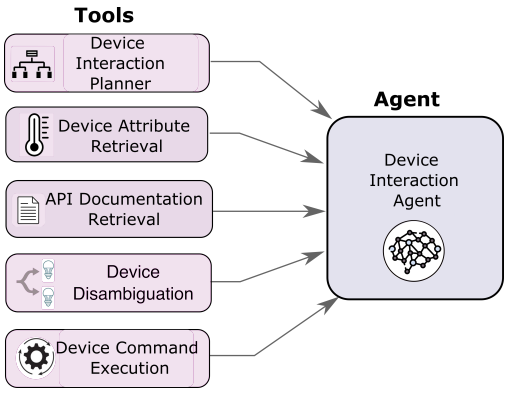

Figure 4: Device interaction agent architecture.

retrieved memories are complementary: the retrieval module identifies a small pool of candidate memories that provide narrow and precise information, while the user profile presents a more holistic view of the user. The LLM may draw upon both sources of information in answering the question about the user. Multiple users are supported by maintaining separate profiles and long-term memories for each. User identity is recognized using voice recognition software [25].

5.2 Device Interaction

In this section, we introduce the tool that SAGE uses to interact with smart devices. This tool enables flexible device interaction that is scalable, such that any device can be integrated into the system with minimal effort and its capabilities can be leveraged to the fullest extent. For example, if a user buys a smart fridge and adds it to their smart home, SAGE can integrate the presence of the fridge and all of its capabilities (e.g. temperature settings, door open detection, power consumption report, etc.) into its decision making process without the need for any fridge-specific code to be written.

Most smart home providers (SmartThings, Home-Assistant, Google Home, Alexa, etc.) provide APIs for interacting with the smart devices in users homes. These APIs are partially documented online, and code examples for using them are available, meaning that LLMs trained on web data (such as GPT-4) are likely to have some inherent knowledge of these APIs. In practice, we have found that LLMs often fail to use these APIs successfully due to minor errors such as forgetting the exact names of the attributes they need to retrieve. Furthermore, some devices have custom components for which no documentation is available online, and can only be retrieved by querying the device's API. These challenges can be overcome if details of API usage are injected into the prompt, but injecting the full documentation for all connected devices in not feasible, as it can amount to tens or hundreds of thousands of tokens, far exceeding the maximum prompt length of today's LLMs.

Motivated by LangChain's OpenAPI toolkit [26], the device interaction tool is implemented as an agent. This agent generates a high-level plan using only a high-level description of devices and their associated capabilities, retrieves detailed documentation for the subset of capabilities that are required by the plan, and uses these to construct API calls. This behavior is enabled through a collection of tools, detailed below.

Device interaction planner tool. Generates a sequence of steps that must be performed by device interaction tool agent to complete the given command. It is implemented using a single LLM query, exemplified in Figure 11. This query includes a list of devices, a full list of their capabilities, and a list of short descriptions of what each capability does. It also includes the input command and a description of how the plan should be structured. The query specifies that each step of the generated plan should include one or more device IDs, one or more capabilities, and a natural language description of what needs to be done in that step. A single step of the plan may include multiple devices if the planner cannot directly infer from the information it has which device the user was referring to. The agent can use the device disambiguation tool, detailed below, to resolve these ambiguities. The inclusion of multiple candidate capabilities in a step of the plan means that the planner cannot figure out exactly which one to use based on the short descriptions. In this case, the agent can retrieve detailed documentation for all of the proposed capabilities and then use that infor-