GENAI HACKATHON

Project Title: Flavour Fusion: AI-Driven Recipe Blogging

Team Name: PromptPioneers

Team Members:

- KASTURI
- SUMAVARSHA
- NEHA
- JEEVAN

Phase-1: Brainstorming & Ideation

Objective:

To develop an AI-powered recipe blogging tool that simplifies recipe generation while enhancing engagement and customization.

1. **Problem Statement:**
   Flavour Fusion: AI-Driven Recipe Blogging is a web application that leverages Google's Generative AI to create unique and customized recipe blogs. The app provides users with the ability to input a topic and specify the desired word count for their recipe blog. Using the specified parameters, the AI generates detailed and engaging recipe content. Additionally, the app includes a fun feature where it tells a programmer joke to entertain users while the AI is generating the content.

2. **Proposed Solution: Flavour Fusion**

- Customizable Recipes: Tailored to dietary preferences, word count, and cuisine type.
- Engaging Content: Adds humour (e.g., programmer jokes) for a unique touch.
- SEO-Optimized Output: Ensures blog-ready formatting for better reach.
- Time-Saving Automation: Helps bloggers and content creators generate
- high-quality recipes instantly.

Target Users:

📌 Food Bloggers & Content Creators – SEO-friendly, engaging recipe generation.

📌 Home Cooks & Chefs – Quick, customized recipes for daily use.

📌 Food Brands & Influencers – AI-powered content for marketing & engagement.

Expected Outcome:

✅ Effortless Recipe Generation – AI creates structured, engaging, and SEO-friendly recipes in seconds.

✅ Increased Productivity – Bloggers and creators save time while maintaining high content quality.

✅ Enhanced User Engagement – Humour and storytelling make recipes more enjoyable and shareable.

✅ Wider Audience Reach – SEO-optimized content improves visibility and traffic.

✅ Customization & Personalization – Recipes tailored to dietary needs, preferences, and word count.

Phase-2: Requirement Analysis

Objective:

To develop an AI-driven web application that generates customized recipe blogs based on user inputs and entertains users with programmer jokes while the content is being generated.

Key Points:

Technical Requirements

- Programming Language: Python
- Backend: Google Gemini Flash API
- Frontend: Streamlit Web Framework
- Database API-based queries

Functional Requirements

- Ability to fetch vehicle details using Gemini Flash API.
- Display specifications, reviews, and comparisons in an intuitive UI.
- Provide real-time vehicle maintenance tips based on seasons.
- Allow users to search eco-friendly vehicles based on emissions and incentives.

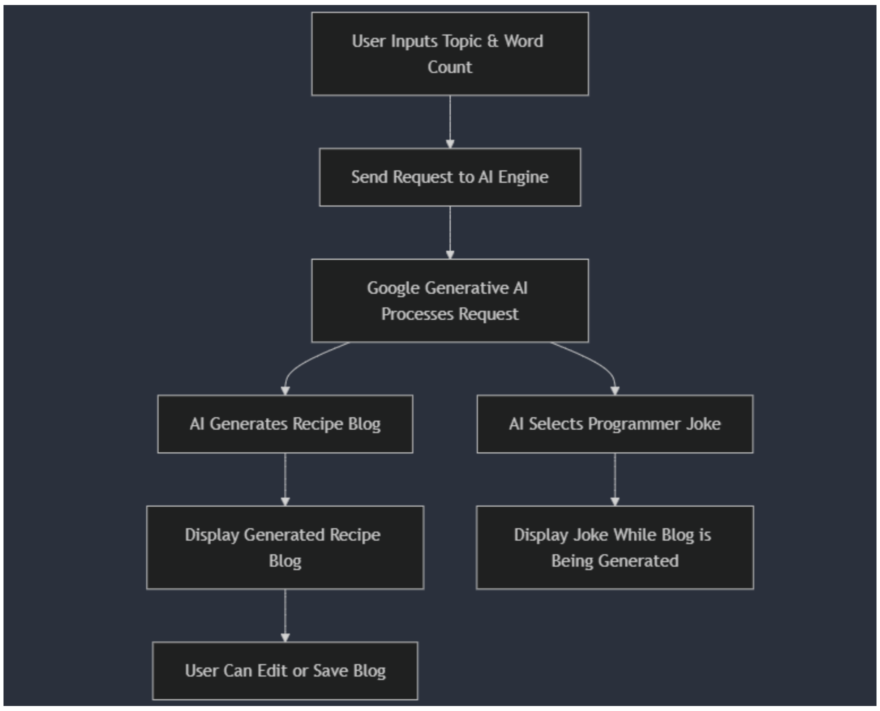
Constraints & Challenges

- Ensuring real-time updates from Gemini API.
- Handling API rate limits and optimizing API calls.
- Providing a smooth UI experience with Streamlit.

Phase-3: Project Design

Objective:

To develop the architecture and user flow of the AutoSage application.


User Inputs Topic & Word Count
Send Request to AI Engine
Google Generative AI Processes Request
AI Generates Recipe Blog
AI Selects Programmer Joke
Display Generated Recipe Blog
Display Joke While Blog is Being Generated
User Can Edit or Save Blog

Key Points

1. System Architecture

- User Input: User inputs a topic and desired word count for the recipe blog.
- Processing: Request is sent to Google Generative AI.
- Data Handling: AI generates the recipe content and selects a programmer joke.
- Frontend Display: Generated recipe blog and joke are displayed on the user interface.

2. User Flow

- Step 1: User inputs a topic and word count for the recipe blog.
- Step 2: The AI processes the request and starts generating the recipe content.
- Step 3: While the recipe content is being generated, the AI selects and displays a programmer joke.
- Step 4: The app displays the generated recipe blog in an easy-to-read format, with options for the user to edit or save the blog.

3. UI/UX Considerations

- Minimalist Interface: Provide a user-friendly interface for seamless navigation.
- Customization: Allow users to customize recipe details such as dietary preferences and cuisine type.
- Humour Element: Display programmer jokes while the recipe is being generated for added engagement.
- Dark & Light Mode: Offer dark and light mode for better user experience.

Phase-4: Project Planning (Agile Methodologies)

Objective:

Break down development tasks for efficient completion.

Sprint	Task	Priority	Duration	Deadline	Assigned To	Dependencies	Expected Outcome
Sprint 1	Environment Setup & API Integration	🔴 High	6 hours (Day 1)	End of Day 1	Shanawaz	Google API Key, Python, Streamlit setup	API connection established & working
Sprint 1	Frontend UI Development	🟡 Medium	2 hours (Day 1)	End of Day 1	Member 2	API response format finalized	Basic UI with input fields

Sprint 2	Vehicle Search & Comparison	🔴 High	3 hours (Day 2)	Mid-Day 2	Anwar	API response, UI elements ready	Search functionality with filters
Sprint 2	Error Handling & Debugging	🔴 High	1.5 hours (Day 2)	Mid-Day 2	Member 1 & 4	API logs, UI inputs	Improved API stability
Sprint 3	Testing & UI Enhancements	🟡 Medium	1.5 hours (Day 2)	Mid-Day 2	Mohammad	API response, UI layout completed	Responsive UI, better user experience
Sprint 3	Final Presentation & Deployment	🟢 Low	1 hour (Day 2)	End of Day 2	Entire Team	Working prototype	Demo-ready project

Sprint Planning with Priorities

Sprint 1 – Setup & Integration (Day 1)

- 🔴 **High Priority: Set up the environment & install dependencies.**
- 🔴 **High Priority: Integrate Google Gemini API.**
- 🟡 **Medium Priority: Build a basic UI with input fields.**

Sprint 2 – Core Features & Debugging (Day 2)

- 🔴 **High Priority: Implement search & comparison functionalities.**
- 🔴 **High Priority: Debug API issues & handle errors in queries.**

Sprint 3 – Testing, Enhancements & Submission (Day 2)

- 🟡 **Medium Priority: Test API responses, refine UI, & fix UI bugs.**
- 🟢 **Low Priority: Final demo preparation & deployment.**

Phase-5: Project Development

Objective:

Implement core features of the AutoSage App.
Key Points

1. Technology Stack Used

- Frontend: Streamlit
- Backend: Google Generative AI
- Programming Language: Python

2. Development Process

- API Integration: Implement API key authentication and Generative AI integration.
- Content Generation: Develop logic for generating customized recipe blogs.
- User Interaction: Implement programmer joke feature during content generation.
- Optimization: Optimize AI queries for performance and relevance.

3. Challenges & Fixes

- Challenge: Delayed AI response times. Fix: Implement caching to store frequently queried topics.
- Challenge: Limited API calls per minute. Fix: Optimize queries to fetch only necessary data

---

Phase-6: Functional & Performance Testing

Objective

Test Case ID	Category	Test Scenario	Expected Outcome	Status	Tester
TC-001	Functional Testing	Query "Vegan chocolate cake recipe"	Relevant recipe content should be displayed.	✅ Passed	Shanawaz
TC-002	Functional Testing	Query "Low-carb breakfast ideas"	Detailed low-carb recipes should be provided.	✅ Passed	Anwar

TC-003	Performance Testing	AI response time under 500ms	AI should return generated content quickly.	⚠ Needs Optimization	Tester 3
TC-004	Bug Fixes & Improvements	Fixed incorrect recipe generation.	Recipe accuracy should be improved.	✅ Fixed	Developer
TC-005	Final Validation	Ensure UI is responsive across devices.	UI should work on mobile & desktop.	❌ Failed - UI broken on mobile	Tester 2
TC-006	Deployment Testing	Host the app using Streamlit Sharing	App should be accessible online.	🚀 Deployed	DevOps

---

Final Submission

1. **Project Report Based on the templates**
2. **Demo Video (3-5 Minutes)**
3. **GitHub/Code Repository Link**
4. **Presentation**