Hackathon Challenge: Form-to-Conversational ChatBot Tool

# Problem Statement

Design and build a Form-to-Convo Tool that can convert any HTML form—including complex, multi-page, wizard-style forms—into an intelligent, multilingual, conversational chatbot. The chatbot should interpret the form, interact with users, validate their inputs, and auto-submit the form based on confirmed responses. The Form-to-Convo Tool should support:

* Support for multi-page “wizard” style forms
* Conversation preview interface (text or basic web UI)
* RAG-based prompt enrichment using form metadata
* Must allow interface in a **preferred user language** while keeping form data in its **original language (**Use open-source multilingual models like [IndicTrans2](https://github.com/AI4Bharat/IndicTrans2))
* Support a simple React based UI for the form to conversational chatbot that allows user to provide a http url that points to a form or upload a html page that contains a form and the UI then provides a chat interface to collect the inputs from the user and fill up the form.
* Support API based interaction with the tool.

# Solution Overview

## Example Input Form (HTML Content)

<form id="scholarshipForm">

<label for="name">Full Name:</label>

<input type="text" id="name" name="fullname" required>

<label for="dob">Date of Birth:</label>

<input type="date" id="dob" name="dob" required>

<label for="category">Caste Category:</label>

<select id="category" name="category">

<option value="gen">General</option>

<option value="obc">OBC</option>

<option value="sc">SC</option>

<option value="st">ST</option>

</select>

<label for="income">Annual Family Income (in ₹):</label>

<input type="number" id="income" name="income">

<button type="submit">Submit Application</button>

</form>

## Expected Outcome

* A **conversational chatbot** that:
  + Reads and understands the form structure
  + Asks questions like “What is your full name?” or “What is your caste category?”
  + Detects and handles invalid/irrelevant responses (e.g., income as “ABC”)
  + Can translate form prompts and user responses across Indian languages (e.g., Hindi → Telugu)
  + Auto-fills and submits the original form via a headless browser/API on user confirmation

## Implementation Guidance (Indicative Only)

## Architecture Sketch:

HTML Parser → Field Extractor → Prompt Generator → Conversational Agent (LangChain)

→ Input Validator → Multilingual Translator

→ Form Auto-filler (headless browser/API) → Submit

## Suggested Phases:

1. HTML Form Parsing:

* Use BeautifulSoup, Selenium, or Puppeteer to parse and navigate the DOM
* Extract fields, labels, input types, validation constraints

1. Prompt Generation:

* Generate natural language questions using field metadata
* Use heuristics or LLM (e.g. LLaMA or GPT) via LangChain

1. Conversational Agent:

* Use LangChain agents or LangGraph to orchestrate the conversation
* Add basic fallback logic for irrelevant input handling

1. Translation:

* Integrate with Bhashini, IndicTrans, or Google Translate API for multi-language support

1. Form Auto-filler & Submit:

* Use Selenium or Playwright to auto-populate the form fields and simulate submission on user confirmation

1. API Support:

* Expose APIs to allow programmatic form ingestion and chatbot invocation
* Sample: /generate-chatbot, /submit-form

## Tech Stack Recommendations

|  |  |
| --- | --- |
| Functionality | Recommended Tech |
| HTML Parsing | BeautifulSoup, Selenium, Puppeteer |
| LLM + Agents | Python, LangChain, LangGraph |
| Translation | IndicTrans2, Bhashini API, Google Translate |
| Conversational Interface | Gradio, Streamlit, or React |
| Form Autofill | Selenium, Playwright |
| API Support | FastAPI, Flask |
| Deployment | Docker, GitHub |

## Example Datasets on Kaggle / Open Source

 Indian Government Schemes Dataset – for prompt simulation

 Multi-language Form Data (Synthetic) – simulate multilingual prompts

 HTML Form Parsing Examples (via Selenium)

https://paperswithcode.com/dataset/forms-dataset?utm\_source=chatgpt.com

## Note

This suggested architecture and flow is indicative only. Participants are encouraged to innovate and improve upon it with better ideas or open-source tools of their choice. Evaluation will consider completeness, robustness, usability, and creativity.

# Key Deliverables

# Standalone Python component + APIs

# Demo video showing conversational interaction + form autofill/submit (preferably in GCP)

# Brief design doc explaining architecture and components.