

Week 1 Recap

Akshaj Kadaveru

You can start coding your websites now!! Project Skeleton is released on Piazza

- **Read the README for instructions on how to use (don't clone the skeleton!)**
- Milestone 2 (Minimal viable product) due Tuesday Jan 18, 2022 (6pm)
- **Hackathon Friday!** Essentially extended office hours from 7pm-1am, we help you with making your site
 - Start asap so that you can get help at office hours on Wednesday / so that you will have better questions to ask during the hackathon

Office Hours tonight

- Monday and Wednesday 7-9pm in 32-082 or on weblab.to/q
- be there!!

Review Resource <http://weblab.to/week1>

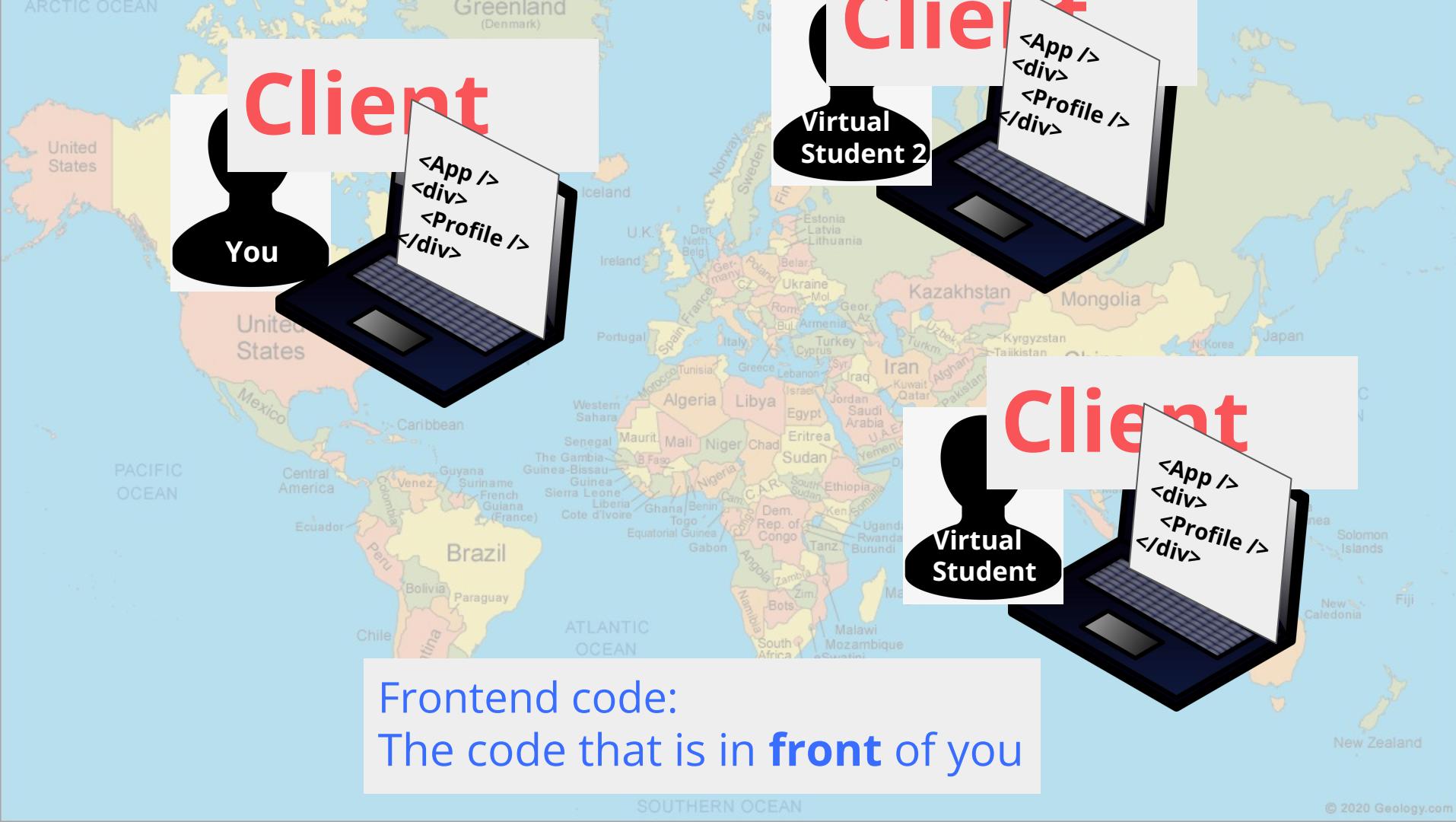
Front End:

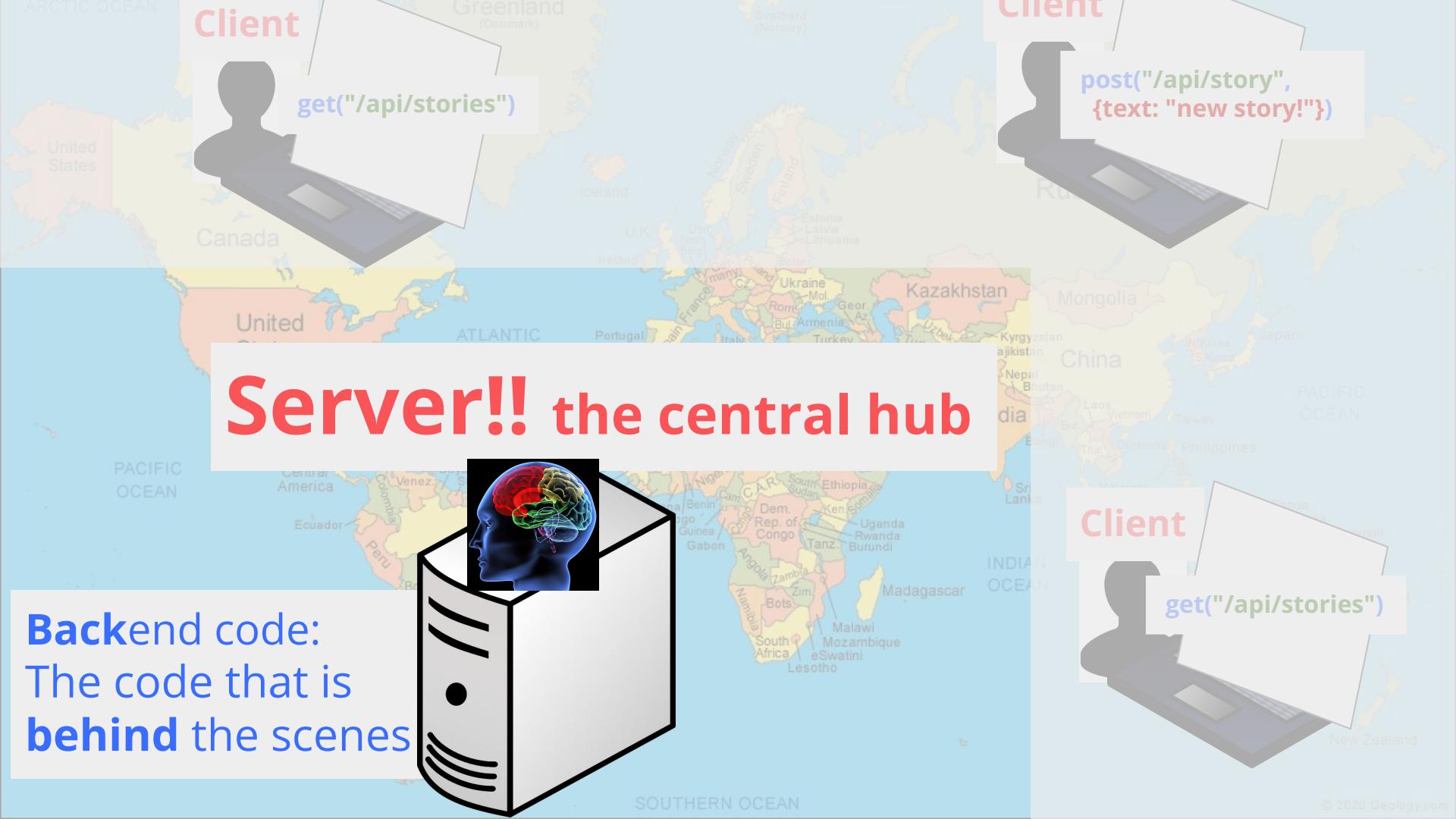
- [React Study Guide](#)
- Workshop 2 for a simple React app
- Workshop 3 for a more complex app, including how to use **get** and **post** requests to communicate with the backend
- [To-do List React practice](#)

Back End:

- Workshop 5 for Catbook with both a frontend and backend
- Workshop 6 for Catbook with a frontend, backend, and database
- api.js is the important backend file. You don't need to understand the server.js file. Not important.
- The backend is a bunch of descriptions of how to respond to certain get and post requests: [Catbook Backend Guide](#)
- Databases are a place to store data, and we will store data on MongoDB Atlas
- [A guide to MongoDB/Mongoose](#)

Recap: Frontend (Client) vs Backend (Server)





Server!! the central hub

Backend code:
The code that is
behind the scenes



Front-end (client)

- What the users see
- Every user sees this in their browser
- In **front** of the user
- Makes your site look pretty / easy to use

Back-end (server)

- Responds to get/post requests from the users
- Runs on the server
- **Behind** the scenes
- The ***core functionality*** of your site

Front-end

```
import React from "react";
import NavBar from "./NavBar.js";
import Intro from "./Intro.js";
import Photos from "./Photos.js";

const App = () => {
  return (
    <div>
      <NavBar />
      <div>
        <div>
          <Intro />
          <Photos />
        </div>
        <Post />
      </div>
    </div>
  );
};

export default App;
```

Back-end

```
router.get("/test", (req, res) => {
  });

router.get("/stories", (req, res) => {
  });

router.get("/comment", (req, res) => {
  });

router.post("/story", (req, res) => {
  });

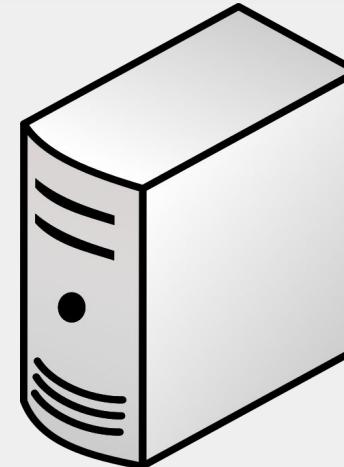
router.post("/comment", (req, res) => {
  });

9
```

Catbook could still exist with no frontend

- We'd have to send our own post requests to the server to add new stories and comments
- We'd have to send our own get requests to the server to read stories and comments
- It works but is **bad user experience**

Server



**Front end
developer**



**Back end
developer**

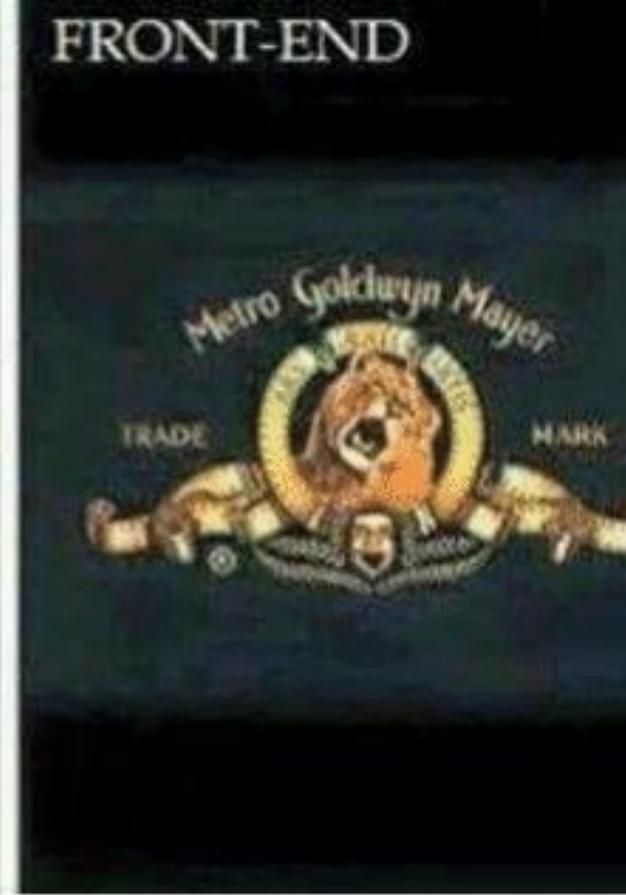


MUshaf Ali Zaidi

BACK-END



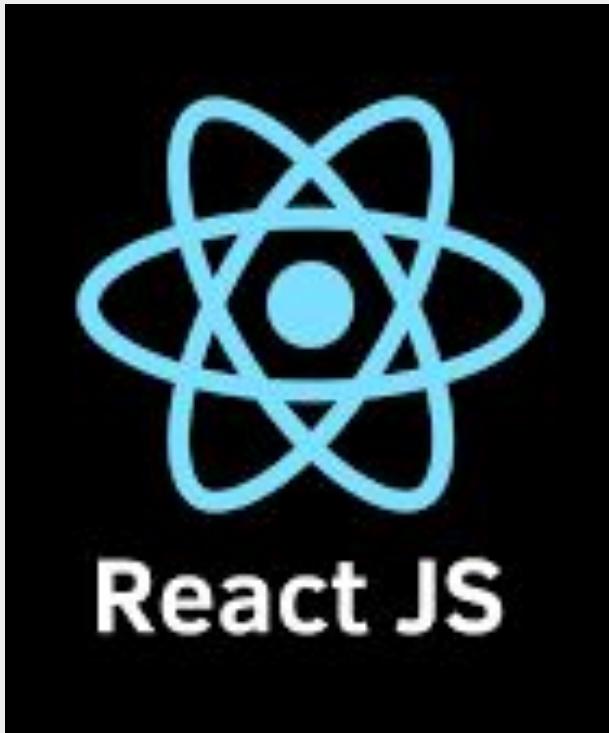
FRONT-END



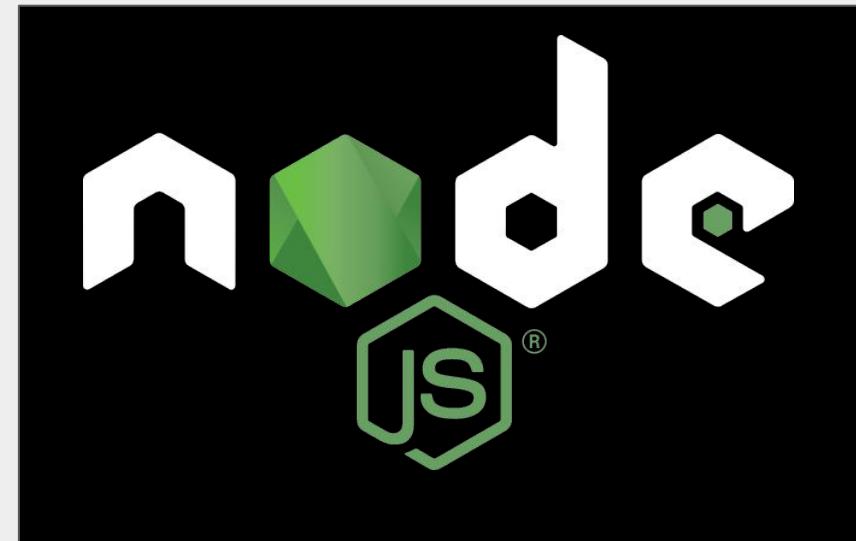
Front-End & Back-End

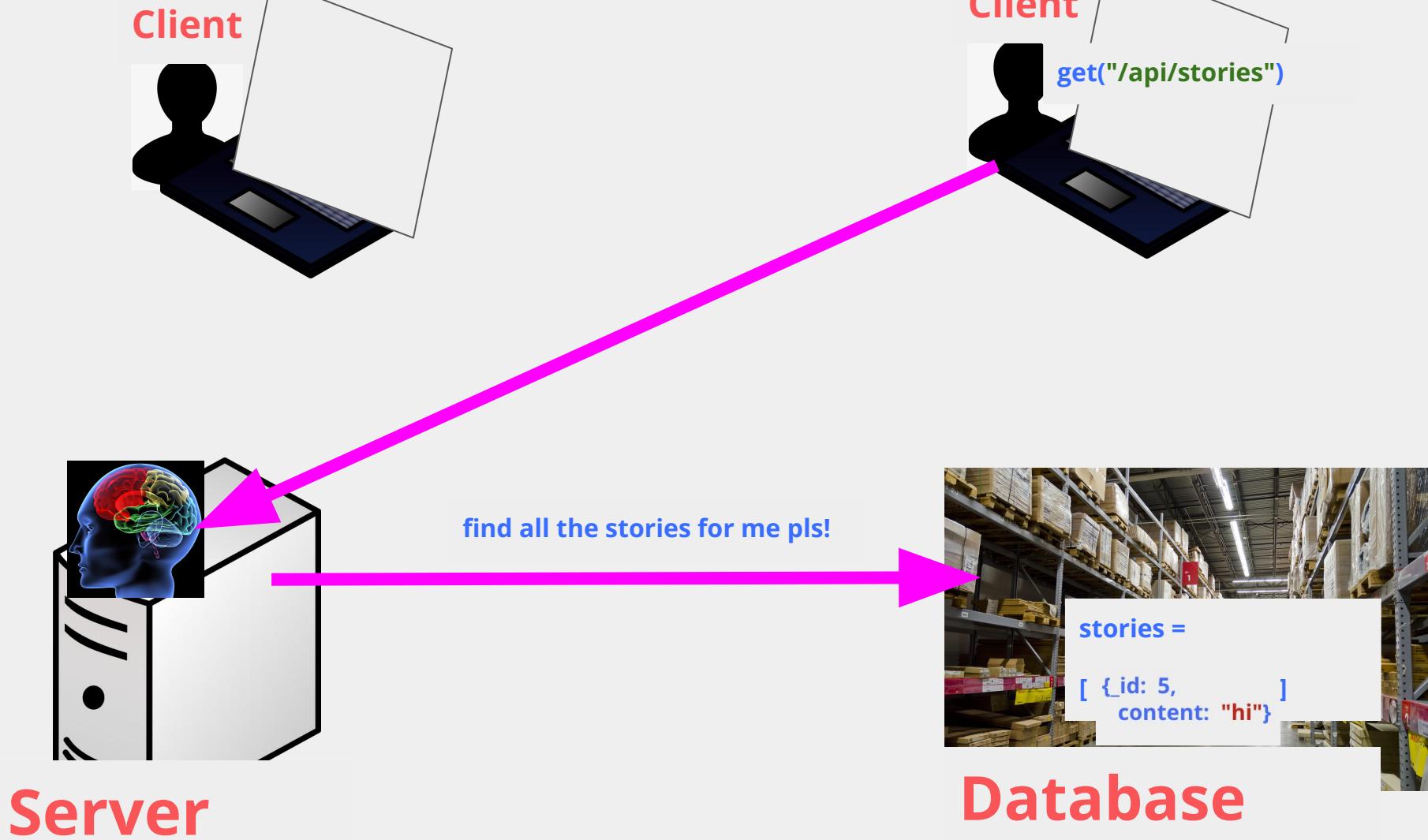


Front-end



Back-end





Databases... Schemas

```
//define a story schema for the database
const StorySchema = new mongoose.Schema({
  creator_name: String,
  content: String,
});
```

```
//define a comment schema for the database
const CommentSchema = new mongoose.Schema({
  creator_name: String,
  parent: String, // links to the _id of a pa
  content: String,
});
```

Databases... Retrieving Data from MongoDB

```
router.get("/stories", (req, res) => {
  // empty selector means get all documents
  Story.find({}).then((stories) => {
    res.send(stories)
  });
});
```

Gets all the stories from
MongoDB

Databases... Adding Data to MongoDB

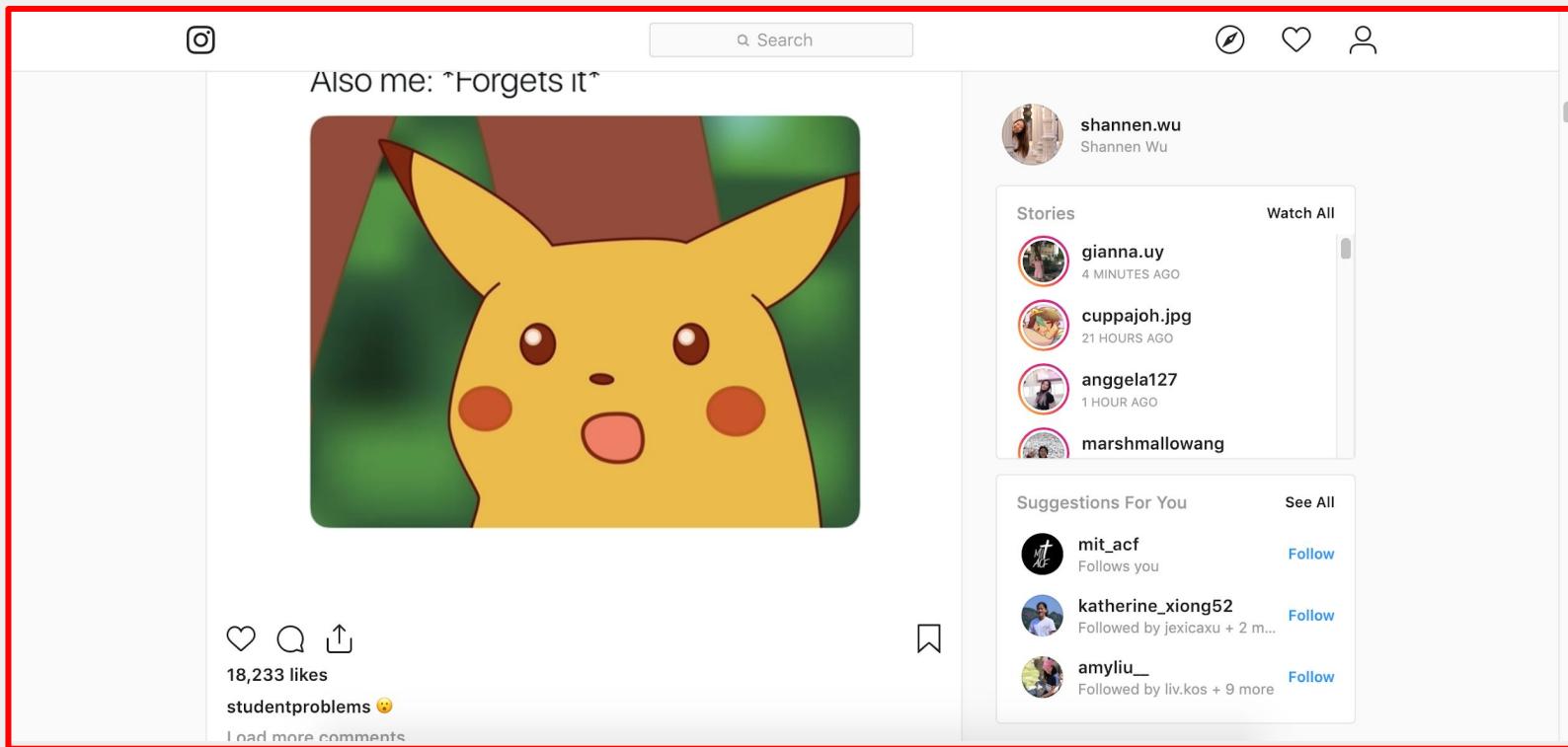
```
router.post("/story", (req, res) => {
  const newStory = new Story({
    creator_name: MY_NAME,
    content: req.body.content,
  });
  
  newStory.save();
});
```

Saves newStory to MongoDB

React

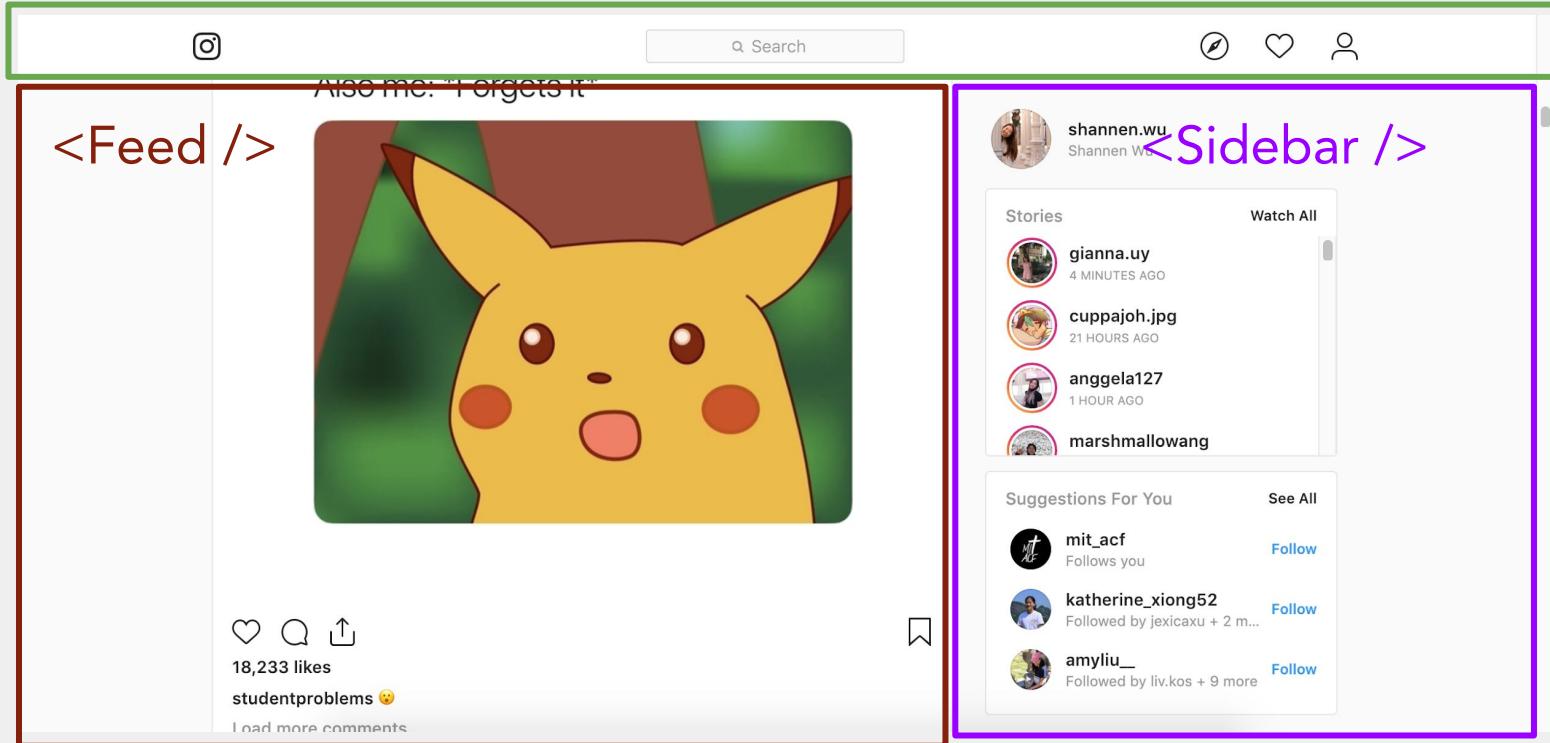
React Apps are "components of components"

The App component (<App />)

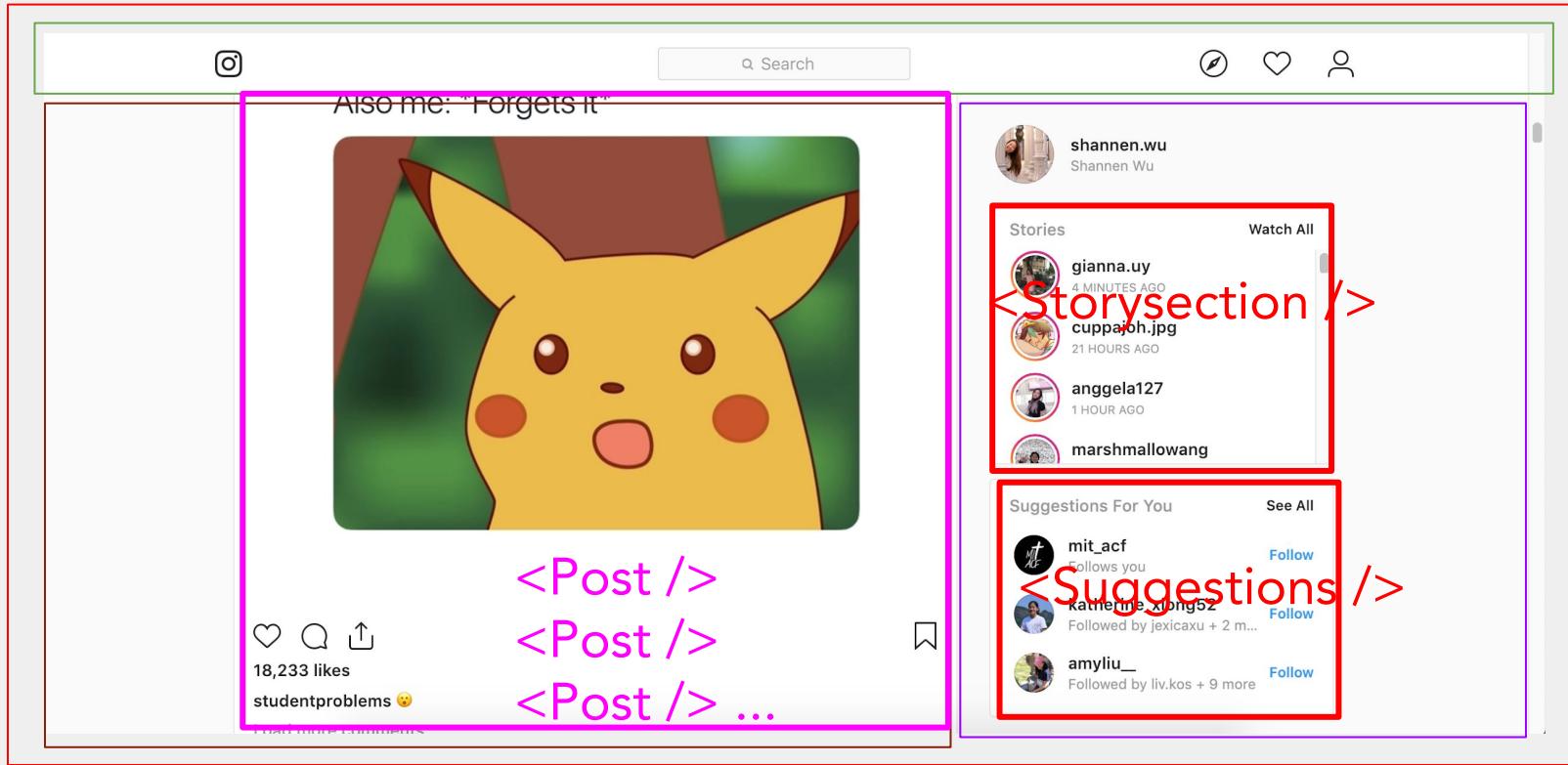


React Apps are "components of components"

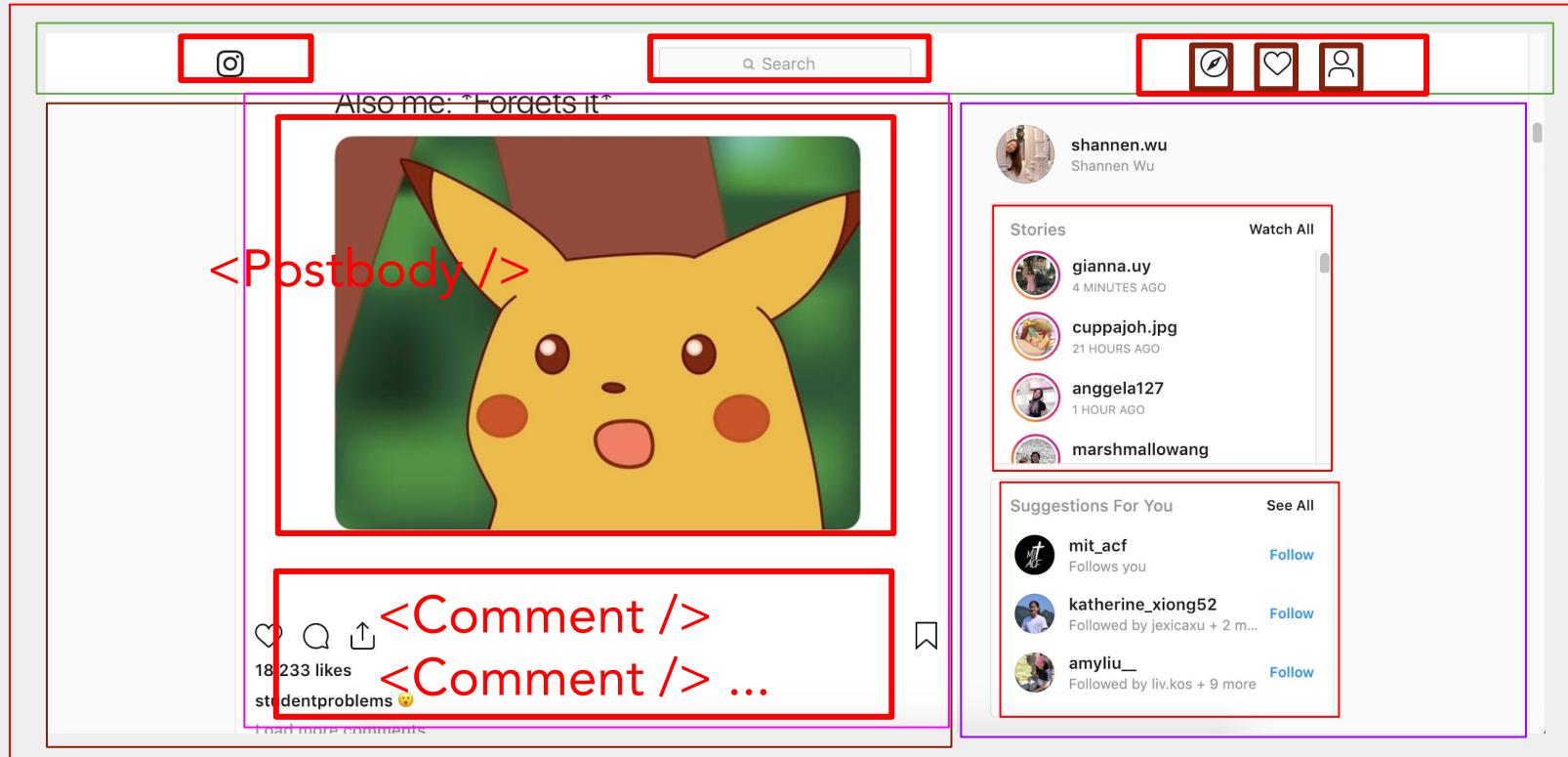
<Navbar />



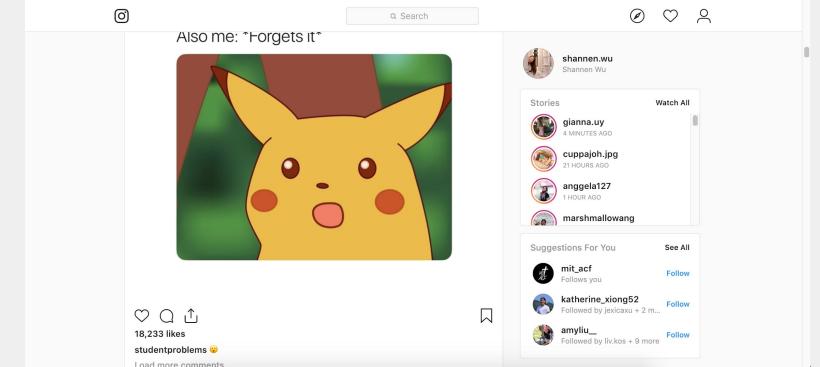
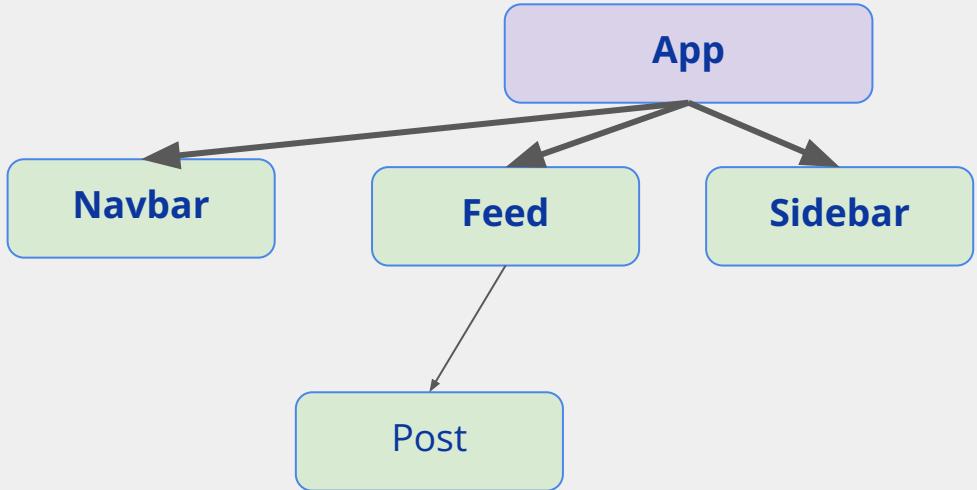
React Apps are "components of components"



React Apps are "components of components"



Big Picture



App

```
// App.js
```

```
const App = () => {
  return (
    <div>
      <Navbar />
      <div>
        <Feed />
        <Sidebar />
      </div>
    </div>
  )
}
```

Props ('Inputs')

Inputs passed from a parent to a child component

These are all props! (the inputs)



```
<Post name="Akshaj" text="Welcome to web lab!" />
```

here, props = {name: "Akshaj", text: "Welcome to web lab!"}

State

Updatable pieces of information maintained by a component.

```
const [status, setStatus] = useState("busy");
const [isOnline, setIsOnline] = useState(false);
```

What we've learned in React

- Websites are divided into **components**
- components can have inputs, which are **props**
- components can have their own private, updatable information: **state**

React: useEffect and get/post
(communicating with server)

Weather app

```
const WeatherApp = () => {
  const [weather, setWeather] = useState("Cloudy");

  return <div>{weather}</div>;
};
```

Why can't we do this

```
const WeatherApp = () => {
  const [weather, setWeather] = useState("Cloudy");

  const actualWeather = get("/api/theweather");
  setWeather(actualWeather);

  return <div>{weather}</div>;
};
```

Why can't we do this

```
const WeatherApp = () => {  
  const [weather, setWeather] = useState("Cloudy");  
  
  const actualWeather = get("/api/theweather");  
  setWeather(actualWeather);  
  
  return <div>{weather}</div>;  
};
```

Asynchronous.. we need to
wait for the get request to
finish first

Why does this still not work (we get error infinite render??)

```
const WeatherApp = () => {
  const [weather, setWeather] = useState("Cloudy");

  get("/api/theweather").then((actualWeather) => {
    setWeather(actualWeather);
  });

  return <div>{weather}</div>;
};
```

Why does this still not work? (we get an infinite render??)

```
const Weather = () => {  
  const [weather, setWeather] = useState("Cloudy");  
  
  useEffect(() => {  
    const interval = setInterval(() => {  
      const newWeather = ["Sunny", "Cloudy", "Rainy", "Snowy", "Thunderstorms"][Math.floor(Math.random() * 5)];  
      setWeather(newWeather);  
    }, 1000);  
  
    return () => clearInterval(interval);  
  }, []);  
  
  return <div>{weather}</div>;  
};
```

**Components re-run
their code
whenever state or
props changes**

Cloudy");

Weather) => {

Solution: useEffect

```
const WeatherApp = () => {
  const [weather, setWeather] = useState("Cloudy");

  useEffect(() => {
    get("/api/weather").then((actualWeather) => {
      setWeather(actualWeather);
    });
  }, []);

  return <div>{weather}</div>;
};
```

runs only once now!



Talking to the Backend (Server) from the Frontend (Client)

```
useEffect(() => {  
  get("/api/songs", {genre: "pop", year: 2000})  
  .then((songsFromServer)=>{  
    setSongs(songsFromServer);  
  })  
, [])
```

The endpoint you want to hit

What to do after the data comes back

What we've learned in React

- Websites are divided into **components**
- components can have inputs, which are **props**
- components can have their own private, updatable information: **state**
- Components run their code **once at the beginning**, and then **once every time a prop or a state changes**
- To run something only once at the beginning, we use **useEffect(function, [])**
- **get**("/api/something", someData) does a get request to "/api/something", and gets back data (asynchronously)
- **post**("/api/something", someData) does a post request to "/api/something" (asynchronously)
- **Router** is our way of having different 'pages' (with their own URL) on your site

Connecting everything together

Where to start



How to start coding up your webapp

1. (Backend) Figure out what you're storing in your database
2. (Backend) Figure out what actions your server will need to be able to do
3. (Frontend) Divide your design into components
4. (Frontend) Figure out the props/state of each component, and where your get/post requests will happen

<http://weblab.to/recap-app>

Welcome to AceBook!

 Post

Tobechi Onwuka: its no broken!

Alexander Pink: ;)

Jesus Smith: it's broken?

Eloise Zeng: hiii

Lucas Blakeslee: Hi!

Sky Loke: michael broke it

Kayla Almonte: poggg

Jesus Smith: wooo

Hannah Zhang: hellllooo

Alexander Tong: <h1>test</h1>

Christine Tang: hello!

Jeremias Figueiredo: cats>dogs

Braden Hechmer: wassup

Alea Pindat Kahele: hello

Tobechi Onwuka: supaa coool!

Michael Phillips:

Hi

Step 1: What do we need to store in our database

Welcome to AceBook!

Post

Tobechi Onwuka: its no broken!
Alexander Pink: :)
Jesus Smith: it's broken?
Eloise Zeng: hiii
Lucas Blakeslee: Hi!
Sky Loke: michael broke it
Kayla Almonte: poggg
Jesus Smith: wooo
Hannah Zhang: hellllooo
Alexander Tong: <h1>test</h1>
Christine Tang: hello!
Jeremias Figueiredo: cats>dogs
Braden Hechmer: wassup
Alea Pindat Kahale: hello
Tobechi Onwuka: supaa coool!
Michael Phillips:
Hi.....

Step 1: What do we need to store in our database

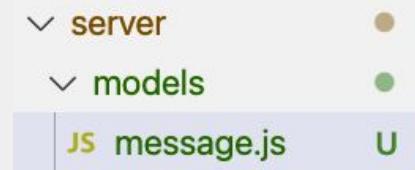
● Messages

Welcome to AceBook!

Tobechi Onwuka: its no broken!
Alexander Pink: :)
Jesus Smith: it's broken?
Eloise Zeng: hiii
Lucas Blakeslee: Hi!
Sky Loke: michael broke it
Kayla Almonte: poggg
Jesus Smith: wooo
Hannah Zhang: hellllooo
Alexander Tong: <h1>test</h1>
Christine Tang: hello!
Jeremias Figueiredo: cats>dogs
Braden Hechmer: wassup
Alea Pindat Kahale: hello
Tobechi Onwuka: supaa coool!
Michael Phillips:
Hi.....

Step 1: What do we need to store in our database

- Add a 'Message' schema



server > models > **JS** message.js > ...

```
1  const mongoose = require("mongoose");
2
3  const MessageSchema = new mongoose.Schema({
4      name: String,
5      text: String,
6  });
7
8  module.exports = mongoose.model("message", MessageSchema);
```

Step 1 Practice: What Schemas might Piazza store?

The screenshot shows the Piazza interface. At the top, there's a navigation bar with links for LIVE Q&A, Drafts, Reading list, and various class management options like logistics, milestone0, milestone1, milestone2, final_submission, setup, deployment, frontend, backend, and more. A user profile for Akshaj Kadaveru is visible.

The main content area displays a note titled "Project Skeleton Released" by "note @243". The note content includes:

You can get started on your projects now! The project skeleton has been released. It has most of the frontend and backend setup that you will need and things like Google login and sockets which won't be taught until tomorrow.

Read the README for instructions on how to use this skeleton.

Important: do not clone this skeleton! Instead, you will download all of the files, including hidden ones, manually and move them to your team's repository. Detailed instructions including a video demo are in the README.

If you or your team want help getting started (turning your wireframe into code, doing this setup, or anything) please come to office hours on Monday or ask on Piazza!

The sidebar on the left, under the "PINNED" section, lists several posts:

- Instr Project Skeleton Released (1/9/22)
- Instr Week 1 Overview (1/7/22)
- Instr Day 5 Recap (1/7/22)
- Instr 2022 Sponsor Prize Categories (1/4/22)
- Instr Weblab Lecture Livestreams... (12/31/21)
- Private GitHub (1:07AM)
- Set up skeleton code (12:58AM)

At the bottom, there are statistics: Average Response Time (2 min), Special Mentions (Ilya Gulko), Online Now (31), and This Week (481). There's also a footer with copyright information: Copyright © 2022 Piazza Technologies, Inc. All Rights Reserved. Privacy Policy, Copyright Policy, Terms of Use, Blog, Report Bug!

Step 1 Practice: What Schemas might Piazza store?

The screenshot shows the Piazza application interface. At the top, there's a navigation bar with links for Q & A, Resources, Statistics, Manage Class, and a user profile for Akshaj Kadavaru. Below the navigation is a search bar and a note history section. The main content area displays several posts:

- PINNED**
 - Instr Project Skeleton Released (1/9/22)
You can get started on your projects now! The project skeleton has been released. It has most of the frontend and backend.
 - Added to reading list
 - An instructor thinks this is a good note
 - Instr Week 1 Overview (1/7/22)
We put together a quick week 1 overview of most of the core content: take a look here!
 - Week 1 Overview
 - An instructor thinks this is a good note
 - Instr Day 5 Recap (1/7/22)
Hi Everyone! Congrats on finishing the first week of weblab!! 🎉 Reminder we have OH from 3-4 PM ET today on the queue
 - An instructor thinks this is a good note
 - Instr 2022 Sponsor Prize Categori... (1/4/22)
Every year, web.lab is made possible due to the generous support of our sponsors. In addition, three of our sponsors are
 - An instructor thinks this is a good note
 - Instr Weblab Lecture Livestreams... (12/31/21)
Hey guys! We've been getting lots of questions about lectures being live-streamed/recorded for Weblab and TLDR: we
 - An instructor thinks this is a good note
- TODAY**
 - Private GitHub (1:07AM)
Hello! I apologize for the inconvenience but I recently realized that my github invitation has expired. If possible, co
 - i
 - Set up skeleton code (12:58AM)
When I try to set up the skeleton code I get the errors: \$ git commit -m "Skeleton
 - i

- User
- Class
- Question
- Answer
- School
- Tag
- etc.

Step 1 Practice: What Schemas might Piazza store?

```
const ClassSchema = new mongoose.Schema({  
    title: String,  
    instructors: [String],  
    description: String,  
    schoolId: String,  
});
```

```
const QuestionSchema = new mongoose.Schema({  
    questionText: String,  
    userId: String,  
    classId: String,  
    tags: [String],  
    time: Date,  
});
```

Step 2: What server endpoints will we need?

Welcome to AceBook!

Post

Tobechi Onwuka: its no broken!
Alexander Pink: :)
Jesus Smith: it's broken?
Eloise Zeng: hiii
Lucas Blakeslee: Hi!
Sky Loke: michael broke it
Kayla Almonte: poggg
Jesus Smith: wooo
Hannah Zhang: hellllooo
Alexander Tong: <h1>test</h1>
Christine Tang: hello!
Jeremias Figueiredo: cats>dogs
Braden Hechmer: wassup
Alea Pindat Kahale: hello
Tobechi Onwuka: supaa coool!
Michael Phillips:
Hi.....

Step 2: What server endpoints will we need?

- Get all the messages
- Post a new message

Welcome to AceBook!

Tobechi Onwuka: its no broken!

Alexander Pink: ;)

Jesus Smith: it's broken?

Eloise Zeng: hiii

Lucas Blakeslee: Hi!

Sky Loke: michael broke it

Kayla Almonte: poggg

Jesus Smith: wooo

Hannah Zhang: hellllooo

Alexander Tong: <h1>test</h1>

Christine Tang: hello!

Jeremias Figueiredo: cats>dogs

Braden Hechmer: wassup

Alea Pindat Kahale: hello

Tobechi Onwuka: supaa coool!

Michael Phillips:

Post

Step 2: What server endpoints will we need?

```
// Get all of the messages
router.get("/allmessages", (req, res) => {

});

// Send a message to everyone
// Needs the message text as an input
router.post("/newmessage", (req, res) => {

});
```

Step 2: What server endpoints will we need?

```
router.get("/allmessages", (req, res) => {
  Message.find({}).then((messagesFromDB) => {
    res.send(messagesFromDB);
  });
});

router.post("/newmessage", (req, res) => {
  const newMessage = new Message({
    name: "Anonymous",
    text: req.body.text,
  });
  newMessage.save();
});
```

Step 2 Practice: What server endpoints will we need?

The screenshot shows the Piazza interface with a note titled "Project Skeleton Released". The note content is as follows:

Project Skeleton Released
You can get started on your projects now!
The project skeleton has been released. It has most of the frontend and backend.

- Added to reading list
- An instructor thinks this is a good note

Instr Week 1 Overview
1/7/22
We put together a quick week 1 overview of most of the core content: take a look here! Week 1 Overview

- An instructor thinks this is a good note

Instr Day 5 Recap
1/7/22
Hi Everyone! Congrats on finishing the first week of weblab!! 🎉 Reminder we have OH from 3-4 PM ET today on the queue

- An instructor thinks this is a good note

Instr 2022 Sponsor Prize Categories...
1/4/22
Every year, web.lab is made possible due to the generous support of our sponsors. In addition, three of our sponsors are

- An instructor thinks this is a good note

Instr Weblab Lecture Livestreams...
12/31/21
Hey guys! We've been getting lots of questions about lectures being live-streamed/recoded for Weblab and TLDR: we

- An instructor thinks this is a good note

TODAY

Private GitHub
1:07AM
Hello! I apologize for the inconvenience but I recently realized that my github invitation has expired. If possible, co

- I

Set up skeleton code
12:58AM
When I try to set up the skeleton code I get the errors: \$ git commit -m "Skeleton

- I

note @243 75 views

stop following

Average Response Time: 2 min | Special Mentions: Ilya Gulko answered does weblab have merch? in 1 min. 5 hours... | Online Now | This Week: 31 | 481

Copyright © 2022 Piazza Technologies, Inc. All Rights Reserved. Privacy Policy Copyright Policy Terms of Use Blog Report Bug!

Step 2 Practice: What server endpoints will we need?

The screenshot shows the Piazza interface. On the left, there's a sidebar with pinned posts and a search bar. The main area displays a list of posts:

- Instr Project Skeleton Released** (1/9/22): You can get started on your projects now! The project skeleton has been released. It has most of the frontend and backend.
- Instr Week 1 Overview** (1/7/22): We put together a quick week 1 overview of most of the core content: take a look here!
- Instr Day 5 Recap** (1/7/22): Hi Everyone! Congrats on finishing the first week of weblab! Reminder we have OH from 3-4 PM ET today on the queue.
- Instr 2022 Sponsor Prize Categor...** (1/4/22): Every year, web.lab is made possible due to the generous support of our sponsors. In addition, three of our sponsors are
- Instr Weblab Lecture Livestreams...** (12/31/21): Hey guys! We've been getting lots of questions about lectures being live-streamed/recorded for Weblab and TLDR: we
- Private GitHub** (1:07AM): Hello! I apologize for the inconvenience but I recently realized that my github invitation has expired. If possible, co
- Set up skeleton code** (12:58AM): When I try to set up the skeleton code I get the errors: \$ git commit -m "Skeleton

On the right, a detailed view of a note titled "Project Skeleton Released" is shown. The note content includes:

You can get started on your projects now!
The project skeleton has been released. It has most of the frontend and backend setup that you will need and things like Google login and sockets which won't be taught until tomorrow.
***Read the README for instructions on how to use this skeleton.**
Important: do not clone this skeleton! Instead, you will download all of the files, including hidden ones, manually and move them to your teams repository. Detailed instructions including a video demo are in the README.
If you or your team want help getting started (turning your wireframe into code, doing this setup, or anything) please come to office hours on Monday or ask on Piazza.
An instructor thinks this is a good note

Followup discussions for lingering questions and comments are listed below.

- Add new question
- Edit question
- Delete question
- Add answer
- edit answer
- add user to class
- add new class
- add tag to question
- remove tag from question
- get all the questions for a class
- get all the answers for a question

Step 2 Practice: What server endpoints will we need?

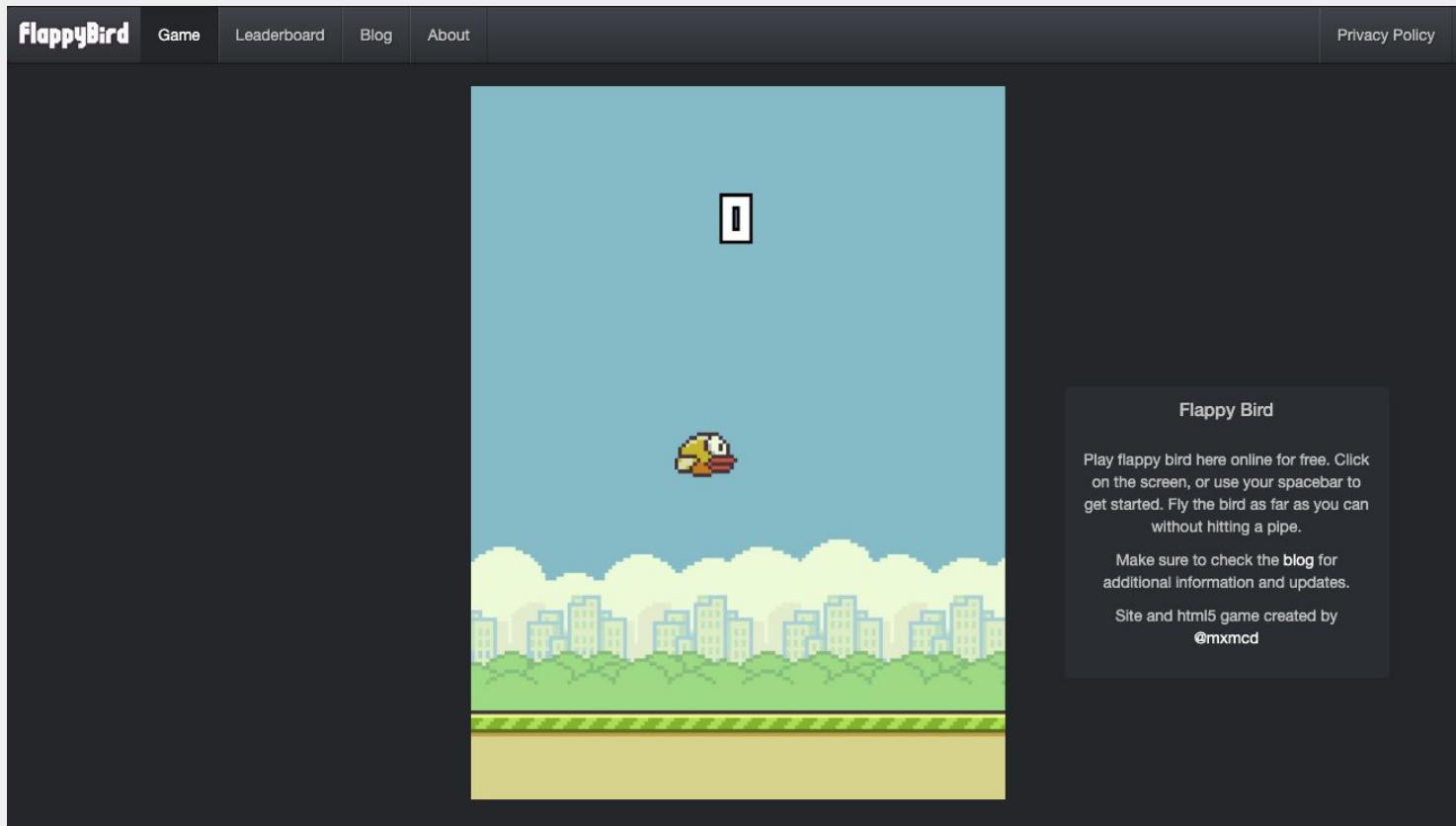
```
router.post("/addnewquestion", (req, res) => {  
});  
  
router.post("/addnewusertoclass", (req, res) => {  
})  
  
router.get("/allanswersforquestion", (req, res) => {  
})
```

Step 2 Practice: What server endpoints will we need?

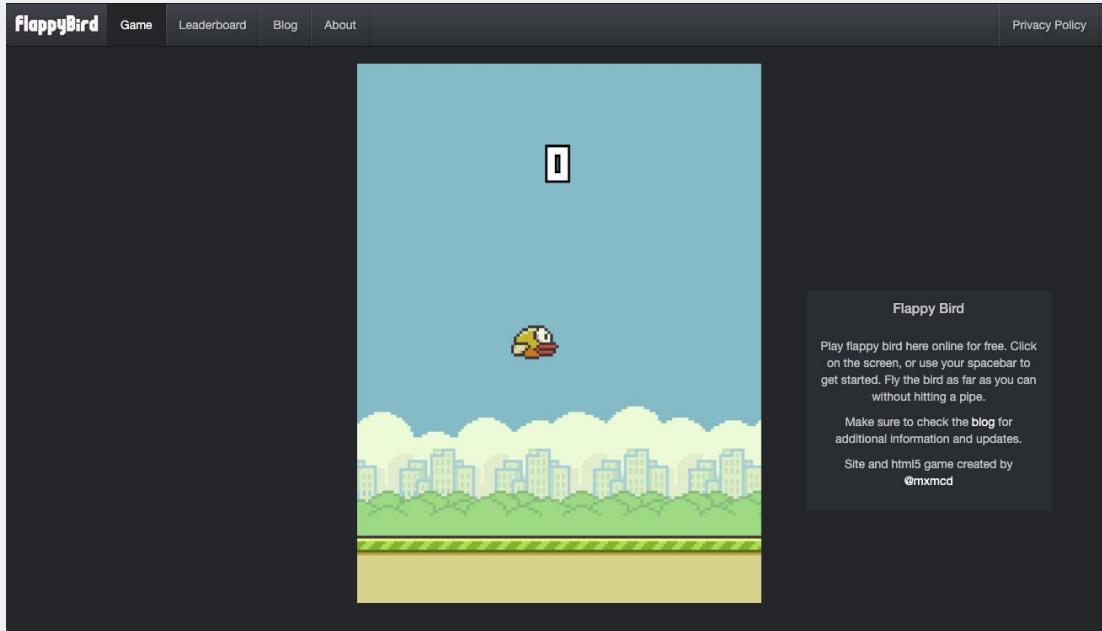
```
router.post("/addnewquestion", (req, res) => {
  const newQuestion = new Question({
    questionText: req.body.text,
    classId: req.body.classId
  })
  newQuestion.save();
});

router.get("/allanswersforquestion", (req, res) => {
  const allAnswers = Answer.find({questionId: req.query.questionId}).then((answers) => {
    res.send(answers)
  })
})
```

Step 2 Practice: What server endpoints will we need?



Step 2 Practice: What server endpoints will we need?



- Add to leaderboard
- Get top ten scores in the last hour
- Get top 40 scores in the last 24 hours

Step 2 Practice: What server endpoints will we need?

```
router.post("/addscoretoleaderboard", (req, res) => {
  const newLeaderboardEntry = new LeaderboardEntry({
    timestamp: Date.now(),
    score: req.body.score,
    name: req.body.name
  })
  newLeaderboardEntry.save();
});

router.get("/toptenscoresinlasthour", (req, res) => {
  LeaderboardEntry.find({}).then((entries) => {
    // filter entries for top 10 scores in last hour
    return filteredEntries;
  })
})
```

Step 3: What are the components?

Welcome to AceBook!

Post

Tobechi Onwuka: its no broken!

Alexander Pink: ;)

Jesus Smith: it's broken?

Eloise Zeng: hiii

Lucas Blakeslee: Hi!

Sky Loke: michael broke it

Kayla Almonte: poggg

Jesus Smith: wooo

Hannah Zhang: hellllooo

Alexander Tong: <h1>test</h1>

Christine Tang: hello!

Jeremias Figueiredo: cats>dogs

Braden Hechmer: wassup

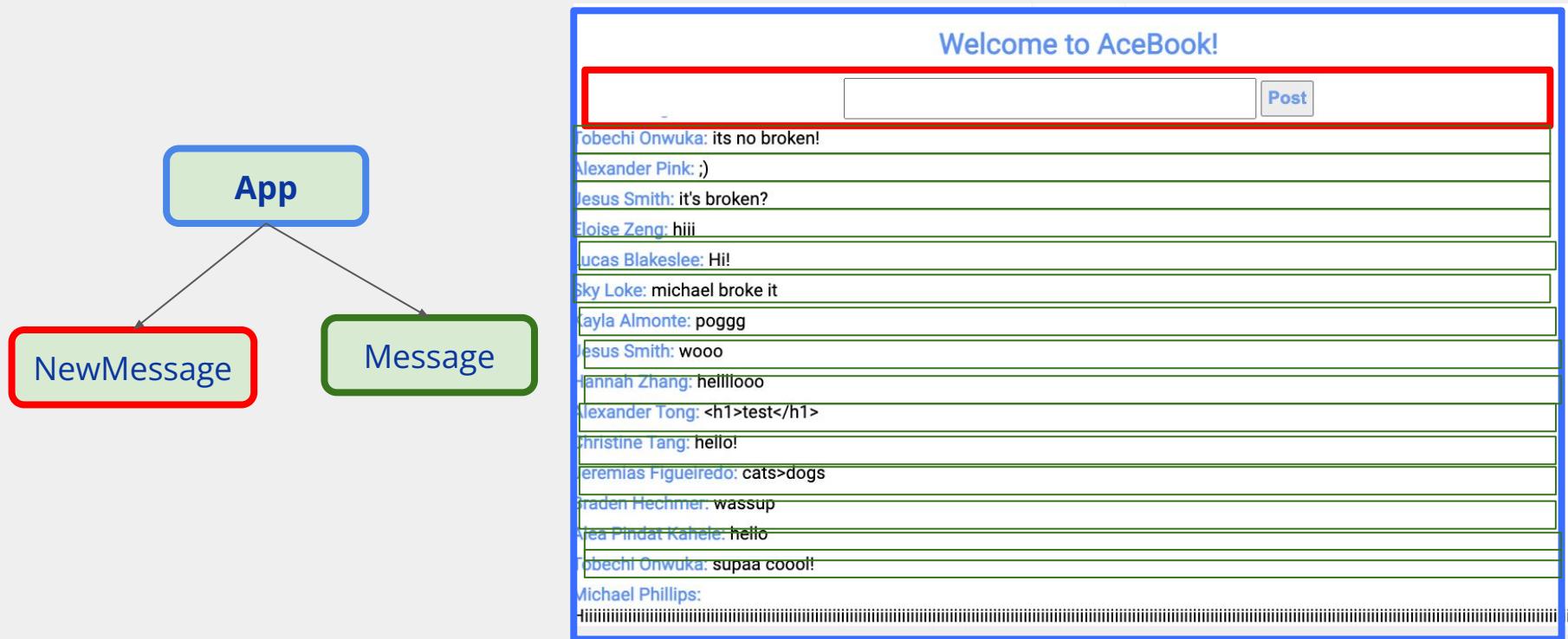
Alea Pindat Kahele: hello

Tobechi Onwuka: supaa coool!

Michael Phillips:

Hi|||||

Step 3: What are the components?



Step 3 Practice: What are the components?

The screenshot shows the Piazza interface. At the top, there's a navigation bar with 'LIVE Q&A', 'Drafts', 'Reading list', 'logistics', 'milestone0', 'milestone1', 'milestone2', 'final_submission', 'setup', 'deployment', 'frontend', 'backend', and 'more'. A user profile for 'Akshaj Kadaveru' is visible. Below the navigation, a search bar says 'Search or add a post...'. On the left, a sidebar lists 'PINNED' posts:

- Instr Project Skeleton Released** (1/9/22)
You can get started on your projects now! The project skeleton has been released. It has most of the frontend and backend.
 - Added to reading list
 - An instructor thinks this is a good note
- Instr Week 1 Overview** (1/7/22)
We put together a quick week 1 overview of most of the core content: take a look here!
 - Week 1 Overview
 - An instructor thinks this is a good note
- Instr Day 5 Recap** (1/7/22)
Hi Everyone! Congrats on finishing the first week of web.lab! 🎉 Reminder we have OH from 3-4 PM ET today on the queue
 - An instructor thinks this is a good note
- Instr 2022 Sponsor Prize Categor...** (1/4/22)
Every year, web.lab is made possible due to the generous support of our sponsors. In addition, three of our sponsors are
 - An instructor thinks this is a good note
- Instr Weblab Lecture Livestreams...** (12/31/21)
Hey guys! We've been getting lots of questions about lectures being live-streamed/recoded for Weblab and TLDR: we
 - An instructor thinks this is a good note

A section labeled 'TODAY' follows:

- Private GitHub** (1:07AM)
Hello! I apologize for the inconvenience but I recently realized that my github invitation has expired. If possible, co
 - 1 Unresolved Followup
- Set up skeleton code** (12:58AM)
When I try to set up the skeleton code I get the errors: \$ git commit -m 'Skeleton code'Author identity unknown
 - 1 Unresolved Followup
- useState** (12:28AM)

The main content area shows a note titled 'Project Skeleton Released' with 77 views. The note content is identical to the pinned post. Below it is a 'followup discussions' section with a 'Start a new followup discussion' button and a text input field.

At the bottom, there are statistics: Average Response Time: 2 min, Special Mentions: Ilya Gulko answered does weblab have merch? in 1 min. 5 hours ago, Online Now | This Week: 25 | 481. The footer includes copyright information: Copyright © 2022 Piazza Technologies, Inc. All Rights Reserved. Privacy Policy, Copyright Policy, Terms of Use, Blog, Report Bug!

Step 3 Practice: What are the components?

The screenshot shows the Piazza interface with a note highlighted by a pink border. The note is titled "Project Skeleton Released" and contains the following text:

You can get started on your projects now! The project skeleton has been released. It has most of the frontend and backend.

- Added to reading list
- An instructor thinks this is a good note

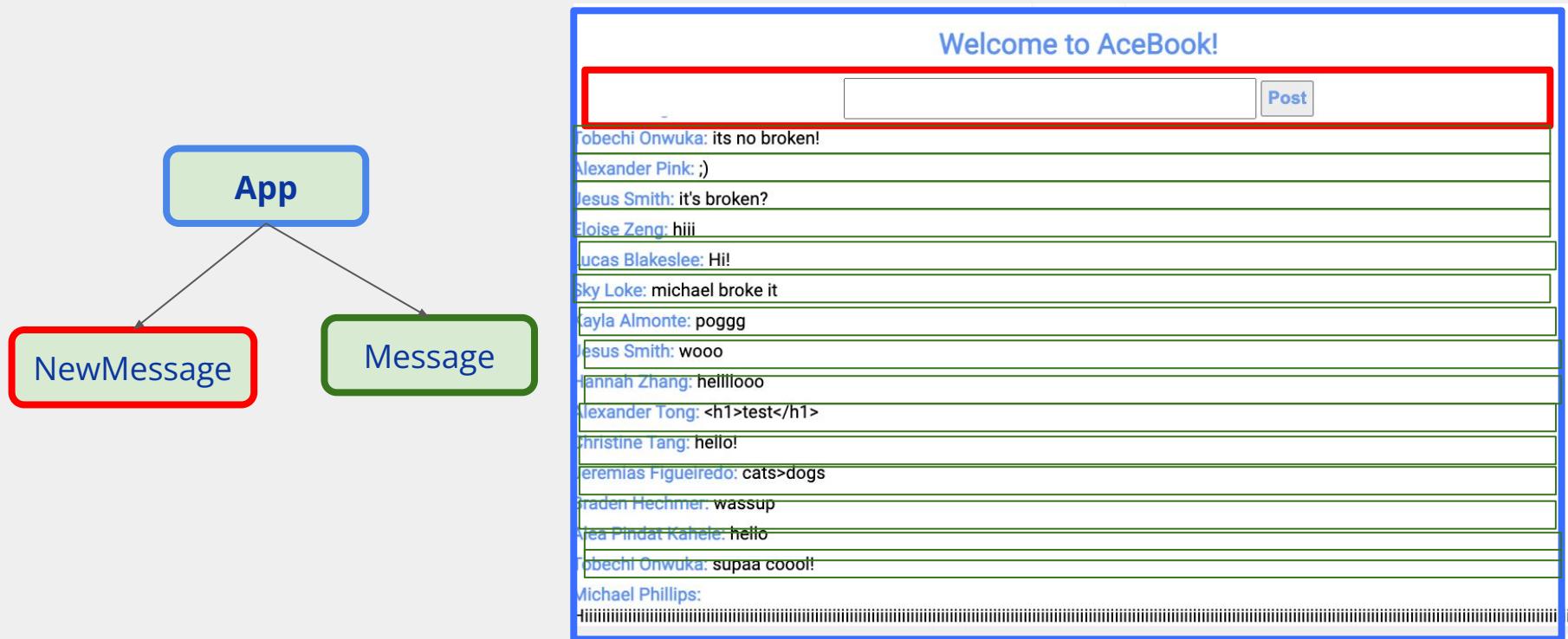
Below the note, there is a "TODAY" section containing a message from a private GitHub user:

Hello! I apologize for the inconvenience but I recently realized that my github invitation has expired. If possible, co

At the bottom of the page, there is a summary bar with the following information:

- Average Response Time: 2 min
- Special Mentions: Ilya Gulko answered does weblab have merch? in 1 min. 5 hours ago
- Online Now | This Week: 25 | 481

Step 4 : Where do we use our get/post requests? What are the state/props for our components?



App.js

```
const App = () => {
  const [messages, setMessages] = useState([]);
  useEffect(() => {
    get("/api/messages").then((messagesFromServer) => {
      setMessages(messagesFromServer);
    });
  }, []);
  return (
    <>
      <NewMessage />
      {messages.map((message) => (
        <Message text={message.text} name={message.name} />
      ))}
    </>
  );
};
```

Message.js

```
const Message = (props) => {
  return <div>{props.name + ": " + props.text}</div>;
};
```

NewMessage.js

```
const NewMessage = () => {
  const [content, setContent] = useState("");
  return (
    <>
      <input
        type="text"
        value={content}
        onChange={(event) => {
          setContent(event.target.value);
        }}
      />
      <button
        onClick={() => {
          post("/api/newmessage", { text: content });
        }}
      >
        Post
      </button>
    </>
  );
};
```

Summary: This is all the code we wrote

Database Schemas

message.js

```
const MessageSchema = new mongoose.Schema({
  name: String,
  text: String,
});
```

Backend methods

api.js

```
router.get("/allmessages", (req, res) => {
  Message.find({}).then(messagesFromDB) => {
    res.send(messagesFromDB);
  );
});

router.post("/newmessage", (req, res) => {
  const newMessage = new Message({
    name: "Anonymous",
    text: req.body.text,
  );
  newMessage.save();
});
```

React components (Frontend)

App.js

```
const App = () => {
  const [messages, setMessages] = useState([]);
  useEffect(() => {
    get("/api/messages").then((messagesFromServer) => {
      setMessages(messagesFromServer);
    });
  }, []);
  return (
    <>
      <NewMessage />
      {messages.map((message) => (
        <Message text={message.text} name={message.name} />
      ))}
    </>
  );
};
```

React components (Frontend)

NewMessage.js

Message.js

```
const Message = (props) => {
|   return <div>{props.name + ": " + props.text}</div>;
};
```

```
const NewMessage = () => {
  const [content, setContent] = useState("");
  return (
    <>
      <input
        type="text"
        value={content}
        onChange={(event) => {
          setContent(event.target.value);
        }}
      />
      <button
        onClick={() => {
          post("/api/newmessage", { text: content });
        }}
      >
        Post
      </button>
    </>
  );
};
```

Mini-workshop: Add the Date to catbook

Hints:

- Add 'date' as a field in the Story model, of type Date
- Pass in the current date `Date.now()` when adding a new story to the database
- Make sure a story's date is also passed as a prop from Feed.js to Card.js to SingleStory.js, and displayed on the story
- Test if it works by adding new stories! (Old stories won't have dates)

New Story Submit

Anonymous User
Mon Jan 10 2022 06:08:11 GMT-0500 (Eastern Standard Time)

WOW

New Comment Submit

Anonymous User
Mon Jan 10 2022 06:08:07 GMT-0500 (Eastern Standard Time)

test

New Comment Submit

```
git fetch  
git reset --hard  
git checkout w6-complete
```

Solution

```
const StorySchema = new mongoose.Schema({  
  creator_name: String,  
  content: String,  
  date: Date,  
});
```

```
router.post("/story", (req, res) => {  
  const newStory = new Story({  
    creator_name: MY_NAME,  
    content: req.body.content,  
    date: Date.now(),  
  });  
  
  newStory.save().then((story) => res.send(story));  
});
```

```
git reset --hard  
git checkout w6-date
```

```
const SingleStory = (props) => {  
  return (  
    <div className="Card-story">  
      <span className="u-bold">{props.creator_name}</span>  
      <div>{new Date(props.date).toString()}</div>  
      <p className="Card-storyContent">{props.content}</p>  
    </div>  
  );  
};
```