**Hackathon Project Phases Template** for the **Blog Creator using llama 2 model and Streamlit**

# Hackathon Project Template

**Project Title:**

**Blog Creator Using Meta’s Llama 2 model and Streamlit**

**Team Name:**

Ai Blog Creator Squad 2025

**Team Members:**

* Abhishek Nemana
* Sudheer Kumar Navle
* Naveen Puppala
* Pranay Yamjala
* S Pavan Chary

## Phase-1: Brainstorming & Ideation

**Objective:**

To develop an AI-powered blog generation tool using **LLaMA 2** for natural language processing and **Streamlit** for an interactive user interface. The tool will allow users to input topics, keywords, and writing styles, generating high-quality blog posts efficiently.

**Key Points:**

1. **Problem Statement:**

**Writing blogs can be time-consuming and challenging, especially for those who struggle with ideas, structure, or consistency. Many existing tools are either complex or lack flexibility.**

**This project aims to create a simple and user-friendly blog generation tool using LLaMA 2 and Streamlit. Users can enter a topic, keywords, and preferred writing style to generate high-quality blog content quickly and efficiently. This will help bloggers, marketers, and businesses save time and improve content creation.**

1. **Proposed Solution:**

**User Input Panel** – Users enter a topic, keywords, and preferred tone/style.

○ **AI-Powered Blog Generation** – LLaMA 2 processes the input and generates a structured, high-quality blog post..

1. **Target Users:**

* 1. **Bloggers & Content Creators** – Individuals who regularly write blogs and need AI assistance for idea generation and drafting..

○ **Digital Marketers – Professionals looking for quick and engaging blog content for SEO and online marketing campaigns**.

○ **Students & Researchers – Individuals who need assistance in writing summaries, reports, or informative articles.**

1. **Expected Outcome:**

* 1. A functional **Blog Creator App** that provides information based on real-time data and user queries.

## Phase-2: Requirement Analysis

**Objective:**

Define the technical and functional requirements for the Blog Generator app

**Key Points:**

1. **Technical Requirements:**

* 1. Programming Language: **Python**

○ Backend: **Llama 2 Meta ai , langchain**

○ Frontend: **Streamlit Web Framework**

○ Database: SQLite (Lightweight & Local Storage)

1. **Functional Requirements:**

* 1. Ability to **fetch Blogs** using Llama 2 model and generate on website using Streamlit..

○ Display **relevant text** in an intuitive UI.

○ Provide **real-time imformation about the topic** based on the input user gives in specific number of words.

**Constraints & Challenges:**

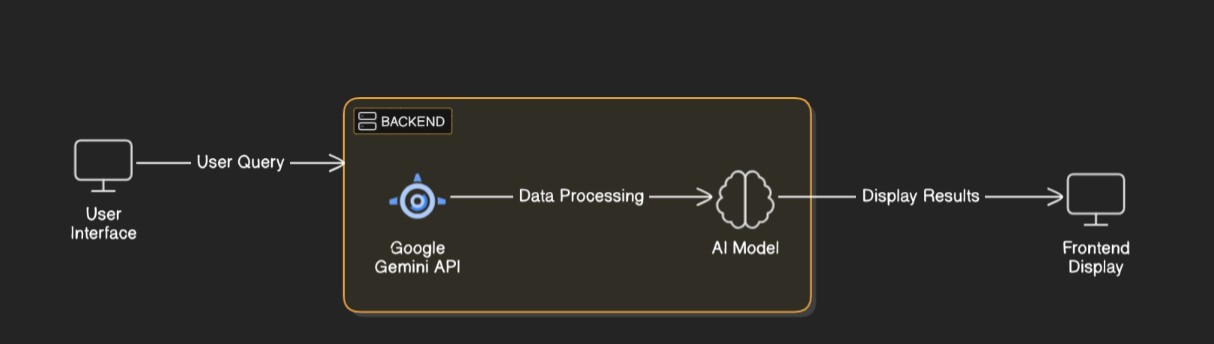
* 1. Ensuring real-time updates from **Llama 2** .

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

## Phase-3: Project Design

**Objective:**

Develop the architecture and user flow of the application.



**Key Points:**

1. **System Architecture:**

* 1. User enters a topic query via UI.

○ Query is processed using **Llama 2 tool and processed in langchain.**

○ AI model fetches and processes the data.

○ The frontend displays **the blog text on the web**.

1. **User Flow:**

* 1. Step 1: User enters a query (e.g., "Global Warming").

○ Step 2: The backend **calls the Llama 2 LLM** to retrieve vehicle data.

○ Step 3: The app processes the data and **displays results** in an easy-to-read format.

1. **UI/UX Considerations:**

* 1. **Minimalist, user-friendly interface** for seamless navigation.

○  **Displays the text for a topic seamlessly especially useful for blog creators**.

○ **Dark & light mode** for better user experience.

## 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 | Member 1 | Google API Key,  Python, Streamlit setup | API connection established & working |
| Sprint 1 | Frontend UI Development | 🟡  Medium | 2 hours  (Day 1) | End of Day  1 | Member 2 | API response format finalized | Basic UI with input fields |
| Sprint 2 | Vehicle Search &  Comparison | 🔴 High | 3 hours  (Day 2) | Mid-Day 2 | Member 1& 2 | API response, UI elements ready | Search functionality with filters |
| Sprint 2 | Error Handling &  Debugging | 🔴 High | 1.5 hours  (Day 2) | Mid-Day 2 | Member 1&4 | API logs, UI inputs | Improved API stability |
| Sprint 3 | Testing & UI  Enhancements | 🟡  Medium | 1.5 hours  (Day 2) | Mid-Day 2 | Member 2& 3 | 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 Blog Creator Web.

**Key Points:**

1. **Technology Stack Used:**

* 1. **Frontend:** Streamlit

○ **Backend:** Llama 2 model , Langchain.

○ **Programming Language:** Python

1. **Development Process:**

* 1. Implement **API key authentication** and Llama 2 LLM Model **integration**.

○ Develop **a blog for the user.**

. ○ Optimize **search queries for information 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 AutoSage App works as expected.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Case ID** | **Category** | **Test Scenario** | **Expected Outcome** | **Status** | **Tester** |
| TC-001 | Functional  Testing | Query "Best budget cars under ₹10 lakh" | Relevant budget cars should be displayed. | ✅ Passed | Tester 1 |
| TC-002 | Functional  Testing | Query "Motorcycle maintenance tips for  winter" | Seasonal tips 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 | Develop er |
| 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**