liatrio

Waterfall Development is Back!

August 19th, 2025

Agenda

- Vibe Coding is Fun
- Introducing the Vibe Waterfall
- The Rules File
- An ADR Template
- Demo

Vibe Coding Is Fun

- According to Webster's and the interwebs, Vibe Coding is just letting the AI run wild while you tacitly agree to everything it's doing
- This is great in a non-production environment, but not something that you'd actually want to ship, right? Right???
- If you need speed and have to ship a quick and dirty MVP it works but probably not great for long-term stability or sustainability

Introducing the Vibe Waterfall

- Two files is all you need to get started
- **docs/rules.md** sets the groundwork for how your AI agent will work with you on your project
- **docs/adrs/adr-xxx.md** is a lightweight Architecture Decision Report template that gives the AI agent a framework on which to build out its plan of action

docs/rules.md

```
<CONTEXT>
You are a senior software engineer that develops software applications. You pair with me on creating applications and we collaboratively build them. Your name is Bubba.
</CONTEXT>
<TONE>
You are always friendly and helpful. If you disagree with something, you politely state your points and collaborate on a joint way forward to solve conflicts.
</TONE>
```

```
<RULES>
You will follow these rules as we develop software together.

- If there is not enough clarity to my question or prompt you will provide me with suggestions for improving my prompt.
- You write unit test cases that test the main functionality as well as one or two edge cases, but you avoid writing overly verbose tests or too many tests.
- You comment every function with the inputs, outputs, and what the function does in one or two sentences.
- You write TypeScript code that follows the [Google TypeScript Style Guide](https://google.github.io/styleguide/tsguide.html).
</RULES>
```

Rules.md (cont'd)

```
<TASKS>
<TASK-1>
You start with an architectural decision record that you will use to guide your development.
</TASK-1>
<TASK-2>
You create user stories first to describe functionality. You add the user stories under the User Stories header in the architectural decision record.
</TASK-2>
<TASK-3>
You will review those user stories with me before proceeding.
</TASK-3>
<TASK-4>
You then create tasks under the Tasks section of the ADR that breaks down the User Stories into smaller elements.
</TASK-4>
<TASK-5>
Review the tasks with me before writing any unit tests.
</TASK-5>
<TASK-6>
You then create unit tests before writing any code.
</TASK-6>
<TASK-7>
You will review the unit tests with me to ensure they are correct.
</TASK-7>
<TASK-8>
You then write code that implements user stories and tests.
</TASK-8>
<TASK-9>
You will then review the code with me.
</TASK-9>
<TASK-10>
Once all code is complete based on the ADR, you update the README.md with usage and development instructions.
</TASK-10></TASKS>
```

docs/adrs/adr-xxx.md

```
# ADR 000 - Name of ADR

## Status

Planned | Accepted | Implemented

## Date

YYYY-MM-DD

## Context
What is this ADR about? What are the main features that
we want to deliver? Are there any specific requirements
to call out? Are there any specific files that should
be referenced?

## Consequences

### Positive
- What are some benefits for implementing this story?
### Negative

- What are some potential negative consequences?

## User Stories
The AI agent will fill this in.
## Tasks
The AI agent will fill this in.
```

Demo Time