Reinforcement Learning for Restaurant Inventory & Preparation Management (Profit Objective)

Research & Project Plan (Living Document)

Team

November 3, 2025

Status & Scope

Status: Living document to be updated as design and experiments progress.
Scope: Build, evaluate, and document a reinforcement learning (RL) system that *jointly* orders raw ingredients and schedules in-house preparations to maximise expected discounted profit under budget, storage, lead-time, perishability (first-in–first-out; *FIFO*), and labour constraints. The policy must function across heterogeneous restaurants with variable menus and changing item sets.

1 Technical Implementation Sketch (from formulation notes)

1.1 Item taxonomy, demand coupling, and production

- **Two inventory types.** (i) *Raw ingredients* purchased from suppliers; (ii) *Preparations* (processed ingredients) produced in-house. Preparations consume raw inputs and *staff-hours* and have their own shelf-lives. We deplete stock via **FIFO** and track age-buckets.
- **Dish-driven demand.** We forecast per-dish demand $\hat{d}_{k,t}$ and translate to inventory requirements using a bill-of-materials (**BOM**) matrix $A \in \mathbb{R}^{N \times M}$:

$$x_{i,t}^{\mathrm{req}} = \sum_{k=1}^{M} A_{ik}\, \hat{d}_{k,t}.$$

- **Preparation production.** Daily prep decisions $y_{j,t} \geq 0$ for preparation j consume raw inputs $\sum_i B_{ij} y_{j,t}$ (prep-recipe matrix B), use τ_j staff-hours per unit, and yield new prep inventory at age 0.

1.2 MDP (*Markov decision process*) and state

Daily periodic review. State s_t concatenates global and item-wise components:

$$s_t = \Big(\underbrace{\mathrm{calendar}_t,\, B_t,\, C,\, \bar{H}_t}_{\text{global: budget, capacity, staff-hours}} ,\quad \underbrace{\{z_{i,t}\}_{i \in \mathcal{I}_t}}_{\text{variable set of item tokens}} \Big),$$

where each item token $z_{i,t}$ includes: on-hand age histogram, pipeline-by-days-to-arrival, purchase cost c_i, holding cost h_i, unit space u_i, case pack/minimum order multiple m_i, fixed order cost K_i, shelf-life L_i^{life}, price attribution p_i (from dish mix), recent sales, and forecast summaries for implied usage. For preparations: labour τ_i and yield parameters. The index set $\mathcal{I}_t$ changes over time (items appear/disappear) and across restaurants.

Permutation-invariant policy for an unfixed item set. We use a DeepSets/attention encoder–decoder to handle a variable number of items:

$$D_i = \mathrm{enc}_\theta(s_t^{\mathrm{global}},\, z_{i,t}), \quad g_t = \mathrm{pool}\big(\{D_i\}\big), \quad \tilde{a}_{i,t} = \mathrm{dec}_{\theta'}(D_i, g_t),$$

with shared weights across items (*SKU*-wise parameter sharing; SKU = stock-keeping unit). The decoder emits continuous proposals $(\tilde{q}_{i,t}, \tilde{y}_{i,t})$ for raw orders and prep batches.

Alternative: multi-agent formulation. As an alternative to the set-encoder, we can model each item i as an *agent* with parameter sharing across agents and a centralised critic (or value mixing). Communication/coordination can be implemented via attention over messages or mean-field signals. This naturally accommodates a variable item set, allows per-item exploration schedules, and enables explicit credit assignment, while keeping training stable through centralised training with decentralised execution.

1.3 Constraints and feasibility projection

Project proposals to feasible $(q_{i,t}, y_{j,t})$:

$$\sum_i c_i q_{i,t} \le B_t, \quad \sum_i u_i(I_{i,t} + q_{i,t}) + \sum_j u_j^{\mathrm{prep}} y_{j,t} \le C, \quad \sum_j \tau_j y_{j,t} \le \bar{H}_t, \quad q_{i,t} \in m_i\mathbb{Z}_+.$$

Projection uses a greedy knapsack routine ranking items by expected marginal profit-per-cost/per-space, with rounding to case packs. This stabilises training and guarantees constraint satisfaction.

1.4 Transitions with FIFO

Arrivals are dequeued from pipeline; new orders are enqueued at their lead-time slot. Preparation batches consume raw inputs via FIFO and create prep stock at age 0. Dish demand is sampled (Section 1.6); implied item usage is $A\,D_t^{\mathrm{dish}}$; sales are limited by available stock, consuming oldest ages first. Age advance and expiry produce spoilage write-offs.

1.5 Profit objective with explicit lost-sales penalty

Let $S_{k,t}^{\mathrm{dish}}$ be realised dish sales and margin_k the dish unit margin. Per-period reward:

$$\begin{aligned} r_t = {} & \underbrace{\sum_k p_k S_{k,t}^{\mathrm{dish}}}_{\text{revenue}} - \underbrace{\sum_i c_i q_{i,t} + \sum_{i:\, q_{i,t}>0} K_i}_{\text{purchasing}} - \underbrace{\sum_i h_i I_{i,t+1}}_{\text{holding}} - \underbrace{\sum_i \kappa_i\, \mathrm{Spoil}_{i,t}}_{\text{spoilage}} \\ & - \underbrace{w \cdot \sum_j \tau_j y_{j,t}}_{\text{labour}} - \underbrace{\lambda \sum_k \big(d_{k,t} - S_{k,t}^{\mathrm{dish}}\big)_+ \mathrm{margin}_k}_{\text{lost-sales penalty}}, \end{aligned}$$

with $\lambda \in [0,1]$ (usually small since lost sales already reduce revenue). Set κ_i slightly above replacement cost to discourage waste.

1.6 Forecasting and simulator sampling

A light quantile forecaster (shared trunk; per-dish heads) outputs $\{Q_{0.5}, Q_{0.8}, Q_{0.95}\}$ over a horizon $H \geq \max_i L_i$, trained with pinball loss. We use MC-dropout or small ensembles to obtain uncertainty. Simulator samples dish demand from these distributions conditional on calendar and events. Forecast summaries (means/quantiles/variance) are included in state so the policy is *forecast-aware*. This module is replaceable without touching the controller.

1.7 Algorithms

Actor–critic with: (i) global trunk for shared context; (ii) item towers with shared parameters; (iii) attention pooling to form a critic context. Start with PPO (proximal policy optimisation; stable on-policy) with entropy bonus; compare to SAC (soft actor–critic; off-policy continuous control). Optionally evaluate differentiable-simulator objectives (e.g., pathwise/DirectBackprop) once the base PPO is stable.

2 Simulator: Pythonic Pseudocode (illustrative)

```
class KitchenEnv:
    def __init__(self, params, A_bom, B_prep, forecaster):
        self.A = A_bom          # [N x M] items <- dishes
        self.B = B_prep         # raw usage per prep unit
        self.fcst = forecaster  # dish-level quantile model with sampling
        self.cost = params.cost     # c_i, K_i, h_i, p_k, margin_k, kappa_i, w
        self.conf = params.conf     # L_i, m_i, u_i, shelf_i, capacity C
        self.hours = params.hours   # tau_j, daily limit H_bar(t)
        self.budget = params.budget # B_t schedule
        self.calendar = params.calendar
        self._init_state()

    def _project(self, q_raw, y_prep):
        q_raw = round_to_casepacks(q_raw, m=self.conf.m)
        y_prep = cap_to_hours(y_prep, tau=self.hours.tau, H_bar=self.hours.today())
        q_raw, y_prep = knapsack_project(q_raw, y_prep,
                                         costs=self.cost, space=self.conf.u,
                                         C=self.conf.C, B=self.budget.today())
        return q_raw, y_prep

    def step(self, action):
        q, y = self._project(action.q, action.y)

        # Arrivals & pipeline rotation
        A = self.P[:, 0]
        self.P = shift_left(self.P)
        self.P[np.arange(self.N), self.conf.L-1] += q

        # Produce preparations (consume raws by FIFO; add prep at age 0)
        raw_need = self.B @ y
        self.I_age = fifo_consume(self.I_age, raw_need)   # raises if infeasible
        self.I_age[prep_ids, 0] += y

        # Sample dish demand and translate to item usage
        d_dish = self.fcst.sample(self.calendar.features())
```

```
        use_items = self.A @ d_dish

        # FIFO sales & ageing
        S_items, I_next, spoil = fifo_sell_and_age(self.I_age, A, use_items,
                                                   shelf=self.conf.shelf)
        # Map S_items back to dish sales for revenue allocation if needed
        S_dish = infer_dish_sales(S_items, self.A)

        # Costs and reward
        H_use = float(self.hours.tau @ y)
        revenue  = float(prices @ S_dish)
        purchase = float(c @ q) + float((q > 0) @ K)
        holding  = float(h @ I_next.sum(axis=1))
        lost_pen = float(margins @ np.maximum(d_dish - S_dish, 0.0)) * lambda_
        spoilage = float(kappa @ spoil)

        r = revenue - purchase - holding - spoilage - w * H_use - lost_pen
        self.I_age = I_next
        obs = self._make_obs()
        done = self.calendar.is_terminal()
        info = {"profit": r}
        return obs, r, done, info
```

3 Baselines and Controls

1. **Heuristic (no optimisation): Demand + safety stock.** For each item, compute lead-time demand using forecast medians and add a fixed safety stock (e.g., based on historical service-level quantiles or a simple multiple of forecast σ):

$$q_{i,t}^{\text{heur}} = \Big[(\hat{\mu}_{i,L} + z_\alpha \hat{\sigma}_{i,L}) - \text{stock\_position}_{i,t}\Big]_+,$$

   followed by budget/capacity-aware proportional downscaling and rounding to case packs. **This heuristic also serves as a *warm-start policy*** for RL by (a) initialising the actor via behaviour cloning on simulated trajectories, or (b) mixing its actions via ϵ-greedy during early training.

2. **Static par levels (control).** Fixed days-of-cover targets per item (menu-category specific), adjusted seasonally; case-pack rounding only. No optimisation beyond feasibility checks.

4 Training Plan

Data

Daily dish sales; BOMs (dishes $\to$ ingredients/preps); item costs c_i, dish prices p_k and margins; pack size m_i; unit space u_i; fixed order costs K_i; shelf-lives; supplier lead-times; labour rates and prep labour τ_j. Calendar features: day-of-week, week-of-year, holidays, events.

Curriculum and domain randomisation

Begin with small item sets, no fixed order costs, short lead-times, ample capacity. Gradually introduce K_i, tighter $B_t, C, \bar{H}_t$, longer lead-times, delivery uncertainty, and menu rotation (items join/leave the set). Randomise costs/lead-times/budgets/capacity within 10 %–20 % to improve transfer across restaurants.

Evaluation metrics

Episode profit, gross margin, fill-rate, stockout-days, inventory turns, waste rate, average and $p95$ on-hand value, % capacity used, staff-hours, order frequency. Stress tests: holiday spikes; supplier delays.

5 Shadow/Offline Evaluation Protocol

Train forecaster on historical data; replay policy decisions offline using recorded receipts/sales; recompute KPIs subject to observed budgets/capacity/staff. Compare against baselines with paired tests and cost-component breakdowns.

6 Open Questions for Literature Review (to resolve with provided sources)

1. **Set-encoder vs. multi-agent control.**
   - 1.a) Under what conditions (state/action coupling, scale, constraint structure) do permutation-invariant set encoders with shared towers outperform multi-agent (parameter-shared) policies with a central critic?
   - 1.b) What coordination mechanisms (attention pooling, message passing, value mixing) are most sample-efficient for variable item sets and changing menus?
   - 1.c) How does credit assignment compare (variance, stability) across the two formulations in inventory domains with hard feasibility projections?
2. **Heuristic warm-start effectiveness.**
   - 2.a) Does behaviour cloning from a demand + safety-stock heuristic measurably improve early-stage learning curves, and what is the best mixture schedule (e.g., ϵ-greedy blending vs. supervised pretrain)?
   - 2.b) Which safety-stock constructions (lead-time quantiles vs. $(\mu + z_\alpha \sigma)$) provide the most effective initialisation under lost-sales and fixed order costs?
3. **Profit shaping and lost-sales penalties.**
   - 3.a) What ranges of loss-scaling λ (relative to dish margin) lead to faster convergence without distorting the asymptotic profit optimum?
   - 3.b) Are there principled potential-based shapings for profit (including spoilage and labour) that preserve optimal policies?
4. **Perishability modelling granularity.**
   - 4.a) What age-bucket resolution is empirically sufficient for FIFO without incurring prohibitive state size?
   - 4.b) Is waste cost better modelled as replacement cost, opportunity cost, or a calibrated overage factor to reflect operational externalities?
5. **Feasibility projection design.**

5.a) Which projection heuristics (greedy knapsack by margin-per-cost/per-space vs. proportional scaling) offer the best stability/return trade-off when embedded in the policy?

5.b) What rounding schemes to case packs minimise bias (e.g., stochastic rounding vs. deterministic nearest-multiple) under budget and capacity?

6. **Preparation decisions and labour coupling.**

6.a) How should prep batching and staff-hour costs be represented (continuous vs. integer batches) to balance realism and learnability?

6.b) Are there known policies for smoothing labour utilisation (e.g., variance penalties) that improve profit/waste without harming fill-rate?

7. **Forecast-aware vs. forecast-free states.**

7.a) Which forecast summaries (quantiles, moments, encoder tokens) most improve policy performance and robustness?

7.b) How sensitive is the policy to forecast misspecification, and which uncertainty treatments (MC-dropout, ensembles) mitigate degradation?

8. **Differentiable simulation (*exo*-MDP) vs. standard RL.**

8.a) In inventory tasks with exogenous demand and feasibility layers, when do pathwise/DirectBackprop objectives outperform PPO/SAC in sample efficiency and final return?

8.b) What reparameterisations are recommended for demand sampling and FIFO ageing to enable low-variance gradients?

9. **Domain randomisation and cross-restaurant transfer.**

9.a) What distributions and magnitudes of randomisation (lead times, MOQs, costs, capacity) yield the best out-of-domain generalisation?

9.b) How should restaurant/context embeddings be constructed (one-hot vs. learned features) for transfer without overfitting?

10. **Evaluation protocol details.**

10.a) Which metrics beyond profit (e.g., p95 on-hand value, waste rate, order frequency, labour variability) are most predictive of field success?

10.b) What stress scenarios (supplier delays, holiday spikes, menu rotations) are standardised in the literature for perishables?

11. **Demand censoring and data issues.**

11.a) How should censored sales (stockouts) be handled in forecaster training and offline evaluation to avoid optimistic/biased results?

11.b) Are there recommended filters or imputation strategies for building training windows with "healthy" on-hand conditions?

Sources

#	Citation	ID (DOI / arXiv / name)	Summary (relevance)	Takeaways for our plan	Notes
1	Madeka, Torkkola, Eisenach, Luo, Foster, Kakade (2022), *Deep Inventory Management.*	arXiv:2210.03137	DRL with lost sales/lead-times; differentiable exogenous simulator and direct backprop show strong performance vs OR baselines.	Consider differentiable variants once PPO is stable; encode inventory structure in policy.	link
2	Maggiar, Andaz, Bagaria, Eisenach, Foster, Gottesman, Perrault-Joncas (2025), *Structure-Informed DRL for Inventory.*	arXiv:2507.22040	Structure-aware policies (lead-times, stock position) and direct objectives across classical settings including perishables.	Use permutation-invariant item towers and structure tokens; aids generalisation.	link
3	Sultana, Singh, et al. (2020), *Reinforcement Learning for Multi-Product Multi-Node Inventory Management.*	arXiv:2006.04037	Multi-product, capacity-coupled networks; actor–critic at scale.	Guides constraint handling and parameter sharing across large SKU sets.	link
4	Kara & Doğan (2018), *Reinforcement learning approaches for determining the optimal ordering policy for perishable inventory systems. Expert Systems with Applications.*	10.1016/j.eswa.2017.08.046	Age-aware state and FIFO materially improve control for perishables; RL exceeds heuristics.	Include explicit age buckets and FIFO; model waste in reward.	link
5	(EJOR Survey) *Reinforcement learning in inventory management: A roadmap* (2022).	10.1016/j.ejor.2021.07.016	Comprehensive survey; benchmarks, evaluation pitfalls, metrics.	Informs baselines, ablations, and robustness metrics.	link
6	Kavoosi et al. (2025), *Dynamic pricing and perishable inventory management using deep RL. Expert Systems with Applications.*	Name: S095741742502189X	Joint pricing–ordering with perishables; continuous actions; vendor-managed context.	Blueprint for optional pricing extension with action pruning.	link
7	Nomura & Liu (2025), *Deep RL for Dynamic Pricing and Perishable Inventory. Applied Sciences.*	10.3390/app15052421	Age-dependent demand with DRL; discusses invalid-action pruning.	Reinforces feasibility layer and action pruning ideas.	link
8	Kaur & Prakash (2025), *Adaptive Inventory Strategies using DRL for Agri-Food.*	arXiv:2507.16670	DRL for perishable agri-food; uncertainty and robustness.	Stress-test patterns for perishables and uncertainty handling.	link

#	Citation	ID (DOI / arXiv / name)	Summary (relevance)	Takeaways for our plan	Notes
9	(Tech/whitepaper) *End_to_End_revision3-10.pdf.*	Name: End_to_End_revision3-10	Industrial implementation details (pipelines/KPIs/offline testing).	Implementation templates for evaluation and deployment hygiene.	(no public lin
10	Medium (R. Ghosh), *Reinforcement Learning in Inventory Management: Leveraging AI for Optimal Order Management.*	Blog name	Pedagogical single-node sim and RL agents.	Simulator scaffolding and sanity-check baselines.	link
11	Towards Data Science, *RL for Inventory Optimisation Series II: Multi-Echelon.*	Blog name	Multi-echelon example with PPO and action discretisation.	Reference patterns for networked settings and baselines.	link
12	Medium (P. Kor), *Optimising Inventory Management with RL: A Hands-on Python Guide.*	Blog name	Q-learning tutorial; end-to-end code structure.	Quick prototype and unit-test fodder for the sim.	link