Day 2

Let's set Up A React.js Project

A thorough explanation

kodemaven-portfolio.vercel.app

Prerequisites

Since React.js is a Javascript library, we will be needing an environment to run this language smoothly. So we will need to install:

1. Node.js Package: A JavaScript runtime that allows us to run JavaScript outside of a web browser. React relies on Node.js to manage dependencies and run development tools.

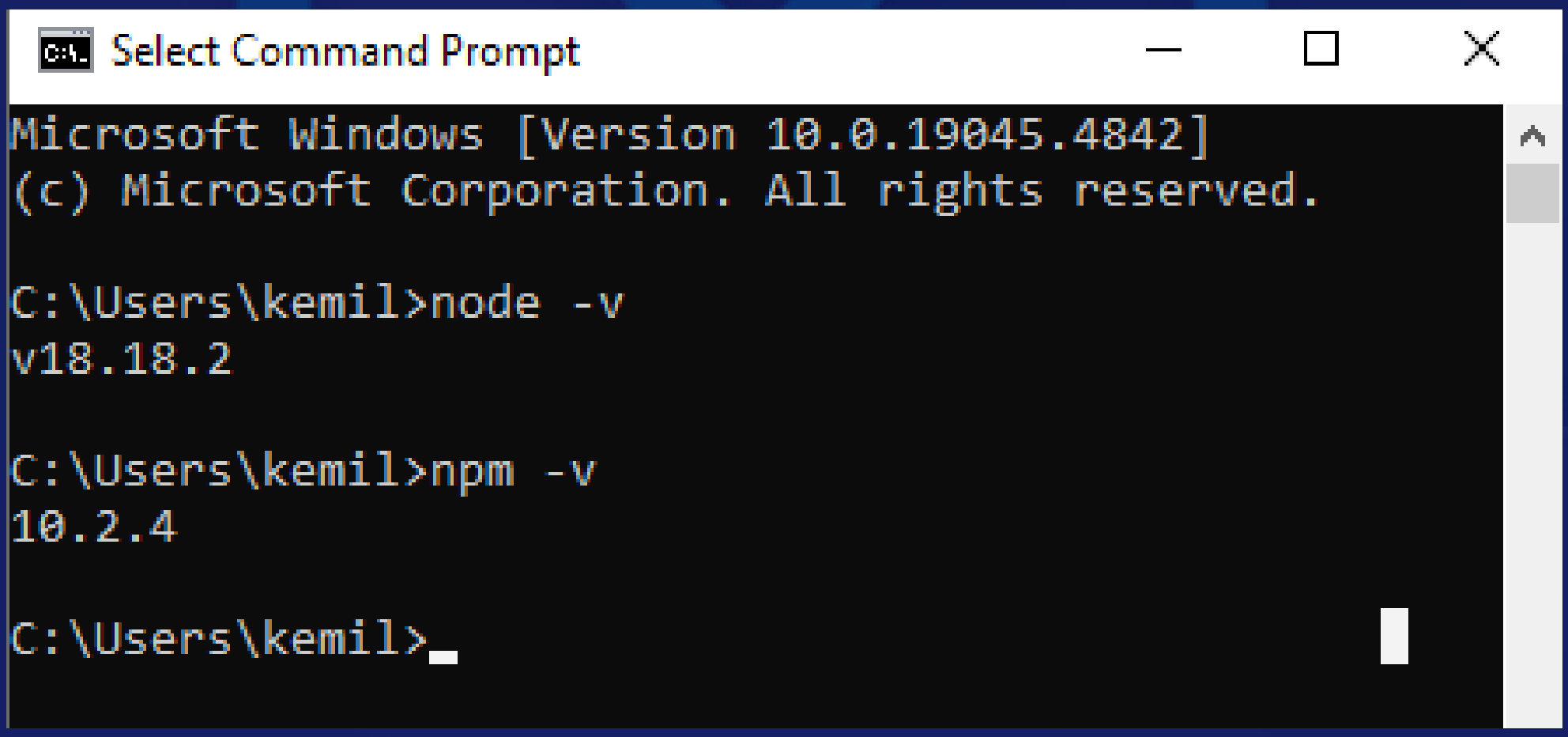
2. npm (Node Package Manager): Comes with Node.js and is used to install libraries and packages needed for React development.

3. Code Editor: A good code editor can greatly enhance your development experience. Some popular choices are visual Studio Code (VS Code), Atom, Sublime.

To install Node.js and npm:

- Visit the Node.js official website (https://nodejs.org/en) and download the **LTS (Long-Term Support)** version.
- Follow the installation instructions for your operating system, whether for Windows, Linux, or MacOS.
- To verify the installation, open your terminal or command prompt and type:

```
node -v
npm -v
```



Setting Up A React Project

There are two primary ways to quickly set up a React project: Create React App (CRA) and Vite. Both offer different benefits, so let’s explore how to use each and understand their folder structures.

Using “create-react-app” (CRA)

Create React App (CRA) is a tool provided by the React team that sets up a modern React project with a single command.

It includes everything you need to get started, like **Webpack** for bundling, **Babel** for JavaScript transpiling, and **ESLint** for linting.

Steps:

1. Open your terminal on your code editor or command prompt. I will be using VS code terminal.

```
kemil@DESKTOP-QKVDT62 MINGW64 /c/Desktop
$
```

2. Create a folder called “**my-react-project**” uisng the **“mkdir”** terminal command or just directly opening an already created folder in your editor. I will be using the command prompt:

```
kemil@DESKTOP-QKVDT62 MINGW64 /c/Desktop/My-Projects
$ mkdir my-react-app
```

3. Use the **change directory (cd)** command to navigate into the created folder like this:

```
kemil@DESKTOP-QKVDT62 MINGW64 /c/Desktop/My-Projects
$ cd my-react-app

kemil@DESKTOP-QKVDT62 MINGW64 /c/Desktop/My-Projects/my-react-app
$
```

4. We are now in the **“my-react-app”** directory. Now we will write the command for creating our React project “**npx create-react-app**”.

```
kemil@DESKTOP-QKVDT62 MINGW64 /c/Desktop/My-Projects/my-react-app
$ npx create-react-app ./

Creating a new React app in C:\Desktop\My-Projects\my-react-app.
```

Observe that I added “./” to my command. This is because I want the react project created directly in my root directory.

There is a **faster way** of creating a react project without having to create a folder first, and that is by creating your root folder alongside your project, like this:

```
1 npx create-react-app my-app
2 cd my-app
```

Folder name

Naviagate to project folder

npx create-react-app my-app: Creates a new React project in a folder called my-app.

cd my-app: Changes the directory to the newly created project folder.

After the successfull creation of your project, you should have a bunch of folders and files in your root folder now. Git is also initialized automatically for the project.

5. The next step, is to run the development server so that we can view our project live. Make sure that you are in the **root directory** of your project, in this case “**my-react-app**”.

The command for running a “create-react-app” project is “**npm start**”.

npm start: Starts the development server.

```
kemil@DESKTOP-QKVDT62 MINGW64 /c/Desktop/My-Projects/my-react-app (main)
$ npm start
```

After successfully starting the development server, two URLs are provided, like this:

```
Compiled successfully!

You can now view my-react-app in the browser.

  Local:            http://localhost:3000
  On Your Network:  http://192.168.250.125:3000

Note that the development build is not optimized.
To create a production build, use npm run build.

webpack compiled successfully
```

1. Local: http://localhost:3000: This is the URL you can use to access your React application on your own computer (local machine).

The term localhost refers to your local machine, meaning it is only accessible from the computer on which the development server is running.

The number **3000** is the port number on your local machine where the development server is listening for incoming requests.

2. On Your Network: http://192.168.250.125:3000

This is the URL you can use to access your React application from **other devices** on the same local network (such as other computers, tablets, or smartphones **connected to the same Wi-Fi**).

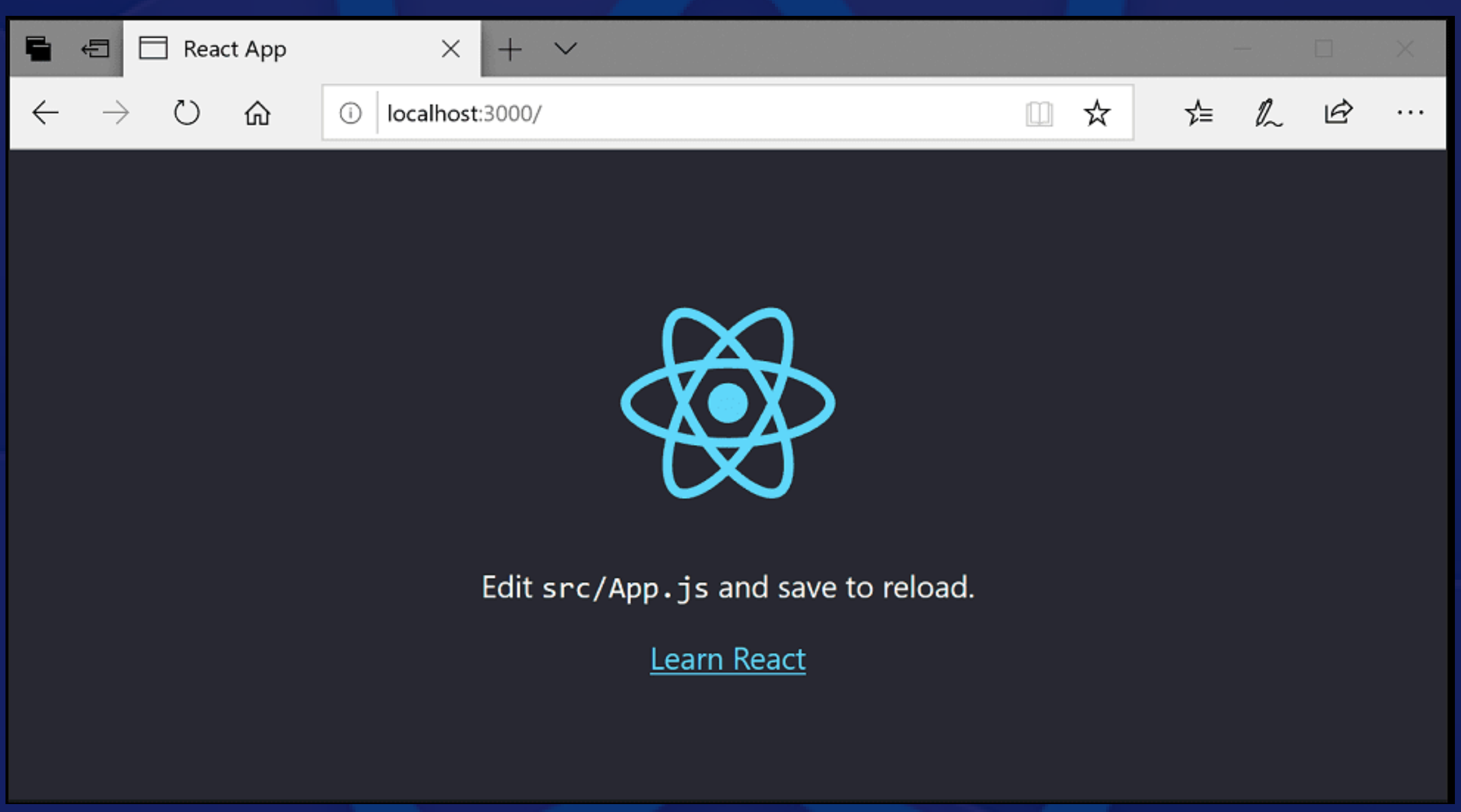
The IP address **192.168.250.125** is your computer's local network address. It allows other devices on the same network to connect to your computer.

The port **3000** is still the same, indicating where the development server is listening for requests on your network IP.

This URL is useful for testing your application across different devices, especially when styling the app for responsive views or for showing your work to others without needing to deploy it to a remote server.

6. Follow the URL (http://localhost:3000) in your terminal or copy and paste it on your browser to open your React project.

You should see the default React welcome page like this.



Congratulations, you just created your first raect project 🎉🎉🎉🎉🎉

Folder Structure of a CRA Project

```
my-app/
├── node_modules/
├── public/
│   ├── index.html
│   └── favicon.ico
├── src/
│   ├── App.css
│   ├── App.js
│   ├── App.test.js
│   ├── index.css
│   ├── index.js
│   └── reportWebVitals.js
├── .gitignore
├── package.json
└── README.md
```

Key Folders and Files:

- **node_modules/:** Contains all the dependencies and packages installed via npm.
- **public/:** Includes the static assets of your app. The main file here is index.html, which is the entry point of your React app.
- **src/:** This is where your React components and application logic reside. In here, we have:
  - index.js: The entry JavaScript file. It renders your root React component (App.js) into the index.html.
  - App.js: The main React component, which serves as the starting point for your app.
  - App.css: Contains styles for the App.js component.
  - index.css: General styling file for the entire app.
  - App.test.js: A sample test file for App.js.

    - reportWebVitals.js: Used to measure and report performance metrics.

- .gitignore: Specifies which files and directories should be ignored by Git.
- package.json: Contains metadata about your project, including dependencies, scripts, and project configurations.
- README.md: A basic documentation file about your project.

Setting Up a React Project with Vite

Vite is a modern build tool that provides a faster and leaner development experience for modern web projects.

Unlike CRA, Vite uses native ES modules in the browser for faster Hot Module Replacement (HMR) and a quicker build process.

To create a Vite project, we will follow the same steps we used using the “create-react-app” method. The major differences between these two methods for creating a React project are the commands for creating them and folder structure.

Command for CRA - npx create-react-app@latest

Command for Vite - npm create vite@latest

Observe that I added “**@latest**” to the 2 commands above, this is used when we want to the project to be the updated one (i.e. the latest version of react).

There are other commands added to this line to specify what template or language we want in the project.

Here is the full command:

```
npm create vite@latest my-vite-app -- --template react
cd my-vite-app
npm install
npm run dev
```

- **npm create vite@latest my-vite-app:** Creates a new Vite project in a folder named my-vite-app.
- **--template react:** Specifies that the template to use is for a React app.
- **cd my-vite-app:** Changes directory to the new project folder.
- **npm install:** Installs the necessary dependencies.
- **npm run dev:** Starts the Vite development server.

You will be prompted to choose the variant of the programming language you want to use for your project.

Use the keyboard up and down arrows to select your variant and press the enter button to select it, just like this:

```
kemil@DESKTOP-QKVDT62 MINGW64 /c/Desktop/My-Projects
$ npm create vite@latest my-vite-app
√ Select a framework: » React
? Select a variant: » - Use arrow-keys. Return to submit.
    TypeScript
    TypeScript + SWC
>   JavaScript
    JavaScript + SWC
    Remix ↗
```

- Then run the commands “**cd my-vite-app**” (to navigate to your project directory), then “**npm install**”, followed by “**npm run dev**” to start the development server.

The development URl is opened and you have two URLs just like the CRA.

For Vite project, the localhost port is “5173” or any “517-” that your project returns, instead of “3000” for CRA.

```
kemil@DESKTOP-QKVDT62 MINGW64 /c/Desktop/My-Projects/my-vite-app
$ npm run dev

> my-vite-app@0.0.0 dev
> vite


  VITE v5.4.2  ready in 1256 ms

  ➜  Local:   http://localhost:5173/
  ➜  Network: use --host to expose
  ➜  press h + enter to show help
```

And the **Network host** is not exposed unlike CRA. You will have to configure it in the **vite.config.js** file to expose your network port like this:

```ts
export default defineConfig({
  plugins: [react()],
  server: {
    host: "0.0.0.0",
    port: 3000,
  },
});
```

Server configuration:

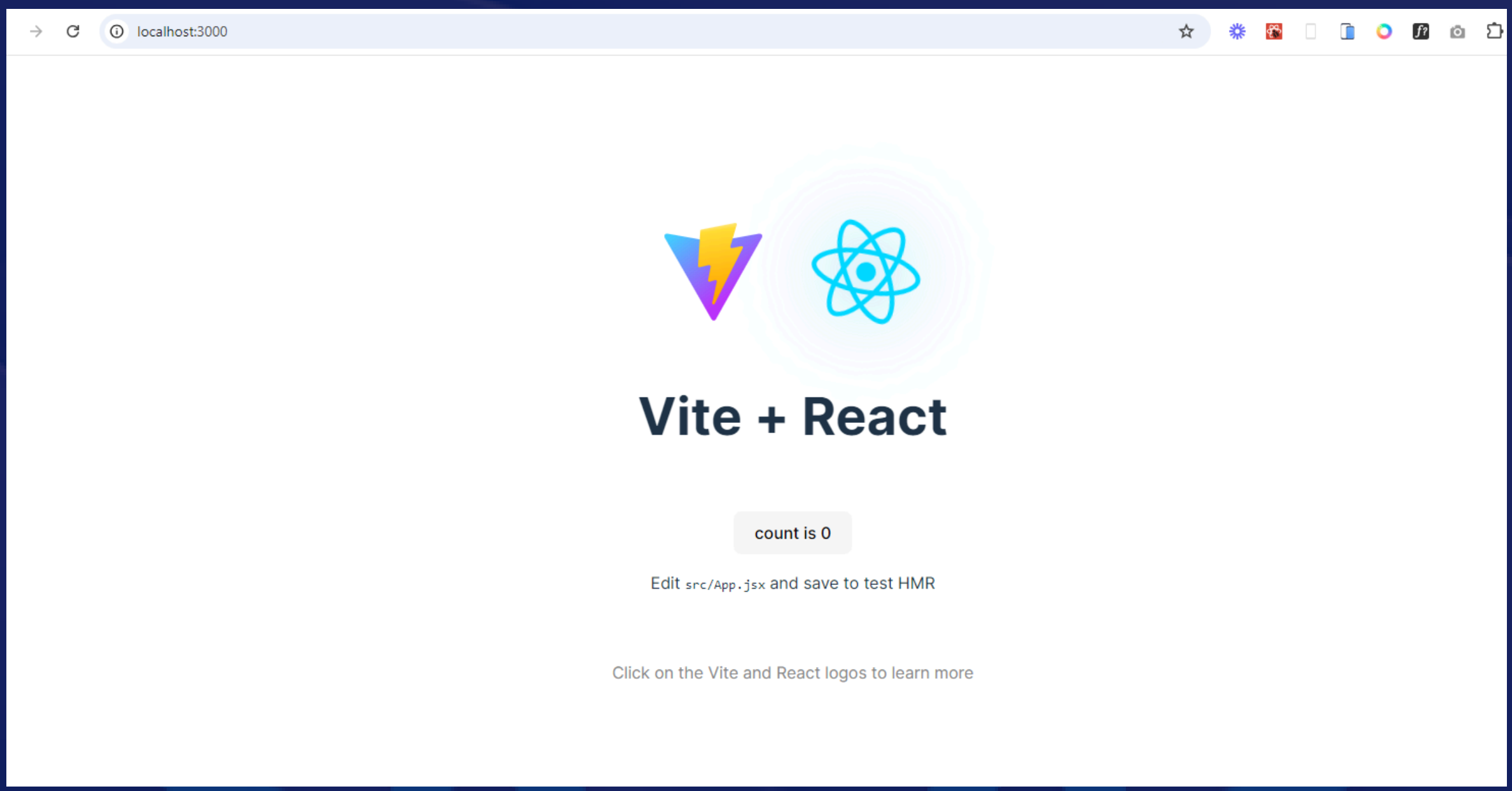
host: '0.0.0.0': Listen on all addresses, including LAN

port: 3000: Optionally, specify a port (default is 5173)

```
➜  Local:   http://localhost:3000/
➜  Network: http://192.168.65.125:3000/
```

Now the network's IP address is exposed and running on port 3000 as specified.

Now Open the project in your local browser by clicking on any of the URLs.



Congratulations, you just created a Vite + React project successfully 🎉

kodemaven-portfolio.vercel.app

Folder Structure of a Vite Project

```
my-vite-app/
├── node_modules/
├── public/
│   └── vite.svg
├── src/
│   ├── App.css
│   ├── App.jsx
│   ├── index.css
│   └── main.jsx
├── .gitignore
├── index.html
├── package.json
├── README.md
└── vite.config.js
```

Key Folders and Files:

- **node_modules/:** Contains all the dependencies and packages installed via npm.
- **public/:** Includes the static assets of your app. **vite.svg** is a sample file; you can add your assets here.
- **src/:** This is where your React components and application logic reside. In here, we have:
  - main.jsx: The entry JavaScript file for Vite, similar to index.js in CRA. It renders the root React component.
  - App.jsx: The main React component, similar to App.js in CRA.
  - index.css: General styling file for the entire app.
  - App.css: Contains styles for the App.js component.
  - App.test.js: A sample test file for App.js.

- .gitignore: Specifies which files and directories should be ignored by Git.

- index.html: The entry HTML file that Vite uses. Unlike CRA, this file is directly in the root.

- vite.config.js: Vite configuration file, allowing customization of the build process.

- package.json: Contains metadata about your project, including dependencies, scripts, and project configurations.

- README.md: A basic documentation file about your project.

Comparing CRA and Vite

- **Setup Speed:** CRA sets up a new React app quickly with one command, making it very beginner-friendly, **while** Vite offers an even faster setup and project creation, with near-instant startup times due to its optimized build process.

- **Performance:** CRA uses Webpack, which can be slower during development because it rebuilds the entire application every time you save a file **while** Vite leverages modern browser features and an optimized development server, which rebuilds only the changed parts, leading to much faster hot module replacement (HMR) and overall development performance.

- **Build Size and Speed:** CRA tends to have larger bundle sizes by default, as it includes many dependencies and configurations designed to work out of the box for a wide range of projects, **while** Vite, being more modern, produces smaller and faster builds by optimizing the dependencies it uses, reducing bundle sizes, and using advanced techniques like ES module imports.

- **Configuration Flexibility:** CRA provides a zero-config approach, which is great for beginners who want to start coding immediately, but it can be limited if you need to customize the build configuration, **while** Vite is highly configurable and allows developers to easily adjust settings for advanced needs like custom plugins or specific build optimizations.

- **Development Experience:** CRA provides a stable and familiar development environment that is ideal for newcomers, **while** Vite offers a more modern development experience with instant updates on file changes, better TypeScript support, and enhanced error reporting, making it ideal for more experienced developers looking for an efficient workflow.

A simple tabular representation, comparing further CRA and Vite Applications.

Feature	CRA	Vite
• Setup Time	Moderate	Very Fast
• Build Time	Slower	Very Fast
• HRM	Slower	Lightening Fast
• Configuration Flexibility	Less Flexible	Highly Flexible
• Initial Bundle Size	Larger	Smaller
• Development Experience	Standard React Experience	Modern and Enhanced

Personal Conclusion

Both **Create React App (CRA)** and **Vite** are great tools for setting up a React project.

Use CRA if you need a simple, opinionated starter that just works out of the box.

Choose Vite for a faster, more modern build experience, especially if you're working with newer technologies and want quicker builds and faster development speed.

No matter which tool you choose, you'll be well-equipped to build amazing React applications!

I hope you found this material useful and helpful.

Remember to:

Like

Save for future reference

&

Share with your network, be helpful to someone 👌

kodemaven-portfolio.vercel.app

Hi There!

Thank you for reading through

Did you enjoy this knowledge?

💼 Follow my LinkedIn page for more work-life balancing and Coding tips.

🌐 LinkedIn: Oluwakemi Oluwadahunsi

kodemaven-portfolio.vercel.app