Fitness Assistant System Report

Prerak Patidar

July 6, 2025

Abstract

Fitness Assistant is a full-stack health and wellness management platform built to generate personalized workout plans, diet macros, and daily nutrition guides. It includes AI-backed features for smart Q&A, tracks user data via TinyDB, and leverages a clean Vue-based frontend layout. This document summarizes the architecture, workflows, and user interface.

1 Architecture Overview

1.1 1. Tech Stack

- **Frontend**: Vue 3, Pinia, Axios, Vite
- **Backend**: FastAPI with modular routers
- **AI Services**: Gemini (or OpenAI), called via ‘ask$_{g}emini()$‘**Database** : $TinyDB(JSON-basedembeddedNoSQLDB)$

1.2 2. Infrastructure Flow

- The Vue app calls FastAPI endpoints via Axios.
- FastAPI serves modular routes for ‘/diet‘, ‘/macros‘, ‘/ask‘, ‘/users‘, and ‘/notes‘.
- User profiles are saved and retrieved using TinyDB (‘db.json‘).
- Diet/macros are generated using prompt-engineered AI or rule-based functions.
- Each route sanitizes input and handles partial/fallback logic.

2 Frontend UI & Screenshots

Macros View

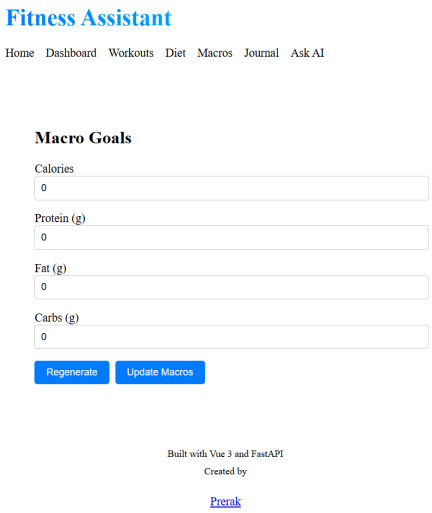

Figure 1: Macro Goals page. This view shows the user's calculated calorie and macro targets based on profile and fitness goal. Regeneration and updates are handled through FastAPI calls.

Ask AI Page

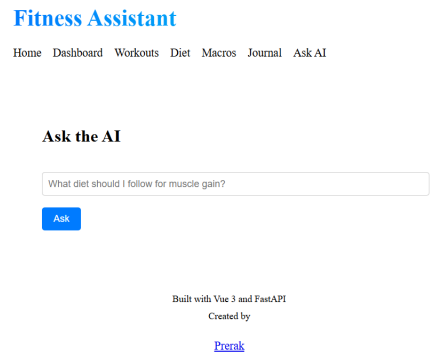

Figure 2: Ask AI page. Users can ask fitness or nutrition questions. The backend wraps the prompt and fetches answers from Gemini.

Workout Dashboard

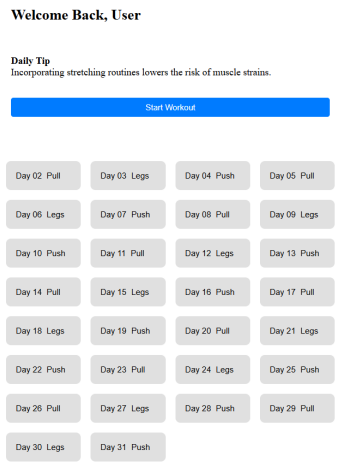

Figure 3: Workout scheduler showing Push-Pull-Legs split over 31 days. Each button loads a structured plan for that day.

Diet Plan Layout

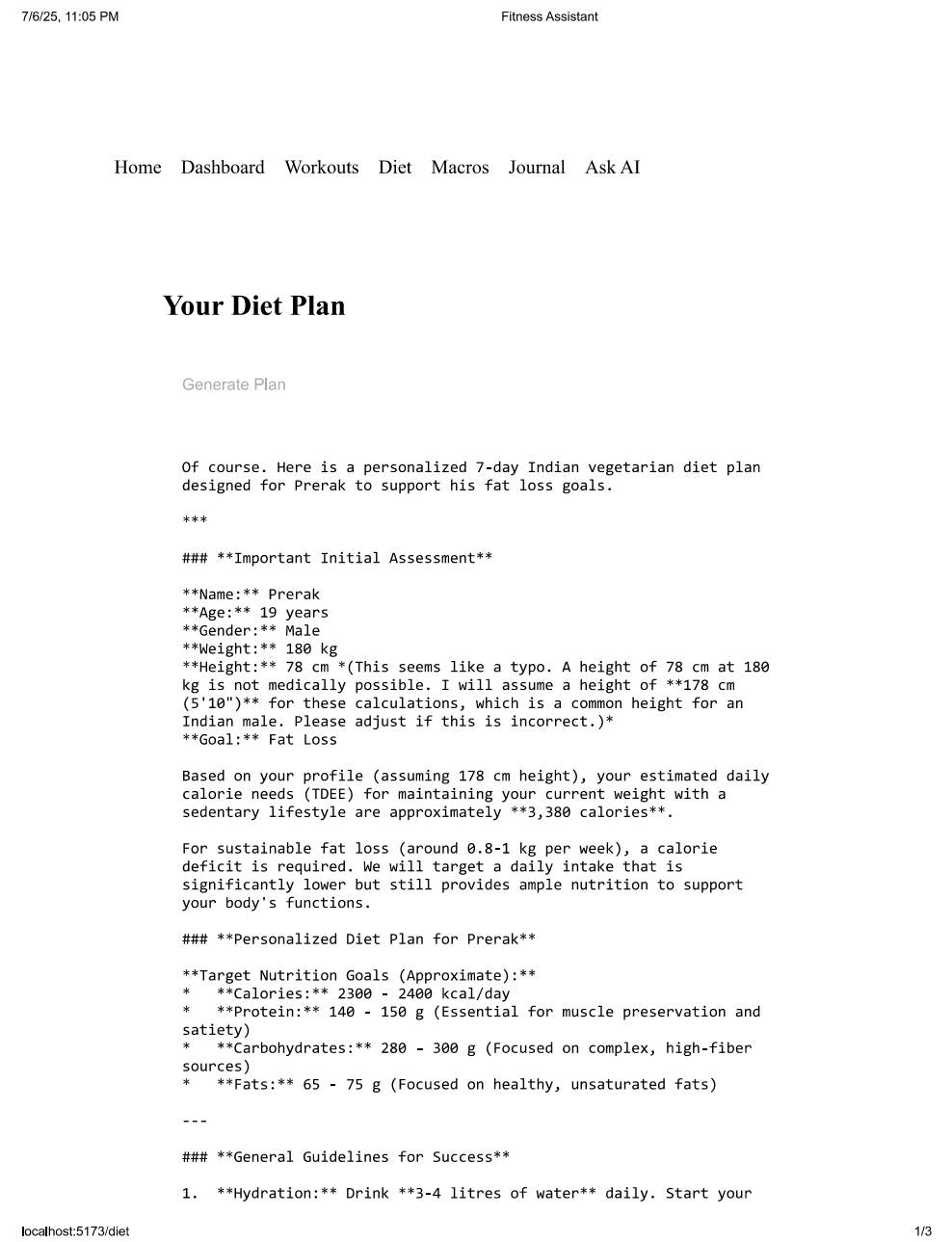

Figure 4: A dynamic diet plan rendered from the AI-generated text. Uses profile age, weight, activity level and fitness goal.

Journal View

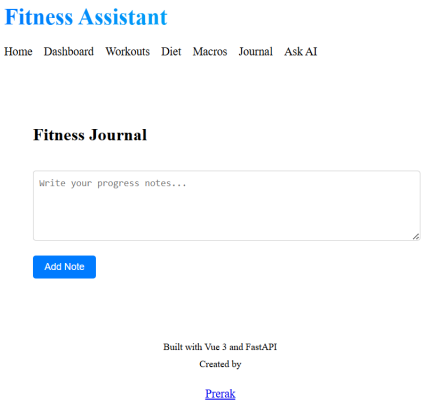

Figure 5: (Planned) Journal screen allows users to record daily progress, notes, and insights. To be stored in the ‘notes‘ table of TinyDB.

Application Layout & Architecture

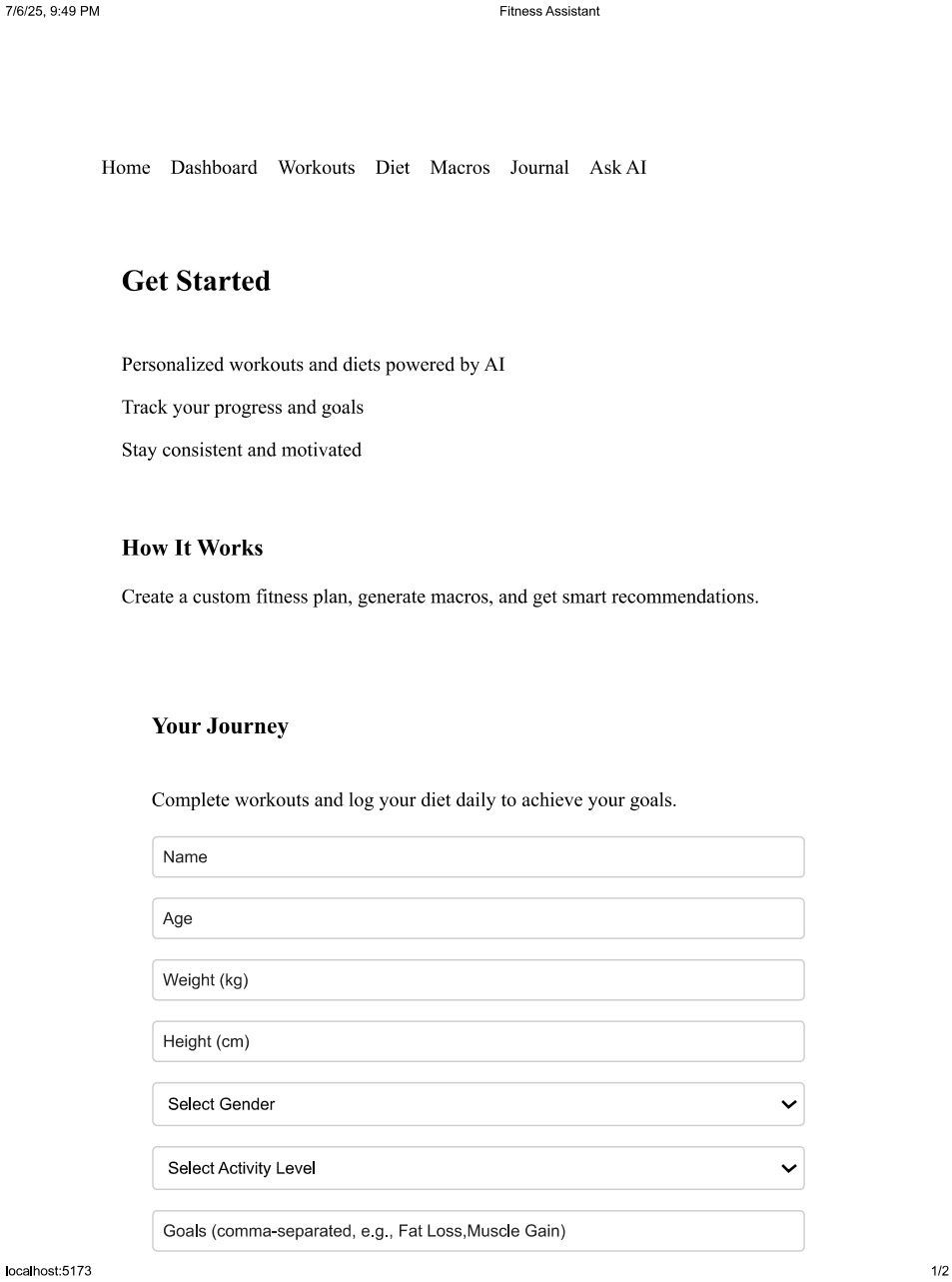

Figure 6: Application structure showing flow between frontend views, API routers, services, and TinyDB.

3 Backend Functional Flow

1. Macro Calculator

- Endpoint: ‘POST /api/macros‘
- Uses profile data to calculate calories and macro goals.

• ‘generate$_{m}acros()$‘$handleslogicandfallbackdefaults$.

2. Diet Generation

- Endpoint: ‘POST /api/diet/‘
- Profile may come from frontend or fallback to TinyDB.

- AI-based text generation via ‘generate$_d$iet()‘.

3. Ask AI

- Endpoint: ‘POST /api/ask‘
- Adds prompt wrapper and routes to Gemini API
- Fallback response is returned if LLM fails

4. Journal/Notes

- Endpoints: ‘GET /api/notes/user$_i$d‘, ‘$POST/api/notes/$‘$Eachentryincludestimestamp, content, andu$

4 Database Schema

TinyDB schema as structured in ‘db.json‘:

- **users**: ‘user$_i$d‘, ‘$name$‘, ‘age‘, ‘$weight$‘, ‘$height$‘, ‘$gender$‘, ‘$activity_level$‘, ‘$goals$‘**diets** : ‘$user_id$‘, ‘$meals[]$‘, ‘$timestamp$‘

- **workouts**: ‘user$_i$d‘, ‘$type$‘, ‘$exercises[]$‘, ‘$timestamp$‘**notes** : ‘$user_id$‘, ‘$text$‘, ‘$timestamp$‘

5 Conclusion

The Fitness Assistant project successfully combines full-stack modularity with AI integration. Its strengths lie in:

- User-personalized flows
- Dynamic AI-backed content
- Lightweight and easily extensible backend
- Clean and minimal frontend UI

Created with Vue 3, FastAPI and curiosity. Developed by Prerak Patidar.