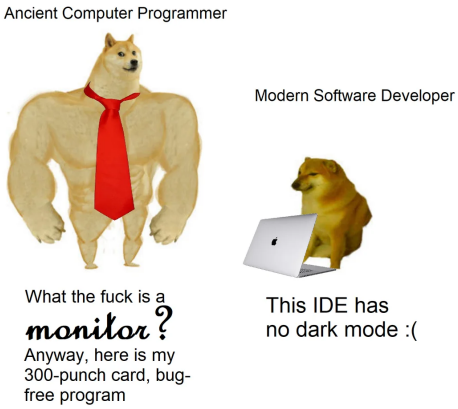
Tech16 Final Project

- author : david howard
- linkedin : https://www.linkedin.com/in/david-howard-95482a1/
- github : https://github.com/dmh2000/tech16-cli



Because I am an old school terminal guy, I decided to cobble some command line AI tools for use in shell scripts. I initially started on a do-everything cli tool (see src/tech16-cli) but I pivoted to add a couple of special purpose cli tools that do one thing only.

P.S. I did not request a grade. The last time I was in university was 1981 so I don't need the credits.

The Tools

- **src/tech-planner**
  - a python cli tool to generate plans from a command line or script
  - has a system prompt that configures it as a planning assistant.
  - it will try to plan just about anything, such as a code project or a trip.
  - **see examples/planner**
- **src/tech-coder**
  - a python cli tool to generate code from command line or script
  - has a system prompt that configures it as a coding assistant.
  - it will try to generate code based on any prompt you give it
  - **see examples/coder**
- src/tech-cli : tries to cover all bases. not fully tested.

The intent is providing cli tools that could be combined in a shell script to create an 'agent'.

To run the tools,

- run "pip -r requirements.txt" to get the dependencies
- *src/tech16-coder/tech16-coder*
- *src/tech16-planner/tech16-planner*

- **To get the best view of how it all works, look in the shell scripts in 'examples' and the output of the examples**:
    - examples/coder/mlb (mlb project script)
        - /mlb/mlb (output of mlb script)
    - examples/planner/
        - trip.sh
        - trip.md (prompt for a sight seeing trip)
            - output/trip-plan-1.md and output/trip-plan-2.md (two different runs)
        - space.sh (prompt for a tour of the solar system)
            - output/space-plan-1.md and output/space-plan-2.md
- The cli programs support multiple LLM providers and models. You can get a list if you run one of them with no command line arguments. Each provider requires the associated API key in the environment:
    - ANTHROPIC_API_KEY for Anthropic models
    - OPENAI_API_KEY for OpenAI models
    - GOOGLE_API_KEY for Gemini models

Claude

I generated most of the code with Anthropic Claude Code using sonnet-4 and haiku. The tool I built support models from Anthropic, Gemini and OpenAI.

tech16-coder

tech16-coder is reasonably easy to use. You specify a model (required) and a file containing a prompt of what you want, plus (optional) files and urls as context, and it will attempt to write the requestd code. This app will write the file or files it generates based on the paths and filenames the LLM called for. It's not a smart agent like Claude, Cline, Cursor etc, so right now its best for simple requests, like "write me a single file python program that does X'.

Anomalies I ran into with code generation

- I ran into some issues with max output token limit too small in some of my code generation. I fixed that for this project but I would probably go back and implement the LLM API to use LangChain because it supports chunking output to avoid the output limits.
- I had to add "Do not output any explanations, descriptions or other non-code text. output the code files only." to the system prompt or the LLMs would create a lot of extraneous output.
- I found that none of the LLMs would generate python web servers that included the proper CORS support unless I explicitly told them to. Without that pages with APIs wouldn't work.
- I had to massage the coder system prompt to get it to create code files with the proper filename annotation. They were very stubborn.

System Prompt

Uses a system prompt for a coder. This system prompt was created in steps:

- asked Perplexity what a system prompt for a coder assistant should have
- had anthropic claude generate the system prompt, following the recommendations from Perplexity

- hard coded this prompt in the application
- tweaked it until it gave consistent results for all models

Implementation

98% of the code in tech16-coder is generated using Claude Code and the claude-4-sonnet model. I didn't use Opus because its really expensive. The code was created step by step:

1. I wrote a rough description of what I wanted this app to do.
2. Gave the description to Claude and asked it to create a comprehensive implementation plan.
3. Gave that implementation plan to Claude and told it to implement the app.
4. I tweaked some of the code by hand because sometimes its easier to do that than come up with a specific enought prompt.

Examples

Shell scripts that run the coder cli. Feel free to run the scripts again. They will overwrite the results with new ones. The 'mlb' app is the most ambituous. It is composed of primitive 'agents' running sequentially to build the application.

```
exmples/coder
|   ├── coder-hello-prompt.md
|   ├── coder-hello.sh
|   ├── coder.sh
|   ├── **mlb (web site with live updates for major league baseball
games)**
|   |   ├── api-plan.md
|   |   ├── coder-mlb-api.sh (codes API server for processed game data)
|   |   ├── coder-mlb-csv.sh (scrapes the mlb website for game data)
|   |   ├── coder-mlb-prompt.md
|   |   ├── **coder-mlb.sh (run this to build the 3 components in the
proper order)
|   |   ├── coder-mlb-web.sh (codes web server for home page)
|   |   ├── **mlb (code generated from the shell scripts)**
|   |   |   | To see the results, run api.py in one terminal and server.py
in another
|   |   |   | then open a browser to http://localhost:8000
|   |   |   ├── api.py
|   |   |   ├── index.css
|   |   |   ├── index.html
|   |   |   ├── index.js
|   |   |   ├── mlb.csv
|   |   |   └── server.py
|   |   ├── mlb-csv.log
|   |   └── tech16.log
|   └── **web (simple web app and server\*\*)
|   | ├── coder-web-prompt.md
|   | └── coder-web.sh
+──
```

Usage

```
ech16-coder - Code generation assistant

USAGE:
  tech16-coder --model MODEL_NAME [FILES_AND_URLS...]

ARGUMENTS:
  --model MODEL_NAME   Model to use (required, must be first argument)
  FILES_AND_URLS       Any number of files and URLs to analyze

EXAMPLES:
  tech16-coder --model claude-sonnet-4 requirements.md
  tech16-coder --model o4-mini file1.txt file2.py https://example.com/docs
  tech16-coder --model gemini-2.5-pro spec.txt https://docs.api.com

SUPPORTED PROVIDERS AND MODELS:

  ANTHROPIC:
    - claude-3-5-haiku-20241022
    - claude-sonnet-4-20250514

  GEMINI:
    - gemini-2.5-flash
    - gemini-2.5-pro

  OPENAI:
    - o3-mini
    - o4-mini

Total models available: 6

NOTE: Requires appropriate API keys set as environment variables:
  - ANTHROPIC_API_KEY for Anthropic models
  - OPENAI_API_KEY for OpenAI models
  - GOOGLE_API_KEY for Gemini models
```

Example Shell Script

```
#!/bin/sh

# create two temporary files
MLB=$(mktemp)
trap 'rm -f "$MLB"' EXIT

# ------------------------------------
# scrape and process the mlb web page
# one-shot with example
# ------------------------------------
echo "<prompt>\
```

```
the input data is today's major league baseball games.\
-from that data create a file "mlb/mlb.csv" that contains: \
  -the visitor team abbreviation, \
  -the home team abbreviation, \
  -the current visitor score or 0 if not playing yet \
  -the current home team score or 0 if not playing yet \
  -an indicator of either: \
      -game time if not started \
      -current inning if in progress \
      -"final" if game is over \
\
Here is an example of the output file, not real data \
\
visitor,home,visitor_score,home_score,status \
TOR,BAL,0,0,6:35 PM ET \
COL,CLE,0,0,7 \
AZ,DET,0,0,final \
\
write the file to 'mlb/mlb.csv \
</prompt> \
\
do not output any description or examplation. output only the mlb/mlb.csv
file. \
" >$MLB

# generate the csv file from the prompt, mlb web site and LLM
../../../src/tech16-coder/tech16-coder --model o4-mini
https://mlb.com/schedule $MLB >mlb-csv.log
```

tech16-planner

tech16-planner is an assistant that will generate a plan for just about anything you ask for that could be planned. Again, pretty easy to use.

System Prompt

Uses a system prompt for a coder. This system prompt was created in steps:

- asked Perplexity what a system prompt for a planner assistant should have
- had anthropic claude generate the system prompt, following the recommendations from Perplexity
- hard coded this prompt in the application

Implementation

98% of the code in tech16-coder is generated using Claude Code and the claude-4-sonnet model. I didn't use Opus because its really expensive. The code was created step by step:

1. I wrote a rough description of what I wanted this app to do.
2. Gave the description to Claude and asked it to create a comprehensive implementation plan.

3. Gave that implementation plan to Claude and told it to implement the app.
4. I tweaked some of the code by hand because sometimes its easier to do that than come up with a specific enought prompt.

Examples

Shell scripts that execute some runs of the planner cli.

```
examples/planner
├── all.sh
├── hello.md (prompt)
├── hello.sh
├── output (results from runs of the shell scripts)
│   ├── hello-plan-1.md
│   ├── hello-plan-2.md
│   ├── space-plan-1.md
│   ├── space-plan-2.md
│   ├── trip-plan-1.md
│   └── trip-plan-2.md
├── space.md (prompt)
├── space.sh
├── trip.md (prompt)
└── trip.sh
```

Usage

```
tech16-planner - General purpose planning assistant CLI tool

USAGE:
  tech16-planner --model MODEL_NAME [FILES_AND_URLS...]

ARGUMENTS:
  --model MODEL_NAME   Model to use (required, must be first argument)
  FILES_AND_URLS       Any number of files and URLs to analyze

EXAMPLES:
  tech16-planner --model claude-sonnet-4 project-docs.md
  tech16-planner --model o4-mini file1.txt file2.py
https://example.com/docs
  tech16-planner --model gemini-2.5-pro requirements.txt
https://docs.api.com

SUPPORTED PROVIDERS AND MODELS:

  ANTHROPIC:
    - claude-3-5-haiku-20241022
    - claude-sonnet-4-20250514

  GEMINI:
```

```
    - gemini-2.5-flash
    - gemini-2.5-pro

  OPENAI:
    - o3-mini
    - o4-mini

Total models available: 6

NOTE: Requires appropriate API keys set as environment variables:
  - ANTHROPIC_API_KEY for Anthropic models
  - OPENAI_API_KEY for OpenAI models
  - GOOGLE_API_KEY for Gemini models
```

Example Prompt and Shell Script

```
I want to generate a plan for a trip to visit interesting sights in our
solar system.

- This is science fiction, so you can create any technologies you need, but
try to keep them at least marginally feasible. No teleportation.
- Include stops at interesting features at each of the major planets of the
solar system.
- Assume the mode of travel has a speed of 1% of light speed
- If you need any additional information before creating the plan, let me
know and I
  will modify this prompt.
```

```
#!/bin/sh

../../src/tech16-planner/tech16-planner --model o4-mini space.md
>output/space-plan.md
```

tech16-cli

This tool is a more general purpose tool that does not have a specific system prompt. It has an optional command line argument to specify a system prompt.

Its a work in progress so you can try it if you like but no guarantees.

Usage

tech16-cli - Query LLMs with custom prompts and context

USAGE: tech16-cli [OPTIONS] [FILES_AND_URLS...]

OPTIONS: --prompt FILENAME File containing the prompt to use

--model MODEL_NAME Model to use (default: o4-mini) --help Show this help message

EXAMPLES: tech16-cli file.txt tech16-cli --model claude-sonnet-4 file.txt tech16-cli --prompt system.txt --model o4-mini https://example.com/docs echo "analyze this" | tech16-cli --prompt review.txt tech16-cli --prompt plan.txt file1.py file2.py https://docs.example.com

SUPPORTED PROVIDERS AND MODELS:

ANTHROPIC: - claude-3-5-haiku-20241022 - claude-sonnet-4-20250514

GEMINI: - gemini-2.5-flash - gemini-2.5-pro

OPENAI: - gpt-4o-mini - o3-mini - o4-mini (default)

Total models available: 7

NOTE: Requires appropriate API keys set as environment variables:

- ANTHROPIC_API_KEY for Anthropic models
- OPENAI_API_KEY for OpenAI models
- GOOGLE_API_KEY for Gemini models