffing ratio	6	0.702	0.700–0.708	12
Threshold sensitivity	1	0.976	—	18

Six factors reached their configured clamps: two order-edit bias factors pinned at the upper bound of 1.25 (the system learned to order 25% more for those items, the maximum permitted adjustment), and four staffing-ratio factors pinned at the lower bound of 0.70. When a factor is pinned at its clamp for consecutive updates, the system raises a structural-change anomaly rather than continuing to push against the bound.

The forecasting slow path ran 57 backtests using rolling-origin evaluation. The champion/challenger gate prevented promotion of models that won on point accuracy but failed on interval coverage, and automatically demoted champions when a challenger demonstrated sustained improvement.

4.7 Notification system

489 outbound emails were generated in dry-run mode over the year: 365 daily operational summaries and 124 autonomous purchase orders. The email outbox prevents duplicate sends across crash recovery through idempotency keys. No notification-path failures were observed during the simulation run.

5 Failures discovered during iterative simulation

The simulation environment surfaced engineering bugs that were not anticipated during development. Table 8 summarizes all six failures, their detection signals, root causes, and verification.

Table 8: Engineering failures discovered during iterative simulation.

Failure	Detection signal	Root cause	Verification
Quantile grid inversion	~50% systematic under-ordering	Quantile labels described error rank, not demand rank	Restock coverage recovered to configured range
Reorder point filter	Forecast-driven orders blocked during seasonal demand increase	Static threshold redundant in forecast mode	Proactive ordering verified across seasonal transitions
Shrinkage floor race	Inventory positions went negative between pre-check and write	Time-of-check/time-of-use race; cap logic outside transaction boundary	Atomic capping enforced; no subsequent negatives
Scheduling contiguity	Solver produced non-contiguous shift assignments	Triple-based constraint missed single-slot gaps	Auxiliary boolean pattern enforced; no subsequent gaps
Novelty gate averaging	Novelty gate failed to flag unusual orders	Flat SQL average underweighted recent patterns	EWMA replaced flat average; novel orders correctly escalated
Bias self-chase	Forecast bias correction amplified its own signal	Correction applied before recovering raw prediction	Raw prediction recovered before observation; bias converged

Two failures are worth examining in detail. The quantile grid inversion was the most operationally impactful: the ordering engine was systematically under-ordering by approximately 50% across all items. Database inspection of the quantile grids revealed that the labels described forecast-error percentile rank rather than demand rank, effectively inverting the safety-stock calculation. The fix reversed values across quantile keys before applying the prediction-minus-residual calculation and clamped results to non-negative values. This bug would have been difficult to detect through unit tests because each component produced internally consistent outputs; the error was only visible in the aggregate ordering behavior over multiple simulated weeks.

The shrinkage floor race was the most architecturally instructive. A time-of-check/time-of-use race condition in stock depletion operations allowed inventory positions to go negative between the pre-check and the database write. The root cause was that the capping logic ran outside the transaction boundary, creating a window in which concurrent depletions could both pass the non-negativity check before either committed. The fix moved the capping logic inside the database transaction for atomic enforcement. This class of bug is invisible in single-threaded testing and only manifests under the concurrent event processing that the simulation exercises.

Each failure was diagnosed through SQL queries against the simulation database and verified corrected in the final 365-day run. The simulation's value as a test harness is that it surfaces integration and runtime failures that unit tests and code review did not anticipate.

6 Results summary

These results establish operational execution and architectural reliability within the simulated environment; they do not establish labor savings, inventory-cost reduction, or incremental business profit.

Table 9: Full-year simulation results.

Metric	Result
Simulated duration	365 days
Sales processed	26,369
Simulated sales revenue	$145,337
Inventory transactions	101,511
Purchase orders received	353
Autonomous orders	124
Autonomous purchasing volume	$17,856
Suppliers granted autonomy	3 of 3
Weekly schedules published	52
Coverage gaps	0
Owner schedule edits	26
Anomalies detected	409
Anomalies with operator disposition	409/409
Correction factors active	23
Factor history entries	954
Backtests executed	57
Null quantile grids	0%
Shrinkage rate	2.28%
Emails composed	489

Table 10: Evidence status of evaluation claims.

Claim	Evidence	Status
Full-year closed-loop operation	365-day simulation completed	Demonstrated
Per-supplier autonomy earned	3 of 3 suppliers promoted via deterministic protocol	Demonstrated
Bounded parameter adaptation	23 correction factors remained within configured bounds; 6 reached clamps	Demonstrated
Failure discovery through simulation	6 engineering bugs surfaced and corrected	Demonstrated
Forecast pattern recovery	Synthetic demand only	Partially demonstrated
Real supplier reliability	Dry-run email only	Not evaluated
Real customer demand	No live POS data	Not evaluated
Multi-business generalization	Single vertical template only	Not evaluated
Production deployment uptime	Simulation environment only	Not evaluated

7 Limitations and threats to validity

7.1 Demand-generator alignment

The synthetic demand generator uses day-of-week, temperature, and seasonal effects. The forecasting engine uses day-of-week, temperature, and seasonal features and estimates their effects from observed sales data. The forecaster is therefore partially recovering the generator's structure. No public dataset provides the combination required to evaluate this system end-to-end: single-location point-of-sale data at modifier-level granularity, aligned with daily weather, supplier ordering history, staffing records, and inventory counts over a full seasonal cycle. Existing retail datasets either operate at aggregate level, cover too short a period, or omit the operational dimensions (supplier, staffing, inventory) that Tally's loops depend on. The synthetic approach was adopted because it was the only way to test the complete integrated system. The generator was designed so that the engine receives only observable sales and weather inputs without access to the demand-generation process or its parameters.

The seasonal naive model's competitive aggregate MAE (32.49 vs 33.13 for Poisson GLM) confirms that the dominant demand signals are simple calendar effects that any reasonable baseline captures. The trained models incorporate temperature-conditional effects, and the prediction intervals provide the uncertainty estimates consumed by the ordering policy. The marginal improvement in point-forecast MAE is small because the problem is well-specified for naive methods. A real business would present demand patterns not present in the generator: local events, social-media driven traffic,

competitor behavior, and menu changes. These would increase the gap between naive and trained models, but that gap has not been demonstrated here.

This limitation does not undermine the engineering evaluation. It means the forecasting system's accuracy advantage over naive is untested against realistic demand complexity, while the forecasting infrastructure— including champion/challenger management, rolling backtests, bounded correction factors, and quantile-grid generation—was exercised continuously throughout the simulated year.

7.2 Supplier behavior

Simulated suppliers always deliver what was ordered (with timing variance and occasional short deliveries). They do not change prices, stock out, substitute items, or fail to communicate. Real supplier behavior introduces risks the ordering engine has not been tested against. The engine's architectural response to these risks exists (anomaly detection for price jumps and delivery failures, nudge timeouts for unconfirmed orders) but has not been exercised under realistic supplier conditions. This is a recognized constraint, and it informed a deliberate architectural decision: Tally sends the initial order email but does not automate supplier negotiation, confirmation, or dispute resolution. The owner remains in the communication loop for everything after the first outbound message. Price changes, substitutions, short deliveries, and miscommunications are handled by a human who can exercise judgment that no current model reliably provides. The system's autonomy ends at the point where real-world supplier complexity begins; the boundary is intentional.

7.3 Operator consistency

The virtual operator engages every day without exception. A real operator may ignore the system for days, defer decisions, or interact in patterns the simulation does not model. However, the operator's feedback is content-independent (random approval, random edits, random dismissals), which means the engine faces pure noise rather than learnable signal. A rational operator who edits for predictable reasons would provide signal the engine could learn from. The random operator therefore tests whether the adaptation mechanisms remain stable under uninformative feedback rather than whether they can exploit operator intent. Tally reduces daily operational involvement to minutes, but it does not eliminate the need for regular engagement. The system assumes an owner who checks in daily out of self-interest in their own business — reviewing summaries, responding to anomalies, approving or granting autonomy when prompted. The engine tolerates reasonable gaps: missed counts trigger reminder anomalies, deferred proposal approvals queue without data loss, and the ordering policy continues operating within existing authority during periods of inattention. However, the system is not designed for extended unattended operation. Prolonged disengagement would result in stale correction factors, unreviewed anomalies, and uncalibrated thresholds. Regular owner engagement is a stated operational requirement of the system.

7.4 Random seed and reproducibility

The simulation does not use a fixed random seed. Different runs produce quantitatively different results with the same structural properties. The reported numbers are from a single complete run; exact quantitative results are not reproducible from the disclosed configuration alone.

7.5 Scope of the evaluation

The simulation does not test integration with a live point-of-sale system, real email delivery to suppliers, DNS/SPF/DKIM configuration, real supplier responses, messy POS data (refunds, voids, partial transactions), equipment failures, or multi-location operation. These are deployment validation questions. The system's architecture handles each of these (POS sync with reconciliation, outbox with retry logic, anomaly detection for data quality issues), but the code paths have been exercised only against synthetic inputs.

The evaluation also does not test the system's behavior over multiple years. A single year of operation does not demonstrate long-term drift handling, major seasonal regime changes, or model degradation over time.

8 Reproducibility and proprietary components

Tally is proprietary software. The source code, model parameters, internal architecture, and implementation details are not publicly released. The evaluation methodology, simulation configuration, evaluation criteria, aggregate results, and known limitations are documented here to make the scope and interpretation of the experiment clear.

The following experimental parameters are disclosed: simulation duration (365 days), date range (2019), location and weather source (Open-Meteo, Vancouver), business configuration (Table 1), demand generation structure (day-of-week multipliers, temperature sensitivity, shock dates and magnitudes), virtual operator behavior (Section 2.4), and all measured outcomes. The generation process's exact parameters, the engine's internal algorithms, database schema, and deployment infrastructure are not disclosed.

The simulation exercises the production application code for event consumption, inventory mutation, ordering, scheduling, learning, and autonomy evaluation; only the event source, clock, and operator interface are replaced by simulation components. All consumer code, scheduling jobs, learning loops, and autonomy evaluation run identically.

Ethics and disclosure

Tally is not affiliated with any point-of-sale provider, supplier, or financial institution. The simulation uses real historical weather data from Open-Meteo's public archive; no customer, employee, or supplier data were collected or used. The system operates in dry-run mode by default and cannot send real communications or commit real financial transactions until explicitly enabled. Institutional affiliations are provided for author identification only. The authors used AI-assisted tools for drafting support and structural refinement. All claims, analyses, and final text were reviewed and approved by the human authors, who take full responsibility for the work.

References

Open-Meteo. Open-Meteo: Free weather API. `https://open-meteo.com`, 2024. Historical weather archive and forecast API.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. LightGBM: A highly efficient gradient boosting decision tree. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

Leonard J Tashman. Out-of-sample tests of forecasting accuracy: an analysis and review. *International Journal of Forecasting*, 16(4):437–450, 2000.

Charles W Gross and Jeffrey E Sohl. Disaggregation methods to expedite product line forecasting. *Journal of Forecasting*, 9(3):233–254, 1990.

Rob J Hyndman and George Athanasopoulos. *Forecasting: Principles and Practice*. OTexts, Melbourne, Australia, 3rd edition, 2021. `https://otexts.com/fpp3/`.

Kenneth J Arrow, Theodore Harris, and Jacob Marschak. Optimal inventory policy. *Econometrica*, 19(3):250–272, 1951.

Herbert Scarf. The optimality of (S, s) policies in the dynamic inventory problem. In Kenneth J Arrow, Samuel Karlin, and Patrick Suppes, editors, *Mathematical Methods in the Social Sciences*, pages 196–202. Stanford University Press, 1960.

Google LLC. OR-Tools: Google optimization tools. `https://developers.google.com/optimization`, 2024. CP-SAT constraint programming solver.