Internship Report – Frontend Dev

Week 6: React.js Basics

Name: Zainab
Father Name: Assad Qayyum
Date: 28th July, 2025
Internship Domain: Front-end Intern
Task: Intro to React, create-react-app setup, JSX, Components (Functional), Props

Task Overview: (Day1)

In today's task, I was introduced to React.js and learned the basics of setting up a React project using create-react-app. I also explored JSX syntax, functional components, and how to pass data using props, gaining a foundational understanding of React's component-based architecture.

Content Covered:

The content covered today is:

- Introduction to React
- Create-react-app setup
- JSX (JavaScript XML)
- Components (Functional)
- Props (Properties)

1. Introduction to React:

React is a **JavaScript library** developed by Facebook for building **user interfaces (UI).** It allows you to create reusable UI components, making development faster and easier.

React uses a component-based architecture, meaning your app is divided into small, independent, reusable pieces. It uses a virtual DOM, which efficiently updates the UI when data changes, improving performance.

Why Use React?

Reusable Components: Build once, use multiple times.

Fast: Virtual DOM only updates parts of the page that change.

Interactive: Makes websites dynamic (e.g., live search, chats).

Declarative: You describe what you want to see and React handles the DOM updates.

Key Concepts:

- **JSX:** A syntax that lets you write HTML-like code inside JavaScript.
- **Components:** Building blocks of a React app (can be functional or class-based).
- **Props:** Data passed from a parent to a child component.
- **State:** Data that changes over time within a component.
- **Lifecycle:** How a component is created, updated, and removed.

2. Create-react-app Setup

Create-react-app is a tool that sets up everything you need to start a React project without manual configuration.

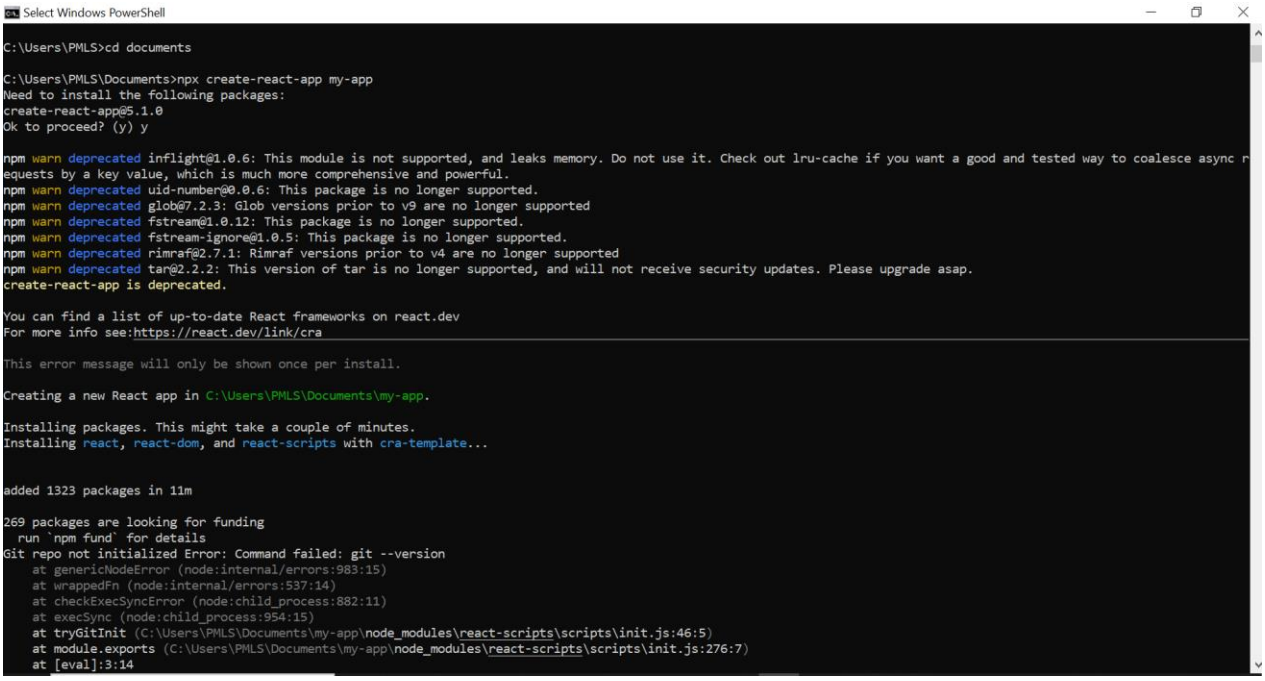
Steps to Create a React App:

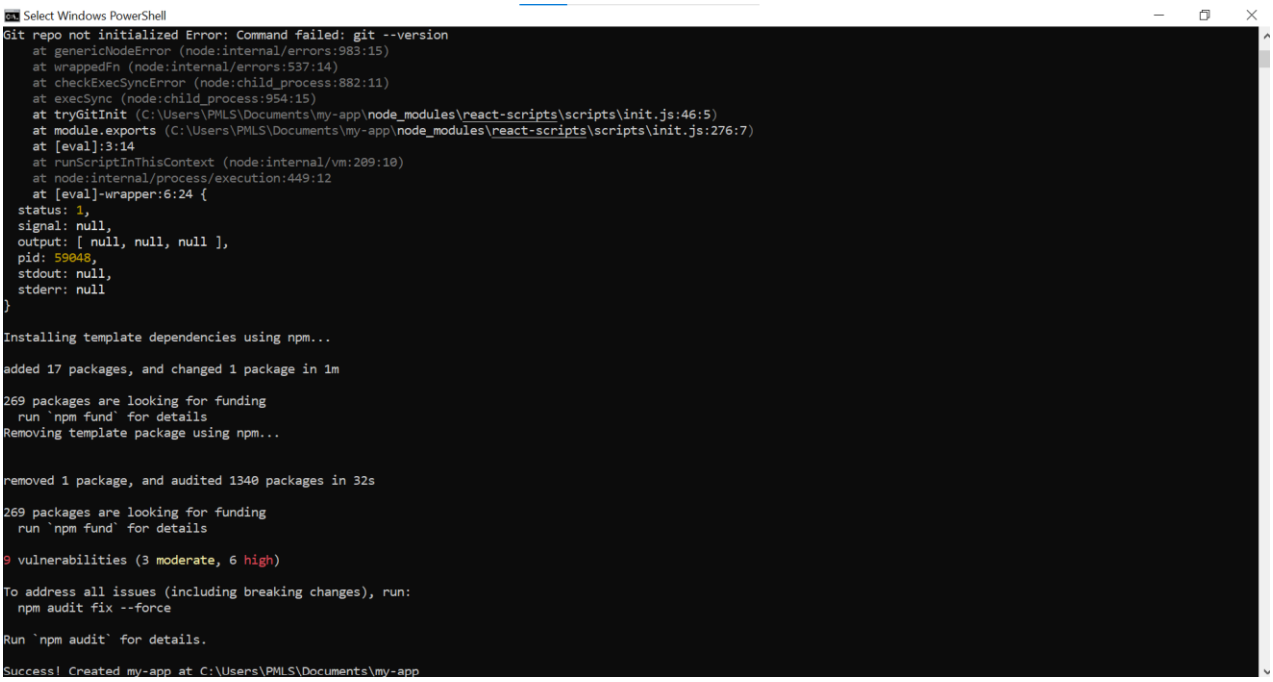
1. **Install Node.js & npm (if not already installed)**
   Check with node -v npm -v

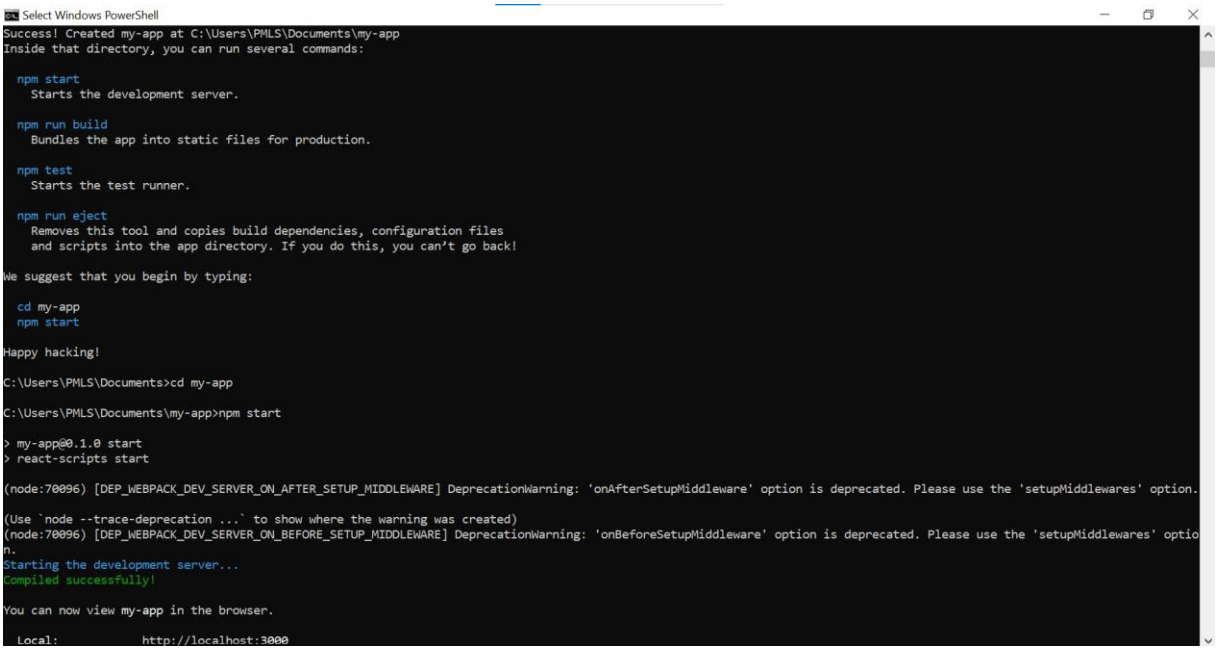
2. **Create a new React project:**
   npx create-react-app my-app

3. **Navigate to the project folder:**
   cd my-app

4. **Start the development server:**
   npm start (Opens the app in the browser at http://localhost:3000)







3. JSX (JavaScript XML)

JSX is a syntax extension for JavaScript used in React. It looks like HTML but is actually JavaScript under the hood. It allows you to write UI elements directly inside JavaScript.

Example:

Without JSX:

```
const element = React.createElement('h1', null, 'Hello, React!');
```

With JSX:

```
const element = <h1>Hello, React!</h1>;
```

Rules

- JSX must have one parent element (wrap multiple elements in a div or <> fragment).
- Use className instead of class for CSS.
- Embed JavaScript using curly braces {}:

```
const name = "Zainab";
const element = <h1>Hello, {name}</h1>;
```

4. Components (Functional)

- Components are building blocks of a React app.
- A component is a function or class that returns JSX.
- Functional components are simple JavaScript functions.

Example:

```
function Welcome() {
  return <h2>Welcome to React!</h2>;
}
```

You can use this component inside another like an HTML tag:

```
function App() {
  return (
    <div>
      <Welcome />
    </div>
  );   }
```

5. Props (Properties)

- Props are a way to pass data from a parent component to a child component.
- Think of it like HTML attributes.
- Props are read-only (cannot be modified by the child component).

Example:

```
function Welcome(props) {
  return <h2>Hello, {props.name}</h2>;
}

function App() {
  return (
    <div>
      <Welcome name="Zainab" />
      <Welcome name="Ali" />
    </div>
  );
}
```

- name="Zainab" → passed as a prop to Welcome.
- Output:

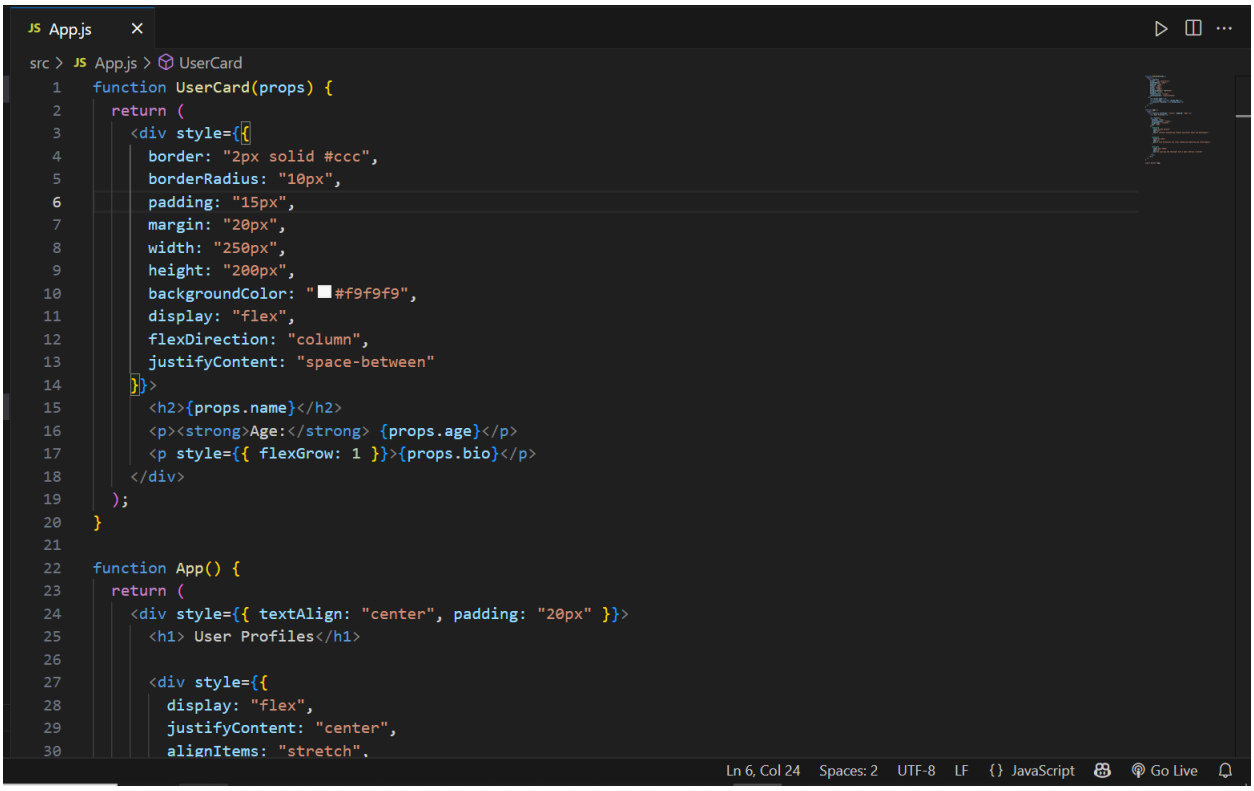
  Hello, Zainab

  Hello, Ali

Inside the src/ folder, we'll mainly work with these:

- App.js → Main component displayed on the screen (we'll often start editing here).
- App.css → Styles for the App component.
- index.js → Entry point of the application; usually not modified often.
- index.css → Global styles for the app.

Practice Code:

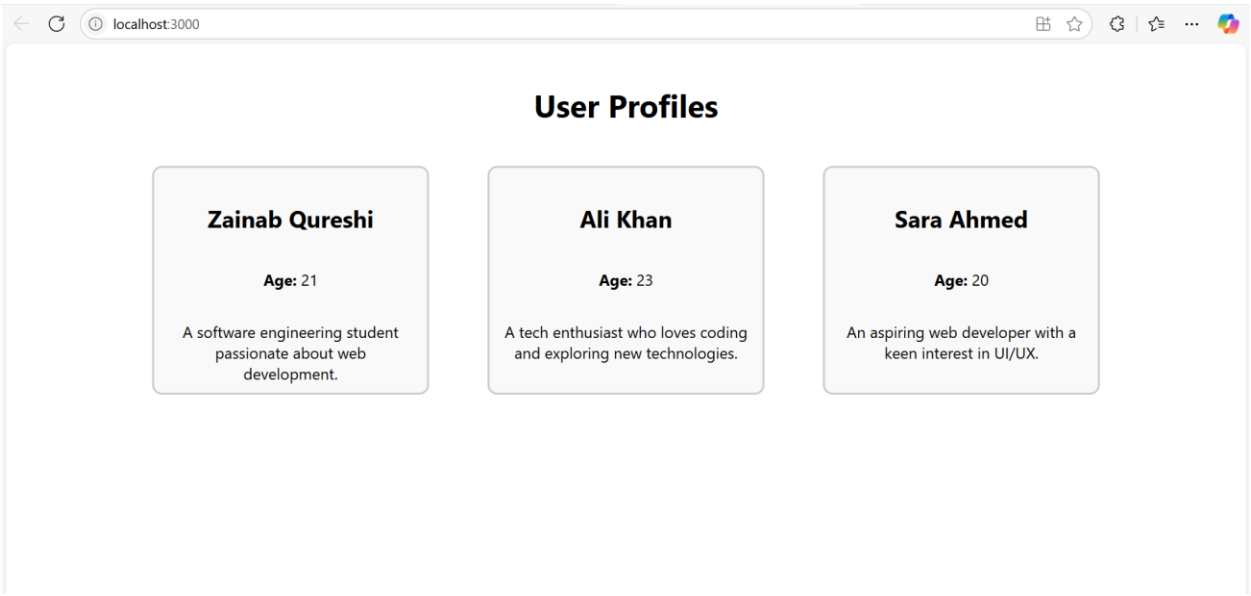


App.js

src > App.js > UserCard

```jsx
function App() {

      <div style={{
        display: "flex",
        justifyContent: "center",
        alignItems: "stretch",
        gap: "20px"
      }}>
        <UserCard
          name="Zainab Qureshi"
          age="21"
          bio="A software engineering student passionate about web development."
        />

        <UserCard
          name="Ali Khan"
          age="23"
          bio="A tech enthusiast who loves coding and exploring new technologies."
        />

        <UserCard
          name="Sara Ahmed"
          age="20"
          bio="An aspiring web developer with a keen interest in UI/UX."
        />
      </div>
    </div>
  );
}

```

Ln 6, Col 24 Spaces: 2 UTF-8 LF {} JavaScript Go Live



Conclusion:

This task helped me understand the core concepts of React.js and how it simplifies building dynamic, reusable user interfaces. I successfully created a React project, practiced JSX, built functional components, and used props for passing data.