**Multi-Agentic Health Assistant**

**Design Document**

**Introduction** This document outlines the step-by-step design and architecture of a multi-agentic model designed to support an individual's physical and mental well-being through personalized diet, exercise, and mental health management.

The system uses multiple LLMs and vision models and provides a unified interface through a Web-based frontend.

**Step 1: User Profile Initialization** User inputs collected at launch:

* Name
* Age
* Gender
* Height
* Weight
* Fitness goal
* Daily/Today’s Activity Level (not active, lightly active, active, very active)
* Dietary preferences
* Mental health background (optional)
* Daily schedule / Time availability
* Medical conditions (if any)

**Step 2: System Architecture Overview**

Three LLM agents:

**Agents:**

1. Mental Health LLM Agent (Knowledgebase)
2. Diet LLM Agent (with vision model) (Database)
3. Exercise LLM Agent (Database)

**Databases:**

* SQLite/Postgre
* SQL DB

Mental Health LLM Agent

**Functions:**

* Daily check-ins and emotional support
* Guided journaling and motivational prompts
* Memory mode to store flagged conversations

**Features:**

* Sentiment analysis and emotion classification
* Secure memory log for past conversations
* Stores summaries with tags and timestamps
* Enables recall of important past discussions (Flagged by the user through the toggle)

Diet LLM Agent

**Functions:**

* Meal planning and nutritional suggestions
* Vision-based analysis of consumed meals (Vision LLM)

**Features:**

* Vision model integration to analyze meal photos
* Nutrient breakdown and diet goal comparison

Exercise LLM Agent

**Functions:**

* Generate personalized routines
* Track workout completion and calories burned (Store in the database)

**Features:**

* Adjusts difficulty and goals weekly
* Logs performance data

**Extra Feature: Streamlit/Web Frontend (Full Vibe-coding)**

**Tabs:**

* Dashboard
* Mental Health (chat with memory toggle)
* Diet Tracker (upload images, view breakdown, both image and text)
* Exercise Plan (view and log workouts)

**Extra feature (Recommended): Token and History Management**

* Rolling window for each LLM to manage context
* Summarization of older messages
* Long-term memory only for mental health agent

**Extra feature (Optional): Audio Messages**

* User should be able to communicate with LLMs with Audio
* Transcribed text should appear in frontend as user message

**API keys and Tech stack**

For API key, login to: <https://www.together.ai/>

Make an account. It grants you $1 in credits that should be enough for the project.

**Models to use:**

Text Model: LLaMA 3.1 8b

Vision Model: LLaMA 3.2 11b Vision

Embedding: Use any available in Together.ai

**Requirements & Blueprint**

**Mental Health Agent LLM:**

Functional:

* take

**Diet LLM:**

Functional:

* LLaMA 3.2, 11b parameters
* Vision model integration to analyze meal photos
* Nutrient breakdown and diet goal comparison
* Just use API, nutrient breakdown is required as well after picture submission
* “Nutrient breakdown and diet goal comparison” use as prompt with every picture submitted along with

**Exercise LLM:**

**Frontend:**

Site should have 5 pages:

1. Login
2. Signup
3. User-Information view and Changes & Progress Tracker Viewer
4. Main ChatBot

**Database(s):**

1. PostgreSQL

**user\_health\_info:**

id is a primary key and a foreign key used across all tables to track the activity of user across them all

user\_profile:

* user\_info(class with the following attributes, not to be stored as json or anything)

1. name (String)
2. age (4 digit, 1 after decimal)
3. gender (Female, Male are the options, default = Female)
4. height (meters, 4 digit, 3 after decimal, default = 1.7)
5. weight (kgs, 6 digit, 3 after decimal, default =66.4)

* fitness\_goal (90 character max String, default = 'Get into better shape')
* diet\_pref (7 options, 'vegan', 'carnivore', 'both', 'balanced', 'vegetarian', 'pescatarian', 'any', default= any)
* daily\_availibility (class with the following attributes, not to be stored as json or anything)

1. time\_arr   
   2d array, x=3, y=2 time storage arrrays take a value in 24 hour format and minutes, take 1-3 time ranges, which are at max 20 minutes apart, like 11:20 and 11:40 or 23:40 and 00:00, if user enters a time value, compute the 20 minute later value and insert it. Example of a full array:  
   11:23 23:46 19:59  
   11:43 23:59 20:09  
   Example of a default array:  
   12:00 NULL NULL  
   12:20 NULL NULL  
   Example of an array where only 1 time value is given by user:  
   13:50 NULL NULL  
   14:10 NULL NULL(calculate next value 20 minutes apart)

* mental\_health\_background (optional, default = null)
* medical\_conditions(if any, then a string with detail regarding it, default is null )

daily\_stats:

* activity\_level (4 options, not active, lightly active, active, very active, default = active)
* todays\_flag(boolean, default value= false)
* progress\_condition(has only 3 values(positive, negative, neutral))
* days\_done(integer, incremented every day, default=0)
* days\_left( integer, value calculated every day = time\_deadline-days\_done)

other\_storage:

* picture\_analysis (Food Picture Analysis, maximum 1000 character string, default="")
* audio\_transcript (Temporary Audio Input Transcription, maximum 3 minute audio recording allowed, maximum 3000 character string, default=""(meaning nothing))

1. Vector Database - Pinecone

**Non-Functional Requirements:**

1. Memory: Keep older memory and summaries of old chats in the database, manage context, this keeps up long term memory