FINAL REPORT

FLAVOUR-FUSION

TEAM ID:LTVIP2026TMIDS65419

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

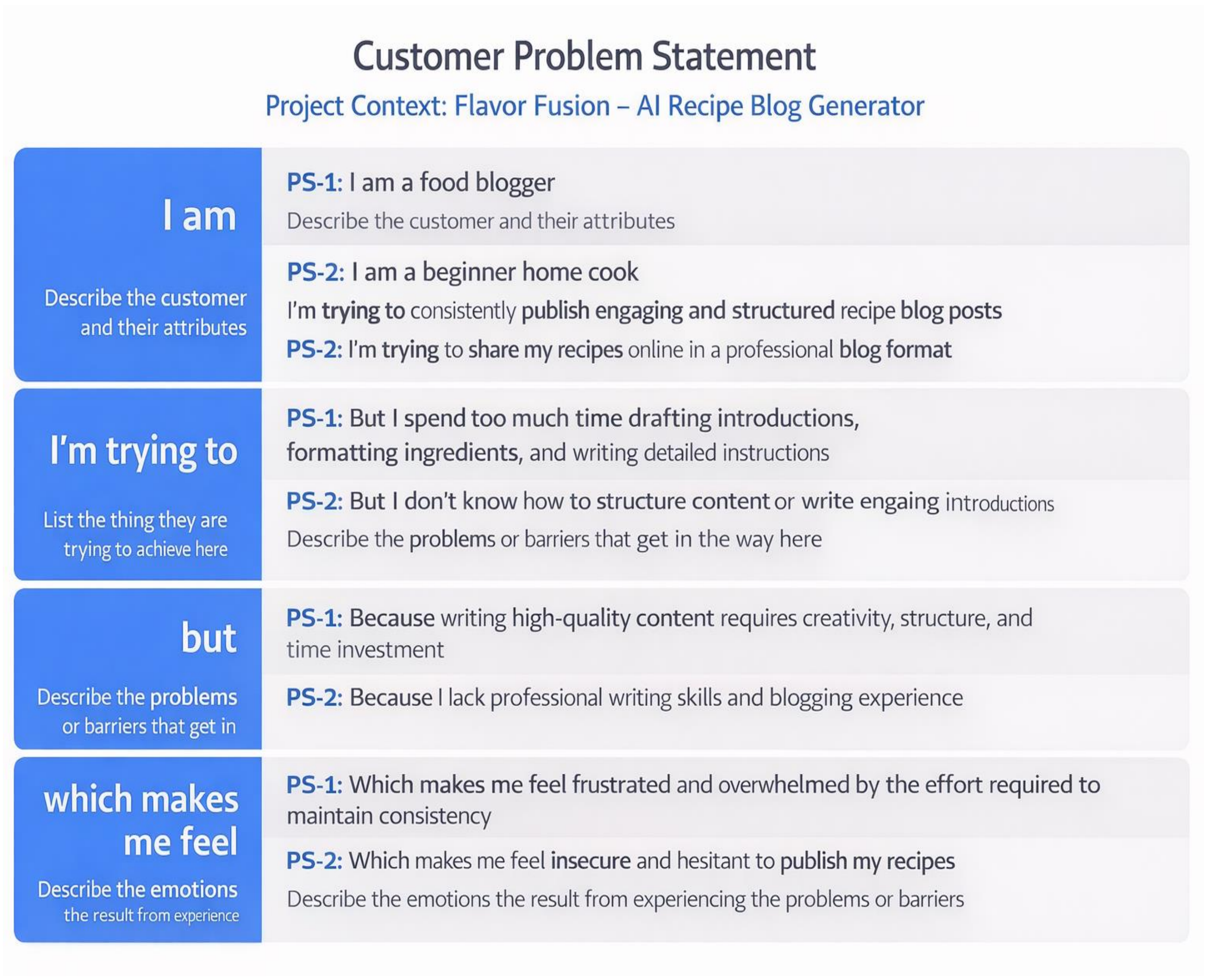
2. IDEATION PHASE

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.



Customer Problem Statement

Project Context: Flavor Fusion – AI Recipe Blog Generator

I am Describe the customer and their attributes	**PS-1:** I am a food blogger Describe the customer and their attributes **PS-2:** I am a beginner home cook I'm **trying to** consistently **publish engaging and structured** recipe **blog posts** **PS-2:** I'm **trying** to **share my recipes** online in a professional **blog format**
I'm trying to List the thing they are trying to achieve here	**PS-1:** But I spend too much time drafting introductions, formatting **ingredients,** and writing detailed instructions **PS-2:** But I don't know how to **structure content** or **write engaging** introductions Describe the **problems** or barriers that get in the way here
but Describe the **problems** or barriers that get in	**PS-1:** Because writing **high-quality content** requires creativity, structure, and time investment **PS-2:** Because I lack professional writing skills and blogging experience
which makes me feel Describe the emotions the result from experience	**PS-1:** Which makes me feel frustrated and overwhelmed by the effort required to maintain consistency **PS-2:** Which makes me feel **insecure** and hesitant to **publish my recipes** Describe the emotions the result from experiencing the problems or barriers

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:

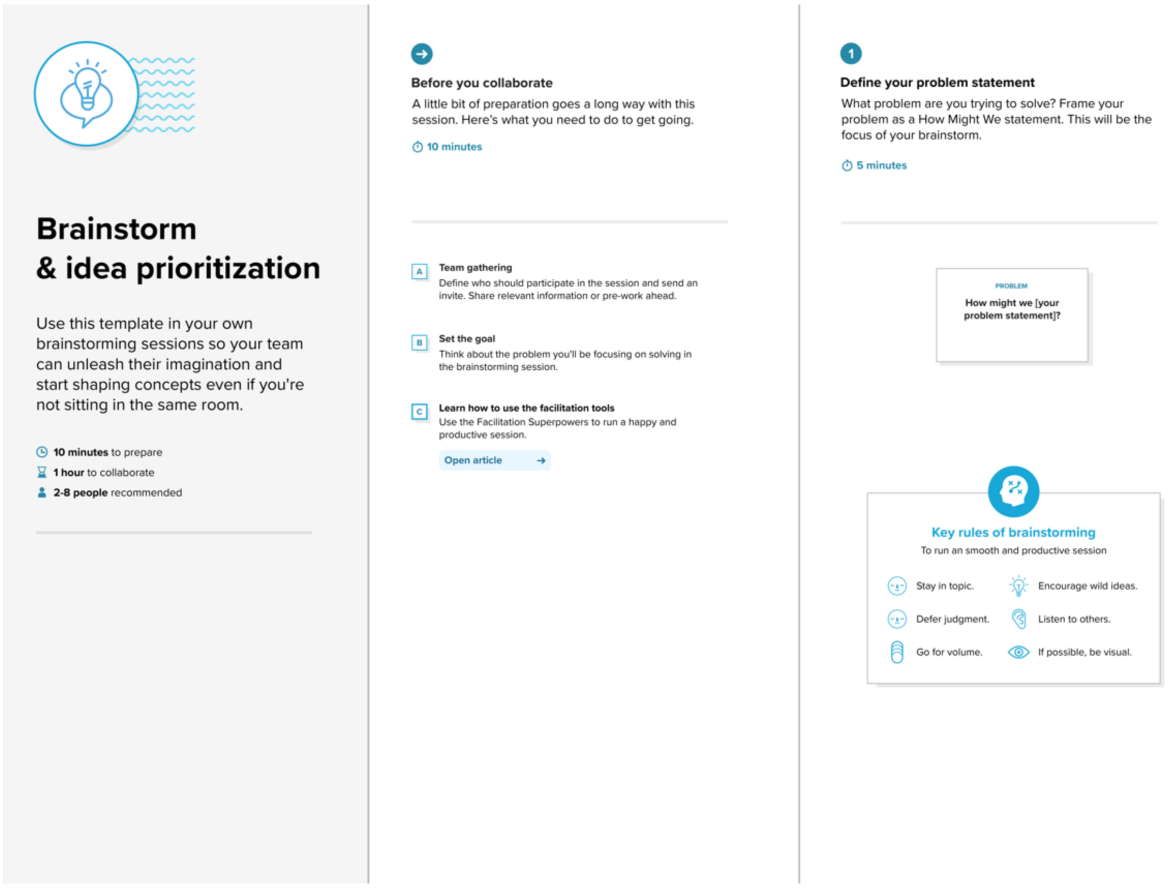
- Travel blog generator
- Fitness plan generator
- Academic summary tool
- AI recipe generator

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

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 →

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

PROBLEM
How might we [your problem statement]?

Key rules of brainstorming
To run an 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

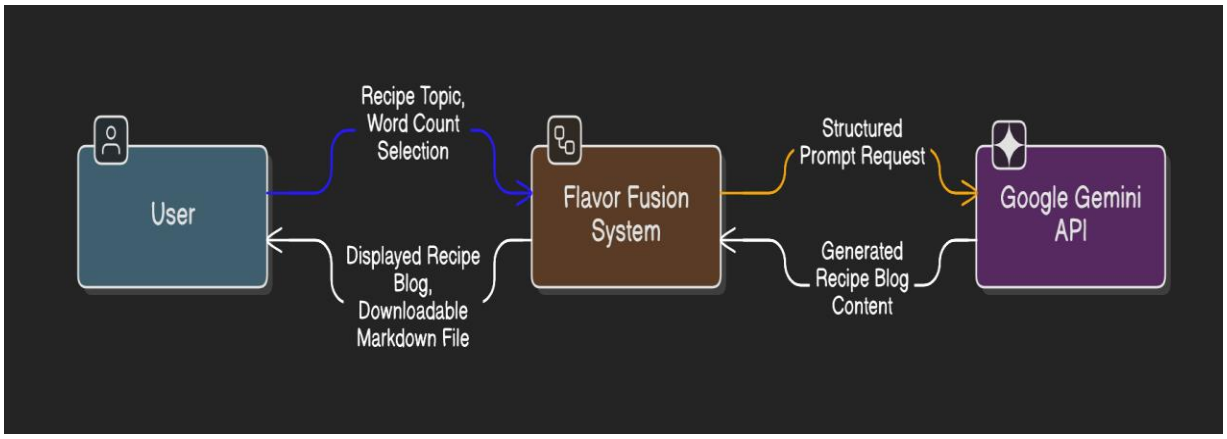
Non-Functional Requirements

- 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

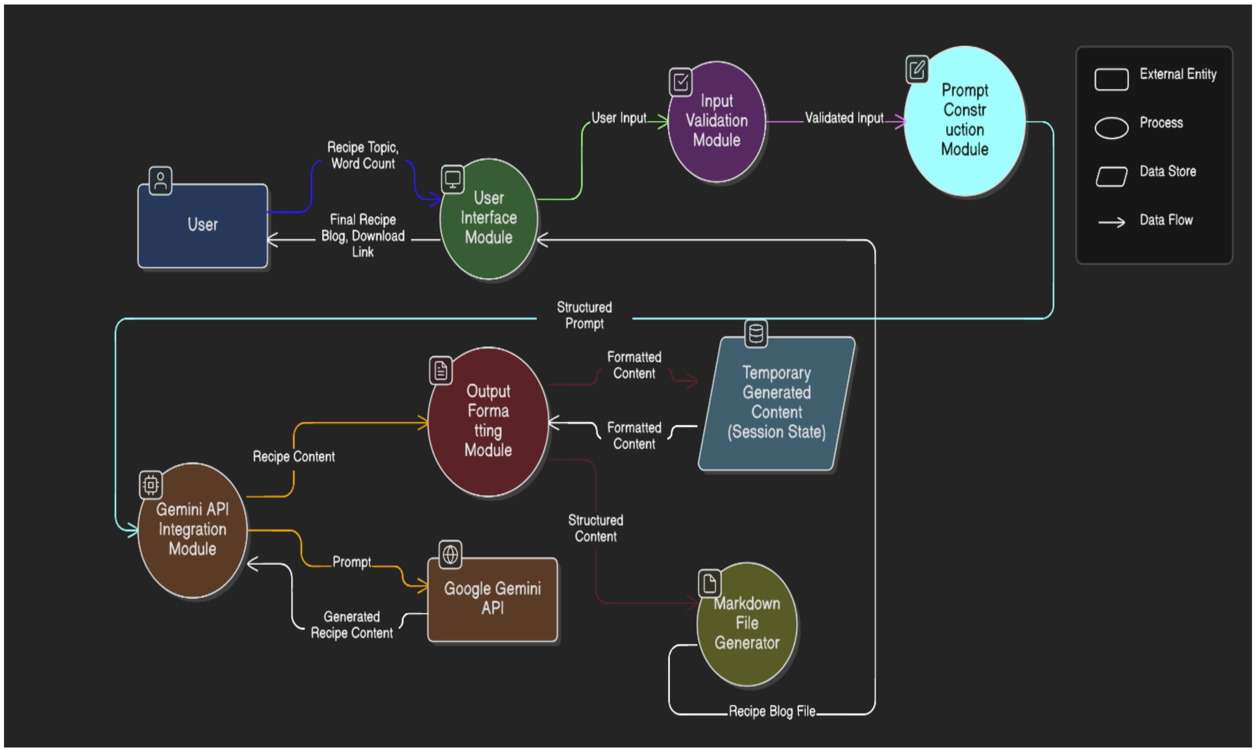
3.3 Data Flow Diagram (Description)

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

DFD Level 0 (Industry Standard)


User
Recipe Topic, Word Count Selection
Displayed Recipe Blog, Downloadable Markdown File
Flavor Fusion System
Structured Prompt Request
Generated Recipe Blog Content
Google Gemini API

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

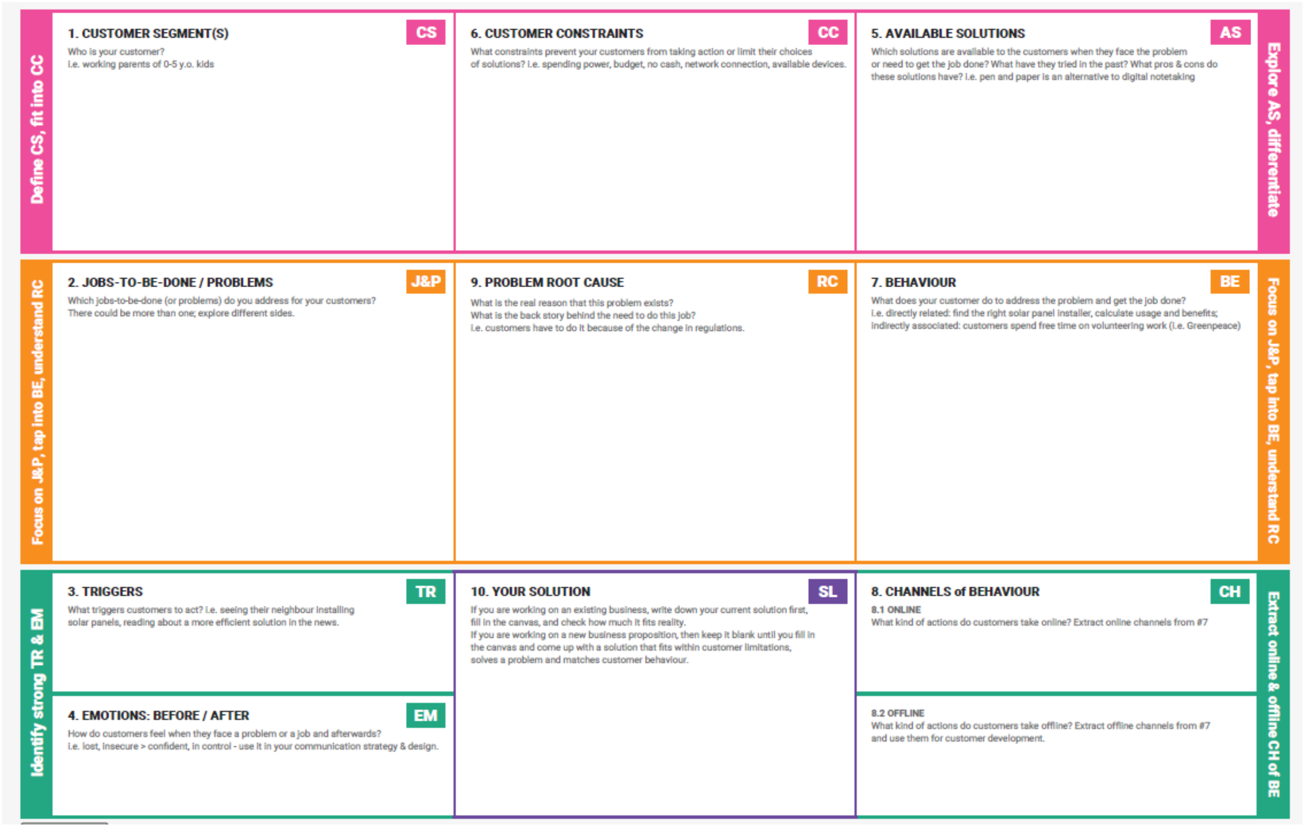
4. PROJECT DESIGN

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) CS
Who is your customer?
i.e. working parents of 0-5 y.o. kids

6. CUSTOMER CONSTRAINTS CC
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.

5. AVAILABLE SOLUTIONS AS
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

Explore AS, differentiate

Focus on J&P, tap into BE, understand RC

2. JOBS-TO-BE-DONE / PROBLEMS J&P
Which jobs-to-be-done (or problems) do you address for your customers?
There could be more than one; explore different sides.

9. PROBLEM ROOT CAUSE RC
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.

7. BEHAVIOUR BE
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 benefits; indirectly associated: customers spend free time on volunteering work (i.e. Greenpeace)

Focus on J&P, tap into BE, understand RC

Identify strong TR & EM

3. TRIGGERS TR
What triggers customers to act? i.e. seeing their neighbour installing solar panels, reading about a more efficient solution in the news.

4. EMOTIONS: BEFORE / AFTER EM
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.

10. YOUR SOLUTION SL
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.

8. CHANNELS of BEHAVIOUR CH
8.1 ONLINE
What kind of actions do customers take online? Extract online channels from #7

8.2 OFFLINE
What kind of actions do customers take offline? Extract offline channels from #7 and use them for customer development.

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:

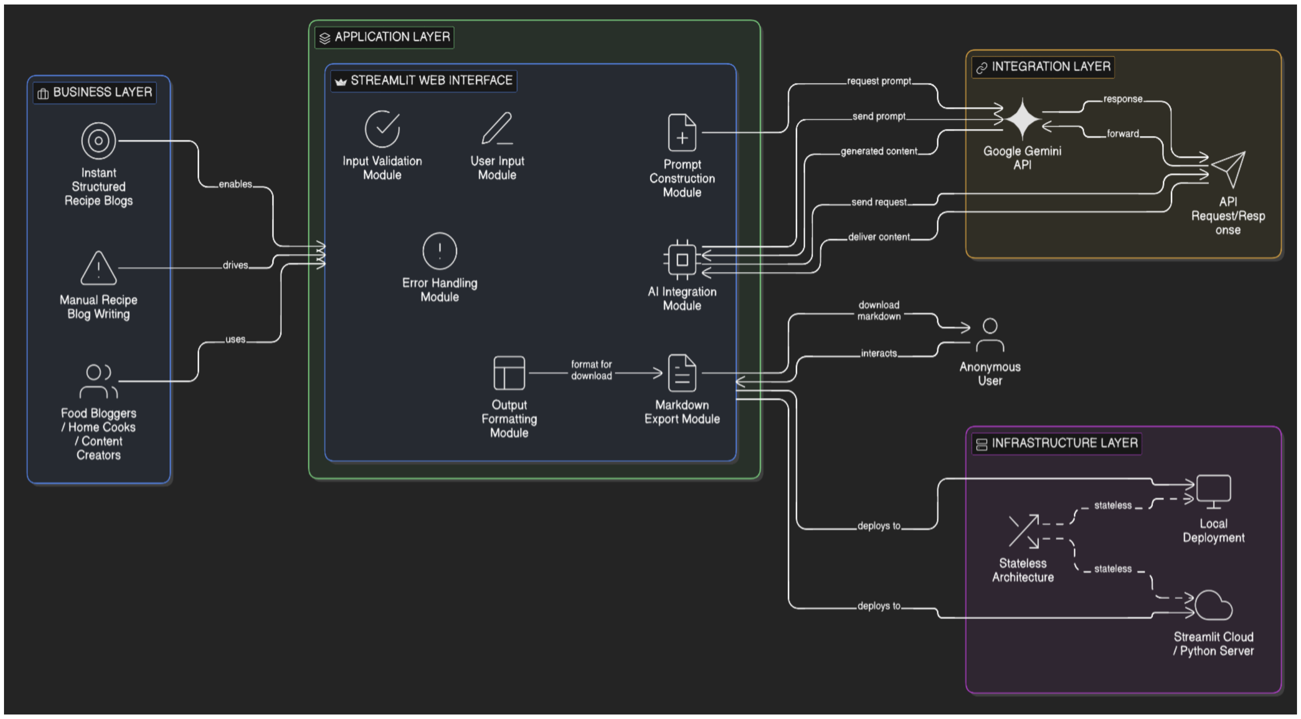
Business Layer:
User need for faster blog creation.

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

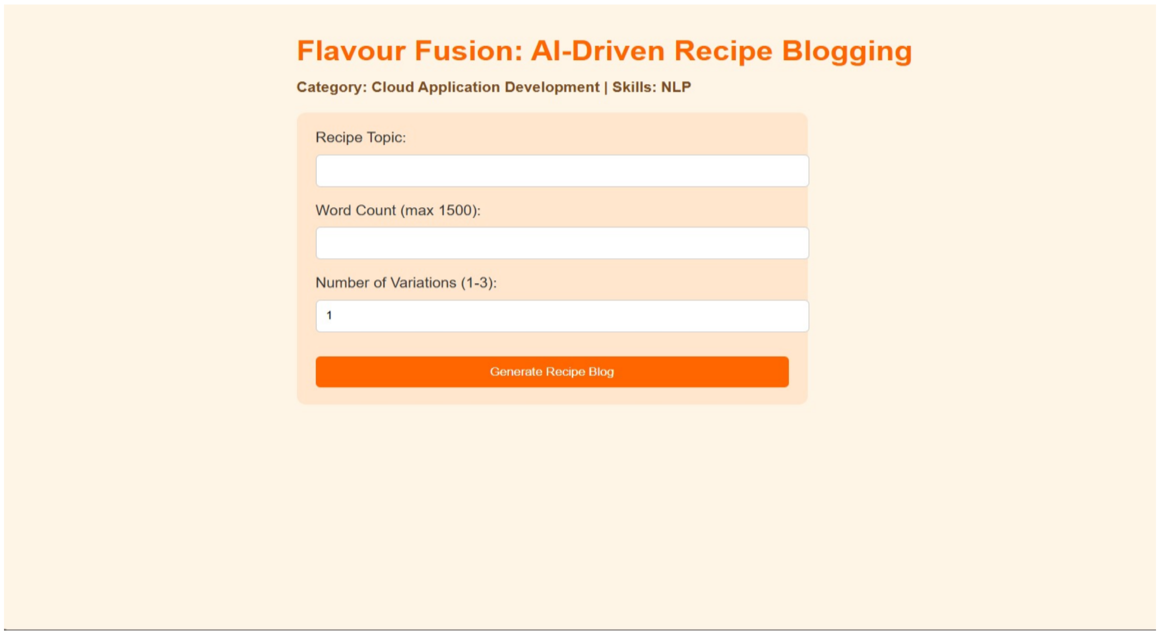
- AI generation time tested under multiple word count conditions.
- Short output (~200 words): 3–5 seconds.
- Long output (~1500 words): 5–8 seconds depending on API latency.
- No application crash during API timeout simulation.

System confirmed stable for UAT.

7. RESULTS

7.1 Output Screenshots

- Main interface screen


Flavour Fusion: AI-Driven Recipe Blogging

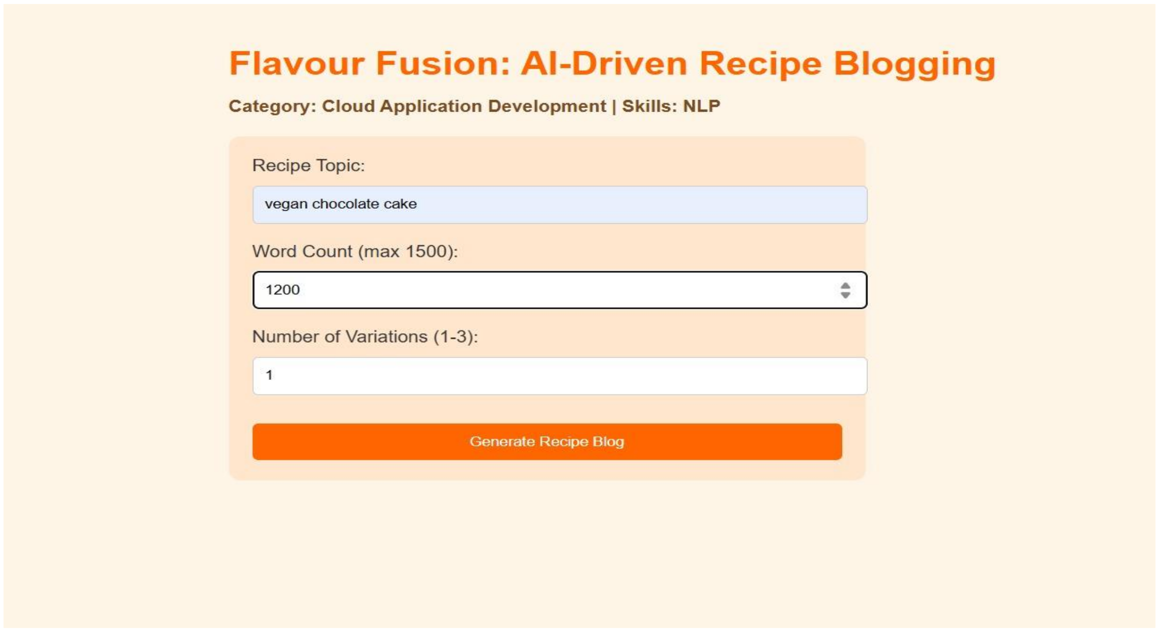
Category: Cloud Application Development | Skills: NLP

Recipe Topic:

Word Count (max 1500):

Number of Variations (1-3): 1

- Input section


Flavour Fusion: AI-Driven Recipe Blogging

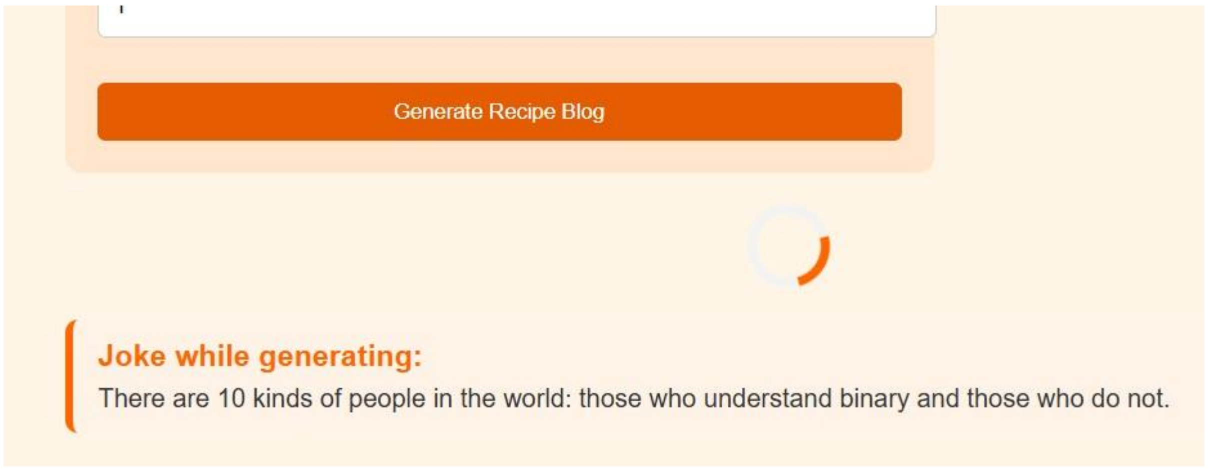
Category: Cloud Application Development | Skills: NLP

Recipe Topic: vegan chocolate cake

Word Count (max 1500): 1200

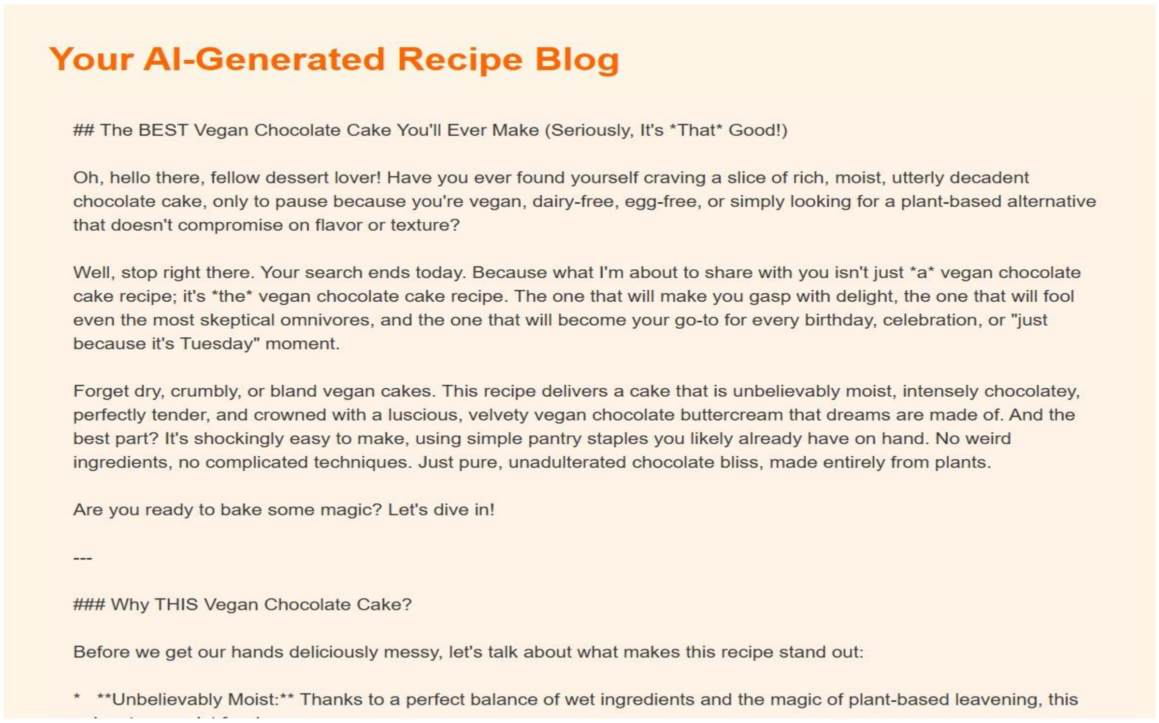
Number of Variations (1-3): 1

- Waiting Time



Joke while generating:
There are 10 kinds of people in the world: those who understand binary and those who do not.

- Generated recipe output



Your AI-Generated Recipe Blog

The BEST Vegan Chocolate Cake You'll Ever Make (Seriously, It's *That* Good!)

Oh, hello there, fellow dessert lover! Have you ever found yourself craving a slice of rich, moist, utterly decadent chocolate cake, only to pause because you're vegan, dairy-free, egg-free, or simply looking for a plant-based alternative that doesn't compromise on flavor or texture?

Well, stop right there. Your search ends today. Because what I'm about to share with you isn't just *a* vegan chocolate cake recipe; it's *the* vegan chocolate cake recipe. The one that will make you gasp with delight, the one that will fool even the most skeptical omnivores, and the one that will become your go-to for every birthday, celebration, or "just because it's Tuesday" moment.

Forget dry, crumbly, or bland vegan cakes. This recipe delivers a cake that is unbelievably moist, intensely chocolatey, perfectly tender, and crowned with a luscious, velvety vegan chocolate buttercream that dreams are made of. And the best part? It's shockingly easy to make, using simple pantry staples you likely already have on hand. No weird ingredients, no complicated techniques. Just pure, unadulterated chocolate bliss, made entirely from plants.

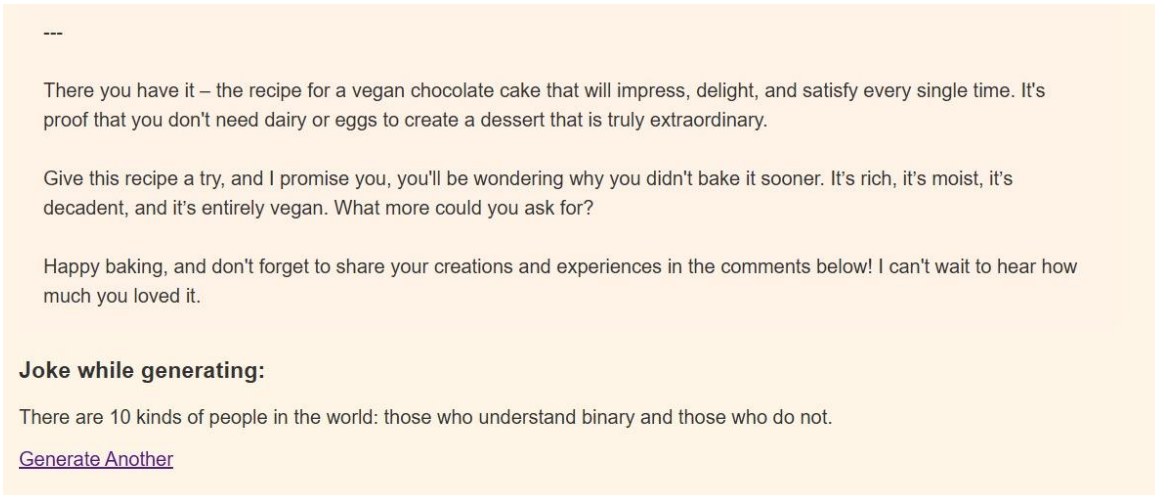
Are you ready to bake some magic? Let's dive in!

Why THIS Vegan Chocolate Cake?

Before we get our hands deliciously messy, let's talk about what makes this recipe stand out:

*   **Unbelievably Moist:** Thanks to a perfect balance of wet ingredients and the magic of plant-based leavening, this

- Generate another recipe confirmation



There you have it – the recipe for a vegan chocolate cake that will impress, delight, and satisfy every single time. It's proof that you don't need dairy or eggs to create a dessert that is truly extraordinary.

Give this recipe a try, and I promise you, you'll be wondering why you didn't bake it sooner. It's rich, it's moist, it's decadent, and it's entirely vegan. What more could you ask for?

Happy baking, and don't forget to share your creations and experiences in the comments below! I can't wait to hear how much you loved it.

Joke while generating:

There are 10 kinds of people in the world: those who understand binary and those who do not.

Generate Another

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 os
from flask import Flask, render_template, request
from dotenv import load_dotenv

try:
    # Preferred SDK (actively maintained)
    from google import genai as google_genai
    _GENAI_MODE = "new"
except ImportError:
    # Backward-compatible fallback for older environments
    import google.generativeai as google_genai
    _GENAI_MODE = "legacy"

# Load environment variables
load_dotenv()

app = Flask(__name__)

# Get API key
API_KEY = os.getenv("GOOGLE_API_KEY", "").strip()
print(f"API key loaded: {'yes' if API_KEY else 'no'}")

# Configure legacy Gemini SDK
if API_KEY and _GENAI_MODE == "legacy":
    google_genai.configure(api_key=API_KEY)

# Programmer jokes
PROGRAMMER_JOKES = [
    "Why do programmers prefer dark mode? Because light attracts bugs!",
    "A SQL query walks into a bar, walks up to two tables, and asks: Can I join you?",
    "There are 10 kinds of people in the world: those who understand binary and those who do not.",
    "Debugging is like being the detective in a crime movie where you are also the murderer.",
]

def fallback_recipe(topic: str, word_count: int) -> str:
    return (
        f"{topic} - Simple Home Recipe Guide\n\n"
        "Ingredients:\n"
        "- 2 cups main ingredient of choice\n"
        "- 1 tbsp oil or butter\n"
        "- 1 tsp salt (adjust to taste)\n"
        "- 1 tsp spices/herbs\n"
        "- Optional garnish\n\n"
        "Steps:\n"
        "1. Prep all ingredients and preheat cookware.\n"
        "2. Cook base ingredients on medium heat until aromatic.\n"
        "3. Add seasonings and simmer until textures are just right.\n"
        "4. Plate, garnish, and serve warm.\n\n"
        "Cooking Tips:\n"
        "- Taste and adjust seasoning near the end.\n"
        "- Keep heat moderate to avoid burning.\n"
        "- Pair with a fresh side salad or bread.\n\n"
        f"Requested target length: ~{word_count} words."
    )

def call_gemini(prompt: str) -> str:
```

```
if not API_KEY:
    raise RuntimeError("GOOGLE_API_KEY is not configured.")

if _GENAI_MODE == "new":
    client = google_genai.Client(api_key=API_KEY)
    model_candidates = ["gemini-2.5-flash", "gemini-2.0-flash", "gemini-1.5-flash"]
    last_error = None
    for model_name in model_candidates:
        try:
            response = client.models.generate_content(
                model=model_name,
                contents=prompt,
            )
            text = (response.text or "").strip()
            if text:
                return text
        except Exception as exc:
            last_error = exc
    raise RuntimeError(f"Gemini new SDK failed for all model candidates: {last_error}")

model_candidates = ["gemini-2.0-flash", "gemini-1.5-flash", "gemini-1.5-flash-latest"]
last_error = None
for model_name in model_candidates:
    try:
        model = google_genai.GenerativeModel(model_name)
        response = model.generate_content(prompt)
        text = (response.text or "").strip()
        if text:
            return text
    except Exception as exc:
        last_error = exc
raise RuntimeError(f"Gemini legacy SDK failed for all model candidates: {last_error}")
```

```
def generate_recipe(topic: str, word_count: int) -> str:
    print(f"Generating recipe for topic: {topic}")

    prompt = (
        f"Write a detailed recipe blog about '{topic}' "
        f"with approximately {word_count} words. "
        "Include ingredients, numbered steps, prep/cook times, "
        "and practical cooking tips."
    )

    try:
        ai_text = call_gemini(prompt)
        if ai_text:
            return ai_text
        raise RuntimeError("Gemini response did not include text.")
    except Exception as exc:
        print(f"Gemini generation failed: {exc}")
        return fallback_recipe(topic, word_count)

@app.route("/")
def home():
    return render_template("index.html", jokes=PROGRAMMER_JOKES)

@app.route("/generate", methods=["POST"])
def generate():
    topic = request.form.get("topic", "Quick and Easy Cooking")
    word_count_raw = request.form.get("word_count", 300)
    variations_raw = request.form.get("variations", 1)
    selected_joke = request.form.get("joke") or PROGRAMMER_JOKES[0]

    try:
        word_count = int(word_count_raw)
    except (TypeError, ValueError):
        word_count = 300
    word_count = max(50, min(1500, word_count))

    try:
        variations = int(variations_raw)
    except (TypeError, ValueError):
        variations = 1
    variations = max(1, min(3, variations))

    blogs = [generate_recipe(topic, word_count) for _ in range(variations)]

    return render_template("result.html", blog=blogs, joke=selected_joke)

if __name__ == "__main__":
    app.run(debug=True)
```

GitHub Repository

https://github.com/Nehapriya30/Flavour-Fusion-AI-Driven-Recipe-Blogging

Project Demo Link

https://drive.google.com/drive/folders/1_DviBK6TFRJokXzs2ZzIddlmzVLZCMdh?usp=sharing