Hackathon Project Phases

Project Title:

MailMatic - AI-Powered Email Generator Using Hugging Face API

Team Name:

MAILMATIC

Team Members:

- Avula Navya
- Bagothula Nityasri
- Batthula Siri Jahnavi
- Dondapati Devi Sri Chandana
- Yeluka Sowmya

Phase-1: Brainstorming & Ideation

Objective:

Develop an AI-powered email generation tool using Hugging Face API to help users create professional, well-structured emails with ease.

Key Points:

1. **Problem Statement:**
    - Many users struggle with writing professional emails, requiring assistance in structuring and wording them effectively.
    - Users need a simple tool to generate emails based on different contexts, recipients, and tones.

2. **Proposed Solution:**

   - A web-based AI-powered application using **Hugging Face API** and **Streamlit** to generate well-structured emails.
   - The tool allows users to input **email purpose, recipient details, and salutation preferences**, and generates a ready-to-send email.

3. **Target Users:**

   - Professionals and business users who need assistance in drafting emails.
   - Students and job seekers preparing formal email communication.
   - Anyone looking to automate email creation for different use cases..

4. **Expected Outcome:**

   - A functional AI-powered email generator that allows users to customize email content based on their needs.

Phase-2: Requirement Analysis

Objective:

Define the technical and functional requirements for MailMatic.

Key Points:

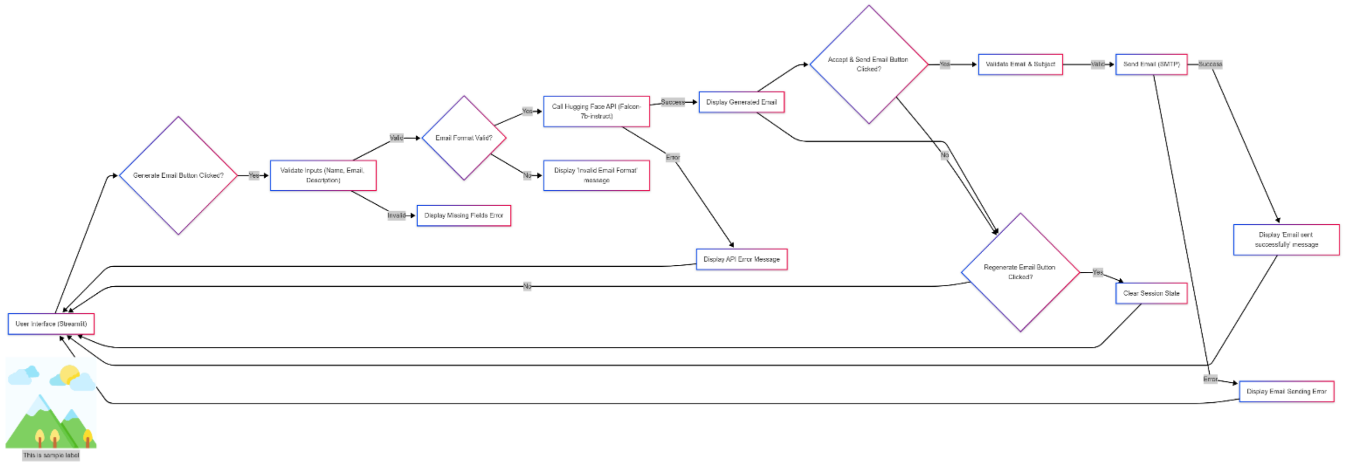
1. **Technical Requirements:**
   - **Programming Language:** Python
   - **Backend:** Hugging Face API for text generation
   - **Frontend:** Streamlit Web Framework
   - **Database:** Not required initially (API-based queries)
2. **Functional Requirements:**
   - Ability to generate professional emails based on user input.
   - Customization options including **salutation, recipient email, and tone**.
   - Option to add **multiple recipients** dynamically.
   - Allow users to **copy or download** the generated email.
3. **Constraints & Challenges:**

   - Ensuring API responses are **contextually relevant**.
   - Handling API rate limits and optimizing API calls.
   - Providing a smooth **UI experience** with Streamlit.

Phase-3: Project Design

Objective:

Develop the architecture and user flow of the application.


User Interface (Streamlit)
Generate Email Button Clicked?
Yes
Validate Inputs (Name, Email, Description)
Valid
Invalid
Display Missing Fields Error
Email Format Valid?
Yes
No
Call Hugging Face API (Falcon-7b-instruct)
Display 'Invalid Email Format' message
Success
Error
Display Generated Email
Display API Error Message
Accept & Send Email Button Clicked?
Yes
No
Validate Email & Subject
Valid
Send Email (SMTP)
Success
Error
Display 'Email sent successfully' message
Display Email Sending Error
Regenerate Email Button Clicked?
Yes
No
Clear Session State
This is sample label

Key Points:

1. **System Architecture:**
    - User inputs email details via **Streamlit UI**.
    - The input query is processed by the **Hugging Face API**.
    - AI model generates a structured email.
    - The **frontend displays** the generated email with an option to copy or download.
2. **User Flow:**
    - User enters **short description**, selects **salutation**, and **adds recipient(s)**.
    - The backend **calls the Hugging Face API** to generate email content.
    - The app **displays the AI-generated email**.
    - User can **edit, copy, or download** the email.
3. **UI/UX Considerations:**
    - **Minimalist, user-friendly interface** for seamless navigation.
    - **Dynamic recipient input fields** with an "+" button to add multiple recipients.
    - **Copy/download options** for ease of use.

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	Navya	Google API Key, Python, Streamlit setup	API connection established & working
Sprint 1	Frontend UI Development	⍰ Medium	2 hours (Day 1)	End of Day 1	DeviSriChandhana	API response format finalized	Basic UI with input fields
Sprint 2	Email Generation Logic	⍰ High	3 hours (Day 2)	Mid-Day 2	Sowmya&DeviSriChandana	API response, UI elements ready	AI-generated emails with user input
Sprint 2	Error Handling & Debugging	⍰ High	1.5 hours (Day 2)	Mid-Day 2	Nityasri&Siri Jahnavi	API logs, UI inputs	Improved API stability
Sprint 3	Testing & UI Enhancements	⍰ Medium	1.5 hours (Day 2)	Mid-Day 2	Navya	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 Plan for Mailmatics – AI-Powered Email Generator

🛠 Sprint 1 – Setup & Integration (Day 1)

- ● **High Priority –** Set up the development environment in Google Colab & install required dependencies.
- ● **High Priority –** Integrate the Hugging Face API for AI-powered email generation.
- ▯ **Medium Priority –** Build a basic UI using Streamlit with input fields for recipient email, subject, and short description.

🚀 Sprint 2 – Core Features & Debugging (Day 2)

- ● **High Priority –** Implement dynamic input fields to allow multiple recipients.
- ● **High Priority –** Debug API integration issues and handle errors like missing API keys or model loading failures.
- ▯ **Medium Priority –** Add a "Salutation" selection button to personalize emails.

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

- ▯ **Medium Priority –** Test AI-generated email responses, refine the UI, and fix any UI-related bugs.
- ▯ **Low Priority** – Final demo preparation & deployment for submission, ensuring smooth performance.

Phase-5: Project Development

Objective:

Implement core features of the AutoSage App.

Key Points:

1. **Technology Stack Used:**
   - **Frontend:** Streamlit
   - **Backend:** Hugging Face API
   - **Programming Language:** Python
2. **Development Process:**
   - Implement **API key authentication** and Hugging Face API integration.
   - Develop **email generation logic** and input handling.
   - Optimize API calls for performance and relevance.
3. **Challenges & Fixes:**
   - **Challenge:** API response delays
     Fix: Optimize API queries and minimize unnecessary calls
   - **Challenge:** UI responsiveness
     Fix: Ensure dynamic elements work across devices

Phase-6: Functional & Performance Testing

Objective:

Ensure that the AutoSage App works as expected.

Test Case ID	Category	Test Scenario	Expected Outcome	Status	Tester
TC-001	Functional Testing	User enters an email description	AI-generated email should be displayed	✓ Passed	Tester 1

TC-002	Functional Testing	User adds multiple recipients	Email should include all recipient emails	✅ Passed	Tester 2
TC-003	Performance Testing	API response time under 500ms	Email should generate quickly	⚠ Needs Optimization	Tester 3
TC-004	Bug Fixes & Improvements	Fixed incorrect AI responses	Emails should be relevant	✅ Fixed	Developer
TC-005	Final Validation	Ensure UI is responsive across devices.	UI should work on mobile & desktop.	❌ Failed - UI broken on mobile	Tester 2
TC-006	Deployment Testing	Host the app using Streamlit Sharing	App should be accessible online.	⍰ Deployed	DevOps

---

Final Submission

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