# Hackathon Project Phases Template

**Project Title:**

AI PERSONALIZED EMAIL GENERATOR

**Team Name:**

THUNDERBOLTZZZ

**Team Members:**

* DIVYA MANASA
* ROHITH KARTHIKEYA
* SHIVA KRISHNA
* SHARAN KUMAR
* ARAVINDU

## Phase-1: Brainstorming & Ideation

**Objective:**

Develop an AI-Personalized email generator using Gemini Flash to help users in communication which contains various features and writing down message and sending it to the concerned person through email.

**Key Points:**

1. **Problem Statement:**

* Many businesses and individuals struggle to craft personalized emails that resonate with their recipients. With the rise of digital communication, email has become an essential tool for marketing, sales, and customer support. However, generic, impersonal emails often end up in spam folders or get deleted without being read.

1. **Proposed Solution:**

* An AI Powered application using Gemini Flash to Generate email context according to the criteria and specifications.

1. **Target Users:**

* **EMPLOYEE** looking for leave.
* **STUDENTS** writing formal mails to their respective educational institution.
* **BUSINESS ORGANISATONS**  communicating with clients.

1. **Expected Outcome:**

* 1. A functional **AI- personalized email generator** that generates the desired email according to the user requirements and automatically sends to the selected receiver mail id.

## Phase-2: Requirement Analysis

**Objective:**

Define the technical and functional requirements for the email generator.

**Key Points:**

1. **Technical Requirements:**

* 1. Programming Language: **Python**

○ Backend: **Google Gemini Flash API**

○ Frontend: **Streamlit Web Framework**

○ Database: **Not Required**

1. **Functional Requirements:**

* 1. Ability to **write body of the mail and send to receivers mail id** using Gemini Flash.

○ Displays several options for writing a mail in different categories.

.

1. **Constraints & Challenges:**

* 1. Ensuring real-time updates from **Gemini API**.

○ Handling **API rate limits** and optimizing API calls.

○ Providing a **smooth UI experience** with Streamlit.

## Phase-3: Project Design

**Objective:**

* Automate the email drafting process.
* Enhance personalization based on user data.
* Improve engagement rates by tailoring content.
* Reduce time spent on email composition.

**Key Points:**

1. **User Input:**

* User provides email details (subject, keywords, tone).
* Optionally integrates CRM data for personalization.

2. **Data Processing & Analysis:**

* Extract relevant recipient details.
* Analyse past emails for consistency.

3. **AI-Powered Email Generation:**

* Generate email content using NLP models (GPT, LLaMA, etc.).
* Adjust tone and structure based on input.
* Email is sent via integrated platforms (Gmail, Outlook, etc.).
* Option to save drafts or schedule emails.

4. **Feedback & Learning Loop:**

* AI collects feedback for improvements.
* Model fine-tunes based on user interactions

## Phase-4: Project Planning (Agile Methodologies)

**Objective:**

Break down development tasks for efficient completion.

Here’s a **Project Planning Table** for the **AI Personalized Email Generator Model** within a **3-day timeframe**:

|  |  |  |  |
| --- | --- | --- | --- |
| **Phase** | **Duration** | **Key Tasks** | **Deliverables** |
| **Day 1 - Research & Planning** | 3 Hours | - Define project scope & objectives - Identify key features & tech stack - Conduct market & competitor analysis | Project requirement document Tech stack selection Initial design prototype |
| **Day 1 - Data Collection & Model Setup** | 5 Hours | - Gather business email datasets - Preprocess data and clean text - Select and configure AI model (GPT, LLaMA) | Cleaned dataset Model setup completed |
| **Day 1 - Backend Development** | 4 Hours | - Develop API for email generation - Set up database for user preferences | Functional API & Database Schema |
| **Day 2 - AI Model Training & Testing** | 6 Hours | - Fine-tune model on email dataset - Implement reinforcement learning for personalization - Run initial test cases | Trained AI model Initial performance evaluation |
| **Day 2 - Frontend Development** | 6 Hours | - Build a minimal UI (React.js/Vue.js) - Implement email templates & customization options | Basic functional UI Interactive email customization panel |
| **Day 3 - Testing & Optimization** | 5 Hours | - Conduct unit & integration testing - Improve email quality & personalization - Gather feedback from internal testers | Debugged and optimized AI model Performance reports |
| **Day 3 - Deployment & Launch** | 5 Hours | - Deploy AI model to cloud/hosting service - Integrate with email clients (Gmail, Outlook) - Release MVP for real-world testing | Live product Initial user onboarding |
| **Day 3 - Post-Launch Enhancements** | 2 Hours | - Monitor user feedback & refine AI model - Add minor feature enhancements (A/B testing, templates) | Improved personalization features User-driven updates |

**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 3)**

**(**🟡 **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-personalized email generator.

**Key Points:**

1. **Technology Stack Used:**

* 1. **Frontend:** Streamlit

○ **Backend:** Google Gemini Flash API

○ **Programming Language:** Python

1. **Development Process:**

* 1. Implement **API key authentication** and **Gemini API integration**.

○ Develop **vehicle comparison and maintenance tips logic**. ○ Optimize **search queries for performance and relevance**.

1. **Challenges & Fixes:**

* 1. **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 model works as expected.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Case ID** | **Category** | **Test Scenario** | **Steps to Execute** | **Expected Outcome** | **Status** |
| TC-01 | **Functionality** | Generate a formal business email | 1. Input subject, keywords, and select "Formal" tone 2. Click generate | AI generates a structured and professional email | Passed |
| TC-02 | **Functionality** | Generate a casual email | 1. Input subject, keywords, and select "Casual" tone 2. Click generate | AI generates a friendly and engaging email | Passed |
| TC-03 | **Functionality** | Generate a follow-up email | 1. Provide details about a previous conversation | AI generates a coherent follow-up email | Passed |
| TC-04 | **Personalization** | Use recipient details for a personalized email | 1. Input recipient name and preferences 2. Click generate | Email includes recipient-specific details | Passed |
| TC-05 | **Usability** | User edits AI-generated email | 1. Generate email 2. Make manual edits 3. Save changes | AI adapts and learns from edits for future | Passed |
| TC-06 | **Usability** | UI responsiveness | 1. Load email generator on web interface 2. Interact with settings | Smooth and lag-free UI experience | Passed |
| TC-07 | **Integration** | API integration with Gmail | 1. Connect Gmail 2. Send AI-generated email | Email successfully sent via Gmail | Passed |
| TC-08 | **Integration** | CRM data retrieval | 1. Connect CRM 2. Generate an email based on CRM history | AI retrieves relevant past interactions | Passed |
| TC-09 | **Performance** | Email generation speed | 1. Input details 2. Click generate | Email is generated within 2-3 seconds | Passed |