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

Hackathon Project Phases Template

Project Title:

TransLingua: AI-Powered Multi-Language Translator

Team Name:

Autogen coders

Team Members:

.K.Akash ram

M.Abhinav

P.Gopikrishna

G.Deekshith

M.Harika

Phase-1: Brainstorming & Ideation

Objective:

TransLingua is an AI-powered web application that provides seamless and accurate language translation with a user-friendly interface. It leverages advanced AI models for fast, reliable translations, helping businesses expand their market reach through multilingual communication. Additionally, it enhances translation efficiency for personal, educational, and professional use.

Key Points:

1. **Problem Statement:**

- **TransLingua is an AI-powered web application** that provides seamless and accurate language translation using advanced AI models. It ensures fast and reliable translations through a user-friendly interface for multiple languages.

- **TransLingua helps businesses expand their market reach** by enabling accurate translation of documents and customer interactions. This allows companies to maintain consistency across languages and engage a global audience.

2. **Proposed Solution:**
   TransLingua enables businesses to translate documents and communications into multiple languages with accuracy and consistency. This helps companies effectively reach non-English-speaking regions and expand their market presence.
   It ensures seamless translation of promotional and technical content to maintain clarity across different languages. This allows businesses to engage local audiences and enhance customer communication globally.

3. **Target Users:**

   - **Vehicle buyers** looking for specifications and comparisons.
   - **Vehicle owners** needing seasonal maintenance tips.
   - **Eco-conscious consumers** searching for hybrid and electric vehicle options.

4. **Expected Outcome:**

Accurate and fast AI-powered translations for multiple languages.

Seamless business expansion through effective multilingual communication.

Enhanced customer engagement with clear and localized messaging.

Phase-2: Requirement Analysis

Objective:

Define the technical and functional requirements for the TransLingua web application.

Key Points:

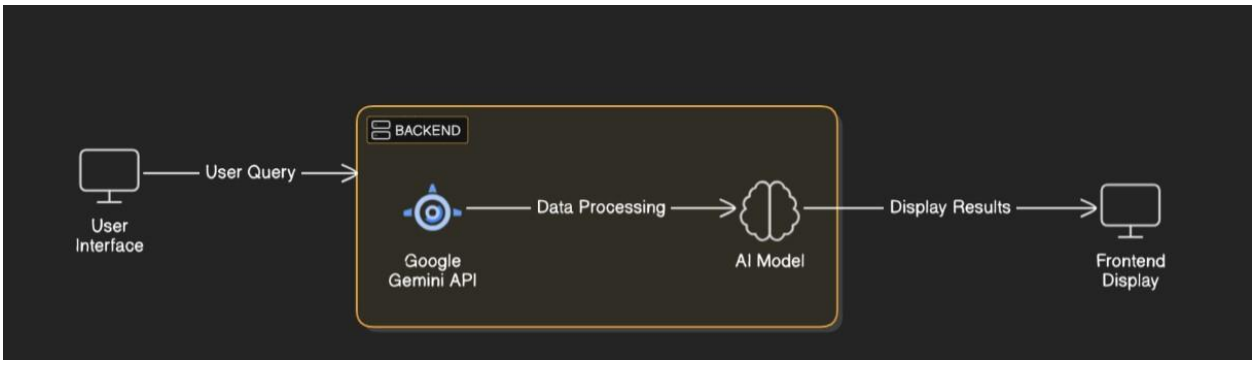
1. **Technical Requirements:**
   - **Programming Language:** Python
   - **Backend:** Google Generative AI API
   - **Frontend:** Streamlit Web Framework
   - **Database:** Not required initially (API-based queries)
2. **Functional Requirements:**
   - Ability to fetch and process translations via Google Generative AI API.
   - Display translations in an intuitive and user-friendly interface.
   - Support multiple languages with high translation accuracy.
   - Allow users to input text and select source and target languages.
3. **Constraints & Challenges:**
   Ensuring real-time and accurate translation outputs.
   - Handling API rate limits and optimizing API usage.
   - Providing a seamless UI experience with Streamlit.

.

Phase-3: Project Design

Objective:

Develop the system architecture and user flow for the TransLingua application.



Key Points:

System Architecture:

○ User inputs text and selects source and target languages.
○ The backend processes the request using Google's Generative AI API.
○ AI model translates and returns the output.
○ The frontend displays the translated text in an intuitive UI.

User Flow:

Step 1: User enters text and selects languages.
Step 2: API request is sent to Google Generative AI.
Step 3: The system processes and returns the translation.
Step 4: The translated text is displayed in the UI.

UI/UX Considerations:

○ Clean and minimalist design for easy navigation.
○ Dark & light mode for better user experience.
○ Simple language selection process with dropdown menus.

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	Member 1	Google API Key, Python, Streamlit setup	API connection established & working
Sprint 1	Frontend UI Development	Medium	2 hours (Day 1)	End of Day 1	Member 2	API response format finalized	Basic UI with input fields
Sprint 2	Translation Processing	High	3 hours (Day 2)	Mid-Day 2	Member 1& 2	API response, UI elements ready	Translation functionality implemented

Sprint 2	Error Handling & Debugging	High	1.5 hours (Day 2)	Mid-Day 2	Member 1&4	API logs, UI inputs	Improved API stability
Sprint 3	Testing & UI Enhancements	Medium	1.5 hours (Day 2)	Mid-Day 2	Member 2& 3	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 **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 TransLingua application.

Key Points:

Technology Stack Used:

○ **Frontend**: Streamlit
○ **Backend**: Google Generative AI API
○ **Programming Language**: Python

Development Process:

○ Implement API key authentication and AI model integration.
○ Develop language selection and translation logic.
○ Optimize performance for quick responses.

Challenges & Fixes:

○ **Challenge**: Slow API response times.
Fix: Implement caching for frequently used translations.
○ **Challenge**: API call limitations.
Fix: Optimize API usage and batch requests when necessary.

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	Translate English text to French "	Relevant budget cars should be displayed.	Passed	Tester 1
TC-002	Functional Testing	Input special characters	Seasonal tips should be provided.	Passed	Tester 2
TC-003	Performance Testing	API response time under 500ms	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 UI is responsive across devices.	UI 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

- **Project Report – Based on the templates.**
- **Demo Video – 3-5 minutes showcasing app functionality.**
- **GitHub/Code Repository Link – With final implementation.**
- **Presentation – Covering project details and outcomes.**