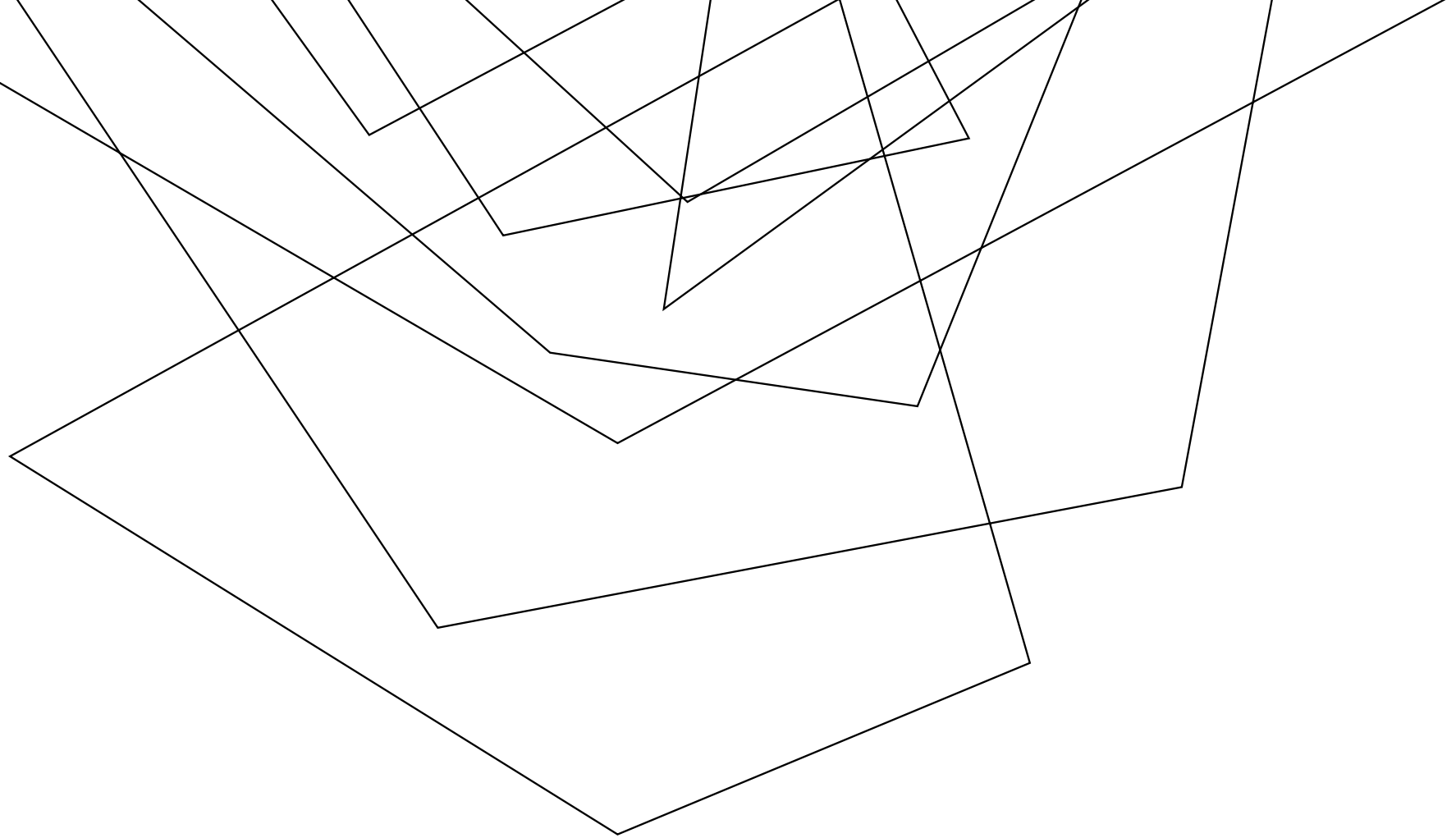
EXERCISE

WEB SECURITY REVIEW

What is the same origin policy?

ADMINISTRIVIA AND ANNOUNCEMENTS



WEB CLIENT SECURITY

EECS 677: Software Security Evaluation

Drew Davidson

LAST TIME

REVIEW: WEB RESOURCES

Basic web endpoint

- Handles user input
- Sanitizes input

```
import express from "express";
import validator from "validator";
const app = express();
app.get("/page", (req, res) => {
  let v = req.query.v ? req.query.v : "world";
   v = validator.escape(v);
  res.send(`<h1>Hello ${v}!</h1>`);
});
app.listen(3001);
```

TODAY

WEB FRONTENDS

Frontend Security

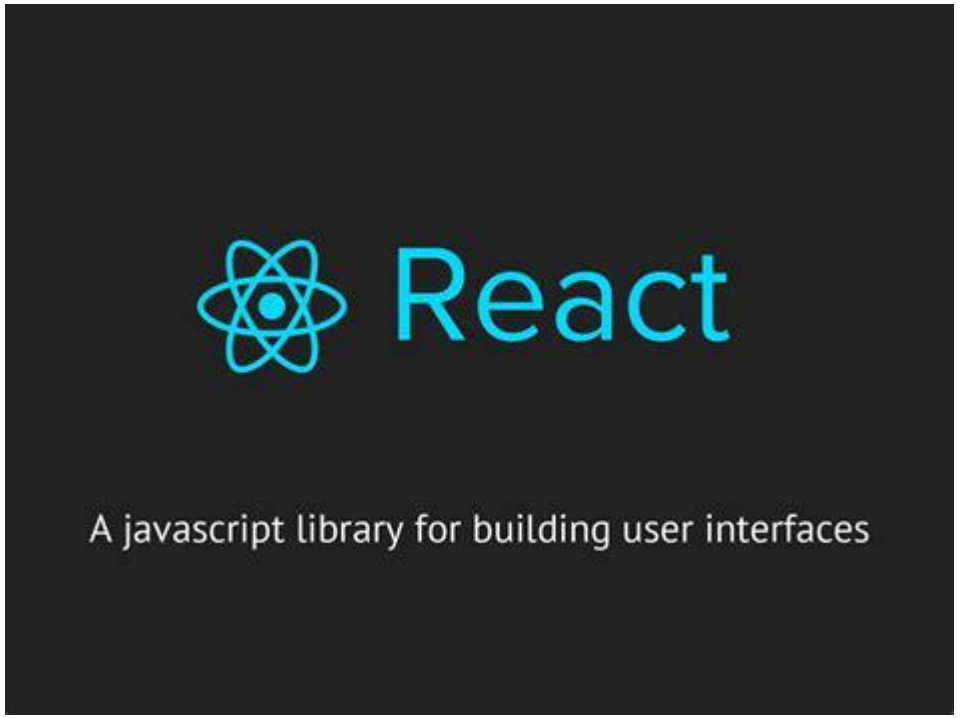
- Elaborate the UI and create more of the web “stack”
- Create a frontend with react
- Secure a third-party resource
- Interact with the data endpoint

REACT.JS AS MODEL FRONTEND

WEB CLIENT SECURITY

Basic runtime workflow

- Present a minimal html “shell”
- Inject components from the Javascript library



INTRO TO REACT.JS

WEB FRONTENDS

Basic runtime workflow

- Present a minimal html “shell”
- Inject components from the Javascript library

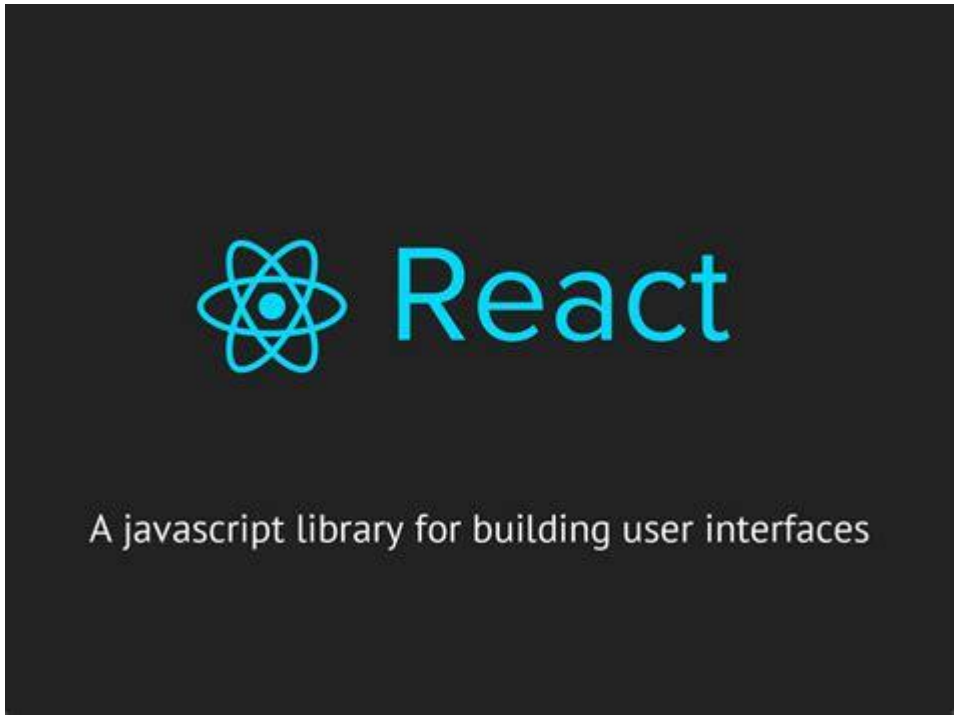
Create a project

npm init react-app frontend

cd frontend

npm start

Note: If using WSL, don’t put this directory in the windows filesystem (it’s very slow!)



INTRO TO REACT.JS

WEB FRONTENDS

Basic runtime workflow

- Present a minimal html “shell”
- Inject components from the Javascript library

Create a project

npm init react-app frontend

cd frontend

npm start

Note: If using WSL, don’t put this directory in the windows filesystem (it’s very slow!)

Project structure

public/

Where the html static shell webpage being delivered lives

INTRO TO REACT.JS

WEB FRONTENDS

Project structure

public/

Where the html static shell webpage being delivered lives

We could actually add additional scripts to this page...

public/index.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <link rel="icon" href="%PUBLIC_URL%/favicon.ico" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <meta name="theme-color" content="#000000" />
    <meta name="description‟ content="Web site created using react‟ />
    <link rel="apple-touch-icon" href="%PUBLIC_URL%/logo192.png" />
    <link rel="manifest" href="%PUBLIC_URL%/manifest.json" />
    <title>React App</title>
    <script defer src="/static/js/bundle.js"></script>
  </head>
  <body>
    <noscript>You need to enable JavaScript to run this app.</noscript>
    <div id="root"></div>
  </body>
</html>
```

INTRO TO REACT.JS

WEB FRONTENDS

We could actually add additional scripts to this page...

Project structure

public/

Where the html static shell webpage being delivered lives

public/index.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <link rel="icon" href="%PUBLIC_URL%/favicon.ico" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <meta name="theme-color" content="#000000" />
    <meta name="description" content="Web site created using react" />
    <link rel="apple-touch-icon" href="%PUBLIC_URL%/logo192.png" />
    <link rel="manifest" href="%PUBLIC_URL%/manifest.json" />
    <script src="https://drew.davidson.cool/rainbow.js" async></script>
    <title>React App</title>
    <script defer src="/static/js/bundle.js"></script>
  </head>
  <body>
    <noscript>You need to enable JavaScript to run this app.</noscript>
    <div id="root"></div>
  </body>
</html>
```

INTRO TO REACT.JS

WEB FRONTENDS

Project structure

public/

Where the html static shell webpage being delivered lives

src/

Where the injected javascript source files live

src/App.js

```
import './App.css';
function App() {
  return (
    <div>
        <span>Username</span>
        <input type="text" />
        <br />
        <span>Password</span>
        <input type="text" />
        <br />
        <input type="button" value="Create Account" />
    </div>
  );
}
export default App;
```

A SIMPLE LOGIN SCREEN

WEB FRONTENDS

Project structure

public/

Where the html static shell webpage being delivered lives

src/

Where the injected javascript source files live

src/App.js

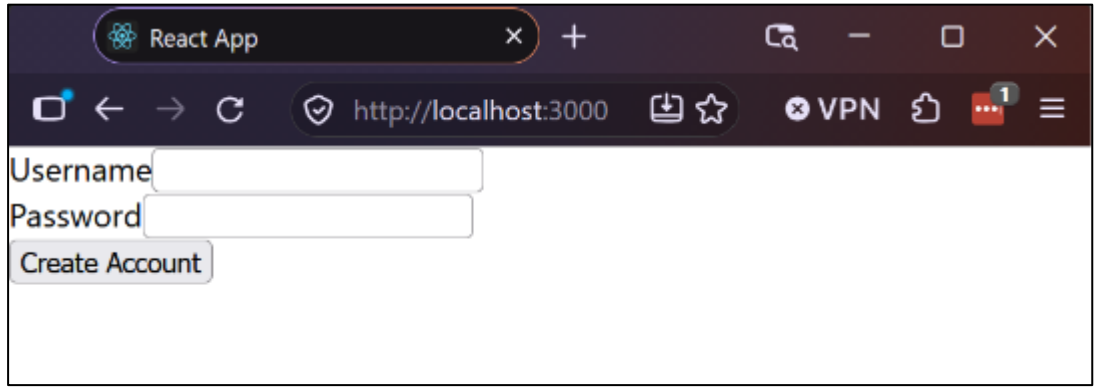
```
import './App.css';
function App() {
  return (
    <div>
        <span>Username</span>
        <input type="text" />
        <br />
        <span>Password</span>
        <input type="text" />
        <br />
        <input type="button" value="Create Account" />
    </div>
  );
}
export default App;
```

A SIMPLE LOGIN SCREEN

WEB FRONTENDS

src/App.js

```
import './App.css';
function App() {
  return (
    <div>
        <span>Username</span>
        <input type="text" />
        <br />
        <span>Password</span>
        <input type="text" />
        <br />
        <input type="button" value="Create Account" />
    </div>
  );
}
export default App;
```

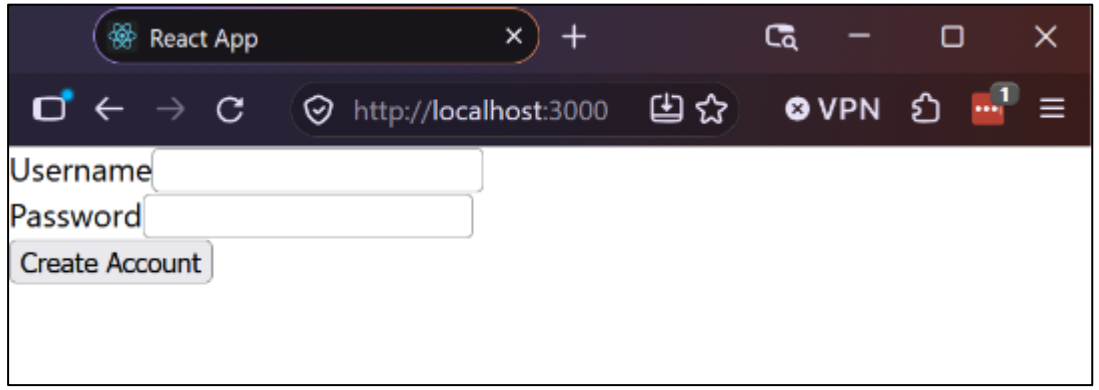


A SIMPLE LOGIN SCREEN

WEB FRONTENDS

src/App.js

```
import './App.css';
import { useState } from 'react'
function App() {
  return (
    <div>
        <span>Username</span>
        <input type="text" />
        <br />
        <span>Password</span>
        <input type="text" />
        <br />
        <input type="button" value="Create Account" />
    </div>
  );
}
export default App;
```

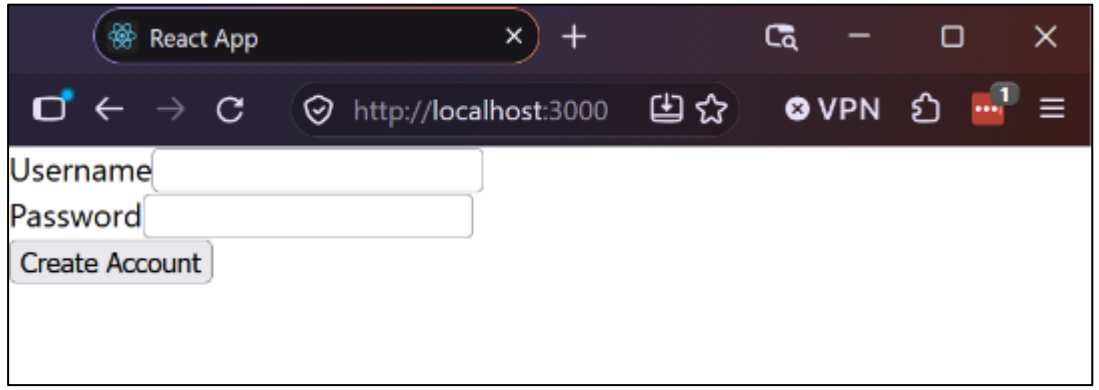


A SIMPLE LOGIN SCREEN

WEB FRONTENDS

src/App.js

```
import './App.css';
import { useState } from 'react'
function App() {
  const [uname, setUname] = useState("");
  const [pw, setPW] = useState("");
  return (
    <div>
        <span>Username</span>
        <input type="text" />
        <br />
        <span>Password</span>
        <input type="text" />
        <br />
        <input type="button" value="Create Account" />
    </div>
  );
}
export default App;
```

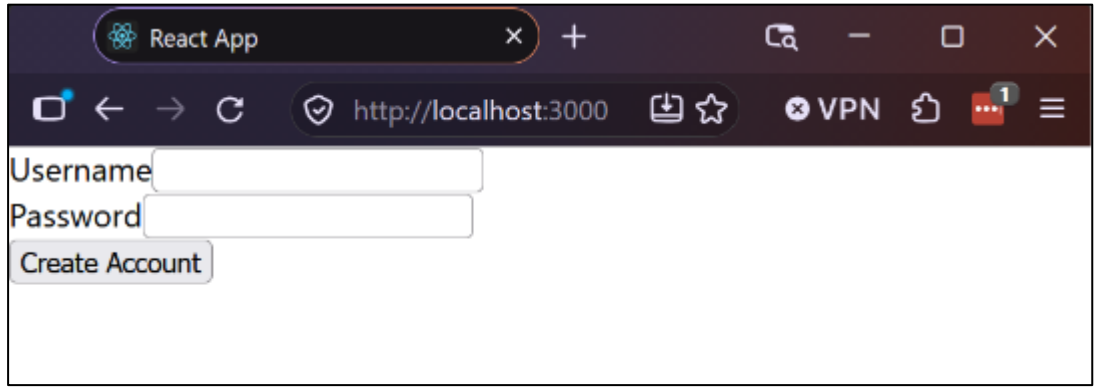


A SIMPLE LOGIN SCREEN

WEB FRONTENDS

src/App.js

```
import './App.css';
import { useState } from 'react'
function App() {
  const [uname, setUname] = useState("");
  const [pw, setPW] = useState("");
  return (
    <div>
        <span>Username</span>
        <input type="text" onChange={(evt) => {setUname(evt.target.value)}} />
        <br />
        <span>Password</span>
        <input type="text" />
        <br />
        <input type="button" value="Create Account" />
    </div>
  );
}
export default App;
```

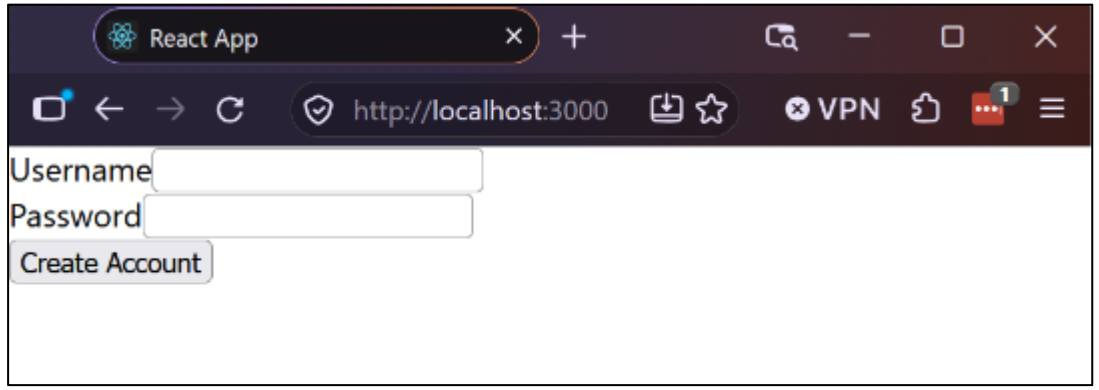


A SIMPLE LOGIN SCREEN

WEB FRONTENDS

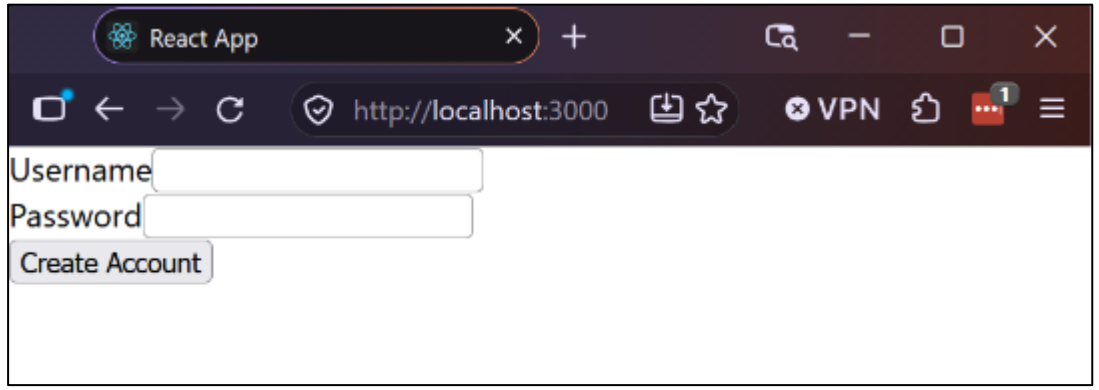
src/App.js

```
import './App.css';
import { useState } from 'react'
function App() {
  const [uname, setUname] = useState("");
  const [pw, setPW] = useState("");
  return (
    <div>
        <span>Username</span>
        <input type="text"  onChange={(evt) => {setUname(evt.target.value)}} />
        <br />
        <span>Password</span>
        <input type="text" onChange={(evt) => {setPW(evt.target.value)}}/>
        <br />
        <input type="button" value="Create Account" />
    </div>
  );
}
export default App;
```



A SIMPLE LOGIN SCREEN

WEB FRONTENDS



src/App.js

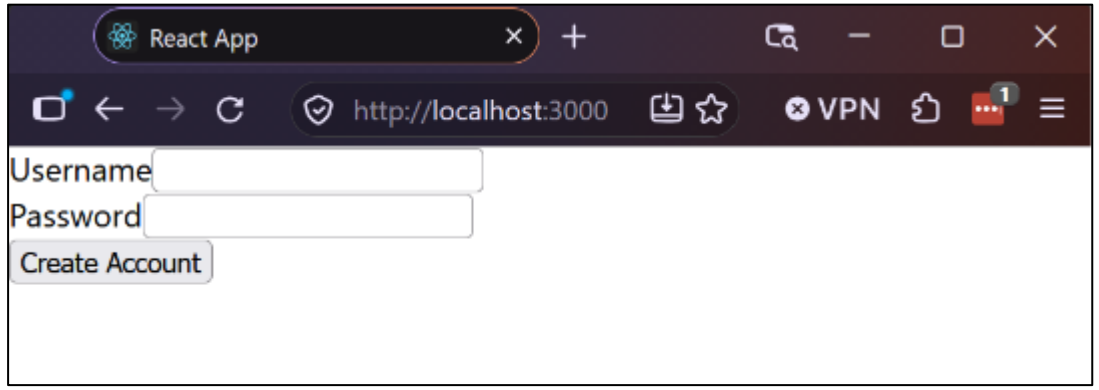
```
import './App.css';
import { useState } from 'react'
function App() {
  const [uname, setUname] = useState("");
  const [pw, setPW] = useState("");
  return (
    <div>
        <span>Username</span>
        <input type="text"  onChange={(evt) => {setUname(evt.target.value)}} />
        <br />
        <span>Password</span>
        <input type="text" onChange={(evt) => {setPW(evt.target.value)}}/>
        <br />
        <input type="button" value="Create Account" onClick={submit}/>
    </div>
  );
}
export default App;
```

A SIMPLE LOGIN SCREEN

WEB FRONTENDS

src/App.js

```
import './App.css';
import { useState } from 'react'
import axios  from 'axios'
function App() {
  const [uname, setUname] = useState("");
  const [pw, setPW] = useState("");
  return (
    <div>
        <span>Username</span>
        <input type="text"  onChange={(evt) => {setUname(evt.target.value)}} />
        <br />
        <span>Password</span>
        <input type="text" onChange={(evt) => {setPW(evt.target.value)}}/>
        <br />
        <input type="button" value="Create Account"  onClick={submit} />
    </div>
  );
}
export default App;
```

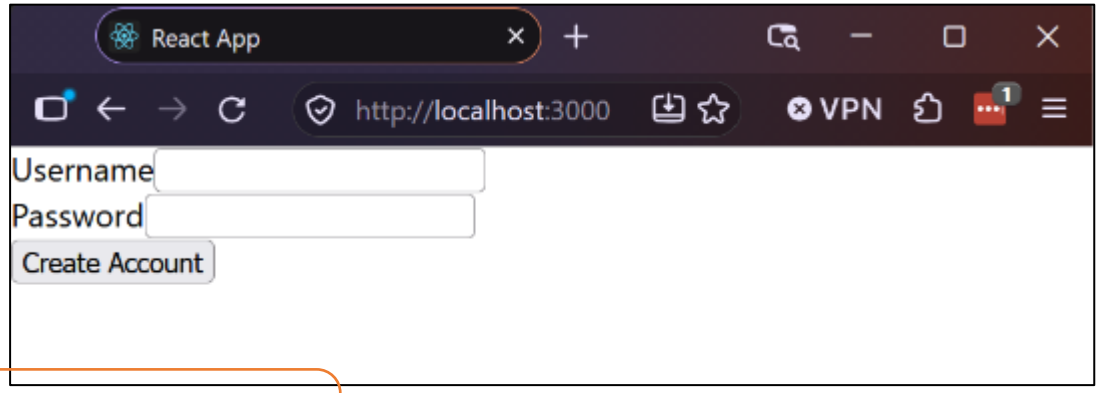


A SIMPLE LOGIN SCREEN

WEB FRONTENDS

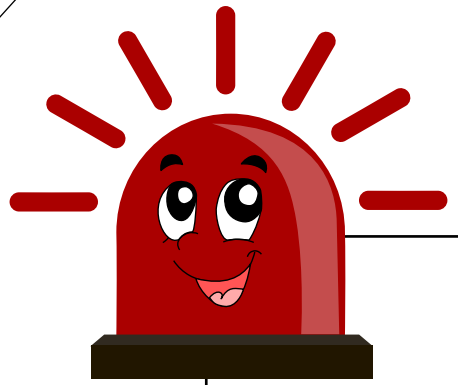
src/App.js

```
import './App.css';
import { useState } from 'react'
import axios  from 'axios'
function App() {
  const [uname, setUname] = useState("");
  const [pw, setPW] = useState("");
  async function submit(){
     let request = {user:uname, pw:pw}
     axios.get("http://localhost:3001/register", request)
  }
  return (
    <div>
        <span>Username</span>
        <input type="text"  onChange={(evt) => {setUname(evt.target.value)}} />
        <br />
        <span>Password</span>
        <input type="text" onChange={(evt) => {setPW(evt.target.value)}}/>
        <br />
        <input type="button" value="Create Account"  onClick={submit}/>
    </div>
  );
}
export default App;
```



PROBLEMS

WEB CLIENT SECURITY



The first-party web developer could do a lot more to protect the security of a client in this example code

PROTECTING RESOURCES

WEB PROTECTIONS

Recall our HTML shell

It includes a 3rd-party script

public/index.html

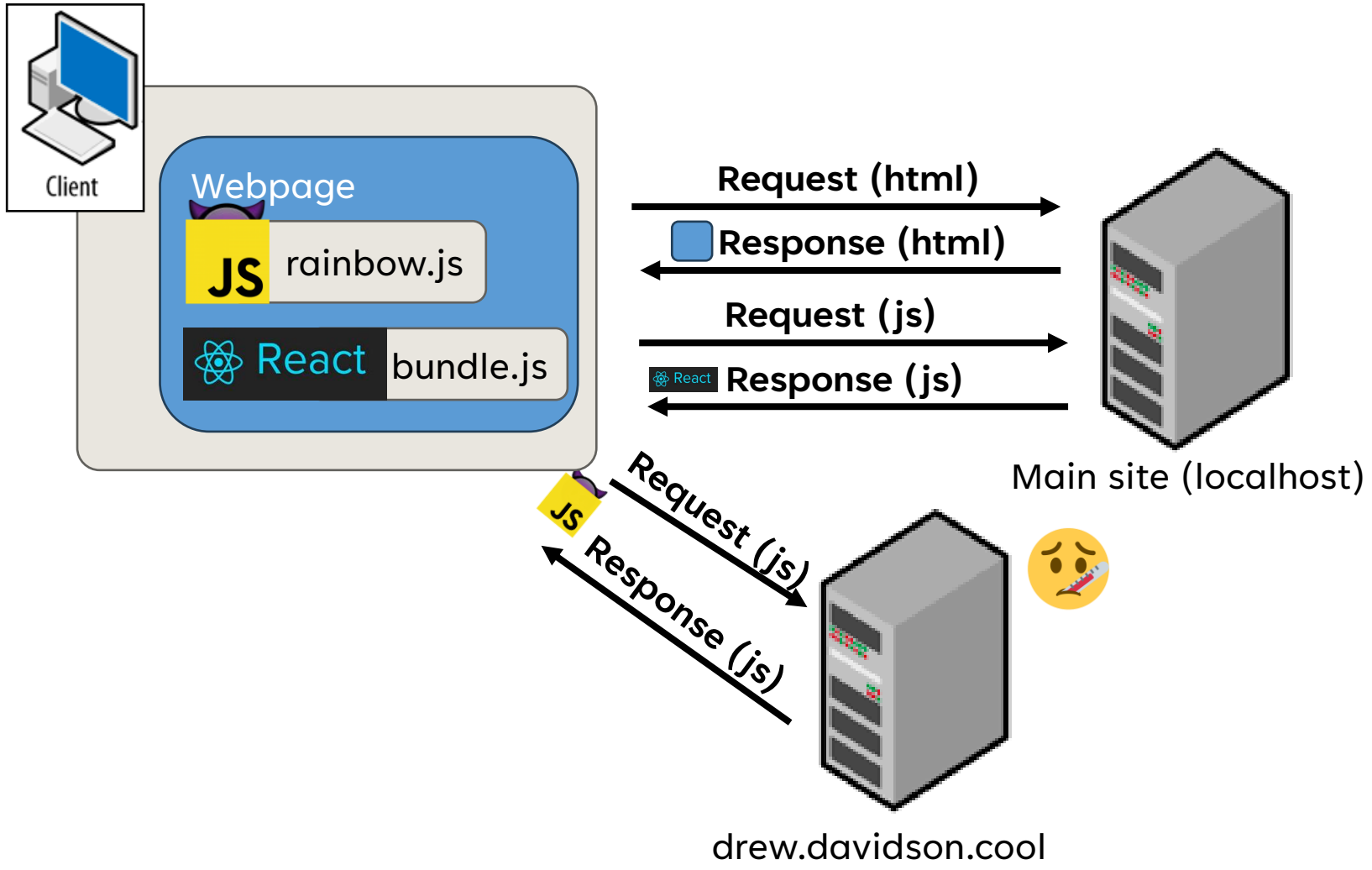
```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <link rel="icon" href="%PUBLIC_URL%/favicon.ico" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <meta name="theme-color" content="#000000" />
    <meta name="description" content="Web site created using react" />
    <link rel="apple-touch-icon" href="%PUBLIC_URL%/logo192.png" />
    <link rel="manifest" href="%PUBLIC_URL%/manifest.json" />
    <script src="https://drew.davidson.cool/rainbow.js" async></script>
    <title>React App</title>
    <script defer src="/static/js/bundle.js"></script>
  </head>
  <body>
    <noscript>You need to enable JavaScript to run this app.</noscript>
    <div id="root"></div>
  </body>
</html>
```

PROTECTING RESOURCES

WEB PROTECTIONS

Recall our HTML shell

It includes a 3rd-party script



SUB-RESOURCE INTEGRITY

WEB PROTECTIONS

Ensuring script integrity

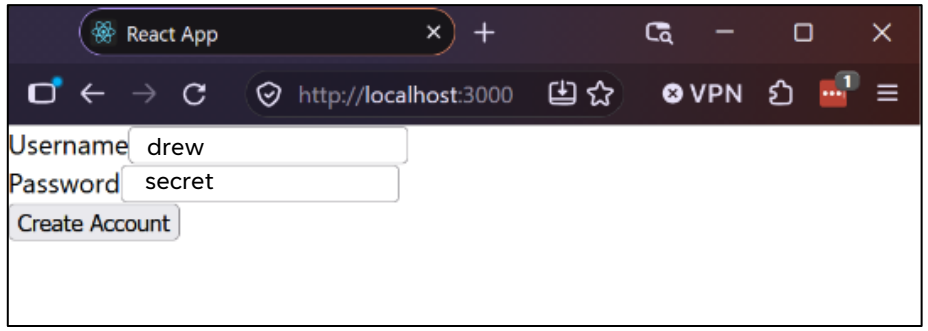
```
<script
   src="https://drew.davidson.cool/rainbow.js"
   integrity="sha384-oqVuAfXRKap7fdgcCY5uykM6+R9GqQ8K/uxy9rx7HNQlGYl1kPzQho1wx4JwY8wC"
   crossorigin="anonymous"></script>
```

Generating the hash

```
cat rainbow.js | openssl dgst -sha384 -binary | openssl base64 -A
```

PASSWORD ENTRY

WEB CLIENT SECURITY



Our password is a plain text box!

We should indicate that it is sensitive data

src/App.js

```
import './App.css';
import { useState } from 'react'
import axios  from 'axios'

function App() {
  const [uname, setUname] = useState("");
  const [pw, setPW] = useState("");

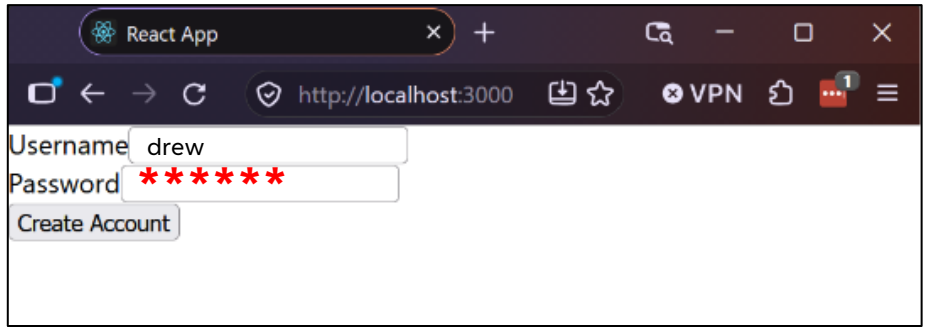
  async function submit(){
     let request = {params:{user:uname, pw:pw}}
     axios.get("http://localhost:3001/register", request)

  }
  return (
    <div>
        <span>Username</span>
        <input type="text"  onChange={(evt) => {setUname(evt.target.value)}} />
        <br />
        <span>Password</span>
        <input type="text" onChange={(evt) => {setPW(evt.target.value)}}/>
        <br />
        <input type="button" value="Create Account"  onClick={submit}/>
    </div>
  );
}
export default App;
```

password

PASSWORD ENTRY

WEB CLIENT SECURITY



Our password is a plain text box!

We should indicate that it is sensitive data

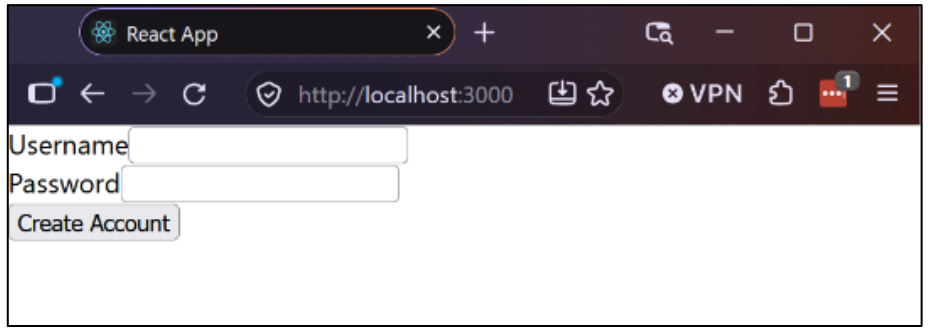
src/App.js

```
import './App.css';
import { useState } from 'react'
import axios  from 'axios'
function App() {
  const [uname, setUname] = useState("");
  const [pw, setPW] = useState("");
  async function submit(){
     let request = {params:{user:uname, pw:pw}}
     axios.get("http://localhost:3001/register", request)
  }
  return (
    <div>
        <span>Username</span>
        <input type="text"  onChange={(evt) => {setUname(evt.target.value)}} />
        <br />
        <span>Password</span>
        <input type="text" onChange={(evt) => {setPW(evt.target.value)}}/>
        <br />
        <input type="button" value="Create Account"  onClick={submit}/>
    </div>
  );
}
export default App;
```

password

PASSWORD ENTRY

WEB CLIENT SECURITY

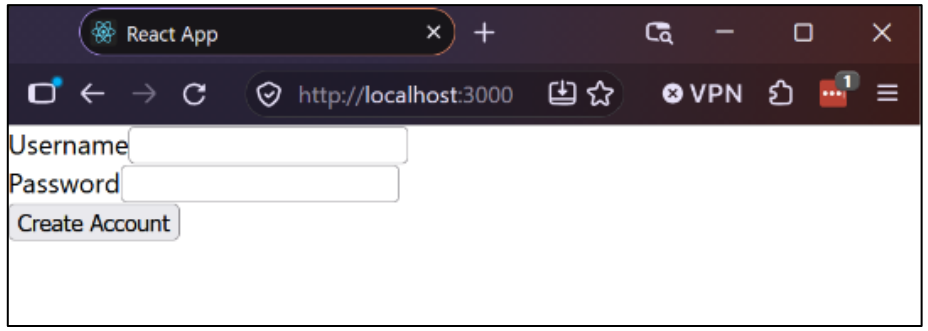


src/App.js

```
import './App.css';
import { useState } from 'react'
import axios  from 'axios'
function App() {
  const [uname, setUname] = useState("");
  const [pw, setPW] = useState("");
  async function submit(){
     let request = {params:{user:uname, pw:pw}}
     axios.get("http://localhost:3001/register", request)
  }
  return (
    <div>
        <span>Username</span>
        <input type="text"  onChange={(evt) => {setUname(evt.target.value)}} />
        <br />
        <span>Password</span>
        <input type="password" onChange={(evt) => {setPW(evt.target.value)}}/>
        <br />
        <input type="button" value="Create Account"  onClick={submit}/>
    </div>
  );
}
export default App;
```

POST VS GET REQUESTS

WEB CLIENT SECURITY



src/App.js

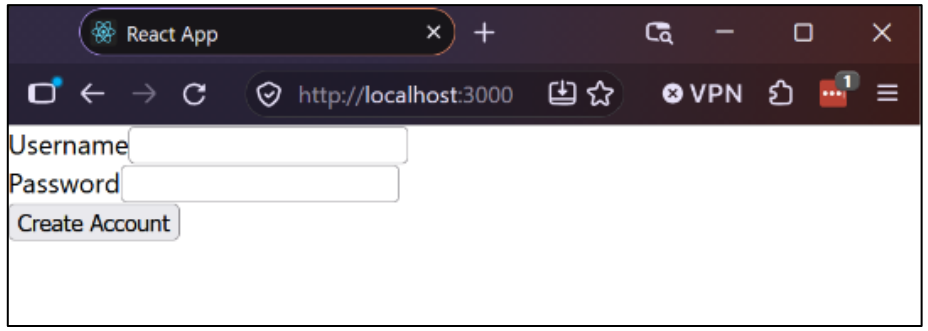
```
import './App.css';
import { useState } from 'react'
import axios  from 'axios'
function App() {
  const [uname, setUname] = useState("");
  const [pw, setPW] = useState("");
  async function submit(){
     let request = {params:{user:uname, pw:pw}}
     axios.get("http://localhost:3001/register", request)
  }
  return (
    <div>
        <span>Username</span>
        <input type="text"  onChange={(evt) => {setUname(evt.target.value)}} />
        <br />
        <span>Password</span>
        <input type="password" onChange={(evt) => {setPW(evt.target.value)}}/>
        <br />
        <input type="button" value="Create Account"  onClick={submit}/>
    </div>
  );
}
export default App;
```

POST VS GET REQUESTS

WEB CLIENT SECURITY

GET is the wrong way to pass passwords

We should make it a POST request



src/App.js

```
import './App.css';
import { useState } from 'react'
import axios  from 'axios'

function App() {
  const [uname, setUname] = useState("");
  const [pw, setPW] = useState("");
  async function submit(){
     let request = {params:{user:uname, pw:pw}}
     axios.get("http://localhost:3001/register", request)
  }
  return (
    <div>
        <span>Username</span>
        <input type="text"  onChange={(evt) => {setUname(evt.target.value)}} />
        <br />
        <span>Password</span>
        <input type="password" onChange={(evt) => {setPW(evt.target.value)}}/>
        <br />
        <input type="button" value="Create Account"  onClick={submit}/>
    </div>
  );
}
export default App;
```

(slightly different argument format)

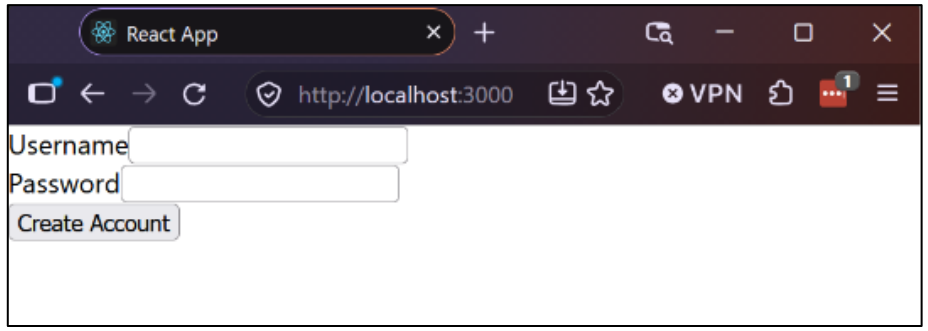
post

POST VS GET REQUESTS

WEB CLIENT SECURITY

GET is the wrong way to pass passwords

We should make it a POST request

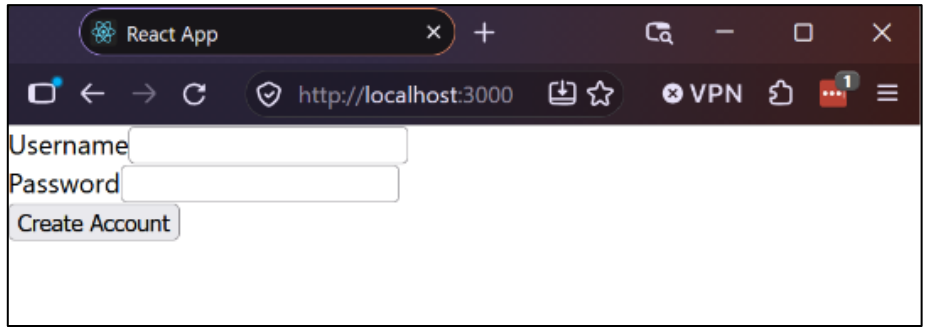


src/App.js

```
import './App.css';
import { useState } from 'react'
import axios  from 'axios'
function App() {
  const [uname, setUname] = useState("");
  const [pw, setPW] = useState("");
  async function submit(){
     let request = {user:uname, pw:pw}
     axios.post("http://localhost:3001/register", request)
  }
  return (
    <div>
        <span>Username</span>
        <input type="text"  onChange={(evt) => {setUname(evt.target.value)}} />
        <br />
        <span>Password</span>
        <input type="password" onChange={(evt) => {setPW(evt.target.value)}}/>
        <br />
        <input type="button" value="Create Account"  onClick={submit}/>
    </div>
  );
}
export default App;
```

HANDLING REQUEST FAILURE

WEB CLIENT SECURITY



We aren't providing any error handling

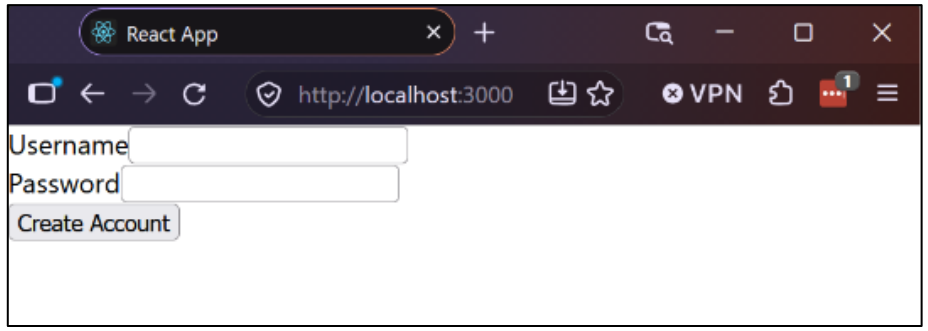
We should fail gracefully (and probably indicate success while we're at it)

src/App.js

```
import './App.css';
import { useState } from 'react'
import axios  from 'axios'
function App() {
  const [uname, setUname] = useState("");
  const [pw, setPW] = useState("");
  async function submit(){
     let request = {user:uname, pw:pw}
     axios.post("http://localhost:3001/register", request)
  }
  return (
    <div>
        <span>Username</span>
        <input type="text"  onChange={(evt) => {setUname(evt.target.value)}} />
        <br />
        <span>Password</span>
        <input type="password" onChange={(evt) => {setPW(evt.target.value)}}/>
        <br />
        <input type="button" value="Create Account"  onClick={submit}/>
    </div>
  );
}
export default App;
```

HANDLING REQUEST FAILURE

WEB CLIENT SECURITY



We aren't providing any error handling

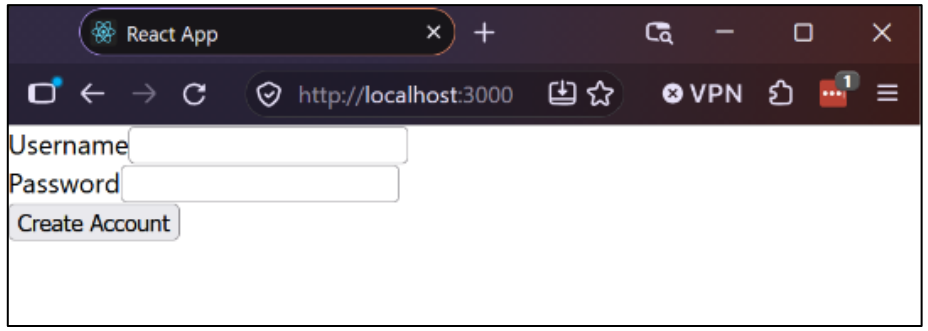
We should fail gracefully (and probably indicate success while we're at it)

src/App.js

```
import './App.css';
import { useState } from 'react'
import axios  from 'axios'
function App() {
  const [uname, setUname] = useState("");
  const [pw, setPW] = useState("");
  async function submit(){
     let request = {user:uname, pw:pw}
     axios.post("http://localhost:3001/register", request)
  }
  …
}
export default App;
```

HANDLING REQUEST FAILURE

WEB CLIENT SECURITY



We aren't providing any error handling

We should fail gracefully (and probably indicate success while we're at it)

src/App.js

```
import './App.css';
import { useState } from 'react'
import axios  from 'axios'
function App() {
  const [uname, setUname] = useState("");
  const [pw, setPW] = useState("");
  async function submit(){
     let request = {user:uname, pw:pw}
     axios.post("http://localhost:3001/register", request)
    .then((resp) => {
       resp.data.success ? alert("success!") : alert("failure");
    })
    .error((err) => {
      console.log("error: ", err);
      alert("error");
    })
  }
  …
}
export default App;
```

FRONTEND/BACKEND CONNECTION

WEB SERVER SECURITY

FRONTEND/BACKEND CONNECTION

WEB SERVER SECURITY

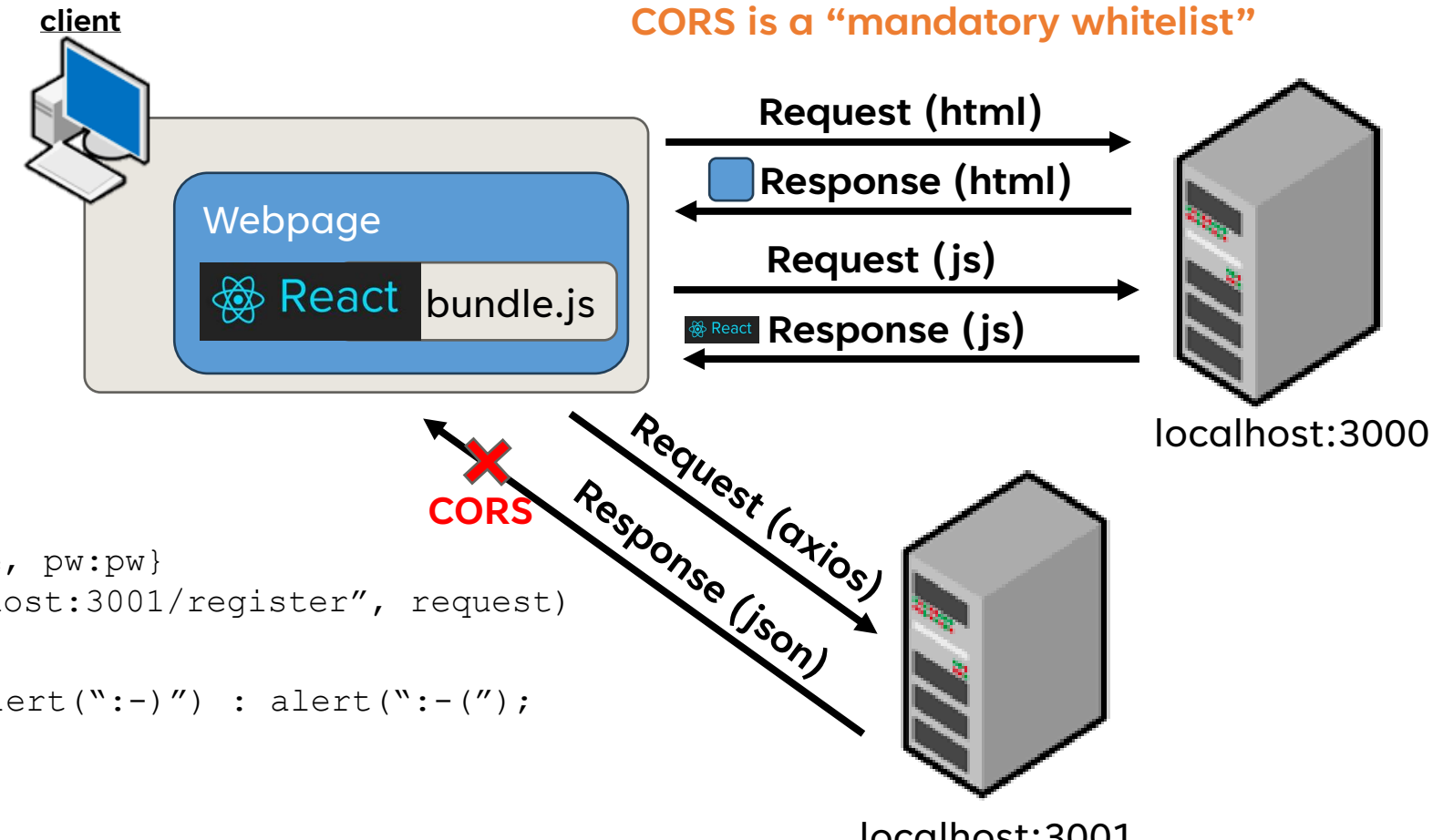
TIME TO PROCESS THE FRONTEND REQUEST

We are technically on two different origins!

http://localhost:3000

http://localhost:3001

CORS "lets" a server (e.g. localhost:3001) tell the browser when a webpage from a different origin (e.g. localhost:3000) is allowed to read its response



frontend src/App.js

```
…
async function submit(){
    let request = {user:uname, pw:pw}
    axios.post("http://localhost:3001/register", request)
    .then((resp) => {
       resp.data.success ? alert(":-)") : alert(":-(");
    })
…
```

FRONTEND/BACKEND CONNECTION

WEB SERVER SECURITY

TIME TO PROCESS THE FRONTEND REQUEST

We are technically on two different origins!

http://localhost:3000

http://localhost:3001

backend: src/index.js

```
import express from 'express'
import validator from 'validator'
const app = express()
app.get("/page", (req, res) => {
  let v = req.query.v ? req.query.v : "world";
  v = validator.escape(v);
  res.send(`<h1>Hello ${v}!</h1>`);
});
app.listen(3001);
```

FRONTEND/BACKEND CONNECTION

WEB SERVER SECURITY

TIME TO PROCESS THE FRONTEND REQUEST

We are technically on two different origins!

http://localhost:3000

http://localhost:3001

backend: src/index.js

```
import express from 'express'
import validator from 'validator'
import cors from 'cors'
const app = express()
app.get("/page", (req, res) => {
  let v = req.query.v ? req.query.v : "world";
  v = validator.escape(v);
  res.send(`<h1>Hello ${v}!</h1>`);
});
app.listen(3001);
```

FRONTEND/BACKEND CONNECTION

WEB SERVER SECURITY

TIME TO PROCESS THE FRONTEND REQUEST

We are technically on two different origins!

http://localhost:3000

http://localhost:3001

backend: src/index.js

```
import express from 'express'
import validator from 'validator'
import cors from 'cors'

const app = express()
app.use(cors({origin:"http://localhost:3000"}));
app.get("/page", (req, res) => {
  let v = req.query.v ? req.query.v : "world";
  v = validator.escape(v);
  res.send(`<h1>Hello ${v}!</h1>`);
});
app.listen(3001);
```

FRONTEND/BACKEND CONNECTION

WEB SERVER SECURITY

TIME TO PROCESS THE FRONTEND REQUEST

Handled CORS

Accept POST request

backend: src/index.js

```
import express from 'express'
import validator from 'validator'
import cors from 'cors'

const app = express()

app.use(cors({origin:“http://localhost:3000"}));

app.get("/page", (req, res) => {
  let v = req.query.v ? req.query.v : "world";
  v = validator.escape(v);

  res.send(`<h1>Hello ${v}!</h1>`);
});

app.listen(3001);
```

FRONTEND/BACKEND CONNECTION

WEB SERVER SECURITY

TIME TO PROCESS THE FRONTEND REQUEST

Handled CORS

Accept POST request

backend: src/index.js

```
import express from 'express'
import validator from 'validator'
import cors from 'cors'
const app = express()
app.use(cors({origin:"http://localhost:3000"}));
app.get("/page", (req, res) => {
  let v = req.query.v ? req.query.v : "world";
  v = validator.escape(v);
  res.send(`<h1>Hello ${v}!</h1>`);
});
app.post("/register", (req, res) => {
  res.send(JSON.stringify({'success':true}));
 })
app.listen(3001);
```

FRONTEND/BACKEND CONNECTION

WEB SERVER SECURITY



TIME TO PROCESS THE FRONTEND REQUEST

Handled CORS

Accept POST request

Persist user data (e.g. in sqlite)

backend: src/index.js

```
import express from 'express'
import validator from 'validator'
import cors from 'cors'

const app = express()

app.use(cors({origin:"http://localhost:3000"}));

app.use(express.json())

app.get("/page", (req, res) => {
  let v = req.query.v ? req.query.v : "world";
  v = validator.escape(v);
  res.send(`<h1>Hello ${v}!</h1>`);
});

app.post("/register", (req, res) => {
  res.send(JSON.stringify({'success':true}));
 })
app.listen(3001);
```

FRONTEND/BACKEND CONNECTION

WEB SERVER SECURITY



TIME TO PROCESS THE FRONTEND REQUEST

Handled CORS

Accept POST request

Persist user data (e.g. in sqlite)

backend: src/index.js

```
import express from 'express'
import validator from 'validator'
import cors from 'cors'
import { DatabaseSync } from 'node:sqlite';

const app = express()
app.use(cors({origin:"http://localhost:3000"}));
app.use(express.json())
app.get("/page", (req, res) => {
  let v = req.query.v ? req.query.v : "world";
  v = validator.escape(v);
  res.send(`<h1>Hello ${v}!</h1>`);
});
app.post("/register", (req, res) => {
  res.send(JSON.stringify({'success':true}));
 })
app.listen(3001);
```

FRONTEND/BACKEND CONNECTION

WEB SERVER SECURITY



TIME TO PROCESS THE FRONTEND REQUEST

Handled CORS

Accept POST request

Persist user data (e.g. in sqlite)

backend: src/index.js

```
import express from 'express'
import validator from 'validator'
import cors from 'cors'
import { DatabaseSync } from 'node:sqlite';
const db = new DatabaseSync("./auth.db");
const app = express()
app.use(cors({origin:"http://localhost:3000"}));
app.use(express.json())
app.get("/page", (req, res) => {
  let v = req.query.v ? req.query.v : "world";
  v = validator.escape(v);
  res.send(`<h1>Hello ${v}!</h1>`);
});
app.post("/register", (req, res) => {
  res.send(JSON.stringify({'success':true}));
 })
app.listen(3001);
```

FRONTEND/BACKEND CONNECTION

WEB SERVER SECURITY



TIME TO PROCESS THE FRONTEND REQUEST

Handled CORS

Accept POST request

Persist user data (e.g. in sqlite)

backend: src/index.js

```
import express from 'express'
import validator from 'validator'
import cors from 'cors'
import { DatabaseSync } from 'node:sqlite';
const db = new DatabaseSync("./auth.db");
db.exec(`CREATE TABLE IF NOT EXISTS users (
    name TEXT UNIQUE NOT NULL, pw TEXT NOT NULL)`);
const app = express()
app.use(cors({origin:"http://localhost:3000"}));
app.use(express.json())
app.get("/page", (req, res) => {
  let v = req.query.v ? req.query.v : "world";
  v = validator.escape(v);
  res.send(`<h1>Hello ${v}!</h1>`);
});
app.post("/register", (req, res) => {
  res.send(JSON.stringify({'success':true}));
 })
app.listen(3001);
```

FRONTEND/BACKEND CONNECTION

WEB SERVER SECURITY



TIME TO PROCESS THE FRONTEND REQUEST

Handled CORS

Accept POST request

Persist user data (e.g. in sqlite)

backend: src/index.js

```
import express from 'express'
import validator from 'validator'
import cors from 'cors'
import { DatabaseSync } from 'node:sqlite';
const db = new DatabaseSync("./auth.db");
db.exec(`CREATE TABLE IF NOT EXISTS users (
    name TEXT UNIQUE NOT NULL, pw TEXT NOT NULL)`);
const app = express()
app.use(cors({origin:"http://localhost:3000"}));
app.use(express.json())
app.get("/page", (req, res) => {
  let v = req.query.v ? req.query.v : "world";
  v = validator.escape(v);
  res.send(`<h1>Hello ${v}!</h1>`);
});
app.post("/register", (req, res) => {
  let u = req.body.user;
  let p = req.body.pw;
  db.exec(`INSERT INTO users VALUES '${u}','{$p}'`)
  res.send(JSON.stringify({'success':true}));
 })
app.listen(3001);
```

FRONTEND/BACKEND CONNECTION

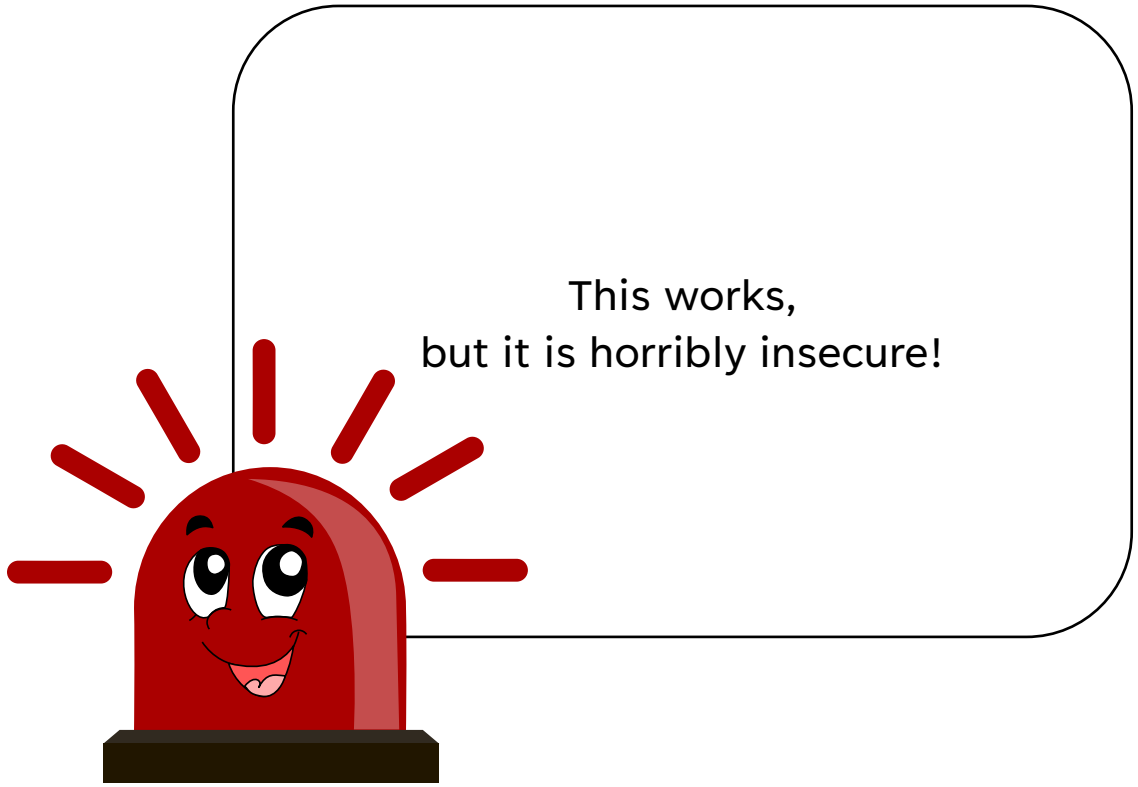
WEB SERVER SECURITY

TIME TO PROCESS THE FRONTEND REQUEST

Handled CORS

Accept POST request

Persist user data (e.g. in sqlite)



backend: src/index.js

```
import express from 'express'
import validator from 'validator'
import cors from 'cors'
import { DatabaseSync } from 'node:sqlite';
const db = new DatabaseSync("./auth.db");
db.exec(`CREATE TABLE IF NOT EXISTS users (
    name TEXT UNIQUE NOT NULL, pw TEXT NOT NULL)`);
const app = express()
app.use(cors({origin:"http://localhost:3000"}));
app.use(express.json())
app.get("/page", (req, res) => {
  let v = req.query.v ? req.query.v : "world";
  v = validator.escape(v);
  res.send(`<h1>Hello ${v}!</h1>`);
});
app.post("/register", (req, res) => {
  let u = req.body.user;
  let p = req.body.pw;
  db.exec(`INSERT INTO users VALUES '${u}','{$p}'`)
  res.send(JSON.stringify({'success':true}));
 })
app.listen(3001);
```

FRONTEND/BACKEND CONNECTION

WEB SERVER SECURITY

LET'S FOCUS OUR CODE ON THE FRAGMENTS OF CONCERN

backend: src/index.js

```
…
db.exec(`CREATE TABLE IF NOT EXISTS users (
    name TEXT UNIQUE NOT NULL, pw TEXT NOT NULL)`);
  …
app.post("/register", (req, res) => {
  let u = req.body.user;
  let p = req.body.pw;
  db.exec(`INSERT INTO users VALUES '${u}','{$p}'`)
  …
 })
…
```

No validation!

Cleartext password storage

PASSWORD STORAGE

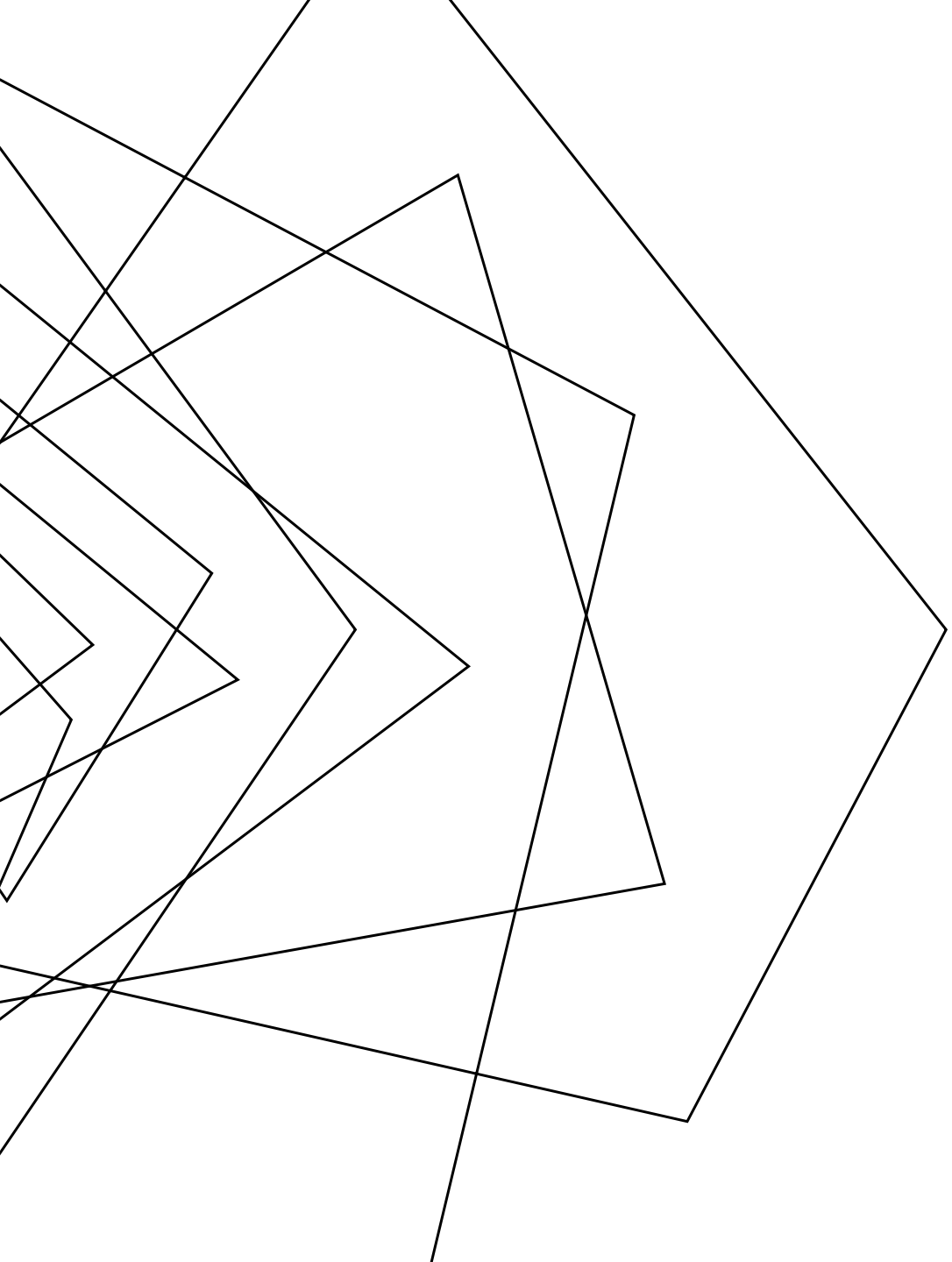
WEB SERVER SECURITY

Password Storage

Sensitive data should not be stored in the clear!

```
 let p = req.body.pw;
 bcrypt.hash(password, 10, (err, hash) => {
if (err) throw err;         db.run('INSERT INTO
users (username, password) VALUES (?, ?)',
[username, hash]);    });

 db.exec(`INSERT INTO users VALUES '${u}','{$p}'`)
```



LECTURE END!

Next Time: Securing the database and handling user authentication