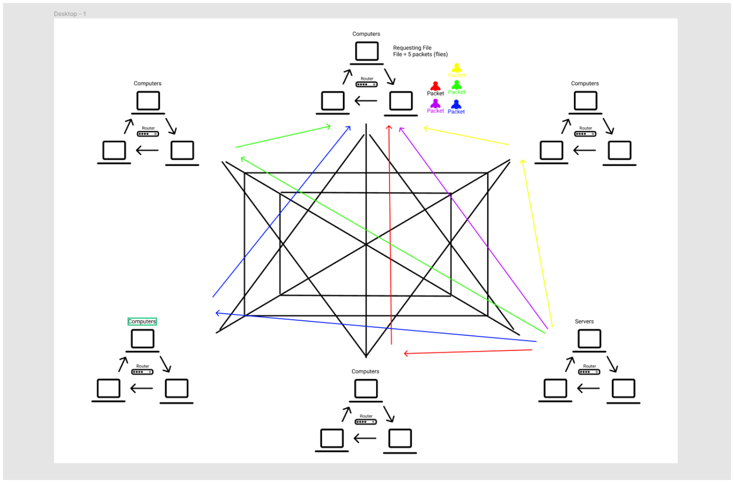
How the Web Works
In this lab, you'll be working with a partner to explore a little more about the internet, the web, requests, responses and more. You'll be reading and writing about concepts as well as practicing some of the commands that we saw during the lecture earlier.

Topic 1: The Internet and the World Wide Web

1) What is the internet? (hint: here) The Internet is a worldwide network of networks that uses the Internet protocol suite
2) What is the world wide web? (hint: here) an interconnected system of public webpages accessible through the Internet.
3) Partner One: read this page on how the internet works, Partner Two: read this page on how the world wide web works. When you're done reading, come back together and and answer the following questions
    a) What are networks? a collection of computer systems and devices which are linked together using a wireless network or via communication devices and transmission media.
    b) What are servers? a computer or system that provides resources, data, services, or programs to other computers, known as clients, over a network.
    c) What are routers? Routers connect computers and other devices to the Internet
    d) What are packets? a formatted unit of data carried by a packet-switched network.
4) Come up with a metaphor for the internet and the web, you can do a single one if you think of one that puts them together or two separate ones (feel free to use one you've heard today or read about if you can't think of a new one, but spend at least 10 minutes trying to think of something different before you resort to that)

The internet is the spiderweb and the web is the flies caught on the spiderweb

5) Draw out a diagram of the infrastructure of the internet and how a request and response travel using your metaphor (like the map and letters we saw during the lecture). Insert the drawing into this document (can be a picture of a physical drawing, a Google Drawing, a Figma drawing, etc)



Topic 2: IP Addresses and Domains

1) What is the difference between an IP address and a domain name? The IP address is a set of numbers that point to the exact address while a domain name links to the IP address and is easier for humans to use
2) What's devmountain.com's IP address? (Hint: use 'ping' in the terminal) 172.67.9.59
3) Try to access devmountain.com by its IP address. It shouldn't work because we have our sites protected by a service called CloudFlare. Why might it be important to not let users access your site directly at the IP address? A malicious user with access to an IP address could make changes to a website; having services such as CloudFlare protects an IP address from attacks.
4) How do our browsers know the IP address of a website when we type in its domain name? (If you need a refresher, go read this comic linked in the handout from this lecture) The domain name is part of the URL, which points to the IP address. The domain name is a shortcut to the IP address

Topic 3: How a web page loads into a browser

The steps of how a web page is requested and sent are in the table below. However, **they are out of order**. Unscramble them and explain your thinking/reasoning in the second two columns of the table.

Steps Scrambled	Steps in Correct Order	Why did you put this step in this position?
Example: Here is an example step	*Here is an example step*	*- I put this step first because _____* *- I put this step before/after _____ because _____*
Request reaches app server	2	I put this step second because the request needs to reach the server in order to send the necessary information back
HTML processing finishes	5	I put this as 5th because the HTML loads after javascript
App code finishes execution	3	I put this 3rd because javascript renders before html
Initial request (link clicked, URL visited)	1	This is first because to connect to any

		IP/domain/web page, we need to make the request
Page rendered in browser	6	This is the completed request
Browser receives HTML, begins processing	4	Once the server responds with the packet of information, it sends it back to the local computer to begin processing The computer has to process the files to load

Topic 4: Requests and Responses

Setup

- Download the folder for this exercise from Frodo.
- Make sure you unzip it.
- Open it in VS Code
- Run `npm i` in the terminal (make sure you're in the web-works folder you just downloaded).
    - You'll know it was successful if you see a node_modules folder in the web-works folder.
- Run `node server.js` in the terminal (also in the web-works folder) and you should see a log to the terminal saying 'serving up port 4500'
- You'll be using this file to figure out what will happen when you make requests to this server, so read it over to see what's going on. We'll be getting into the two GET functions and the POST function.

Part A: GET /

- You'll start by looking at the function that runs when we make a get request to /, which looks like this: http://localhost:4500 or http://localhost:4500/
- You'll use the curl command to make a request and read the response in your terminal

1) Predict what you'll see as the body of the response: "Jurnee"
2) Predict what the content-type of the response will be:

- Open a terminal window and run `curl -i http:localhost:4500`

3) Were you correct about the body? If yes, how/why did you make your prediction? If not, what was it and why? We were correct, we saw the h1 and h2
4) Were you correct about the content-type of the response? If yes, how/why did you make your prediction? If not, what was it and why? Yes, we assumed it would just be text

Part B: GET /entries

- Now look at the next function, the one that runs on get requests to /entries.
- You'll use the curl command again. This time, you'll need to figure out how to modify it to get the response that you need.

1) Predict what you'll see as the body of the response: ???
2) Predict what the content-type of the response will be:

- In your terminal, run a curl command to get request this server for /entries

3) Were you correct about the body? If yes, how/why did you make your prediction? If not, what was it and why? No, we were not sure what to expect from this command
4) Were you correct about the content-type of the response? If yes, how/why did you make your prediction? If not, what was it and why? No, because we did not know what to expect

Part C: POST /entry

- Last, read over the function that runs a post request.

1) At a base level, what is this function doing? (There are four parts to this) It seems this code is pushing a new entry into the entries variable that includes id, date, and content
2) To get this function to work, we need to send a body object with our request. Looking at the function in server.js, what properties do you know you'll need to include on that body object? And what data types will they be (hint: look at the objects in the entries array)? ID, date, content
3) Plan the object that you'll send with your request. Remember that it needs to be written as a JSON object inside strings. JSON objects properties/keys and values need to be in **double quotes** and separated by commas.

```
{
        Id: 3,
        Date: 'August 30',
        Content: 'Pushin the stuff'
}
```

4) What URL will you be making this request to? localhost:4500
5) Predict what you'll see as the body of the response: {"Id":3, "date": "August 30", "Content": "Pushin the stuff" }
6) Predict what the content-type of the response will be: application/json; charset=utf-8

- In your terminal, enter the curl command to make this request. It should look something like the example below, with the information you decided on in steps 3 and 4 instead of the ALL CAPS WORDS.
    - curl -i -X POST -H 'Content-type: application/json' -d JSONOBJECT URL

7) Were you correct about the body? If yes, how/why did you make your prediction? If not, what was it and why? Yes, we were correct / we assumed it would push a new entry

8) Were you correct about the content-type of the response? If yes, how/why did you make your prediction? If not, what was it and why? Yes, we thought it be application/json; charset=utf-8

Submission

1. Save this document as a PDF
2. Go to Github and create a new repository. (Click the little + in the upper right hand corner.)
3. Name your repository "web-works" (or something like that).
4. Click "uploading an existing file" under the "Quick setup heading".
5. Choose your web works PDF document to upload.
6. Add "commit message" under the heading "Commit changes". A good commit message would be something like "Adding web works problems."
7. Click commit changes.

Further Study: More curl

Visit this link and do the exercises using the website provided. Keep track of the commands you used in this document. (Don't forget to resubmit to GitHub when you complete this section)