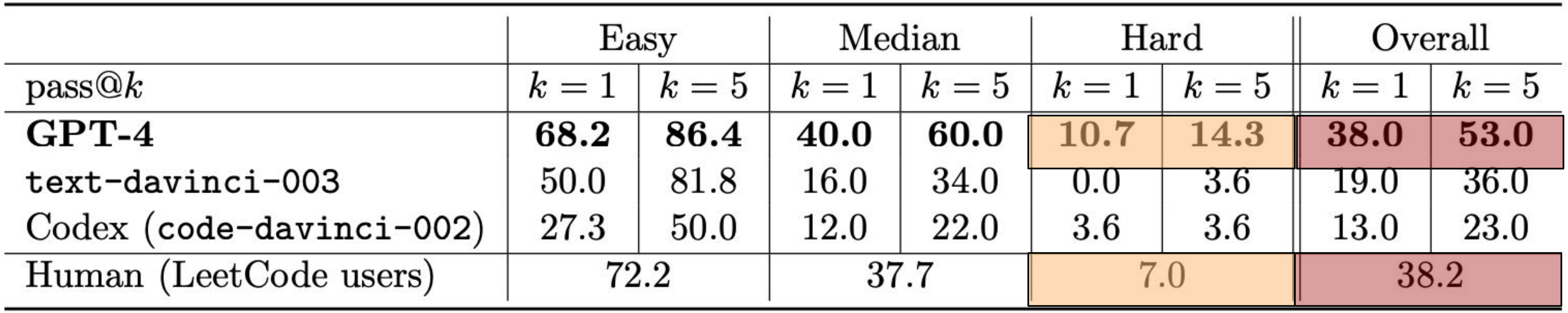
Language Models in Programming

Schäffer Krisztián

aijunior.dev
github.com/tisztamo
linkedin.com/in/tisztamo/

LLMs today: How good are they?



	Easy		Median		Hard		Overall	
pass@k	$k=1$	$k=5$	$k=1$	$k=5$	$k=1$	$k=5$	$k=1$	$k=5$
GPT-4	**68.2**	**86.4**	**40.0**	**60.0**	**10.7**	**14.3**	**38.0**	**53.0**
`text-davinci-003`	50.0	81.8	16.0	34.0	0.0	3.6	19.0	36.0
Codex (`code-davinci-002`)	27.3	50.0	12.0	22.0	3.6	3.6	13.0	23.0
Human (LeetCode users)	72.2		37.7		7.0		38.2	

Table 2: Zero-shot pass@1 and pass@5 accuracies (%) on LeetCode.

Single prompt at its max

Sparks of Artificial General Intelligence: Early experiments with GPT-4

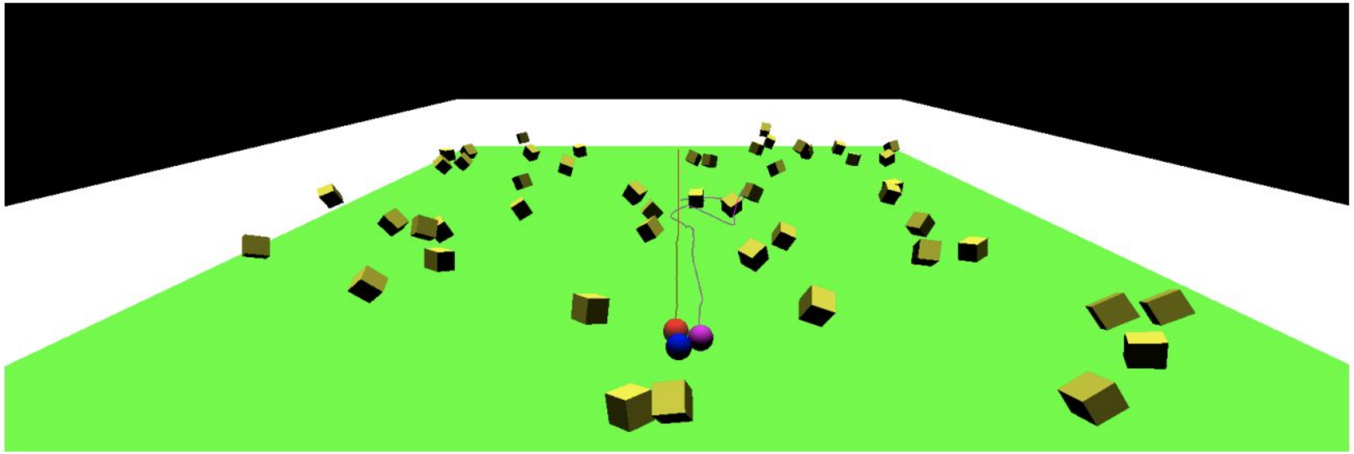
Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, Harsha Nori, Hamid Palangi, Marco Tulio Ribeiro, Yi Zhang

March 2023

GPT-4

Prompt:

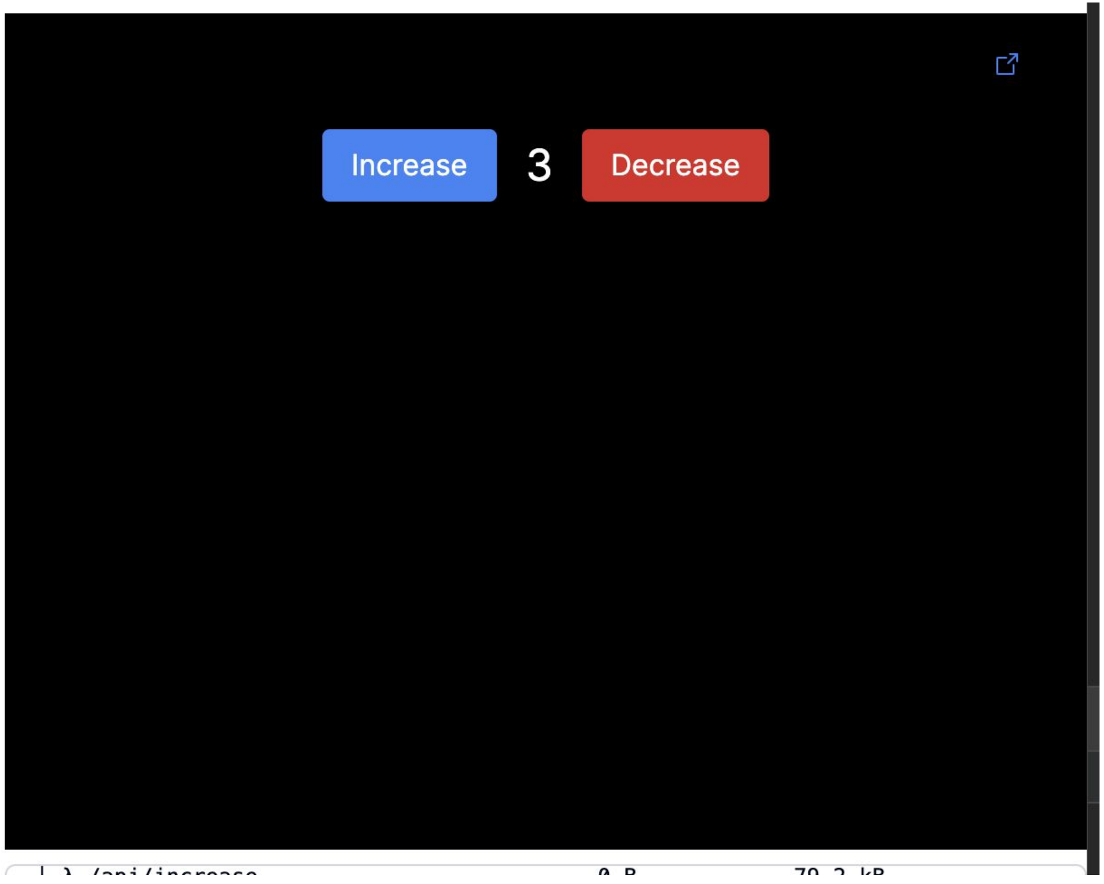
```
Can you write a 3D game in HTML with Javascript, I want:
-There are three avatars, each is a sphere.
-The player controls its avatar using arrow keys to move.
-The enemy avatar is trying to catch the player.
-The defender avatar is trying to block the enemy.
-There are also random obstacles as cubes spawned randomly at the beginning and moving randomly. The avatars cannot cross those cubes.
-The player moves on a 2D plane surrounded by walls that he cannot cross. The wall should cover the boundary of the entire plane.
-Add physics to the environment using cannon.
-If the enemy catches the player, the game is over.
-Plot the trajectories of all the three avatars.
```



Shiny, but...

https://codestub.ai/stub/stub-ai/d406b100-d3b0-51b8-b72f-ae6a8dd795ca

Create a counter component that has a button that increases a counter every time its pressed. Style it well with Tailwind using effects like hover. Add this component to the index page. Add a backend API endpoint that is called every time the counter is updated. Connect the Counter component with this API. Also initialize the counter value with an API that returns the current value of the counter in the backend. Add a Decrease button. Add separate API routes and handlers for Increase and Decrease which are called upon pressing the corresponding buttons.



```
| λ /api/increase                0 B            70.2 kB
```

Working Memory of Large Language Models Simplified

LLM context window is the

LLM context window is the model's

LLM context window is the model's view

context window is the model's view of

window is the model's view of our

is the model's view of our conversation

universe

collective consciousnes

health

Working Memory - limitations

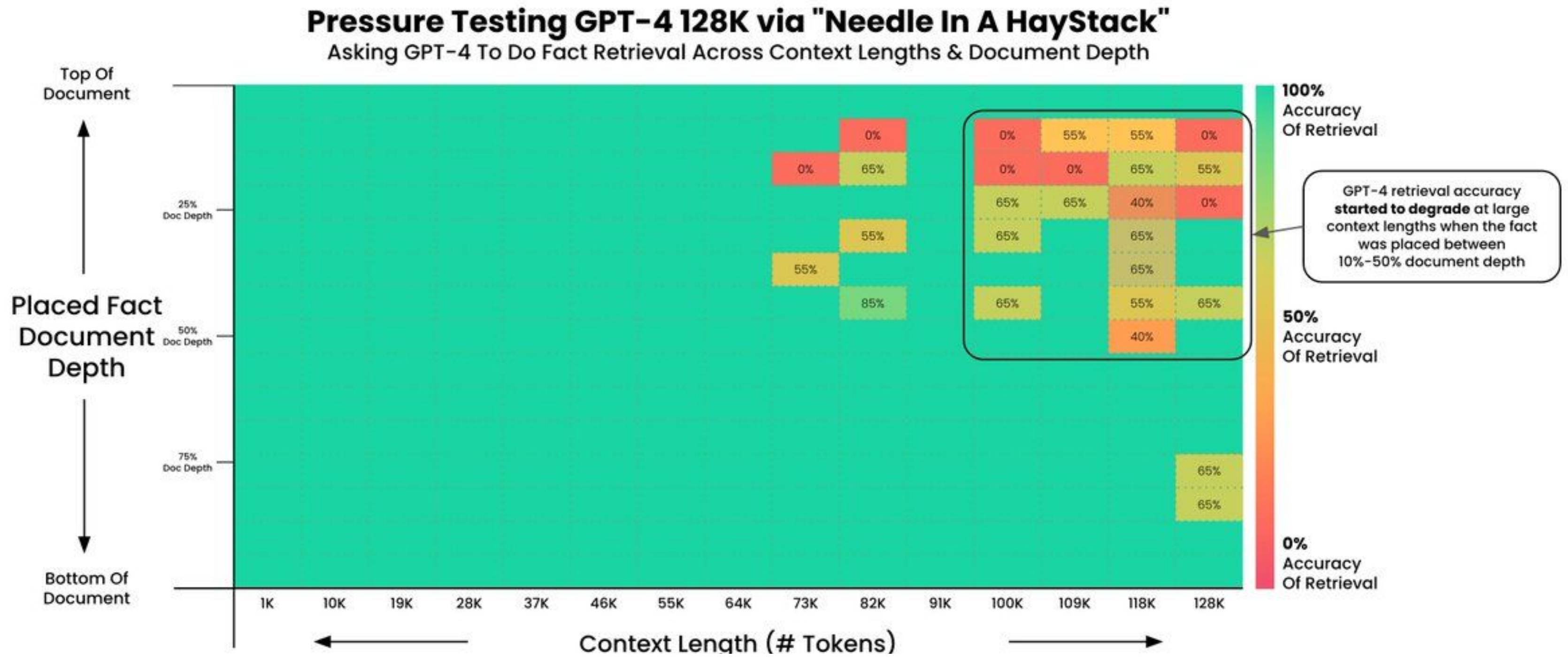
Measured in tokens (a common word, a syllable or a character)

4K - 32K (GPT-4) - 128K (GPT-4 turbo)

Need to fit all of:

- Sources needed to solve the task
- Task description
- Output format description
- Project-specific / non-functional requirements

GPT-4 128K information retrieval accuracy from the prompt



source: https://twitter.com/GregKamradt/status/1722386725635580292

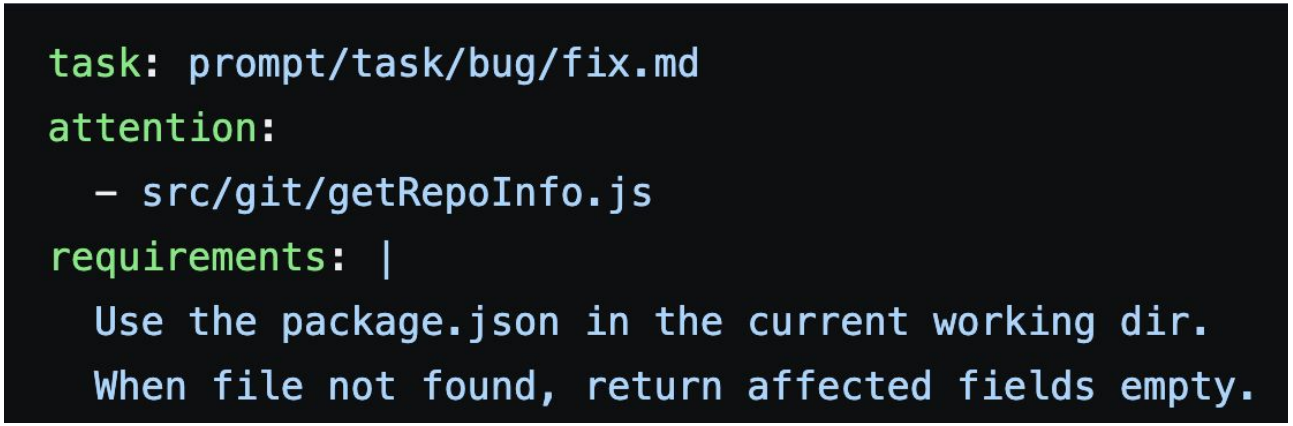
Overcoming Working Memory Limitations in Code Generation

1. Repo Map
    - Only provide headers / names from the full source
    - Partial solution with possibly large footprint
    - Files to be edited or understood must be still shown
2. Vector Database
    - Embed (chunks of) source files
    - Retrieve based on task description
    - Problems:
        - Unrelated files may be needed
        - Providing too much files creates "cognitive load" for the model and extra cost
          (A single 128K call to GPT costs `$1.28`)

Overcoming Working Memory Limitations in Code Generation

3. Manual file selection

- With Vector DB recommendations + search
- Minimal set of files creates context, possibly allowing the prompt to be less precise.
- Developer needs to know the source deeper (and that's good!)



```yaml
task: prompt/task/bug/fix.md
attention:
  - src/git/getRepoInfo.js
requirements: |
  Use the package.json in the current working dir.
  When file not found, return affected fields empty.
```

https://github.com/tisztamo/Junior/blob/main/prompt/history/2023/09/06/13%3A26_Fix%20getRepoInfo%20package.json%20issue/prompt.yaml

“Cognitive Load” the models can bear

- Number of distinct commands
  - Junior self-development: 13 fixed + task
- How far the model must go from its training knowledge
  - E.g. coding in less popular frameworks like Solid.js
- Discrepancies between commanded format and the current state
  - E.g. After changing code style: "JavaScript files should only export named entities, do not use default exports" - cannot be fixed locally
  - Possible solution: Do not command style, leave the code as generated.

Minimizing “Cognitive Load”

- Selecting an output format the model can handle with ease
  - The lightest is the normal chat with code blocks, only extended with filenames - problematic to parse
  - Fine-tuning may help with more complex formats
  - “Function calls” are not so reliable yet
  - Generating diffs is just hard
    - diffs + fuzzy apply is a way
    - Always asking for full files is another
- Prompt engineering
  - Provide Example Output
  - Mark prompt sections clearly (markdown works)
  - ...

Minimizing “Cognitive Load” as a user (app developer)

- Prefer small files
- Clearly define requirements
  - Learn prompt engineering and the specific model
  - Avoid “it”, “that” etc. when referring to a previously mentioned concept.
  - Provide only the needed files
- Baby Steps
  - Avoid listing multiple tasks in the same prompt, except for trivial ones.
  - Decompose problems to smaller units and define them in separate prompts.

Approaches to LLM Coding 1: One Prompt Generators

- “Build an Entire App with a Single Prompt”
- Automated Waterfall
- e.g.: GPT-Engineer
  - https://github.com/AntonOsika/gpt-engineer

Approaches to LLM Coding 2: Iterative systems

- Focus on code
  - Typically VS Code based
  - e.g.: Rift: https://github.com/morph-labs/rift/blob/pranav/dev/assets/code-edit.gif
  - GitHub Copilot

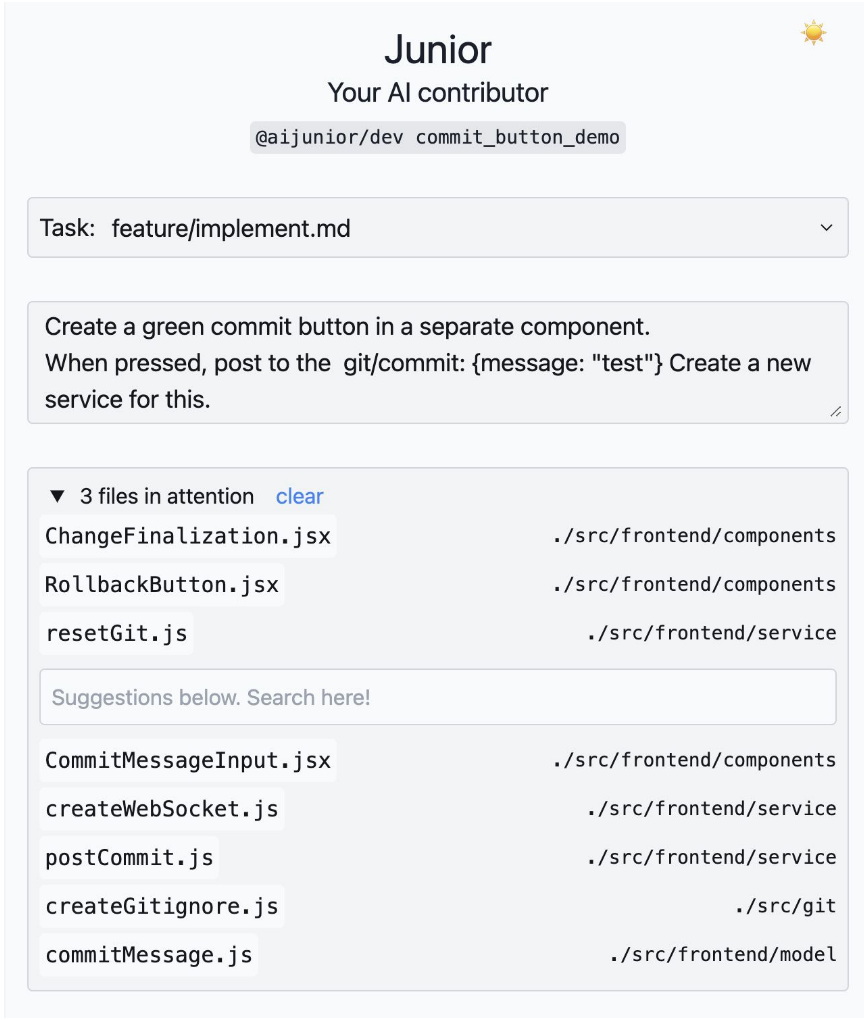
- Focus on requirements
  - Typically GitHub based
  - e.g.: Sweep: https://sweep.dev/

- Focus on Human-AI interaction
  - Own UI or CLI
  - aider: https://aider.chat
  - Junior: https://aijunior.dev
  - Copilot Workspace (research prototype from GitHub)

Junior Demo, Q & A

Schäffer Krisztián

linkedin.com/in/tisztamo/
aijunior.dev
github.com/tisztamo



Junior

Your AI contributor

`@aijunior/dev commit_button_demo`

Task: feature/implement.md

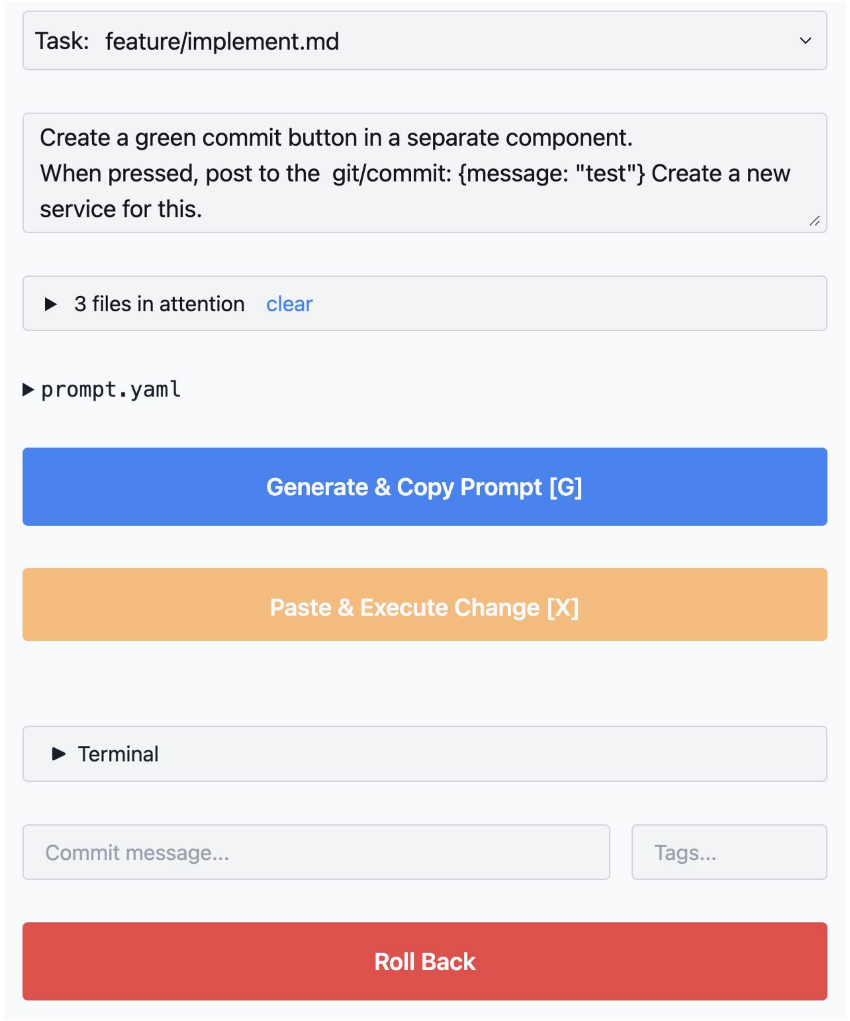
Create a green commit button in a separate component.
When pressed, post to the git/commit: {message: "test"} Create a new service for this.

▼ 3 files in attention clear

ChangeFinalization.jsx	./src/frontend/components
RollbackButton.jsx	./src/frontend/components
resetGit.js	./src/frontend/service

Suggestions below. Search here!

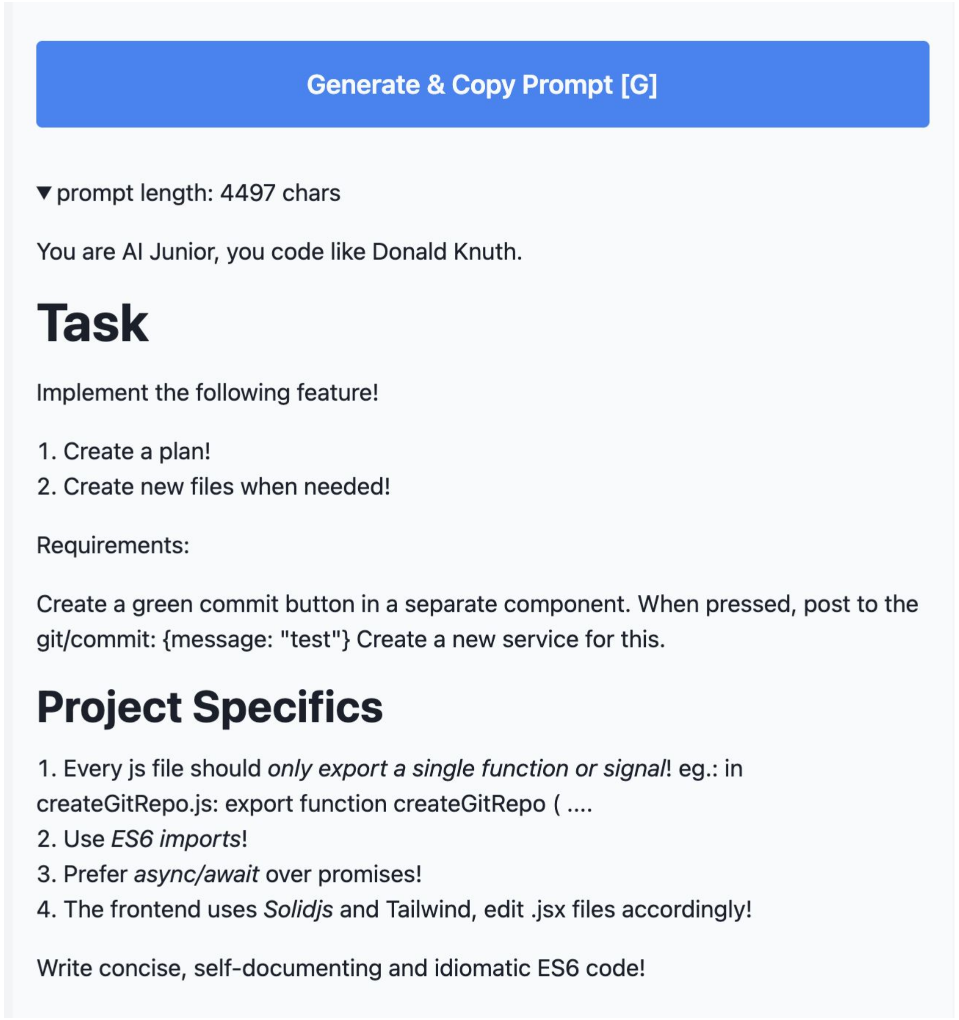
CommitMessageInput.jsx	./src/frontend/components
createWebSocket.js	./src/frontend/service
postCommit.js	./src/frontend/service
createGitignore.js	./src/git
commitMessage.js	./src/frontend/model


Task: feature/implement.md

Create a green commit button in a separate component.
When pressed, post to the git/commit: {message: "test"} Create a new service for this.

▶ 3 files in attention clear

▶ prompt.yaml



▼ prompt length: 4497 chars

You are AI Junior, you code like Donald Knuth.

Task

Implement the following feature!

1. Create a plan!
2. Create new files when needed!

Requirements:

Create a green commit button in a separate component. When pressed, post to the git/commit: {message: "test"} Create a new service for this.

Project Specifics

1. Every js file should *only export a single function or signal*! eg.: in createGitRepo.js: export function createGitRepo (....
2. Use *ES6 imports*!
3. Prefer *async/await* over promises!
4. The frontend uses *Solidjs* and Tailwind, edit .jsx files accordingly!

Write concise, self-documenting and idiomatic ES6 code!

Output Format

Encode and enclose your results as ./change.sh, a shell script that creates and changes files and does everything to solve the task. Avoid using sed. Always heredoc full files.

OS: OSX

Installed tools: npm, jq

EXAMPLE START

```
#!/bin/sh
set -e
goal=[Task description, max 9 words]
echo "Plan:"
echo "1. [...]"
cat > x.js << 'EOF'
[FULL content of the file]
EOF
echo "\033[32mDone: $goal\033[0m\n"
```

EXAMPLE END

Before starting, check if you need more files or info to solve the task.

If the task is not clear:

EXAMPLE START I need more information to solve the task. [Description of the missing info] EXAMPLE END

Do not edit files not provided in the working set! If you need more files:

Do not edit files not provided in the working set! If you need more files:

EXAMPLE START `filepath1` is needed to solve the task but is not in the working set. EXAMPLE END

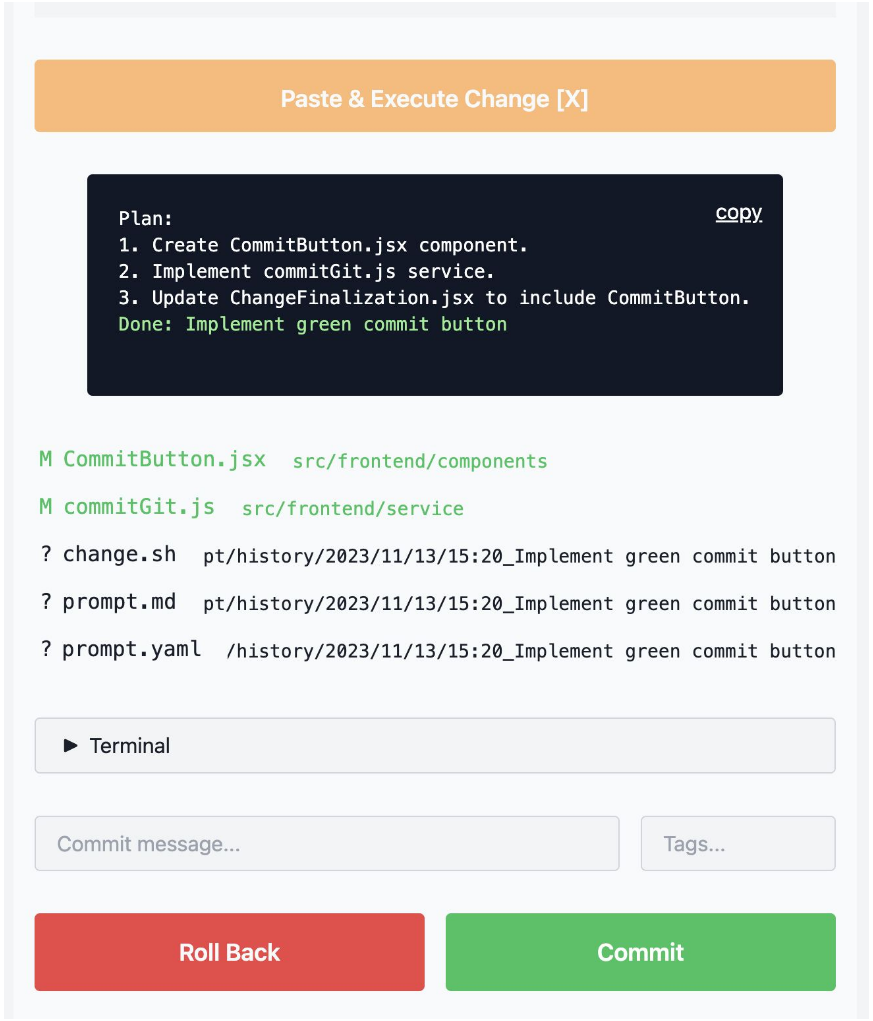
Working set

./src/frontend/components/ChangeFinalization.jsx:

```
import CommitMessageInput from './CommitMessageInput';
import TagsInput from './TagsInput';
import RollbackButton from './RollbackButton';

const ChangeFinalization = () => {
  return (
    <>
      <div className="flex w-full space-x-4">
        <div className="flex-grow w-3/4">
          <CommitMessageInput />
        </div>
        <div className="w-1/4">
          <TagsInput />
        </div>
      </div>
      <div className="flex w-full mt-1 space-x-4">
        <RollbackButton />
      </div>
    </>
  );
};

export default ChangeFinalization;
```



```
Plan:
1. Create CommitButton.jsx component.
2. Implement commitGit.js service.
3. Update ChangeFinalization.jsx to include CommitButton.
Done: Implement green commit button
```

```
M CommitButton.jsx   src/frontend/components
M commitGit.js   src/frontend/service
? change.sh   pt/history/2023/11/13/15:20_Implement green commit button
? prompt.md   pt/history/2023/11/13/15:20_Implement green commit button
? prompt.yaml   /history/2023/11/13/15:20_Implement green commit button
```

https://chat.openai.com/share/b6c216aa-8f2c-4aae-91fe-9973da6bf01a