Hackathon Project Phases Template

Project Title:

Blog Generation Using LLaMA 2 and Streamlit

Team Name:

Hackstorm

Team Members:

B. Shanmukh

T. Navadeep

N. Madhu Priya

Y. Harika

P. Shivani

Phase-1: Brainstorming & Ideation

Objective:

Develop an AI-powered blog generation tool using LLaMA 2 and Streamlit to help users create high-quality, structured blog content efficiently.

Key Points:

1.Problem Statement:

- Content creators often struggle with writer's block and structuring their blog posts effectively.

- Manually researching and generating ideas can be time-consuming.

2.Proposed Solution:

- A web-based application using LLaMA 2 to generate blog content based on user prompts.
- Streamlit-powered frontend for easy accessibility and user interaction.

3.Target Users:

- Bloggers and content writers looking for AI-assisted content creation.
- Businesses and marketers needing SEO-friendly blog posts.
- Casual users interested in generating blogs quickly.

Expected Outcome:

- A functional AI-powered blog generation app with structured content output based on user input.

Phase-2: Requirement Analysis

Objective:

Define the technical and functional requirements for the AI Blog Generator

Key Points:

Technical Requirements:

- Programming Language: **Python**
- Backend: **LLaMA 2 API (Hugging Face)**
- Frontend: **Streamlit Web Framework**
- Database: **Not required initially (API-based queries)**

Functional Requirements:

- Generate blog content based on a given topic or keywords.
- Allow users to specify the tone and style of the content.
- Provide an option to generate outlines before full content creation.

Constraints & Challenges:

- Handling API rate limits efficiently.
- Ensuring generated content is coherent and engaging.
- Optimizing the model's response time.

Phase-3: Project Design

Objective:

Develop the architecture and user flow of the application.

Sprint	Task	Priority	Duration	Deadline	Assigned To	Dependencies	Expected Outcome
Sprint 1	Environment Setup & API Integration	🔴 High	6 hours	End of Day 1	[Member]	API Key, Python, Streamlit setup	API connection established
Sprint 1	Frontend UI Development	🟡 Medium	2 hours	End of Day 1	[Member]	API response format finalized	Basic UI with input fields
Sprint 2	Blog Generation Functionality	🔴 High	4 hours	Mid-Day 2	[Member]	API response, UI elements ready	Functional blog generation
Sprint 2	Error Handling & Debugging	🔴 High	2 hours	Mid-Day 2	[Member]	API logs, UI inputs	Improved API stability
Sprint 3	UI Enhancements & Features	🟡 Medium	3 hours	End of Day 2	[Member]	API response, UI layout completed	Responsive UI, export feature
Sprint 3	Final Testing & Deployment	🟢 Low	2 hours	End of Day 2 ↓	Entire Team	Working prototype	Demo-ready project

Key Points:

System Architecture:

- User inputs a blog topic, keywords, or desired structure.
- The system processes the input and sends a request to the LLaMA 2 API.
- The AI model generates blog content.
- The frontend displays the generated blog with options for refinement.

User Flow:

- Step 1: User enters a topic (e.g., "The Future of AI in Healthcare").
- Step 2: The backend calls the LLaMA 2 API to generate structured content.
- Step 3: The app processes and formats the data.
- Step 4: The user can edit, refine, or regenerate sections as needed.

UI/UX Considerations:

- Clean, minimalistic design for easy navigation.
- Options for different tones/styles (e.g., formal, conversational, technical).
- Export functionality for easy content saving.

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	End of Day 1	[Member]	API Key, Python, Streamlit setup	API connection established
Sprint 1	Frontend UI Development	🟡 Medium	2 hours	End of Day 1	[Member]	API response format finalized	Basic UI with input fields
Sprint 2	Blog Generation Functionality	🔴 High	4 hours	Mid-Day 2	[Member]	API response, UI elements ready	Functional blog generation
Sprint 2	Error Handling & Debugging	🔴 High	2 hours	Mid-Day 2	[Member]	API logs, UI inputs	Improved API stability
Sprint 3	Test API responses, refine UI, & fix UI bugs	🟡 Medium	End of Day 2	[Member]	Working prototype	Refined UI, fixed bugs, API response tests completed	
Sprint 3	Final demo preparation & deployment	🟢 Low	End of Day 2	Entire Team	Functional app	Final demo-ready project, deployment	

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 AI Blog Generator.

Key Points:

Technology Stack Used:

- Frontend: Streamlit
- Backend: LLaMA 2 API (Hugging Face)
- Programming Language: Python

Development Process:

1. Implement API key authentication and LLaMA 2 API integration.

2. Develop content generation logic.

3. Optimize user input handling for better blog structure.

Challenges & Fixes:

- Challenge: Ensuring logically structured content.
- Fix: Implement outline-based content generation.
- Challenge: Handling API response delays.
- Fix: Implement caching for frequently used topics.

Phase-6: Functional & Performance Testing

Test Case ID	Category	Test Scenario	Expected Outcome	Status	Tester
TC-001	Functional Testing	Generate a blog on "AI in Education"	Well-structured content	✅ Passed	[Tester]
TC-002	Functional Testing	Generate an outline before full blog	Outline should be relevant	✅ Passed	[Tester]
TC-003	Performance Testing	API response time under 1 second	Fast content generation	⚠️ Needs Optimization	[Tester]
TC-004	Bug Fixes & Improvements	Fixed incorrect sentence structures	Improved readability	✅ Fixed	[Developer]
TC-005	Final Validation	Ensure UI is responsive	UI should work on all devices	❌ Failed - Needs Fixing	[Tester]
TC-006	Deployment Testing	Host the app on Streamlit Sharing	App should be accessible	🚀 Deployed	DevOps