Introduction to React.js

Understanding:

React.js is a JavaScript library developed by Facebook to build interactive and efficient user interfaces. React.js allows developers to create reusable UI components, as well as facilitating efficient state management and rendering. With its declarative approach, React makes code easier to predict and debug. Additionally, React uses Virtual DOM, which allows UI updates to be done quickly without having to reload the entire page.

Laptop Requirements for Development with React.js:

To develop applications using React.js smoothly, the laptop used should meet several technical requirements.

- The recommended operating system is Windows 10 or later, macOS Mojave (10.14) or later, or Linux with the latest distribution.
- Minimum processor is Intel Core i3 or equivalent, but Intel Core i5 or higher is recommended.
- Minimum RAM is 4GB, but 8GB or more is recommended for better performance.
- Minimum storage of 20GB free space, with SSD highly recommended.
- The minimum screen resolution is 1366 x 768 pixels, with 1920 x 1080 pixels or higher being the best choice.

Apart from hardware specifications, some supporting software is also required.

- The latest version of Node.js (LTS recommended) along with npm (Node Package Manager) must be installed, which can be downloaded from nodejs.org.
- A code editor such as Visual Studio Code is highly recommended, although other text editors that support JavaScript and JSX can also be used, such as Atom, Sublime Text, or Notepad++.
- A modern browser such as Google Chrome or Firefox is required to support web developer tools.

To prepare the development environment, make sure Node.js and the code editor are installed, and the operating system and software are updated to the latest versions. An internet connection is required to download the required packages and dependencies. By fulfilling these requirements, students will be ready to take part in the React.js practicum smoothly and effectively.

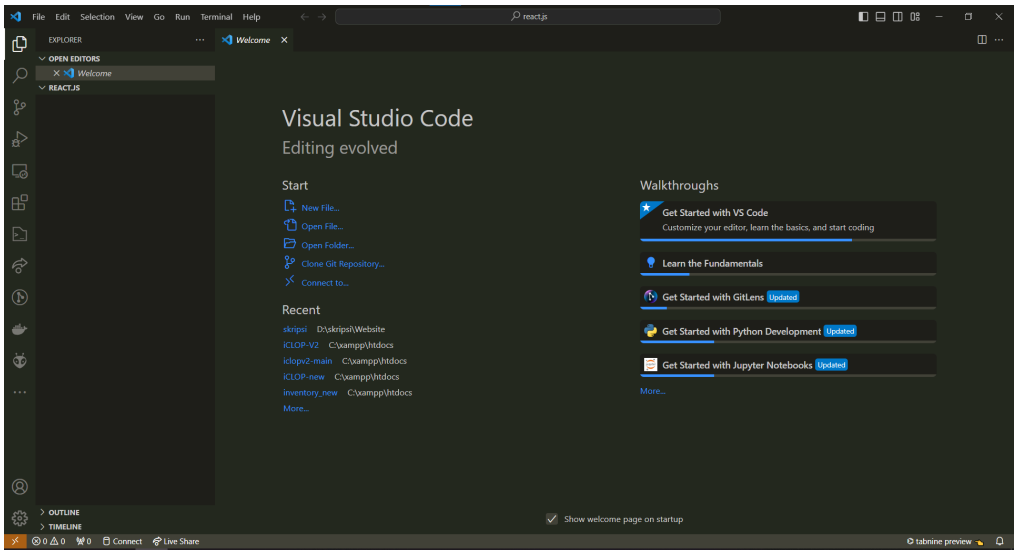
Conclusion:

In this practicum, students will learn to create basic React.js applications, starting from project setup to creating simple interactive components. This practicum provides an understanding of how to create and manage components in React, as well as how to manipulate state and event handling to create responsive and interactive applications. Students are also expected to be able to make further modifications and developments based on the understanding they have gained.

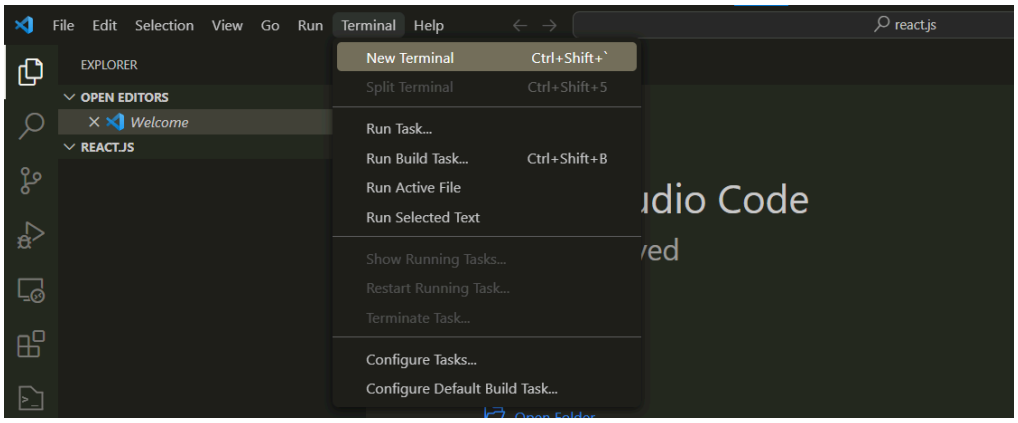
Steps to Create a React.js Project:

1. Create a new folder with the name react.js.

2. Open the notepad that you use, such as: notepad++, Visual Studio Code, etc.



3. Next open the terminal.



4. Once the terminal opens, enter the following command to create a React.js project.

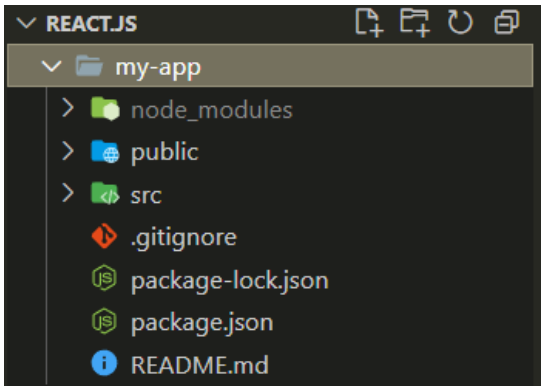
```
PS D:\react.js> npx create-react-app my-app
```

Naming my-app adapt to the name of the application you created.

```
PS D:\react.js> npx create-react-app my-app
npm WARN config global `--global`, `--local` are deprecated. Use `--location=global` instead.

Creating a new React app in D:\react.js\my-app.

npm WARN config global `--global`, `--local` are deprecated. Use `--location=global` instead.
Installing packages. This might take a couple of minutes.
Installing react, react-dom, and react-scripts with cra-template...

npm WARN config global `--global`, `--local` are deprecated. Use `--location=global` instead.
[##################] \ reify:@types/node: http fetch GET 200 https://registry.npmjs.org/@types/node/-/node-20.11.20.tgz 11941ms (cache miss)
```

Wait for the React.js project creation process above to complete.

5. After the React.js project has been successfully created, there will be a folder *my-app* which contains the react project files.



6. Then enter the following command to enter the React.js project folder named *my-app*.

```
PS D:\react.js> cd my-app
```

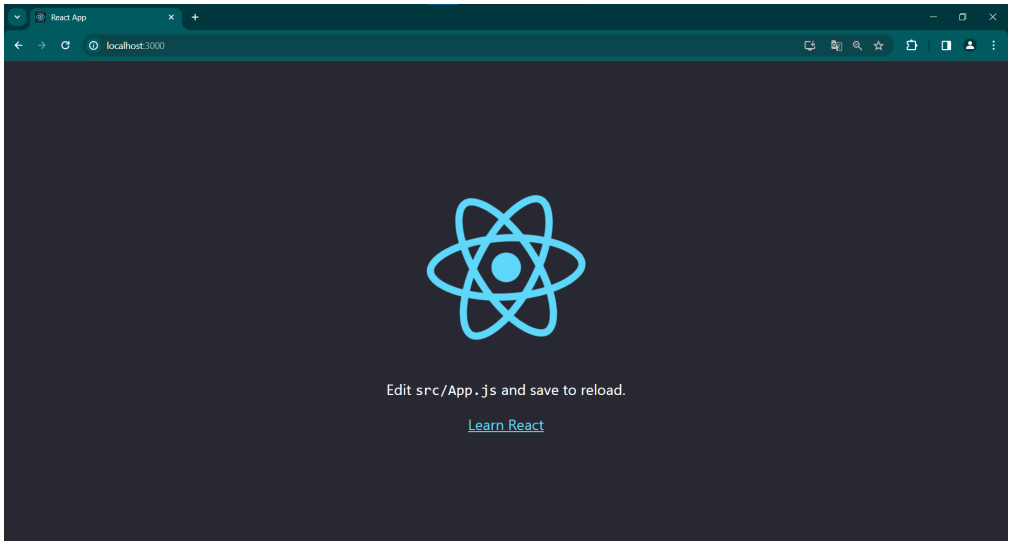
7. Next, if you want to run the React.js application, do the following command.

```
PS D:\react.js\my-app> npm start
```

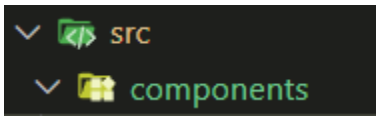
If successful, it will appear in the terminal as follows

```
Compiled successfully!

You can now view my-app in the browser.

  Local:            http://localhost:3000
  On Your Network:  http://192.168.1.2:3000

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

webpack compiled successfully
```

In the picture above there is the local host IP and the local network IP that you are using. And a welcome page will appear in your browser.

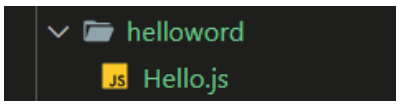


8. Next, create interactions in the form of simple buttons and alerts. Create a components folder in the src folder.



9. Next, create a new folder called helloword in the components folder.

10. After creating the folder, create a file called Hello.js in the helloword folder.



11. Then fill the Hello.js file with the following code.

```
tabnine: test | explain | document | ask
export default function App() {
    const handleHelloWorld = () => {
        alert('Hello!');

    };
    return (
        <Toolbar
        onHelloWord={handleHelloWorld}
        />
    );
}

tabnine: test | explain | document | ask
function Toolbar({onHelloWord}){
    return (
        <div>
            <Button onClick={onHelloWord}>HelloWord</Button>
        </div>
    );

}

tabnine: test | explain | document | ask
function Button({ onClick, children }) {
    return (
      <button onClick={onClick}>
        {children}
      </button>
    );
  }
```

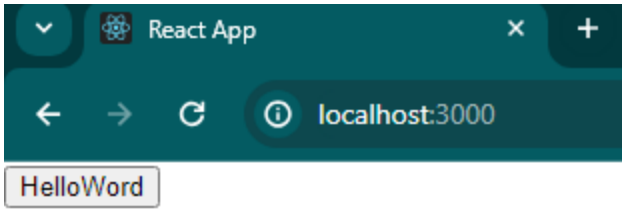
12. Next, import the Hello.js file into App.js and declare Hello.js so that the returns to the Hello.js file appear on the React App page.

```
import HelloWord from "./components/helloword/Hello";
```

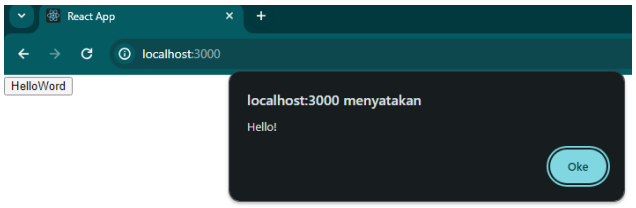
```
< HelloWord/>
```

Every time you add new components, you must import and declare your components as in the image above.

13. The browser will automatically open and then a page like the following will appear.



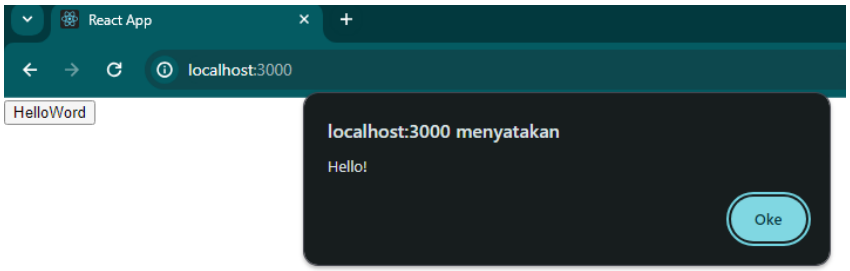
14. Then, if the button is clicked, an alert will appear as follows.



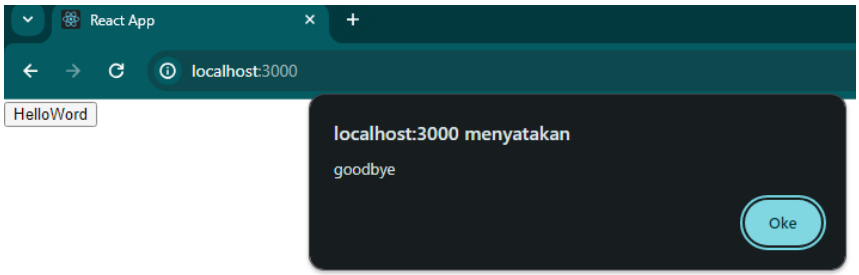
Test Case

Modify the button above when the alert appears and click "Okay" then the alert "GoodBye!" Like the following:

Before clicking "OK"

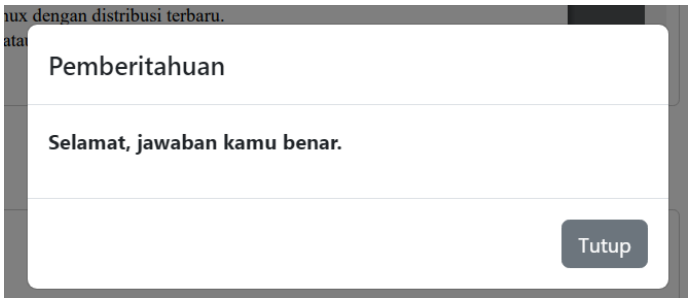


After clicking "OK"



If you have uploaded the Hello.js file into iCLOP:

If the uploaded file matches the practical commands above, it will appear as follows:



If the file does not match the command above, it will appear as follows: