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

Hackathon Project Phases Template

Project Title:

Blog Generation Using LLaMA 2 and Streamlit

Team Name:

Byte Stormers

Team Members:

- Shaik Zuveriya Tabassum
- Vemuri Devi Sai Sudha
- Yadapalli Anuradha

Phase-1: Brainstorming & Ideation

Objective:

This project aims to create a customizable, cost-effective, and privacy-focused AI blog generator using LLaMA 2 and Streamlit, ensuring ease of use and scalability for various users.

Key Points:

1. **Problem Statement:**

High Cost & Dependency on Proprietary AI Tools – Many blog generation tools rely on expensive, subscription-based AI models, limiting access for individuals and small businesses.

Lack of Customization & Privacy – Existing solutions often lack real-time customization options and may store user data on cloud servers, raising privacy concerns.

2. **Proposed Solution:**

Cost-Effective & Open-Source – Uses LLaMA 2 instead of expensive proprietary AI models.

Interactive & Customizable – Allows real-time adjustments to tone, style, and length via Streamlit.

Privacy-Focused – Supports local deployment, ensuring data security.

User-Friendly & Scalable – Simple interface accessible to both developers and non-developers.

Versatile Application – Suitable for bloggers, businesses, and researchers to generate high-quality content efficiently.

3. **Target Users:**

Businesses & Marketers – Create engaging marketing content, articles, and product descriptions.

Students & Researchers – Summarize research papers, draft reports, and write academic content.

Educators & Trainers – Develop educational materials and learning resources with ease.

Tech Enthusiasts & Developers – Experiment with AI-generated content and build innovative applications.

4. **Expected Outcome:**

This provides a cost-effective, customizable, and secure AI-powered blog generator using LLaMA 2 and Streamlit.

It offers real-time content adjustments, ensures data privacy, and serves bloggers, businesses, and researchers with an efficient and user-friendly writing too.

Phase-2: Requirement Analysis

Objective:

The objective of requirement analysis is to **define the technical and functional needs** of the AI-powered blog generator, ensuring **feasibility, scalability, and user-friendliness**. It helps in selecting the right **technology stack, optimizing performance, and ensuring data privacy** for an efficient and effective system..

Key Points:

1. **Technical Requirements:**

- **LLaMA 2 & Python** for AI-driven text generation.
- **Streamlit** for an interactive web UI.
- **Libraries:** Transformers (Hugging Face), PyTorch, Requests/JSON.
- **Deployment:** Streamlit Cloud, Hugging Face Spaces, or local hosting.
- **Version Control:** Git/GitHub for collaboration.

2. **Functional Requirements:**

- **User Input & Customization** – Adjust tone, style, and length.
- **Real-Time Blog Generation** – Instant AI-powered content creation.
- **Privacy & Security** – Supports local deployment.
- **User-Friendly Interface** – Accessible for all users.
- **Optimized Performance** – Fast and efficient processing.

3. **Constraints & Challenges:**

1. **Accuracy & Bias** – AI-generated content may lack factual accuracy, coherence, or reflect biases from training data.
2. **Performance & Privacy** – High computational resources are needed for local execution, and cloud deployment may pose security risks.
3. **Creativity & Dependence** – AI may struggle with highly creative writing, and future updates to LLaMA 2 could impact functionality.

Phase-3: Project Design

Objective:

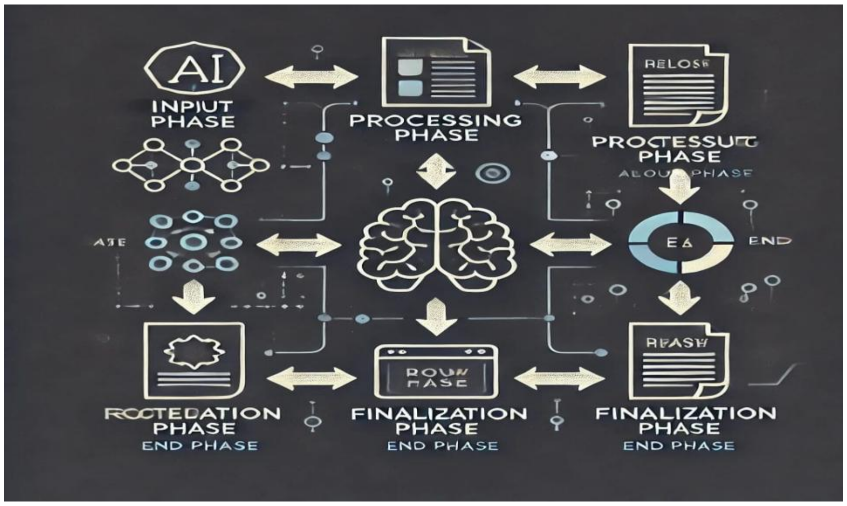
Architecture:

- **Frontend:** Streamlit-based UI for user input and interaction.

- **Backend:** LLaMA 2 processes inputs via Hugging Face & PyTorch.
- **Deployment:** Hosted on **Streamlit Cloud, Hugging Face Spaces, or locally**.
-

Key Points:

1. **System Architecture Diagram:**



2. **User Flow:**

1. **Open the Application** – Access the AI-powered blog generator via the web interface.
2. **Enter Blog Topic & Preferences** – Input a topic and select tone, style, and length.
3. **Generate Blog Content** – Click the "Generate" button to process input using LLaMA 2.
4. **View & Edit Content** – Review the generated blog and make refinements if needed.
5. **Copy or Download** – Save or copy the final content for further use.

UI/UX Considerations:

- **Simple & Responsive Design** – Ensures smooth navigation across devices.
- **Customization & Real-Time Feedback** – Users can adjust **tone, style, and length** with instant previews.
- **Privacy & Performance** – Optimized for **fast processing** while ensuring **data security**.

Phase-4: Project Planning (Agile Methodologies)

Objective:

- Break down the tasks using Agile methodologies.

Key Points:

Sprint Planning

Sprint	Task	Priority	Duration	Deadline	Assigned To	Dependencies	Expected Outcome
Sprint 1	Environment Setup & LLaMA 2 Integration	High	6 hours (Day 1)	End of Day 1	Member 1	Python, Hugging Face API, PyTorch	AI model integrated and working
Sprint 1	Frontend UI Development (Streamlit)	Medium	2 hours (Day 1)	End of Day 1	Member 2	UI design finalized	Basic UI with input fields
Sprint 2	Blog Generation & Customization Features	High	3 hours (Day 2)	Mid-Day 2	Member 1 & 2	AI model response, UI components	Customizable content generation enabled
Sprint 2	Error Handling & Debugging	High	1.5 hours (Day 2)	Mid-Day 2	Member 1 & 3	AI logs, UI input validation	Improved AI response & stability
Sprint 3	Testing & UI Enhancements	Medium	1.5 hours (Day 2)	Mid-Day 2	Member 2 & 3	AI 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	Fully functional system	Demo-ready project, deployed & tested

Sprint 1: Setup & Initial Development

- The first sprint focuses on setting up the **development environment** and integrating **LLaMA 2** for blog generation.
- The **frontend UI** is developed using **Streamlit**, providing basic input fields for user interaction.

Sprint 2: Core Functionality & AI Processing

- The AI-powered **blog generation and customization features** are implemented, allowing users to adjust tone, style, and length.
- **Error handling and debugging** ensure the AI responses are stable and optimized for better performance.

Sprint 3: Testing, UI Enhancements & Deployment

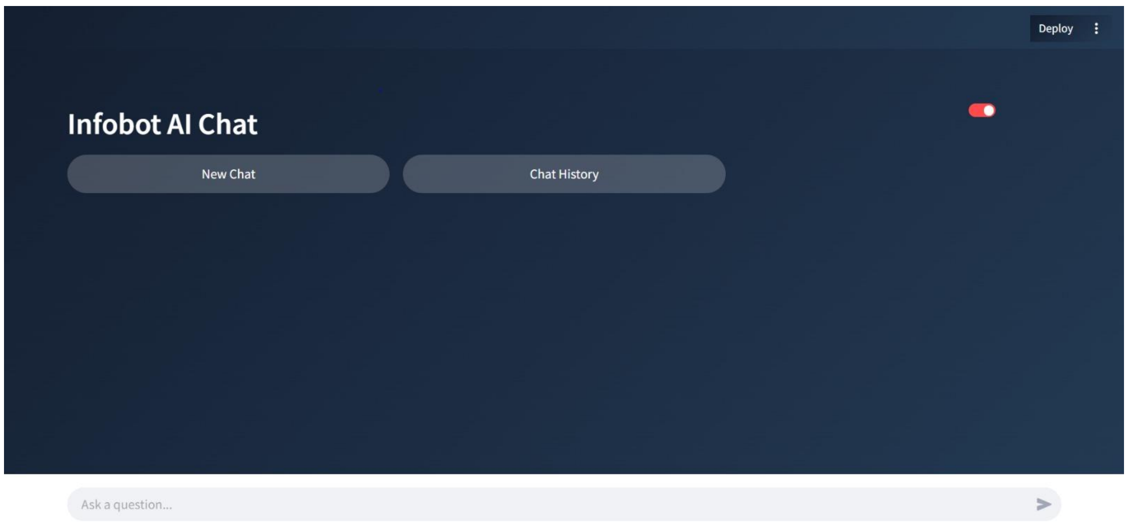
- The user interface is improved for **better responsiveness and experience**.
- Thorough **testing** is conducted, followed by **final deployment** on **Streamlit Cloud or Hugging Face Spaces**.
- The project is made **demo-ready** for presentation and future scalability.

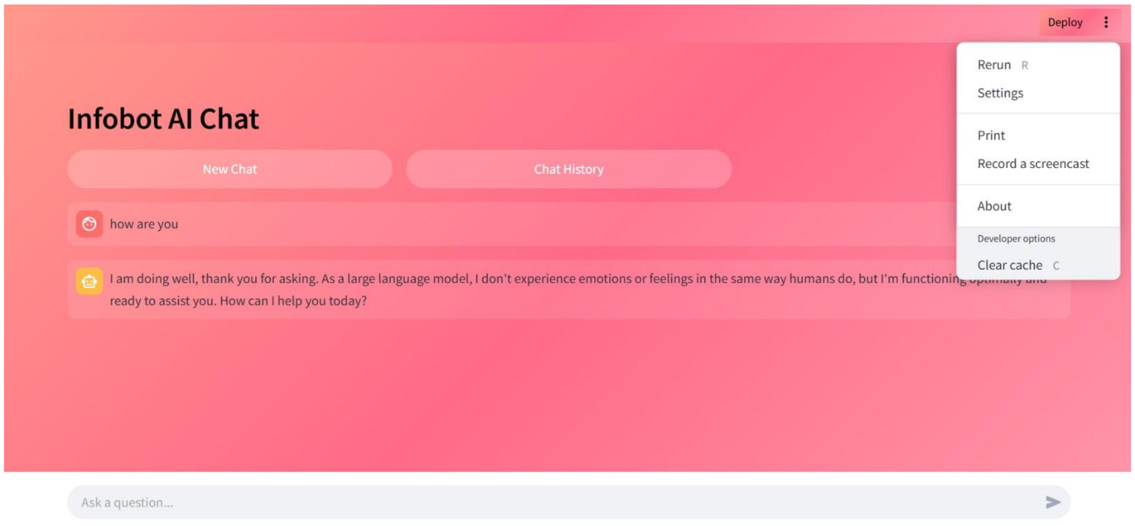
Phase-5: Project Development

Objective:

```
import streamlit as st
import os
import google.generativeai as genai

os.environ['GEMINI_API_KEY'] = 'AIzaSyDhNBPxFPvwArIBlNc1hk0s8JKYmo6-yek'  # Replace
with a secure method in production

genai.configure(api_key=os.environ['GEMINI_API_KEY'])

model = genai.GenerativeModel("gemini-1.5-flash")
chat = model.start_chat()

def get_gemini_response(question):
    response = chat.send_message(question, stream=True)
    return "".join([chunk.text for chunk in response])

st.set_page_config(page_title="Infobot", layout="wide")

if "dark_mode" not in st.session_state:
    st.session_state.dark_mode = True

if st.session_state.dark_mode:
    gradient_bg = "linear-gradient(135deg, #141E30, #243B55)"  # Dark Mode
    text_color = "#FFFFFF"
else:
    gradient_bg = "linear-gradient(135deg, #FF9A8B, #FF6A88, #FF99AC)"  # Light Mode
    text_color = "#000000"


st.markdown(f"""
    <style>
        html, body, [data-testid="stAppViewContainer"], [data-testid="stHeader"],
        [data-testid="stToolbar"], [data-testid="stSidebar"], [data-testid="stFooter"]
{{
            background: {gradient_bg} !important;
            background-attachment: fixed;
            background-size: cover;
            color: {text_color};
        }}

        .stTextInput>div>div>input {{
            border-radius: 10px;
            padding: 12px;
            border: 2px solid white;
            background-color: rgba(255, 255, 255, 0.2);
            color: {text_color};
        }}

        .stButton>button {{
            background: rgba(255, 255, 255, 0.2);
            border: none;
            border-radius: 50px;
            padding: 10px 15px;
            color: white;
            font-size: 18px;
            transition: 0.3s;
        }}
```

Output:





how are you

I am doing well, thank you for asking. As a large language model, I don't experience emotions or feelings in the same way humans do, but I'm functioning optimally and ready to assist you. How can I help you today?

Key Points:

1. **Technology Stack Used:**

Programming Languages:

Python

APIs & Libraries:

LLaMA 2 (Meta AI)

Hugging Face Transformers

PyTorch

Streamlit

Requests & JSON

Deployment & Version Control:

Streamlit Cloud / Hugging Face Spaces

Git & GitHub

2. **Development Process:**

Setup – Install dependencies and configure GitHub.

LLaMA 2 Integration – Load the AI model with Hugging Face & PyTorch.

UI Development – Build a Streamlit interface with customization options.

Backend Implementation – Process inputs and generate blog content.

Testing & Debugging – Fix bugs and optimize performance.

Deployment – Host on Streamlit Cloud or Hugging Face Spaces.

3. **Challenges & Fixes:**

- **High Computational Requirements** – Running LLaMA 2 locally required significant resources.
    - *Solution:* Used **optimized model versions** and leveraged **cloud deployment** on Hugging Face Spaces.
- **Slow Response Time** – AI-generated content processing took longer than expected.
    - *Solution:* Implemented **efficient caching** and **input optimization** to reduce processing time.

- **Content Accuracy & Coherence** – The AI sometimes generated **irrelevant or incoherent** blog content.
    - *Solution :* Fine-tuned prompts and applied **post-processing techniques** to improve content quality.

Phase-6: Functional & Performance Testing

Objective:

- Ensure the LLAMA2 as expected.

Key Points:

Test Case ID	Category	Test Scenario	Expected Outcome	Status	Tester
TC-001	Functional Testing	Generate a blog on "AI in Healthcare".	AI should generate a relevant blog post.	✓ Passed	Tester 1
TC-002	Functional Testing	Customize blog with "Casual" tone.	The blog should match the selected tone.	✓ Passed	Tester 2
TC-003	Performance Testing	AI response time under 500ms.	AI should return results quickly.	⚠ Needs Optimization	Tester 3
TC-004	Bug Fixes & Improvements	Fix incorrect AI-generated content.	Content should be accurate and relevant.	✓ Fixed	Developer
TC-005	Final Validation	Ensure UI is responsive on all devices.	UI should work on mobile & desktop.	✕ Failed - UI issue on mobile	Tester 2
TC-006	Deployment Testing	Deploy app using Streamlit Cloud.	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**