# **Code Mapping: History Functionality Flow**

Here's a detailed mapping of how the history functionality works across your system:

## **1. Knowledge Base Initialization and Storage**

**File:** core/knowledge\_base.py

* **Class:** KnowledgeBase
* **Methods:**
  + set\_item() - Stores data
  + get\_item() - Retrieves data
  + set\_item\_async(), get\_item\_async() - Async versions

## **2. Session Management & History Storage**

**File:** main.py

* **Function:** interactive\_async\_main()
* **Lines:** ~20-30 (initialization of history structure)

python

Copy

if not kb.get\_item("session\_history"):

kb.set\_item("session\_history", {

"sessions": [],

"math\_questions": [],

"general\_questions": [],

"energy\_models": []

})

* **Lines:** ~200-250 (storing session data)

python

Copy

# Get existing history

history = kb.get\_item("session\_history")

# Add new session

history["sessions"].append(session\_data)

# Categorize prompts (math, general, model)

# ...

kb.set\_item("session\_history", history)

## **3. History Query Detection**

**File:** utils/general\_knowledge.py

* **Function:** answer\_general\_question()
* **Lines:** ~20-55 (detecting history queries)

python

Copy

# Check if the question is about previous session content

prompt\_lower = prompt.lower()

# Enhanced keywords for context detection

session\_keywords = ["last session", "previous session", "earlier session"]

question\_keywords = ["last question", "previous question"]

math\_keywords = ["math", "calculation", "equation", "+"]

model\_keywords = ["model", "energy", "renewable"]

history\_keywords = ["history", "earlier", "before", "past", "previous"]

# Detection logic

is\_history\_query = (

any(kw in prompt\_lower for kw in session\_keywords) or

any(kw in prompt\_lower for kw in question\_keywords) or

(any(kw in prompt\_lower for kw in history\_keywords) and

(any(kw in prompt\_lower for kw in math\_keywords) or

any(kw in prompt\_lower for kw in model\_keywords)))

)

## **4. History Retrieval and Processing**

**File:** utils/general\_knowledge.py

* **Function:** answer\_general\_question()
* **Lines:** ~60-120 (retrieving and processing history)

python

Copy

# Get session history from KB

history = kb.get\_item("session\_history")

# Determine the type of history (math, model, general)

is\_math\_query = any(kw in prompt\_lower for kw in math\_keywords)

is\_model\_query = any(kw in prompt\_lower for kw in model\_keywords)

# Get specific history type

if is\_model\_query:

model\_history = history.get("energy\_models", [])

latest\_model = model\_history[-1] # Get most recent entry

# Extract and format the response

response = (

f"The last energy model request was: '{latest\_model['prompt']}'\n\n"

f"The result was: {latest\_model['result']}"

)

## **5. Task Creation and Routing**

**File:** agents/nova.py

* **Function:** create\_task\_list\_from\_prompt\_async()
* **Lines:** ~30-60 (creating tasks based on intent categorization)

python

Copy

# Categorize intent using OpenAI

category = await open\_ai\_categorisation\_async(intent\_text, csv\_path)

# For history questions, they need to be categorized as "general\_question"

# to be properly routed to answer\_general\_question

if category.lower() in ["general\_question", "uncategorized"]:

task = Task(

name=f"Handle Intent: {intent\_text[:30]}...",

agent="Nova",

function\_name="answer\_general\_question",

args={"prompt": intent\_text}

)

## **6. Prompt Processing and Result Storage**

**File:** main.py

* **Function:** process\_prompt\_tasks()
* **Lines:** ~10-40 (processing tasks and storing results)

python

Copy

# Execute tasks

await agent.handle\_task\_async(task)

# Store results for history

if task.function\_name == "answer\_general\_question":

general\_answer = await kb.get\_item\_async("general\_answer")

# Result stored and will be available for future history queries

## **Data Flow for History Queries**

1. User asks: "what country did I ask for a model to be built for?"
2. Query is categorized as "general\_question" by OpenAI
3. Task is created with function\_name="answer\_general\_question"
4. answer\_general\_question detects this is a history query about models
5. Function retrieves session\_history from knowledge base
6. Function gets energy\_models array from the history
7. Function extracts the most recent model entry
8. Function formats and returns the information about the previous model

This is the complete flow of how history queries are detected, processed, and answered in your system.