Meal Planning Assistant - AI Agent Design

SECTION 1: BASIC DETAILS

Name: Shweta Yenaji
AI Agent Title/Use Case: AI agent that suggests personalized meal ideas based on available ingredients, cooking skill level, time constraints, and budget for Indian students/bachelors

SECTION 2: PROBLEM FRAMING

1.1 What problem does your AI Agent solve?

Students and bachelors often struggle with meal planning due to limited cooking skills, tight budgets, and time constraints. They end up eating unhealthy food, wasting ingredients, or spending too much on outside food because they don't know what to cook with what they have.

1.2 Why is this agent useful?

This agent provides personalized meal suggestions that match the user's actual situation - their skill level, available ingredients, time, and budget. It helps them eat better, save money, and gradually improve their cooking skills.

1.3 Who is the target user?

Indian college students living in hostels or PG accommodations, and young bachelors living independently with basic kitchen facilities and limited cooking experience.

1.4 What not to include?

- Complex recipes requiring advanced techniques or expensive equipment
- Elaborate meal prep for families
- Restaurant-style dishes that take hours to prepare
- Non-vegetarian options (keeping it simple initially)

SECTION 3: 4-LAYER PROMPT DESIGN

🔹 3.1 INPUT UNDERSTANDING

Prompt:

You are the Input Understanding module of a meal planning assistant for students and bachelors. Your job is to analyze what the user is asking for and extract key information.

From the user's message, identify and extract:
1. Available ingredients (if mentioned)
2. Time constraint (if mentioned - quick, 30 minutes, etc.)
3. Cooking skill level (beginner, intermediate, or not specified)
4. Budget constraint (if mentioned)
5. Dietary preferences (if mentioned)
6. Meal type (breakfast, lunch, dinner, snack, or not specified)

Format your response as:
INGREDIENTS: [list or "not specified"]
TIME: [constraint or "not specified"]
SKILL: [level or "not specified"]
BUDGET: [constraint or "not specified"]
DIETARY: [preferences or "not specified"]
MEAL_TYPE: [type or "not specified"]
INTENT: [brief summary of what user wants]

What is this prompt responsible for? This prompt extracts structured information from natural language user requests to help other modules understand exactly what the user needs.

🔹 3.2 STATE TRACKER

Prompt:

You are the State Tracker for a meal planning assistant. Your job is to maintain context about the user across conversations.

Current user profile:
- Skill Level: [Beginner/Intermediate/Advanced - default: Beginner]
- Preferred Ingredients: [list of ingredients user commonly has]
- Dietary Restrictions: [any restrictions mentioned]
- Budget Range: [typical budget mentioned]
- Kitchen Equipment: [basic stove/advanced - default: basic]
- Past Liked Meals: [meals user showed positive response to]

- Past Disliked: [meals user rejected or didn't like]

When new information comes in, update the relevant fields and maintain this profile. If this is a new conversation, start with default values.

Current conversation context:
- Last suggestion given: [what was suggested]
- User feedback on last suggestion: [positive/negative/none]
- Current request: [what user is asking now]

How does this help the agent "remember"? This maintains a user profile across conversations, remembering preferences, skill level, and feedback to provide increasingly personalized suggestions.

3.3 TASK PLANNER

Prompt:

You are the Task Planner for a meal planning assistant. Based on the input analysis and user state, decide what steps to take.

Your decision tree:
1. If user provided specific ingredients → suggest recipes using those ingredients
2. If user mentioned time constraint → filter by cooking time
3. If user mentioned budget → consider cost-effective options
4. If user's skill level is beginner → prioritize simple recipes
5. If user gave feedback on previous suggestion → adjust accordingly

Steps to execute:
1. FILTER: Based on constraints (time, budget, skill, ingredients)
2. PERSONALIZE: Consider user's past preferences and profile
3. DIVERSIFY: If suggesting multiple options, vary the suggestions
4. EDUCATE: If user is beginner, consider slightly challenging options to help them grow

Output your plan as:
APPROACH: [how you'll handle this request]
CONSTRAINTS: [what limitations to consider]
PERSONALIZATION: [how you'll customize for this user]

What steps does your agent take internally? The agent filters options based on constraints, personalizes based on user history, and balances meeting current needs with helping the user grow their cooking skills.

🔹 3.4 OUTPUT GENERATOR

Prompt:

You are the Output Generator for a meal planning assistant. Create helpful, encouraging responses for Indian students and bachelors.

Generate responses that include:
1. 2-3 meal suggestions with:
   - Recipe name
   - Cooking time
   - Difficulty level (🌶️ = Easy, 🌶️🌶️ = Medium)
   - Key ingredients needed
   - Brief cooking steps (3-4 steps max)
   - Approximate cost

2. Encouraging tone that makes cooking feel achievable
3. One tip related to their skill level or situation
4. Ask a follow-up question to keep conversation going

Format:
🍽️ **[Recipe Name]** (⏰[time] | [difficulty] | 💰₹[cost])
Ingredients: [list]
Steps: [brief steps]

Pro Tip: [helpful tip]
What sounds good to you? or similar engaging question

What kind of output formatting did you aim for? Friendly, structured format with emojis and clear information hierarchy. Includes practical details (time, cost, difficulty) and maintains an encouraging, conversational tone.

SECTION 4: CHATGPT EXPLORATION LOG

Attempt	Prompt Variant	What Happened	What I Changed	Why I Changed It
1	Input Understanding - basic test	Extracted ingredients and intent correctly in a simple sentence	No changes needed	Baseline check to confirm info extraction works on straightforward input
2	State Tracker - contradiction test	User said "too spicy" then "not spicy enough"—agent resolved contradiction by checking latest input	Made sure latest user feedback always overrides past one	To make agent more responsive to changing preferences
3	Task Planner - skill vs spice clash	User was beginner but wanted very spicy dish → agent avoided complex spicy meals	Adjusted logic to balance spice level with beginner-friendly methods	Prevent overwhelming beginners while honoring taste
4	Output Generator - final polish	Generated 4 good recipes, but two were similar in style (gravy-based)	Added explicit DIVERSIFY logic to mix dry, gravy, and bread dishes	Improve variety and keep user engagement
5	Output tone control test	Tried making agent more casual/funny, but it sounded unnatural	Reverted to warm, friendly tone with light encouragement	Tone should be motivational but not try-hard or artificial
6	Budget edge case	Tested with input: "I have only ₹30" → initial response gave too costly options	Adjusted cost filter logic and included ₹10-₹30 options	Ensure agent remains useful even in very tight budget cases
7	Vague input test	Input like "What to cook?" → agent still asked smart follow-ups	Ensured agent asks clarification questions instead of guessing	Helps gather more info and keeps conversation going

8	Recipe repeat prevention	Agent repeated same recipe after similar request	Added memory check to avoid repeating recently suggested meals	Keep things fresh and surprising for the user

SECTION 5: OUTPUT TESTS

Test 1: Normal Input

Input: "I have rice, dal, and onions. Need something quick for dinner" **Output**:

INGREDIENTS: rice, dal, onions
TIME: quick
SKILL: not specified
BUDGET: not specified
DIETARY: not specified
MEAL_TYPE: dinner
INTENT: User wants a quick dinner recipe using available ingredients

Test 2: Vague Input

Input: "I'm hungry, what should I cook?"

Output: [Mostly "not specified" fields, handled appropriately]

Test 3: Complex Input

Input: "I have ₹100 and 20 minutes, I'm a beginner"

Output: [Extracted budget, time, and skill level correctly]

Test 4: Full Agent Flow

Input: Spicy lunch request with rice/dal/onions/bread/eggs

Output: 4 varied spicy Indian lunch recipes with perfect formatting, costs, times, and engagement

SECTION 6: REFLECTION

6.1 What was the hardest part of this assignment?

Balancing flexibility with structure. Designing prompts that could handle everything from vague inputs (“I’m hungry”) to very specific constraints (₹30, 15 minutes, beginner) required a lot of iteration. Ensuring each layer worked independently but also contributed to the full pipeline was conceptually challenging but rewarding.

6.2 What part did you enjoy the most?

Crafting the Output Generator. Designing a response format that’s helpful, friendly, and motivating for beginner cooks was really fun. Seeing the agent suggest realistic, spicy Indian meals using limited ingredients and still make them sound exciting felt like creating a real virtual buddy for students.

6.3 If given more time, what would you improve or add?

- Integrate a visual layer (images of dishes, basic kitchen tools, spice level icons)
- Support regional Indian cuisines based on user preference (e.g. South Indian, Gujarati)
- Let users “bookmark” or rate recipes for future personalization
- Add support for meal prep plans (e.g. “Make once, eat thrice” strategy for busy weeks)

6.4 What did you learn about ChatGPT or prompt design?

Good prompt design is like good UX—users should feel guided without realizing they're being guided. Modular prompt layers (Input → State → Planner → Output) give much better control and reduce failure cases. Also, small wording changes in prompts can drastically affect tone, clarity, and specificity.

6.5 Did you ever feel stuck? How did you handle it?

Yes—especially when trying to balance "spicy" with "beginner-friendly" cooking. Many spicy Indian dishes require complex techniques. I solved this by curating a small set of spicy yet simple dishes (e.g., masala scrambled eggs, tadka dal fry) and building a rule in the Task Planner to prioritize those for such scenarios.

SECTION 7: HACK VALUE

- Introduced a spice-skill balance logic: spicy dishes that are still beginner-friendly
- Designed multi-dish variation logic: dry/gravy/bread-based per request
- Added positive habit nudging: includes skill-building tips, confidence boosters
- Created an engaging output template with emojis, levels, and follow-up questions
- Agent is highly modular – each layer can be independently improved or swapped