Hackathon Project Phases Template for the **AutoSage App** project.

Hackathon Project Phases Template

Project Title:

Trans Lingua: AI-powered multi language translator

Team Name:

Vignan's

Team Members:

- B. Adithya
- A. Akhil
- M. Ajay
- B. Praveen Kumar

Phase-1: Brainstorming & Ideation

Objective:

To develop an advanced AI-powered multi-language translator that leverages natural language processing (NLP) and machine learning algorithms to provide accurate, real-time translations across multiple languages, enabling seamless communication and fostering global communiction.

Key Points:

1. **Problem Statement:**

    - Tran Lingua is a cutting-edge web application designed to harness the power of advanced AI to provide seamless language translation services.
      By simply inputting text and selecting the desired source and target language, users can instantly receive accurate translations powered by the latest AI models.

○

2. **Proposed Solution:**

    ○ Integrate the AI model to process text or speech in real-time, ensuring lowlatency and accurate translations.
    ○ Develop an intuitive interface, such as a mobile app or web platform, that provides easy access to the translation system.
3. **Target Users:**

    ○ **Travelers & Tourists** need Seamless communication in foreign countries where the user doesn't speak the local language.
    ○ **Language learners & Students** need assistance in learning new languages by providing immediate translation and contextual understanding.
    ○ **Healthcare Professionals** need communicating effectively with patients who speak different languages, especially in emergency or multicultural settings.
4. **Expected Outcome:**

    ○ By providing real-time and accurate translations, users will be able to complete tasks more efficiently, reducing the time spent on manual translations or reliance on human translators.

Phase-2: Requirement Analysis

Objective:

Define the technical and functional requirements for the AutoSage App.

Key Points:

1. **Technical Requirements:**

    ○ Programming Language: **Python**
    ○ Backend: **Python**
    ○ Frontend: **Streamlit Web Framework**
    ○ Database: **Not required initially (API-based queries)**
2. **Functional Requirements:**

    ○ The system must support translation for a wide range of languages, including major world languages as well as regional and less commonly spoken languages. The system must provide real-time translation for text inputs, enabling users to get immediate results.
    ○ The system should provide the option to adjust tone (formal/informal) based on the context, especially for business or legal translations.

○

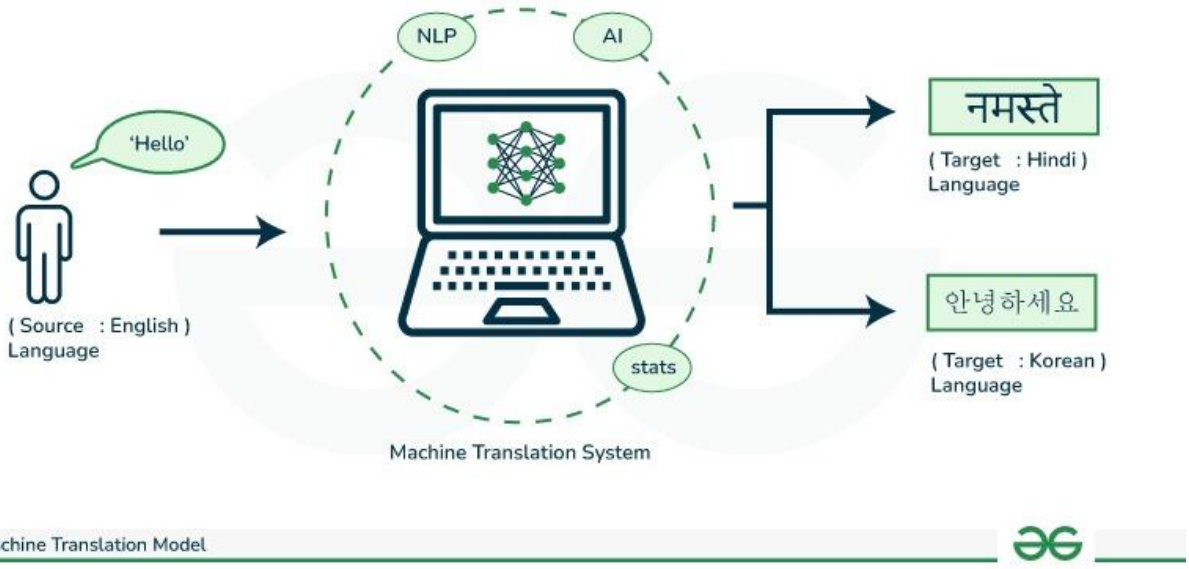
3. **Constraints & Challenges:**

    - The development of an **AI-powered Multi-Language translator** that need to be addressed to ensure the tool is effective, scalable, and accessible.
    - To succeed, developers must strike a balance between technological complexity, and user satisfaction, ensuring that the translation system is accurate, scalable, and adaptable to various use cases and languages..

Phase-3: Project Design

Objective:

Develop the architecture and user flow of the application.



Key Points:

1. **System Architecture:**

- **Web/Mobile Application**: A front-end interface where users can input text, choose languages (source and destination), and view the translated output.
- **Voice Input/Output**: Optionally, the system could accept voice input using speech recognition (e.g., Google Speech-to-Text) and output the translation via text-to-speech (TTS).
- **Text Input**: Users can type the text they want to translate.

-

UI/UX Considerations:

- Provide a **large, easy-to-use text box** for the user to input the text they want to translate.
- Ensure the text field allows for multi-line input for longer translations.
- **Auto-Detect Option**: Include an option to **auto-detect** the source language. This can be a useful feature for users who are unsure of the language they are inputting.

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	Finding libraries installation	High	6 hours (Day 1)	End of Day 1	Akhil	Google API Key, Python, Streamlit setup	API connection established & working
Sprint 1	Finding libraries installation	Medium	2 hours (Day 1)	End of Day 1	Adithya	Streamlit,VS code	Basic UI with input fields
Sprint 2	Error Handling & Debugging	High	1.5 hours (Day 2)	Mid-Day 2	Adithya&Praveen	API logs, UI inputs	Improved API stability
Sprint 3	Final Presentation & Deployment	Medium	1.5 hours (Day 2)	Mid-Day 2	Ajay	API response, UI layout completed	Responsive UI, better user experience

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 AutoSage App.

Key Points:

1. **Technology Stack Used:**
    - **Frontend:** Streamlit
    - **Backend:** Google Gemini Flash API
    - **Programming Language:** Python
2. **Development Process:**
    - Developed by using stream lit integrated with vs code.
3. **Challenges & Fixes:**
    - **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	Adithya
TC-002	Functional Testing	Query "Motorcycle maintenance tips for winter"	Seasonal tips should be provided.	☑ Passed	Akhil
TC-003	Performance Testing	API response time under 500ms	API should return results quickly.	Needs Optimization	Team
TC-004	Bug Fixes & Improvements	Fixed incorrect API responses.	Data accuracy should be improved.	☑ Fixed	Entire Team
TC-005	Final Validation	Ensure UI is responsive across devices.	UI should work on mobile & desktop.	☑ Failed - UI broken on mobile	Entire team
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: https://github.com/adithya996/Vignans**
4. **Presentation**