



312 lines (267 loc) · 15.5 KB

Preview

Code

Blame

Raw



# MIRA = Multi-Intent Recognition Assistant

A fully local, voice-first home assistant powered by LLMs: No cloud. No tracking.

It is **very** alpha and currently only suited to DIY enthusiasts.

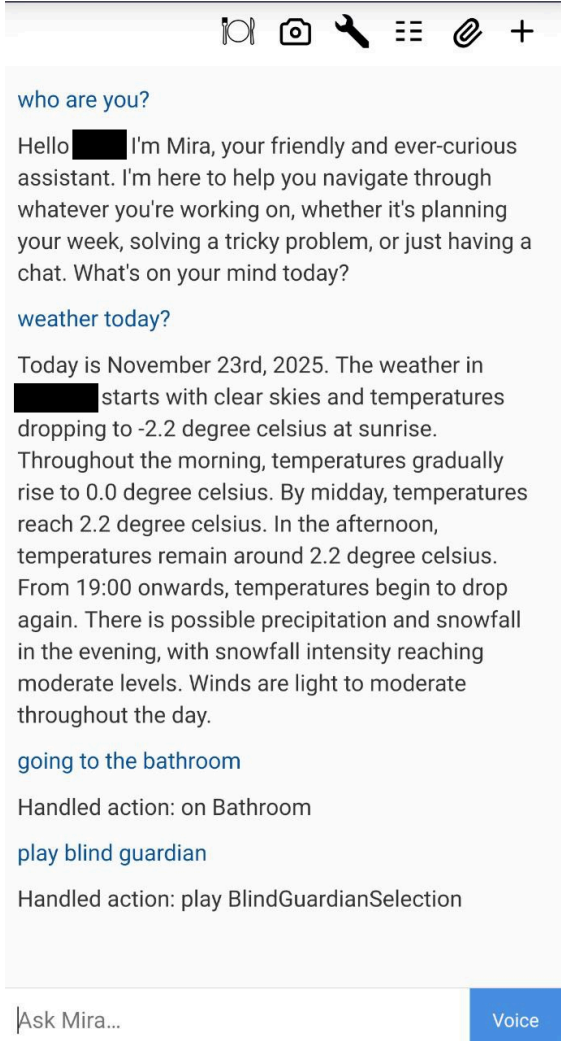
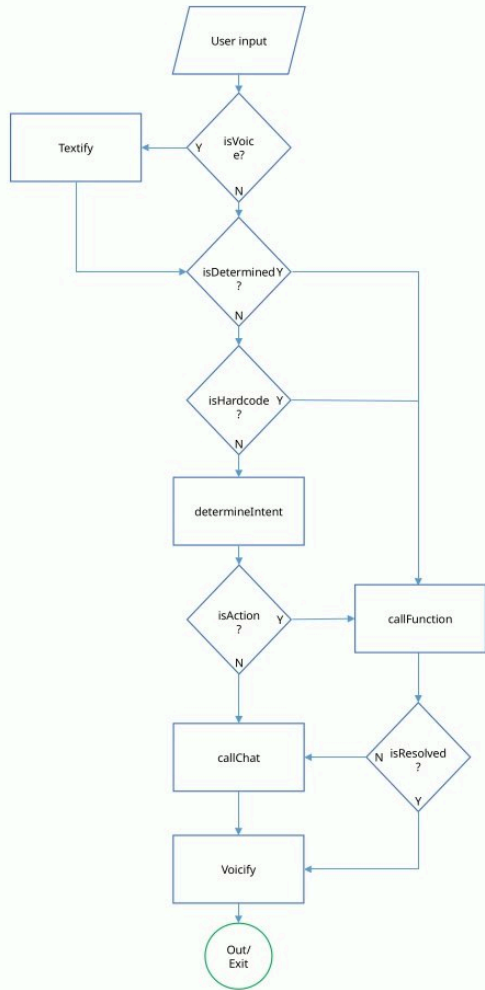
- I apologize in advance to anyone going insane trying to install.
- Check [install.md](#) for a rough guide (I'll help if asked).

## General idea:

- Google, Alexa et al. are not allowed in my home or phone: **DIY**.
- Have a **voice-in, voice-out**, LLM powered home assistant.
  - (Privately) accessible through a cloudflare tunnel from anywhere using a phone or device with a browser.
  - Currently, controls my smart plugs, music, shopping and to-do list, searches Wikipedia, gives weather forecast and more.
  - Has a Chromium/Firefox extension (unsigned for now, requires dev-mode enabled).
- Run the user query through an LLM intent interpreter that catches malformed input and decides what to do.
- Use it for data sensitive tasks.

## Hardware:

Minimum	Recommended	Comment
16GB VRAM	More	16GB will leave about 4GB for context window.

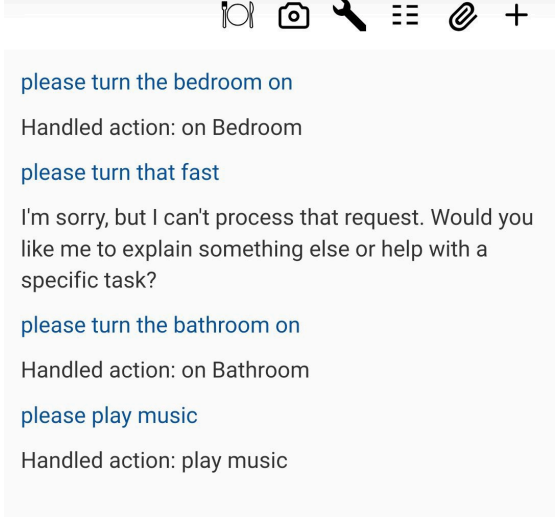
Minimum	Recommended	Comment
50 TFLOPS	Faster	Speed is king in our use case.
32GB RAM	32GB RAM	Doesn't really matter. Vosk is 4GB and held in RAM.
6 core CPU	Faster	Not overly important.
Demo		Flowchart
 <p>who are you?</p> <p>Hello [redacted] I'm Mira, your friendly and ever-curious assistant. I'm here to help you navigate through whatever you're working on, whether it's planning your week, solving a tricky problem, or just having a chat. What's on your mind today?</p> <p>weather today?</p> <p>Today is November 23rd, 2025. The weather in [redacted] starts with clear skies and temperatures dropping to -2.2 degree celsius at sunrise. Throughout the morning, temperatures gradually rise to 0.0 degree celsius. By midday, temperatures reach 2.2 degree celsius. In the afternoon, temperatures remain around 2.2 degree celsius. From 19:00 onwards, temperatures begin to drop again. There is possible precipitation and snowfall in the evening, with snowfall intensity reaching moderate levels. Winds are light to moderate throughout the day.</p> <p>going to the bathroom</p> <p>Handled action: on Bathroom</p> <p>play blind guardian</p> <p>Handled action: play BlindGuardianSelection</p> <p>Ask Mira... Voice</p>		 <pre> graph TD     Start([User input]) --&gt; IsVoice{isVoice?}     IsVoice -- Y --&gt; Textify[Textify]     IsVoice -- N --&gt; IsDetermined{isDetermined?}     Textify --&gt; IsDetermined     IsDetermined -- Y --&gt; IsHardcode{isHardcode?}     IsDetermined -- N --&gt; DetermineIntent[determineIntent]     IsHardcode -- Y --&gt; CallFunction[callFunction]     IsHardcode -- N --&gt; DetermineIntent     DetermineIntent --&gt; IsAction{isAction?}     IsAction -- Y --&gt; CallFunction     IsAction -- N --&gt; CallChat[callChat]     CallFunction --&gt; IsResolved{isResolved?}     IsResolved -- N --&gt; CallChat     IsResolved -- Y --&gt; Voicify[Voicify]     CallChat --&gt; Voicify     Voicify --&gt; End((Out/Exit))   </pre>

## isVoice+Textify

Currently using Vosk, running on CPU.

- Results are mixed, the LLM interpreter catches some but not all errors.
  - Vosk is a very polite listener:
    - Starting a query with "please" will jolt it to attention with the P and familiar word and reduce error rate.
    - Having no accent/being a native speaker helps.
    - Not having a cold helps.

- Pressing the voice button a short while before speaking and holding it a little afterward helps.
- Solution (implemented, testing): Have the user record a wake word (please). Prepend to every voice query.
  - Until frontend implementation: Drop your own "please" into ...mira/static/sounds/please.wav
- Solution (not implemented): Make the button give visual feedback .2 seconds after being pressed and let it record .2 seconds longer.

 <p>please turn the bedroom on Handled action: on Bedroom</p> <p>please turn that fast I'm sorry, but I can't process that request. Would you like me to explain something else or help with a specific task?</p> <p>please turn the bathroom on Handled action: on Bathroom</p> <p>please play music Handled action: play music</p>	<p>First, third and fourth query are correct.</p> <p>Second query input was equal to third. Vosk deviated too far for the LLM intent interpreter.</p>
--	---

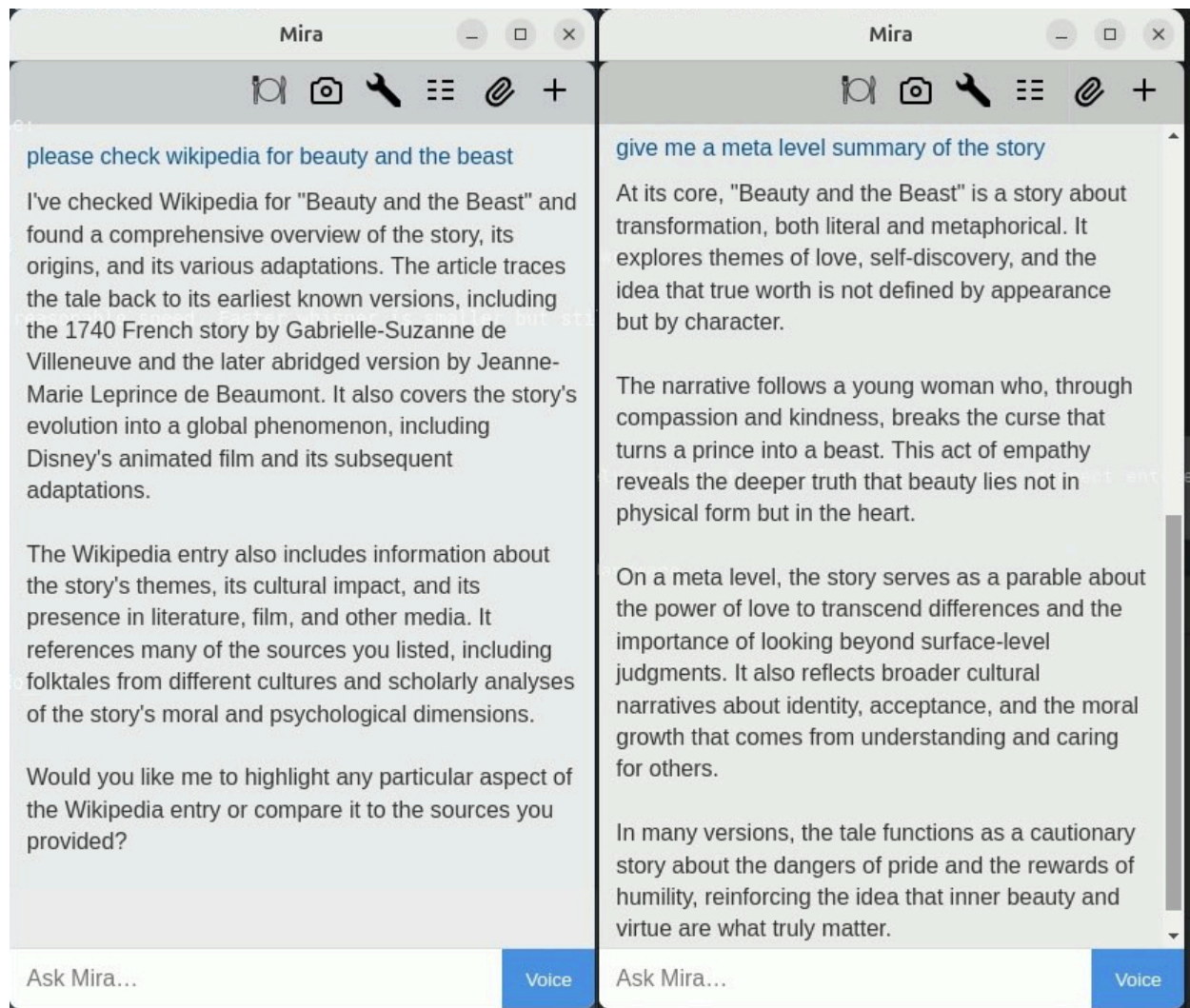
- **Long Term Solution:** Use Nvidia canary which is apparently 4x as accurate; Problem: 5GB model and can't run on cpu with reasonable speed. Faster whisper is smaller but still must be held in VRAM at all times for response time. FW is also not CUDA13 updated.
  - Canary is probably required for proper "real world" application with low error rates.

## IsDetermined

- Already builds a DB of input and matched command.
- **Not implemented:** For test and debug reasons I don't want to match the input to the resolved commands in the DB yet.
  - **Long Term:** Implement and save the inference cost for repeating commands.

## IsHardcode

- Does what it does but is not very extensive.
- **Wikipedia:** Grabs roughly 2k context from the wikipedia entry and answers questions.
  - Works reasonably well for a quick check.
  - The wikipedia keyword must be present.



[BeforeRequest] Authenticated.



[User] ask\_intent user\_msg: give me a meta level summary of the story

[Intent] Generating response...

Llama.generate: 3 prefix-match hit, remaining 281 prompt tokens to eval

```
llama_perf_context_print:      load time =      71.06 ms
llama_perf_context_print: prompt eval time =     53.51 ms /   281
tokens (   0.19 ms per token, 5251.35 tokens per second)
llama_perf_context_print:      eval time =     281.73 ms /    23
runs   (  12.25 ms per token,  81.64 tokens per second)
llama_perf_context_print:      total time =     341.08 ms /   304
tokens
```

```
llama_perf_context_print:    graphs reused =         22
```

[User] give me a meta level summary of the story

[Mira] Generating response...

Llama.generate: 3 prefix-match hit, remaining 3340 prompt tokens to eval

```
llama_perf_context_print:      load time =      71.06 ms
llama_perf_context_print: prompt eval time =     520.63 ms /  3340
tokens (   0.16 ms per token, 6415.32 tokens per second)
llama_perf_context_print:      eval time =     2427.68 ms /   185
runs   (  13.12 ms per token,  76.20 tokens per second)
llama_perf_context_print:      total time =    3003.96 ms /  3525
```

tokens

llama\_perf\_context\_print:      graphs reused =      184

- **Web search:**

- **Dummy pipeline:** Gets results from DuckDuckGo but the converter struggles with the shitload of adds etc.
- Possible Solution: Query/prefer specific user defined sites (get through front end or file).
- Possible Solution: Use the LLM to check individual entries: extract text, score by relevance, possibly prefilter content.
- Then feed aggregate to chat as context.
- web search keyword must be present (currently "search" and "web").
- Solution (not implemented) to remove hardcoded keywords: Build an (optional) confirmation loop: intent:web\_search/wikipedia, get\_confirmation, do\_web\_search

## Determine Intent

- Pretty solid, needs some prompt engineering, possibly better naming schemes for the commands and some tightening of the voice-in error correction.
- Currently, it does a single command per query (debug behaviour).
  - **Long Term:** Wrap the chat+intent route in a for every command loop.
    - Solution: Make the intent LLM capable of emitting multiple commands and adapt the chat route.
      - Needs an additional flask route, a refactor of the chat+intent route and a back and forth between front- and backend.
    - Problem: Harder to debug individual modules, therefore currently inconvenient.

## CallFunction

```
if command == "play music": music_play()
if command in ("pause playback", "stop playback"): music_pause()
if command == "next song": music_next()
if command == "previous song": music_previous()
if command.startswith("play "):
    stem = command.removeprefix("play ").strip()
    if stem in PLAYLIST_STEM: music_load(command)
# chromium
if command == "open Chromium": chromium_start()
if command == "close Chromium": chromium_terminate()
# gui
```



```
if command == "remove attachment":
    HasAttachment.set_attachment(False)
# ShoppingList
if command == "new ShoppingList": new_shopping_list(list_items)
if command == "append ShoppingList":
    append_shopping_list(list_items)
# To-Do List
if command == "new ToDoList": new_to_do_list(list_items)
if command == "append ToDoList": append_to_do_list(list_items)
# Smart plugs
for plug_name in PLUGS:
    if command == f"on {plug_name.capitalize()}":
        turn_on(plug_name)
    elif command == f"off {plug_name.capitalize()}":
        turn_off(plug_name)
```

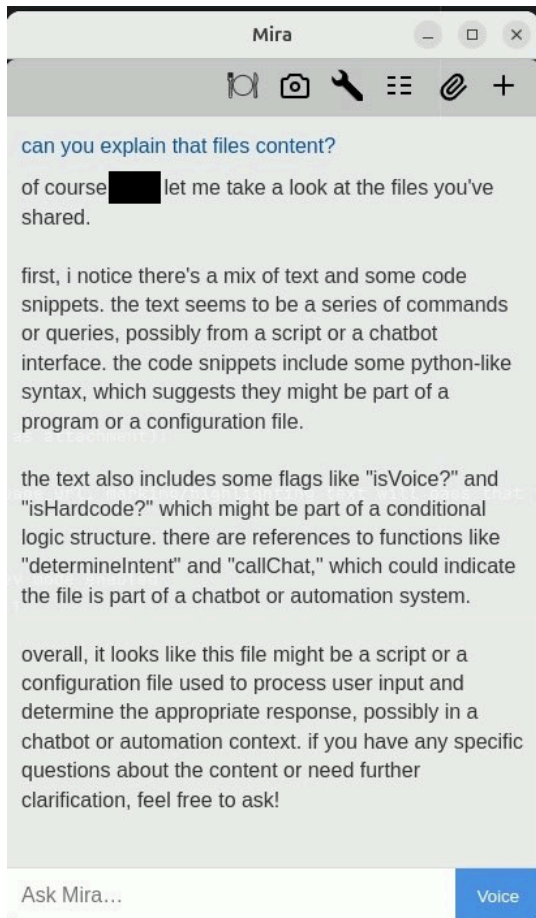
## CallChat

Answers anything that is not a command

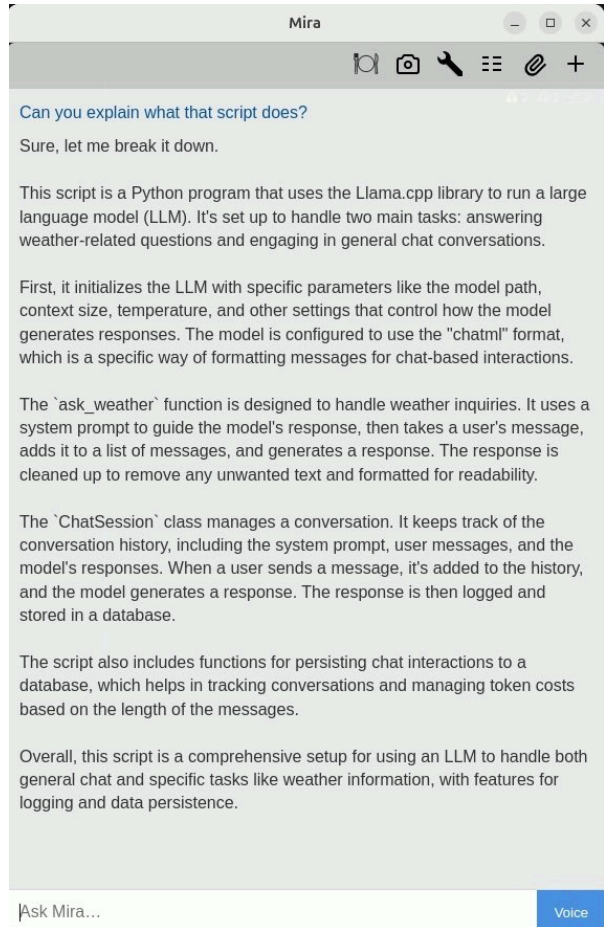
- Reads and discusses files:
  - LibreOffice-supported input formats for conversion to PDF.
  - Will likely not do very well with non-text/complex/long documents (it's a small model).
  - Outright fail if no text, misses context from pictures and likely also from complex tables.
  - Reads: .doc, .docx, .odt, .rtf, .xls, .xlsx, .ods, .csv, .ppt, .pptx, .odp, .html, .pdf, .txt, .py, .js, .ts, .css, .md, .json, .xml, .yaml, .yml, .toml, .sh, .c, .cpp, .java, .rb, .go, .rs



## Flowchart (flowchart.odp to PDF to pdfminer6)



## Python Code (services/llm\_chat.py)



## Voice out:

Currently using xtts-v2.

[output\\_demo.wav](#)

Your browser does not support the audio element. [Download demo](#)

- Results are good. Text needs some preprocessing (ongoing tweak) before given to the model.
- Newer and better models for inference are available, but they need stronger hardware.
  - Xtts-v2 struggles somewhat with punctuation, code, numbers, etc. but does well on "normal" text.
- (For now) Can change voice by replacing .../mira/static/xtts-v2/samples/en-sample.wav
  - Use Spongebob, 7of9 or whatever.
  - I like the xtts-v2 sample, actually.
- Also streams the voice chunks to the frontend (output\_1 is already playing while chunk 4 is being made):

can you explain that files  
a entry or compar  
of course [REDACTED] let me tal  
shared.

Clicking a link will pass the link\_url, empty space on the page will pass the page\_url, marking/highlighting text will pass that text, image needs qwen vl(not implemented).



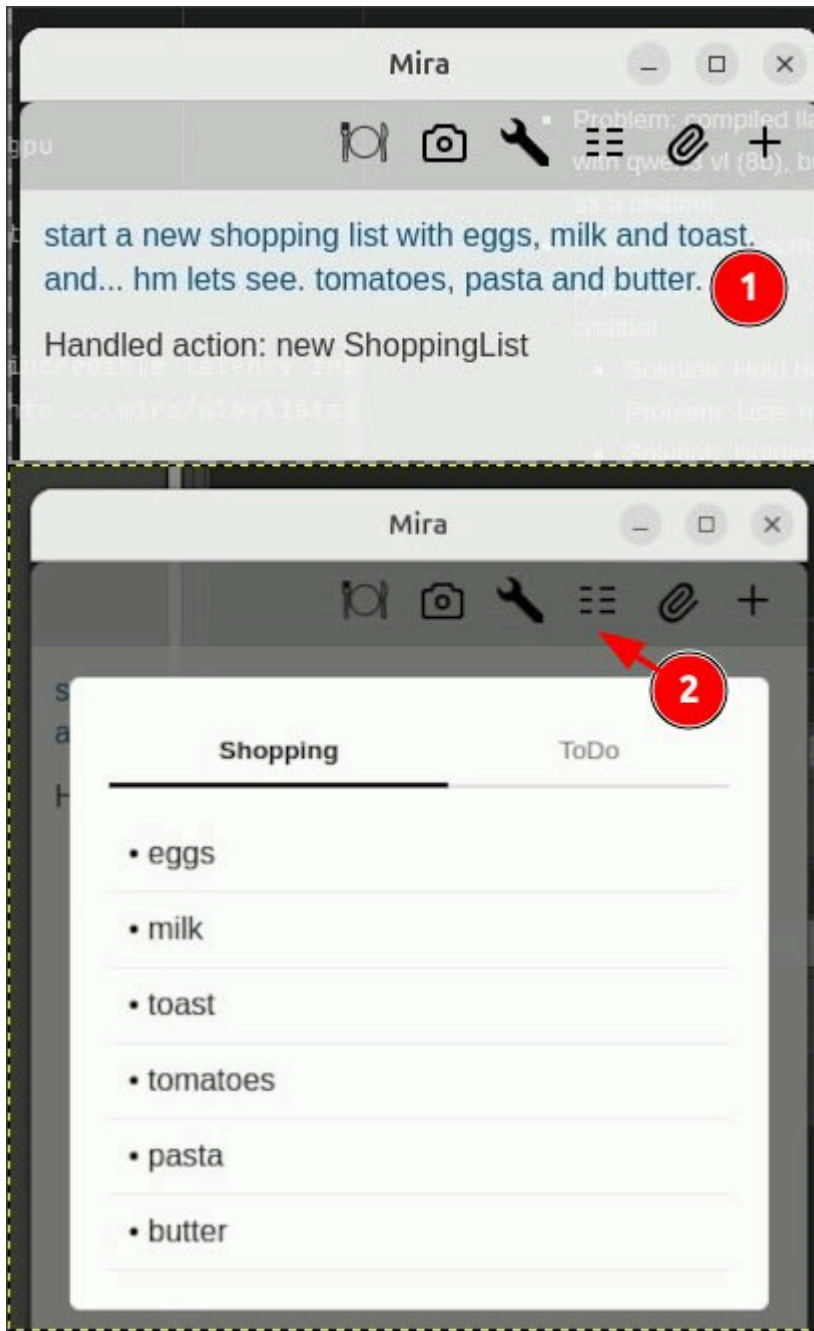


- Install from ...mira/services/browser/chromium\_extension with Chromium's dev mode enabled
- Firefox extension exists and should work but is more annoying to use/install
  - Both extensions need to eventually be verified and available through proper channels.

## Roadmap:

- Have it accessible from any client in the local network and from my smartphone through the web (done)
  - Creates additional Qt6 window on localhost
  - Setting the web API requires much DIY (cloudflare tunnel, buy domain, file edits)
  - Serving to home network with https requires to create a ssl cert with mkcert, otherwise much access to the phone (camera, mic) is restricted.
    - Solution: check cert health: auto generate if close to expiry or not present; Problem fails without mkcert installed
- Give control of Tasmota smart plugs (done, can be entered with name+ip in front end)
  - Commands dynamically generated: on Name; off Name
- Give control of heating units (TO-DO: seems to need integration with home assistant)
- Have it read files (done for many common formats, but not pictures)
  - Images need qwen vl: We currently constantly hold vosk@cpu, assistant(7,5GB)@gpu, text to speech (2,1GB)@gpu
    - Let's assume 2GB VRAM for OS:  $2+2,1+7,5=11,6$ GB: leaves ~4GB VRAM for context
    - Problem: compiled llama-cpp-python to work with qwen3 vl (8b), but it's borderline unusable as a chatbot
    - Problem: tested someone else's llama-cpp-python for qwen3 vl, but it's still unusable as a chatbot
      - Solution: Hold both models in VRAM; Problem: User needs at least 24gb VRAM, I don't have enough VRAM to test
      - Solution: Unload the text model and load the vl model, do inference, reload the text model; Problem: incredible latency increase; likely unfeasible
- Have it control music (done, can dynamically play any playlist that clementine can process). Drop playlist into ...mira/playlists)
  - I still buy music, preferably directly from the artist: Probably no spotify support from me.
- Come up with a way to have it read images (ongoing)

- Use for shopping and to-do list (done)
  - Accepts new shopping/to do list + items and append shopping/to do list + items
  - Delete items from front end (tap) or by starting new list



- Interpret weather data (done, currently can give today, tomorrow, the day after).
  - More context is possible but these small models will start struggling (needs prompt engineering).
  - Queries open-meteo.com API and exposes ip, longitude, latitude
  - Must enter Lat and Lon in frontend for the query to succeed
- make chat sessions client unique and allow multiple users (not implemented)
- make chat context a rolling window (not implemented)
  - Currently fails on max context (start new conversation with + button)

## Understand time and user schedule (done, sort of, needs some more tweaking and calendar access)

- It references fun because sunday is defined as free time in my schedule.



## Read phone calendar (not implemented)

- Most of these calendar apps use sqlite (far as i can tell)
  - That means only 1 write con: Potentially locks the DB if I add write to calendar functions.
  - Can use read only to make periodic copies
  - Add calendar items to the prompt/schedule
  - Add a nightly check to inform of upcoming calendar items ahead of time.

## Integrate with Thunderbird (not implemented)

- Preliminary review suggests it could be relatively simple with a custom extension.
  - Should be able to make it store mail+attachment in file system
  - Should be able to make it write basic mails.
  - Should be able to summarise new and unread mails.

## Take a picture (WIP - not functional)

Really requires more hardware than I currently have.

- Camera access from phone and ability to send a picture to flask (done)
- Find a way for qwen vl addition (ongoing)

## Generate a picture (not implemented)

- Same issue: Needs a vl model.

## Nutrition (WIP - not functional)

The idea is to create a database of

- kcal, carbs, fat, protein,
- daily\_total, daily\_allowed
- which products
- activities
- dietary goal
- age, sex, weight, bmi, etc.

Then query the llm for improvements, have it suggest substitute products, suggest a meal, encourage, whatever.

### What does work?

- The **barcode scanner** (sort of)
  - I've tried a lot of these, and they are all finicky one way or another.
  - Solution1: Use qwen vl; Problem: Hardware
    - Must keep js scanner as optional (improve or use different scanner)
  - Solution2: Possibly send a stream of pictures to the backend and do the calcs on proper hardware myself with python.
- Getting the **product info** from world.openfoodfacts.org (sort of)
  - Problem: Their DB does neither return normalized, nor correct (sample products=8, fully wrong entries=3) results
    - Same field: 38,5g (14 x 2,75g), 1l, 125 g, 140g, 1pcs
  - Solution: Refactor everything to use that DB only as suggestion, aggressively attempt to normalize it, have user correct entries
    - eventually builds local trusted DB
      - needs to be updatable, but refuse an overwrite from openfoodfacts
  - Solution: Use qwen vl, have the user take (up to three?) photos; Problem: Hardware
    - Must keep manual input (vl mode must be optional) in any case
- Wasn't really able to test the front end beyond scanning

### Give it its own shell and user privileges (not implemented)

- Simple enough. Don't have a good use case yet.

### Have it serving files in a NAS like fashion (not implemented)



**Make the installation process so easy that people don't turn insane (not implemented)**