

# **GOAL: MULTIPLE-PAGE WEB APPLICATION (CHAT)**

Version 1:

- User will see chat page
- User can add message
- User will see updated messages
- A different user can load the page
- Other user will see updated messages
- Other user can add

# **BREAKING DOWN THE NEEDS:**

- One "page"
  - We will add other pages to handle login later
- All dynamic HTML
- Static CSS
- Only server-side JS

# ACTIONS

- Initial Load
- Post Message
- More cases later

# DATA (STATE)

- List of users
- List of messages
  - Text
  - Sender
  - Timestamp

# **SKELETON OF PARTS**

Best to confirm everything works in small steps

- If something doesn't, you only have a small amount of code to look for a bug in

"Stub" out a lot parts

- functions that don't do anything
- functions that return hardcoded values
- data that is created with some fake contents

# PAGE SKELETON

First, create a dynamic **handler** for the route /

Longer variables names are NOT BAD

But the community convention is to use `req` and `res` instead of `request` and `response`

```
app.get("/", (req, res) => {  
  res.send(`  
    // Paste the chat html here  
  `);  
});
```

Add a `chat.css` to the public folder

Restart the server and confirm it works

# NEXT - MAKE THE HTML FILLED IN DYNAMICALLY

Again, do it in parts and confirm each part

- Break up the page into functions that return those bits of text
  - Each bit reads from the state
- Separate the logic
  - One file controls the routes (`server.js`)
  - One file controls the data logic (`chat.js`)
  - One file wraps the data in HTML (`chat-web.js`)

# SEPARATION OF CONCERNS

Separation of Concerns is a programming strategy.

Parts of an application that don't NEED to be coupled (deeply tied together) AREN'T coupled

You want this separation because it makes changes more simple and easier to understand.

Imagine:

- Artificial kidney: Complex, hard to be confident in
- Water filter: Less complex, understandable



# **LAW OF DEMETER**

Law of Demeter (Principle of Least Knowledge)

The less parts know about other parts, the more each part can change without requiring changes in other parts

# SIDE EFFECTS

Something the function alters outside of return value

"Pure" means without side-effects - same inputs always generate same outputs

A Useful program must have side effects, but side-effects introduce complexity (the enemy!)

So always be careful in accepting side-effects.

Side effects should be

- Obvious from the function name
- Predictable