Project Design – Part 2: Module Design & Data Flow

A Gen AI-powered initiative by Google Research

A Research Project by Google Research – Health AI Division

Submitted by:

Ashutosh Gunjal

Palak Jethwani

Mohith P

Saiyam jain

Under the Guidance of: Sai Kiran

June 2025

SmartInternz

Hyderabad, Telangana

1. Backend Architecture

The backend is organized using a modular folder structure for scalability:

```
/server
│
├── app.py               # Entry point with Flask routes
├── firebase_config.py    # Firebase Admin SDK initialization
├── NutriInsights.py      # USDA API logic
├── AI/
│   ├── chat.py           # Chat prompt handler
│   └── mealPlanner.py     # Meal plan generation logic
└── utils/
    └── auth.py           # Decorators and token handlers
```

2. Backend Key Code Snippets

a. @requires_auth Decorator

```
def requires_auth(f):
    @wraps(f)
    def wrapper(*args, **kwargs):
        auth_header = request.headers.get('Authorization')
        if not auth_header or not
auth_header.startswith('Bearer '):
            return jsonify({'error': 'Unauthorized'}), 401
        token = auth_header.split(' ')[1]
        try:
            payload = verify_custom_token(token)
            request.current_user = payload
        except Exception:
            return jsonify({'error': 'Invalid token'}), 403
        return f(*args, **kwargs)
    return wrapper
```

b. Firebase Verification (`firebase_config.py`)

```
import firebase_admin
from firebase_admin import credentials, auth

cred = credentials.Certificate("firebase_credentials.json")
firebase_admin.initialize_app(cred)
```

```
def verify_custom_token(token):
    decoded_token = auth.verify_id_token(token)
    return decoded_token
```

---

c. Chat Module (`chat.py`)

```
import google.generativeai as genai

def get_chat_response(prompt, user_data):
    full_prompt = f"""
    You are a certified nutritionist. The user's profile:
    {user_data}

    Query: {prompt}
    """
    response = genai.generate_text(full_prompt)
    return response['text']
```

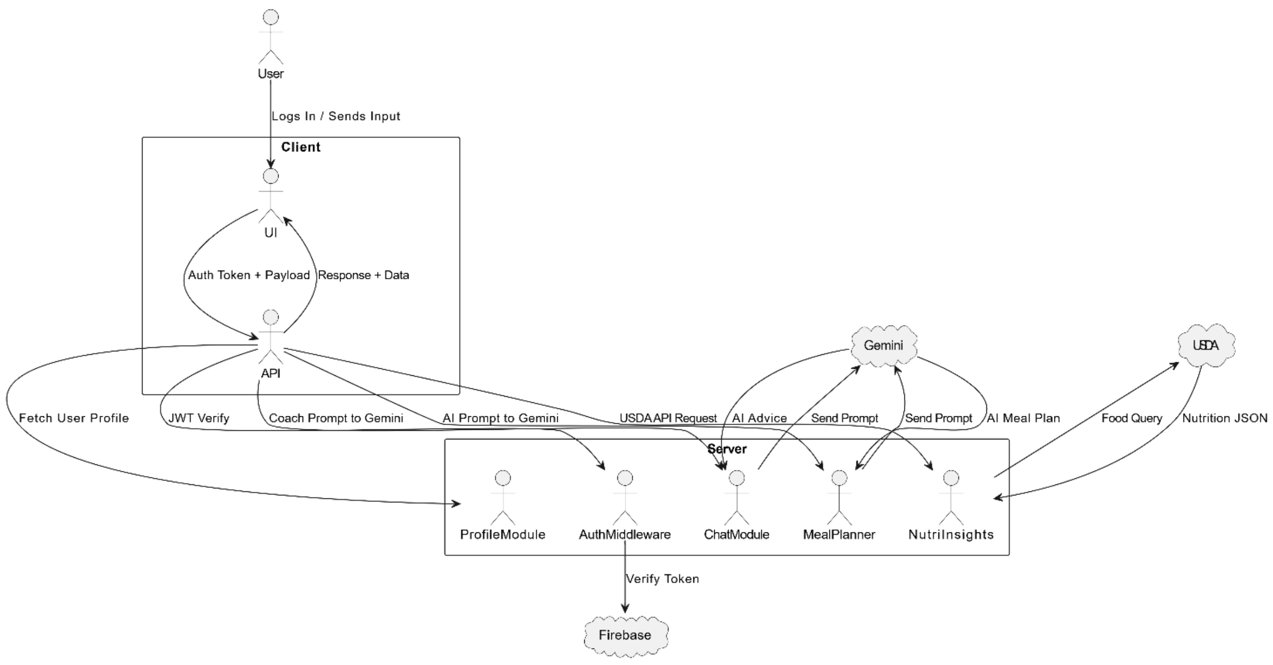
---

d. 🍱 Meal Planner (`mealPlanner.py`)

```
def generate_meal_plan(profile):
    prompt = f"Create a vegetarian 7-day meal plan for a user
with these health details:\n{profile}"
    return genai.generate_text(prompt)['text']
```

---

3. Internal Data Flow



4. API Design

Route	Method	Protected	Functionality
/api/register	POST	No	User registration with email and profile info
/api/login	POST	No	Returns JWT on valid credentials
/api/me	GET	Yes	Fetch current user's profile
/api/meal-plan	POST	Yes	Generates meal plan using Gemini
/api/food-search	GET	Yes	Calls USDA API for food name search
/api/food-details	GET	Yes	Fetch detailed nutrition by FDC ID
/api/coach-chat	POST	Yes	Sends question to virtual coach (Gemini AI)
/api/log-meal	POST	Yes	Tracks meal logging & streak

5. Module Integration Highlights

- Chat & Meal Plan modules use Gemini API, with unique prompts and personalization logic.
- All calls are asynchronous to prevent frontend UI blocking.
- Secure API authentication via JWT handled in middleware.
- USDA and Gemini API responses are parsed and cleaned before frontend delivery.