AIChat CLI: Capabilities and User Guide Summary

The aichat CLI is an all-in-one tool designed to integrate Large Language Models (LLMs) directly into your terminal workflow, offering a unified interface for chat, coding, automation, and advanced AI tasks.

I. Core Interaction Modes

You can interact with aichat in three primary ways:

Mode	Command Style	Description
1. One-Liner (CMD)	aichat "your prompt"	Ideal for quick, one-off questions, summaries, or instant output generation. It's fast and doesn't save conversation history unless specified.
2. Chat REPL	aichat -s [session_name]	A powerful, interactive **R**ead-**E**val-**P**rint **L**oop for multi-turn conversations, automatically maintaining context. It supports tab autocompletion, multi-line input, history search, and configurable keybindings (Emacs/VI).
3. Shell Assistant	aichat -e "natural language command"	Use natural language to generate, explain, or execute shell commands. aichat is aware of your operating system and shell environment.

II. Key Capabilities & Workflow Integration

A. Code & Development

- **Code Generation (-c):** Generate pure code snippets, scripts, or functions based on your request.
    - *Example:* aichat -c "python script to read a CSV and calculate the average"
- **Refactoring & Debugging:** Provide code via files or stdin to get refactoring suggestions, performance optimization tips, or debugging assistance.

- **Shell Automation:** Use the **Shell Assistant (-e)** mode for tasks like script generation, analyzing server logs, or quickly generating complex Git commands.

B. Input Flexibility (Multi-Modal & Context)

You can provide context to the LLM using multiple input forms, making the AI's response more relevant:

- **Local Files (-f):** Send the content of single or multiple files (includes text, code, and images) to the LLM.
    - *Example:* aichat -f data.txt "Summarize the key findings."
- **Local Directories (-f):** Send the structure and content of an entire folder for high-level analysis or documentation generation.
    - *Example:* aichat -f ./my_project/ "Explain the architecture of this folder."
- **Remote URLs (-f):** Fetch the content of a web page and process it.
- **Piping (stdin):** Pipe output from other terminal commands directly into aichat.
    - *Example:* git diff | aichat "Write a concise commit message for these changes."

Advanced Workflow 1: Contextual, Multi-Step Analysis (using Chat REPL)

This is the **correct** way to run a multi-step task where the second step relies on the first, ensuring the LLM remembers its evaluation.

1. **Start a named, contextual session, passing the directory content (-f) on the first command:**
   aichat -s repo-analysis -f ./my-git-repo/

   # The model will then prompt you to enter the REPL, often with '>>>'

2. **First turn (within the REPL): Evaluate the project and suggest improvements:**
   >>> Evaluate the project structure and suggest improvements for our Git branching and PR review workflow. Be specific.

   # [LLM provides a detailed evaluation and suggestions]

3. **Second turn (within the REPL): Follow-up based on the memory of the evaluation:**
   >>> Based on that evaluation, please now generate a GitHub pull request template and a minimal .gitignore file for a Python project.

   # [LLM uses the context to generate relevant code blocks]

Advanced Workflow 2: Interactive Session (Save and Resume)

This example shows how to use the interactive REPL mode (-s) for a natural conversation that can be saved and re-accessed later.

1. **Start a new session with a name:**
   aichat -s planning-2026
   # The LLM enters REPL mode

2. **Interact (1st day) and close:**
   >>> I need to start planning our 2026 marketing campaign. Focus on social media strategy.

   # [LLM gives first set of steps]

   >>> .quit
   # (Session is automatically saved under the name 'planning-2026'.)

3. **Resume the session (next day):**
   aichat -s planning-2026

   # The LLM loads the full history and enters REPL mode.

   >>> Regarding the social media plan, let's now drill down into platform-specific content for Instagram and TikTok.

   # [LLM continues the plan with full context]

C. Advanced Features (LLMs, RAG, & Agents)

- **Multi-Model Access (-m):** Seamlessly switch between many LLM platforms (OpenAI, Claude, Gemini, Ollama, Groq, etc.) using a unified interface.
- **Custom Roles (-r):** Define persistent "personas" (e.g., a "Senior Python Developer" or a "Grammar Checker") with custom system prompts and configurations to tailor the LLM's behavior for specific tasks.
- **Retrieval-Augmented Generation (RAG):** Integrate external, private documents and knowledge bases into your conversations, grounding the LLM's responses in your specific context. **RAG is a powerful technique that prevents the LLM from relying only on its pre-trained data (which can be out of date) by fetching real-time or private information (from files, databases, etc.) and injecting it directly into the prompt as authoritative context.**

- **AI Tools & Function Calling:** Connect the LLM to external functions and tools, allowing it to perform actions like file operations or web searches as part of its reasoning process.

III. Configuration and Deployment

A. Session and Conversation Management

- **Unlimited Sessions (-s):** Start new, context-aware conversations that maintain history. aichat often uses automatic message compression to handle long discussions.
- **History Commands:** Manage your saved sessions:
  - aichat --list: View all saved conversations.
  - aichat --show [N]: Print the last N messages of a conversation.
  - aichat --delete [ID]: Delete a specific conversation.

B. Custom Configuration (config.yaml)

Your current setup is highly customized via the config.yaml file, which enables the following:

- **Local LLMs via Ollama:** You are configured to connect to a local (or self-hosted) Ollama server (https://ollama.ldmathes.cc/v1).
- **Default Model:** The default model is set to the powerful **gemma3:4b** for general use.
- **Available Models:** You can easily switch between **gemma3:4b** and the lighter **gemma3:1b** model using the .model command in the REPL or the -m flag.

C. Local API Server

- **Serve Mode (--serve):** You can start a lightweight local HTTP server that exposes an **OpenAI-compatible API** for the LLMs configured in aichat. **This means that any existing application, script, or framework (like LangChain) written to use the standard OpenAI API can be seamlessly re-routed to use your local aichat server and the models it hosts (e.g., gemma3:4b), simply by changing the api_base URL.**