**Project Title:**

**AI POWERED MULTI LANGUAGE GENERATOR**

**Team Name:**

INNOVATECH

**Team Members:**

* Aravind Reddy
* Hameed Abbas
* Harshith Reddy
* Charan Teja

### ****Phase 1: Brainstorming & Ideation****

#### **Objective:**

To develop an AI-powered multi-language translator that accepts input in the form of text, files, or images and translates it into any language selected by the user. The main aim is to provide flexible and accurate language translations that can handle multiple input formats.

#### **Key Points:**

**Problem Statement:**

* Many users struggle to find a reliable translation tool that supports multiple languages and various types of input, such as text, files, or images.
* Users often need real-time, context-aware translations that go beyond simple word-for-word translations.

**Proposed Solution:**

* An AI-powered translation application that can handle text, files, and images as input and output the translation in a user-selected language.
* The tool will use advanced language models to provide context-sensitive translations, ensuring a more accurate result.

**Target Users:**

* Individuals needing personal translations for text, files, or images.
* Businesses looking for multi-language support for documents, communications, or marketing materials.
* Language learners or travelers needing quick translations in different languages.

**Expected Outcome:**

* A fully functional AI-based multi-language translator capable of handling various input types (text, files, images) and providing accurate translations in multiple languages.

### ****Phase 2: Requirement Analysis****

#### **Objective:**

Define the technical and functional requirements for the multi-language translation tool.

#### **Key Points:**

**Technical Requirements:**

* **Programming Languages:** Python
* **Backend:** AI model (e.g., Google Cloud Translation API, OpenAI Language Models)
* **Frontend:** Streamlit (for user interface)
* **Input Formats Supported:** Text, file uploads (Word, PDF, etc.), image uploads (with Optical Character Recognition, OCR)
* **Database:** Not required initially (API-based translation)

**Functional Requirements:**

* Accepts user input in the form of text, files, or images.
* Translates input into any desired language.
* Supports multiple languages for both input and output.
* Provides context-aware translations that can be adjusted according to the tone or style of language.
* OCR support for image input (extracts text from images for translation).

**Constraints & Challenges:**

* Handling large file sizes for input documents.
* Optimizing translation speed and accuracy, especially for images.
* API limitations on translation volume and rate.

### ****Phase 3: Project Design****

#### **Objective:**

Develop the architecture and user flow for the application.

#### **Key Points:**

**System Architecture:**

1. **User Input:**
   * User uploads text, file, or image.
2. **Pre-processing:**
   * For text: Directly sent to the AI model for translation.
   * For files: Extract text from files using libraries (e.g., PyPDF2 for PDFs, python-docx for Word files).
   * For images: Use OCR to extract text (e.g., Tesseract OCR).
3. **AI Translation:**
   * Send text to translation API or AI model.
   * Output translated text in the selected language.
4. **User Output:**
   * Display translated text or provide downloadable translated file.

**User Flow:**

1. **Step 1:** User selects input type (text, file, image).
2. **Step 2:** User uploads content (text, file, or image).
3. **Step 3:** The backend processes the input (text, OCR for images, file content extraction).
4. **Step 4:** The AI model performs the translation.
5. **Step 5:** Translated content is displayed or offered for download.

**UI/UX Considerations:**

* Simple and intuitive interface for easy language selection and file/image upload.
* Clear output display with an option to download the translated file.
* Support for multiple languages in the selection dropdown.
* Mobile-responsive design.

### ****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 | Charan | API Key, Python, Streamlit setup | API connection established & working |
| **Sprint 1** | Frontend UI Development | 🟡 Medium | 2 hours (Day 1) | End of Day 1 | Aravind | API response format finalized | Basic UI with input fields |
| **Sprint 2** | File & Image Input Processing | 🔴 High | 4 hours (Day 2) | Mid-Day 2 | Hameed | Image processing libraries | File/image processing integrated |
| **Sprint 2** | Translation Model Integration | 🔴 High | 3 hours (Day 2) | Mid-Day 2 | Charan | API setup and model integration | Translation model integrated |
| **Sprint 3** | Testing & UI Enhancements | 🟡 Medium | 2 hours (Day 2) | Mid-Day 2 | Harshith | File/imageupload integration | Responsive UI, optimized file uploads |
| **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 translation API.
* 🟡 Medium Priority: Build basic UI for text input.

**Sprint 2 – Core Features & Debugging (Day 2)**

* 🔴 High Priority: Implement file and image input processing.
* 🔴 High Priority: Integrate translation model with backend.

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

* 🟡 Medium Priority: Test file/image input processing and translation accuracy.
* 🟢 Low Priority: Final demo preparation & deployment.

### ****Phase 5: Project Development****

#### **Objective:**

Implement the core features of the multi-language translator tool.

#### **Key Points:**

**Technology Stack Used:**

* **Frontend:** Streamlit
* **Backend:** AI translation model (Google Cloud Translation API, OpenAI Models)
* **Programming Language:** Python
* **Libraries:** Tesseract (OCR), PyPDF2 (PDF), python-docx (Word)

**Development Process:**

* Implement API key authentication and translation API integration.
* Develop file and image input processing logic.
* Optimize translation speed and accuracy.

**Challenges & Fixes:**

* **Challenge:** OCR accuracy for image input.  
  **Fix:** Use advanced pre-processing for image clarity and multiple OCR libraries.
* **Challenge:** Large file sizes for document uploads.  
  **Fix:** Implement file size restrictions and batch processing for large files.

### ****Phase 6: Functional & Performance Testing****

#### **Objective:**

Ensure that the multi-language translator tool functions correctly and performs optimally.

| **Test Case ID** | **Category** | **Test Scenario** | **Expected Outcome** | **Status** | **Tester** |
| --- | --- | --- | --- | --- | --- |
| **TC-001** | Functional Testing | Translate "Hello" from English to Spanish | "Hola" should be displayed. | ✅ Passed | Aravind |
| **TC-002** | Functional Testing | Translate a document from English to French | The full document should be translated. | ✅ Passed | Charan |
| **TC-003** | Performance Testing | OCR accuracy for an image with text | Text should be correctly extracted. | ⚠ Needs Optimization | Harshith |
| **TC-004** | Bug Fixes & Improvements | Handle large PDF files properly | Large files should be processed without error. | ✅ Fixed | Aravind |
| **TC-005** | Final Validation | Ensure mobile responsiveness | UI should work on mobile & desktop. | FIXED | Charan |
| **TC-006** | Deployment Testing | Host the app using Streamlit Sharing | App should be accessible online. | 🚀 Deployed | Hameed |

**Final Submission**

1. **Project Report Based on the templates**
2. **GitHub/Code Repository Link**
3. **Presentation**