Hackathon Project Phases Template for the **Skikshak Mahoday** project.

Hackathon Project Phases Template

Project Title:

Shikshak Mahoday: Palm-Powered Data Science Tutor

Team Name:

BinaryBeasts

Team Members:

- B.Ashrita
- G.Ashwini
- D.Neha
- T.Saisree
- V.S.Spoorthy

Phase-1: Brainstorming & Ideation

Objective:

Develop an AI-powered data science tutor, **Shikshak Mahoday,** using Flask framework , gTTs(Google text to speech) library to provide personalized, text and audio-based explanations of data science concepts. The system enables adaptive learning, interactive Q&A, and structured progress tracking to enhance accessibility and engagement.

Key Points:

1. **Problem Statement:**

- Many learners struggle to find reliable, easy-to-understand resources for data science concepts.
- Users need personalized guidance based on their skill level and learning preferences.

2. **Proposed Solution:**

- An AI-powered tutor, **Shikshak Mahoday**, using Flask framework , gTTs(Google text to speech) library to generate customized explanations in text and audio formats.
- The system provides interactive Q&A, progress tracking, and adaptive learning paths to enhance user engagement.

3.Target Users:

- Aspiring data scientists seeking simplified, structured learning.
- Students & professionals looking to enhance their knowledge with AI-driven explanations.
- Self-learners who prefer personalized, multimodal education (text & audio).

4. **Expected Outcome:**

- A functional AI-powered data science tutor that delivers real-time, adaptive learning experiences, making data science accessible and engaging for everyone.

Phase-2: Requirement Analysis

Objective:

Define the technical and functional architecture for Shikshak Mahoday, an AI-driven data science tutor that delivers personalized, interactive learning experiences using Flask framework , gTTs(Google text to speech) library and Text-to-Speech (TTS) technology.

Key Points:

1.Technical Requirements:

- Programming Language: Python
- Backend: Flask framework , gTTs(Google text to speech) library
- Frontend: Streamlit Web Framework
- Database: Not required initially (API-based content generation)

2.Functional Requirements:

- Generate personalized data science explanations based on user queries.
- Deliver text and audio-based responses using Text-to-Speech (TTS).
- Support interactive Q&A for deeper understanding.
- Provide progress tracking and adaptive learning paths for learners.

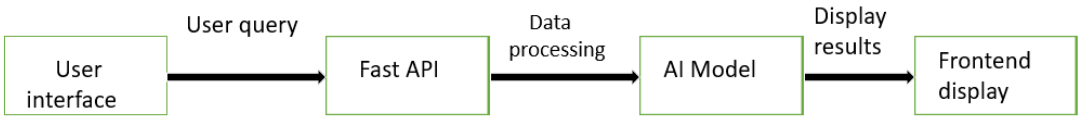
3.Constraints & Challenges:

- Ensuring high-quality, context-aware AI-generated responses.
- Managing API rate limits and optimizing response times.
- Designing an intuitive and engaging UI with Streamlit.

Phase-3: Project Design

Objective:

Develop the architecture and user flow of the application.



Key Points:

1.System Architecture:

- User enters a data science topic and expertise level via the UI.
- Query is processed using Flask framework , gTTs(Google text to speech) library.
- AI model generates a personalized explanation in text format.
- Text-to-Speech (TTS) converts the explanation into audio format.
- The frontend displays the text and audio response for the users.

2.User Flow:

- Step 1: User enters a topic (e.g., "Explain Linear Regression for beginners").
- Step 2: The backend calls Flask framework , gTTs(Google text to speech) library to generate an explanation.
- Step 3: The system processes the response, converts it into text and audio, and displays it in an easy-to-understand format.

3. UI/UX Considerations:

- Minimalist, intuitive interface for seamless navigation.
- Interactive Q&A support for deeper learning.
- Dark & light mode for better readability and 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	Ashwini	Fast API Key, Python, Streamlit setup	API connection established & working
Sprint 1	Frontend UI Development	□ Medium	2 hours (Day 1)	End of Day 1	Neha	API response format finalized	Basic UI with input fields
Sprint 2	AI Response Generation & Processing	□ High	3 hours (Day 2)	Mid-Day 2	Spoorthy,Neha	API response, UI elements ready	AI-generated text & auido explanations
Sprint 2	Error Handling & Debugging	□ High	1.5 hours (Day 2)	Mid-Day 2	Sai Sree,Ashrita	API logs, UI inputs	Improved API stability
Sprint 3	Testing & UI Enhancements	□ Medium	1.5 hours (Day 2)	Mid-Day 2	Ashrita	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 Flask framework API.
(□ Medium Priority) Build a basic UI with **input fields for topic & expertise level**.

Sprint 2 – Core Features & Debugging (Day 2)

(□ High Priority) Implement **AI response generation** (text & audio output).
(□ High Priority) Debug **API issues & optimize AI query handling**.

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

(□ Medium Priority) Test **AI-generated responses**, refine UI, & fix bugs.
(□ Low Priority) Final **demo preparation & deployment**.

Phase-5: Project Development

Objective:

Implement core features of the Shikshak Mahodhay App.

Key Points:

1. **Technology Stack Used:**
    - **Frontend:** Streamlit.
    - **Backend:** Flask framework , gTTs(Google text to speech) library.
    - **Programming Language:** Python.
2. **Development Process:**
    - Implement **API key authentication** and **Flask framework , gTTs(Google text to speech) library integration**.
    - Develop **AI response generation logic** for text and audio explanations.
    - Optimize **query handling for accuracy and efficiency**.
3. **Challenges & Fixes:**
    - **Challenge:** Delayed API response times.
      Fix: Implement **response caching** to store frequently asked explanations.
    - **Challenge:** Limited API calls per minute.
      Fix: Optimize **query structure** to reduce redundant API calls.

Phase-6: Functional & Performance Testing

Objective:

Ensure that the Shikshak Mahodhay App works as expected.

Test Case ID	Category	Test Scenario	Expected Outcome	Status	Tester
TC-001	Functional Testing	Query "Explain Linear Regression for beginners"	AI should generate a clear and simple explanation.	✅Passed	Ashrita
TC-002	Functional Testing	Query "Provide an example of Decision Trees"	AI should return a relevant example with explanation	✅Passed	Sai Sree

TC-003	Performance Testing	AI response time under **700ms**	AI should quickly generate responses.	⚠ Needs Optimization	Spoorthy
TC-004	Bug Fixes & Improvements	Fixed incorrect AI-generated explanations.	Improved accuracy of explanations.	✅Fixed	Neha
TC-005	Final Validation	Ensure UI works across devices (mobile & desktop).	UI should be fully responsive.	❌Failed - UI broken on mobile	Ashwini
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**