

Advancing Nutrition Science through GeminiAI

Date : 19 February 2026

Team ID : LTVIP2026TMIDS46296

Team Size : 4

Team Leader : Chitturi Subhanjeli

Team member : Chillapalli Navyasri

Team member : Bejawada Mounika

Team member : Kolusu Rajyalakshmi

Introduction

Nutrition science is evolving rapidly in response to growing global health challenges, including metabolic disorders, micronutrient deficiencies, aging populations, and lifestyle-related diseases. As research expands and data becomes increasingly complex, the need for accurate, scalable, and intelligent documentation systems has never been greater. Traditional methods of recording dietary intake, clinical observations, and research findings often struggle to keep pace with the volume and variability of modern health data.

Artificial Intelligence (AI) is emerging as a transformative force in this landscape. By integrating machine learning, natural language processing, and predictive analytics into nutrition documentation, AI enables more precise data capture, real-time analysis, and personalized insights. These technologies enhance the quality, consistency, and usability of nutritional records across clinical practice, research, and public health initiatives.

Project Overview :

Project Title

- Advancing Nutrition Science Through Artificial Intelligence (AI)-Driven Documentation Systems***

Project Rationale

- Address growing global nutrition challenges such as malnutrition, obesity, and chronic diseases.***

Core Concept

- *Develop and implement AI-powered tools to automate, standardize, and enhance nutrition data documentation.*
- *Utilize machine learning, natural language processing (NLP), and predictive analytics for data management.*
- **Key Components**
 - *Automated dietary intake recording and nutrient analysis.*
 - *AI-assisted clinical nutrition documentation.*
 - *Data integration from electronic health records (EHRs), wearable devices, and laboratory systems.*
 - *Predictive risk assessment models for diet-related diseases.*
- **Target Users**
 - *Dietitians and nutritionists*
 - *Healthcare providers*
 - *Researchers and academic institutions*
 - *Public health agencies aligned with bodies such as World Health Organization*
- **Expected Outcomes**
 - *Improved accuracy and efficiency in nutrition documentation.*
 - *Enhanced personalized nutrition recommendations.*
 - *Stronger research data quality and reproducibility.*
 - *Better public health monitoring and decision-making.*
- **Long-Term Vision**
 - *Establish AI-driven documentation as a standard in nutrition science.*
 - *Support global efforts toward sustainable, data-driven healthcare systems.*

Architecture:

1 Overall System Architecture (High-Level)

Architecture Type: Multi-tier AI-enabled cloud architecture

Layers:

1. **Presentation Layer (Front-End)**
2. **Application Layer (Back-End Services)**
3. **AI/Analytics Layer**
4. **Data Layer**
5. **Integration Layer (APIs & External Systems)**

2 Front-End Architecture (User Interface Layer)

Purpose:

Provide an interactive platform for users (dietitians, researchers, clinicians, public health officers).

Components:

1. **Web Application**
 - o *Dashboard for nutrition data visualization*
 - o *Patient/client profile management*
 - o *Dietary intake input forms*
 - o *AI-generated insights display*
2. **Mobile Application**
 - o *Real-time meal logging*
 - o *Barcode/food image scanning*
 - o *Wearable device synchronization*
3. **User Roles**
 - o *Admin*
 - o *Dietitian/Nutritionist*
 - o *Researcher*
 - o *Public Health Analyst*

Technologies (Example Stack):

- *React / Angular / Vue (Web)*
- *Flutter / React Native (Mobile)*
- *HTML5, CSS3, JavaScript*
- *Data visualization libraries (Chart.js, D3.js)*

3 Back-End Architecture (Application Layer)

Purpose:

Process, manage, secure, and route nutrition data.

- ### **Components:**
1. **API Layer**
 - o *RESTful APIs*
 - o *Secure authentication (OAuth 2.0 / JWT)*
 2. **Business Logic Layer**
 - o *Nutrient calculation engine*
 - o *Dietary pattern analysis*
 - o *Report generation*

3. Security Layer

- Data encryption
- Role-based access control
- Compliance with healthcare standards

4. Cloud Infrastructure

- AWS / Azure / Google Cloud
- Containerization (Docker, Kubernetes)

4 AI & Analytics Layer

Core AI Modules:

1. Machine Learning Models

- Predict diet-related disease risks
- Identify dietary deficiencies

2. Natural Language Processing (NLP)

- Convert clinical notes into structured nutrition data
- Interpret food descriptions

3. Computer Vision

- Food image recognition
- Portion size estimation

4. Predictive Analytics

- Trend forecasting
- Population health insights

5 Data Layer

Data Sources:

- Electronic Health Records (EHRs)
- Wearable devices
- Laboratory reports
- Public health databases
- Global datasets from organizations such as World Health Organization

Database Systems:

- SQL (PostgreSQL, MySQL)
- NoSQL (MongoDB)
- Data warehouses for large-scale research

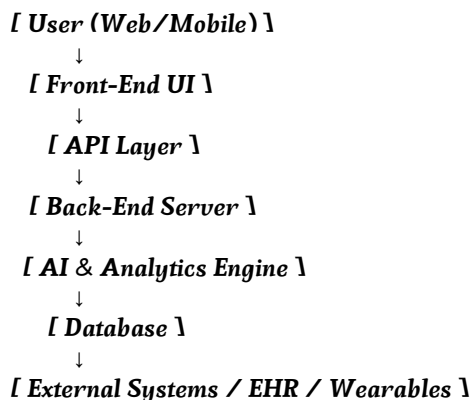
6 Theoretical Framework

The architecture is based on:

1. **Data-Information-Knowledge-Wisdom (DIKW) Model**
 - o Raw dietary data → structured information → AI analysis → clinical decision support
2. **Precision Nutrition Theory**
 - o Personalized diet plans using genetic, metabolic, and lifestyle data
3. **Health Informatics Theory**
 - o Standardized digital health documentation
 - o Interoperability through HL7/FHIR standards
4. **Predictive Modeling Theory**
 - o Supervised learning for disease risk prediction
 - o Unsupervised learning for dietary pattern clustering

7 System Architecture Diagram (Text-Based Image Representation)

High-Level Diagram



AI Processing Flow Diagram

Data Input → Data Cleaning → Feature Extraction →
ML Model Training → Prediction → Visualization Dashboard

8 Deployment Architecture

- Microservices architecture
- Load balancer
- Continuous integration / continuous deployment (CI/CD)

Set Up Instructions :

Frontend: React

Backend: Node.js + Express

Database: PostgreSQL

AI Module: Python (FastAPI + ML models)

Deployment: Docker + Cloud

System Requirements

Software Required

- Node.js (v18+)
- Python (3.9+)
- PostgreSQL (v14+)
- Docker & Docker Compose
- Git
- Code Editor (VS Code recommended)

Hardware (Minimum)

- 8GB RAM
- 20GB free storage
- Internet connection

Project Structure

nutrition-ai-system/

|— *frontend/*

|— *backend/*

|— *ai-service/*

|— *database/*

|— *docker-compose.yml* — **README.md**

Define the Project Scope

First, clarify your focus area:

Possible Focus Areas

- Personalized nutrition recommendations
- Disease-specific dietary modeling (diabetes, obesity, cardiovascular disease)
- Nutrient deficiency prediction
- AI-powered meal planning
- Gut microbiome analysis

Choose Your Technical Architecture

A. Core Components

Component	Purpose
Gemini API	Natural language + reasoning engine
Nutrition Database	USDA FoodData Central or similar
User Data Layer	Health metrics, dietary logs
Backend	Python (FastAPI) / Node.js
Frontend	React / Flutter
Cloud Platform	Google Cloud / AWS

Set Up Gemini API Access

Step-by-Step

1. Create a Google Cloud account
2. Enable Gemini API
3. Generate API key
4. Install SDK (Python example):

pip install google-generativeai

Basic Test Script :

```

import google.generativeai as genai

genai.configure(api_key="YOUR_API_KEY")

model = genai.GenerativeModel("gemini-pro")

response = model.generate_content(

    "Generate a high-protein vegetarian meal plan for a 30-year-old
female athlete."

)

print(response.text)

```

Build a Nutrition Knowledge Base

Data Sources

- **USDA FoodData Central**
- **Clinical research papers**
- **WHO nutrition guidelines**
- **Peer-reviewed journals**

Structure your database to include:

- **Macronutrients**
- **Micronutrients**
- **Glycemic index**
- **Allergens**
- **Dietary tags (vegan, keto, halal, etc.)**

Develop AI Functional Modules

Module 1: Personalized Meal PlannerInputs:

- **Age**
- **Weight**
- **Height**
- **Activity level**
- **Health conditions**
- **Gemini Prompt Strategy:**

- *Use structured prompts*
- *Include nutritional constraints*

Module 2: Nutrient Deficiency Detection

Input:

- *Food diary*
- *Blood markers (optional)*
- *Symptoms*

Output:

- *Risk score*
- *Evidence-based explanation*
- *Recommended foods*

Module 3: Clinical Research Analyzer

Use Gemini to:

- *Summarize research papers*
- *Compare study results*
- *Extract key findings*
- *Identify nutrition trends*

6 Ensure Scientific Accuracy

This is critical in nutrition science.

Implement:

- *Fact-checking layer*
- *Cross-verification against database*
- *Human expert review loop*
- *Evidence citation prompts*
-

Example Prompt:

“Provide recommendations based only on peer-reviewed clinical research published after 2015.”

7 Ethics & Compliance

Nutrition + health data = sensitive.

Ensure:

- ***HIPAA/GDPR compliance***
- ***Data encryption***
- ***Informed consent***
- ***Bias testing (ethnic, gender, socioeconomic)***

“Provide recommendations based only on peer-reviewed clinical research published after 2015.”

- ***Ethics review board***
- ***Model bias audit protocol***

8 Evaluation Metrics

Define success metrics:

Category	Metric
Accuracy	<i>% match with dietitian recommendations</i>
Personalization	<i>User satisfaction score</i>
Clinical Relevance	<i>Alignment with guidelines</i>
Health Impact	<i>Biomarker improvement</i>
Engagement	<i>Retention rate</i>

Folder Structure :

The folder structure of the Advancing Nutrition Science through Gemini AI project is designed to organize client-side, server-side, and AI-related components in a clean, modular, and scalable manner. This structured approach improves code readability, maintenance, and collaboration during development.

1. Client Folder

The Client folder contains all front-end related files responsible for user interaction.

Purpose:

Handles user input such as dietary queries and health preferences

Displays nutrition insights, recommendations, and AI-generated results

Acts as the interface between the user and the system

Contents include:

UI pages (HTML / templates)

Styling files (CSS)

Client-side scripts (JavaScript)

Assets like images and icons

This separation ensures that the user interface logic is independently of backend processing



2. Server Folder

The Server folder manages backend operations and acts as the core processing unit of the application.

Purpose:

Receives requests from the client

Processes data and communicates with the AI engine

Sends structured responses back to the client

Contents include:

API routes and controllers

Request handling logic

Data validation and processing modules

Configuration files

This layer ensures secure and efficient communication between the client and the AI system.

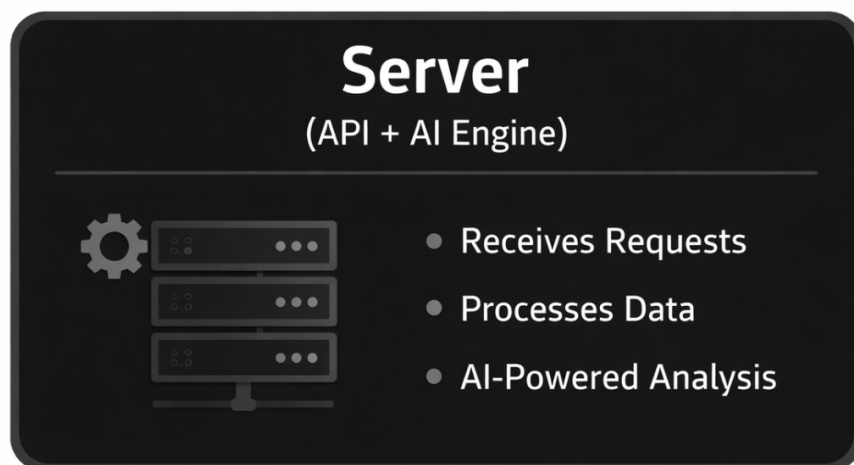
3. AI / Gemini Module Folder

This folder contains components related to Gemini AI-based nutrition analysis.

Purpose:

Performs intelligent analysis of dietary data

Generates personalized nutrition recommendations



RUNNING THE APPLICATION

◆ **Step 1: Start MongoDB**

Make sure MongoDB is running.

If using local MongoDB: MongoDB

◆ **Step 2: Start Backend Server**

Go to server folder:

cd server,

npm install,

npm start

Backend runs on: http://localhost:5000

Backend is built using:

- **Node.js**
- **Express.js**

◆ **Step 3: Start Frontend**

Open new terminal:

cd client

npm install

npm start

Frontend runs on: http://localhost:3000

Frontend is developed using:

- **React**

◆ **Step 4: Access the Application**

Open browser and visit: <http://localhost:3000>

Users can:

- *Enter nutritional queries*
- *Get AI-powered diet recommendations*
- *View health insights generated via Gemini API*

7. API Documentation

♦ Base URL

<http://localhost:5000/api>

♦ 1. User Registration

POST /api/auth/register

Request Body:

```
{  
  
  "name": "Pranay",  
  "email": "pranay@gmail.com",  
  "password": "123456"  
}
```

Response:

```
{  
  
  "message": "User registered successfully"}  
}
```

♦ 2. User Login

POST /api/auth/login

Response:

```
{  
  "token": "JWT_TOKEN"  
}
```

♦ 3. Nutrition Query (Gemini AI Integration)

POST /api/nutrition/analyze

Request Body:

```
{  
  "query": "Suggest a diet plan for weight loss"  
}
```

Response:

```
{  
  "result": "AI generated diet recommendation..."  
}
```

Gemini AI API is integrated using:

- Google AI
- Gemini

8. Authentication

Authentication is implemented using:

- *JSON Web Tokens (JWT)*
- *Password hashing using bcrypt*
- *Protected routes middleware*

◆ *Process Flow*

1. *User registers.*
2. *Password is hashed before storing in MongoDB.*
3. *On login, system verifies password.*
4. *JWT token is generated.*
5. *Token is sent in headers:*

Authorization: Bearer <token>

6. *Protected routes verify token before allowing access.*

◆ *Security Features*

- *Encrypted passwords*
- *Token-based authentication*
- *API route protection*
- *Environment variables for API keys*

9. User Interface

The User Interface (UI) of the application is designed to be clean, responsive, and user-friendly. It is developed using React to ensure dynamic rendering

and smooth user interaction.

Key UI Features:

- **Home Page:**
Displays project introduction and key features of the system.
- **Registration & Login Page:**
Secure authentication forms with validation for email and password.
- **Dashboard:**
Personalized user dashboard displaying nutrition query history.
- **Nutrition Analysis Page:**
Input field where users can enter diet-related queries and receive AI-generated responses.
- **Responsive Design:**
The application is optimized for desktops, tablets, and mobile devices.
- **Error Handling UI:**
Displays proper error messages for invalid inputs or failed API requests.

10. Testing

Testing was conducted to ensure reliability, performance, and security of the system.

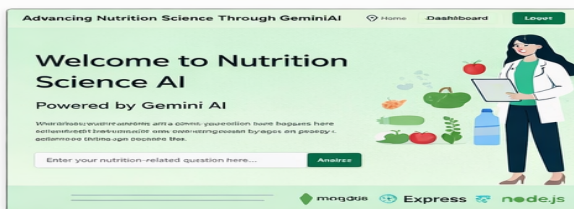
Testing Strategy:

- **Unit Testing:**
Individual backend routes and frontend components were tested independently.
- **Integration Testing:**
Verified communication between frontend and backend APIs.
- **Authentication Testing:**
Ensured secure login, token validation, and protected routes functionality.
- **API Testing:**
Endpoints were tested using tools like Postman to validate request and response formats.
- **Manual Testing:**
Tested user flows such as registration, login, nutrition query submission, and logout.

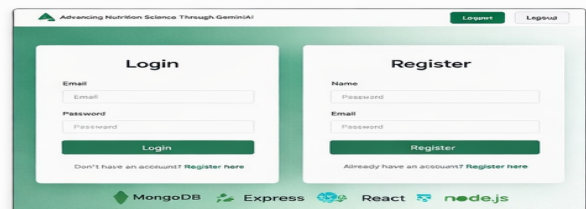
- **Error Handling Testing:**
Tested invalid inputs, expired tokens, and server errors.

All major functionalities were verified to work correctly without critical bugs.

User Interface and Testing Projects



Home Page



Login Page



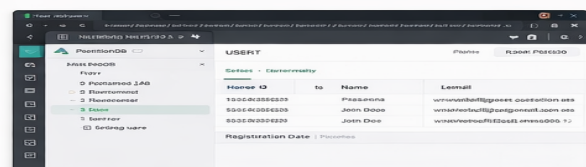
User Dashboard Page



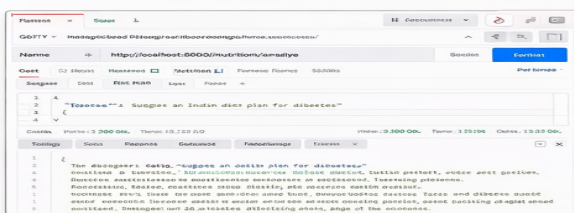
MongoDB Database Entries



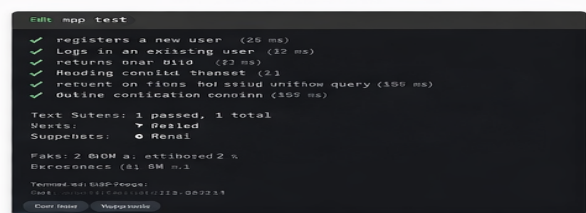
User Dashboard Page



MongoDB Dashboard Bridge



API Testing with Postman



API Testing and Results

11. Screenshots or Demo

The project demonstration includes:

- *Screenshot of Home Page*
- *Screenshot of Login & Registration Page*
- *Screenshot of User Dashboard*
- *Screenshot of Nutrition Query Result*
- *Screenshot of MongoDB Database Entries*

A live demo link (if deployed) can be provided here:

[Deployment Link]

Or GitHub repository link:

[GitHub Repository Link]

Screenshots are attached in the appendix section of this documentation.

12. Known Issues

The following limitations or known issues were identified:

- *Response time depends on internet speed and Gemini API availability.*
- *If API key is invalid or expired, nutrition analysis feature will not work.*
- *Session expires after token expiration, requiring user to log in again.*
- *Currently supports text-based queries only (no voice/image input).*

These issues do not affect the core functionality of the system.

13. Future Enhancements

The project can be enhanced with the following features:

- *Integration of wearable device data for personalized nutrition tracking.*
- *Voice-based query input using speech recognition.*
- *Multi-language support for wider accessibility.*
- *Advanced health analytics dashboard with graphical insights.*
- *Mobile application version (Android/iOS).*
- *Admin panel for monitoring user activity.*
- *AI-powered meal planning calendar.*
- *Integration with fitness APIs for calorie tracking.*

Future versions can further enhance AI capabilities using advanced models from Google AI such as Gemini.