🌍 AI Tour Guide App – Full Overview

This app acts like a multilingual tour guide. You type a country name, and it:

1. Describes the country in **English** using a **Large Language Model (LLM)**.
2. Translates that into **Bangla**.
3. Shows **factual data** (capital, population) using a public **REST API**.

🔁 Step-by-Step Flow

✅ Step 1: User Input

python
CopyEdit
```
country = gr.Textbox(label="Enter a country name")
```

- Gradio input box captures the **country name** (e.g., `"Japan"`).

🤖 Step 2: Generate Description (LLM – Gemini)

python
CopyEdit
```
english_text = ask_gemini(f"Tell me about {country_name}")
```

📍 **Where:** In `ai_tour_guide()`
📦 **LLM Used:** `Gemini` via `Pipecat`
🧠 **Purpose:** Asks Gemini: *"Tell me about Japan"*
🔁 **Output:** A descriptive paragraph about Japan.

➡️ **This is the core LLM functionality.**

🌐 Step 3: Translate to Bangla (LLM or NMT)

python
CopyEdit
```
bangla_text = translate_to_bangla(english_text)
```

📍 **Where:** In `ai_tour_guide()`
📦 **Model Used:** `Gemini`, `Pipecat`, or any multilingual LLM
🧠 **Purpose:** Translates the English text into **Bangla**

➡️ This uses Gemini's translation ability (acts like Google Translate but with better fluency/context).

🌍 Step 4: Fetch Country Info (API)

python
CopyEdit
```
info = get_country_info(country_name)
```

📍 **Where:** In `ai_tour_guide()`
📦 **Data Source:** RESTCountries API
📄 **What It Fetches:**

- Capital
- Population
- Currency
- Languages
- Region

🛠️ **Defined in:** `country_info.py`

🧾 Step 5: Show All Info in UI

python
CopyEdit

```
return english_text, bangla_text, stats
```

📍 Returns three things:

- 🇬🇧 `Description (English)`
- 🇧🇩 `Description (Bangla)`
- 🗺️ `Quick Facts` (e.g., Capital: Tokyo, Population: 125 million)

📍 These are shown in Gradio `Markdown` blocks.

🧠 Summary: Where You Use Each Method

Step	Feature	Method Used
1	Text Input	Gradio Textbox
2	English Description	✅ Gemini LLM (via Pipecat)
3	Bangla Translation	✅ Gemini LLM (or Pipecat)
4	Capital, Population Info	🌐 RESTCountries API
5	Output to UI	Gradio UI Blocks

🗂️ Your File Structure

graphql
CopyEdit

```
ultra-voice-agent/
├── app.py              # Gradio frontend + main pipeline
├── country_info.py     # Country data from REST API
├── gemini_client.py    # LLM calls to Gemini
├── requirements.txt    # Packages
├── .huggingface.yaml   # For Hugging Face Space deployment
```

✅ Example Output

Input: `Bangladesh`

English (LLM):

> Bangladesh is a South Asian country bordered by India and Myanmar...

Bangla (LLM translation):

> বাংলাদেশ দক্ষিণ এশিয়ার একটি দেশ...

Facts (REST API):

> Capital: Dhaka, Population: 170 million

Api
import os API_KEY = os.getenv("GEMINI_KEY") import requests import os API_KEY = os.getenv("GEMINI_KEY") URL = f"https://generativelanguage.googleapis.com/v1beta/models/gemini-2.5-flash:generateContent?key={API_KEY}" HEADERS = {"Content-Type": "application/json"} def ask_gemini(prompt): payload = {"contents": [{"parts": [{"text": prompt}]}]} res = requests.post(URL, headers=HEADERS, json=payload) return res.json()['candidates'][0]['content']['parts'][0]['text'] def translate_to_bangla(english_text): prompt = f"Translate this to Bengali:\\n\\n{english_text}" return ask_gemini(prompt) explain

✅ Purpose

This script uses the **Gemini 2.5 Flash LLM API** to:

1. Generate a response to a user prompt (like “Tell me about Japan”)
2. Translate English to Bengali using the same LLM

🔐 1. Set API Key

python
CopyEdit
```
import os

API_KEY = os.getenv("GEMINI_KEY")
```

- You store your **Gemini API key** as an **environment variable** named `GEMINI_KEY`.
- `os.getenv("GEMINI_KEY")` securely retrieves it from your system environment.
- ❗ If `GEMINI_KEY` is not set, the API call will **fail**.

🌐 2. Define API Endpoint

python
CopyEdit
```
URL =
f"https://generativelanguage.googleapis.com/v1beta/models/gemini-2.5-f
lash:generateContent?key={API_KEY}"
HEADERS = {"Content-Type": "application/json"}
```

- This is the Gemini 2.5 Flash **endpoint** from Google's **Generative Language API**.
- You're calling the `generateContent` endpoint with your API key.
- The request will send a JSON payload, so you declare the `Content-Type`.

🧠 3. Function: ask_gemini(prompt)

python
CopyEdit
```
def ask_gemini(prompt):
    payload = {"contents": [{"parts": [{"text": prompt}]}]}
    res = requests.post(URL, headers=HEADERS, json=payload)
    return res.json()['candidates'][0]['content']['parts'][0]['text']
```

💡 What It Does:

- Takes a **prompt** (like "Tell me about France")
- Sends a **POST request** to Gemini API
- Gets a **response** and extracts only the **generated text**

⚙️ Internals:

- `payload` is the structured input Gemini expects.
- `res.json()` returns full API response.
- `['candidates'][0]['content']['parts'][0]['text']` navigates the JSON to extract just the text part.

✅ Example:

python
CopyEdit
```
ask_gemini("Tell me about Nepal")
# ➡️ "Nepal is a landlocked country in South Asia..."
```

🌐 4. Function: translate_to_bangla(english_text)

python
CopyEdit
```
def translate_to_bangla(english_text):
    prompt = f"Translate this to Bengali:\\n\\n{english_text}"
    return ask_gemini(prompt)
```

💡 What It Does:

- Takes an English sentence like:

```
"Nepal is a landlocked country in South Asia."
```

- Adds the prefix prompt:

```
"Translate this to Bengali:\n\nNepal is..."
```

- Sends it to Gemini for translation

✅ Example:

python
CopyEdit

```
translate_to_bangla("Nepal is a landlocked country in South Asia.")
# ➡️ "নেপাল দক্ষিণ এশিয়ার একটি স্থলবেষ্টিত দেশ।"
```

🧠 Summary Table

Function	Purpose	Uses Gemini API	Input	Output
`ask_gemini(prompt)`	General prompt to LLM	✅ Yes	e.g., `"Tell me about India"`	e.g., `"India is a country..."`
`translate_to_bangla(text)`	Translate English → Bangla using LLM	✅ Yes	e.g., `"India is..."`	e.g., `"ভারত একটি দেশ..."`

```
import requests


def get_country_info(name):
    try:
        url = f"https://restcountries.com/v3.1/name/{name}"
        res = requests.get(url)
        res.raise_for_status()
        data = res.json()[0]


        return {
            "name": data.get("name", {}).get("common", name),
            "official_name": data.get("name", {}).get("official",
"N/A"),
            "capital": data.get("capital", ["N/A"])[0],
            "population": data.get("population", "N/A"),
            "currency": list(data.get("currencies", {}).keys())[0],
            "language": ', '.join(data.get("languages", {}).values()),
            "region": data.get("region", "N/A")
        }


    except Exception as e:
        return {"error": f"Unable to fetch info for '{name}': {e}"}
```

The function `get_country_info(name)` primarily uses these **methods**:

1. **`requests.get(url)`**
    - From the `requests` library.
    - Sends an HTTP GET request to the specified URL and returns a response object.
2. **`res.raise_for_status()`**
    - Method of the `requests.Response` object.
    - Raises an exception if the HTTP response status indicates an error (4xx or 5xx).

3. **`res.json()`**
    - Parses the JSON content of the HTTP response into a Python object (usually a dict or list).
4. **`dict.get(key, default)`**
    - Built-in Python method for dictionaries.
    - Safely gets the value for `key`; if not found, returns `default`.
5. **`list()`**
    - Built-in Python function to create a list from an iterable.
6. **`str.join(iterable)`**
    - Built-in string method.
    - Joins elements of the iterable into a string separated by the string on which it's called (here, `', '`

```
import gradio as gr
from gemini_client import ask_gemini, translate_to_bangla
from country_info import get_country_info  # Uses restcountries.com


def ai_tour_guide(country_name):
    if not country_name.strip():
        return "⚠️ Please enter a country name.", "", ""


    # Gemini Description
    try:
        english_text = ask_gemini(f"Tell me about {country_name}")
    except Exception as e:
        english_text = f"⚠️ Gemini error: {e}"


    # Translate to Bangla
    try:
        bangla_text = translate_to_bangla(english_text)
    except Exception as e:
        bangla_text = f"⚠️ অনুবাদ ব্যর্থ হয়েছে: {e}"


    # Country Info (capital, population)
    try:
        info = get_country_info(country_name)
        if "error" in info:
            raise ValueError(info["error"])
        stats = f"Capital: {info.get('capital', 'N/A')}, Population:
{info.get('population', 'N/A')}"
    except Exception as e:
        stats = f"⚠️ Country info error: {e}"


    return english_text, bangla_text, stats


with gr.Blocks() as demo:
    gr.Markdown("## 🌍 AI Tour Guide (English ➡️ Bangla)")
    country = gr.Textbox(label="Enter a country name")
    btn = gr.Button("Explore")

    out_en = gr.Markdown(label="🇬🇧 Description (English)")
    out_bn = gr.Markdown(label="🇧🇩 Description (Bengali)")
    stats = gr.Markdown(label="🗺️ Quick Facts")


    btn.click(fn=ai_tour_guide, inputs=country, outputs=[out_en, out_bn,
stats])


demo.launch()
```

Purpose:

This is a simple AI-powered **Tour Guide** web app built using **Gradio** that:

- Takes a country name as input.
- Fetches a description about the country in English using **Gemini** (a language model API).

- Translates the English description into **Bangla (Bengali)**.
- Fetches quick factual info (capital and population) from the **REST Countries API**.
- Displays all three outputs in the UI.

Breakdown of the code:

python
CopyEdit

```
import gradio as gr
from gemini_client import ask_gemini, translate_to_bangla
from country_info import get_country_info
```

- Imports:
  - **Gradio**: To create the web interface.
  - `ask_gemini`: Function that calls Gemini API for text generation.
  - `translate_to_bangla`: Function that translates English text to Bangla.
  - `get_country_info`: Function that fetches country info from REST Countries API.

---

python
CopyEdit

```
def ai_tour_guide(country_name):
    if not country_name.strip():
        return "⚠️ Please enter a country name.", "", ""
```

- Defines the main function called when the user clicks the button.
- First, it checks if the input is empty or only spaces. If yes, returns a warning message in English and empty outputs for Bangla and stats.

---

python
CopyEdit
```
  # Gemini Description
   try:
       english_text = ask_gemini(f"Tell me about {country_name}")
   except Exception as e:
       english_text = f"⚠️ Gemini error: {e}"
```

- Calls the Gemini language model API with a prompt like `"Tell me about France"`.
- Stores the returned English description.
- If there's an error (e.g., API failure), it catches and returns an error message.

---

python
CopyEdit
```
  # Translate to Bangla
   try:
       bangla_text = translate_to_bangla(english_text)
   except Exception as e:
       bangla_text = f"⚠️ অনুবাদ ব্যর্থ হয়েছে: {e}"
```

- Uses another function to translate the English text to Bangla.
- If the translation fails, it returns a Bangla error message (meaning "translation failed").

---

python
CopyEdit
```
  # Country Info (capital, population)
   try:
       info = get_country_info(country_name)
       if "error" in info:
           raise ValueError(info["error"])
       stats = f"Capital: {info.get('capital', 'N/A')}, Population:
{info.get('population', 'N/A')}"
```

```
    except Exception as e:
        stats = f"⚠️ Country info error: {e}"
```

- Calls the country info function to get structured info about the country.
- If there's an error key in the result (meaning API failure or no data), raises an exception.
- Formats the capital city and population into a quick summary string.
- If any error occurs, sets an error message for the stats section.

---

python
CopyEdit
```
   return english_text, bangla_text, stats
```

- Returns three strings as output: English description, Bangla translation, and quick country facts.

Gradio Interface:

python
CopyEdit
```
with gr.Blocks() as demo:
    gr.Markdown("## 🌍 AI Tour Guide (English ➡️ Bangla)")
    country = gr.Textbox(label="Enter a country name")
    btn = gr.Button("Explore")

    out_en = gr.Markdown(label="🇬🇧 Description (English)")
    out_bn = gr.Markdown(label="🇧🇩 Description (Bengali)")
    stats = gr.Markdown(label="🗺️ Quick Facts")

    btn.click(fn=ai_tour_guide, inputs=country, outputs=[out_en,
out_bn, stats])

demo.launch()
```

- Creates a Gradio **Blocks** interface named `demo`.
- Adds a heading.
- Adds a textbox for user input (country name).
- Adds a button labeled "Explore".
- Creates three Markdown output areas to display:
    - English description.
    - Bangla description.
    - Quick facts.
- Connects the button's click event to the `ai_tour_guide` function, linking the input textbox to the function's input, and mapping the three outputs to the three Markdown outputs.
- Finally, launches the Gradio app.

Summary:

When you run this code and open the web app:

- You enter a country name.
- Press "Explore".
- The app calls Gemini to get an English description about that country.
- Translates that description into Bangla.
- Fetches quick facts like capital and population.
- Shows all the info neatly in three sections.