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

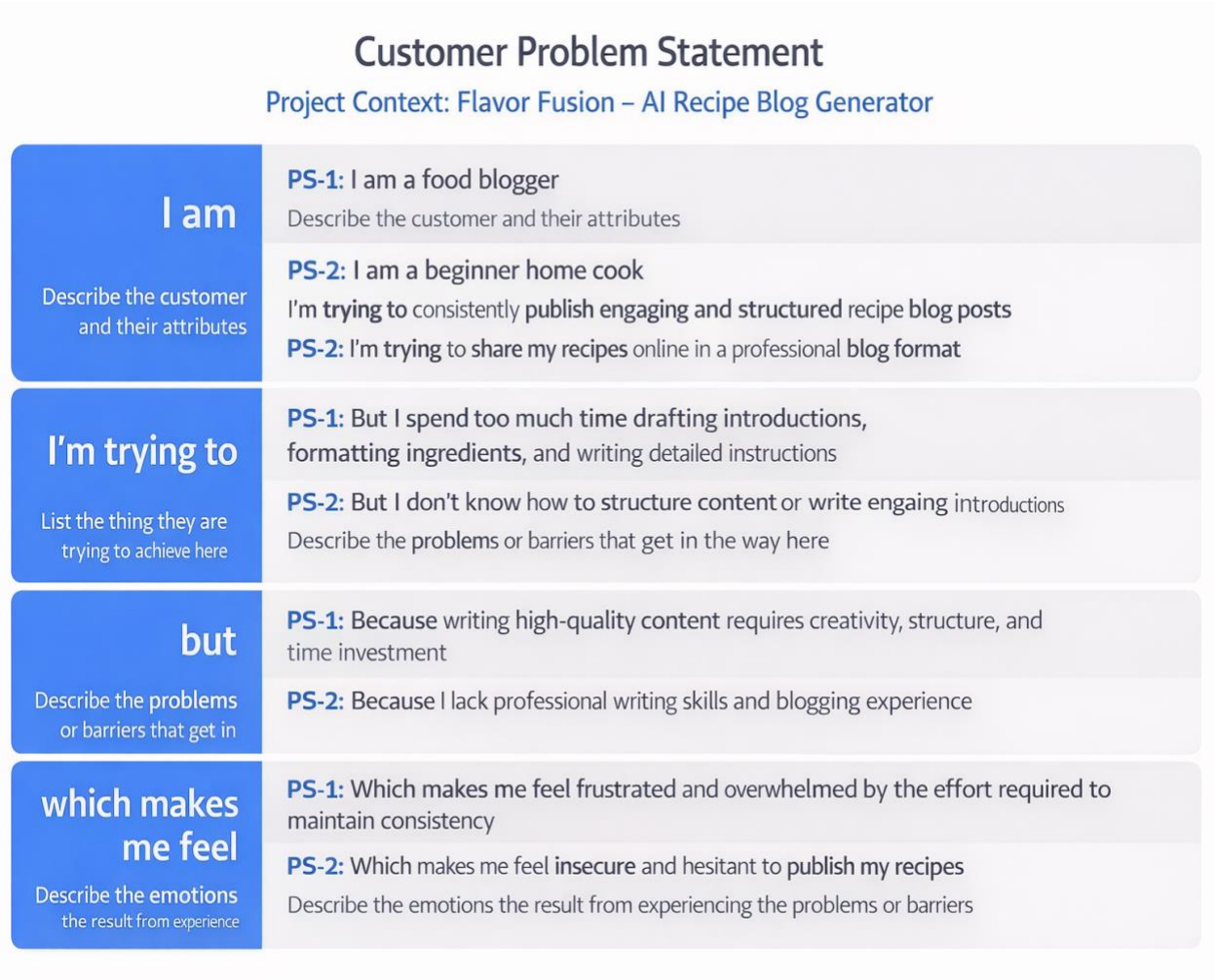
2. IDEATION PHASE

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:

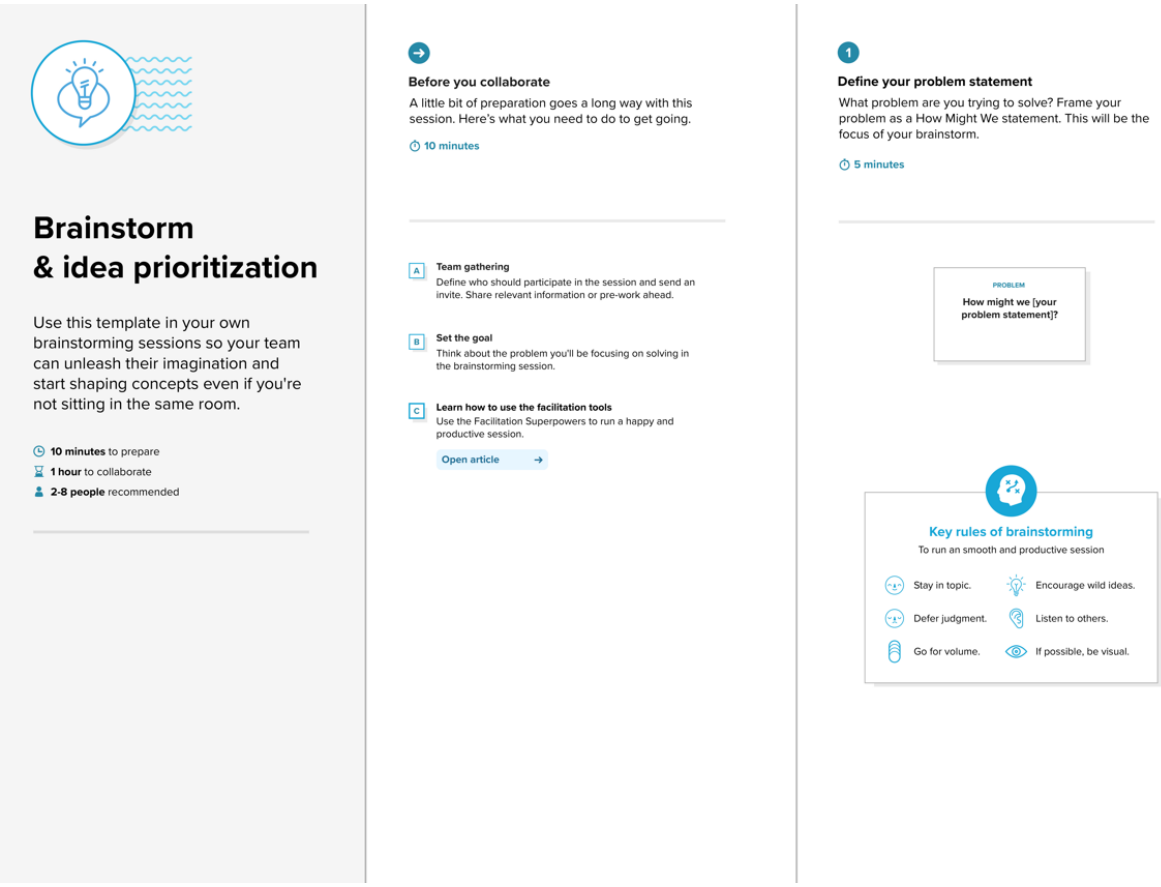
- Travel blog generator
- Fitness plan generator
- Academic summary tool
- AI recipe generator

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.



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

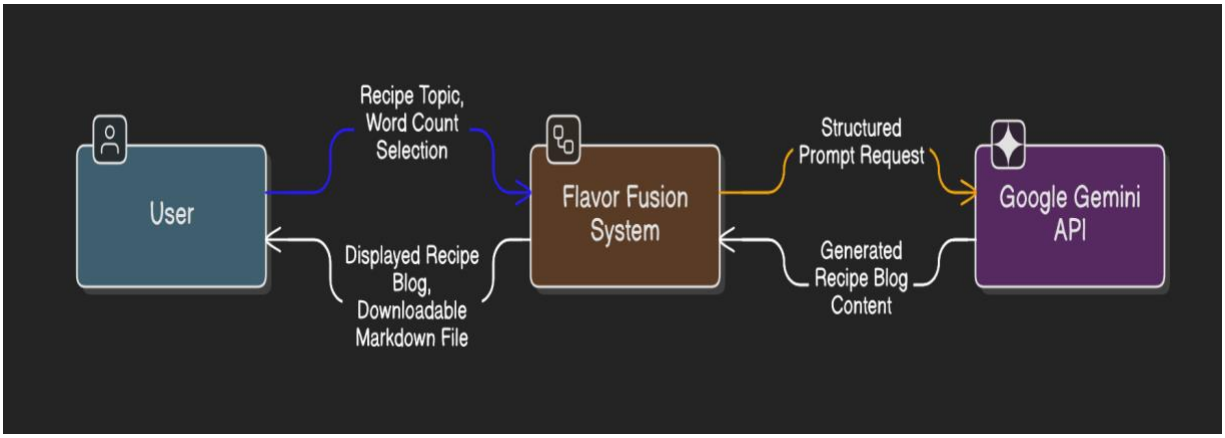
Non-Functional Requirements

- 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

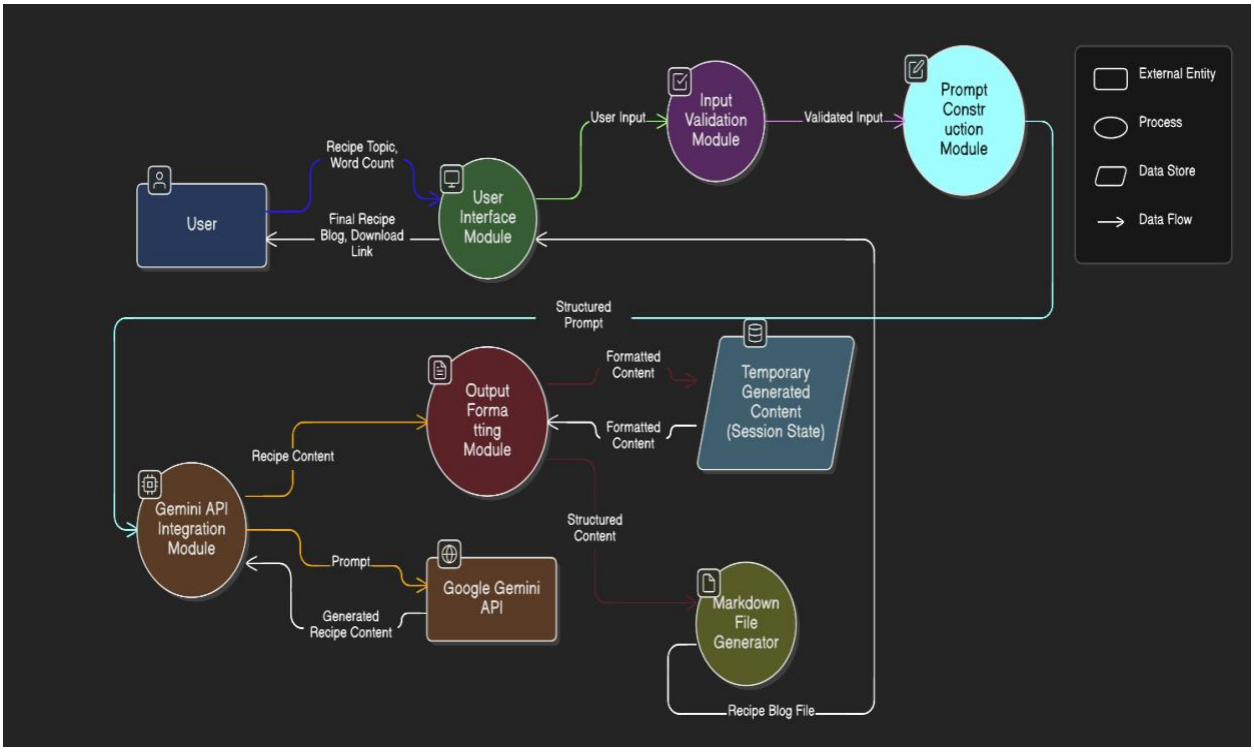
3.3 Data Flow Diagram (Description)

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

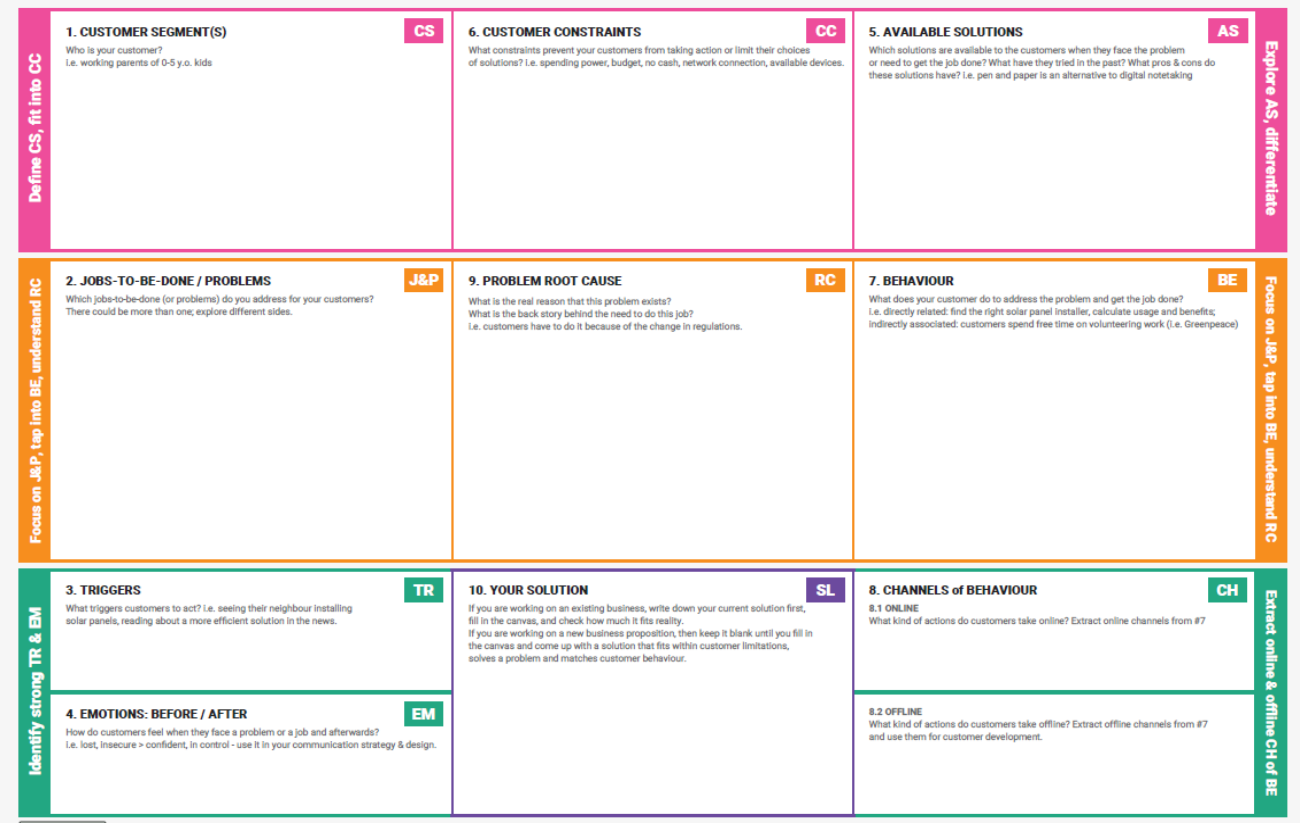
4. PROJECT DESIGN

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.



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:

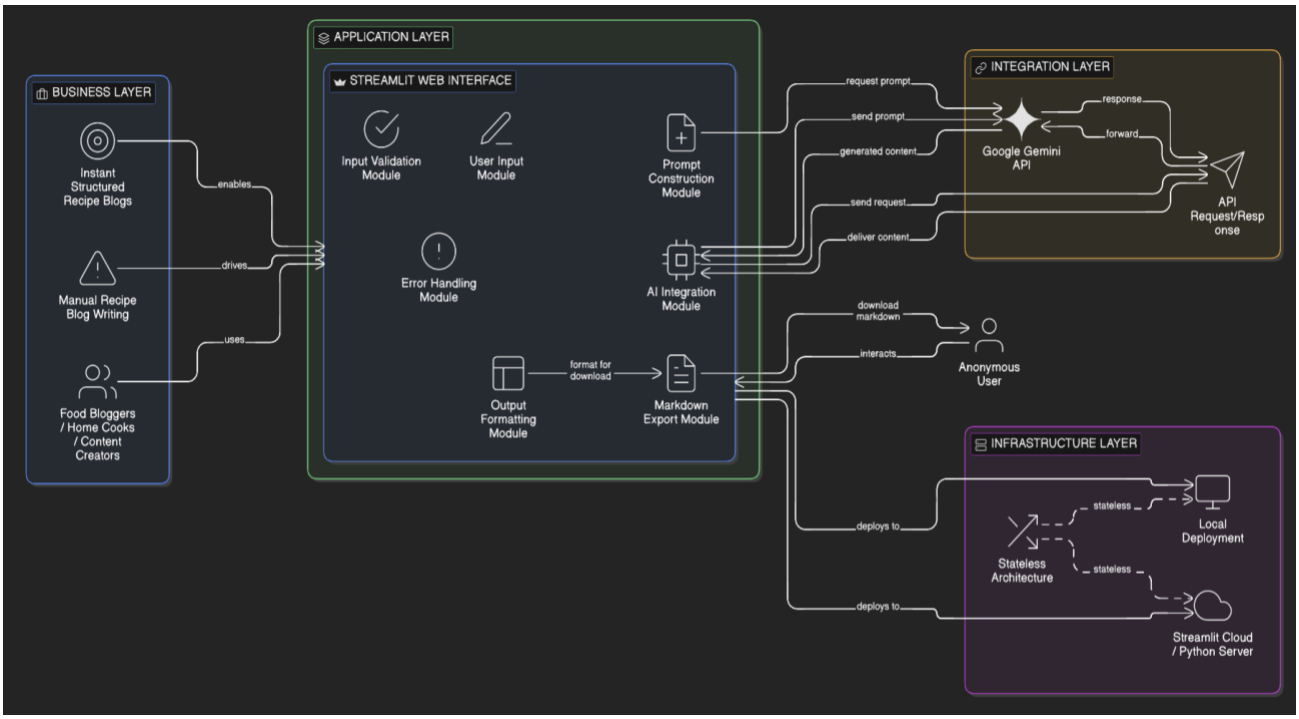
Business Layer:
User need for faster blog creation.

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

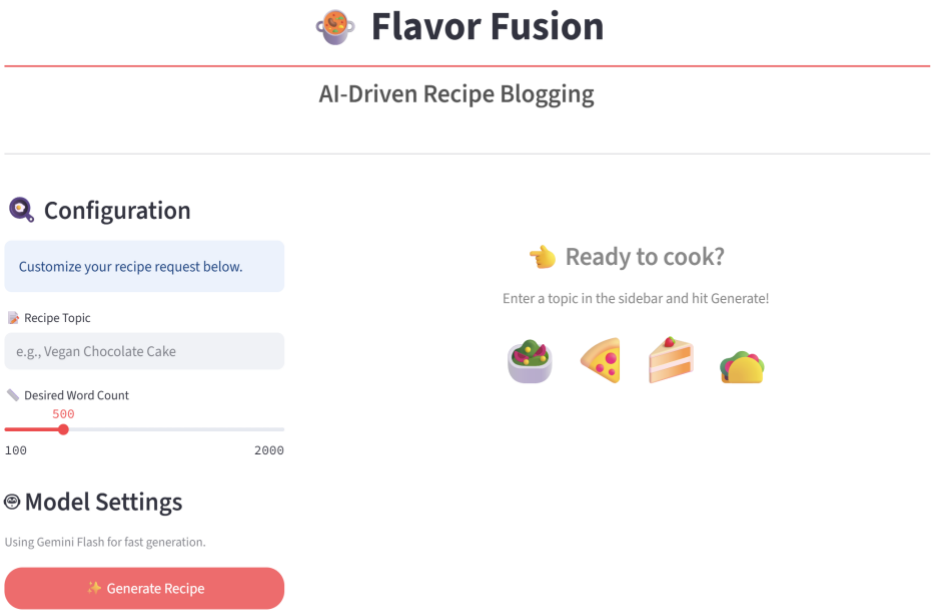
- AI generation time tested under multiple word count conditions.
- Short output (~200 words): 3–5 seconds.
- Long output (~1500 words): 5–8 seconds depending on API latency.
- No application crash during API timeout simulation.

System confirmed stable for UAT.

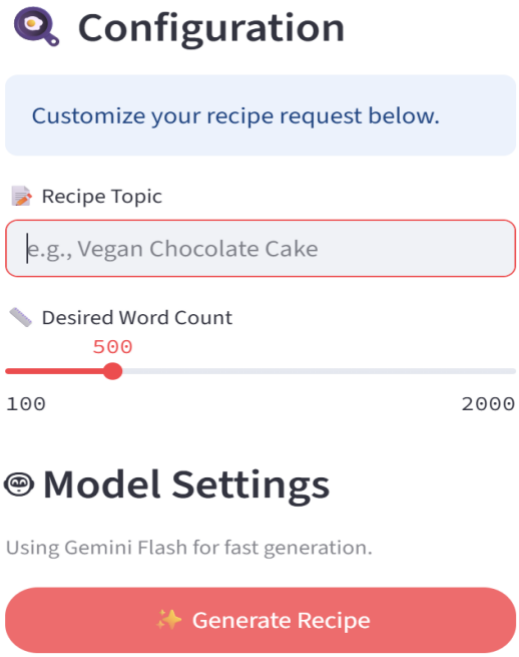
7. RESULTS

7.1 Output Screenshots

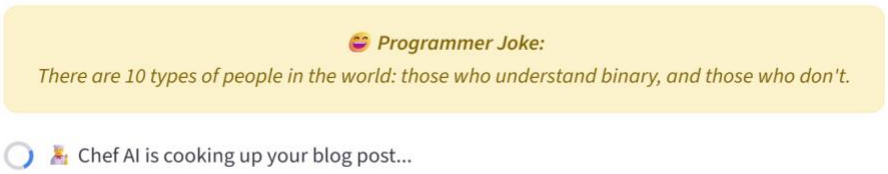
- Main interface screen



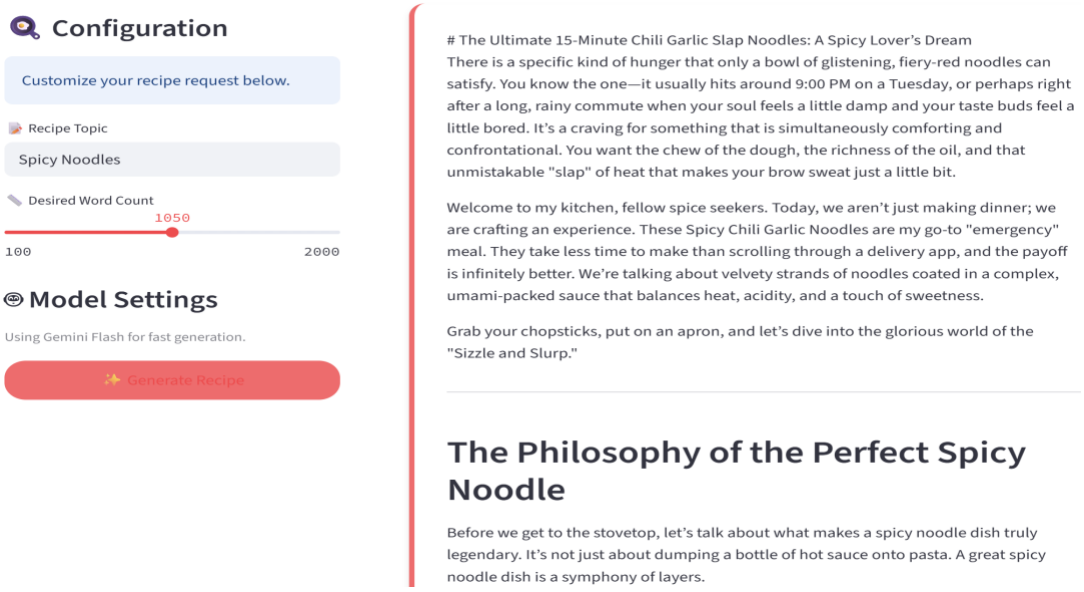
- Input section



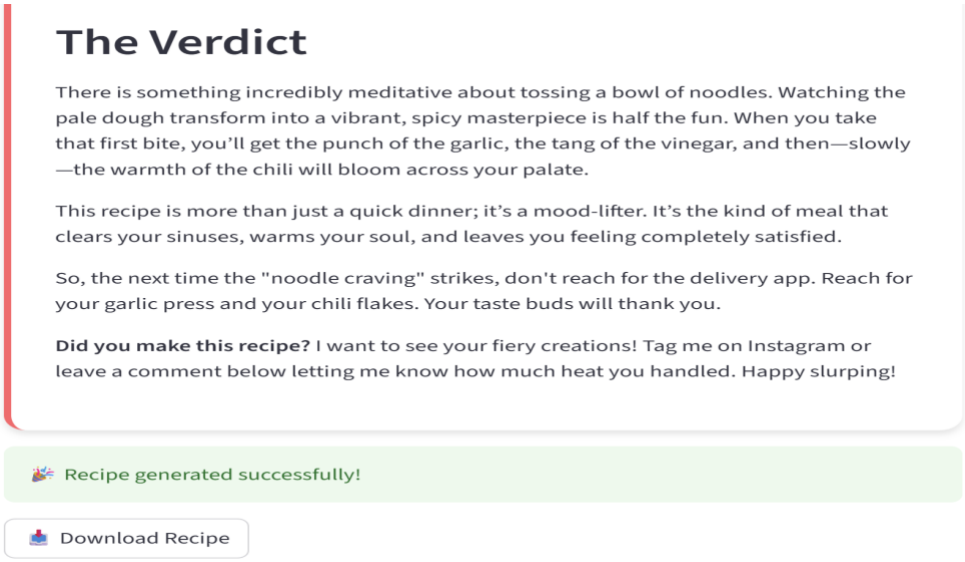
- Waiting Time

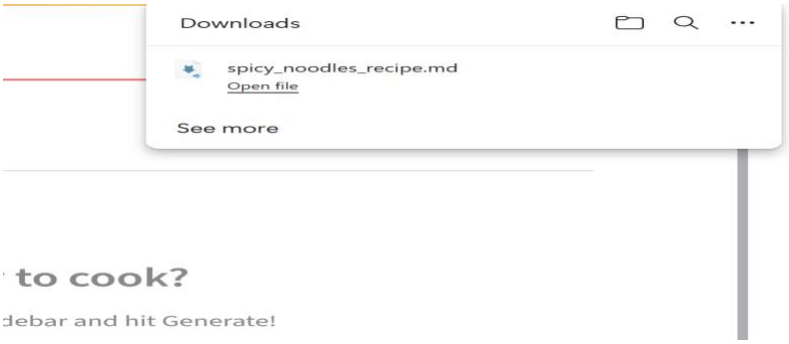


- Generated recipe output



- Markdown download confirmation





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: #fff3cd;
    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