**Hackathon Project Phases Template**  for the **AutoSage App** project.

**Hackathon Project Phases Template**

**Project Title: AI personalized Email Generator**

**AutoSage App Using Gemini Flash**

**Team Name: AI Biters**

**Team Members:**

* SHIVA KUMAR
* YUVARAJ
* SUDHANWA
* SHANKAR

**Phase-1: Brainstorming & Ideation**

**Objective:**

 AI Personalized Email Generator is an innovative project designed to revolutionize email communication by automating the process of drafting personalized emails

**Key Points:**

1. **Problem Statement: Writing personalized emails takes a lot of time,  
    Many people struggle to write emails that feel personal without spending too much time on each one. This is a big issue for people like salespeople, event   organizers, or customer service teams who need to send a lot of personalized         emails quickly.**
2. **Proposed Solution:**The AI email generator would help by automatically creating email drafts for you. You would just provide some basic info, like who the email is for and what it's about, and the AI would write the email for you
3. **Target Users:**

**Students and Educators**

**Description**: Students and teachers who need to draft emails to professors, classmates, or parents.

**Needs**: Writing clear, formal emails for academic purposes while saving time.

**Customer Support Teams**

**Description: Customer service agents who handle numerous queries and need to respond consistently and quickly.**

**Needs: Efficient and consistent responses to a wide variety of customer inquiries.**

**Features of Interest: Pre-written response templates, automatic suggestion for responses based on email content, personalization options.**

1. **Expected Outcome:  
   The expected outcomes for the AI email generator are centered around improving productivity, enhancing communication quality, and reducing time spent on routine email tasks. Here are some key outcomes like.**

**->Time Savings: Reduced Email Drafting Time and Faster Response Times**

**->Scalability: Adaptability for Various Industries: Whether used by small businesses, large corporations, or individual freelancers, the AI will adapt to different use cases, making it scalable across industries.**

**Phase-2: Requirement Analysis**

**Objective:**

Define the technical and functional requirements for the AutoSage App.

**Key Points:**

1. **Technical Requirements:**
   * Programming Language: **Python**
   * Backend: **Google Gemini Flash API**
   * Frontend: **Streamlit Web Framework**
   * Database: **Not required initially (API-based queries)**

**Functional Requirements:  
->Email Composition & Generation: Email Structure & Formatting, Response Suggestions, Tone Adjustment, Customizable Templates etc..**

**->Error Checking: Tone Check ,Readability Analysis, Spelling and Grammar Check etc.**

**->Email Management: Template Library, Save & Edit, Drafts Send Options etc..**

**->Learning and Adaptation: Personalized Suggestions, Custom Feedback:**

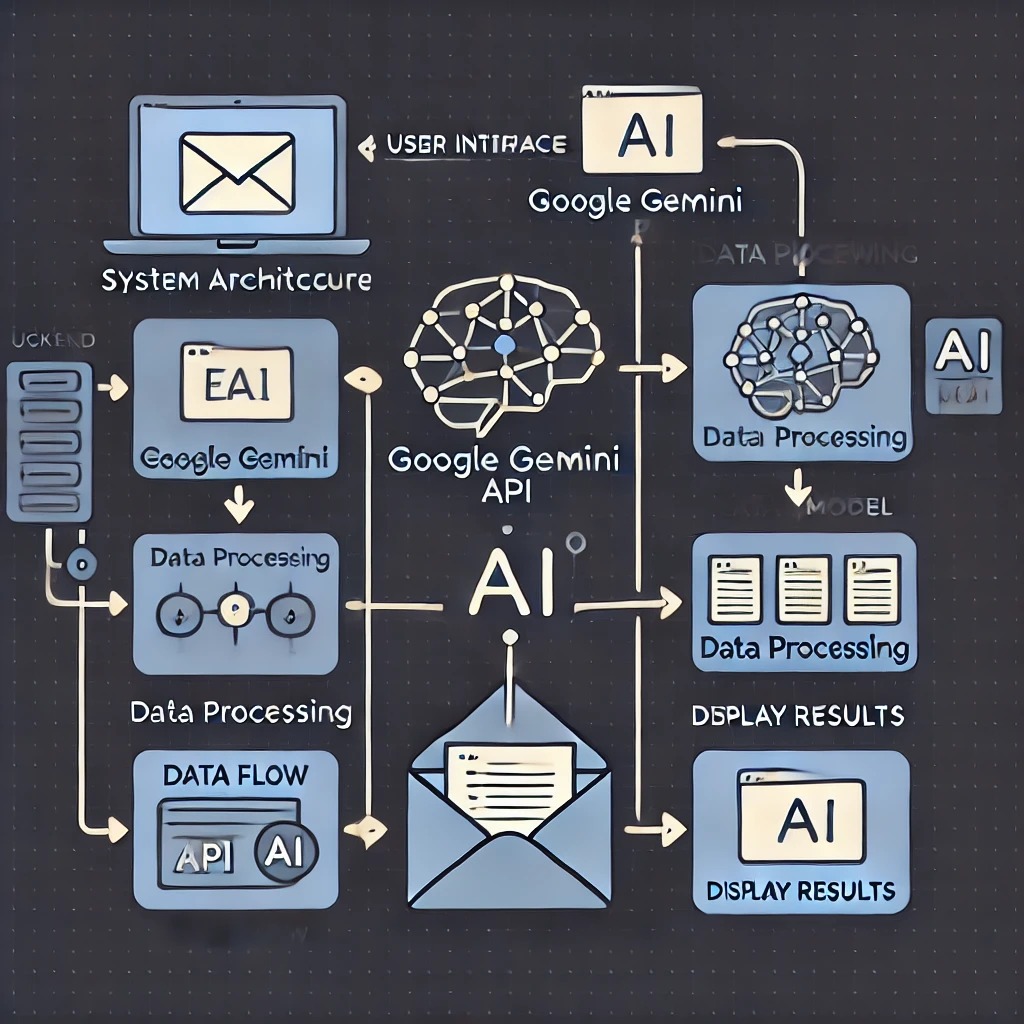
**Constraints & Challenges:  
Data Privacy & Security: Handling sensitive user data such as email content, personal information, and recipient details can create security and privacy concerns**

**Quality and Accuracy of AI-Generated Contentoncerns: Achieving a high level of accuracy in tone, personalization, and intent recognition is difficult for an AI model, particularly in complex or sensitive email situations.**

**Tone and Sentiment Sensitivity: Tone detection algorithms may struggle with understanding the subtleties of human emotions or interpersonal relationships, especially in non-standard communication styles.**

**Phase-3: Project Design**

**Objective:**



Develop the architecture and user flow of the application.

**Key Points:**

1. **System Architecture:**
   * User enters Email related query via UI.
   * Query is processed using **Google Gemini API**.
   * AI model fetches and processes the data.
   * The frontend displays **recipicent’s name ,event name, event date,event location and tone of the Email**
2. **User Flow:**
   * Step 1: User enters a query (e.g.,”jhon doe,annual tech summit,march 15,2025 , formal “)
   * Step 2: The backend **calls the Gemini Flash API** to retrieve recipient’s data.
   * Step 3: The app processes the data and **displays results** in an easy-to-read format.
3. **UI/UX Considerations:**
   * **Minimalist, user-friendly interface** for seamless navigation.
   * Easy to understand the emails generated.

**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 | SHIVA | Google API Key, Python, Streamlit setup | API connection established & working |
| Sprint 1 | Frontend UI Development | 🟡 Medium | 2 hours (Day 1) | End of Day 1 | YUVARAJ | API response format finalized | Basic UI with input fields |
| Sprint 2 | Error Handling & Debugging | 🔴 High | 1.5 hours (Day 2) | Mid-Day 2 | YUVRAJ | API logs, UI inputs | Improved API stability |
| Sprint 3 | Testing & UI Enhancements | 🟡 Medium | 1.5 hours (Day 2) | Mid-Day 2 | SUDHANWA  AND  SHANKAR | 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 AI EMAIL GENERATOR

**Key Points:**

1. **Technology Stack Used:**
   * **Frontend:** Streamlit
   * **Backend:** Google Gemini Flash API
   * **Programming Language:** Python
2. **Development Process:**
   * Implement **API key authentication** and **Gemini API integration**.
   * Develop an AI Email generator for specific usage.
   * Optimize recipients data and tone of mail **queries for generation of Email**
3. **Challenges & Fixes:**
   * **Challenge:** Delayed API response times.  
      **Fix:** Implement **caching** to store frequently queried results.
   * **Challenge:** Limited API calls per minute.  
      **Fix:** Optimize queries to fetch **only necessary data**.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test Case ID | **Category** | Test Scenario | Expected Outcome | Status | Tester |
| TC-001 | **Functional Testing** | Query: "Generate a formal email in English" | A formal email in English should be generated based on input. | ✅ Passed | Tester 1 |
| TC-002 | Functional Testing | Query: "Generate a friendly email in Hindi" | A friendly email in Hindi should be generated based on input. | ✅ Passed | Tester 2 |
| TC-003 | Performance Testing | Check API response time when generating emails | API should return email content within 500ms. | ⚠ Needs Optimization | Tester 3 |
| TC-004 | Bug Fixes & Improvements | Fix issues with incorrect email generation in Telugu | AI should generate accurate Telugu email content. | ✅ Fixed | Developer |
| TC-005 | Final Validation | Ensure the UI is responsive on mobile and desktop devices. | UI should be fully responsive and functional across devices. | ❌ Failed - UI broken on mobile | Tester 2 |
| TC-006 | Deployment Testing | Deploy the app using Streamlit Sharing | App should be accessible online via Streamlit Sharing. | 🚀 Deployed | DevOps |

**Final Submission**

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

v