Hackathon Project Phases Template that ensures students can complete it efficiently while covering all six phases. The template is structured to capture essential information without being time-consuming.

Hackathon Project Phases Template

Project Title:

BlogGen AI: LLaMA 2 & Streamlit Powered Blog Generation

Team Name:

Torchwood

Team Members:

- Seetha Neeraj Kumar
- Tummalapalli Sai Aditya
- Barenkala Laxman
- Bhakti Kushan

Phase-1: Brainstorming & Ideation

Objective:

- Identify the problem statement.
- Define the purpose and impact of the project.

Key Points:

1. **Problem Statement:** Blog Generation Using LLaMA 2 and Streamlit.
2. **Proposed Solution:** We developed a blog generation AI website using Streamlit and LLaMA 2. The platform allows users to input a blog topic, desired tone, and target audience, then leverages the LLaMA 2 model to generate a complete blog post.
3. **Target Users:** Bloggers, content creators, marketing professionals, small business owners, and digital content strategists who require quick and customizable blog content.
4. **Expected Outcome:** A clean, intuitive UI that offers multiple customization options. Users will be able to generate content that fits their preferred style and tone with minimal effort.

Phase-2: Requirement Analysis

Objective:

- Detail the technical and functional requirements and outline potential constraints and challenges.

Key Points:

1. **Technical Requirements:** Python, Langchain, Huggingface,Streamlit, Llama2 API
2. **Functional Requirements:** Blog Topic, Tone, Target Audience Customization, Word Count. Estimated Read Time
3. **Constraints & Challenges:** Prompt Template Refinement, Balancing featureset with User friendly design.

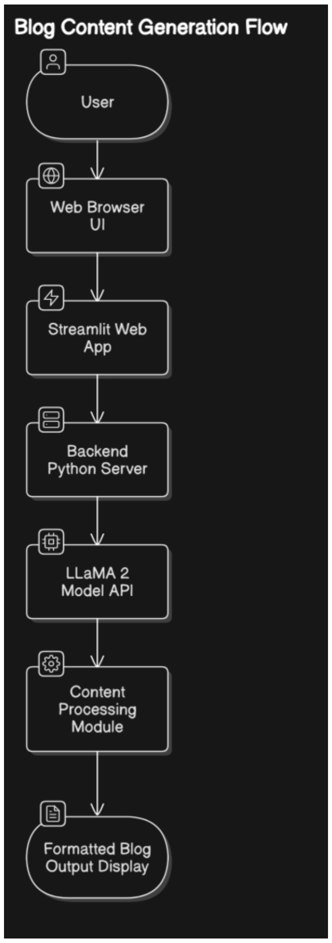
Phase-3: Project Design

Objective:

- Create the architecture and user flow.

Key Points:

1. **System Architecture Diagram:**


Blog Content Generation Flow
User
Web Browser UI
Streamlit Web App
Backend Python Server
LLaMA 2 Model API
Content Processing Module
Formatted Blog Output Display

2. **User Flow:**

   - **Entry Point:** The user accesses the web app via a browser.
   - **Input:** The user fills in the blog topic, selects tone and target audience, and specifies additional options such as word count and style.
   - **Processing:** The app forwards the inputs to the backend, where LLaMA 2 generates draft content.
   - **Post-Processing:** The system refines the output, calculates the estimated read time, and applies formatting options.
   - **Output:** The final blog post is displayed, with options to modify, save, or download the content.

3. **UI/UX Considerations:**
   - A minimalistic and clean interface to reduce clutter and guide users intuitively through the process.
   - Accessibility considerations, including readable fonts, high-contrast color schemes, and clear navigation

---

Phase-4: Project Planning (Agile Methodologies)

Objective:

- Break down the project into sprints with clear task allocation and milestones.

Key Points:

1. **Sprint Planning:**

   - **Sprint 1:** Set up the development environment and basic Streamlit UI layout.
   - **Sprint 2:** Integrate the backend with LLaMA 2 and implement the core blog generation feature.
   - **Sprint 3:** Add advanced options (tone, audience, word count, style) and implement post-processing (read time, formatting).
   - **Sprint 4:** Conduct comprehensive testing, address bugs, and refine the prompt templates.

2. **Task Allocation:**
   - T. Sai Aditya: Project coordination and overall UI/UX design.
   - Seetha Neeraj Kumar: Backend integration and API interfacing with LLaMA 2.
   - Barenkala Laxman: Feature development for input options and functionality testing.
   - Bhakti Kushan: Agile planning, documentation, and final validations.

3. **Timeline & Milestones:**

✓ Complete project setup and initial UI design.
✓ Backend - *Huggingface Endpoint* integration from LangChain and
✓ Adding **Web Search** tool using DuckDuckGo search
✓ Improving User Interface by adding tone, target audience to the UI.
✓ Integration Testing and Debugging.
✓ Refining Prompt Template.
✓ Upgrading Web Search using **Tavily Search API**
✓ Multiple prompt tests and debugging.

Phase-5: Project Development

Objective:

- Code the project and integrate components.

Key Points:

1. **Technology Stack Used:** Python,CSS, Langchain [HuggingFaceEndpoint], Streamlit, TavilySearch API.
2. **Development Process:**
   - **Project Setup:**
     Kick off by setting up the repository, environment, and initial UI layout using Streamlit.
   - **Backend & API Integration:**
     Connect with the Huggingface Endpoint through LangChain, ensuring the model responds accurately to input prompts.
   - **Web Search Tool Implementation:**
     Integrate the DuckDuckGo search API for immediate content retrieval, then upgrade to Tavily Search API to enhance search performance and accuracy.
   - **UI Enhancement & Feature Addition:**
     Incorporate tone and target audience options, enabling users to further customize their blog content.
   - **Testing & Iteration:**
     Conduct integration testing at each milestone, iteratively refine the prompt templates, and perform debugging to address any issues encountered.

3. **Challenges & Fixes:**
   - Ensuring robust and accurate integration between multiple APIs (Huggingface via LangChain, DuckDuckGo, and Tavily Search).
   - Refining prompt templates to address inconsistencies from the LLaMA 2 model outputs.
   - Maintaining a balance between feature richness and a user-friendly interface.

Phase-6: Functional & Performance Testing

Objective:

- Ensure the project works as expected.

Key Points:

Test Cases Executed:

- **UI Input Validation:**
  • Verified that all required fields (blog topic, tone, target audience, word count) accept valid data and display clear error messages for invalid or missing inputs.
- **End-to-End Functionality:**

• Tested the complete flow—from form submission to blog generation via the Huggingface Endpoint through LangChain, including the integration of the Web Search tool (initially DuckDuckGo, then Tavily Search API).

- **Prompt Template Testing:**
  • Ran multiple scenarios with varied topics, tones, and audiences to ensure consistent and coherent blog outputs.
- **Edge Case Handling:**
  • Simulated scenarios with empty inputs, excessively long texts, and special characters to verify robust error handling and graceful degradation.
- **Formatting and Read Time Estimation:**
  • Ensured that the final blog output includes proper formatting, accurate read time calculations, and correct integration of additional features like author names.
- **Performance & Load Testing:**
  • Simulated concurrent user interactions to assess system responsiveness and stability under load.
- **Cross-Device & Browser Compatibility:**
  • Validated that the UI renders correctly across desktops, tablets, and mobile devices as well as on major web browsers.

1. **Bug Fixes & Improvements:**
   - **Integration Bugs:**
     • Resolved issues between the Streamlit UI and the Huggingface Endpoint via LangChain, ensuring seamless data flow.
   - **Timeout & Input Handling:**
     • Fixed scenarios where certain inputs caused timeout errors during blog generation, improving the handling of edge cases.
   - **Prompt Template Optimization:**
     • Refined and standardized prompt templates to enhance the coherence and quality of generated blog content.
   - **Formatting Corrections:**
     • Addressed bugs related to incorrect read time estimation and blog layout formatting, ensuring a polished final output.
   - **Web Search API Enhancements:**
     • Improved the API call logic for both the DuckDuckGo and Tavily Search integrations, leading to more accurate and relevant search results.
   - **User Experience Enhancements:**
     • Made iterative UI adjustments based on user feedback, such as clearer navigation cues and responsive design fixes for various devices.

2. **Final Validation:**
   - **Requirement Fulfillment:**
     • The project successfully meets all initial requirements by providing a complete solution for generating blog content based on user-defined topics, tones, and target audiences.
   - **Feature Integration:**
     • End-to-end tests confirm that all modules—from backend API integration to UI enhancements—work together seamlessly, delivering a coherent and functional user experience.
   - **Performance & Usability:**
     • Comprehensive testing demonstrates that the system is both robust under load and user-friendly, with all functionalities (content generation, web search integration, and formatting) operating as intended.