

FINAL REPORT

FLAVOUR FUSION: AI-Driven Recipe Blogging

1. INTRODUCTION

1.1 Project Overview

Flavor Fusion is a web-based Generative AI application designed to automate the creation of structured recipe blog posts. The system leverages Google's Gemini (gemini-flash-latest) model to generate complete blog-style recipes based on user input.

The application allows users to:

- Enter a recipe topic
- Select desired word count (100–2000 words)
- Generate a structured recipe blog
- Download the output as a Markdown (.md) file

The solution is implemented using Streamlit for the frontend interface and Python for application logic, with Gemini API integration for AI-driven content generation.

1.2 Purpose

The purpose of this project is to:

- Reduce the time required to write structured recipe blogs
- Provide AI-assisted content generation for food enthusiasts and bloggers
- Demonstrate practical integration of Generative AI into a real-world web application
- Deliver a lightweight, scalable, and deployable AI-based system

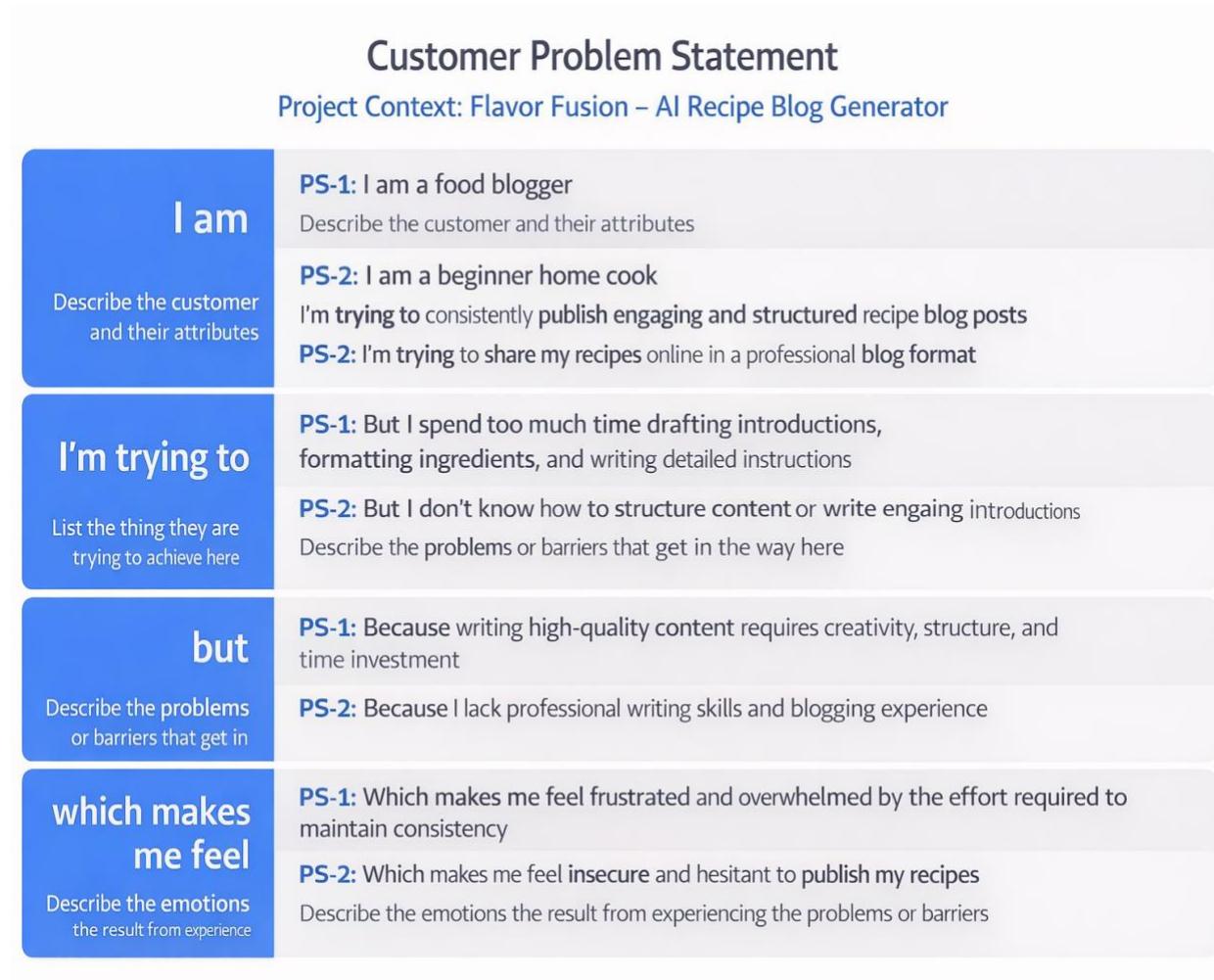
2. IDEATION PHASE

2.1 Problem Statement

Food bloggers and home cooks spend significant time writing structured recipe blog posts. Drafting engaging introductions, organizing ingredients, and formatting instructions requires writing skills and effort.

Traditional recipe websites provide static content and do not assist users in generating customized, blog-ready content instantly.

Flavor Fusion addresses this gap by providing AI-powered automated recipe blog generation.



2.2 Empathy Map Canvas

Target User

Aspiring food blogger or home cook with limited writing time.

Says

- “Writing takes more time than cooking.”
- “I want professional-looking blogs.”

Thinks

- “My content isn’t engaging enough.”
- “I need to post consistently.”

Does

- Searches online for blog format examples
- Spends hours formatting content

Feels

- Frustrated by writing effort
- Insecure about content quality

Gains Expected

- Faster content creation
- Structured blog output
- Increased publishing confidence

2.3 Brainstorming

During ideation, the team evaluated multiple AI-based content generation ideas including:

- Travel blog generator
- Fitness plan generator
- Academic summary tool
- AI recipe generator

The AI recipe blog generator was selected due to:

- Clear user problem
- Feasible implementation

- Strong demonstration of Generative AI capability
- Practical real-world relevance

Features were prioritized based on impact and development effort.

Brainstorm & idea prioritization

Use this template in your own brainstorming sessions so your team can unleash their imagination and start shaping concepts even if you're not sitting in the same room.

⌚ 10 minutes to prepare
⌚ 1 hour to collaborate
👤 2-8 people recommended

Before you collaborate
A little bit of preparation goes a long way with this session. Here's what you need to do to get going.
⌚ 10 minutes

1 Define your problem statement
What problem are you trying to solve? Frame your problem as a How Might We statement. This will be the focus of your brainstorm.
⌚ 5 minutes

A Team gathering
Define who should participate in the session and send an invite. Share relevant information or pre-work ahead.

B Set the goal
Think about the problem you'll be focusing on solving in the brainstorming session.

C Learn how to use the facilitation tools
Use the Facilitation Superpowers to run a happy and productive session.
[Open article](#)

PROBLEM
How might we [your problem statement]?

Key rules of brainstorming
To run a smooth and productive session

- Stay in topic.
- Encourage wild ideas.
- Defer judgment.
- Listen to others.
- Go for volume.
- If possible, be visual.

3. REQUIREMENT ANALYSIS

3.1 Customer Journey Map

1. User opens web application
2. Enters recipe topic
3. Selects word count
4. Clicks “Generate Recipe”
5. Views AI-generated blog
6. Downloads Markdown file

Pain Point Addressed:

Manual content drafting replaced with instant AI generation.

3.2 Solution Requirement

Functional Requirements

- Accept recipe topic input
- Accept word count selection
- Generate structured recipe blog via Gemini API
- Display generated content
- Provide Markdown download
- Show loading feedback
- Handle API errors gracefully

Non-Functional Requirements

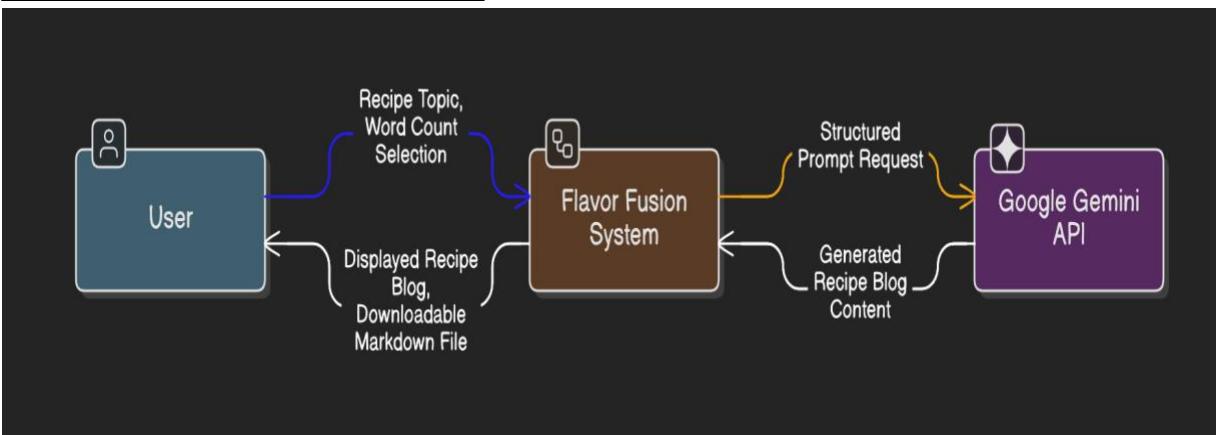
- Usability: Simple and intuitive UI
- Performance: Generation within acceptable API latency
- Reliability: No system crash during API failure
- Security: API key stored securely
- Scalability: Cloud deployable architecture

3.3 Data Flow Diagram (Description)

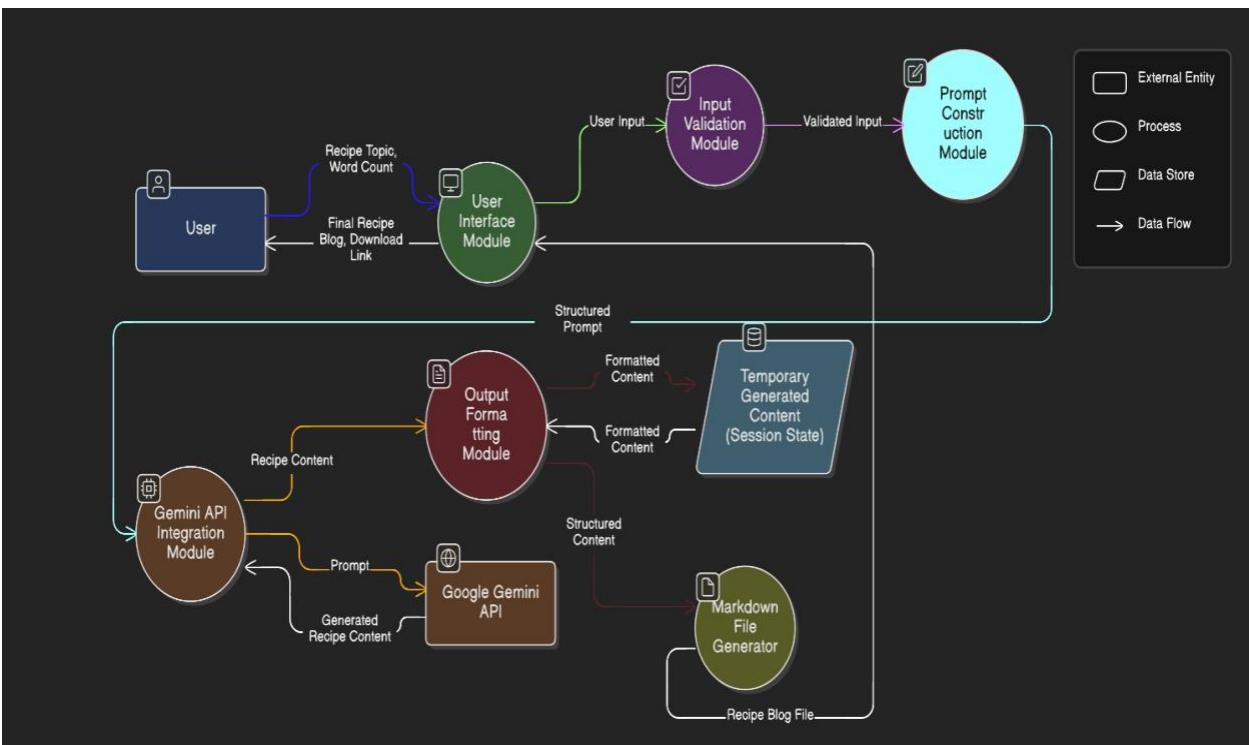
User → Streamlit UI → Application Logic → Gemini API

Gemini API → Application Logic → Content Formatter → Markdown Generator → User

DFD Level 0 (Industry Standard)



DFD Level 1 (Industry Standard)



The system follows a stateless architecture with no persistent database.

3.4 Technology Stack

Frontend: Streamlit

Backend: Python

AI Model: Google Gemini (gemini-flash-latest)

File Format: Markdown (.md)

Deployment: Local / Streamlit Cloud

4. PROJECT DESIGN

4.1 Problem-Solution Fit

Problem:

Time-consuming manual recipe blog writing.

Solution:

AI-powered automated blog generation with structured formatting.

Fit Justification:

The solution directly reduces writing effort while maintaining professional output quality.

Define CS, fit into CC	1. CUSTOMER SEGMENT(S) Who is your customer? I.e. working parents of 0-5 y.o. kids	CS	6. CUSTOMER CONSTRAINTS What constraints prevent your customers from taking action or limit their choices of solutions? I.e. spending power, budget, no cash, network connection, available devices.	CC	5. AVAILABLE SOLUTIONS Which solutions are available to the customers when they face the problem or need to get the job done? What have they tried in the past? What pros & cons do these solutions have? I.e. pen and paper is an alternative to digital notetaking	AS	Explore AS, differentiate
	2. JOBS-TO-BE-DONE / PROBLEMS Which jobs-to-be-done (or problems) do you address for your customers? There could be more than one; explore different sides.	J&P	9. PROBLEM ROOT CAUSE What is the real reason that this problem exists? What is the back story behind the need to do this job? I.e. customers have to do it because of the change in regulations.	RC	7. BEHAVIOUR What does your customer do to address the problem and get the job done? I.e. directly related: find the right solar panel installer, calculate usage and benefit; indirectly associated: customers spend free time on volunteering work (I.e. Greenpeace)	BE	Focus on J&P, tap into RC, understand RC
Identify strong TR & EM	3. TRIGGERS What triggers customers to act? I.e. seeing their neighbour installing solar panels, reading about a more efficient solution in the news.	TR	10. YOUR SOLUTION If you are working on an existing business, write down your current solution first, fill in the canvas, and check how much it fits reality. If you are working on a new business proposition, then keep it blank until you fill in the canvas and come up with a solution that fits within customer limitations, solves a problem and matches customer behaviour.	SL	8. CHANNELS OF BEHAVIOUR 8.1 ONLINE What kind of actions do customers take online? Extract online channels from #7	CH	Focus on TR, tap into BE, understand RC
	4. EMOTIONS: BEFORE / AFTER How do customers feel when they face a problem or a job and afterwards? I.e. lost, Insecure > confident, in control - use it in your communication strategy & design.	EM			8.2 OFFLINE What kind of actions do customers take offline? Extract offline channels from #7 and use them for customer development.	CH	Extract online & offline CH of BE

4.2 Proposed Solution

Flavor Fusion is a lightweight AI-powered web application that:

- Dynamically generates structured recipe blogs

- Provides customizable word length
- Offers export-ready Markdown files
- Enhances user engagement via interactive UI

The system bridges business need (content automation) with AI technology.

4.3 Solution Architecture

The architecture consists of four layers:

Business Layer:

User need for faster blog creation.

Application Layer:

Streamlit UI + Python logic modules.

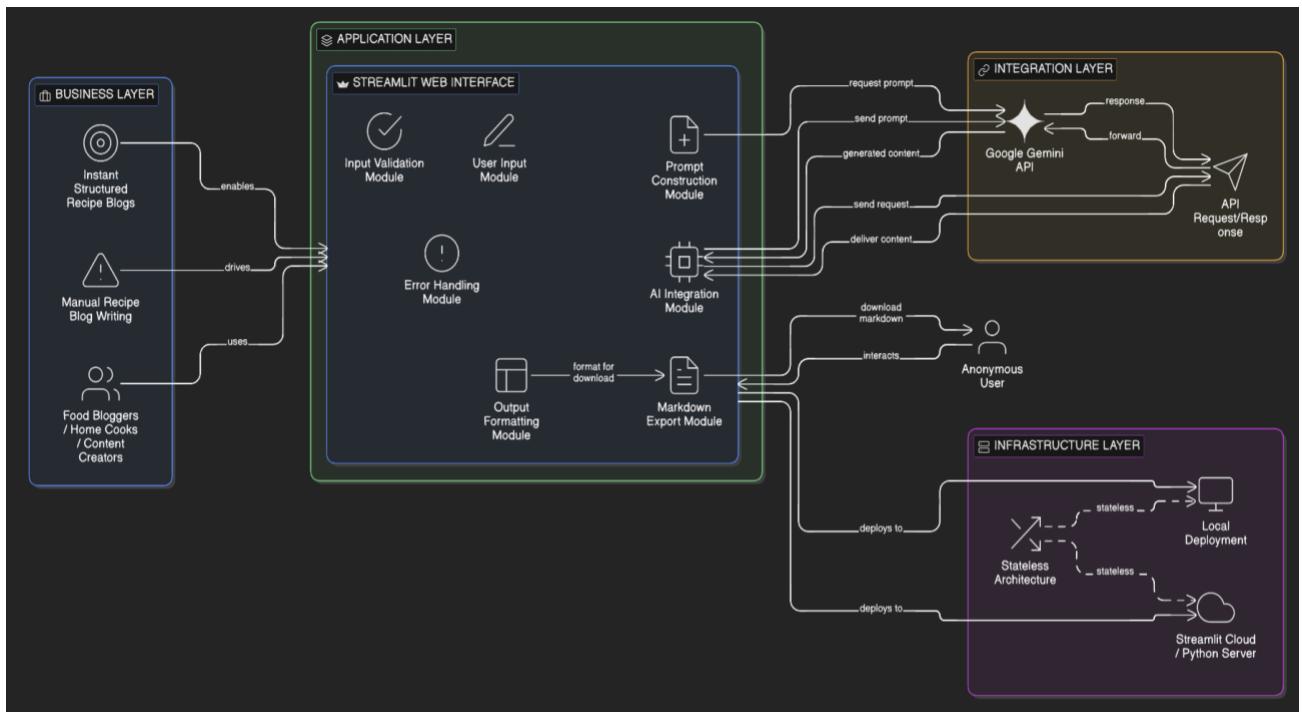
Integration Layer:

Google Gemini API.

Infrastructure Layer:

Local or cloud deployment environment.

The application is stateless and scalable through horizontal cloud deployment.



5. PROJECT PLANNING & SCHEDULING

5.1 Project Planning

Development was completed in two sprints:

Sprint 1:

- UI development
- Gemini API integration
- Prompt engineering

Sprint 2:

- Markdown export
- Loading indicator and joke display
- Error handling
- Testing and deployment

Average team velocity: 19 Story Points per sprint.

6. FUNCTIONAL AND PERFORMANCE TESTING

6.1 Performance Testing

- AI generation time tested under multiple word count conditions.
- Short output (~200 words): 3–5 seconds.
- Long output (~1500 words): 5–8 seconds depending on API latency.
- No application crash during API timeout simulation.

System confirmed stable for UAT.

7. RESULTS

7.1 Output Screenshots

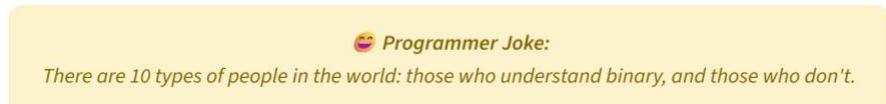
- Main interface screen

The screenshot shows the main interface of the Flavor Fusion application. At the top, there's a logo featuring a stylized face with a spoon and fork, followed by the text "Flavor Fusion". Below that is a red horizontal bar with the text "AI-Driven Recipe Blogging". The main area has two sections: "Configuration" on the left and "Model Settings" on the right. The "Configuration" section contains fields for "Recipe Topic" (with placeholder text "e.g., Vegan Chocolate Cake") and "Desired Word Count" (set to 500). It also features a "Ready to cook?" button with a thumbs-up icon and a note: "Enter a topic in the sidebar and hit Generate!". Below these are four food-related icons: a bowl of soup, a slice of pizza, a slice of cake, and a taco. The "Model Settings" section includes a note: "Using Gemini Flash for fast generation." and a prominent red "Generate Recipe" button with a star icon.

- Input section

This screenshot shows the "Configuration" section of the Flavor Fusion interface. It features a large input field with the placeholder text "Customize your recipe request below." Below this is a "Recipe Topic" input field with the placeholder "e.g., Vegan Chocolate Cake". Underneath is a "Desired Word Count" slider set to 500, with scale marks at 100 and 2000. The "Model Settings" section is visible at the bottom, stating "Using Gemini Flash for fast generation." and containing a red "Generate Recipe" button with a star icon.

- Waiting Time



🕒 🍽️ Chef AI is cooking up your blog post...

- Generated recipe output

🔍 Configuration

Customize your recipe request below.

📍 Recipe Topic
Spicy Noodles

📏 Desired Word Count
1050

⌚ Model Settings

Using Gemini Flash for fast generation.

👉 Generate Recipe

The Ultimate 15-Minute Chili Garlic Slap Noodles: A Spicy Lover's Dream
There is a specific kind of hunger that only a bowl of glistening, fiery-red noodles can satisfy. You know the one—it usually hits around 9:00 PM on a Tuesday, or perhaps right after a long, rainy commute when your soul feels a little damp and your taste buds feel a little bored. It's a craving for something that is simultaneously comforting and confrontational. You want the chew of the dough, the richness of the oil, and that unmistakable "slap" of heat that makes your brow sweat just a little bit.

Welcome to my kitchen, fellow spice seekers. Today, we aren't just making dinner; we are crafting an experience. These Spicy Chili Garlic Noodles are my go-to "emergency" meal. They take less time to make than scrolling through a delivery app, and the payoff is infinitely better. We're talking about velvety strands of noodles coated in a complex, umami-packed sauce that balances heat, acidity, and a touch of sweetness.

Grab your chopsticks, put on an apron, and let's dive into the glorious world of the "Sizzle and Slurp."

The Philosophy of the Perfect Spicy Noodle

Before we get to the stovetop, let's talk about what makes a spicy noodle dish truly legendary. It's not just about dumping a bottle of hot sauce onto pasta. A great spicy noodle dish is a symphony of layers.

- Markdown download confirmation

The Verdict

There is something incredibly meditative about tossing a bowl of noodles. Watching the pale dough transform into a vibrant, spicy masterpiece is half the fun. When you take that first bite, you'll get the punch of the garlic, the tang of the vinegar, and then—slowly—the warmth of the chili will bloom across your palate.

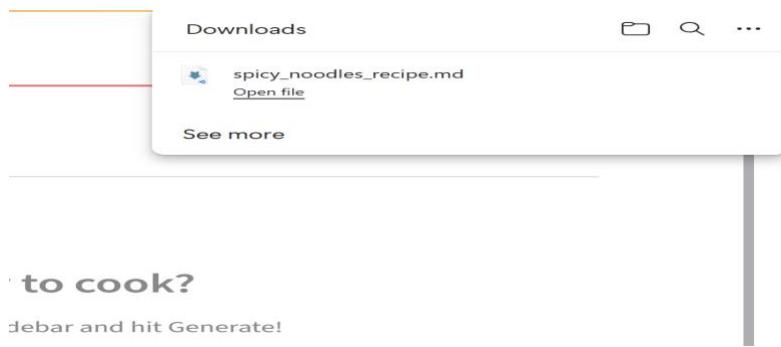
This recipe is more than just a quick dinner; it's a mood-lifter. It's the kind of meal that clears your sinuses, warms your soul, and leaves you feeling completely satisfied.

So, the next time the "noodle craving" strikes, don't reach for the delivery app. Reach for your garlic press and your chili flakes. Your taste buds will thank you.

Did you make this recipe? I want to see your fiery creations! Tag me on Instagram or leave a comment below letting me know how much heat you handled. Happy slurping!

🎉 Recipe generated successfully!

📥 Download Recipe



Generated outputs demonstrated:

- Structured blog format
- Correct word count range
- Professional presentation

8. ADVANTAGES & DISADVANTAGES

Advantages

- Significant time savings
- Structured professional output
- Lightweight architecture
- Easy deployment
- Beginner-friendly UI

Disadvantages

- Dependent on external AI API
- Requires internet connection
- API latency may vary
- No persistent user storage

9. CONCLUSION

Flavor Fusion successfully demonstrates practical integration of Generative AI into a real-world web application.

The system reduces manual effort in content creation while maintaining structured output quality. It validates the feasibility of AI-driven automation for creative blogging tasks.

The project achieves strong problem-solution alignment and demonstrates scalable AI application architecture.

10. FUTURE SCOPE

- Multi-language recipe generation
- SEO keyword optimization
- AI image generation for recipes
- Nutrition analysis integration
- User account and recipe history storage
- Cloud auto-scaling deployment

11. APPENDIX

Source Code (app.py):

```
import streamlit as st
import google.generativeai as genai
import random

# Configure Google Generative AI API key
# Ideally, this should be stored in st.secrets or environment variables
api_key = "YOUR API KEY"
genai.configure(api_key=api_key)

# Generation Configuration
generation_config = {
    "temperature": 0.75,
    "top_p": 0.95,
    "top_k": 64,
    "max_output_tokens": 8192,
    "response_mime_type": "text/plain",
}
```

```

# Function to get a random programming joke
def get_joke():
    jokes = [
        "Why do programmers prefer dark mode? Because light attracts bugs.",
        "Why did the developer go broke? Because he used up all his cache.",
        "How many programmers does it take to change a light bulb? None. It's a hardware problem.",
        "What is a programmer's favorite hangout place? Foo Bar.",
        "Why do Java developers wear glasses? Because they don't see sharp.",
        "A SQL query walks into a bar, walks up to two tables and asks... 'Can I join you?'",
        "Why was the JavaScript developer sad? Because he didn't know how to 'null' his feelings.",
        "What do you call a programmer from Finland? Nerdic.",
        "Why did the computer go to the doctor? Because it had a virus!",
        "There are 10 types of people in the world: those who understand binary, and those who don't."
    ]
    return random.choice(jokes)

# Function to generate recipe blog
def recipe_generation(topic, word_count):
    try:
        model = genai.GenerativeModel(
            model_name="gemini-flash-latest",
            generation_config=generation_config,
        )

        prompt = f"""
        You are a professional food blogger. Create a detailed and engaging recipe blog post about "{topic}".
        The blog post should be approximately {word_count} words long.
        Include a catchy title, an introduction, ingredients list, step-by-step instructions, and a conclusion.
        Make it fun and appetizing.
        """
        response = model.generate_content(prompt)
        return response.text
    except Exception as e:
        return f"Error creating recipe: {str(e)}"

# Streamlit UI
def main():
    st.set_page_config(
        page_title="Flavor Fusion",
        page_icon="➕",
        layout="wide",
        initial_sidebar_state="expanded"
    )

# Custom CSS for styling
st.markdown("""
<style>
.main {
    background-color: #f8f9fa;
}
.stApp {
```

```

max-width: 1200px;
margin: 0 auto;
}
h1{
color: #ff6b6b;
font-family: 'Helvetica Neue', sans-serif;
text-align: center;
padding-bottom: 20px;
border-bottom: 2px solid #ff6b6b;
}
.stButton>button{
background-color: #ff6b6b;
color: white;
border-radius: 20px;
padding: 10px 24px;
border: none;
font-weight: bold;
width: 100%;
transition: all 0.3s ease;
}
.stButton>button:hover{
background-color: #ff4757;
transform: scale(1.02);
}
.stTextInput>div>div>input{
border-radius: 10px;
}
.recipe-card{
background-color: white;
padding: 30px;
border-radius: 15px;
box-shadow: 0 4px 6px rgba(0,0,0,0.1);
border-left: 5px solid #ff6b6b;
}
.joke-box{
background-color: #ffff3cd;
color: #856404;
padding: 15px;
border-radius: 10px;
border: 1px solid #ffeeba;
margin-bottom: 20px;
text-align: center;
font-style: italic;
}
</style>
""", unsafe_allow_html=True)

st.title("👋 Flavor Fusion")
st.markdown("<h3 style='text-align: center; color: #555;'>AI-Driven Recipe Blogging</h3>", unsafe_allow_html=True)
st.markdown("---")

# Layout with columns

```

```

col1, col2 = st.columns([1, 2], gap="large")

with col1:
    st.markdown("### 🔎 Configuration")
    st.info("Customize your recipe request below.")

    topic = st.text_input("📝 Recipe Topic", placeholder="e.g., Vegan Chocolate Cake")
    word_count = st.slider("✍️ Desired Word Count", min_value=100, max_value=2000, value=500, step=50)

    st.markdown("### 🤖 Model Settings")
    st.caption("Using Gemini Flash for fast generation.")

    generate_btn = st.button("👉 Generate Recipe")

with col2:
    if generate_btn:
        if topic:
            # Joke display
            joke = get_joke()

            # Create a placeholder for the joke
            joke_placeholder = st.empty()
            joke_placeholder.markdown(f"""
<div class="joke-box">
    😊 <b>Programmer Joke:</b><br>{joke}
</div>
""", unsafe_allow_html=True)

        with st.spinner("👨‍🍳 Chef AI is cooking up your blog post..."):
            recipe_content = recipe_generation(topic, word_count)

            # Clear joke
            joke_placeholder.empty()

        if recipe_content and "Error" not in recipe_content:
            st.markdown(f"""
<div class="recipe-card">
    {recipe_content}
</div>
""", unsafe_allow_html=True)

            st.success("👉 Recipe generated successfully!")

        st.download_button(
            label="📥 Download Recipe",
            data=recipe_content,
            file_name=f"{topic.replace(' ', '_').lower()}_recipe.md",
            mime="text/markdown",
            help="Save this recipe to your computer"
        )
    else:

```

```
        st.error("😢 " + recipe_content)
    else:
        st.warning("⚠️ Please enter a recipe topic first.")
    else:
        st.markdown(""""
<div style='text-align: center; padding: 50px; color: #888;'>
<h3>👉 Ready to cook?</h3>
<p>Enter a topic in the sidebar and hit Generate!</p>
<div style='font-size: 50px;'>🥗 🍕 🍗 🌯 </div>
</div>
""", unsafe_allow_html=True)

if __name__ == "__main__":
    main()
```

GitHub Repository

<https://github.com/RaniNidadavolu/AIDrivenRecipeBlogging>

Project Demo Link

https://drive.google.com/file/d/1yXiaPI3NcRrDJxg5Y-NFgKvLUIxFRx40/view?usp=drive_link