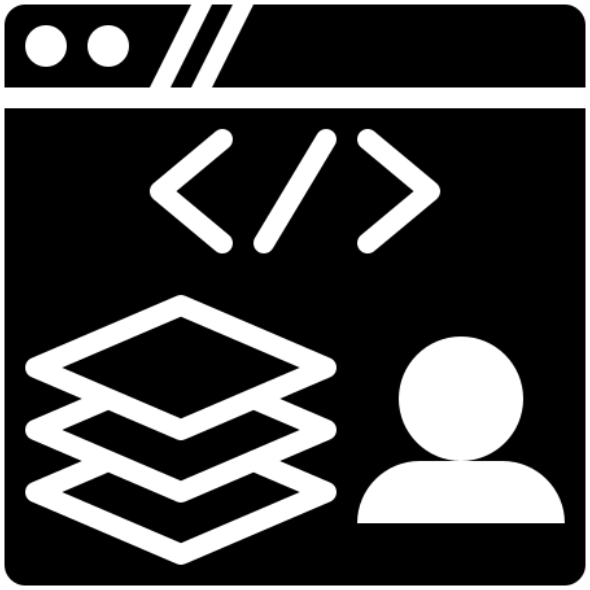


Full Stack

SENG 4640
Software Engineering for Web Apps
Winter 2023

Sina Keshvadi
Thompson Rivers University

What is Front-End Development?

- Designing and implementing user interface
- Using HTML, CSS, and JavaScript
- Creating the visual and interactive aspects of an application

What is Back-End Development?

- Developing server-side of an application
- Managing databases and server-side programming
- Integrating with other systems

What is Full Stack Development?

- Developing both front-end and back-end of web applications
- Combining client-side and server-side technologies

What is a Full Stack Developer?

- Skilled in both front-end and back-end development
- Capable of building complete web applications
- Understanding of how different components work together

What is the MERN stack?

- MongoDB, Express.js, React, Node.js
- Full stack development environment for web applications

MongoDB

- NoSQL database
- Stores data in JSON-like format
- Scalable and efficient for handling large amounts of data

Express.js

- Back-end web framework for Node.js
- Provides tools for building web applications
- Handles HTTP requests and responses

React

- Front-end JavaScript library for building user interfaces
- Declarative and component-based
- Enables building complex UIs with reusable components

Node.js

- Server-side JavaScript runtime
- Allows for building scalable and high-performance applications
- Enables building real-time applications with web sockets

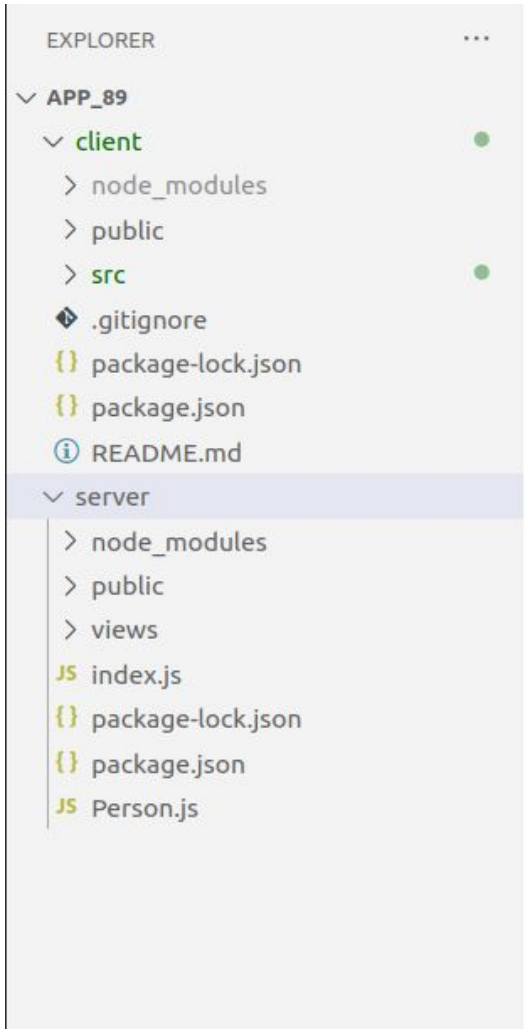
Why MERN stack?

- Efficient for handling large amounts of data
- Real-time updates for collaboration tools or social networking platforms
- Robust and scalable full stack development environment

Why MERN stack?

- The MERN stack is a popular choice for building modern web applications
- Its combination of MongoDB, Express.js, React, and Node.js provides a complete full stack development environment that can handle complex data and real-time updates.

Let’s implement the People example with React



client is react app

server is express and mongo

This was all route

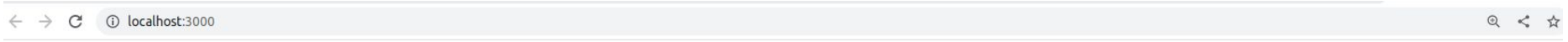
```
app.use("/all", async (req, res) => {
 try {
   const allPeople = await Person.find();
   if (allPeople.length === 0) {
     res.status(200).send("There are no people");
   } else {
     res.render("showAll", { people: allPeople });
   }
 } catch (err) {
   res.status(500).send("Error: " + err);
 }
});
```

Change to send json object

```
app.use("/all", async (req, res) => {
 try {
   const allPeople = await Person.find();
   if (allPeople.length === 0) {
     res.status(200).send("There are no people");
   } else {
     res.status(200).json(allPeople);
   }
 } catch (err) {
   res.status(500).send("Error: " + err);
 }
});
```

```
app.listen(3001, () => {
 console.log("Listening on port 3001");
});
```

Change the port number to something else so that you can run both React and Express servers simultaneously without any port conflicts.



People:

Whoopsie, it's not working!
But don't panic, we'll get it back on track soon!

```
var express = require("express");
var app = express();
var cors = require("cors");


app.set("view engine", "ejs");

var bodyParser = require("body-parser");
app.use(bodyParser.urlencoded({ extended: true }));
app.use(cors());
```

cors stands for "Cross-Origin Resource Sharing" and is a **security feature** that allows resources from different origins (e.g., different domains) to be accessed by a web page. The cors middleware in Express.js enables the server to handle CORS requests from the client-side, thus allowing a client-side application to make requests to an Express.js backend API even if it's hosted on a different domain.

now, let's change react’s App.js

```
import React, { Component } from "react";
import "./App.css";

class App extends Component {
 constructor(props) {
   super(props);
   this.state = { people: [] };
 }

 componentDidMount() {
   fetch("http://localhost:3001/all")
     .then((res) => res.json())
     .then((data) => this.setState({ people: data }))
     .catch((err) => console.error(err));
 }

 render() {
   return (
     <div className="App">
       <h1>People:</h1>
       <ul>
         {this.state.people.map((person) => (
           <li key={person._id}>
             {person.name} ({person.age})
           </li>
         ))}
       </ul>
     </div>
   );
 }
}

export default App;
```

localhost:3000

Thank you All:

-
-
-
-
-

Sany (21)
Alex (22)
Ben (20)
Sam (22)
Trever (22)

This was a great semester!

We are done with Programming Part!

- Thank you for participating in this course.
- I appreciate your patience as this was the first semester of offering this course.
- Your feedback and participation helped me develop this course.
- I hope that what you have learned will be useful to you in the future.