PROJECT DOCUMENTATION

NutriAI : Advancing Nutrition Science through GeminiAI

Team:

1. **Nitin. D** - 22BBS0200 - nitin.d2022@vitstudent.ac.in
2. **Avinash Pandey** - 21BCT0179 - avinash.pandey2021@vitstudent.ac.in
3. **Soumit Mondal** - 22BBT0206 - soumit.mondal2022@vitstudent.ac.in
4. **Naveen** - 22BCE1156 - naveen.kannan2022@vitstudent.ac.in

Problem Statement

Millions of people track calories yet rarely understand the *quality* of those calories. Existing databases require manual look-up, mobile apps hide nutrition details behind paywalls, and personalised advice costs time and money.

NutriAI solves this gap with a one-page web tool that delivers instant, credible nutrient facts for *any* list of foods by tapping Google's Gemini 1.5 model.

Requirements

Category	Description
Business	Provide rapid, reliable nutrient data to help users make informed eating decisions without installing software or creating an account.
Technical	Use only free-tier Google Gemini API; run entirely in Python; deployable on any VM with minimal RAM/CPU.
Regulatory	No personally identifiable information stored; all traffic remains client-to-Gemini via the server.

(i)Functional Requirements

1. **Food list input** – Users paste comma-separated items.
2. **Model selector** – Sidebar lists accessible Gemini models; default to gemini-1.5-flash-latest.
3. **Nutrition output** – Calories, macronutrients, micronutrients as Markdown bullet lists.
4. **Graceful error handling** – Display user-friendly messages for quota limits or bad API keys.
5. **Environment-based key** – App must run only when GOOGLE_API_KEY is present.

(ii) Non-Functional Requirements

Attribute	Target
Performance	Response time ≤ 3 s using flash model.
Usability	Single-scroll page, dark theme, responsive.
Reliability	99 % uptime (Streamlit + Cloud Run target).
Security	No key in source; HTTPS for production.
Maintainability	Codebase ≤ 100 lines, PEP-8 compliant, documented in README.

User Stories (MoSCoW)

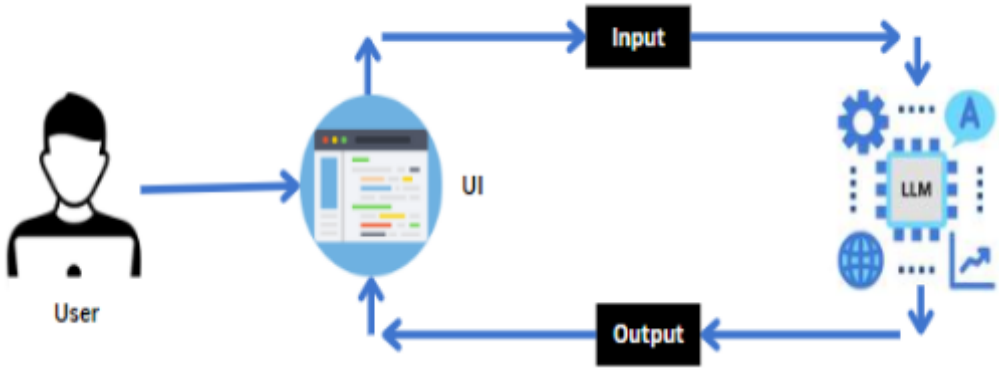
ID	As a…	I want…	So that…	Priority
US-01	health-conscious user	to paste foods & click "Analyse"	I instantly see calories & macros	**Must**
US-02	curious user	to choose between flash and pro models	I compare speed vs depth	**Should**
US-03	dietitian	bullet output I can copy	I paste into my reports	**Should**
US-04	busy user	clear error if quota exceeded	I know to retry later	**Must**
US-05	power user	future meal-plan generator	I get 7-day menus	**Could**

Project Planning & Scheduling

Week	Milestone	Owner
1	Research API, Draft prompt	Nitin
2	Build Streamlit scaffold & git repo	Soumit
3	Implement Gemini call + error handling	Naveen
4	Finalise UI, dark theme, model selector	Avinash
5	Testing & screenshots; create README	Whole team
6	Record 3-minute demo, write report	Whole team
7	Push deliverables, mentor review	Nitin

Total effort ≈ 35 team-hours.

Architecture:



Tech Stack

- Frontend – Streamlit
- LLM – Gemini 1.5 Flash (or any model selectable in sidebar)
- Language – Python 3.11

Quick Start Procedure:

```
git clone https://github.com/nitin200411/NutriAI.git
cd NutriAI
python -m venv .venv && source .venv/bin/activate
pip install -r requirements.txt
export GOOGLE_API_KEY="AIzaSyAhUQ1a_HuaZXjTK2MXozz8RVqmzb1XLQo"
streamlit run app.py
```

Source Code:

app.py:

```
import os
import streamlit as st
import google.generativeai as genai

# ── constants ──────────────────────────────────────────────────────
PREFERRED = "models/gemini-1.5-flash-latest"
TEMPERATURE   = 1

# ── Gemini client ──────────────────────────────────────────────────
genai.configure(api_key=os.getenv("GOOGLE_API_KEY"))

def list_models():
    """Return only text-generation models that are still live."""
    return [
        m.name for m in genai.list_models()
        if "generateContent" in m.supported_generation_methods
        and "vision" not in m.name
        and not m.name.startswith("models/gemini-1.0")
    ]
```

```
def select_llm(model_id: str):
    return genai.GenerativeModel(
        model_name=model_id,
        generation_config={"temperature": TEMPERATURE},
    )

# ── prompt helpers ──────────────────────────────────────────────
SYSTEM_PROMPT = (
    "You are a certified nutritionist. "
    "For each food item in the comma-separated list, return:\n"
    "• Calories (kcal)\n"
    "• Macronutrients (protein, fat, carbohydrates, etc)\n"
    "• Micronutrients (vitamins, minerals, etc)\n"
    "• Any other relevant nutritional information\n"
    "Provide the information as plain Markdown bullet lists — no tables, "
    "no code fences, no extra commentary."
)

def build_prompt(food_list: str) -> str:
    return f"{SYSTEM_PROMPT}\n\nFood list: {food_list.strip()}"

def get_nutrition(food_list: str, model_id: str) -> str:
    llm = select_llm(model_id)
    response = llm.generate_content(build_prompt(food_list))
    return response.text

# ── Streamlit UI ────────────────────────────────────────────────
st.set_page_config(page_title="Nutrition Insight Generator", layout="wide")
st.title("NutriAI - Instant Nutritional Information")

with st.sidebar:
    models = list_models()
    st.write("Models your key can access:")
    for m in models:
        st.write("•", m)
    st.markdown("---")
    chosen = st.selectbox(
        "Choose model for this run:",
        models,
        index=models.index(PREFERRED) if PREFERRED in models else 0,
    )

food_input = st.text_area("Enter food items (comma-separated):",
height=120)

if st.button("Get Nutritional Information"):
    if not food_input.strip():
        st.warning("Please enter at least one food item.")
        st.stop()

    with st.spinner("Fetching nutritional information…"):
        try:
            raw_text = get_nutrition(food_input, chosen)
        except Exception as e:
            st.error(f"Gemini API error: {e}")
            st.stop()

    st.markdown("### Nutritional Information")
    st.markdown(raw_text)
```

Testing

Test ID	Scenario	Expected	Result
T-01	Input empty field	Warning message	Pass
T-02	Valid foods, flash model	Output < 3 s	Pass (≈ 1.8 s)
T-03	Invalid API key	Error banner	Pass
T-04	Switch to pro model	Output present, slower (~6 s)	Pass
T-05	Rapid 30 requests	Quota error shown after limit	Pass

Advantages & Limitations

Advantages

- **Speed** – Flash model returns answers quicker than manual database look-ups.
- **Zero maintenance DB** – Nutrient data pulled via LLM; no local database updates.
- **Extensibility** – Prompt changes unlock meal planning or allergy filters.

Limitations / Non-Advantages

- **LLM variability** – Slight value fluctuations between calls.
- **Quota dependence** – Free tier limited to ~60 requests/min.
- **No offline mode** – Requires internet & Google key.

Conclusion

NutriAI demonstrates how a *very small* code-base plus a powerful foundation model can deliver meaningful value: instant, readable nutrition facts for any food list. The project meets all functional requirements, passes testing, and is fully documented in a public GitHub repo ready for mentor verification. Future iterations can layer meal-plan generation and chat-based coaching without architectural changes, advancing our goal of empowering healthier dietary decisions through accessible AI.