

# 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 decided to add a couple of special purpose cli tools that do one thing only:

- **src/tech-planner**
  - 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**
  - 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 example/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'.

Ancient Computer Programmer



What the fuck is a  
**monitor?**  
Anyway, here is my  
300-punch card, bug-  
free program

Modern Software Developer



This IDE has  
no dark mode :(

- **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

## 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 requested 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 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 enough 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 ambitious. It is composed of primitive 'agents' running sequentially to build the application.

```
examples/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

## 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 enough 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

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 not fully tested so you can try it if you like.

## 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