AI Chat Bots

https://jimfawcett.github.io/NewSite/Code/CodeBites_ChatBotAI.html

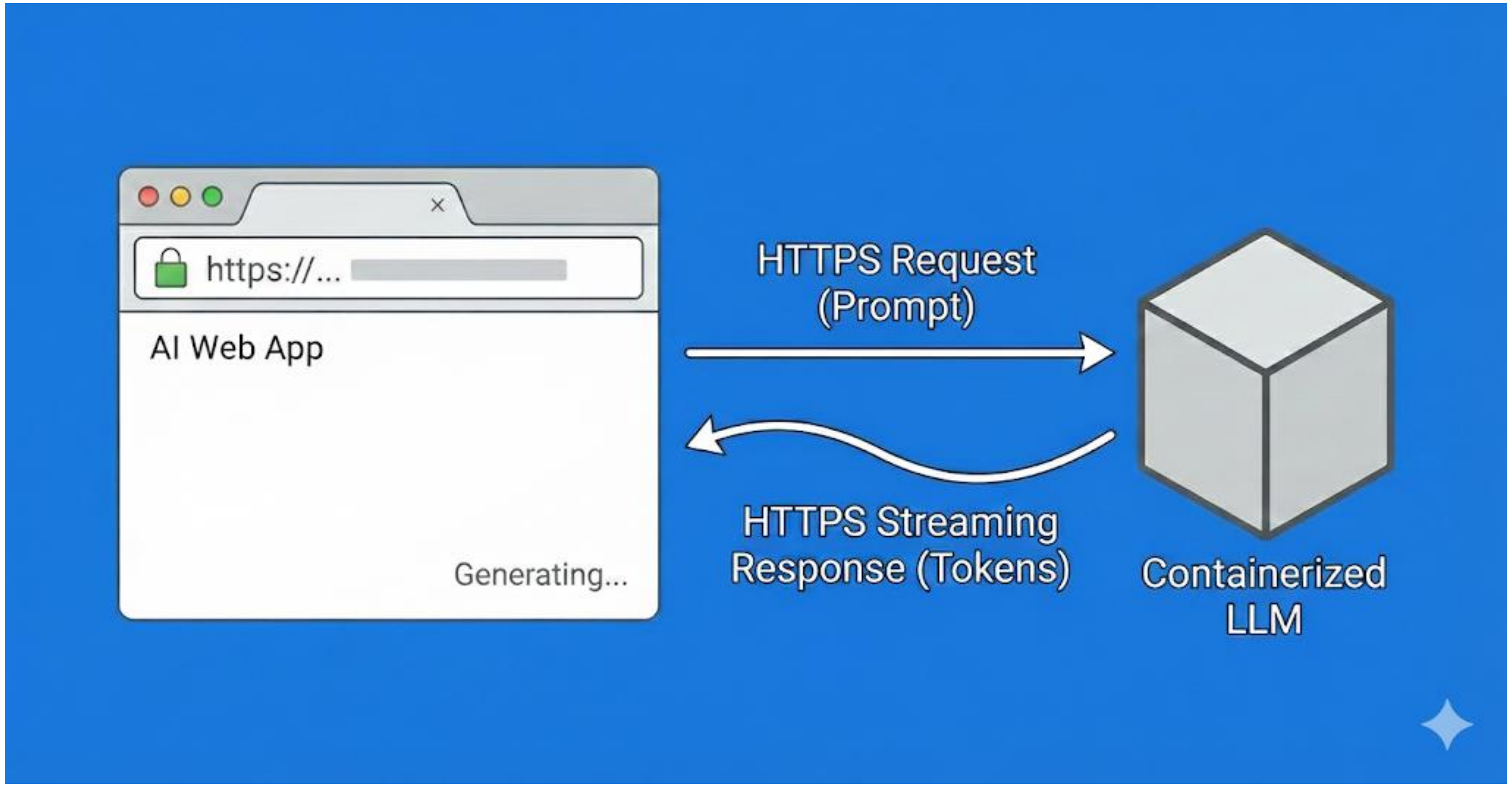
Prologue

- Many AI platforms, e.g., OpenAI, Anthropic, Gemini, … have at least three ways of interacting with an LLM:
  - Web application (topic of this presentation)
    - Can't read from or write to local repositories
    - Can't read code from github repository
    - Can paste a zip with code at the end of a prompt
  - Agent (topic of 2nd presentation)
    - Uses local code application to communicate with LLM
    - Can read from and write to local repositories
  - Console (topic of 3rd presentation)
    - Like agent but you don't have to write agent code
    - You simply use the console

Introduction

- Chat Bots are web applications consisting of:
  - **Browser-based interface** that accepts prompts and displays results
  - A containerized pre-trained **Large Language Model (LLM)** running in a Linux server in a data center
  - **Https-based communication infrastructure** to ferry prompts to, and results from the LLM

AI Web App Structure



Limitations

- AI Chat Bots have limitations:
  - Cannot read from or write to local repositories because they are browser based
  - Cannot read files from github repository

Work Arounds

- You can paste code text at the end of a prompt
  - Prompt specifies what to do with the following code
- You can create a zip for a local repository or download a github zip
  - Paste zip file at the end of a prompt
- You can request Chat Bot to download its results to your local download directory

Example

– Size and Complexity of code functions

- This prompt:

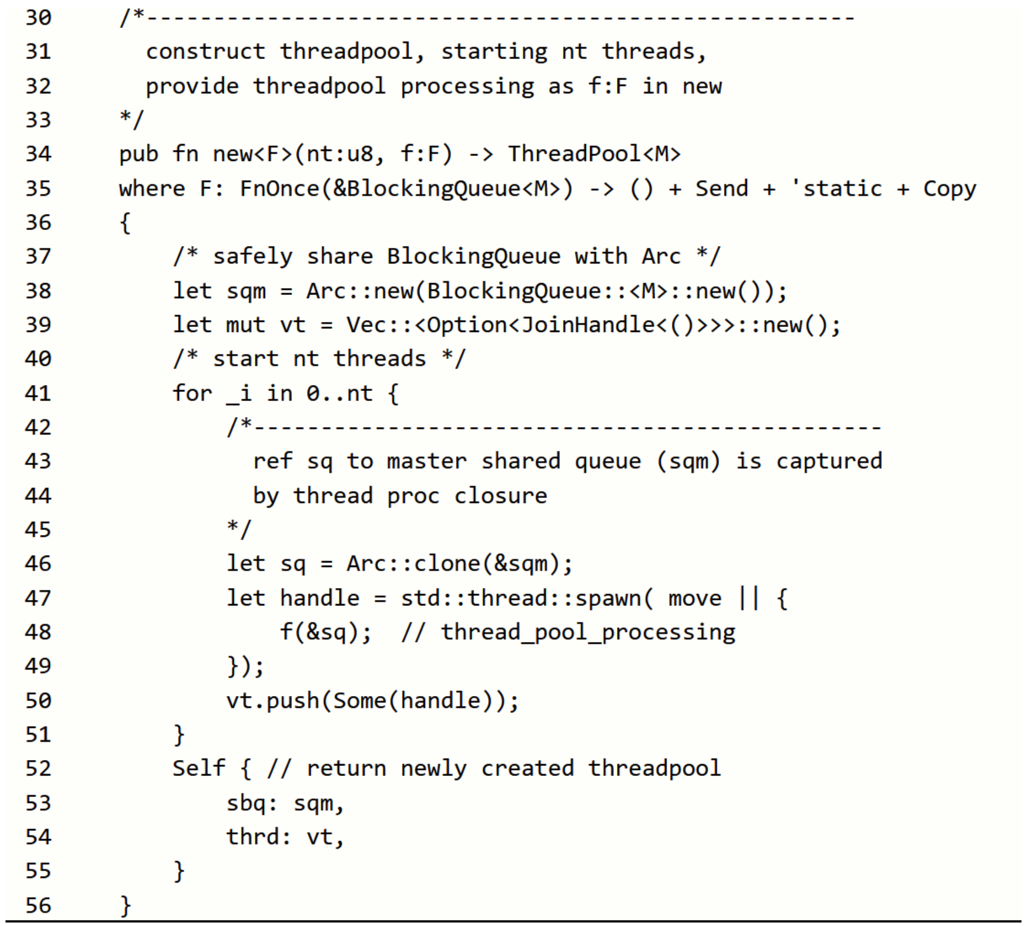
  “Generate a list of function line counts and complexities for all functions in the zip file pasted at end of this prompt.

    Line count is total number of lines including code, whitepace, comments

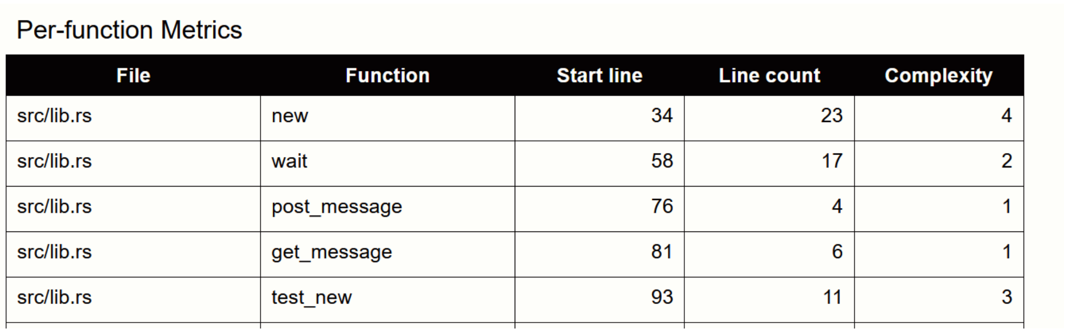
    Complexity is the number of open braces in each function”

- Generated a list of all the functions with correct line counts and complexities for a Rust Thread Pool.
  - Note: Rust code is easier to parse than C++ and doesn’t have function overloads.

Partial Code and Analysis Results



```
30    /*----------------------------------------------------
31      construct threadpool, starting nt threads,
32      provide threadpool processing as f:F in new
33    */
34    pub fn new<F>(nt:u8, f:F) -> ThreadPool<M>
35    where F: FnOnce(&BlockingQueue<M>) -> () + Send + 'static + Copy
36    {
37        /* safely share BlockingQueue with Arc */
38        let sqm = Arc::new(BlockingQueue::<M>::new());
39        let mut vt = Vec::<Option<JoinHandle<()>>>::new();
40        /* start nt threads */
41        for _i in 0..nt {
42            /*-----------------------------------------------
43              ref sq to master shared queue (sqm) is captured
44              by thread proc closure
45            */
46            let sq = Arc::clone(&sqm);
47            let handle = std::thread::spawn( move || {
48                f(&sq);  // thread_pool_processing
49            });
50            vt.push(Some(handle));
51        }
52        Self { // return newly created threadpool
53            sbq: sqm,
54            thrd: vt,
55        }
56    }
```



Per-function Metrics

File	Function	Start line	Line count	Complexity
src/lib.rs	new	34	23	4
src/lib.rs	wait	58	17	2
src/lib.rs	post_message	76	4	1
src/lib.rs	get_message	81	6	1
src/lib.rs	test_new	93	11	3

Conclusion

- You can find all the details here:

  https://jimfawcett.github.io/NewSite/Code/CodeBites_ChatBotAI.html
- The next presentation explores:
  - AI Agents
  - Use local application to communicate with the LLM
  - Uses HTTPS-based API
  - Can read from and write to local file system
- The one after that explores:
  - AI Consoles
  - Have access to local repositories
  - Don’t need to write Agent code