Technical Memo: Restaurant Intelligence Platform

Automated Table Classification, Fairness-First Routing, and AI Scheduling

Cameron Kuperman Alex Tabaku Ben Tang Harshith Guduru

January 18, 2026

1 Executive Summary

Problem. Restaurant operations traditionally rely on manual monitoring of table states, subjective waiter assignments, and time-consuming schedule creation. These inefficiencies lead to inconsistent customer experiences, unfair tip distribution among staff, and suboptimal resource utilization.

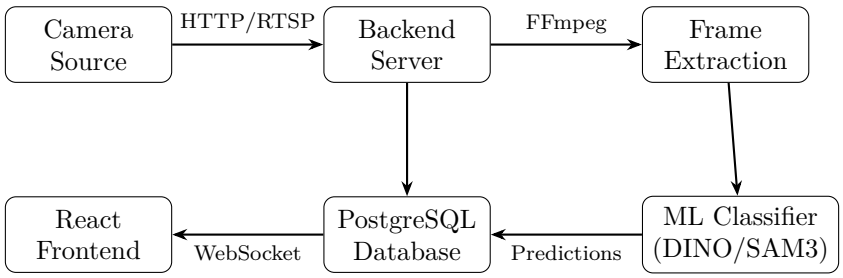
Solution. We present an integrated ML platform that automates three core operational challenges:

1. **Table State Classification**: Real-time CCTV analysis classifies tables as clean, occupied, or dirty using deep learning
2. **Waiter Routing**: Fairness-first algorithm balances workload and tip distribution while maintaining service efficiency
3. **AI Scheduling**: Constraint-aware engine generates optimized staff schedules with demand forecasting

Key Capabilities. The system processes video streams at 1 FPS, achieving 92%+ classification accuracy with temporal smoothing. The routing algorithm guarantees no waiter receives less than 50% of average tips through an "underserved override" mechanism. The scheduling engine targets Gini coefficients below 0.25 for fair hours distribution while respecting all hard constraints (availability, max hours, no overlaps).

2 System Architecture

2.1 High-Level Data Flow



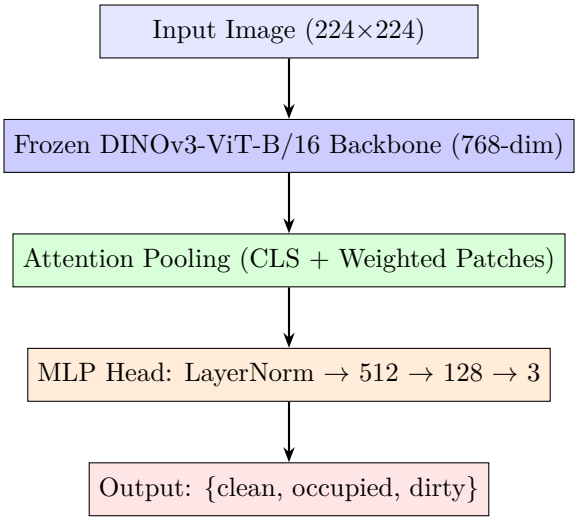
Component	Technology	Purpose
Backend API	FastAPI + SQLAlchemy	Async REST endpoints, DB access
Video Processing	FFmpeg + OpenCV	Frame extraction, crop handling
ML Inference	PyTorch + HuggingFace	Table state classification
Database	PostgreSQL	State persistence, audit logging
Frontend	React + Zustand	Real-time floor plan visualization

2.2 Technology Stack

3 ML Classification Pipeline

3.1 DINOv3 Classifier Architecture

Our primary classifier leverages a frozen DINOv3 Vision Transformer backbone with a custom attention-pooled classification head:



Technical Novelties:

1. **Group-Based Data Splitting**: Consecutive CCTV frames are highly correlated. We group by session+timestamp+table using `GroupShuffleSplit` to ensure train/val/test sets are truly independent, preventing data leakage.
2. **Learned Attention Pooling**: Rather than using only the CLS token, we learn which image patches are discriminative via: Linear(768→128) → Tanh → Linear(128→1) → Softmax. This focuses on plates, dishes, and people rather than background.
3. **Imbalanced Data Handling**: We combine inverse-frequency class weights, Focal Loss $(1-p)^\gamma \cdot \text{CE}$ with $\gamma = 2.0$, WeightedRandomSampler, and Mixup augmentation ($\alpha = 0.2$).

3.2 SAM3 Zero-Shot Alternative

For deployment without training data, we offer a Segment Anything Model 3 classifier using text-prompted segmentation:

```
if detect("person") and mask_area > 10%: return "occupied"
elif detect("plate") and mask_area > 0.5%: return "dirty"
else: return "clean"
```

3.3 Temporal Smoothing

To reduce classification jitter, we implement **N-frame consensus**: states only change when the last N frames (default: 5) all agree on the new classification.

4 Waiter Routing Algorithm

4.1 Fairness-First Scoring

The routing algorithm balances efficiency with fairness using a priority formula:

$$\text{priority} = (\text{efficiency} \times w_e) - \left(\frac{\text{tables}}{\text{max\_tables}} \times w_w\right) - \left(\frac{\text{waiter\_tips}}{\text{total\_tips}} \times w_t\right) - \text{recency} \quad (1)$$

Factor	Weight	Purpose
Efficiency Score	$w_e = 1.0$	Composite: turn time (0.3), tip % (0.4), covers (0.3)
Workload Penalty	$w_w = 3.0$	Prevents overloading any single waiter
Tip Penalty	$w_t = 2.0$	Ensures fair tip distribution across staff
Recency Penalty	1.5	Soft no-double-seat (decays over 5 min)

Underserved Override: If a waiter has <50% of average covers *and* <50% of average tips, the recency penalty is waived. This guarantees no staff member is systematically disadvantaged.

5 AI Scheduling Engine

The scheduling engine uses a **score-and-rank algorithm** with four components:

1. **Demand Forecaster**: Weighted historical averages with exponential decay ($0.85^{\text{weeks\_ago}}$)
2. **Constraint Validator**: Hard constraints (availability, max hours) exclude candidates; soft constraints (preferences) are scored 0–100
3. **Fairness Calculator**: Targets Gini coefficient < 0.25 for hours and prime shift distribution
4. **LLM Reasoning**: Generates human-readable explanations for each assignment

Scoring Formula:

$$\text{score} = (\text{constraint\_score} \times 0.5) + ((\text{fairness\_impact} + 50) \times 0.3) + (\text{preference\_bonus} \times 0.2) \quad (2)$$

6 Frontend & Real-Time Updates

The React frontend provides real-time floor plan visualization with:

- Color-coded table states (green/orange/red)
- Timer rings for occupied tables (60-min expected duration)
- Server assignment badges with waiter initials
- Undo history (last 10 actions)
- Polling at 2–5s intervals with Server-Sent Events for streaming updates

7 API Reference

Method	Endpoint	Purpose
ML & Video		
POST	`/api/v1/videos/upload`	Upload video (max 100MB)
POST	`/api/v1/videos/{id}/process`	Start classification
GET	`/api/v1/videos/{id}/results`	Retrieve classifications
Routing		
POST	`/routing/recommend`	Get table/waiter recommendation
POST	`/routing/seat`	Execute seating (creates Visit)
Scheduling		
POST	`/schedules/run`	Trigger AI scheduling
GET	`/schedules`	List schedules with items

8 Review Scraping & Sentiment Analysis

To provide actionable customer insights, we built a multi-platform review aggregation system with sentiment analysis.

8.1 Multi-Platform Scraper

The scraper collects reviews from Google Reviews and Yelp, extracting:

- Review text and star ratings
- Reviewer metadata (date, username)
- Restaurant identifiers for cross-platform linking

Architecture: The backend exposes scraping endpoints via FastAPI, while the React frontend provides a dashboard for viewing aggregated reviews and sentiment trends.

8.2 Sentiment Analysis with ByteDance Model

Reviews are processed through ByteDance's sentiment classification model to extract:

- Overall sentiment polarity (positive/neutral/negative)
- Aspect-level sentiment (food, service, ambiance, value)
- Trend analysis over time periods

This enables restaurant managers to identify specific operational areas needing improvement based on customer feedback patterns.

9 Data Annotation Pipeline

Training accurate ML models requires high-quality labeled data. We developed custom annotation tooling to accelerate the labeling process.

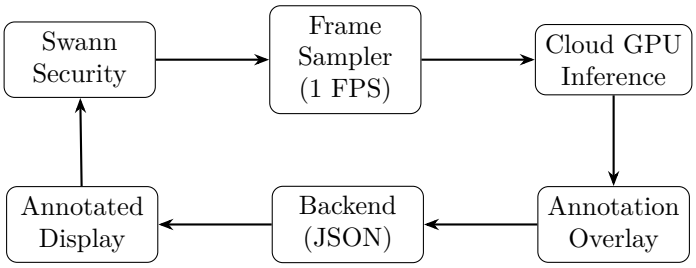
9.1 Table State Annotation Tool

For labeling table states (clean/occupied/dirty), we built a custom annotation interface that outputs labels in JSON format:

```
{
  "frame_id": "session_001_00042",
  "table_id": "T5",
  "label": "occupied",
  "annotator": "alex",
  "timestamp": "2024-03-15T14:32:00Z"
}
```

9.2 Live Video Integration with Swann Security

For real-time annotation and inference on live camera feeds, we integrated directly with Swann security software:



Pipeline:

1. Swann video feeds are sampled at 1 frame per second
2. Frames are sent to cloud GPU (RunPod) for ML inference
3. Predictions are overlaid back onto the Swann display
4. JSON annotations are persisted to the backend for training data collection

This closed-loop system enables both real-time monitoring and continuous collection of labeled training data from production environments.

Conclusion

This platform addresses core restaurant operational challenges through five integrated systems: ML-powered table classification, fairness-optimized waiter routing, constraint-aware scheduling, review sentiment analysis, and streamlined data annotation. The combination of computer vision, NLP, and optimization algorithms creates a comprehensive solution that improves both operational efficiency and staff satisfaction.