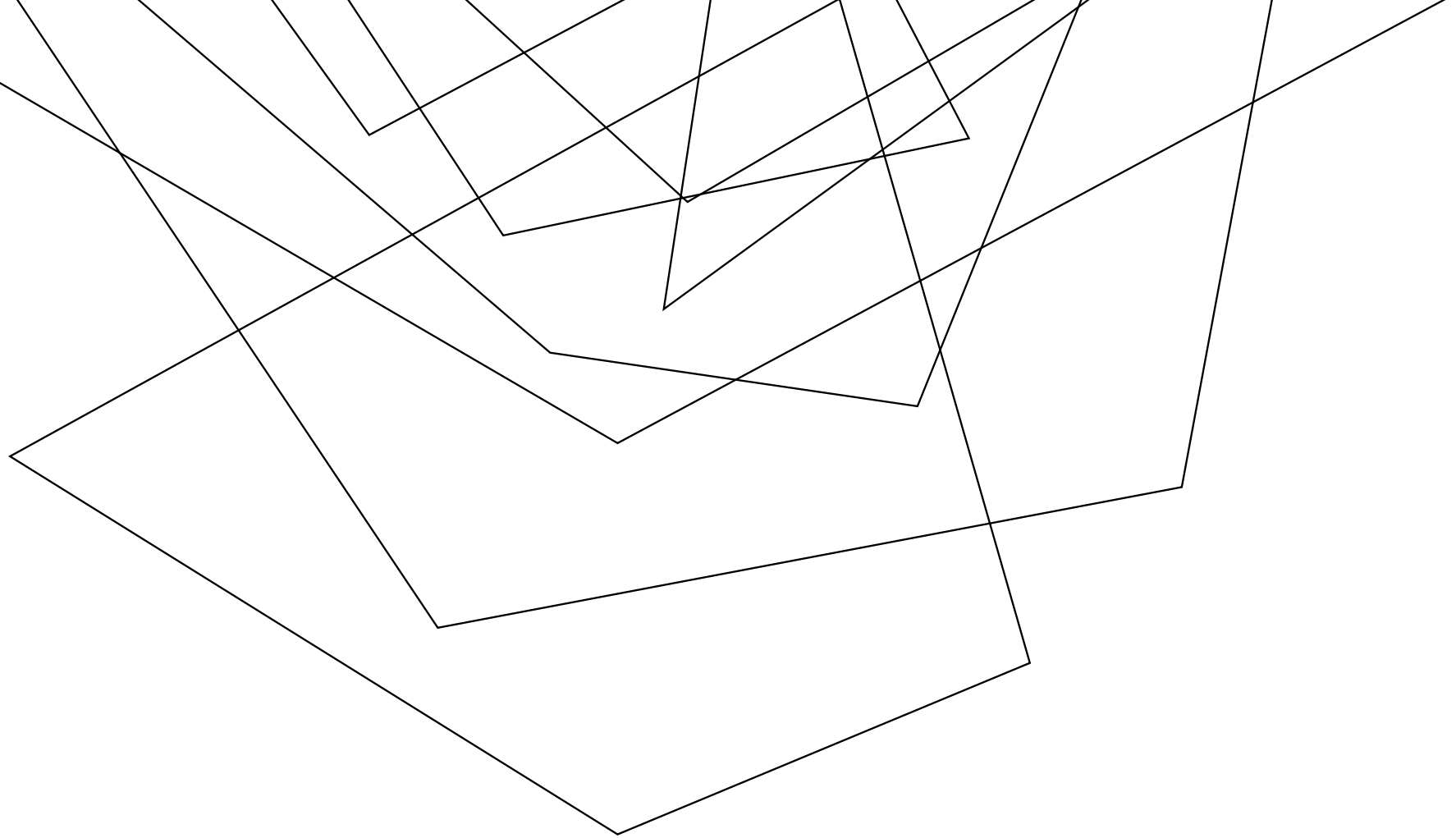
EXERCISE 36

WEB FRAMEWORKS REVIEW

Create a snippet of React code that generates a button that pops up an alert when clicked

ADMINISTRIVIA AND ANNOUNCEMENTS



FULL-STACK SECURITY

EECS 677: Software Security Evaluation

Drew Davidson

LAST TIME

REVIEW: WEB FRAMEWORKS

~~Authentication Security~~

- Introduction to React
- The client/server loop

TODAY

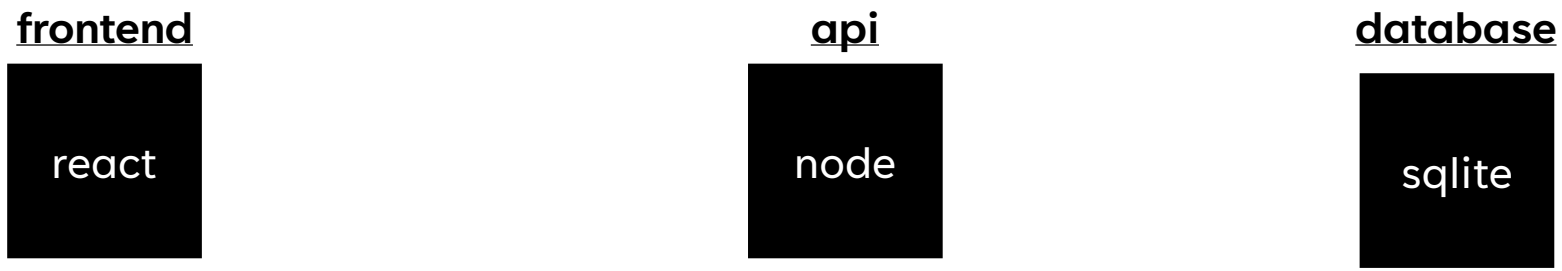
WEB PROTECTIONS

Putting it all together

- Developing a react app and examining security aspects

A MODERN WEB-APP

FULL STACK SECURITY



REACT FRONTEND

WEB FRAMEWORKS

npm create vite@latest
  Project name: frontend
  Framework: React
  Variant: Javascript
cd frontend
npm install axios
npm run dev

```
import { useState } from 'react'
import './App.css'
import { Login } from './Login.jsx'

export default function App() {
        const [token, setToken] =
           useState(window.sessionStorage.getItem("token"));

        if (token === undefined || token === null){
                return <Login tokenSetter={setToken} />
        }
        return (<p>{"Logged in!!!!"}</p>)
}
```

NODE BACKEND SETUP

WEB PROTECTIONS

#Configure codebase

```
npm create
npm install \
    express \
    cors    \
    sqlite3
```

#Configure database

```
CREATE TABLE accounts (
        id INTEGER PRIMARY KEY,
        name TEXT NOT NULL UNIQUE,
        pw TEXT NOT NULL
);

CREATE TABLE tasks (
        id INTEGER PRIMARY KEY,
        owner INTEGER NOT NULL,
        name TEXT NOT NULL
);

.save tasks.db
```

NODE BACKEND API

WEB PROTECTIONS

index.js

```
// initializer server
// initializer database
// service routes
```

```
import express from 'express';
import cors from 'cors';
import sqlite3 from 'sqlite3';
import { encodeJWT, decodeJWT } from "./token.js"

const app = express()
app.use(cors());
app.use(express.json());
```

```
const db = new sqlite3.Database("tasks.db",
sqlite3.OPEN_READWRITE);
```

```
app.post('/login', function(req, res){
… })
app.post('/enroll', function(req, res){
… })
app.post(‘/add', function(req, res){
… })
…
app.listen(3001)
```

NODE BACKEND API

WEB PROTECTIONS

index.js

```
// initializer server
// initializer database
// service routes
```

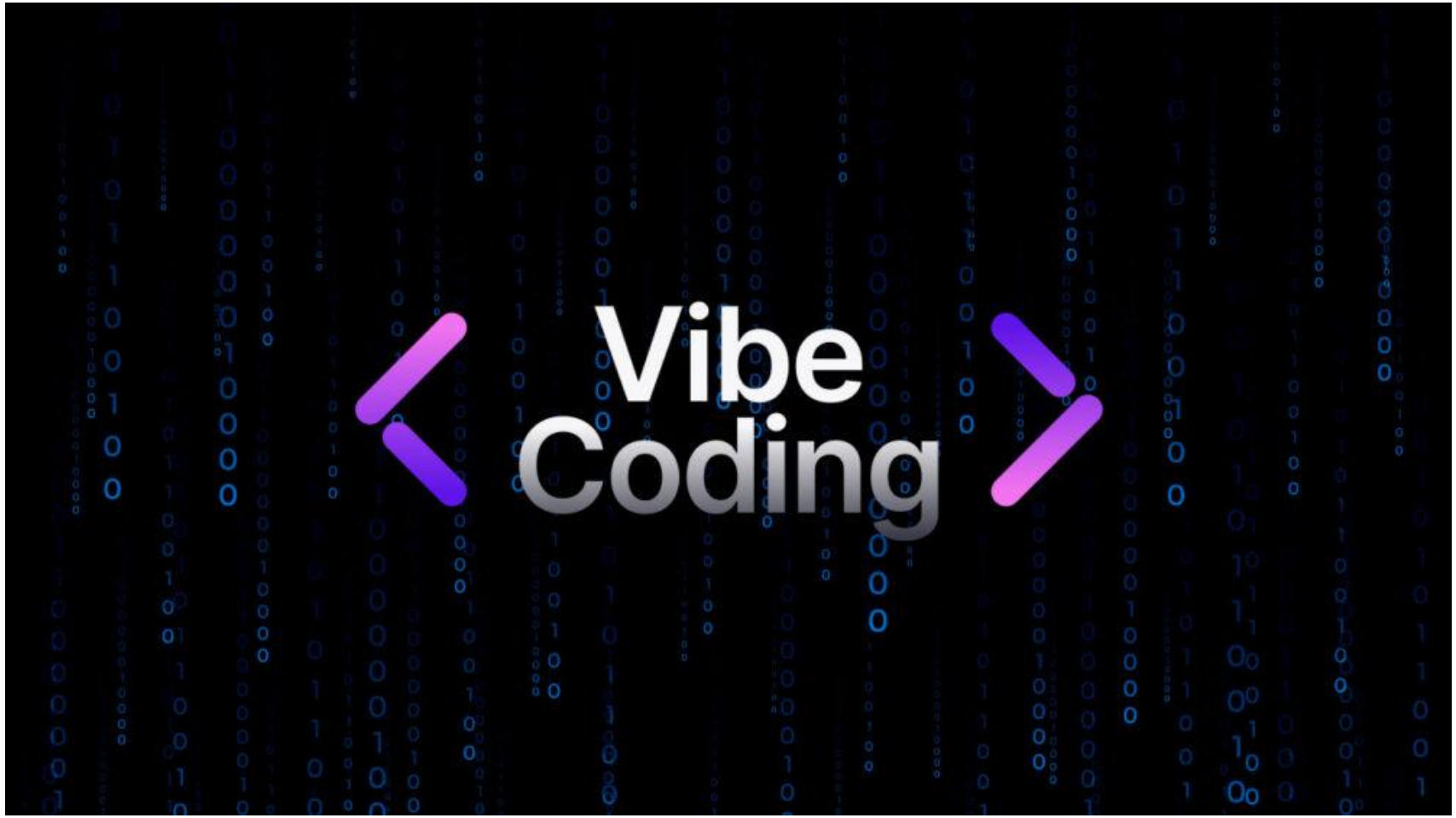
```
app.post('/login', function(req, res){
    const name = req.body.name;
    const pw = req.body.pw;

    const Q = "SELECT name,id FROM accounts"
            + " WHERE pw = '" + pw
              + "' AND name = '" + name + "'";
    db.get(Q, function(err, row){
        if (!row){
            resp.send(JSON.stringify({ok:false}));
            return;
        }
        const creds = { "name":row[‘name’], "id":row['id’]}
        const token = encodeJWT(creds, SECRET);
        const res = {'ok':true,'token':token};
        resp.send(JSON.stringify(res));
    });

})
```

TO THE TERMINAL!

FULL STACK SECURITY

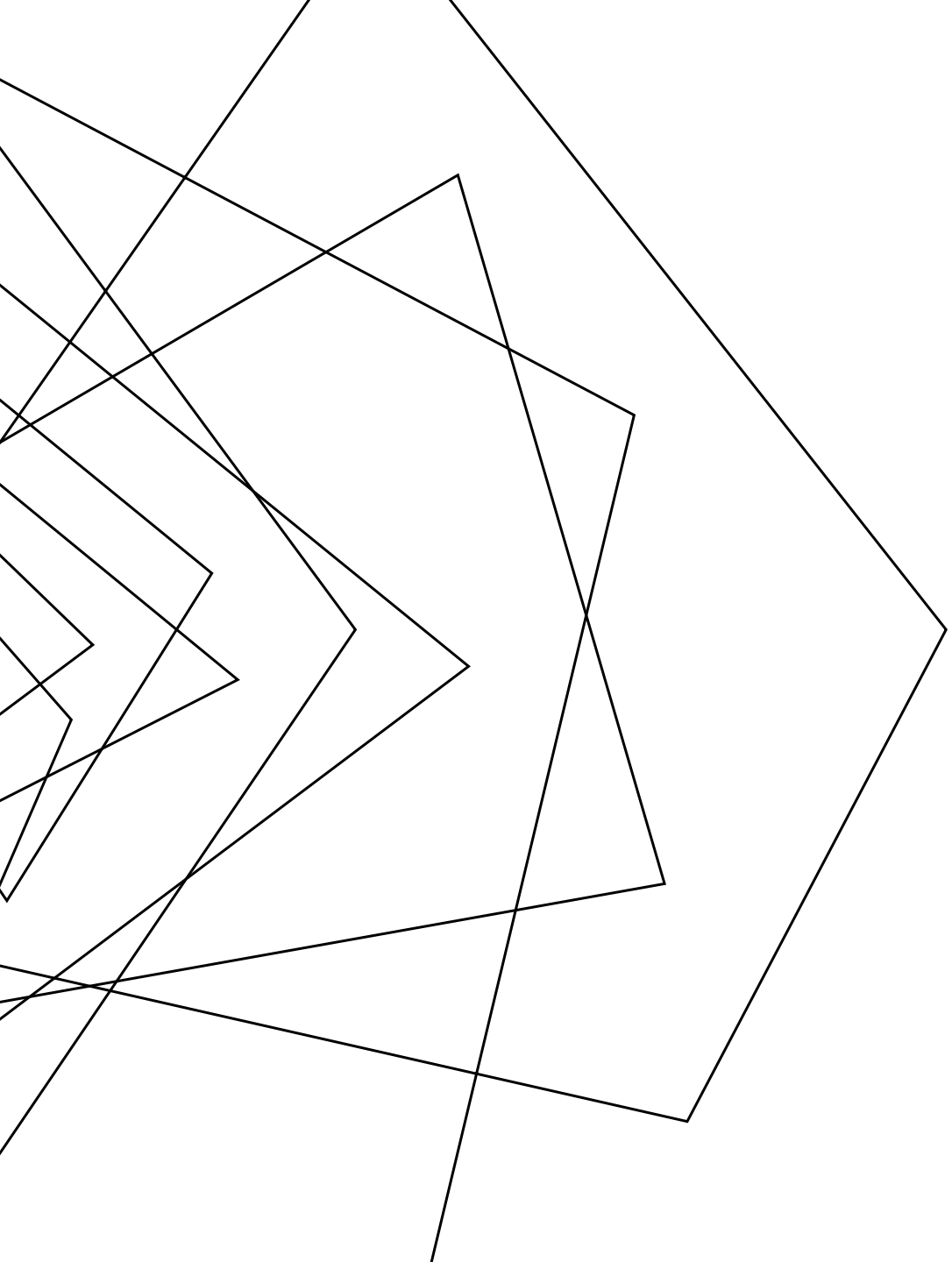
BUT I DON’T LIKE WRITING CODE!

FULL STACK SECURITY



TO THE BROWSER!

FULL STACK SECURITY



LECTURE END!

Basic end-to-end scheme

Next Time: Filling out functionality