

## Recap Assignments:

### Objective:

- Be able to understand the project structure of react app create by create-react-app or vite
- Understand the structure of function components.
- Be able to create react's components.
- Know how to use JSX.
- Be able to understand and use react's useState.

### Init React Application

There are 2 ways that you can create your react application

1. Create React Application via create-react-app

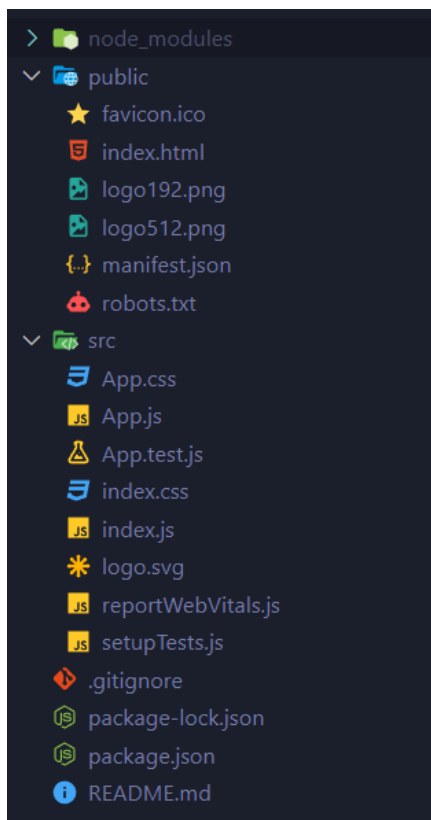
```
$ npx create-react-app <your-project-name>
```

2. You can use vite to create your react application (which is faster)  
(Recommended)

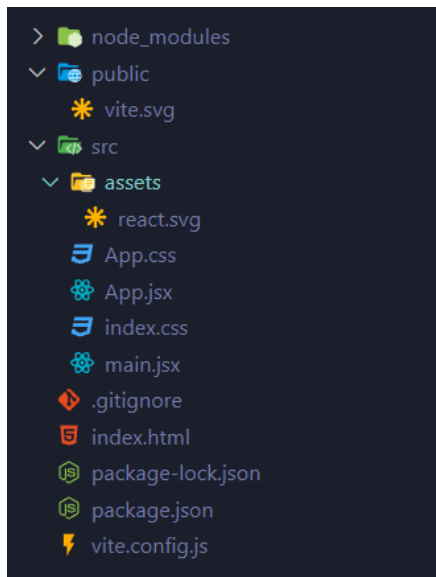
```
$ npm create vite@latest
```

## Project Structure

### 1. Project structure created by create-react-app



### 2. Project structure created by vite



**What can be remove from these template code?**

**(Please remove template code before you start coding/developing below are basic example how to remove template both create-react-app and vite)**

**(If you're using vite skip to page 5)**

### 1. create-react-app

- everything in public folder can be remove **except** index.html

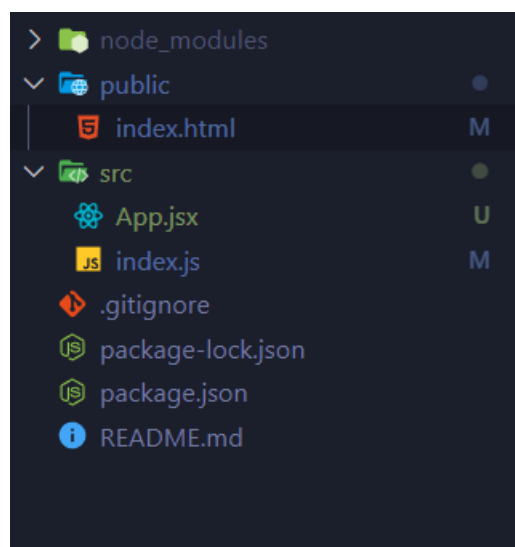
- everything in src folder can be remove **except** for these file

1. index.js

2. App.js

- Start developing with only index.js and App.js are recommended

After you delete the template code, your project structure should look like this.



NOTE: App.js change to App.jsx because it will trigger JSX snippet provided by VSCode. Every file that you want to write JSX should be in .jsx

Your index.html file should look like this

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <title>React App</title>
  </head>
  <body>
    <noscript>You need to enable JavaScript to run this app.</noscript>
    <div id="root"></div>
  </body>
</html>
```

Your App.jsx (Don't create App.jsx just rename your file)

```
function App() {
  return (
    <div className="App">
      <p>Hi!</p>
    </div>
  );
}

export default App;
```

Your index.js

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import App from './App';

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>
);
```

## 2. vite

### 2.1 Remove .svg from assets folder and public folder

### 2.2 Your index.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Vite + React</title>
  </head>
  <body>
    <div id="root"></div>
    <script type="module" src="/src/main.jsx"></script>
  </body>
</html>
```

### 2.3 Your App.jsx

```
function App() {
  return (
    <div className="App">
      <p>Hi!</p>
    </div>
  )
}

export default App
```

### 2.4 Your main.jsx


```
import React from 'react'
import ReactDOM from 'react-dom/client'
import App from './App'



ReactDOM.createRoot(document.getElementById('root')).render(
  <React.StrictMode>
    <App />
  </React.StrictMode>,
)
```


What ever you do **DO NOT REMOVE index.html, index.js/main.jsx**. It is your application's entry point.

**Your project should contain:**

- .gitignore for ignoring the file that you don't want your git to track
- package.json and package-lock.json contain script and what library, package or dependency your application is using/depend on. (you mainly look at package.json)
- node\_modules contain your library/dependency codes. This folder would appear after you run npm install (in case of create-react-app it's already there) NOTE: if you have a look at your .gitignore file node\_modules is in there because you don't want to push library code into your github repository. The node\_modules is HUGE. If you want to send this project to other do not add node\_modules when sending to them

 .gitignore

 package-lock.json  
 package.json

>  node\_modules

## Running your react application by using npm script:

1. create-react-app:

```
$ npm run start
```

2. vite:

```
$ npm run dev
```

If you're not sure what script can be run have a look at your package.json

```
{
  "name": "new-react-assigment-2-vite",
  "private": true,
  "version": "0.0.0",
  "type": "module",
  > Debug
  "scripts": {
    "dev": "vite",
    "build": "vite build",
    "preview": "vite preview"
  },
  "dependencies": {
    "react": "^18.2.0",
    "react-dom": "^18.2.0"
  },
  "devDependencies": {
    "@types/react": "^18.0.28",
    "@types/react-dom": "^18.0.11",
    "@vitejs/plugin-react": "^3.1.0",
    "vite": "^4.2.0"
  }
}
```

1. scripts: what script can be run with your project.
2. dependencies: library/package that your application depend on (those packages that installed via npm install <package-name>)
3. devDependencies: library/package that is for development of your application (those packages that installed via npm install <package-name> --save-dev)

## Additional Setup (reset stylesheet)

In your src create index.css that contain:

```
*, *::before, *::after {  
    box-sizing: border-box;  
    margin: 0;  
}  
  
body {  
    background-color: black;  
    color: white;  
    font-family: Arial, Helvetica, sans-serif;  
}
```

And at the top of your main.jsx or index.js add this line:

```
import './index.css';
```

(this is how you import CSS to your component)



## React Function Component

```
function App() {  
  
  return (  
    <div className="App">  
      <p>Hi!</p>  
    </div>  
  )  
}  
  
export default App
```

Your react component is a function.

Normally we named the file with the same name as the component.

Ex. App.jsx has App function or we called App component

### React component compose of:

- function name (component name)
- your return statement that has JSX
- an export default to export your function component to be use outside the file (without this you cannot use your component outside your file)

**To use your component in another file,** import the component and use  
<YourComponentName />

```
import Test from "../components/Test"  
  
function App() {  
  
  return (  
    <div className="App">  
      <Test />  
    </div>  
  )  
}  
  
export default App
```

**NOTE: Before asking mentors, try searching internet for your problem. Searching internet is essential skill for developer. Ex. Question “How do I import CSS into my react component?”**

## Assignment 1:

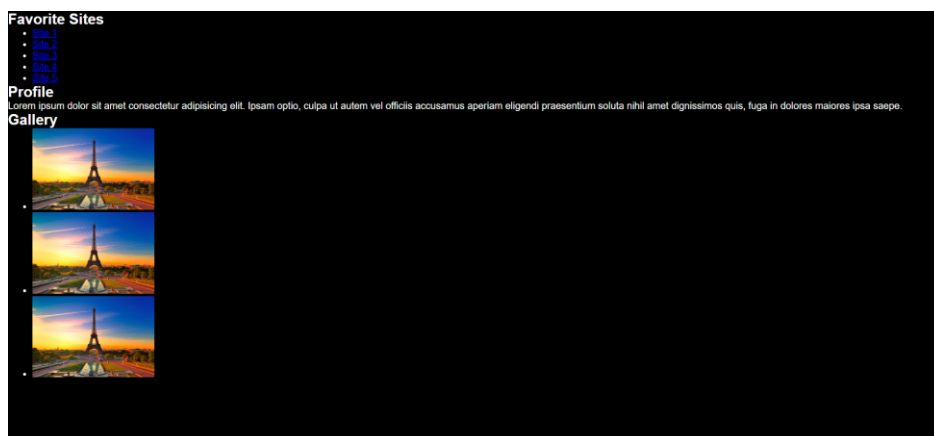
### Objective:

- Understand the structure of function components.
- Be able to create react's components.
- Know how to use JSX.

Create another folder in src folder called “components” where you’re going to store your components in this assignments.

1. Create FavoriteSites component contains `<h2>` tag that has “Favorite Sites” as its text and unordered-list with at least 5 links that is your favorite websites on the internet. All of them are wrapped inside `<section>` tag.
2. Create Profile component contains `<h2>` tag that has “Profile” as its text and paragraph that has information about yourself. All of them are wrapped inside `<section>` tag.
3. Create Gallery component contains `<h2>` tag that has “Gallery” as its text and unordered-list with at least 3 pictures with width set to 200. All of them are wrapper inside `<section>` tag. (use `<img width={200} src="">`)
4. Import all the components to render inside your App component.

Your result should look similar to this picture.



## Assignment 2:

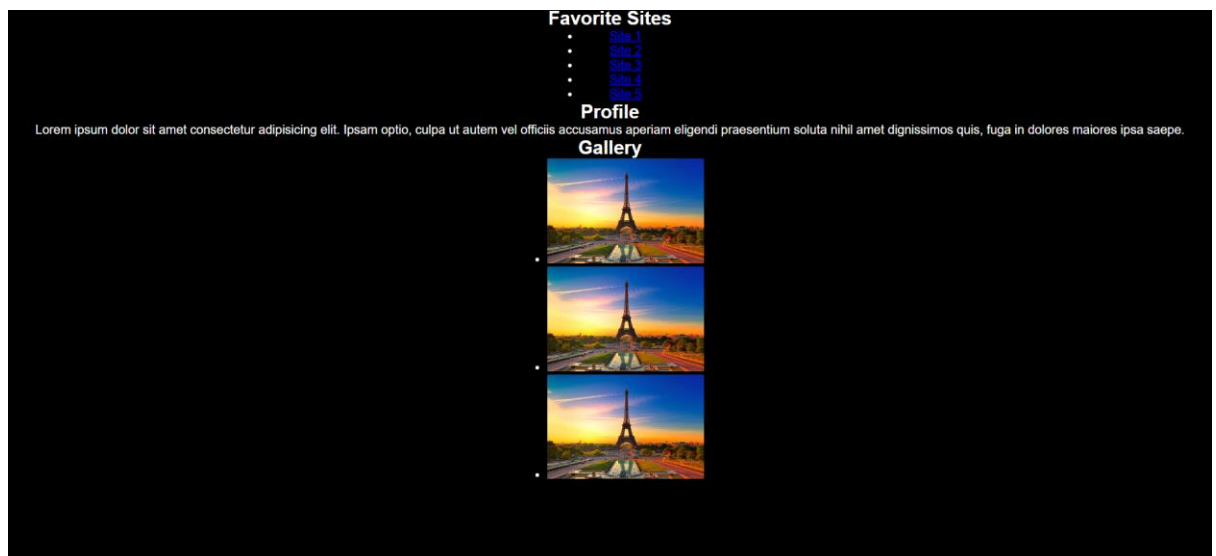
### Objective:

- Be able to apply previous knowledge from HTML/CSS class.

In your src folder, create “styles” folder for storing your stylesheet for each component you created in this assignment.

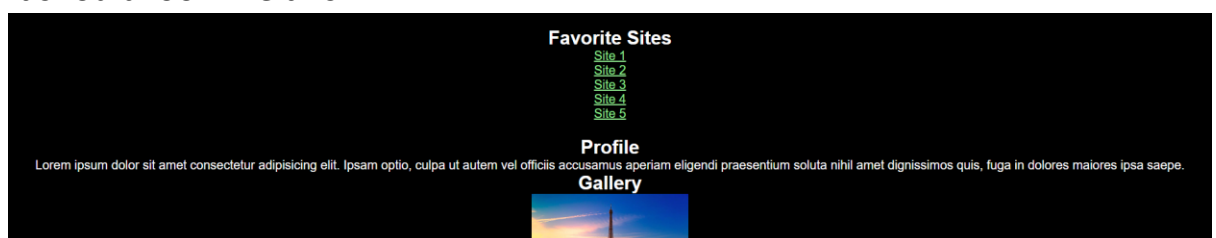
1. In your App.js, <FavoriteSites />, <Profile />, <Gallery /> components should have tag that wrapped around them in order to apply CSS to set them to the center of your screen and the text should be set to center too. (you may use className or id, it’s up to you!)

It should look like this.

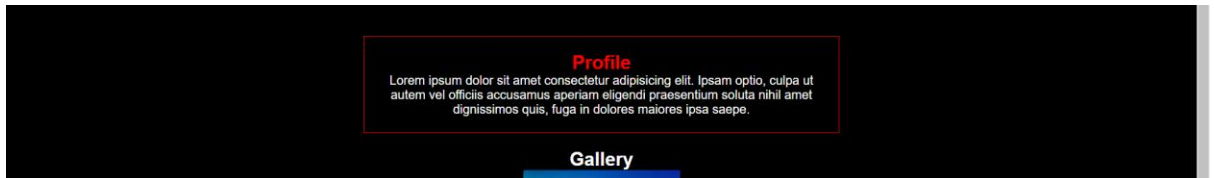


2. In FavoriteSites component, your <ul> shouldn’t have padding and your list items shouldn’t have dot in front of them. The link should be “lightgreen” color and your visited link too. When hover over the link the color should be “red”. Your FavoriteSites component should have margin of 20 pixels.

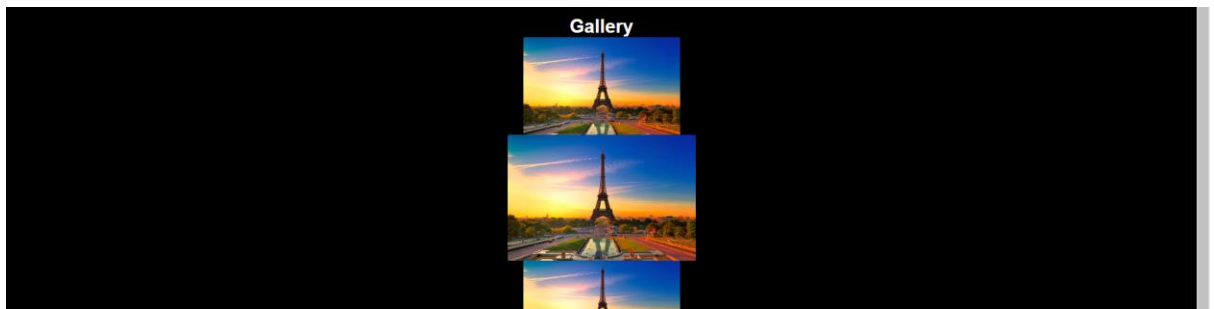
It should look like this.



3. Your Profile component should have 40% of your screen width. With margin and padding of 20px. When hovering over the Profile component, the red border and its <h2> should appear in red color smoothly by using CSS transition within 1 second. It should look like this.



4. Your image in Gallery component should scale by 1.2 times when hovering and scale smoothly by using CSS transition within 1 second. Also add margin of 20 pixels to the gallery component. It should look like this.



## React useState hook

Components often need to change what's on the screen as a result of an interaction. Typing into the form should update the input field, clicking "next" on an image carousel should change which image is displayed, clicking "buy" should put a product in the shopping cart. Components need to "remember" things: the current input value, the current image, the shopping cart. In React, this kind of component-specific memory is called state.

(From the react document <https://react.dev/learn/state-a-components-memory>)

(They recently change their document to a newer version. It is modern and simple to understand. Check it out)

### 1. useState() hook

```
const [state, setState] = useState();
```

The useState() function return 2 values:

1. state the actual value of your state. (or getter)
2. setState, a setter for your state

2. You can name both values what ever you like, conventionally we named them:

```
const [<value-name>, set<value-name>] = useState();
```

3. You can set default value to your state by providing it inside the useState's parameter

```
const [state, setState] = useState("Default value");
```

4. Using state inside JSX, you have to wrap them inside { }. In the example below, the component going to render <div>Default value</div>

```
const [state, setState] = useState("Default value");

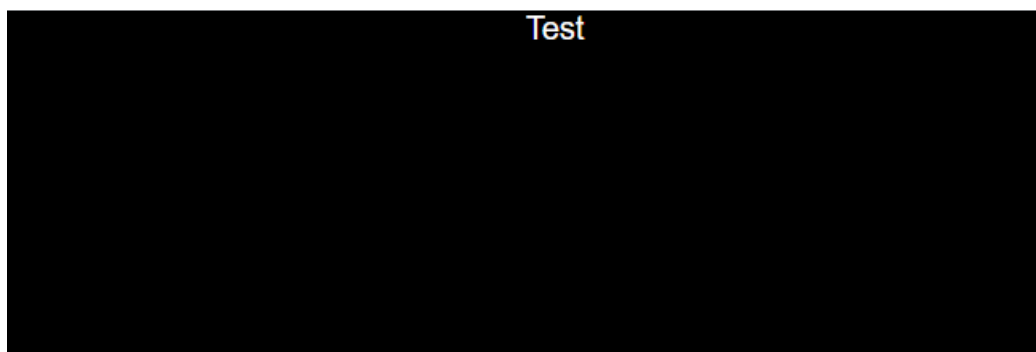
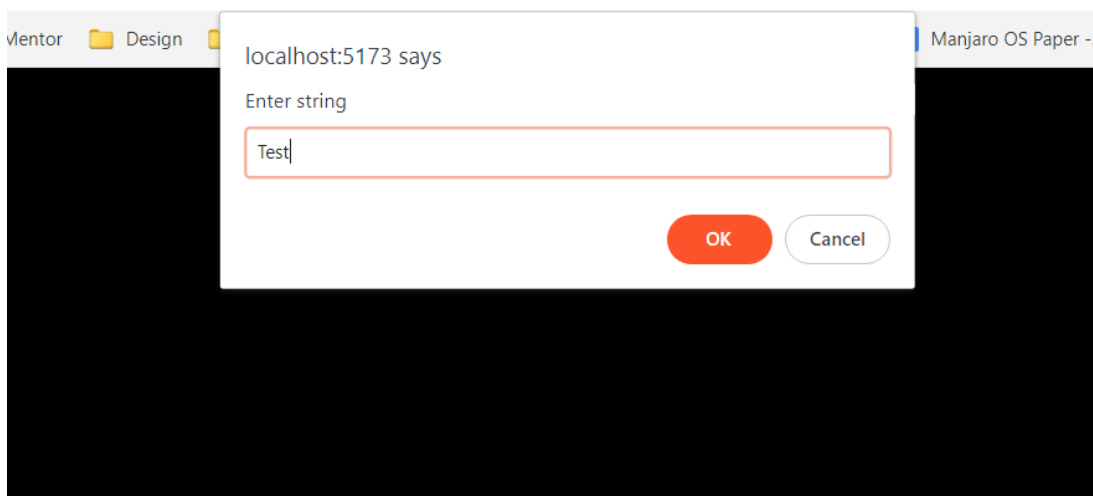
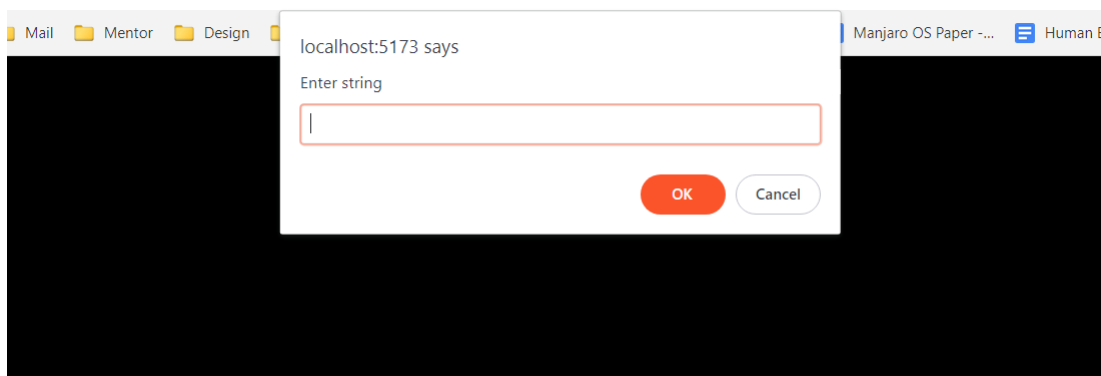
return (
  <div>
    {state}
  </div>
)
```

5. Using setter function, In the example below, when user click the text "Default value" the prompt will show up and asking for string. After user enter the string the prompt() function will return user's string and set the state to be user's string

```
const [state, setState] = useState("Default value");

return (
  <div onClick={() => {
    setState(prompt("Enter string"));
  }}>
    {state}
  </div>
)
```

There is a `<div>` tag with “Default value” at the center of the screen when you clicked it, it will prompt user to enter input. After entering the input the `<div>` value will change to be user’s input.



NOTE: You cannot do much with state without learning about props and form in react. We will get there soon!

**NOTE: you're going to search the internet a lot in this topic if you're not understand.**

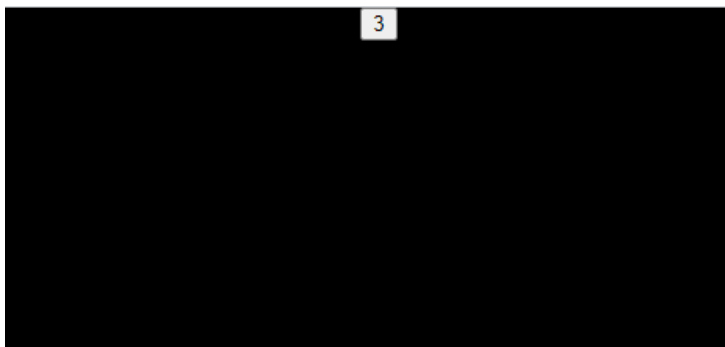
### **Assignment 3:**

#### **Objective:**

- Be able to understand and use react's useState.

We will using the same project from assignment 2 but remove FavoriteSites, Profile, Gallery component from App component so you can see the result clearly.

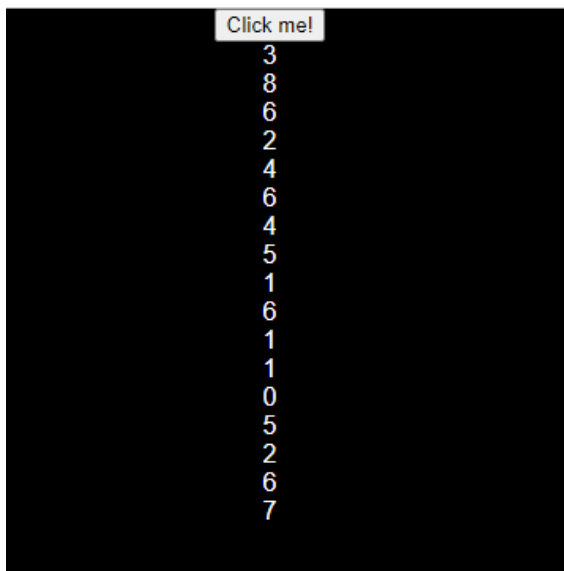
1. Create StateAssignmentOne component inside components folder, the component should wrapped with `<section>` tag. Inside `<section>` tag you should have a button. Its text should be number of time the button has been clicked. The button should listen to `onClick` event when it's click the state should change (number of clicks)  
It should look like picture below and the number increase when click the button.



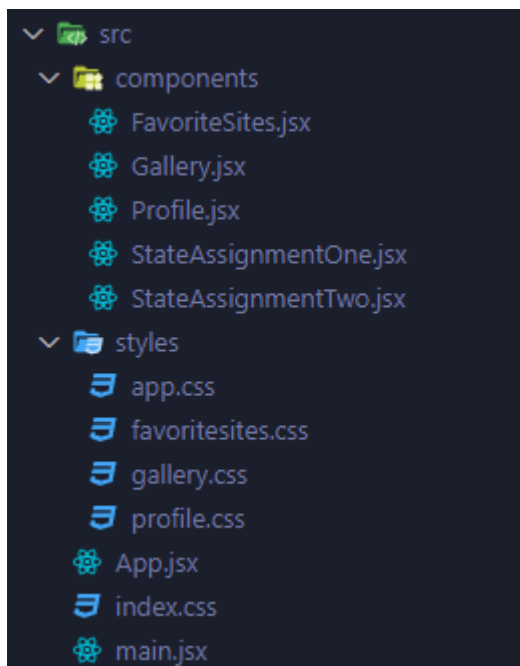


2. Create StateAssignmentTwo component inside component folder. This component has state that store an array of numbers.
- The component should wrap with `<section>` tag.
  - Inside `<section>` tag you should have a button that said, "Click me!".
  - When the button is clicked it change the state of component by adding a random integer number of 0-9 to the array.
  - Under the button tag you should have `<ul>` tag that take your array state to map with `<li>` tag to render random integer number to your screen.

Result:



Your project structure should look like this picture:



Your App.jsx

```
App.jsx
src > App.jsx > ...
1  import FavoriteSites from "../components/FavoriteSites";
2  import Gallery from "../components/Gallery";
3  import Profile from "../components/Profile";
4  import StateAssignmentOne from "../components/StateAssignmentOne";
5  import StateAssignmentTwo from "../components/StateAssignmentTwo";
6  import "../styles/app.css";
7
8  function App() {
9    return (
10     <div className="App">
11       <FavoriteSites />
12       <Profile />
13       <Gallery />
14       <StateAssignmentOne />
15       <StateAssignmentTwo />
16     </div>
17   );
18 }
19
20 export default App;
```

**Finally:** Upload this project to your github repository