Project Documentation: Flavour Fusion

1. Introduction

Project Title: Flavour Fusion: AI-Driven Recipe Blogging Platform

Team ID: LTVIP2026TMIDS73763

Team Size: 4

Team Leader: Ravilla Poojitha

Team Members: Shaik Farju Taj

Soora Charvani Sree

Nellipatla Jahnavi

---

2. Project Overview

Purpose: To eliminate "Writer's Block" for food bloggers by using Generative AI (Gemini 2.5 Flash) to create professional, SEO-friendly recipe blogs instantly from a simple topic.

Features:

AI Content Generation: Automated recipe writing with prep times and instructions.

User Engagement: Interactive loading states with programmer jokes to improve perceived performance.

Professional Export: Direct "Download as PDF" and "Copy to Clipboard" functionality.

Security: Protected API keys using environment variables.

---

3. Architecture

Frontend: Built with **React.js** (and Streamlit for the rapid MVP) to manage state and handle the interactive UI.

Backend: Developed using **Node.js** and **Express.js** to handle API requests and communicate with Google's Generative AI.

Database: MongoDB is used to store user-generated recipes, user profiles, and metadata for the blogging platform.

---

4. Setup Instructions

Prerequisites: Node.js (v18+), MongoDB Atlas account, and a Google Gemini API Key.

Installation:

```
git clone https://github.com/PrernaJha07/Flavour-Fusion-AI-Driven-Recipe-Blogging.git

cd flavour-fusion
```

npm install (in both client and server directories)

Create a .env file in the root directory and add:

```
GOOGLE_API_KEY=your_key_here
```

---

5. Folder Structure

Client: Contains React components, styles (CSS/Tailwind), and the App.js main logic.

Server: Contains Express routes, AI controller logic, and the MongoDB connection config.

---

6. Running the Application

Frontend: ```bash cd client npm start

Backend: ```bash cd server npm run dev

---

7. API Documentation

Endpoint: POST /api/generate-recipe

Method: POST

Parameters: { "topic": "string", "word_count": number }

Example Response: { "recipe": "Full markdown text of the recipe..." }

---

8. Authentication

Method: Handled via **JSON Web Tokens (JWT)** for secure login.

Logic: Users register/login; the server issues a token which is stored in the browser's LocalStorage to authorize recipe generation requests.

---

9. User Interface

Screenshot 1: Home page with recipe input.

Screenshot 2: Loading state with a "Programmer Joke."

Screenshot 3: Generated recipe with PDF download button.

---

10. Testing

Strategy: Manual unit testing for the AI prompt logic and integration testing for the API-to-database connection.

Tools: Postman for API testing and **Jest** for frontend component verification.

---

11. Known Issues

Token Expiry: Some users may need to re-login if the session expires during long generation tasks.

PDF Formatting: Non-Latin characters (like some Emojis) may occasionally skip rendering in the PDF export.

---

12. Future Enhancements

Image Generation: Integrating Google Imagen to create realistic food photos for each blog post.

Social Sharing: Direct "One-Click Share" to Instagram and WordPress.

User Dashboard: A personal library where users can save and edit their generated recipes.