AI Braille Assistant

```
import os
import json
import time
import random
import torch
from transformers import pipeline
from gtts import gTTS
from rich import print as rprint

# ========== SETTINGS ==========
DEVICE = "cuda" if torch.cuda.is_available() else "cpu"
USER_DATA_FILE = "user_data.json"
SUPPORTED_LANGUAGES = ["english", "tamil", "hindi", "kannada", "bengali"]

# ========== LOAD LLM ==========
rprint("[yellow]Loading LLM model...[/]")
llm = pipeline("text2text-generation", model="google/flan-t5-base", device=0 if DEVICE == "cuda" else -1)

# ========== BRAILLE DICTIONARIES ==========
# Only a few entries per language here for brevity. Extend as needed!
braille_dicts = {
    'english': {'a': '⠁', 'b': '⠃', 'c': '⠉', 'd': '⠙', 'e': '⠑', 'f': '⠋', 'g': '⠛', 'h': '⠓', 'i': '⠊', 'j': '⠚', 'k': '⠅', 'l': '⠇', 'm': '⠍', 'n': '⠝', 'o': '⠕', 'p': '⠏', 'q': '⠟', 'r': '⠗', 's': '⠎', 't': '⠞', 'u': '⠥', 'v': '⠧', 'w': '⠺', 'x': '⠭', 'y': '⠽', 'z': '⠵', ' ': ' '},
    'tamil': {'அ': '⠁', 'ஆ': '⠜', 'இ': '⠊', 'ஈ': '⠔', 'உ': '⠥', 'ஊ': '⠳', 'எ': '⠑', 'ஏ': '⠮', 'ஐ': '⠌', 'ஒ': '⠪', 'ஓ': '⠣', 'ஔ': '⠬', 'க': '⠅', 'ச': '⠉', 'ட': '⠞', 'த': '⠹', 'ப': '⠏', 'ம': '⠍', 'ய': '⠽', 'ர': '⠗', 'ல': '⠇', 'வ': '⠧', 'ழ': '⠵', 'ள': '⠻', 'ற': '⠼', 'ன': '⠝', ' ': ' '},
```

'hindi': {'अ': '⠁', 'आ': '⠜', 'इ': '⠊', 'ई': '⠔', 'उ': '⠥', 'ऊ': '⠳', 'ऋ': '⠻', 'ए': '⠑', 'ऐ': '⠌', 'ओ': '⠕', 'औ': '⠬', 'क': '⠅', 'ग': '⠛', 'च': '⠉', 'ज': '⠚', 'ट': '⠞', 'त': '⠹', 'द': '⠙', 'न': '⠝', 'प': '⠏', 'फ': '⠋', 'ब': '⠃', 'म': '⠍', 'य': '⠽', 'र': '⠗', 'ल': '⠇', 'व': '⠧', 'श': '⠩', 'ह': '⠓', ' ': ' '},

'kannada': {'ಅ': '⠁', 'ಆ': '⠜', 'ಇ': '⠊', 'ಈ': '⠔', 'ಉ': '⠥', 'ಊ': '⠳', 'ಋ': '⠻', 'ಎ': '⠑', 'ಐ': '⠌', 'ಒ': '⠕', 'ಕ': '⠅', 'ಗ': '⠛', 'ಚ': '⠉', 'ಜ': '⠚', 'ಟ': '⠞', 'ಡ': '⠙', 'ತ': '⠹', 'ದ': '⠙', 'ನ': '⠝', 'ಪ': '⠏', 'ಬ': '⠃', 'ಮ': '⠍', 'ಯ': '⠽', 'ರ': '⠗', 'ಲ': '⠇', 'ವ': '⠧', 'ಶ': '⠩', 'ಸ': '⠎', 'ಹ': '⠓', ' ': ' '},

'bengali': {'অ': '⠁', 'আ': '⠜', 'ই': '⠊', 'ঈ': '⠔', 'উ': '⠥', 'ঊ': '⠳', 'এ': '⠑', 'ঐ': '⠌', 'ও': '⠕', 'ঔ': '⠬', 'ক': '⠅', 'খ': '⠫', 'গ': '⠛', 'ঘ': '⠣', 'চ': '⠉', 'ছ': '⠡', 'জ': '⠚', 'ট': '⠞', 'ড': '⠙', 'ত': '⠹', 'দ': '⠙', 'ন': '⠝', 'প': '⠏', 'ফ': '⠋', 'ব': '⠃', 'ভ': '⠧', 'ম': '⠍', 'য': '⠽', 'র': '⠗', 'ল': '⠇', 'শ': '⠩', 'স': '⠎', 'হ': '⠓', ' ': ' '}

}

```
# ========== UTILITIES ==========

def speak(text):
    tts = gTTS(text=text, lang='en')
    tts.save("speak.mp3")
    os.system("start speak.mp3" if os.name == 'nt' else "afplay speak.mp3")


def detect_language(text):
    scores = {}
    for lang, braille_dict in braille_dicts.items():
        count = sum(1 for char in text if char in braille_dict)
        scores[lang] = count
    best_match = max(scores, key=scores.get)
    return best_match if scores[best_match] > 0 else 'english'


def translate_to_braille(text, lang):
    d = braille_dicts.get(lang, {})
    return ''.join(d.get(c, c) for c in text)


# ========== DATA TRACKING ==========

def load_user_data():
    if not os.path.exists(USER_DATA_FILE):
```

```
        return {"scores": [], "mistakes": {}}
    with open(USER_DATA_FILE, "r") as f:
        return json.load(f)

def save_user_data(data):
    with open(USER_DATA_FILE, "w") as f:
        json.dump(data, f, indent=2)

def update_performance(correct, incorrect, mistakes):
    data = load_user_data()
    data["scores"].append({"correct": correct, "incorrect": incorrect, "timestamp": time.time()})
    for ch in mistakes:
        data["mistakes"][ch] = data["mistakes"].get(ch, 0) + 1
    save_user_data(data)

def get_suggestion():
    data = load_user_data()
    if not data["scores"]:
        return "Start with basic letters in your preferred language."
    recent = data["scores"][-3:]
    total = sum(s["correct"] + s["incorrect"] for s in recent)
    acc = sum(s["correct"] for s in recent) / max(1, total)
    if acc > 0.8:
        return "You're doing great! Try full word Braille practice."
    elif acc < 0.5:
        return "Focus on individual letter recognition and repeat past tests."
    return "You’re making good progress. Keep practicing mixed letters."

# ========== INTENT-BASED FUNCTION CALL ==========
def call_function_by_intent(user_input):
    prompt = f"""You are a Braille teaching assistant.
```

```
Understand user commands and classify their intent as one of these:
- translate
- quiz
- teach braille
- suggest
- qna
Only return the intent word.

Input: {user_input}
Intent:"""
    result = llm(prompt, num_return_sequences=1)[0]['generated_text'].strip().lower()
    return result

# ========== CHAT INTERFACE ==========
def ai_braille_chat():
    rprint("[bold green]Welcome to the AI Braille Assistant[/]")
    rprint("[bold yellow]Ask anything. Commands: translate, quiz, suggest, or Braille questions. Type 'exit' to quit.[/]\n")

    while True:
        user_input = input("You: ").strip()
        if user_input.lower() == "exit":
            rprint("[bold yellow]Goodbye![/]")
            break

        intent = call_function_by_intent(user_input)

        if "translate" in intent:
            text = input("Text to convert to Braille: ")
            lang = detect_language(text)
            braille = translate_to_braille(text, lang)
```

```
            rprint(f"[magenta]Detected: {lang}[/]\n[cyan]Braille Output:[/] {braille}")
            speak(f"Braille output is {braille}")

        elif "quiz" in intent:
            lang = input("Which language (english, tamil, etc.): ").lower()
            if lang not in braille_dicts:
                rprint("[red]Unsupported language![/]")
                continue
            letters = list(braille_dicts[lang].keys())
            sample = random.sample(letters, min(5, len(letters)))
            responses = {}
            for ch in sample:
                ans = input(f"Braille for '{ch}': ")
                responses[ch] = ans
            correct = sum(1 for ch in sample if responses[ch] == braille_dicts[lang][ch])
            update_performance(correct, len(sample) - correct, [ch for ch in sample if responses[ch] !=
braille_dicts[lang][ch]])
            rprint(f"[bold cyan]Quiz Score: {correct}/{len(sample)}[/]")

        elif "suggest" in intent:
            suggestion = get_suggestion()
            rprint(f"[bold blue]AI Suggestion:[/] {suggestion}")
            speak(suggestion)

        elif "qna" in intent or "question" in user_input.lower():
            answer = llm(user_input, num_return_sequences=1)[0]['generated_text']
            rprint(f"[bold green]Answer:[/] {answer}")
            speak(answer)

        else:
            reply = llm(user_input, num_return_sequences=1)[0]['generated_text']
```

```
        rprint(f"[bold white]Chat:[/] {reply}")


if _name_ == "_main_":
    ai_braille_chat()
```

OUTPUT :

```
Welcome to the AI Braille Assistant
Ask anything. Commands: translate, quiz, suggest, or Braille questions. Type 'exit' to quit.

You: Braille questions
Answer: - What is the first letter of the letter b?
You:
```