AI G-Mail Generator: Project Documentation

Problem Statement

Manual email composition in professional and organizational settings is time-consuming, repetitive, and lacks consistency. While templates offer some structure, they fail to adapt to tone, context, or personalization. This inefficiency results in a need for an AI-driven solution that can intelligently generate structured, context-aware, and professional emails based on minimal input.

Requirements

Functional Requirements

- The system must allow users to choose between different email types: Personalized, Efficient Communication, and Workflow Emails.
- The system should accept relevant input based on the selected scenario.
- The system must dynamically create a natural language prompt using JavaScript.
- The backend must send the prompt to the Gemini API and return an email response.
- The frontend must display the generated email to the user.

Non-Functional Requirements

- The application should respond within 2–3 seconds.
- The system must work reliably on both desktop and mobile.
- The API key must be handled securely.
- The interface must be simple and intuitive.

User Stories

- As a **business professional**, I want to quickly generate customized emails so I can save time on repetitive communication.
- As a **HR executive**, I want to send structured internal communications to team members.
- As a **student or academic**, I want to send professional messages to staff or companies without spending time drafting emails.

Project Planning and Scheduling

Phases and Duration

- Ideation & Research: 2 days — Identify use cases and goals
- Frontend Development: 3 days — Design forms and prompt builder in JavaScript
- Backend Development: 3 days — Build Flask app and connect Gemini API
- Testing & Debugging: 2 days — Validate input handling and email accuracy
- Deployment: 1 day — Host backend on Render, finalize UI

Tools Used

- Git for version control
- Render for backend hosting
- Local testing using Python & browser-based preview

Code Overview

Frontend (HTML/CSS/JS)

- Dynamic form inputs based on email scenario
- JavaScript function creates a fake natural prompt
- Uses `fetch()` to send data to Flask backend

Backend (Python Flask)

- Receives prompt at `/generate-email` endpoint
- Sends request to Gemini using `google-generativeai`
- Returns structured email text to frontend

Testing

Manual Testing

- Validated each scenario for prompt accuracy
- Checked all form inputs and JS event handlers
- Simulated slow and fast API responses

Edge Case Testing

- Empty inputs (to ensure fallback messaging)
- Invalid time formatting
- Unusual characters or long subject text

All tests confirmed email generation is context-aware and consistent with input values.

Advantages

- High-quality, context-sensitive email generation
- Simple user interface
- Scenarios adapt to user needs dynamically
- Gemini reduces hallucination compared to GPT-Neo
- Works without login, immediate use

Limitations

- Requires internet connection for API call
- Limited to the capabilities of the Gemini model
- Token usage could increase if not optimized
- Backend dependency requires server hosting

Conclusion

The AI G-Mail Generator successfully meets its goal of providing quick, intelligent email generation through a simple interface. By dynamically generating prompts based on structured input and leveraging Google's Gemini model, we created an app that blends efficiency with personalization. From switching models due to hallucinations to optimizing deployment on Render, the project represents a well-rounded solution to the repetitive challenge of email writing.

Future work includes integrating the Gmail API for one-click sending, supporting multi-language output, and storing email history for professional users.
