Project Design

System Architecture

1. Frontend: Built using HTML, CSS, and JavaScript, it provides a clean and responsive UI where users input the required email details.

2. Prompt Generator (JS Logic): Based on selected scenario and input fields, JavaScript creates a structured natural language prompt.

3. Backend: A Flask server (app.py) written in Python handles POST requests from the frontend, sends the prompt to the Gemini API, and returns the generated email.

4. AI Model: Google Gemini is used to process the natural prompt and generate the actual email content.

Design Strategy

To ensure that the application met all three required scenarios effectively, we designed a modular frontend that dynamically adjusts input fields based on the selected email type. Each selection ? whether Personalized Email, Efficient Communication, or Workflow Emails ? activates different input fields relevant to the email category. This approach made the UI both clean and functional.

Design Scenarios

We identified three main use cases for email generation:

- Personalized Email Generation: Users input recipient, event name, date/time, and location.

- Efficient Communication: Quick messages such as thank-you notes, inquiries, and follow-ups.

- Streamlined Workflow: Emails directed at teams or groups, such as newsletters or task assignments.

UI/UX Considerations

The interface is simple, accessible, and intuitive:

- Dropdown to select email type

- Inputs update dynamically based on selection

- Real-time feedback as email is generated

- Responsive layout compatible with desktop and mobile

Technology Stack

- Frontend: HTML, CSS, JavaScript

- Backend: Python Flask

- AI Provider: Google Generative AI (Gemini)

- Hosting: Frontend via GitHub Pages (optional), backend via Render

Security & Data Handling

- API key is securely stored (either hardcoded during test phase or through environment variables in deployment)

- User data is not stored or logged, ensuring privacy