Executive Summary — Mealyze

One-sentence Value Proposition
Mealyze transforms the everyday question “What should I eat?” into a low-friction, explainable meal recommendation experience by incorporating user context (goals, time, condition, preferences, and dislikes), and guides post-meal behavior through a non-medical, heuristic visualization of energy and blood-glucose responses over the next four hours.

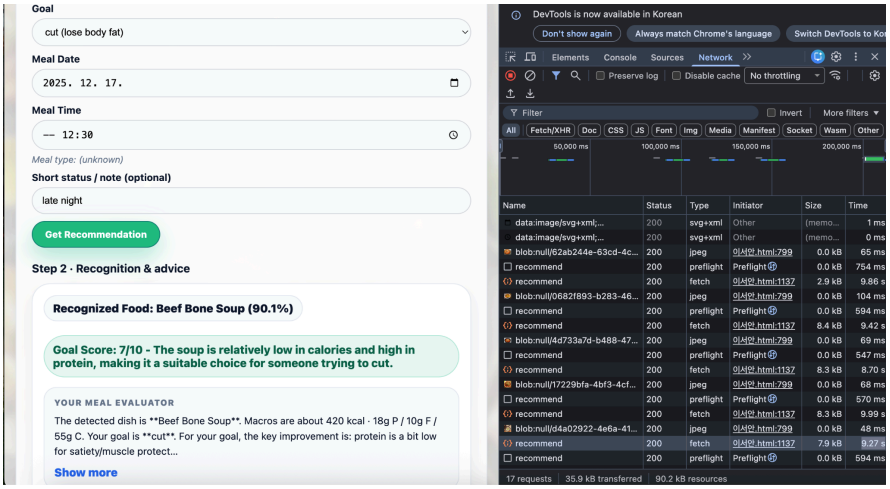
Problem — Why This Matters
Modern eating challenges are not simply about counting calories, but about the accumulation of repeated decision-making failures in daily life.
1) Decision Fatigue
Users face meal decisions multiple times a day. This repetitive process increases cognitive load and often leads to defaulting to familiar high-calorie foods or impulsive choices followed by regret.
2) Input Friction
Many existing tracking-first diet applications require users to search for foods and manually input quantities. While feasible initially, this process introduces high friction and results in poor long-term engagement and early user drop-off.
3) Lack of Temporal Guidance
Although users are interested in post-meal effects such as drowsiness, energy dips, and so-called “blood sugar spikes,” most services fail to present these effects as changes over the next few hours. As a result, users struggle to translate dietary information into concrete actions, such as deciding when to rest or exercise.

Solution — How Mealyze Addresses the Problem (Three-Module Architecture)
Mealyze addresses these challenges through three integrated modules.
Module 1. Low-friction, Context-driven Recommendation
Users can receive recommendations through either food photo uploads or simple text input.
Meal goals (cut / maintain / bulk), meal time, and short contextual notes (e.g., “late-night snack”) are combined to reflect the situational context of eating. If food recognition fails or is ambiguous, users can use Manual Override to directly input the correct food name and immediately re-run the recommendation, ensuring that the interaction flow is not interrupted. Because allergy and dislike information is entered as free text, linguistic variability may occur. To mitigate this, Mealyze applies synonym-based keyword matching, conservative warning messages, and a safety-first policy, while advising users with severe allergies to verify ingredients independently.

Module 2. Explainable Recommendations via a Tagged Database
Recommendations are generated from a tag-based food database containing per-serving nutritional values, taste and category labels, situational tags, and allergy-related tags.
Each result screen presents WHY (reasoning), SUMMARY, and NUTRITION, allowing users to understand why a particular recommendation was produced. This design emphasizes explainable AI, prioritizing user understanding over opaque model outputs.

Module 3. Behavioral Coaching and Long-term Insights
A key differentiator of Mealyze is the visualization of energy and approximate blood-glucose curves over the four hours following a meal. The curves include annotated events such as digestion onset, potential drowsiness windows, and recommended times for light physical activity, enabling users to intuitively choose their next action. These curves are non-medical, heuristic estimations derived from macronutrient composition (P/F/C) and food-type tags, and are explicitly presented as guidance rather than medical diagnosis. Additionally, the History & Insights feature provides 7-day, 30-day, and custom-range summaries of meals, including average calories and macronutrient distributions, extending short-term recommendations into long-term habit awareness.

KPI Definition and Evaluation — Did We Hit Our KPIs?
Project KPIs are divided into implementation KPIs (achieved) and user outcome KPIs (measured in this project).
1) Implementation KPIs — Achieved



KPI 1. Friction Reduction
- Target: ≤ 2 core steps to obtain a recommendation; ≤ 20 seconds in quick mode
- Evidence: Immediate recommendations via text input; Manual Override available when recognition fails
- Status: Achieved

KPI 2. Explainability
- Target: All recommendations include WHY / SUMMARY / NUTRITION
- Evidence: Structured explanation sections present on all result screens
- Status: Achieved

KPI 3. Retention Hooks
- Target: 7-day, 30-day, and custom-range analysis with nutritional summaries
- Evidence: History & Insights UI supports period-based analysis
- Status: Achieved

---

2) User Outcome KPIs — Measured

KPI 4. Food Recognition Accuracy
- Target: Held-out test set Top-1 accuracy ≥ 85% (rubric-aligned)
- Result (KPI evaluation): Held-out Test Set (n=246) Top-1 accuracy = 85.37%
- Additional (demo smoke test): 95.0% (19/20 images) for showcase stability
- Status: Achieved

KPI 5. Latency and Robustness
- Target: Median (p50) end-to-end response time ≤ 10 seconds
- Result: p50 = 9.21 seconds
- Additional Evidence: Human-readable error messages displayed within 2 seconds during API failures, with Manual Override provided as an alternative path
- Status: Achieved

KPI 6. Usability (Task Success Rate)
- Target: Task success rate ≥ 80%
- Result: 100% (10 out of 10 participants successfully completed the task)
- Status: Achieved

KPI 7. Usefulness (Mini User Survey)
- Target: Mean usefulness score ≥ 4.2 / 5 (5-point Likert scale)
- Result: 4.9 / 5 (n=10)
- Additional Evidence: Per-question averages — Overall usefulness 4.9, Context-awareness 4.9, Explainability 4.8, 4-hour curve usefulness 5.0, Lightweight vs tracking apps 4.7
- Status: Achieved

Conclusion — Why Mealyze Matters

Mealyze intervenes at the exact moment of meal decision-making, reducing cognitive load and input friction while combining explainable recommendations, four-hour post-meal response visualization, and history-based insights. By doing so, it moves beyond one-off menu suggestions and supports realistic, sustainable eating habits, transforming everyday food choices into fast, understandable, and action-oriented decisions.

User Persona & Problem Definition

Modern meal selection is no longer simply about “choosing a menu item.” It is a high-frequency decision-making problem (1–3 times per day) where goals (cut / maintain / bulk), physical condition, mood, time constraints, allergies, and exercise routines are simultaneously intertwined. However, at the exact moment this decision must be made, fast and explainable guidance is largely missing.

1) Problem Context: Why does this problem occur?

Meal choices today have expanded beyond “eating something tasty” to include self-management, condition regulation, and routine maintenance. The everyday use of phrases like “What should I eat for lunch/dinner?” reflects how food choice has become a repeated social concern. At the same time, health trends are shifting beyond calorie-centric thinking toward post-meal responses, such as food-induced drowsiness, energy drops, and blood sugar spikes. Users who exercise regularly consider not only what they eat, but also when they should work out or focus after eating.

However, real meal decision moments are typically accompanied by the following constraints:
- Time pressure: Decisions are made right before classes, meetings, or deadlines.
- Increased cognitive load: Choices must be made while tired or stressed.
- Simultaneous constraints: Goals (cut/maintain/bulk) + condition (stomach discomfort, hangover) + time (late night) + allergies.
- Information–action gap: Even if users know kcal / P / F / C, it does not immediately answer “Should I eat this right now?”
- Drop-off from tracking apps: Long-term logging has value, but high input friction prevents sustained use.

In short, the core problem is not a lack of information, but the absence of an intelligent guide that reflects context and supports fast decisions at the moment of choice.

---

2) Target Users

While Mealyze is designed for all age groups, its **initial core target** is users aged **18–34** with high self-management needs and strong digital familiarity. This group:

1. Experiences high decision stress,
2. Has diverse goals,
3. Is sensitive to trends around condition, blood sugar, and post-meal responses,
4. Has low barriers to photo- and text-based input.

Primary Persona (P1): "Repeated Menu Dilemma + Goal-Oriented User"

- **Age / Role:** 18–34 / students, office workers, project team members
- **Meal context:** Mixed environments (cafeteria, delivery, convenience stores, dining out; irregular schedules)
- **Goal:** cut / maintain / bulk (or general health)
- **Preference:** Avoids long-term logging; wants **instant recommendations with clear reasoning**

Moments of Truth

- "Only 10 minutes for lunch → need to decide fast"
- "Team project / work stress → need comfort food (soul_food)"
- "Stomach discomfort → need gentle food (stomach_sensitive)"
- "After drinking → need hangover-friendly food"
- "Hungry at 11 PM → know late-night risk but still need alternatives"

Segment Persona (P2): "Post-Meal Response + Exercise Routine User"

Shares P1 characteristics but places greater emphasis on **post-meal drowsiness, blood sugar fluctuation, and workout timing**.

- Maintains an exercise routine 2–5 times per week (gym, running, swimming, etc.)

Expected value

- "I want to roughly know how I'll feel a few hours after eating."
- "Guidance on when it's safe to exercise would be helpful."

Safety Persona (P3): "Safety-First User (Allergies / Avoidance)"

- Sensitive to specific ingredients (egg, milk, wheat, peanut, seafood, cucumber, etc.)
- Prioritizes **risk exclusion and warnings over variety**

Expected value:
Safe filtering matters more than diverse recommendations.

---

3) Pain Points

These pain points do not merely cause inconvenience; they directly shape user behavior (monotony, regret, churn).

Decision Fatigue (Repeated Choice Exhaustion)

"Where should I eat?" repeats 1–3 times daily.

Result: Defaulting to familiar foods (monotony) or impulsive choices (regret)

Difficulty Translating Goals into Menu Choices

Users have goals but lack immediate, actionable criteria.

Result: Repeated choices disconnected from goals → reduced self-efficacy

Recommendations That Ignore Context Feel Unrealistic

- Stress → comfort food
- Hangover → recovery food
- Stomach discomfort → gentle food
- Late night → caution required

Result: "The app doesn't understand my situation" → loss of trust

Interest in Post-Meal Response, but No Time-Based Feedback

Nutrition tables exist, but users cannot easily see *how their body might respond after eating*.

Result: Continued reliance on personal experience or unreliable information

High Input Friction in Tracking Apps

Initial motivation fades as logging becomes burdensome.

Result: No accumulated data → loss of service value → churn

---

4) Why This Service Is Critical: Mealyze's Strategy

Mealyze does not aim for "perfect nutrition tracking." Instead, it focuses on reducing decision friction (**quick**), explaining recommendations (**explainable**), tolerating real-world errors (**robust**), and offering time-based feedback (**time-aware**).

A) Low-Friction Decision Support: Quick Mode (Text-Based)

Users input their current state (mood, condition, goal, allergies) via text.
A tag-based DB narrows candidates and suggests top options.

Effect: Reduces decision time and enables immediate action

B) Explainable Recommendations: Tag-Based Food DB

The core principle is **not** having the model invent nutrition, but **fixing evidence through structured DB lookup**.

- **Nutrition:** kcal / protein / fat / carbs (per serving)
- **Context:** soul_food, stomach_sensitive, hangover_friendly, late_night
- **Goals:** cut_friendly, bulk_friendly, high_protein, low_carb, low_fat
- **Safety:** contains_* (allergens)

Result: Clear explanations increase trust and transparency

C) Robust in the Real World: Photo Input + Manual Override

Assuming image recognition is imperfect, users can manually correct the food name and rerun recommendations.

Effect: Errors do not break the service; users retain control
(Reduces churn and improves real-world usability)

D) Time-Based Feedback Loop: 4-Hour Curve + History / Insights

Lightweight heuristic logic visualizes post-meal response over 4 hours as a **lifestyle guide (non-medical)**.

Events:
Post-meal → drowsiness risk → exercise window → recovery phase

History / Insights (7-day / 30-day / custom) provide pattern-based feedback:

- Average kcal / PFC

- Late-night eating ratio
- Protein deficiency trends

Effect: Single-meal recommendations scale into long-term habit value

Safety Backbone: Allergy Safety Policy

- **Hard Filter:** Foods with potential allergen inclusion are excluded
- **Soft Warning:** Recipe variation / cross-contamination notices
- **Conservative Labeling:** Any possible inclusion is marked as true

(Meets P3's core requirements)

---

5) Alternatives / Competitor Landscape

- **Tracking-first apps:** Precise but high input friction → churn
- **Recipe / content platforms:** Rich content but limited goal, condition, and allergy awareness
- **Delivery / menu platforms:** Abundant choices, weak health and post-meal perspective

→ Mealyze fills the gap by combining **context-aware decisions, explainable recommendations, and time-based feedback**.

---

6) Success Metrics (KPIs)

KPIs are defined not by feature existence, but by **quantitative system metrics** and **usability indicators**.

- **Accuracy:** Held-out test Top-1 accuracy ≥ 85%
- **Latency & Robustness:** Median (p50) end-to-end response time ≤ 10 seconds, with graceful degradation through human-readable error messages and Manual Override during failures
- **Task Success Rate(Usability):** End-to-end task success rate (upload → recognition → nutrition lookup → recommendation) ≥ **80%** without critical errors
- **Usefulness (small user test):** Mean usefulness score **≥ 4.2 / 5** on a 5-point Likert scale

Data Journey & Ethics

Mealyze adopts a **hybrid data pipeline** consisting of
(1) a food image classification model trained on **publicly licensed image datasets**,
(2) a **structured Food Database (300 foods)** based on nutrition values and service tags, and
(3) an **open-license Real-World Test Set (RWTS)** used **exclusively for evaluation**.

All data are **strictly separated by purpose** (training / evaluation / service knowledge base). Ethical risks—such as **copyright compliance, personally identifiable information (PII), allergy safety, and potential health-related misinterpretation (medical claims)**—are identified in advance and mitigated through explicit policies, including **license verification, PII screening, hard/soft safety rules, and non-medical-use disclaimers**.

Detailed license evidence, URL logs, and model bias analyses (e.g., vulnerability to low-light or extreme-angle images) are provided separately in the **Data Sheet / Evidence Appendix**.

1) Where did the data come from?

1.1 Public Dataset (Training): AI-Hub Korean Food Images

The food image classification model was trained using the **AI-Hub Korean Food Image Dataset** provided by the **National Information Society Agency (NIA), Republic of Korea** (dataSetSn=79).

All usage complied with the dataset documentation and terms of use, which permit **educational and research purposes**. **No redistribution or secondary sharing of the original images** was performed.

Source: AI-Hub (dataSetSn=79)
https://aihub.or.kr/aihubdata/data/view.do?dataSetSn=79

Classes: 31 foods
Images per class: 999
Total images: 31 × 999 = 30,969

Example classes (31):
Beef Bone Soup, Doenjang Stew, Fried Chicken, Gimbap, Jjajangmyeon, Jjamppong, Kimchi Fried Rice, Samgyeopsal, Tteokbokki, Yakgwa, Yukhoe, etc.

Class selection rationale (service-oriented)

Rather than aiming for universal food recognition, the initial model prioritizes **high real-world utility** in early service deployment. The 31 classes were selected based on:

1. **High consumption frequency** among Korean users (delivery, dining out, campus meals, home cooking),
2. Inclusion of **diverse cooking styles and visual patterns** (soups, noodles, rice dishes, fried foods, fermented foods, desserts),
3. A **core menu set** suitable for rapid performance validation in an early-stage service.

1.2 Self-curated Dataset (Evaluation only): RWTS (Real-World Test Set)

To assess whether the model functions beyond studio-quality images, the team constructed a **Real-World Test Set (RWTS)** that is **fully separated from training data**. RWTS images were **never used for training** (to prevent data leakage) and were operated at a small scale for **qualitative robustness checks**, not for formal quantitative benchmarking.

- **Purpose:** Evaluation-only (robustness check)
- **Training usage:** ❌ Not used for training
- **Scale:** ~6 images per class (small, qualitative)

When collecting RWTS, **search engines are not treated as data sources**. Images are collected only from **original pages or repositories with explicitly verifiable licenses**. Accepted licenses are limited to **CC0, Public Domain, or explicitly permitted Creative Commons licenses**. Images with unclear or unverifiable licenses are excluded.

Compared to AI-Hub images, RWTS intentionally reflects **real-world usage conditions**, including:

- Natural light, fluorescent light, and low-light environments
- Cluttered backgrounds (trays, tables)
- Varied camera angles (side views, close-ups, long shots)
- Visual noise (chopsticks, hands, cups, etc.)

1.3 Structured Food Database (Service knowledge base): Food DB (300 foods)

For recommendation logic, explanation, and safety filtering, the team constructed a **structured Food Database** covering **300 foods**, including nutrition values (kcal / protein / fat / carbs) and multi-dimensional tags.

This database serves as both:

- a **hallucination-prevention mechanism**, preventing the model from inventing nutrition facts, and
- the **foundation for explainable recommendations**.
- **Scale:** 300 foods
- **Coverage:** Korean and global cuisines (Japanese, Chinese, Southeast Asian, Indian, Middle Eastern, Western, Mexican, Desserts, Drinks)

- **Design intent:** English-based UI, considering future global expansion

Nutrition value standardization (required statement)

All nutrition values are standardized **per one serving**, as specified in serving_desc (e.g., 1 bowl, 1 plate).
At the prototype stage, the priority is **consistency, comparability, and traceability**, rather than clinical precision. Values are derived from one of the following sources:

1. Public or national nutrition databases,
2. Manufacturer or brand nutrition labels,
3. For foods with high recipe variability, averaged representative values from multiple recipes.

Tag labeling summary (linked to service logic)

Tags are structured to enable **transparent recommendation explanations** and include:

- **Context:** soul_food, stomach_sensitive, hangover_friendly, late_night
- **Goal/Nutrition:** cut_friendly, bulk_friendly, high_protein, low_carb, low_fat
- **Taste/Type:** spicy, oily, deep_fried, soup_or_stew, noodle, rice_based
- **Allergy/Safety:** contains_egg, contains_milk, contains_wheat, contains_peanut, contains_seafood, contains_cucumber, etc.

2) Data governance: PII screening & storage/sharing principles

Images used for training or evaluation are screened to exclude or crop any **personally identifiable information (PII)**, including faces, names, phone numbers, receipts, or addresses.

All data are stored in **internal team repositories only**, with access restricted to team members. Project submissions focus on **results and analyses**, without external redistribution of original datasets.

3) Safety & Ethics policies (two core policies)

3.1 Allergy Safety Policy (Hard filter vs. Soft warning)

- **Hard filter (mandatory exclusion):**
  If a user selects or inputs an allergy or avoided ingredient, foods with contains_* = 1 are removed from recommendation candidates.
- **Soft warning (caution notice):**
  For cases involving recipe variation or potential cross-contamination, caution messages such as *"May vary by restaurant or recipe"* are displayed.
- Conservative labeling: If there is any reasonable possibility of inclusion, the ingredient is labeled as 1, prioritizing safety.

RWTS-mini: License & PII Screening (Condensed Evidence)
RWTS-mini was used only for qualitative robustness checks and Error Board examples (not for KPI scoring). To reduce reporting overhead while keeping evidence transparent, we provide a condensed License & PII Screening log for the exact RWTS-mini samples shown in the Error Board. Each entry includes the source domain, a license-evidence screenshot ID, and a PII screening status (pass/cropped/excluded).

3.2 Health-claim Ethics (4-hour curve)

The **4-hour energy/blood-sugar curve** is a **non-medical, lifestyle-coaching visualization** based on nutritional composition and food-type tags.
It is **not intended for diagnosis or treatment**. Users with medical conditions (e.g., diabetes) are explicitly advised to consult healthcare professionals.

Model Training & Evaluation (Excluding Error Board)

1) Model Goal and Role

In this project, the image classification model predicts one of **31 food classes** from a meal photo. Its output (food label + confidence) becomes the first input to the downstream pipeline: **(1) nutrition table lookup** and **(2) goal-aware recommendation logic** (cut / maintain / bulk). In the deployed prototype, the front-end sends an image to the **/api/recommend** endpoint to request inference.

In addition, the service uses a multimodal generative model (**LLaVA**) for **inference-only** tasks such as explanation/wording. However, **LLaVA was not trained on our project data**, so this "training/evaluation" section focuses strictly on the **MobileNetV2-based image classifier** that we trained and evaluated.

---

2) Dataset Splits (Train / Validation / Held-out) and Structure

To ensure fair evaluation and avoid data leakage, we maintained a held-out test set that is never used during training or validation. We separately report performance on Train, Validation, and Held-out (final generalization).

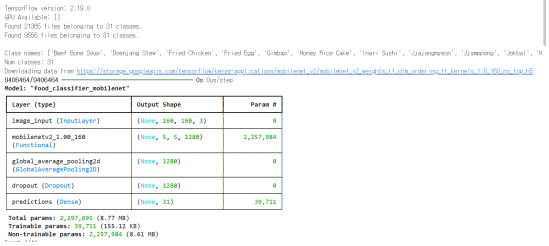
- Number of classes: 31
- Train: 21,365 images (31 classes)
- Validation: 9,556 images (31 classes)
- Held-out Test Set (quantitative): 246 images (self-curated; never used in train/val)

We managed the dataset using a directory-based loading structure in Google Drive (folder name = label) : .../food_dataset/train, .../food_dataset/val, .../food_dataset/rwts

Note on naming consistency:

- RWTS-mini (qualitative): small real-world set used for qualitative robustness checks (and Error Board examples).
- Held-out Test Set (quantitative, n=246): the official KPI-scoring test set used for final accuracy/loss reporting.

(Optional one-line hygiene note, if you want to lock the "30,969 vs 30,921" question inside this section too: "After hygiene filtering (corrupted/duplicate/PII-risk), a small number of images were removed from the raw training corpus, so the final usable train/val count differs slightly from the dataset's raw total.")

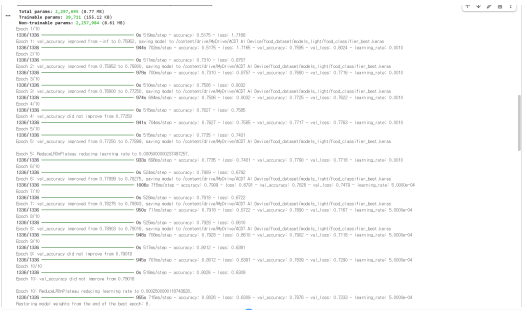


TensorFlow version: 2.19.0
GPU Available: []
Found 21365 files belonging to 31 classes.
Found 9556 files belonging to 31 classes.

Class names: ['Beef Bone Soup', 'Doenjang Stew', 'Fried Chicken', 'Fried Egg', 'Gimbap', 'Honey Rice Cake', 'Imari Sushi', 'Jjajangmyeon', 'Jjamppong', 'Jokbal', 'K
Num classes: 31
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/mobilenet_v2/mobilenet_v2_weights_tf_dim_ordering_tf_kernels_1.0_160_no_top.h5
9406464/9406464 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
Model: "food_classifier_mobilenet"

Layer (type)	Output Shape	Param #
image_input (InputLayer)	(None, 160, 160, 3)	0
mobilenetv2_1.00_160 (Functional)	(None, 5, 5, 1280)	2,257,984
global_average_pooling2d (GlobalAveragePooling2D)	(None, 1280)	0
dropout (Dropout)	(None, 1280)	0
predictions (Dense)	(None, 31)	39,711

Total params: 2,297,695 (8.77 MB)
Trainable params: 39,711 (155.12 KB)
Non-trainable params: 2,257,984 (8.61 MB)



---

3) Preprocessing & Data Augmentation (Generalization)

- Input size standardization: All images are resized to 160×160.
- Normalization / preprocessing: We applied the MobileNetV2 preprocessing function to match the backbone's expected input scale.
- Data augmentation (Train only): to mimic real usage conditions (lighting, angle, framing variability), we applied augmentation only to Train:
  - RandomFlip("horizontal")
  - RandomRotation(0.05)
  - RandomZoom(0.1)
  - RandomContrast(0.1)

Validation and Held-out were evaluated without augmentation to measure "as-is" performance.

---

4) Model Choice, Architecture, and Design Rationale

Because the model must run inside a web service with fast response time, we chose a lightweight, well-validated transfer-learning backbone rather than a heavy CNN.

- Backbone: MobileNetV2 (ImageNet pretrained) Semantic Scholar
- Training strategy: Stage 1 uses a frozen backbone as a feature extractor for stable transfer learning
- Classifier head: GlobalAveragePooling2D → Dropout(0.3) → Dense(31, softmax)
- Artifacts saved for deployment consistency:
  - Best checkpoint: food_classifier_best.keras (by best val_accuracy)
  - Final model: food_classifier_final.keras

- Label mapping: label_map.json (ensures consistent index↔class mapping during inference)

This design supports: (1) stable training, (2) fast inference for deployment, and (3) strong performance within limited development time.

4.1) (Literature-based) Why MobileNetV2: Backbone Comparison Table

Table A. Candidate CNN backbones (literature-based, ImageNet reference) Semantic Scholar+1

Model	Top-1 Acc.	Params	FLOPs / MAdds	One-line interpretation (service context)
MobileNetV2	~72.0%	~3.4M	~300M MAdds	Lightweight/fast → fits "quick response" web inference
ResNet-50	~76.0%	~26M	~4.1B FLOPs	Strong accuracy but heavier → higher latency/cost risk
EfficientNet-B0	~77.1%	~5.3M	~0.39B FLOPs	Good accuracy-efficiency balance, but can be heavier than MobileNet

Model choice rationale (literature-based): Given the project constraint of fast inference in a web service and limited training/development time, we prioritized a backbone known for low computation and reliable transfer learning. We trained and validated MobileNetV2 in this project; we did not run a full experimental benchmark across alternative backbones, so this table is included to transparently describe the trade-off using established references.

4.2) (Literature-based) Why a Hybrid Setup: Classifier + LLaVA (Design Trade-off)

This project uses a hybrid structure:

- Classifier (MobileNetV2): provides a stable food label + confidence
- Food DB: grounds nutrition facts in structured values (not free-text generation)
- LLaVA (inference-only): generates user-friendly explanations/wording when needed arXiv

Table B. Multimodal explanation module options (literature-based)

Option	**Strength**	**Weakness / Risk**	**Role in our service**
LLaVA	Strong at multimodal instruction-following + conversational outputs	Heavy; compute/latency burden possible	Used for explanation text (inference-only)
BLIP-2	Efficient VLP framework using frozen encoders + LLM	Chat-style UX needs extra design work	Alternative candidate for future comparison
Rule-based templates	Fastest + most controllable (low hallucination)	Limited expressiveness/adaptation	Strong fallback mechanism

Table C. Single-model vs hybrid structure (literature-based design comparison) arXiv+1

Approach	**What it does**	**Pros**	**Limitations / Risks**	**Conclusion for our service**
(A) Classifier only	photo → 31-class label	Fast + structured output → clean DB matching	Weak at natural-language explanation; poor handling of ambiguous/multi-food scenes	Best for "correct label," but UX becomes rigid without a text module
(B) LLaVA only	photo + prompt → label + explanation + recommendation text	Natural conversational explanations	Known risk of hallucination (inventing details not in image), which is risky when correctness matters for nutrition/labeling	Too risky to own "ground-truth" fields alone
(C) Hybrid (Classifier + DB + LLaVA)	classifier fixes label + DB fixes nutrition facts → LLaVA explains grounded facts	Reduces hallucination risk by grounding key facts, while improving UX via explanations	More integration complexity; needs prompt/output control	Most practical trade-off for accuracy + explainability

Why we used the hybrid setup (classifier + LLaVA):

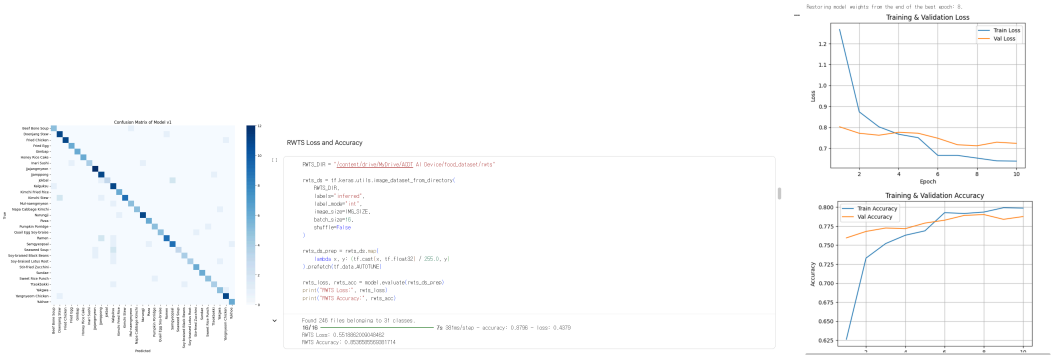
Our service must (1) lock the food label to match a nutrition database consistently and (2) provide readable explanations. A lightweight classifier gives fast, structured outputs; meanwhile, VLM-style systems are strong at explanation but have a

recognized hallucination risk, so we avoid letting them be the single source of truth for label/nutrition. Therefore, we separated responsibilities: classifier for label + DB for facts, LLaVA for explanation. arXiv+1
Note: We directly trained/evaluated only the MobileNetV2 classifier; we did not run a quantitative benchmark of "LLaVA-only vs hybrid," and present the hybrid justification as a literature-based design trade-off.

---

5) Training Configuration

- Framework: TensorFlow / Keras (Colab)
- Batch size: 16
- Epochs: 10
- Optimizer: Adam (learning rate = 1e-3)
- Loss: Sparse Categorical Crossentropy
- Metric: Accuracy
- Callbacks (overfitting control + best-weight capture):
    - ModelCheckpoint (save best by val_accuracy)
    - EarlyStopping (val_accuracy, patience=3, restore best weights)
    - ReduceLROnPlateau (monitor val_loss, factor=0.2, patience=2, min_lr=1e-6)



---

6) Metrics and KPI Definition

Following the rubric guidance, we define KPIs with explicit numeric success criteria (Accuracy / Latency) aligned with service requirements.

- KPI-1 (Held-out Accuracy): Held-out Test Set Top-1 Accuracy ≥ 85%
    - Rationale: if the first food label is wrong, the entire nutrition/recommendation output becomes unreliable.
- KPI-2 (Model Latency): classifier inference time ≤ 1 second / image
    - Rationale: slow response after image upload increases user drop-off.
- Supporting metric: Loss is recorded alongside accuracy to confirm training stability.

(End-to-end latency for the full service request—upload → response—should be reported separately in the Service/Deployment section, because it includes ngrok/network + server routing + extra logic.)

---

7) Performance Results (Train / Validation / Held-out)

(A) Training / Validation (learning process)

- Final epoch: Train Accuracy = 0.8026, Train Loss = 0.6309
- Best checkpoint: Best Validation Accuracy = 0.7902

Validation classification report (to reflect class imbalance, reporting both macro and weighted averages):

- macro avg: precision 0.79 / recall 0.78 / f1-score 0.78 (support=9556)
- weighted avg: precision 0.82 / recall 0.79 / f1-score 0.80 (support=9556)

Interpretation: weighted averages exceeding macro averages suggest performance is more stable on higher-support classes, while some lower-support/difficult classes may have lower recall; therefore final evaluation must be confirmed on the held-out test set.

(B) Held-out Test Set (quantitative, n=246; never seen during training)

- Held-out Loss = 0.5519
- Held-out Accuracy = 0.8537 (≈ 85.37%)
- Runtime log during evaluation: 16 steps, total 7 seconds, ~381 ms/step (batch=16)

KPI achievement:

- KPI-1 (Accuracy ≥ 85%): Achieved (85.37%)
- KPI-2 (Latency ≤ 1s/image): Achieved (model-only estimate)
    - From the batch evaluation log: 381 ms/step ÷ 16 images ≈ 23.8 ms/image

- This estimate reflects model-only / input-pipeline latency, excluding network, routing, and UI rendering.

Consistency note (why held-out can exceed validation):
Although the held-out set includes real-world variability, the license/quality filtering process can increase the proportion of clear, well-framed images; therefore average held-out accuracy may exceed validation. Truly challenging low-light/angle cases are documented separately in the Error Board.

8) Failure Models(RWTS-mini)&What we Changed
In addition to quantitative held-out testing, we summarize key failure modes observed in real-world-like RWTS-mini samples (e.g., low light, angled shots, and multiple dishes). These cases directly motivated our reliability loop: Manual Override allows users to correct the label and re-run analysis, preventing model errors from breaking the recommendation flow."

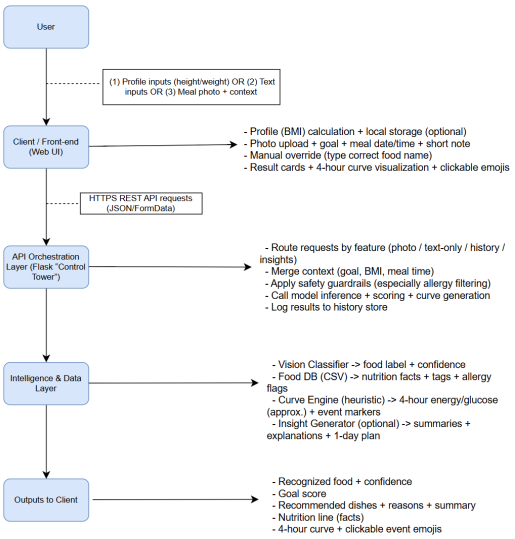
---

9) Section Conclusion
We trained a MobileNetV2-based transfer-learning classifier for 31 food classes and achieved 85.37% Top-1 accuracy on a strictly held-out test set (n=246) that was never used in training/validation. The classifier also showed fast inference based on batch evaluation logs (≈ 24 ms/image model-only estimate), meeting practical speed requirements for web service integration. Overall, this model reliably provides the pipeline's core input: photo → label/confidence → nutrition + recommendation outputs, while explanation text is handled separately by an inference-only multimodal component under a hybrid design.

Error Board
During the development, we analyzed a total of 10 errors. In this report, we focus on explaining the two most critical errors.

1. Food Recognition Failure
   - Problem: Despite successful image uploads and model calls, the system failed to recognize food or returned empty results.
   - Root Cause: Pipeline inconsistency in the web-Colab connection. Image preprocessing differed from training format, and output logic had parsing issues when returning results to the frontend.
   - Solution: Debugged the entire pipeline by verifying image uploads through logs, aligning preprocessing with training settings, fixing label mapping between model outputs and food names, and specifying output formats to prevent empty returns.
2. Recommendation Logic Error
   - Problem: Users selected specific dietary goals (bulk, cut, or maintain) but received recommendations for all goals simultaneously instead of tailored advice.
   - Root Cause: Goal filtering logic was not properly applied in the prompt or post-processing step.
   - Solution: Modified backend logic to generate goal-specific sentences based on user selection and implemented frontend filtering to display only relevant recommendations matching the selected goal.

Service Architecture
1. Purpose and what the system does
Mealyze provides a lightweight meal check experience. It supports photo-based food recognition, database-grounded nutrition information, a 4-hour estimated energy and glucose curve with clickable event emojis, and meal history features for short- and longer-range review.



2. Architecture overview
The system follows a clear left-to-right pipeline from User to Input to Model to Output. Safety and policy considerations are built into the overall flow and are intended to apply consistently across features.

3. Key components and responsibilities
Client and Front-end: A single web interface where users can enter basic profile context, run photo analysis, or use text-based input. Results are presented in readable cards with a curve visualization, and users can correct the food name and re-run when needed. API orchestration layer: Routes requests, merges user context such as goal and meal time, applies safety checks where applicable, calls inference and scoring, and records

outputs for later viewing. Intelligence and data layer: A vision model predicts a food label and confidence. Nutrition facts and tags are retrieved from a structured food database to keep outputs grounded. A heuristic curve engine generates the 4-hour preview. Optional insight generation may summarize patterns from past logs. Storage: Saves minimal interaction records and can keep profile context locally to reduce stored personal data.

4. End-to-end data flow
Users can set profile context, analyze a meal photo with context, optionally correct results through manual override, use a text-based mode, and review saved history. A single dashboard can display logs and run summary views depending on the selected action.

5. Policy, safety, deployment, and robustness
Mealyze uses conservative, non-medical wording. Allergy-related handling is designed to prioritize filtering and warnings. The backend may run in an on-demand mode due to compute constraints; when unavailable, the UI is designed to show a clear unavailable state rather than failing silently. The user flow distinguishes between service unavailability and recognition inaccuracy by offering alternative paths such as text-based use or manual correction.