

I want you to act as a prompt engineer. I will give you some detail on the task, the AI you are designing the prompt for will have to perform. I want you to make sure, the designed prompt covers all important aspects the model will need to provide an accurate answer that is within the bounds specified by my task description. In case there is still some confusion or imprecision, and you want me to add more details for you to adapt the prompt, you will let me know and we will continue until you tell me it is sufficiently good.

Google Gemini 3 Flash Preview

I accept the role. As a prompt engineer, I will focus on optimizing for clarity, persona, constraints, output formatting, and "Chain-of-Thought" reasoning to ensure the AI performs optimally.

To get started, please provide the details of the task. Specifically, try to include:

1. **The Goal:** What is the final objective?
2. **The Context:** Who is the audience? What is the tone?
3. **The Constraints:** Are there word counts, formatting rules (JSON, Markdown, etc.), or things the AI must *not* do?
4. **Examples (Optional):** Do you have any specific examples of the desired output?

Once you provide the initial details, I will either draft the prompt or ask follow-up questions to bridge any gaps. **Please provide the task details.**



1. **Objective:** The main goal of the task is to estimate the calories as well as the nutritional breakdown of a meal in terms of carbs, proteins, and fats. The model is expected to either a) process an image the user has taken of his meal AND a voice

note recorded to further describe the meal or its preparation to give some further context or b) process an image the user has taken of his meal AND a text provided along with the meal to further describe the meal or its preparation or c) only process an image the user has taken of his meal, although a) is the typical use-case for this application

2. Context or Domain The task is situated in the context of calorie tracking so it is a little health related, however, it is not really focused on the medical/ health part, since it should just be a convenient, easy-to-use application for any user that likes to track calories, so we don't really care about that

3. Input The model will receive either a) image and audio input or b) image and text input or c) only image input. The text input is a transcription of a voice note from the user or a text entered by the user, using the app to log his meals

4. Output The model is expected to output *only* a JSON object matching the following exact schema that gives a detailed and accurate nutritional breakdown of calories and nutrients as well as a title for the meal, I give you an example for that schema

```
"success": true,
"requestId": "img_analysis_...",
"items": [
{
"name": "GrilledChickenBreast",
"confidence": 0.9,
"serving_size_grams": 150,
"nutrition": {
"calories": 248,
"protein_g": 46.5,
"fat_g": 5.4,
"carbohydrates_g": 0,
"meal_quality": 9,
"goal_fit_percent": 0.9,
"calorie_density_cal_per_gram": 1.65
}
},
],
"errorMessage": null
}
```

5. Rules The model should adhere to the following rules, with respect to the provided schema - **success** : Set to **true** if a meal could be recognized **false** otherwise

- **requestId** : Generate a unique ID for this analysis
- **items** : Create one object for *each* distinct food item in the image

- **confidence** : Your confidence(0.0 to 1.0).
- **serving_size_grams** : Your best estimate of the item's weight in grams.
- **nutrition** : The nutritional info for that *single item*.
- **meal_quality** : Estimate a reasonable natural number between 0 and 10, where 0 is worst meal quality
- **goal_fit_percent** : Set to a reasonable value between 0 and 1 based on the user's selected goal.
- **calorie_density_cal_per_gram** : With the estimated meal calories and weight, calculate the calorie density of the meal.
- **errorMessage** : Set to a reason if **success** is **false** , otherwise **null** ; It is crucial that estimates are as accurate as possible. For estimating meal quality, take into account how each of the meal's components contribute to the overall healthiness of the meal, and also consider how the meal has been cooked, e.g., being deep-fried or steamed would be examples that may impact meal quality. The model should always use all information it can gather from the inputs, so we can make sure its answer is as accurate as possible.

6. Desired format: As already mentioned in point 4, only the JSON object should be returned following the template given.

7. Example Input: Image (see attachment) + voice recording describing the meal I made. Instead of sending you an audio now, I will just skip the step of recording an audio and will just directly supply you with the audio transcription. The transcription part starts and ends with triple quotes: "" I made a smashed burger with a 200g beef patty, brioche bun, some cheddar, onions and bacon as well as a little coleslaw. Additionally, I made some steak fries in my air fryer as a side dish."" The desired output would then look something like this, although calorie values are just randomly chosen here and NOT accurate for this meal. ""

```

{
  "success": true,
  "requestId": "img_analysis_...",
  "items": [
    {
      "name": "SmashedBurger with Steak Fries",
      "confidence": 0.9,
      "serving_size_grams": 400,
      "nutrition": {
        "calories": 900,
        "protein_g": 90.2,
        "fat_g": 30,
        "carbohydrates_g": 20,
        "meal_quality": 5,
        "goal_fit_percent": 0.5,
        "calorie_density_cal_per_gram": 1.65
      }
    }
  ]
}

```

```

    }
  ],
  "errorMessage": null
}'''

```

Google Gemini3 Flash Preview

This is an excellent foundation. To ensure the model provides the highest level of accuracy, I noticed a few areas that need clarification or explicit instructions within the prompt:

- Conflicting instructions on JSON structure** In Rule #3, you say "Create one object for each distinct food item," but in your example, you grouped the meat, bun, and fries into a single entry called "Smashed Burger with Steak Fries." I need to know if you want the model to break the plate down into components (e.g., Patty, Bun, Fries, Coleslaw) or provide a single aggregate meal object.
- goal_fit_percent context** You mention the value should be based on the "user's selected goal." However, the model currently doesn't know the goal. Should I include a placeholder for the user's goal (e.g., weight loss, muscle gain) in the prompt, or should the model assume a "standard healthy diet" as the baseline?
- Ambiguity in Image vs. Text** Sometimes users might say "I ate a small burger" in the audio, but the image shows a triple-patty burger. We need a "Source of Truth" rule (usually, trust the image for portion size, trust the text for ingredients/preparation methods).

Here is the draft of the optimized prompt based on your current details. I have addressed the logic for multi-modal analysis and formatting constraints.

The Engineered Prompt

Role: You are a professional Nutritionist and Vision Recognition AI specialized in meal tracking.

Task: Analyze the provided image and accompanying text/audio transcription to estimate the nutritional content of the meal.

Input Handling Instructions

- Analyze Image:** Identify all food items, estimating volume and weight based on visual cues (e.g., plate size, utensil scale).
- Analyze Text/Audio:** Use the provided text to refine ingredient identification and preparation methods (e.g., "air-fried" vs "deep-fried", "200g patty").
- Resolve Conflicts:** If the text provides specific weights (e.g., "200g beef"), prioritize the text over visual estimation. If the text mentions an item that is not visible (e.g., "I also had a soda"), include it in the analysis.

Nutritional Logic

- Calorie Density:** Calculate as $\text{calories} / \text{serving_size_grams}$.
- Meal Quality (0-10):** Rate based on nutrient density, processing level, and cooking method. High quality (8-10) = whole foods, steamed/grilled. Low quality (0-3) = highly processed, deep-fried.

quality (8-10) = whole foods, steamed/grilled Low quality (0-5) = highly processed/deep-fried, high sugar.

- **Goal Fit (0.0 - 1.0):** Unless otherwise specified, assume a "General Health & Maintenance" goal. Use 0.7-1.0 for balanced macros and 0.0-0.4 for excessive saturated fats or empty calories.

Output Format

Return **ONLY** a valid JSON object. Do not include any conversational text, markdown formatting blocks (unless requested), or explanations.

Schema

json

🔍 Einklappen Kopieren

```
{
  "success": boolean,
  "requestId": "img_analysis_[unique_hash]",
  "items": [
    {
      "name": "String",
      "confidence": float (0.0-1.0),
      "serving_size_grams": integer,
      "nutrition": {
        "calories": integer,
        "protein_g": float,
        "fat_g": float,
        "carbohydrates_g": float,
        "meal_quality": integer (0-10),
        "goal_fit_percent": float (0.0-1.0),
        "calorie_density_cal_per_gram": float
      }
    }
  ],
  "errorMessage": "String or null"
}
```

Constraint Checklist

- If no food is recognized, set **success** to **false** and provide an **errorMessage**.
- Ensure the sum of macros roughly aligns with the total calorie count (4/4/9 kcal per gram).
- Create a distinct object in the **items** array for each primary component of the meal.

How do we proceed?

Does the "one object per food item" rule (Rule 3) vs. your "single meal object" example need to be adjusted? Once you clarify if you want **granular items** (Patty, Bun, Fries) or a **total meal summary**, I will

Thank you for the detailed explanation. Let's clarify the three points:

1. I want the model to provide a single aggregated meal object, such that we only have one nutrition info for the whole input.
2. You are right, I will implement this afterwards. Let's for now use a placeholder. I will then change it to the respective variable once available.
3. That's a good point. Let's stick with your proposed "Source of Truth" rule: trust the image for portion size, trust the text for ingredients & preparation methods.

Google Gemini 3 Flash Preview

This clarity helps perfect the logic. I have refined the prompt to ensure the model aggregates all components into a single `items` entry while maintaining the "Source of Truth" hierarchy for conflicting data.

Here is the final version of the engineered prompt:

The Engineered Prompt

Role: You are a professional Nutritionist and Vision Recognition AI specialized in high-accuracy meal tracking.

Task: Perform a multimodal nutritional analysis of a meal using the provided image and a text-based description (which may be a transcription of a voice note). Your goal is to provide a single, comprehensive nutritional estimate for the entire meal.

Input Evaluation Logic

1. **Visual Analysis:** Identify all food items, estimating volume and weight based on visual cues (plate size, scale of items).
2. **Contextual Analysis (Text/Audio):** Extract specific details regarding ingredients & hidden components (e.g., "oils," "sugar"), and preparation methods (e.g., "air-fried" vs "deep-fried").
3. **The "Source of Truth" Rule:**
 - **Portion Size:** Trust the visual evidence in the image for volume and scale unless the text specifies an exact weight (e.g., "200g patty").
 - **Ingredients & Preparation:** Trust the text for what is *inside* the food or how it was cooked, especially for things not visible to the eye (e.g., "cooked in butter," "lean beef").
4. **Aggregation:** Combine all recognized components (e.g., burger, side of fries, sauce) into a single consolidated entry in the `items` array.

Calculations

- **Calories:** Ensure the total calories are consistent with the estimated macros (approx 4 kcal/g for protein/carbs and 9 kcal/g for fats).

- **Meal Quality (0-10):** Evaluate based on nutrient density and cooking methods (e.g., Steamed broccoli + Salmon = 9; Deep-fried burger + Fries = 4).
- **Goal Fit (0.0 - 1.0):** Placeholder logic: Assume the user's goal is `{USER_GOAL_PLACEHOLDER}`. Score 1.0 for perfect alignment and 0.0 for total misalignment.
- **Calorie Density:** `total_calories / total_serving_size_grams`.

Output Constraints

- Return **ONLY** a valid JSON object
- No markdown code blocks, no preamble, and no post-analysis commentary
- If no food is detected or the image is too blurry, set `success: false` and provide a descriptive `errorMessage`.

JSON Schema

json

⌵ Einklappen Kopieren

```
{
  "success": boolean,
  "requestId": "img_analysis_[timestamp_random_hash]",
  "items": [
    {
      "name": "Single string describing the full meal",
      "confidence": float (0.0-1.0),
      "serving_size_grams": integer,
      "nutrition": {
        "calories": integer,
        "protein_g": float,
        "fat_g": float,
        "carbohydrates_g": float,
        "meal_quality": integer (0-10),
        "goal_fit_percent": float (0.0-1.0),
        "calorie_density_cal_per_gram": float
      }
    }
  ],
  "errorMessage": string or null
}
```

Engineering Notes for the User:

- **Universal Placeholder** I inserted `{USER_GOAL_PLACEHOLDER}`. When you send this to the API, simply string-replace that with "Weight Loss," "Muscle Gain," or "Maintenance"
- **Multi-item vs. Single-item:** This prompt now strictly enforces the "Single Aggregate Object"

requirement you requested in your last message, ensuring the `items` array always contains exactly one object if successful

- **Handling Audio:** Since you mentioned audio is a typical use case, ensure your application passes the *transcription* of that audio into the text field of the model alongside the image.

Is this prompt ready for your application, or would you like to run a test with the burger image provided earlier?

Great, could you also give me a corresponding system prompt I can use?

Google Gemini 3 Flash Preview

To provide the best results, you should split the logic: the **System Prompt** defines the persona, rules, and output constraints (the "who" and "how"), while the **User Prompt** provides the specific data (the "what").

Here is the recommended **System Prompt** designed to keep the model strictly within its role and the JSON format.

System Prompt

text

🔼 Einklappen Kopieren

You are a professional Nutritionist and Vision Recognition AI. Your sole purpose is to

OPERATIONAL RULES:

1. SOURCE OF TRUTH: If there is a conflict, trust the IMAGE for portion sizes and volume.
2. AGGREGATION: Consolidate all food items found in the input into a single aggregate object.
3. CALCULATION RULES:
 - Calories must be mathematically consistent with macros (4kcal/g for protein/carbs, 9kcal/g for fats).
 - Meal Quality (0-10): Rate 10 for nutrient-dense, whole foods; 0 for highly processed.
 - Calorie Density: Total Calories divided by Serving Size in grams.
4. GOAL FIT: Evaluate "goal_fit_percent" based on the user's specific goal provided in the input.
5. OUTPUT RESTRICTION: You must output ONLY a valid JSON object. Do not include markdown or any other text.

JSON SCHEMA:

```
{
  "success": boolean,
  "requestId": "string",
  "items": [
    {
      "name": "string",
      "confidence": float
```



```

    "confidence": float,
    "serving_size_grams": integer,
    "nutrition": {
      "calories": integer,
      "protein_g": float,
      "fat_g": float,
      "carbohydrates_g": float,
      "meal_quality": integer,
      "goal_fit_percent": float,
      "calorie_density_cal_per_gram": float
    }
  }
],
"errorMessage": string or null
}

```

How to use this with the User Prompt

When you send the request to the API, you will pair the System Prompt above with a **User Prompt** structured like this:

User Prompt Example

text

🔼 Einklappen Kopieren

```

USER GOAL: {User's Goal, e.g., Muscle Gain}
INPUT TRANSCRIPTION: "" I made a smashed burger with a 200g beef patty, brioche buns,
IMAGE: [Attached Image]

```

Why this works

1. **Strictness** By putting the JSON schema and the "source of truth" rules in the System Prompt, the model is less likely to deviate even if the user provides confusing audio transcriptions.
2. **Clean Output** The instruction "Do not include markdown formatting" ensures your code doesn't break when trying to parse the JSON (it removes the ````json` wrapper).
3. **Accuracy** It forces the model to verify its own math (the 4/4/9 calorie rule).

Does this structure work for your implementation?

