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

# **Hackathon Project Phases Template**

## **Project Title:**

**AI Personalized Email Generator**

## **Team Name:**

(Email Enchanters)

## **Team Members:**

* B.Himabindu
* A.Kavya Naga Sree
* T.Haripriya
* M.Harshitha

## **Phase-1: Brainstorming & Ideation**

### **Objective:**

### Develop an AI-powered Personalized Email Generator that automates the drafting of personalized emails, enhancing communication efficiency and effectiveness.

### **Key Points:**

1. **Problem Statement:**
   * Email communication often requires personalization to faster relationships in professional settings. Drafting tailored emails can be time-consuming and challenging for users.
   * The AI Personalized Email Generator will automate this process, generating customized email content based on user inputs, significantly reducing the effort and time involved.
2. **Proposed Solution:**
   * An intelligent application that leverages advanced language models (like GPT-4) to create personalized emails. Users will input details such as the recipient's name, event specifics, and desired tone (e.g., formal, professional, casual), and the application will output a coherent email draft.
3. **Target Users:**
   * Professionals who frequently communicate via email and require timely, personalized messages.
   * Businesses looking to enhance communication efficiency while maintaining a personal touch with clients and colleagues.
   * Anyone needing to send custom emails for occasions such as invitations, follow-ups, or greetings.

1. **Expected Outcome:**
   * A user-friendly application that generates accurate and personalized emails, saving time and effort for users while improving their email communication quality.

## **Phase-2: Requirement Analysis**

### **Objective:**

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

### **Key Points:**

1. **Technical Requirements:**
   * **Programming Language:** Python
   * **Backend:** OpenAI API (GPT-4)
   * **Frontend:** Web application (possibly using frameworks such as Flask or Streamlit)
   * **Database:** Optional, for storing user templates and previous emails.
   * **API Integration:** Seamless integration with the OpenAI API for generating email content.
2. **Functional Requirements:**
   * User input fields for:

 Recipient's name

Occasion/Event name

Special instructions (if any)

Tone of the email (e.g., formal, casual, professional)

 Generate a personalized email draft based on the input data.

* + Allow users to edit the generated email before sending.
  + Option to save frequently used email templates for future use.
  + User-friendly interface with clear instructions for input.

1. **Constraints & Challenges:**
   * Ensuring accurate and contextually relevant email generation based on diverse user inputs.
   * Managing API response times and potential rate limits from the OpenAI API.
   * Requirement for robust error handling for invalid inputs or API failures.
   *  Providing a smooth and responsive user experience in the web interface.

## **Phase-3: Project Design**

### **Objective:**

Develop the architecture and user flow of the AI Personalized Email Generator application.



### **Key Points:**

1. **System Architecture:**
   * **Frontend:** User interface (web application) where users input details for email generation (name, occasion, tone).
   * **Backend:**

Input is captured from the frontend and sent to the OpenAI API.

The API processes the input using the GPT-4 model to generate the email

content

* + **Output:** The generated email is sent back to the frontend for display and editing.

1. **User Flow:**
   * **Step 1:** User navigates to the web application and sees input fields for recipient name, occasion, tone, and any special instructions.
   * **Step 2:** User enters the required information and submits the form.
   * **Step 3:** The application makes a request to the OpenAI API with the user’s input.
   * **Step 4:** The API returns a generated email draft.
   * **Step 5:** The user can review, edit, and finalize the email before sending it.
2. **UI/UX Considerations:**
   * **User-Friendly Interface:** Intuitive design to ensure ease of use for all users.
   * **Input Validation:** Prompt users for required information and provide error messages for incorrect entries.
   * **Templates and History:** Options to create and store email templates for repeated use.
   * **Responsive Design:** Ensure the application works smoothly on both desktop and mobile devices.
   * **Accessibility Features:** Implement features for users with disabilities, such as screen reader compatibility and keyboard navigation.

.

## 

## **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 | 4 hours (Day 1) | End of Day 1 | himabindu | OpenAI API Key | API connection established & working |
| Sprint 1 | Frontend UI Development | 🟡 Medium | 3 hours (Day 1) | End of Day 1 | kavya | API response format finalized | Basic ui with input fields |
| Sprint 2 | Core Email Generation Logic | 🔴 High | 5 hours (Day 2) | Mid-Day 2 | haripriya | UI Development | Search functionality with filters |
| Sprint 2 | Error Handling & Debugging | 🔴 High | 2 hours (Day 2) | Mid-Day 2 | Member 1&2 | API integration | Improved API stability |
| Sprint 3 | Testing & User Feedback | 🟡 Medium | 4 hours (Day 2) | Mid-Day 2 | Member 3& 4 | API response, UI layout completed | UI & Email Logic |
| Sprint 3 | UI Enhancements & Final Review | 🟢 Low | 2 hour (Day 2) | End of Day 2 | Entire Team | Testing | Demo-ready project |

### 

### **Sprint Planning with Priorities**

### **Sprint 1 – Environment Setup & API Integration (Day 1)**

.**(🔴 High Priority)** Set up the development environment (installing necessary software, libraries, and tools).

**(🔴 High Priority)** Integrate OpenAI API for email generation. Obtain API keys and ensure successful connection to the API.

**(🟡 Medium Priority)** Create a basic frontend UI with input fields for the user to enter email details (recipient name, occasion, tone).

### **Sprint 2 – Core Features & Implementation (Day 2)**

**(🔴 High Priority)** Implement email generation logic: Develop the backend functionality that processes user input, communicates with the OpenAI API, and retrieves generated email drafts.

**(🔴 High Priority)** Conduct error handling and debugging: Ensure that the application can handle various user input scenarios and API errors gracefully.

### **Sprint 3 – Testing, Enhancements & Submission (Day 2)**

**(🟡 Medium Priority)** Testing of the generated emails: Validate that the email content aligns with user expectations and fine-tune the API requests if needed.

**(🟡 Medium Priority)** UI enhancements: Refine the user interface based on feedback from the testing phase, making necessary improvements for usability and aesthetics.

**(🟢 Low Priority)** Final preparation for presentation: Compile project documentation, create a demo video, and prepare slides for the final presentation.

## **Phase-5: Project Development**

### **Objective:**

Implement core features of the AI Personalized Email Generator.

### **Key Points:**

1. **Technology Stack Used:**
2.  **Frontend:** Streamlit (for building the web application).
3.  **Backend:** OpenAI API (for generating personalized email content).
4.  **Programming Language:** Python (for handling both frontend and backend logic).

   * **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**.

## **Phase-6: Functional & Performance Testing**

### **Objective:**

Ensure that the works as expected.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Case ID** | **Category** | **Test Scenario** | **Expected Outcome** | **Status** | **Tester** |
| TC-001 | Functional Testing | Generate an email for a known recipient with details | Email should contain personalized content for the recipient. | ✅ Passed | Tester 1 |
| TC-002 | Functional Testing | Generate an email with missing recipient name | Generate an error message or prompt for name input. | ❌ Failed - No error message | Tester 2 |
| TC-003 | Performance Testing | Measure response time for email generation | Email generation should take less than 2 seconds. | ✅ Passed | Tester 3 |
| TC-004 | Bug Fixes & Improvements | Measure response time for email generation | Generated emails should have accurate content relevant to the prompts provided. | ✅ Fixed | Developer |
| TC-005 | Final Validation | Ensure the system can handle multiple concurrent requests | System should maintain performance and generate emails accurately. | ❌ Failed - Issues at high load | Tester 2 |
| TC-006 | Deployment Testing | Host the application using a cloud service | App should be accessible online and function correctly. | 🚀 Deployed | DevOps |