1. **UserProxyAgent Initialization:**
   * **The log ERROR - agents.user\_proxy\_agent - user\_proxy\_agent.<module>:L34 - UserProxy received an unexpected LLM configuration type: <class 'autogen\_ext.models.openai.\_openai\_client.OpenAIChatCompletionClient'>. Expected a dict (llm\_config) or None. Initializing without LLM. shows your user\_proxy\_agent.py logic is working as intended: get\_llm\_client("UserProxy") (configured for "gemini") returns an OpenAIChatCompletionClient object, but your agent correctly identifies that the autogen\_agentchat.UserProxyAgent expects an llm\_config dict and thus sets its internal LLM config to None. This is fine if you want an LLM-less UserProxy.**
   * **If you want UserProxy to have an LLM, ensure its type in AGENT\_LLM\_MAPPINGS is "openai" so get\_llm\_client returns a dict.**
2. **Logging:**
   * **Your logging.basicConfig in main\_autogen.py is being called *after* config.py and other agent modules are imported. Modules often get their loggers via logging.getLogger(\_\_name\_\_) at import time. If basicConfig hasn't run yet, they might get default handlers or levels.**
   * **Move logging.basicConfig(...) to be the *very first executable line* in main\_autogen.py, before any other imports (except import logging and import sys itself). This ensures it configures the root logger before any other module tries to use it.**
3. **Selector Agent's Prompting:  
   The Selector LLM (Ollama gemma3:1b) sometimes outputs Python code or lengthy text instead of just an agent name. This makes the SelectorGroupChat re-prompt it.**
   * **Action: Refine the selector\_prompt in main\_autogen.py. Make it extremely clear and forceful that the *only* valid output is a single agent name from the provided list, with no other text, explanations, or code. You might add examples of good/bad outputs in the prompt.  
     Example addition to the prompt:  
     "...Only return the agent name from {participants}. For example, if CodeAgent should speak next, your entire response must be: CodeAgent"**
4. **Agent Capabilities (Tools):**
   * **CodeAgent (Ollama) and TradingAgent (Ollama) are initialized with function\_calling: False in their model\_info. This is correct if they are not using tools.**
   * **BacktesterAgent (Gemini) is initialized with function\_calling: True. This is correct if you intend for it to use tools (like PythonCodeExecutionTool). You'll need to actually provide the tools list during its AssistantAgent initialization in agents/backtester\_agent.py for it to use them. Currently, your log suggests it's initialized without tools (the ---------- MemoryQueryEvent (BacktesterAgent) ---------- is about memory, not tools).**
5. **Review Agent Logic and Prompts:**
   * **The agents are generating a lot of code, some of it repetitive or flawed. This is typical of LLMs without very precise instructions or good examples.**
   * **Refine the system messages for each agent to be more specific about their exact task, expected output format, and when to pass control.**
   * **For CodeAgent, tell it to only output the strategy class and necessary functions, not the full cerebro.run() or data loading, as BacktesterAgent should handle that.**
   * **For BacktesterAgent, if it's to use tools, its prompt should guide it to take the code, save it to a file, and then use its PythonCodeExecutionTool to run it via common\_logic.backtesting\_services.backtrader\_runner.py.**

**This run was a huge step forward! The core client initializations are working, and the agents are conversing. The main things to address now are:**

* **Fixing the ModelFamily import to remove the dummy warning and ensure correct type.**
* **Refining agent prompts (especially the Selector's) for more controlled behavior.**
* **Starting to integrate actual tool use for agents that need it (like BacktesterAgent).**
* **Ensuring logging is configured first for better debug visibility.**

**Recap Message for Next Time:**

"Hi, we're continuing with the multi-agent trading system using autogen-agentchat 0.6.1.

* **Current Status:**
  + Successfully initialized all main agents (CodeAgent, BacktesterAgent, TradingAgent, Selector) with their respective LLM clients (OllamaChatCompletionClient for Ollama models like gemma3:1b, and OpenAIChatCompletionClient for Gemini models like gemini-1.5-flash-latest).
  + UserProxyAgent is initialized as LLM-less (as autogen\_agentchat.agents.UserProxyAgent doesn't take llm\_config in \_\_init\_\_).
  + The main agent chat loop is running, with the Selector choosing agents and agents responding. We've seen several turns of conversation.
  + The GEMINI\_API\_KEY is now correctly picked up.
  + The previous AttributeError: 'NoneType' object has no attribute 'model\_info' related to client initialization seems resolved for active agents.
  + The Console and wrapped\_stream\_generator are displaying TextMessage objects correctly.
* **Last Trace Issues/Observations (from run ending approx. 2025-06-06 16:08:XX):**
  + **ModelFamily Import Warning:** Still seeing WARNING:root:config.<module>:L52 - autogen\_ext.models.ollama.ModelFamily not found. Using dummy. This needs to be fixed by removing the incorrect/redundant ModelFamily import attempt around line 52 in config.py and ensuring the correct one from autogen\_core.models is used.
  + **Logging Visibility:** Some INFO logs from config.py or agent init files weren't showing up until I made the root logger config in main\_autogen.py more forceful and set to DEBUG. We should ensure logging is configured optimally.
  + **Selector Agent Behavior:** The Selector LLM (Ollama gemma3:1b) sometimes outputs full code blocks or lengthy text instead of just a single agent name, requiring re-prompts by the SelectorGroupChat logic. Its prompt needs refinement.
  + **Agent Response Quality:** The LLM-generated code and analysis is a good starting point but has flaws (e.g., CodeAgent's initial Backtrader code, TradingAgent echoing). Agent system messages need refinement.
  + **Tool Usage Not Yet Implemented/Tested:** Agents like BacktesterAgent are configured for Gemini with function\_calling: True but are not yet actively using tools (like PythonCodeExecutionTool).
* **Next Steps We Were Discussing:**
  + **Clean up ModelFamily import in config.py.**
  + **Refine Selector agent's prompt.**
  + **Refine system messages for participant agents** (CodeAgent, BacktesterAgent, TradingAgent) for more focused and accurate outputs.
  + **Integrate and test actual tool usage** for BacktesterAgent (e.g., executing the generated backtesting code).
  + Further review and optimize logging configuration.
  + Address the UserProxyAgent's LLM configuration if it's decided it needs one (e.g., by using autogen.UserProxyAgent and patching produced\_message\_types).
  + Address the placeholder PythonCodeExecutionTool and executor in agents/backtester\_agent.py and ensure it's correctly defined and passed when tools are enabled.
* **Current Goal:** To get the BacktesterAgent to successfully receive code from CodeAgent, execute it using a tool, and report the results, with the Selector correctly managing the flow.

Let's start by fixing the ModelFamily import in config.py and then work on refining the Selector's prompt.