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

Hackathon Project Phases Template

Project Title: Blog generator using LLaMA2 and streamlit

Team Name:

STARBOYS :

Team Members:

- Member 1: Kola Ranjith
- Member 2: A. Vijay kumar
- Member 3: Saddi Ram reddy
- Member 4: Janmula Rakesh

Phase-1: Brainstorming & Ideation

Objective:

To develop an interactive AI-powered blog generator using LLaMA 2 and Streamlit, enabling users to effortlessly generate well-structured, high-quality blog posts tailored to their needs. The platform will provide customization options for tone, style, and length while ensuring SEO optimization and ease of editing

Key Points:

1. **Problem Statement:**
   - Non-technical users find it difficult to interact with AI models effectively.
   - In today's fast-paced digital world, content creation is crucial for businesses, marketers, and individual blogger
2. **Proposed Solution:**

    - Use LLaMA 2 to generate structured, well-written blog posts
    - Provide a user-friendly Streamlit interface for seamless content creation
3. **Target Users:**
    - Independent bloggers looking for quick and engaging content.
    - Niche content creators who need structured blog posts on specialized topics.
    - YouTubers & podcasters who need blog summaries of their content.
4. **Expected Outcome:**
    - The blog generator will deliver fast, high-quality, customizable, and SEO-friendly content while enhancing productivity

---

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)**
2. **Functional Requirements:**
    - Allow users to choose a writing tone (formal, casual, storytelling, persuasive, etc.).
    - Implement prompt engineering for improved content accuracy.

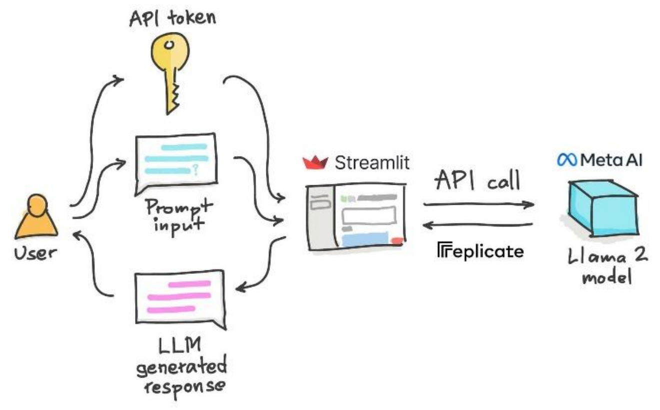
3.**challenges :**

- Challenge: LLaMA 2 (especially the 13B & 65B models) requires significant GPU resources, making deployment costly.
- Impact: Running the model locally or in production environments can be expensive and slow.
- Mitigation: Use LLaMA 2 (7B) for efficiency or leverage cloud-based APIs (e.g., Hugging Face, Replicate).

Phase-3: Project Design

Objective:

Blog generation using LlaMA2 and streamlit.



Key points:

1. **System Architecture:**
    - User enters benefits of yoga related query using UI.
    - Query is processed using Google Gemini flash 2.0.
    - AI model fetches and process the data.
    - The frontend displays the benefits of yoga in an blog mode.
2. **User Flow:**
    - Step 1: User enters a query (e.g., "benefits of yoga").
    - Step 2: The backend **calls the Gemini Flash 2.0 API** show the benefits of yoga
    - 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.
    - **Filters for Blog titles, outlines, drafts, metadiscriptions**.

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	K. Ranjith	Google API Key, Python, Streamlit setup	API connection established & working
Sprint 1	Frontend UI Development	🟡 Medium	2 hours (Day 1)	End of Day 1	A. Vijay	API response format finalized	Basic UI with input fields
Sprint 2		🔴 High	3 hours (Day 2)	Mid-Day 2	A. Vijay & S.Ramreddy	API response, UI elements ready	Search functionality with filters
Sprint 2	Error Handling & Debugging	🔴 High	1.5 hours (Day 2)	Mid-Day 2	K. Ranjith &J. Rakesh	API logs, UI inputs	Improved API stability
Sprint 3	Testing & UI Enhancements	🟡 Medium	1.5 hours (Day 2)	Mid-Day 2	S. Ramreddy &J. Rakesh	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 AutoSage App.

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

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	Query "Benefits of yoga"	Benefits of yoga displayed	✅ Passed	Tester 1
TC-002	Functional Testing	Query "list the rules in cricket"	Rules of cricket should be provided.	✅ Passed	Tester 2
TC-003	Performance Testing	API response time under 500ms	API should return results quickly.	⚠ Needs Optimization	Tester 3
TC-004	Bug Fixes & Improvements	Fixed incorrect API responses.	Data accuracy should be improved.	✅ 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**