LilAI

# Problem Statement

Completed on: 02 June 2025

Design and implement a lightweight, modular python AI assistant framework named LILAI (Low-Intensity Local AI) that is:

* ADHD-friendly and supportive for focus, memory, and planning
* Modular, testable, and extendable
* Designed for offline/local usage (privacy-first)
* Based on structured memory, planning, and response generation

# Objective:

* Support natural conversation and task execution
* Incorporate persistent memory for user context
* Build in a flexible architecture to later add speech or UI components

# Problem Solving Steps / Test

Phase 1: Project Foundation

1. Create project structure:   
    - /assistant  
    -\_init\_.py  
    - core.py  
    - memory.py  
    - planner.py  
    - /src  
    - main.py  
    - /requirements.txt

Phase 2: Build Memory Module (memory.py)

1. Define MemoryChunk using @dataclass  
    - Fields: id (UUID), text, tags, timestamp
2. Implement Memory class with methods:  
    - store (chunk), retrieve (id), search (query)
3. Save to and load from persistent JSON file
4. Use pathlib and datetime for clean structure
5. Optional: add fuzzy search with rapidfuzz
6. Unit tests for storing, retrieving, and searching

Phase 3: Core Assistant Logic (core.py)

1. Define Assistant class with methods:  
    - handle\_input()  
    - query\_memory()  
    - generate\_response() (basic version)
2. Hook memory into assistant workflow
3. Plan out state awareness and soft goals.

Phase 4: Planner Module (planner.py)

1. Build lightweight planner class to:  
    - Break user input into steps  
    - Reference previous actions  
    - Suggest next steps
2. Tie into Assistant.generate\_response() pipeline (early stages)

Phase 5: Entry Point & CLI (main.py)

1. Create simple CLI for testing conversation
2. Add logging and memory inspection tools
3. Add interactive dev/test mode

# Results / Conclusions

Initial version of LILAI successfully demonstrates:

* Clean project structure
* Working memory recall and tagging
* Preliminary assistant behavior using modular functions
* Planning logic scaffolded and tied into core loop

Concepts Demonstrated:

* Python modular design
* Encapsulation and abstraction (separate logic layers)
* Persistent memory using file storage
* Planning with basic NLP parsing
* Early support for user-centric context tracking

# Next Steps

* Expand planner logic with goal/task trees
* Add memory visualization tools (JSON viewer)
* Build optional speech interface with TTS and STT
* Develop ADHD-focused routines (timers, encouragement loops)
* Create testing suite for edge cases
* ~~Add Markdown and PDF export tools for memory reports~~
* ~~Begin UI component scaffolding (optional shell or web app)~~
* Add fuzzy search to improve memory querying
* Write memory index builder and archiving utility