AI for Software Dev
Video #3 – AI Consoles

Jim Fawcett & Mike Corley

AI Consoles

- AI Consoles are specialized terminal applications that:
  - Have a large set of commands. Partial list of Gemini commands: /about, /copy, /directory, /help, /rewind, /init, /mcp, /tools, /skills, …
  - Send all text inputs to remote LLM via web messaging.
    - One exception: you can execute a shell command by prepending "!"
  - Can read from and write to the local file system
- They act like agents without you providing code for the application
  - Likely to be slower and more expensive than agents
    - They don't build in local operations
- May use a platform key to access remote LLM

Advantages

- AI Consoles can read from and write to files in the local filesystem
- You don't have to write any code for the application
  - Benefit:
    - More flexible, not limited by constraints built into agent code
    - Immediately productive
  - Liability:
    - Less productive than an agent for fixed repetitive tasks

Example – Create Basic Projects

- Create Rust "Hello World" project
  - Create Rust crate structure
  - Define, build, and execute Hello Word application
  - Runs Rust code analyzer: cargo clippy
- Create C++ "Hello World" project
  - Create C++ project structure
  - Define CMake file for building
  - Define, build, and execute C++ Hello World application

Example – Prompt and Partial Response


```
> create Hello World project in Rust but do not run

✦ Okay, I'll create a "Hello World" Rust project for you.

  First, I'll use cargo new to create the project:

  ┌─────────────────────────────────────────────────────────────────────────────────────────────┐
  │ ✓  Shell cargo new hello_world_rust [current working directory C:\github\Jimfawcett\NewSite\Code\ai\DemoAIConsole-Gemini] (Crea… │
  │                                                                                               │
  │     Creating binary (application) `hello_world_rust` package                                  │
  │ note: see more `Cargo.toml` keys and their definitions at https://doc.rust-lang.org/cargo/reference/manifest.html │
  └─────────────────────────────────────────────────────────────────────────────────────────────┘
✦ The Rust project "hello_world_rust" has been created. I'll now confirm the "Hello World" code by reading src/main.rs within that
  directory.

  ┌─────────────────────────────────────────────────────────────────────────────────────────────┐
  │ ✓  ReadFile hello_world_rust\src\main.rs                                                      │
  │                                                                                               │
  └─────────────────────────────────────────────────────────────────────────────────────────────┘
✦ I've created the Rust "Hello World" project and verified src/main.rs content, as requested, without running it. Output.txt remains
  untouched. Request complete.

> run

✦ "Run" likely means executing the Rust "Hello World" project. I'll cd hello_world_rust then cargo run.

  ┌─────────────────────────────────────────────────────────────────────────────────────────────┐
  │ ✓  Shell cargo run [in hello_world_rust] (Running the 'hello_world_rust' project.)            │
  │                                                                                               │
  │     Compiling hello_world_rust v0.1.0 (C:\github\Jimfawcett\NewSite\Code\ai\DemoAIConsole-Gemini\hello_world_rust) │
  │      Finished `dev` profile [unoptimized + debuginfo] target(s) in 0.58s                      │
  │       Running `target\debug\hello_world_rust.exe`                                             │
  │ Hello, world!                                                                                 │
  └─────────────────────────────────────────────────────────────────────────────────────────────┘
✦ "Hello, world!" output confirms successful execution of the Rust project. Request complete.
```

Conclusion

- This third presentation focuses on AI Command Line Interface (CLI)
  - You can find all the details here:
    https://jimfawcett.github.io/NewSite/Code/CodeBites_ConsoleAI.html

- The first presentation explores:
  - AI Chat Bots
  - Use web application defined by platform to communicate with remote LLM
  - Cannot read or write local files

- The second presentation explores:
  - AI Agents
  - Use local application to communicate with remote LLM
  - Uses HTTPS-based API
  - Can read from and write to local file system

- The fourth presentation explores AI Skills
  - Available only on Anthropic platform
  - Can simulate with OpenAI and Gemini