```
import asyncio
import os
from typing import Optional

from dotenv import load_dotenv

from livekit import agents
from livekit.agents import Agent, AgentSession, RoomInputOptions

# --- Plugins ---
from livekit.plugins import (
    google,              # livekit-plugins-google (Gemini realtime + Google TTS)
    noise_cancellation,   # livekit-plugins-noise-cancellation
    deepgram,            # livekit-plugins-deepgram (for STT)
)

load_dotenv()

# --- Configuration ---
GOOGLE_API_KEY = os.getenv("GOOGLE_API_KEY")
DEEPGRAM_API_KEY = os.getenv("DEEPGRAM_API_KEY")

SUPPORTED_LANGS = ["hi", "en"]
DEFAULT_GREETING = (
    "नमस्ते! (Namaste!) Hello! I can speak in Hindi and English. "
    "How can I help you today?"
)
```

```
class Assistant(Agent):
    """High-level Agent personality & behavior."""

    def __init__(self) -> None:
        super().__init__(
            instructions=(
                "You are a helpful multilingual voice AI assistant. "
                    "When the user speaks, automatically detect the language and respond in that
language. "
                "Keep replies concise and conversational."
            )
        )

class LanguageRouter:
    """Determines reply language from last STT result; defaults to English."""

    last_lang: str = "en"

    def update(self, stt_result_lang: Optional[str]):
        if stt_result_lang and stt_result_lang in SUPPORTED_LANGS:
            self.last_lang = stt_result_lang

    def current(self) -> str:
        return self.last_lang

async def build_stt():
    """Create Deepgram STT engine (multilingual). Reads DEEPGRAM_API_KEY from env."""
    if not DEEPGRAM_API_KEY:
        raise RuntimeError("DEEPGRAM_API_KEY missing in environment.")



    return deepgram.STT(
        model="nova-2"
    )

async def build_tts():
    """Create TTS engine (Google neural voices)."""
    if not GOOGLE_API_KEY:
        raise RuntimeError("GOOGLE_API_KEY missing in environment.")

    '''return google.tts.StreamingTTS(
        api_key=GOOGLE_API_KEY,
        voices={
            "hi": "hi-IN-Neural2-A",
            "en": "en-US-Neural2-C",
        },
    )'''

async def build_llm():
    """Create the real-time LLM (Gemini)."""
    if not GOOGLE_API_KEY:
        raise RuntimeError("GOOGLE_API_KEY missing in environment.")

    return google.beta.realtime.RealtimeModel(
        api_key=GOOGLE_API_KEY,
        model="gemini-2.0-flash-exp",
        voice="Puck",
        temperature=0.7,
        instructions=(
            "Be polite and concise. Mirror the user's language (Hindi or English). "
            "If you cannot determine the language, prefer English."
        ),
    )

async def entrypoint(ctx: agents.JobContext):
    """Worker entrypoint used by LiveKit Agents CLI."""
    # Build pipeline components
    stt_engine = await build_stt()
    #tts_engine = await build_tts()
    llm_engine = await build_llm()



    lang_router = LanguageRouter()



    if hasattr(stt_engine, "on_final"):
        def _on_final(result):
            # e.g., result.lang may be "hi" or "en"; fall back handled by LanguageRouter
            lang_router.update(getattr(result, "lang", None))
        stt_engine.on_final(_on_final)



    session = AgentSession(
        llm=llm_engine,
        stt=stt_engine,
        #tts=tts_engine,
    )

    # Start media & agent
    await session.start(
        room=ctx.room,
        agent=Assistant(),
        room_input_options=RoomInputOptions(
            noise_cancellation=noise_cancellation.BVC(),
        ),
    )


    await session.generate_reply(instructions=DEFAULT_GREETING)



if __name__ == "__main__":
    agents.cli.run_app(agents.WorkerOptions(entrypoint_fnc=entrypoint))
```