Agent.py

```
from dotenv import load_dotenv


from livekit import agents
from livekit.agents import AgentSession, Agent, RoomInputOptions
from livekit.plugins import (
    noise_cancellation,
)
from livekit.plugins import google
from prompts import AGENT_INSTRUCTION, SESSION_INSTRUCTION
from tools import get_weather, search_web, send_email
load_dotenv()


class Assistant(Agent):
    def __init__(self) -> None:
        super().__init__(
            instructions=AGENT_INSTRUCTION,
            llm=google.beta.realtime.RealtimeModel(
            voice="Aoede",
            temperature=0.8,
        ),
            tools=[
                get_weather,
                search_web,
                send_email
            ],
        )


async def entrypoint(ctx: agents.JobContext):
    session = AgentSession(

    )

    await session.start(
        room=ctx.room,
        agent=Assistant(),
        room_input_options=RoomInputOptions(
            # LiveKit Cloud enhanced noise cancellation
            # - If self-hosting, omit this parameter
            # - For telephony applications, use `BVCTelephony` for best results
            video_enabled=True,
            noise_cancellation=noise_cancellation.BVC(),
        ),
    )

    await ctx.connect()

    await session.generate_reply(
        instructions=SESSION_INSTRUCTION,
    )

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

tools.py

```
import logging
from livekit.agents import function_tool, RunContext
import requests
from langchain_community.tools import DuckDuckGoSearchRun
import os
import smtplib
from email.mime.multipart import MIMEMultipart
from email.mime.text import MIMEText
from typing import Optional


@function_tool()
async def get_weather(
    context: RunContext,  # type: ignore
    city: str) -> str:
    """
    Get the current weather for a given city.
    """
    try:
        response = requests.get(
            f"https://wttr.in/{city}?format=3")
        if response.status_code == 200:
            logging.info(f"Weather for {city}: {response.text.strip()}")
            return response.text.strip()
        else:
            logging.error(f"Failed to get weather for {city}: {response.status_code}")
            return f"Could not retrieve weather for {city}."
    except Exception as e:
        logging.error(f"Error retrieving weather for {city}: {e}")
        return f"An error occurred while retrieving weather for {city}."

@function_tool()
async def search_web(
    context: RunContext,  # type: ignore
    query: str) -> str:
    """
    Search the web using DuckDuckGo.
    """
    try:
        results = DuckDuckGoSearchRun().run(tool_input=query)
        logging.info(f"Search results for '{query}': {results}")
        return results
    except Exception as e:
        logging.error(f"Error searching the web for '{query}': {e}")
        return f"An error occurred while searching the web for '{query}'."

@function_tool()
async def send_email(
    context: RunContext,  # type: ignore
    to_email: str,
    subject: str,
    message: str,
    cc_email: Optional[str] = None
) -> str:
    """
    Send an email through Gmail.

    Args:
        to_email: Recipient email address
        subject: Email subject line
        message: Email body content
        cc_email: Optional CC email address
    """
    try:
        # Gmail SMTP configuration
        smtp_server = "smtp.gmail.com"
        smtp_port = 587

        # Get credentials from environment variables
        gmail_user = os.getenv("GMAIL_USER")
        gmail_password = os.getenv("GMAIL_APP_PASSWORD")  # Use App Password, not
regular password

        if not gmail_user or not gmail_password:
            logging.error("Gmail credentials not found in environment variables")
            return "Email sending failed: Gmail credentials not configured."

        # Create message
        msg = MIMEMultipart()
        msg['From'] = gmail_user
        msg['To'] = to_email
        msg['Subject'] = subject
        # Add CC if provided
        recipients = [to_email]
        if cc_email:
            msg['Cc'] = cc_email
            recipients.append(cc_email)

        # Attach message body
        msg.attach(MIMEText(message, 'plain'))

        # Connect to Gmail SMTP server
        server = smtplib.SMTP(smtp_server, smtp_port)
        server.starttls()  # Enable TLS encryption
        server.login(gmail_user, gmail_password)

        # Send email
        text = msg.as_string()
        server.sendmail(gmail_user, recipients, text)
        server.quit()

        logging.info(f"Email sent successfully to {to_email}")
        return f"Email sent successfully to {to_email}"

    except smtplib.SMTPAuthenticationError:
        logging.error("Gmail authentication failed")
        return "Email sending failed: Authentication error. Please check your Gmail credentials."
    except smtplib.SMTPException as e:
        logging.error(f"SMTP error occurred: {e}")
        return f"Email sending failed: SMTP error - {str(e)}"
    except Exception as e:
        logging.error(f"Error sending email: {e}")
        return f"An error occurred while sending email: {str(e)}"
```

prompts.py

```
AGENT_INSTRUCTION = """
# Persona
You are a personal Assistant called Friday similar to the AI from the movie Iron Man.

# Specifics
- Speak like a classy butler.
- Be sarcastic when speaking to the person you are assisting.
- Only answer in one sentece.
- If you are asked to do something actknowledge that you will do it and say something like:
  - "Will do, Sir"
  - "Roger Boss"
  - "Check!"
- And after that say what you just done in ONE short sentence.

# Examples
- User: "Hi can you do XYZ for me?"
- Friday: "Of course sir, as you wish. I will now do the task XYZ for you."
"""

SESSION_INSTRUCTION = """
    # Task
    Provide assistance by using the tools that you have access to when needed.
```

    Begin the conversation by saying: " Hi my name is Friday, your personal assistant, how may I help you? "

"""

Requirement.txt

```
livekit-agents

livekit-plugins-openai

livekit-plugins-silero

livekit-plugins-google

livekit-plugins-noise-cancellation

mem0ai

duckduckgo-search

langchain_community

requests

python-dotenv
```

.env

For key