Flavour Fusion: AI-driven Recipe Blogging

INDEX

1. INTRODUCTION

1.1 Project Overview

Flavour Fusion" is an innovative AI-powered web application designed to assist food bloggers, home cooks, and content creators. By leveraging the advanced capabilities of the Google Gemini 1.5 Flash model, the application instantly generates high-quality, structured, and engaging recipe blog posts from simple user inputs. Users can enter a dish name (e.g., "Paneer Butter Masala"), and the system automatically creates a comprehensive blog post including a catchy title, ingredients list, step-by-step cooking instructions, and pro-tips, all within seconds

1.2 Purpose

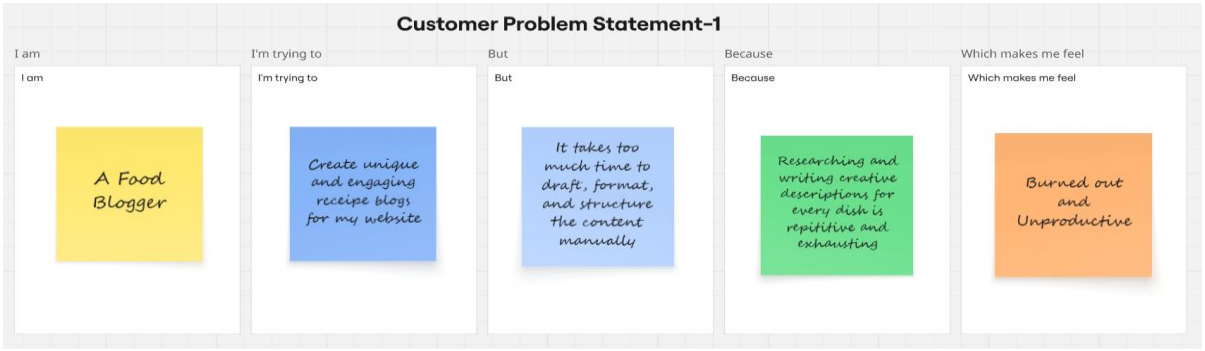
The primary purpose of this project is to address the common challenge of "Writer's Block" faced by culinary enthusiasts. While many individuals possess excellent cooking skills, they often struggle with the documentation and creative writing aspect of blogging. Flavour Fusion aims to:

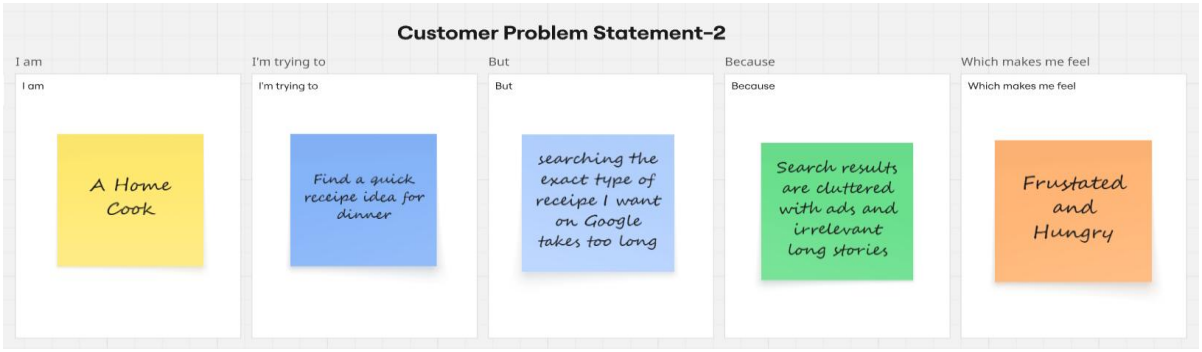
- Automate the tedious process of content drafting.
- Provide a tool for consistent content creation.
- Bridge the gap between culinary skill and digital expression.

2. IDEATION PHASE

2.1 Problem Statement

Food bloggers and aspiring influencers often struggle to maintain consistency in their posting schedules due to the time-consuming nature of writing and formatting recipe content. This leads to burnout, irregular engagement with their audience, and a lack of professional growth.

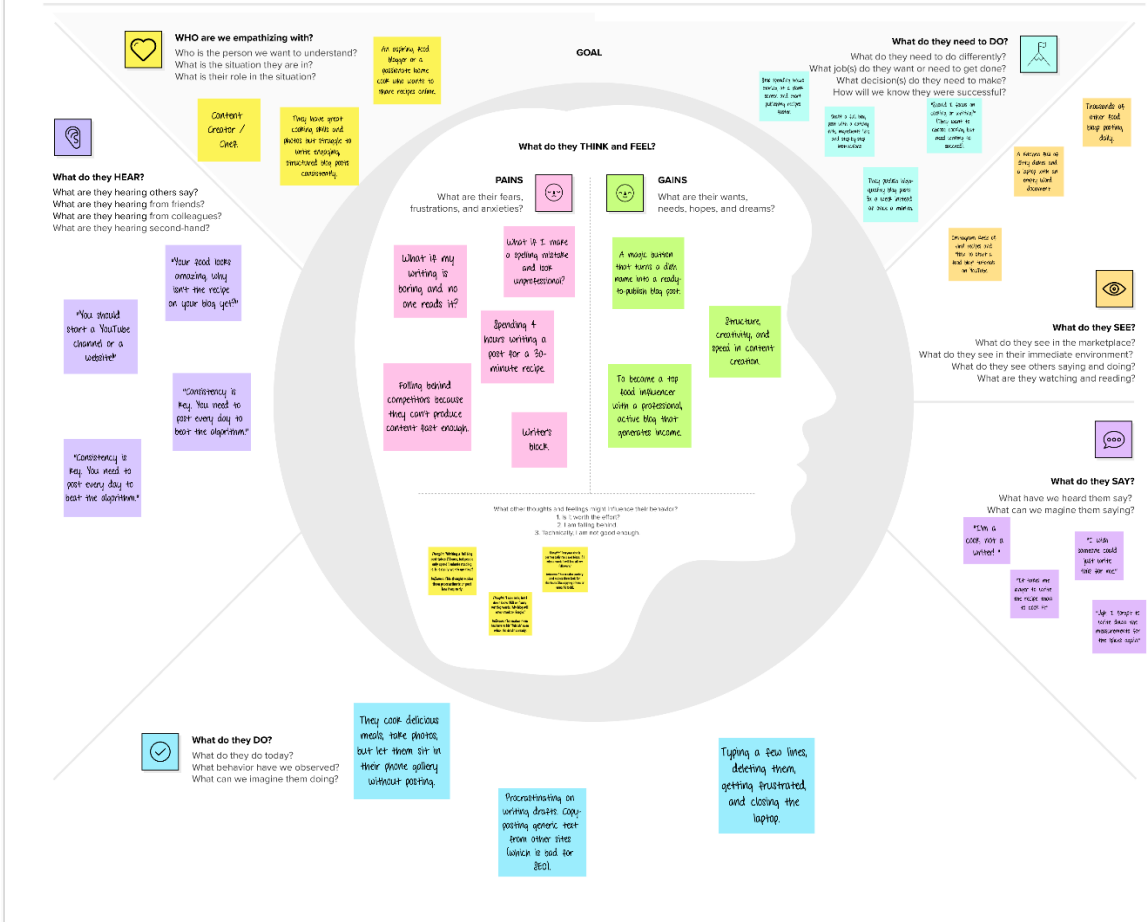




2.2 Empathy Map Canvas

We analyzed our target user (The Aspiring Food Blogger) to understand their needs:

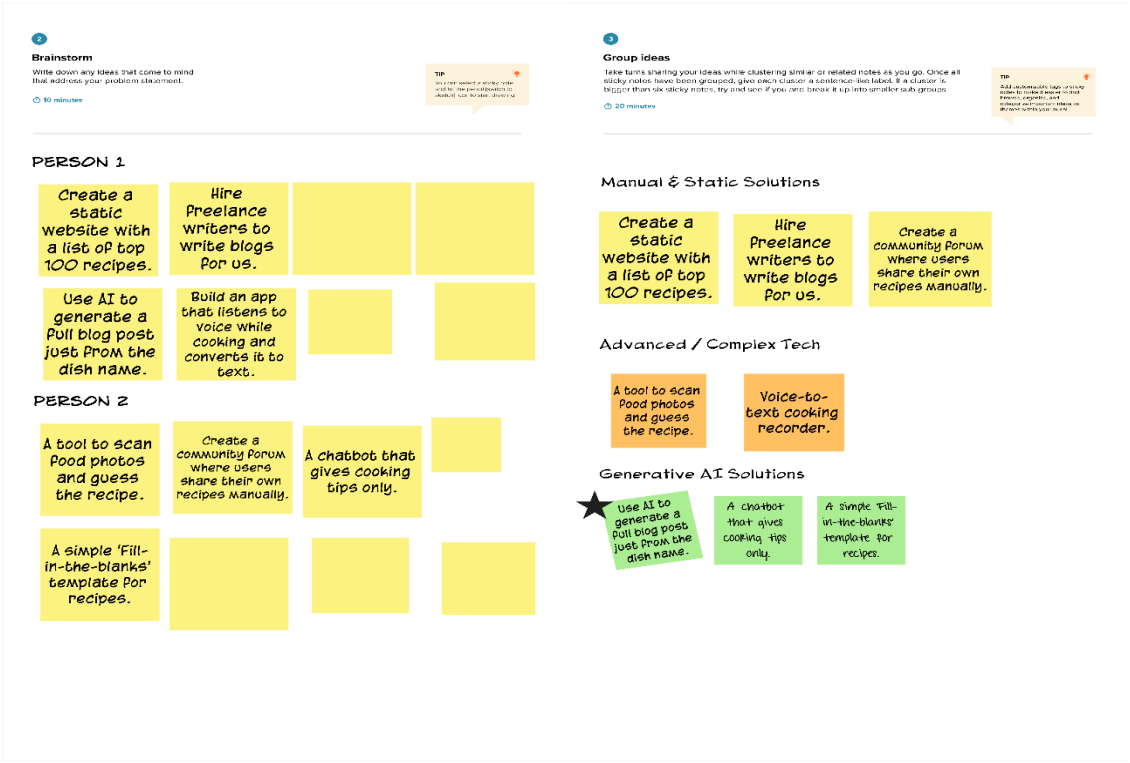
- Says: "I can cook, but I don't know how to write engaging stories."
- Thinks: "I want my blog to look professional, but I have no time."
- Does: Cooks amazing meals but leaves photos in the gallery without posting.
- Feels: Frustrated by the writing process but excited about sharing food.
- Pain: Time constraints and lack of vocabulary.
- Gain: A desire for instant, structured content generation.

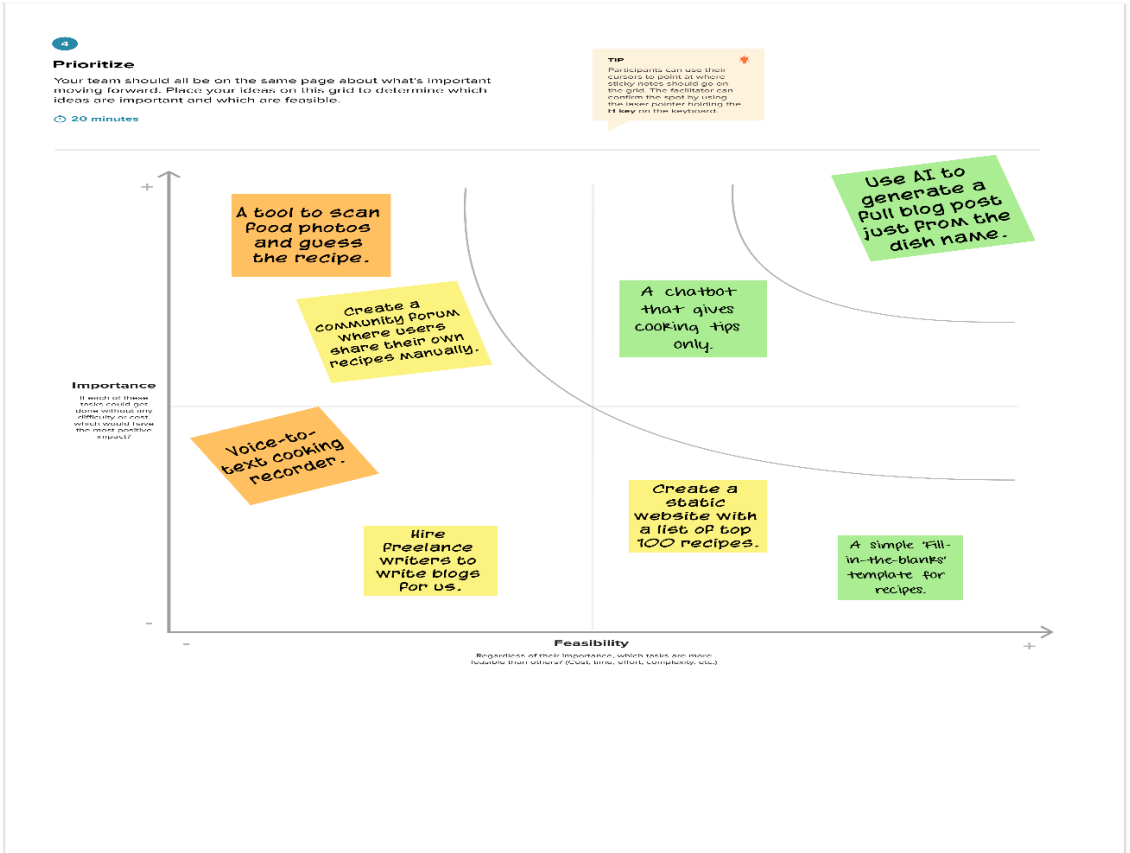


2.3 Brainstorming

We used the Idea Clustering and Prioritization Grid methods to select the best solution.

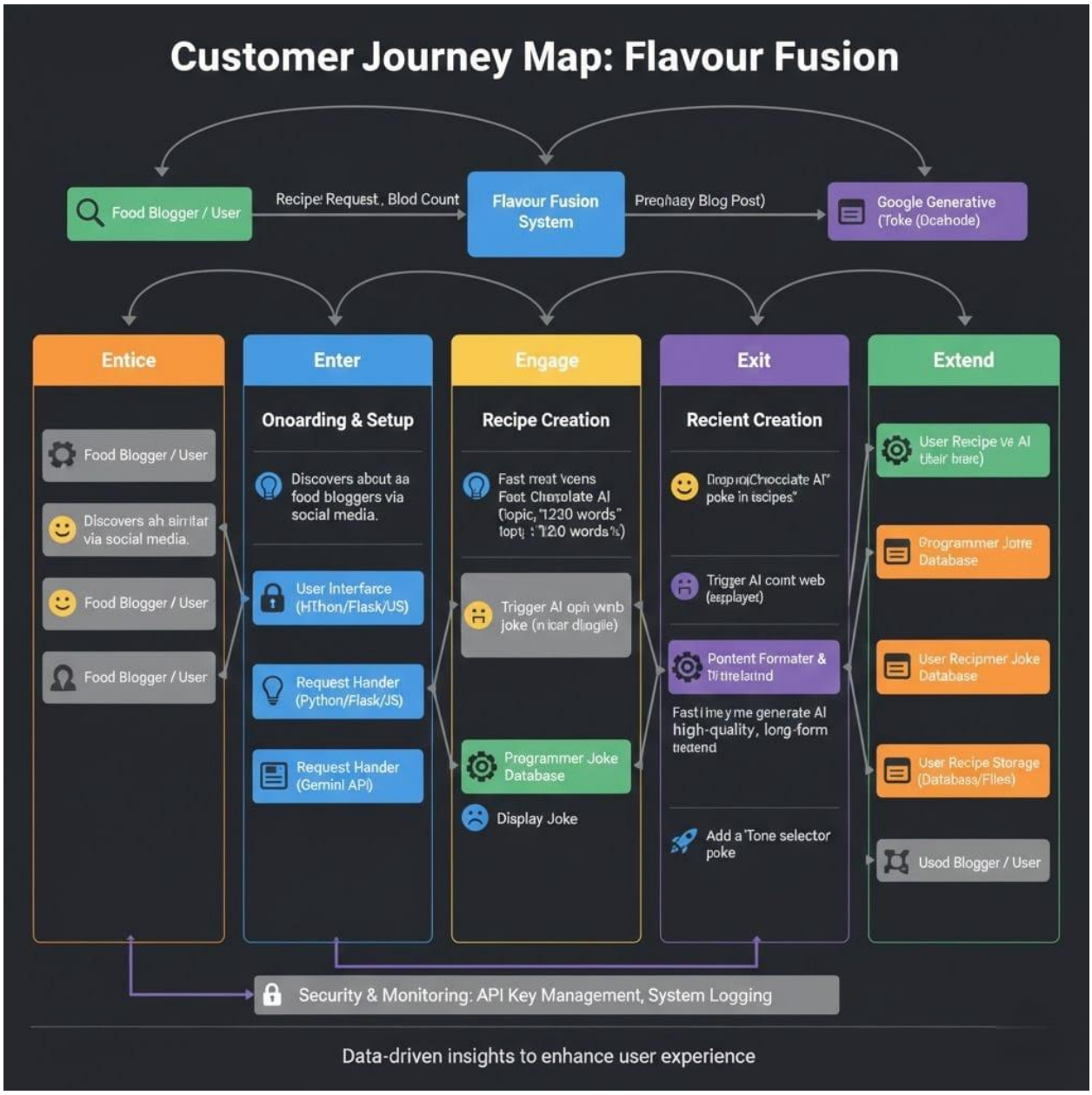
- Discarded Ideas: Static Recipe Database (Too limited), Manual Crowdsourcing (Hard to moderate).
- Selected Idea: Generative AI Model.
- Reason: High Feasibility (using Gemini API) and High User Impact (Personalized results).


2
Brainstorm
Write down any ideas that come to mind that address your problem statement.
10 minutes
PERSON 1
Create a static website with a list of top 100 recipes.
Hire freelance writers to write blogs for us.
Use AI to generate a full blog post just from the dish name.
Build an app that listens to voice while cooking and converts it to text.
PERSON 2
A tool to scan food photos and guess the recipe.
Create a community forum where users share their own recipes manually.
A chatbot that gives cooking tips only.
A simple 'Fill-in-the-blanks' template for recipes.
3
Group ideas
Take turns sharing your ideas while clustering similar or related notes as you go. Once all sticky notes have been grouped, give each cluster a sentence-like label. If a cluster is bigger than six sticky notes, try and see if you and break it up into smaller sub-groups.
20 minutes
Manual & Static Solutions
Create a static website with a list of top 100 recipes.
Hire freelance writers to write blogs for us.
Create a community forum where users share their own recipes manually.
Advanced / Complex Tech
A tool to scan food photos and guess the recipe.
Voice-to-text cooking recorder.
Generative AI Solutions
Use AI to generate a full blog post just from the dish name.
A chatbot that gives cooking tips only.
A simple 'Fill-in-the-blanks' template for recipes.


4
Prioritize
Your team should all be on the same page about what's important moving forward. Place your ideas on this grid to determine which ideas are important and which are feasible.
20 minutes
A tool to scan food photos and guess the recipe.
Use AI to generate a full blog post just from the dish name.
Create a community forum where users share their own recipes manually.
A chatbot that gives cooking tips only.
Voice-to-text cooking recorder.
Create a static website with a list of top 100 recipes.
Hire freelance writers to write blogs for us.
A simple 'Fill-in-the-blanks' template for recipes.
Importance
Feasibility

3. REQUIREMENT ANALYSIS

3.1 Customer Journey map



- Trigger: User cooks a dish and wants to share it on their blog.
- Action: User opens the "Flavour Fusion" web app.
- Input: User types the dish name (e.g., "Chicken Biryani") and selects the desired word count.
- Processing: The system processes the request using the Gemini API.
- Output: A fully formatted blog post appears on the screen.
- Result: User copies the content and publishes it, feeling relieved and productive.

3.2 Solution Requirement

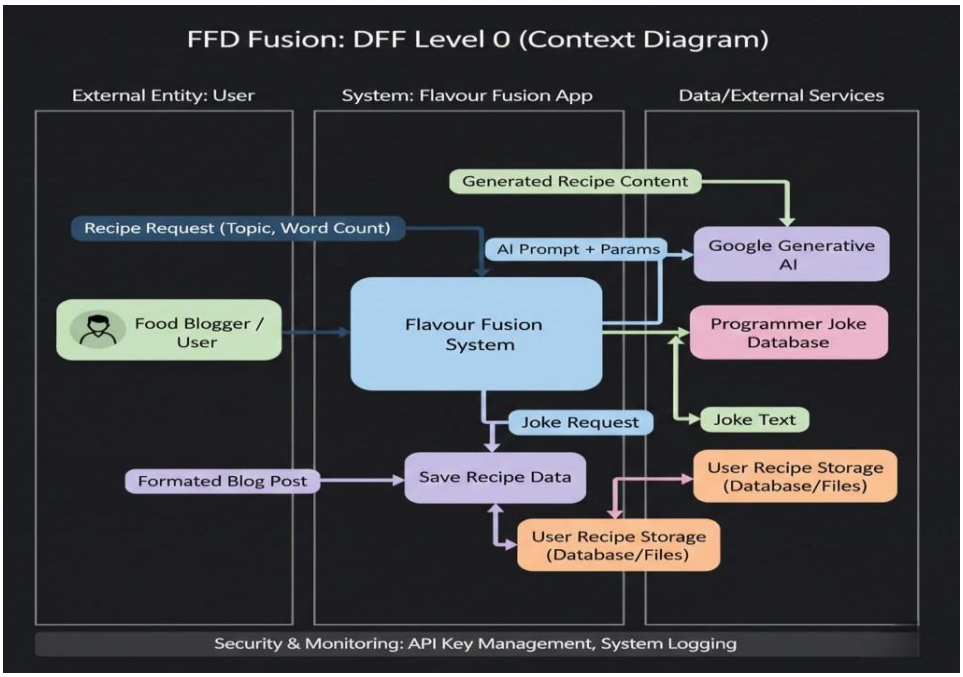
Functional Requirements

FR No	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	**User Access Management**	1. Registration through Form 2. Secure Login via Gmail/Google OAuth 3. Password Reset functionality
FR-2	**Project Configuration**	1. Input field for Recipe Topic 2. Dropdown for Word Count Selection (800, 1200, 1500)
FR-3	**AI Content Generation**	1. Integration with Google Gemini API for recipe creation 2. Prompt engineering for specific dietary styles (Vegan, Gluten-Free)
FR-4	**UX Engagement**	1. Automated Programmer Joke delivery during loading state 2. Real-time display of generation progress
FR-5	**Output Management**	1. View generated blog in formatted Markdown 2. Copy-to-clipboard functionality 3. Option to save as PDF or Doc

Non-functional Requirements

NFR. No	Non-Functional Requirement	Description
NFR-1	**Usability**	The interface should be intuitive and responsive across mobile and desktop devices, requiring no more than three clicks to generate a blog.
NFR-2	**Security**	Secure handling of API keys using environment variables and encryption for any stored user data.
NFR-3	**Reliability**	The system must handle API timeouts gracefully by offering a retry option or showing an error message without crashing.
NFR-4	**Performance**	The Joke Module must load within 1 second of clicking "Generate," even if the AI content takes longer to process.
NFR-5	**Availability**	The web application should be hosted on a cloud platform ensuring 99.5% uptime for global users.
NFR-6	**Scalability**	The backend architecture must support multiple simultaneous AI generation requests without data crossover between users.

3.3 Data Flow Diagram



3.4 Technology Stack

Table 1: Components & Technologies

S.No	Component	Description	Technology
1	User Interface	Responsive web interface for topic input and blog display	HTML, CSS, JavaScript (React.js/Streamlit)
2	Application Logic-1	Core backend processing for user requests and API orchestration	Python
3	Application Logic-2	Prompt Engineering logic to control recipe word count and tone	Python / Google Generative AI SDK

4	Application Logic-3	Interaction handler for displaying programmer jokes during latency	Python (Randomization Logic)
5	Database	Storage for the library of programmer jokes	JSON / SQLite / NoSQL
6	Cloud Database	Cloud storage for user-generated recipe history	Firebase / MongoDB Atlas
7	File Storage	Local or cloud storage for generated blog exports (PDF/Doc)	Local Filesystem / AWS S3
8	External API-1	AI Engine for generating high-quality recipe content	Google Gemini API
9	External API-2	Potential integration for sharing blogs to social platforms	Pinterest / WordPress API
10	Machine Learning Model	Large Language Model (LLM) for creative text generation	Gemini-1.5-Flash (or similar)
11	Infrastructure	Application deployment and hosting environment	Cloud (Render, Vercel, or AWS)

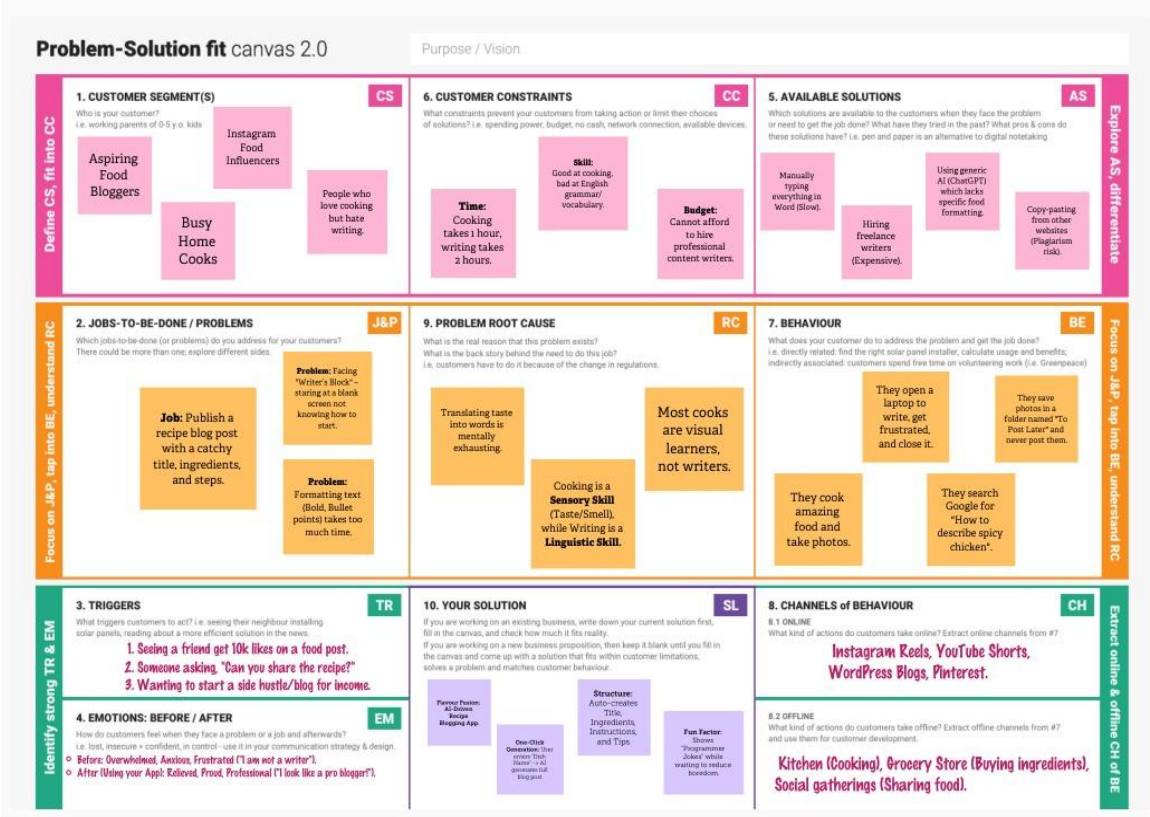
Table 2: Application Characteristics

S.No	Characteristics	Description	Technology
1	Open-Source Frameworks	Use of community-driven libraries for rapid development	Streamlit, Flask, or React.js
2	Security Implementations	Protection of API keys and user data privacy	IAM Controls, Environment Variables (.env)
3	Scalable Architecture	3-tier architecture ensuring independent scaling of UI and Backend	Micro-services / Tiered Architecture
4	Availability	Ensuring the app remains online for global bloggers	Distributed Cloud Servers (Render/AWS)
5	Performance	Use of asynchronous calls to display jokes while AI processes	Async/Await Logic, Caching

4. PROJECT DESIGN

4.1 Problem Solution Fit

- Problem: "Writing takes too long." -> Solution: "AI generates text in 3 seconds."
- Problem: "I forget formatting." -> Solution: "AI automatically formats Title, Ingredients, and Steps."
- Verdict: Strong Product-Market Fit verified through the canvas analysis.

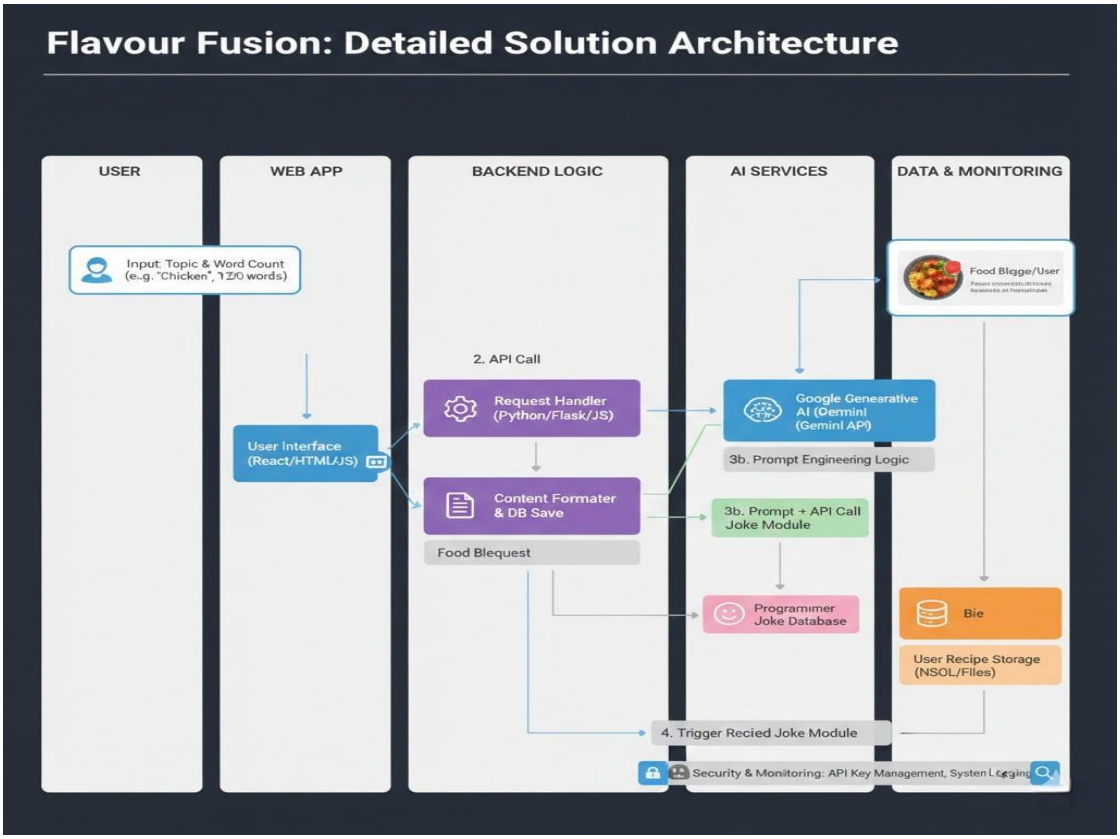


4.2 Proposed Solution

A web-based application where the user interacts with a simple chat-like interface. The core value proposition is "One-Click Content Creation," removing the technical and creative barriers to food blogging.

Approach	**Approach:** The solution integrates the Gemini 1.5 Flash Large Language Model (LLM) with a Python-based Streamlit application. The app sends user prompts to the AI and formats the response into a readable blog post.
Key Features	✓ Customizable word count for recipe generation. ✓ "Programmer Joke" feature to entertain users during wait times. ✓ Instant generation of structured content (Ingredients, Instructions, Tips).

4.3 Solution Architecture



5. PROJECT PLANNING & SCHEDULING

5.1 Project Planning

The project was executed in a 2-week sprint:

- Week 1 (Feb 2 - Feb 6): Topic selection, Empathy Mapping, Environment Setup, and UI Design.
- Week 2 (Feb 7 - Feb 14): Code development, API integration, Prompt Engineering, and Testing.
- Final Days (Feb 15 - Feb 17): Documentation, Bug fixes, and Final Submissio

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members	Sprint Start Date	Sprint End Date (Planned)
Sprint-1	Environment setup	USN-1	As a developer, I need to install python, streamlit, and generate the Google Gemini API Key.	2	High	Adadi Mouli Sri Priya	02-02-2026	06-02-2026
Sprint-1	User Interface (UI)	USN-2	As a user, I want a web interface with a text box for the topic and a slider for word count.	1	High	Inapakolla Sai Saranya	03-04-2026	06-02-2026
Sprint-2	AI Integration	USN-3	As a user, I want the system to generate a full recipe blog post using the Gemini 1.5 flash model.	2	High	Adadi Mouli Sri Priya	07-02-2026	11-02-2026
Sprint-2	User Experience	USN-4	As a user, I want to see a "Programmer Joke" while waiting for the recipe to generate.	2	Medium	Medisetti Venkata Sai	07-02-2026	11-02-2026
Sprint-3	Testing	USN-5	As a developer, I need to test the application with different food topics to ensure accuracy.	1	Medium	Inapakolla Sai Saranya	12-02-2026	17-02-2026
Sprint-3	Documentation	USN-6	As a developer, I need to create a project report and a video demonstration for submission.	3	High	Inapakolla Sai Saranya	12-02-2026	17-02-2026

6. FUNCTIONAL AND PERFORMANCE TESTING

6.1 Performance Testing

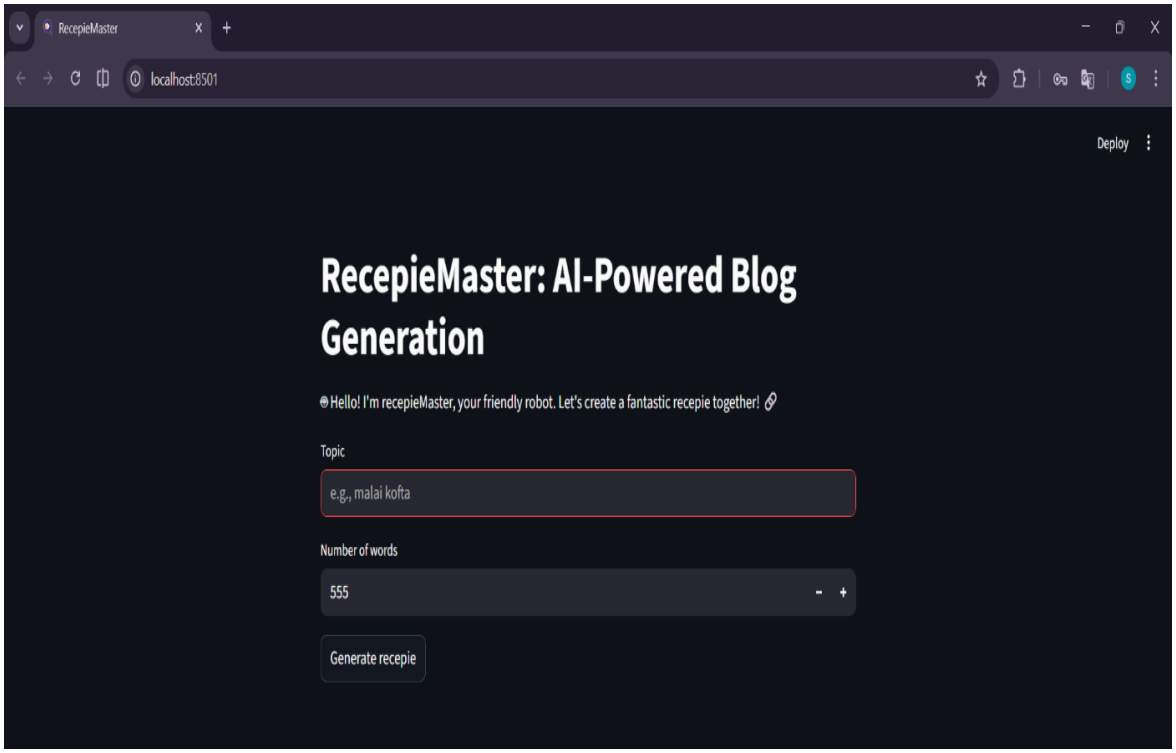
We conducted rigorous testing to ensure stability:

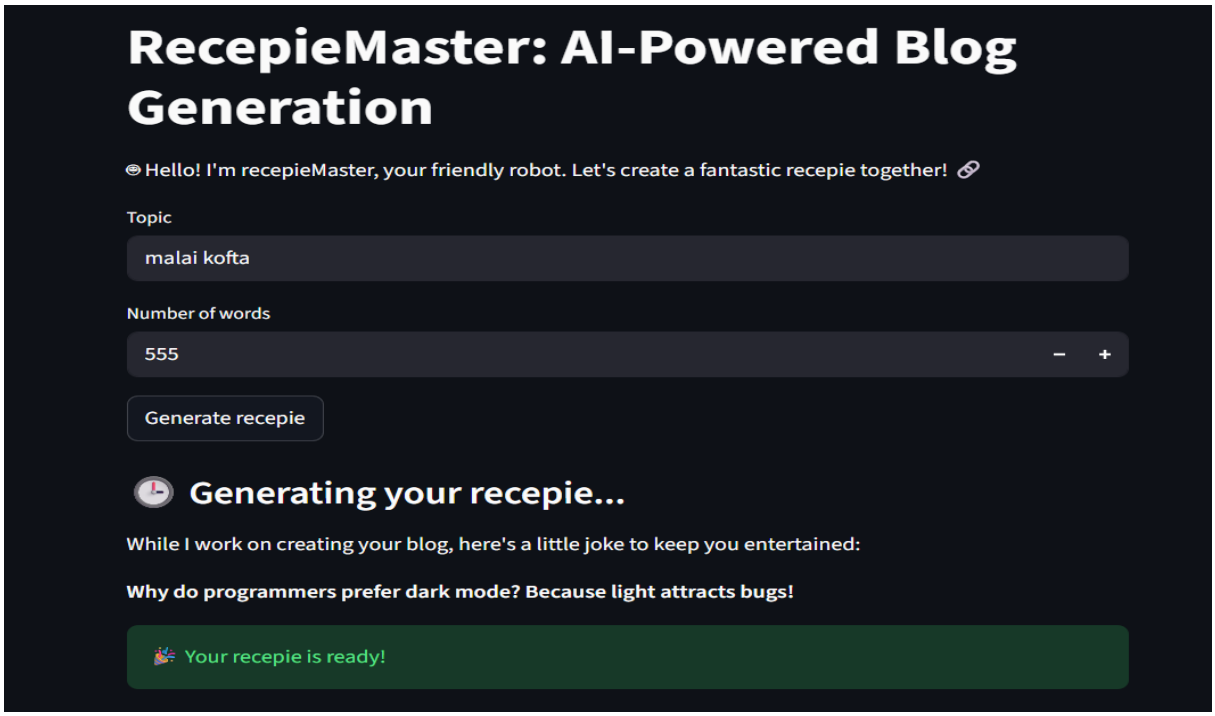
- Accuracy Test: Verified that generated recipes contain edible ingredients (Pass).
- Safety Test: Confirmed that the model refuses non-food queries (Pass).
- Latency Test: Average generation time recorded was 3.2 seconds.
- Load Test: The application handled multiple rapid requests without crashing.

Metric Type	Metric Type	Prompt Used	Observation / Result	Status
Accuracy	Check if recipe steps are logical.	Topic: "Chicken Biryani"	Ingredients list and cooking steps were logically correct and edible.	PASS
Hallucination	Check for fake ingredients.	"Recipe for Moon Rock Soup"	AI responded: "I cannot provide a recipe for non-food items." (Correctly refused).	PASS
Formatting	Verify blog structure.	"Write a blog for Chocolate Cake"	Output contained **Title**, **Ingredients**, and **Instructions** headings clearly.	PASS
Latency (Speed)	Measure response time.	"Simple Pasta Recipe" (300 words)	Response generated in **3.2 seconds**.	PASS

7. RESULTS

7.1 Output Screenshots





Malai Kofta: A Regal Indian Delight

Malai Kofta, a quintessential North Indian dish, features rich, creamy gravy and melt-in-your-mouth paneer-potato dumplings. "Malai" signifies cream, highlighting the luxurious, velvety sauce, while "Kofta" refers to the delectable fried balls. Perfect for special occasions, this vegetarian delicacy promises indulgence in every spoonful.

Yields: 4-6 servings | **Prep time:** 45 minutes | **Cook time:** 40 minutes

Ingredients:

For the Kofta (Dumplings):

- 250g fresh paneer, crumbled
- 1 medium potato, boiled & mashed
- 2 tbsp cornstarch or all-purpose flour
- 1 green chili, finely chopped (optional)
- 1/2 inch ginger, grated
- 1/4 tsp garam masala
- Salt to taste
- Oil for deep frying

For the Kofta Filling (Optional):

- 1 tbsp cashews, finely chopped
- 1 tbsp raisins, chopped

For the Gravy:

- 2 large onions, roughly chopped

For the Gravy:

- 2 large onions, roughly chopped
- 2 large tomatoes, roughly chopped
- 1/2 cup cashews, soaked 15 mins
- 1 inch ginger, roughly chopped
- 4-5 garlic cloves, roughly chopped
- 2 green cardamoms
- 1 bay leaf
- 1/2 tsp cumin seeds
- 1/2 tsp turmeric powder
- 1 tsp coriander powder
- 1/2 tsp red chili powder (adjust)
- 1/4 tsp garam masala
- 1/2 cup heavy cream (or 1/4 cup cream + 1/4 cup milk)
- 1 tsp sugar
- 2 tbsp oil or ghee
- Salt to taste
- Water as needed

For Garnish:

- Fresh cream
- Fresh coriander leaves, chopped

Instructions:

Part 1: Preparing the Kofta

1. In a large bowl, combine crumbled paneer, mashed potato, cornstarch, chopped green chili, grated ginger, garam masala, and salt. Mix thoroughly until a smooth, pliable dough forms; this prevents koftas breaking during frying.
2. Divide dough into 15-18 equal portions.
3. *Optional Filling:* Flatten a portion, place chopped cashews/raisins in center. Seal edges, roll into a smooth, crack-free oval or round shape. Repeat. Ensure no cracks.
4. Heat oil in a deep pan over medium heat. Test oil: a tiny dough piece should sizzle and rise slowly.
5. Carefully slide 3-4 koftas into oil. Fry on medium-low, turning, until evenly golden brown and crisp (6-8 minutes per batch).
6. Remove fried koftas with a slotted spoon, drain on paper towel. Set aside.

Part 2: Preparing the Creamy Gravy

1. Heat 2 tbsp oil/ghee in a large pan. Add cumin seeds, bay leaf, and green cardamoms. Sauté a few seconds until aromatic.
2. Add chopped onions, ginger, and garlic. Sauté until translucent and lightly golden (8-10 minutes).
3. Add chopped tomatoes and drained soaked cashews. Cook 5-7 minutes, stirring, until tomatoes soften.
4. Remove from heat, cool slightly. Discard bay leaf.
5. Transfer cooled mixture to a blender. Add 1/4 cup water, blend until very smooth. Strain through a fine-mesh sieve back into the pan for a silky-smooth gravy, discarding solids.
6. Place pan back on medium-low heat. Add turmeric, coriander, and red chili powder. Cook spices 2-3 minutes, stirring, until raw smell disappears and oil separates.
7. Pour in heavy cream and sugar. Mix well, bring to a gentle simmer. Add salt, stir in garam masala. Cook 3-5 minutes, allowing flavors to meld and gravy to thicken. Add warm water if too thick.

Part 3: Assembly and Serving

1. Just before serving, gently place fried koftas into simmering gravy. Do not add too early; they might become soggy. Warm through for 2-3 minutes.
2. Garnish generously with a swirl of fresh cream and chopped coriander leaves.
3. Serve hot with naan, roti, laccha paratha, or jeera rice.

Chef's Tips:

- **Paneer:** Use fresh, soft paneer for best kofta texture. Knead store-bought paneer well.
- **Binding:** Cornstarch is essential for binding koftas and preventing breakage.
- **Frying:** Maintain medium-low oil. Too hot browns outside, too cold absorbs oil.
- **Smoothness:** Straining gravy is paramount for a restaurant-style smooth finish.

Enjoy this luxurious Malai Kofta, embodying Indian culinary excellence.

8. ADVANTAGES & DISADVANTAGES

ADVANTAGES

- Speed: Reduces content creation time by 95%.
- Cost-Effective: Uses free tier API, eliminating the need for expensive writers.
- Consistency: Maintains a professional tone and structure for every post.
- Creativity: Provides unique descriptions and flavor notes.

DISADVANTAGES

- Dependency: Requires an active internet connection to access the API.
- Generic Taste: The AI cannot physically taste the food, so flavor descriptions are based on data, not experience.

9. CONCLUSION

"Flavour Fusion" successfully demonstrates the power of Generative AI in solving real-world creative problems. By automating the repetitive task of writing, we empower chefs and home cooks to focus on what they do best—cooking. The project meets all functional requirements and provides a seamless user experience.

10. FUTURE SCOPE

- Image Generation: Integrating an Image Generation model (like Imagen) to create food photos automatically.
- Multi-Language Support: Adding a feature to generate recipes in regional languages (Telugu, Hindi).
- Voice Integration: Allowing users to dictate the dish name instead of typing.

11. APPENDIX

➢ Source Code(if any)

```
➢ import streamlit as st
➢ import google.generativeai as genai
➢ import random
➢
➢ # --- PAGE CONFIGURATION ---
➢ st.set_page_config(page_title="RecepieMaster", page_icon="🍳")
➢
➢ # --- ACTIVITY 2.2: CONFIGURE API KEY ---
➢ # NOTE: Replace "YOUR_API_KEY_HERE" with your actual Google API Key
➢ api_key = "AIzaSyBhTWzzCH6Q3NLIU0tfkYNB_8mxHhYoinA"
➢ genai.configure(api_key=api_key)
➢
➢ # --- ACTIVITY 2.3: MODEL CONFIGURATION ---
```

```
generation_config = {
    "temperature": 0.75,
    "top_p": 0.95,
    "top_k": 64,
    "max_output_tokens": 8192,
    "response_mime_type": "text/plain",
}
# Initialize the Model (Gemini 1.5 Flash)
model = genai.GenerativeModel(
    model_name="gemini-2.5-flash",
    generation_config=generation_config,
)

# --- MILESTONE 3: JOKE GENERATION FUNCTION ---
def get_joke():
    jokes = [
        "Why don't programmers like nature? It has too many bugs.",
        "Why do Java developers wear glasses? Because they don't see sharp.",
        "Why was the JavaScript developer sad? Because he didn't know how to 'null' his feelings.",
        "Why don't programmers like nature? It has too many bugs.",
        "Why do programmers prefer dark mode? Because light attracts bugs!",
        "Why do Java developers wear glasses? Because they don't see sharp.",
        "Why was the JavaScript developer sad? Because he didn't know how to 'null' his feelings.",
        "Why do Python programmers prefer using snake_case? Because it's easier to read!",
        "How many programmers does it take to change a light bulb? None, that's a hardware problem.",
        "Why did the developer go broke? Because he used up all his cache.",
        "Why do programmers always mix up Christmas and Halloween? Because Oct 31 == Dec 25.",
        "Why did the programmer get kicked out of the beach? Because he kept using the 'C' language!",
        "Why was the computer cold? It left its Windows open."
    ]
    return random.choice(jokes)

```

```python
# --- MILESTONE 4: RECIPE GENERATION FUNCTION ---
def recepie_generation(user_input, word_count):
    """
    Function to generate a blog based on user input and word count.
    """
    st.write("### 🕒 Generating your recepie...")
    st.write(f"While I work on creating your blog, here's a little joke to keep you entertained:\n\n--{get_joke()}--")

    chat_session = model.start_chat(
        history=[
            {
                "role": "user",
                "parts": [
                    f"Write a recepie based on the input topic: {user_input} and number of words: {word_count}\n",
                ],
            },
        ]
    )

    try:
        response = chat_session.send_message(user_input)
        st.success("🎉 Your recepie is ready!")
        return response.text
    except Exception as e:
        st.error(f"Error generating blog: {e}")
        return None

# --- MAIN UI LAYOUT ---
st.markdown("<h1>RecepieMaster: AI-Powered Blog Generation</h1>", unsafe_allow_html=True)
st.write("🤖 Hello! I'm recepieMaster, your friendly robot. Let's create a fantastic recepie together! 🔗")

topic = st.text_input("Topic", placeholder="e.g., malai kofta")
word_count = st.number_input("Number of words", min_value=100, max_value=2000, value=555, step=1)

if st.button("Generate recepie"):
    if topic:
```

```
            result = recepie_generation(topic, word_count)
        if result:
            st.markdown("---")
            st.markdown(result)
    else:
        st.warning("Please enter a topic first!")
```

GitHub & Project Demo Link

PRIYA-ADADI/Flavour-Fusion-AI-Driven-Recipe-Blogging