Check-In Code: woohackschool!

Hack School 5: APIs Part 2 & Finishing Touches

ACM at UCSD

woohackschool!

Check-In
acmucsd.com

Check-In Code: woohackschool!

ACM at UCSD

acmurl.com/hackschool5-checkin

Check-In Code: woohackschool!

ACM at UCSD

today's agenda

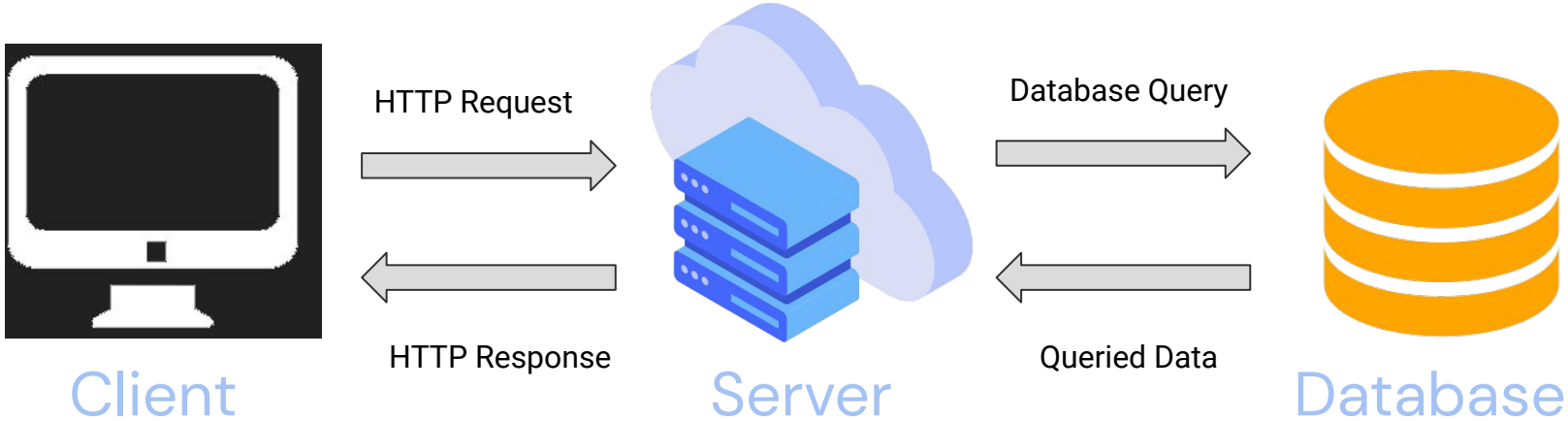
1. API
2. Connect Frontend & Backend
3. LinkedIn, Resume, GitHub
4. Social Destress

Check-In Code: woohackschool!

ACM at UCSD

Check-In Code: woohackschool!

Our Purchase Tracker Architecture



ACM at UCSD

Check-In Code: woohackschool!

API calls in Frontend

These allow the frontend to interact with the backend... Exciting !

- We can make **GET** and **POST** requests to our own local server to create and read Purchases
  - **axios** - Node.js library that allows us to make API calls programmatically
- We will create a **separate JavaScript file** as a utility to make API calls
  - client/src/API.js
- We will use this utility in our ViewPurchase page to fetch purchases, and in our CreatePurchase page to create purchase

Check-In Code: woohackschool!

GET request

- Grabs data from the server, and returns a **JSON object**
- Takes **one parameter**, which is the **link** that we are getting data from
- Wrapped in **an outer function**
- Need to await the function call since a Promise<Response> is returned from axios

```
// API.js
const serverURL = "apiName.someAPI.com";
export default {
      getMemeSongs: function() {
      return axios.get(`${serverURL}/api/meme-songs`);
      }
}

// jsx file
import API from './API.js';
const memeSongs = await API.getMemeSongs();
```

ACM at UCSD

Check-In Code: woohackschool!

GET request

```
// API.js
const serverURL = "apiName.someAPI.com";
const API = {
    getMemeSongs: function() {
        return axios.get(`${serverURL}/api/meme-songs`);
    }
}
export default API;

// jsx file
import API from './API.js';
const memeSongs = await API.getMemeSongs();
```

Specify our server location here, as all of our requests will need to use it!

ACM at UCSD

Check-In Code: woohackschool!

GET request

```
// API.js
const serverURL = "apiName.someAPI.com";
const API = {
    getMemeSongs: function() {
        return axios.get(`${serverURL}/api/meme-songs`);
    }
}
export default API;

// jsx file
import API from './API.js';
const memeSongs = await API.getMemeSongs();
```

Define our API object with functions

ACM at UCSD

Check-In Code: woohackschool!

GET request

```
// API.js
const serverURL = "apiName.someAPI.com";
const API = {
    getMemeSongs: function() {
        return axios.get(`${serverURL}/api/meme-songs`);
    }
}
export default API;

// jsx file
import API from './API.js';
const memeSongs = await API.getMemeSongs();
```

We need an outer action to wrap our requests for access outside this file. They can also be used to pass in necessary parameters

ACM at UCSD

Check-In Code: woohackschool!

GET request

```
// API.js
const serverURL = "apiName.someAPI.com";
const API = {
    getMemeSongs: function() {
        return axios.get(`${serverURL}/api/meme-songs`);
    }
}
export default API;

// jsx file
import API from './API.js';
const memeSongs = await API.getMemeSongs();
```

The actual code that sends the GET request. Use the data route defined in our backend!

ACM at UCSD

Check-In Code: woohackschool!

GET request

```
// API.js
const serverURL = "apiName.someAPI.com";
const API = {
    getMemeSongs: function() {
        return axios.get(`${serverURL}/api/meme-songs`);
    }
}
export default API;

// jsx file
import API from './API.js';
const memeSongs = await API.getMemeSongs();
```

Import API.js, and call the defined functions here!

ACM at UCSD

Check-In Code: woohackschool!

POST Requests

- Posts data to the server, and returns a **JavaScript promise**
- Takes **one parameter**, which is a **JSON object** that configures our POST request
- Wrapped in **an outer function** that **passes down a JavaScript object** containing the data that we want to send to the server

Check-In Code: woohackschool!

POST Requests

```
const serverURL = "apiName.someAPI.com";
const API = {
     createMemeSong: function(payload) {
        return axios.post(`${serverURL}/api/memes`, payload);
     }
}
export default API;
// jsx file
import API from './API.js';
const song4 = {
     title: "Welcome to the Black Parade",
     ...
}
await API.createMemeSong(song4);
```

Pass in the payload to our function and call axios.post() on our API url with our payload

ACM at UCSD

Check-In Code: woohackschool!

POST Requests

```
const serverURL = "apiName.someAPI.com";
const API = {
    createMemeSong: function(payload) {
        return axios.post(`${serverURL}/api/memes`, payload);
    }
}
export default API;
// jsx file
import API from './API.js';
const song4 = {
    title: "Welcome to the Black Parade",
    ...
}
await API.createMemeSong(song4);
```

Define our object and call API to post it

ACM at UCSD

Live Demo

Demo to connect Frontend &
Backend for our Finance Buddy site

Check-In Code: woohackschool!

ACM at UCSD

Your Turn!

- In API.js write the function for createPurchase similar to the getPurchase one we wrote

Check-In Code: woohackschool!

Your Turn!

- Make the description, location, date, cost, and method handle changes to the form

Refer to how we handled change to the form for name and make use of our handleChange function!

Check-In Code: woohackschool!

ACM at UCSD

Potential Improvements

What about some stretch goals?

- We have implemented our MVP! Here's some stretch functionalities:
- Filtering by type - On the View Purchases page, have a select dropdown that, when its value changes, filters the list of Purchases by that specific type
  - Hint 1 : The onChange event handles value changes
  - Hint 2: The filter() function will be extremely helpful! (documentation)
- Only allow one exact same entry
  - Hint 1: Add a condition so you only send the information to the server if it's not the same as existing data in server
  - Look at online tutorials

Check-In Code: woohackschool!

ACM at UCSD

And even more general goals?

- Implement other API Routes
  - Updating a purchase, deleting a purchase?
- Build more UIs
  - Pie-chart style breakdown of existing purchases filtered however you want
- Make it secure
  - Authentication
- Make it live??
  - Deploy it
    - Rip Heroku

Check-In Code: woohackschool!

ACM at UCSD

Check-In Code: woohackschool!

Resume, LinkedIn, GitHub

ACM at UCSD

Push your code to GitHub!

- We want our code shown off on our GitHub! This is why we forked it in workshop 1
- Open a terminal in the project directory, and enter the following commands:

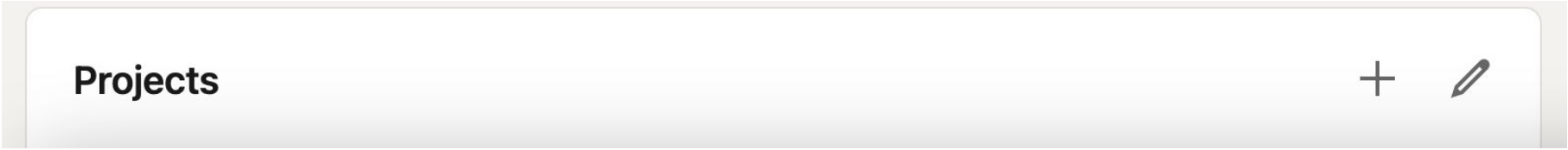
```
$ git add -A

$ git commit -m "Finish project code"

$ git push origin main
```

Check-In Code: woohackschool!

ACM at UCSD

Adding our Project to LinkedIn/Resume

- Let's utilize the LinkedIn Projects section!



- Resume tips
  - Brandon's workshops next quarter!

Check-In Code: woohackschool!

ACM at UCSD

Discord Bot | *TypeScript, Node.js, MongoDB, C#, ASP.NET* May 2018 – May 2020

- Led a team of 3 developers to develop Discord chat bot using **TypeScript** and **Node.js** for GAME group creation
- Handles verification, group management, and moderation for **60,000+ players** across **20+ chat groups**
- Used **MongoDB** to store user data, group settings, other important configurations, improving efficiency by 20%
- Created a **C# ASP.NET** application to scrape data from player statistics website for GAME

...and that's a wrap!

Thank you everyone for sticking with us for **6 workshops!**

Check-In Code: woohackschool!

ACM at UCSD

Future Hack Workshops (Tentative):

Khushi: Intro to Python Series

- Week 3-5 from 5-7pm

Nikhil: APIs and SQL

https://acmurl.com/workshop-topics

Check-In Code: woohackschool!

ACM at UCSD