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

# **Hackathon Project Phases Template**

## **Project Title:**

**Blog Generator Using LLaMA 2 And Streamlit.**

## **Team Name:**

Medicoders

## **Team Members:**

* G. Bhavani
* M. Keerthi
* Ch. Lasya
* B. Deepthi

## **Phase-1: Brainstorming & Ideation**

### **Objective:**

The application generates a detailed and technical blog post suitable for academic and professional readers.

### **Key Points:**

1. **Problem Statement:**
   * This project leverages the capabilities of the LLaMA 2 language model to generate blog content based on user input.
   * The application is built using Streamlit, which provides an interactive web interface where users can specify the topic, word count, and target audience for the blog.
2. **Proposed Solution:**
   * An AI-powered application using LLaMA2 to provide contextually relevant blog posts tailored to the specified parameters.
   * The model generates coherent parameters which quickly generate high-quality written content.
3. **Target Users:**
   * **Content Creator** use this generator to streamline their audience engagement, save time, and improve content quality.
   * **Professional** looking for freeing them to focus on more strategic aspects like research, analysis and editing.
   * **Researches** utilize comments sections on blogs to gather feedback from readers and potential collaborators**.**
4. **Expected Outcome:**
   * A functional **AI-powered blog post application that provides insights based on user queries.**

## **Phase-2: Requirement Analysis**

### **Objective:**

Define the technical and functional requirements for the blog generator.

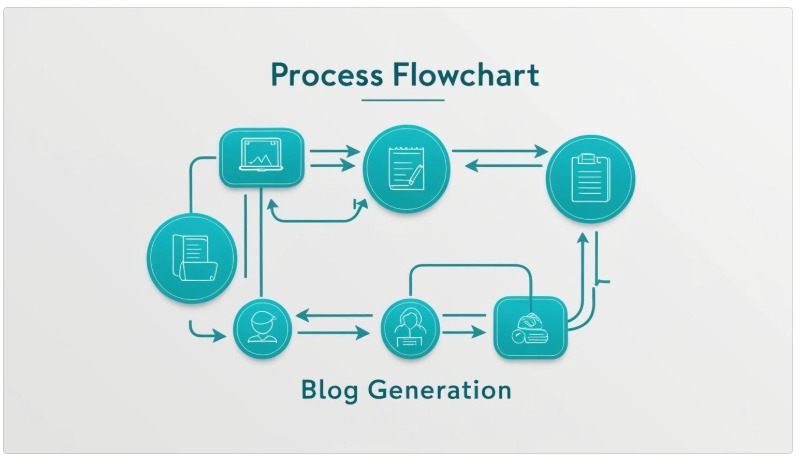
### **Key Points:**

1. **Technical Requirements:**
   * Programming Language: **Python**
   * Backend: **LLaMA2**
   * Frontend: **Streamlit Web Framework**
   * Database: **Not required initially (API-based queries)**
2. **Functional Requirements:**
   * Users should be able to choose the desired writing style (e.g., informative, persuasive, casual, formal) to match their audience and brand voice.
   * The ability to define the number of paragraphs, subheadings, and desired length (word count) of the generated blog post.
   * Features to allow users to manually edit and refine the generated content after it is produced.
3. **Constraints & Challenges:**
   * Ensuring real-time updates from Streamlit.
   * Ensuring generated content is factually correct and up-to-date, especially when dealing with rapidly evolving topics.

## **Phase-3: Project Design**

### **Objective:**

Develop the architecture and user flow of the application.



### 

### **Key Points:**

1. **System Architecture:**
   * Query is processed using **Streamlit.**
   * AI model fetches and processes the data.
   * The frontend (displays), number Enter the prompt, number of words, blog target audience and generate the blog.
2. **User Flow:**
   * Step 1: User enters a query (e.g., "write about cricket”).
   * Step 2: The backend **calls the Streamlit to retrieve the data.**
   * Step 3: The app processes the data and **displays results** in an easy-to-read format.
3. **UI/UX Considerations:**
   * "UX" is more relevant than "UI".
   * AI blog post generators have the power to transform the content creation process for businesses of all sizes.

## 

## **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 | Create the virtual environment for the code | 🔴 High | 4 hours (Day 1) | End of Day 1 | B. Deepthi | Langchain community, Python, Streamlit setup | Streamlit established & working |
| Sprint 1 | Creating requirements.txt | 🟡 Medium | 1 hours (Day 1) | End of Day 1 | Ch. Lasya | Pip installing requirememts in Anakonda promt | Succesfully installed into app.py |
| Sprint 2 | Coding (app.py) the application | 🔴 High | 3 hours (Day 2) | Mid-Day 2 | B. Deepthi | LLaMA2, UX elements ready | Search functionality  With generated content |
| Sprint 2 | Error Handling & Debugging | 🔴 High | 1.5 hours (Day 2) | Mid-Day 2 | G. Bhavani and Ch. Lasya and M. Keerthi | UX inputs | Improved query stability |
| Sprint 3 | Testing & UX Enhancements | 🟡 Medium | 1.5 hours (Day 2) | Mid-Day 2 | G. Bhavani and M. Keerthi | LLaMA2 response, UX layout completed | Responsive UX, 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 **Streamlit, langchain community.**  
 **(🟡 Medium Priority)** Build a **basic requirements and command it in Anaconda prompt(install).**

### **Sprint 2 – Core Features & Debugging (Day 2)**

**(🔴 High Priority)** Implement **LLaMA 2 and UX elements.**  
 **(🔴 High Priority)** Debug UX issue & handle **errors in queries**.

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

**(🟡 Medium Priority)** Test LLaMA 2 response and UX bugs  
 **(🟢 Low Priority)** Final **demo preparation & deployment**.

## **Phase-5: Project Development**

### **Objective:**

Implement core features of the Blog Generator.

### **Key Points:**

1. **Technology Stack Used:**
   * **Frontend:** Streamlit
   * **Backend:** LLaMA2
   * **Programming Language:** Python
2. **Development Process:**
   * Implement LLaMA 2 and Streamlit.
   * Developing the application and **maintenance tips logic**.
   * Optimize **search queries for performance and relevance**.
3. **Challenges & Fixes:**
   * **Challenge:** To get Backend code  
      **Fix:** Implement **codepen for optimizing the code.**

## **Phase-6: Functional & Performance Testing**

### **Objective:**

Ensure that the blog generator works as expected.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Case ID** | **Category** | **Test Scenario** | **Expected Outcome** | **Status** | **Tester** |
| TC-001 | Functional Testing | Query "How do become a good GenAI Developer" | Relevant blog post should be displayed. | ✅ Passed | Tester 1 |
| TC-002 | Functional Testing | Query "How is deepseek taking over chatgpt" | Challenges should be provided. | ✅ Passed | Tester 2 |
| TC-003 | Performance Testing | API response time  is slow | 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 UX is responsive across devices. | UX 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**