Project Title : AI Personalized Email Generator

Team Name:

Conditional Coders

Team Members:

- V. Navya
- E. Keerthi
- B. Harini
- N. Amrutha Varshini
- A. Akshitha

Phase-1: Brainstorming & Ideation

Objective:

Develop an AI-personalized Email Generator that enables users to create professional and personalized emails effortlessly using Gemini Flash.

Key Points:

1. **Problem Statement:**
    - Many users struggle with writing professional, well-structured emails for various purposes .
    - Drafting emails manually can be time-consuming and lack consistency.
2. **Proposed Solution:**
    - An AI-powered web application that generates personalized emails using Gemini Flash API.
    - The app will allow users to input key details such as subject, recipient, purpose, and tone to generate customized emails.
    - Features include copying emails, clearing input fields, and selecting AI models for generation.
3. **Target Users:**
    - Professionals needing formal and business emails.
    - Students & job seekers for application and inquiry emails**.**
    - Businesses & marketers requiring client and outreach emails
4. **Expected Outcome:**
    - A functional AI-personalized Email Generation App that helps users quickly draft well-structured emails with minimal effort.

Phase-2: Requirement Analysis

Objective:

Defining the technical and functional requirements for the AI Personalized Email Generator .

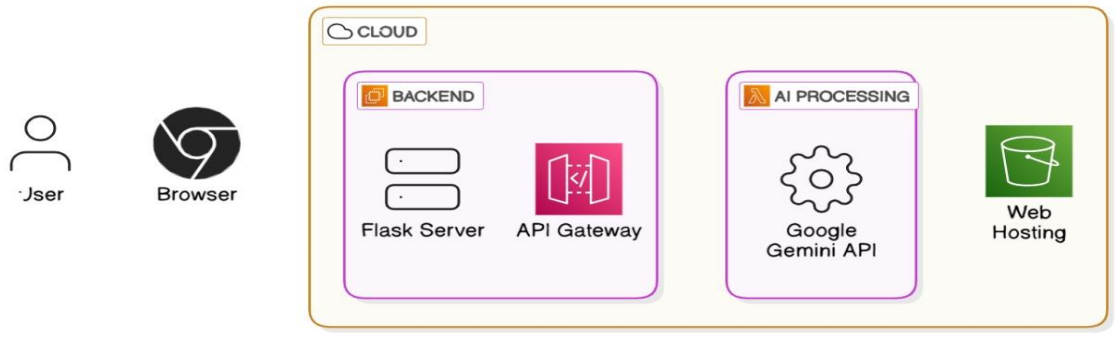
Key Points:

1. **Technical Requirements:**
    - Programming Language: **Python**
    - Backend: **Flask with Google Gemini Flash API integration**
    - Frontend: **HTML, CSS, JavaScript**
    - Database: **Not required initially (API-based queries)**
2. **Functional Requirements:**
    - Ability to generate professional emails using Gemini Flash API.
    - Users can input subject, recipient, purpose, tone, and special instructions.
    - Copy, clear, and modify generated emails easily.
    - Support for multiple tones (polite, formal, informal, enthusiastic).
    - Selection of different AI models (Gemini 1.5 Pro, Gemini 1.5 Flash).
3. **Constraints & Challenges:**
    - Ensuring real-time response from the **Gemini API.**
    - Handling **API rate limits** efficiently.
    - Designing an **intuitive UI** for seamless user experience.

Phase-3: Project Design

Objective:

Developing the architecture and user flow of the application.



Key Points:

1. **System Architecture:**
    - user enters email details such as recipient, subject, purpose, tone, and special instructions via the UI.
    - Query is processed using **Google Gemini API**.
    - The AI processes the input and generates a well-structured email based on the given details.
    - Displaying Results:
    - The frontend displays the generated email in a text box with options to copy, edit, or regenerate if needed.
2. **User Flow:**
    - Step 1: The user enters email details such as subject, recipient, purpose, tone, and instructions.
    - Step 2: The backend sends the structured query to the Gemini Flash API for processing.
    - Step 3: The AI generates a professional email based on the given details.
    - Step 4: The generated email is displayed on the frontend with options to copy, edit, or regenerate.
3. **UI/UX Considerations:**
    - **Minimalist, user-friendly interface** for seamless navigation.
    - **Filters for price, mileage, and features**.
    - **Dark & light mode** for better user experience.

Phase-4: Project Planning (Agile Methodologies)

Objective:

Breaking down the 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	Tester 1	Google API Key, Python, Flask setup	API connection established & working

Sprint 1	Frontend UI Development	🟡 Medium	2 hours (Day 1)	End of Day 1	Tester 2	API response format finalized	Basic UI with input fields
Sprint 2	Vehicle Search & Comparison	🔴 High	3 hours (Day 2)	Mid-Day 2	Tester 3	API response, UI elements ready	Email generator based on user inputs
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	Tester 4	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

Implemention of core features of the AI-personalized email generator.

Key Points:

1. **Technology Stack Used:**
    - **Frontend:** Implement core features of the AI-personalized email generator.
    - **Backend:** Flask (Python), Google Gemini Flash API
    - **Programming Language:** Python
2. **Development Process:**
    - Set up Flask backend to handle email generation requests.
    - Integrate Google Gemini Flash API for AI-powered email creation.
    - Design an interactive UI using HTML, CSS, and JavaScript for a seamless user experience.
    - Implement API request handling, ensuring structured prompts for effective email generation.
    - Enable tone selection (Polite, Formal, Informal, Enthusiastic) for personalized emails.
    - Implement features like copy to clipboard and clearing output.
3. **Challenges & Fixes:**
    - **Challenge:** API response delays
    - **Fix:** Implement caching to store frequently generated emails, reducing the need for repeated API calls and improving response times
    - **Challenge:** Limited API calls per minute
    - **Fix:** Optimize API queries by structuring requests efficiently, minimizing redundant API calls, and ensuring only necessary data is fetched.
    - **Challenge:** Handling errors in API responses
    - **Fix:** Implement robust error handling mechanisms to detect API failures, provide fallback responses, and display meaningful error messages to users.
    - **Challenge:** Ensuring a smooth user experience
    - **Fix:** Optimize frontend performance by enhancing UI responsiveness, reducing input lag, and ensuring a seamless interaction between the user and the AI-generated email output.

Phase-6: Functional & Performance Testing

Test Case ID	Category	Test Scenario	Expected Outcome	Status	Tester
TC-001	Functional Testing	Generate an email with valid inputs	A properly formatted email should be returned	✅ Passed	Tester 1
TC-002	Functional Testing	Request email generation with missing fields	API should return an error message	✅ Passed	Tester 2
TC-003	Performance Testing	API response time under 500ms	API should return results quickly	⚠ Needs Optimization	Tester 3
TC-004	Bug Fixes & Improvements	Handle API failure gracefully	API should return an appropriate error	✅ Fixed	Developer
TC-005	Final Validation	Ensure API handles special characters properly	Email should not break or return errors	❌ Failed - Formatting Issue	Tester 4
TC-006	Deployment Testing	Deploy API and test accessibility	API should be accessible and functional	🚀 Deployed	DevOps