Date:

Experiment - 8

Aim:

Implement React Elements and Components

Description:

React Elements

- The **smallest building blocks** of a React application.
- Represent **UI elements** such as buttons, headings, paragraphs, or divs.
- Can be created using **React.createElement()** or **JSX syntax** (e.g., <h1>Hello</h1>).
- **Immutable** once created, meaning they cannot be changed after rendering.
- React Elements are responsible for **describing what should appear on the screen**.

React Components

- **Reusable UI pieces** that return **React Elements**.
- Help in structuring **complex UIs** by breaking them into smaller parts.
- Two types of components:
    - **Functional Components**: Defined as functions that return JSX, recommended for most cases.
    - **Class Components**: Defined using ES6 classes, primarily used when lifecycle methods are needed.
- Components **can be nested inside other components** to create a **hierarchical UI structure**.
- React components allow **code reusability, better maintainability, and efficient rendering**.

Program:

App.js

```
// Importing React and child components
import React from "react";
import Header from "./Header";
import Main from "./Main";
import Footer from "./Footer";

// Root component that holds the structure of the application
function App() {
    return (
        <div style={{ textAlign: "center", fontFamily: "Arial,
sans-serif" }}>
            {/* Header Component */}
            <Header />

            {/* Main Content Component */}
            <Main />

            {/* Footer Component */}
            <Footer />
        </div>
    );
}

// Export App component for use in index.js
export default App;
```

Header.js

```
// Importing React
import React from "react";

// Header Component to display the title
function Header() {
    return (
        <header>
            <h1>React Elements & Components</h1>
        </header>
    );
}

// Export Header component for use in App.js
export default Header;
```

Main.js

```
// Importing React and child components
import React from "react";
import WelcomeElement from "./WelcomeElement";
import FunctionalComponent from "./FunctionalComponent";
import ClassComponent from "./ClassComponent";

// Main Component that contains different types of elements
and components
function Main() {
    return (
        <main>
            {/* Using a React Element */}
            <WelcomeElement />
            {/* Using a Functional Component with props */}
            <FunctionalComponent name="Alice" />

            {/* Using a Class Component with props */}
            <ClassComponent name="Bob" />
        </main>
    );
}

// Export Main component for use in App.js
export default Main;
```

WelcomeElement.js

```
// Importing React
import React from "react";

// Creating a React element using React.createElement()
const WelcomeElement = () => {
    return React.createElement("h2", {}, "Welcome to React
Elements!");
};

// Export WelcomeElement for use in Main.js
export default WelcomeElement;
```

FunctionalComponent.js

```
// Importing React
import React from "react";

// Functional Component that receives props
function FunctionalComponent(props) {
    return <h3>Hello, {props.name}! This is a Functional
Component.</h3>;
}

// Export FunctionalComponent for use in Main.js
export default FunctionalComponent;
```

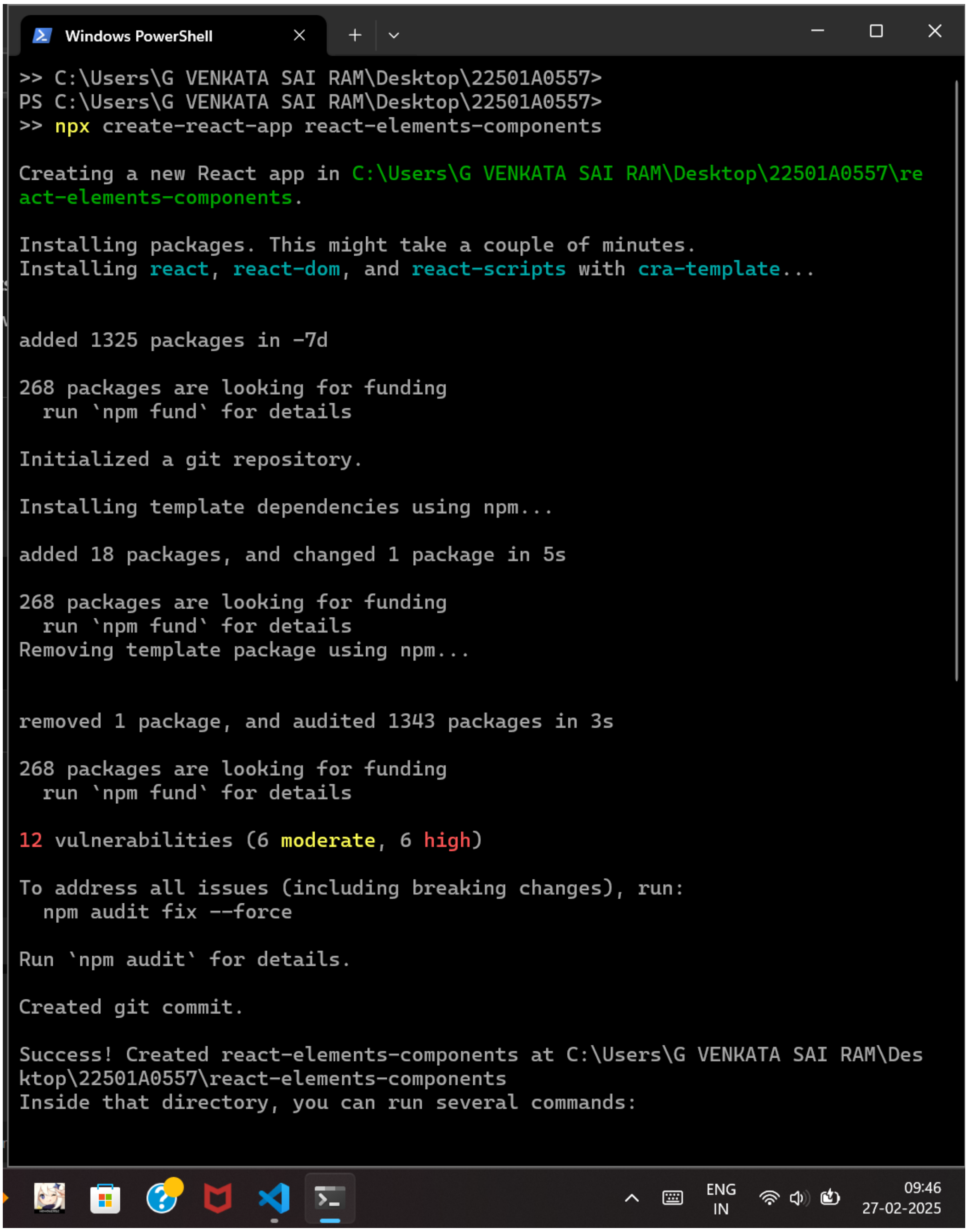
ClassComponent.js

```
// Importing React and Component class
import React, { Component } from "react";

// Class Component that receives props
class ClassComponent extends Component {
    render() {
        return <h3>Hello, {this.props.name}! This is a Class
Component.</h3>;
    }
}

// Export ClassComponent for use in Main.js
export default ClassComponent;
```

Footer.js

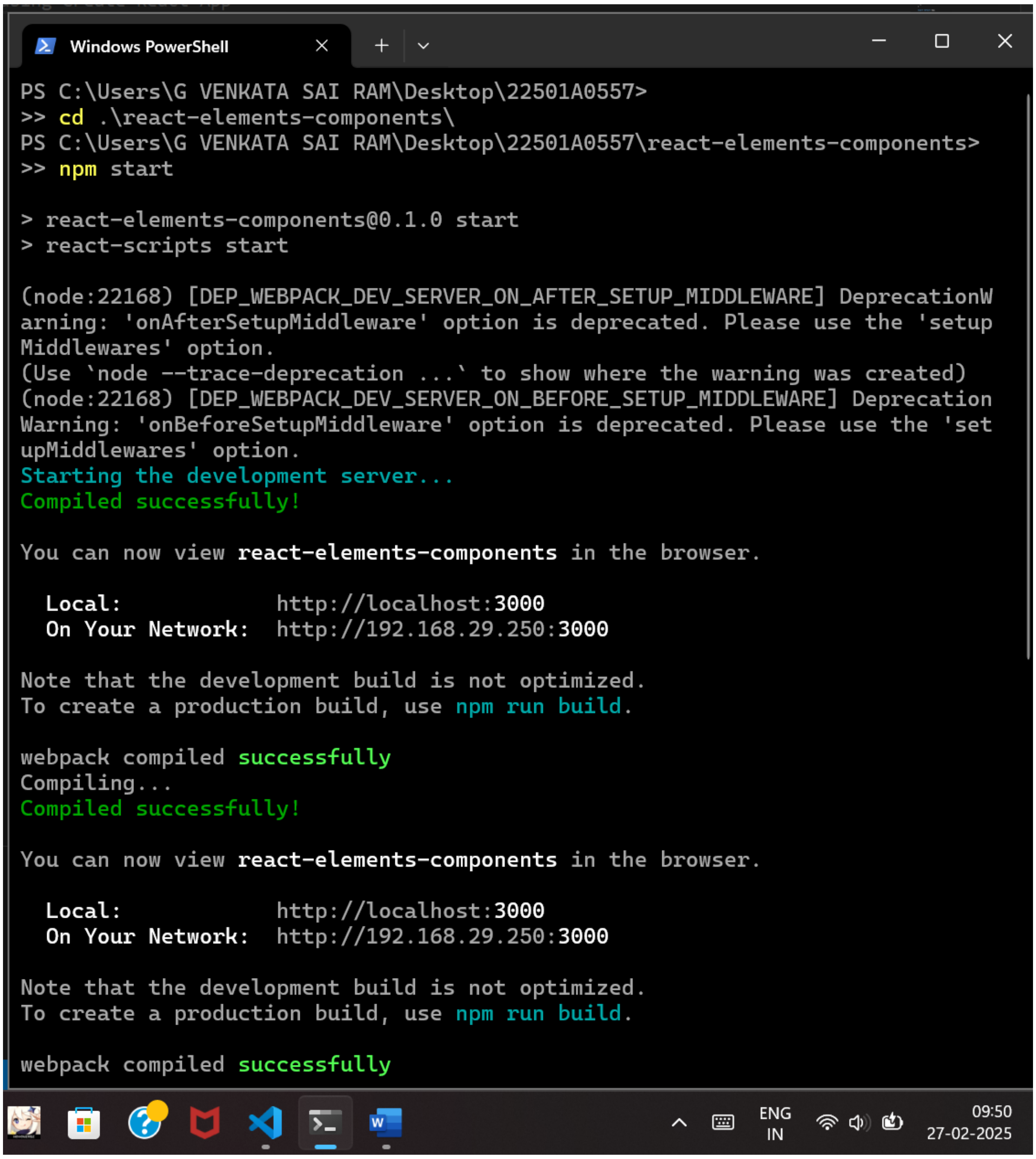
```
// Importing React
import React from "react";

// Footer Component to display the copyright notice
function Footer() {
    return (
        <footer>
            <p>© 2024 React Elements & Components</p>
        </footer>
    );
}

// Export Footer component for use in App.js
export default Footer;
```

Output:



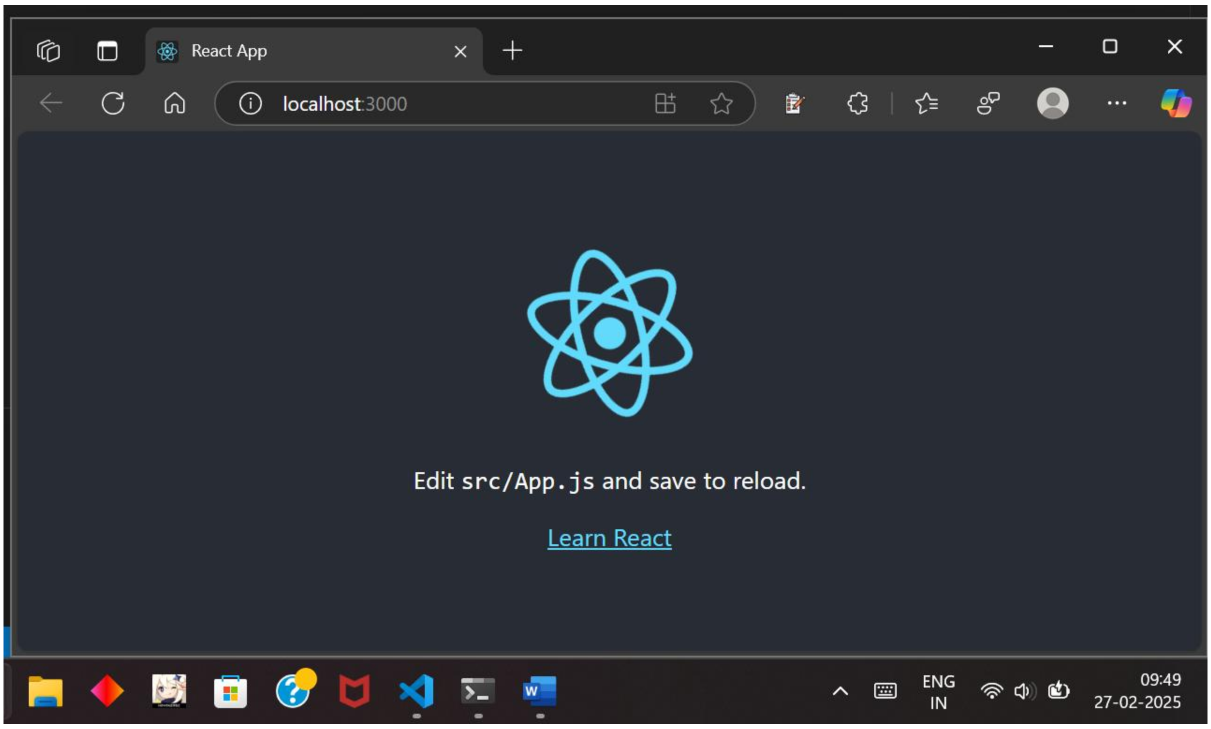
```
>> C:\Users\G VENKATA SAI RAM\Desktop\22501A0557>
PS C:\Users\G VENKATA SAI RAM\Desktop\22501A0557>
>> npx create-react-app react-elements-components

Creating a new React app in C:\Users\G VENKATA SAI RAM\Desktop\22501A0557\re
act-elements-components.

Installing packages. This might take a couple of minutes.
Installing react, react-dom, and react-scripts with cra-template...


added 1325 packages in -7d

268 packages are looking for funding
  run `npm fund` for details

Initialized a git repository.

Installing template dependencies using npm...

added 18 packages, and changed 1 package in 5s

268 packages are looking for funding
  run `npm fund` for details
Removing template package using npm...


removed 1 package, and audited 1343 packages in 3s

268 packages are looking for funding
  run `npm fund` for details

12 vulnerabilities (6 moderate, 6 high)

To address all issues (including breaking changes), run:
  npm audit fix --force

Run `npm audit` for details.

Created git commit.

Success! Created react-elements-components at C:\Users\G VENKATA SAI RAM\Des
ktop\22501A0557\react-elements-components
Inside that directory, you can run several commands:
```

Creating a react app using npx

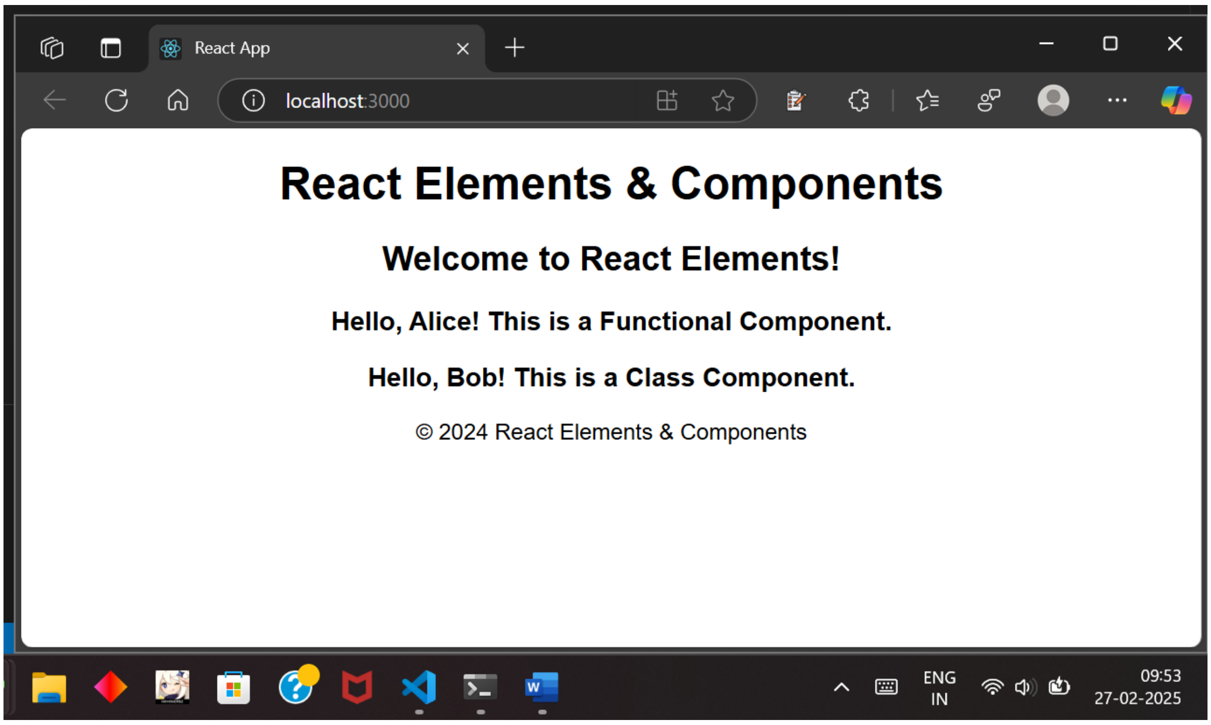


```
PS C:\Users\G VENKATA SAI RAM\Desktop\22501A0557>
>> cd .\react-elements-components\
PS C:\Users\G VENKATA SAI RAM\Desktop\22501A0557\react-elements-components>
>> npm start

> react-elements-components@0.1.0 start
> react-scripts start

(node:22168) [DEP_WEBPACK_DEV_SERVER_ON_AFTER_SETUP_MIDDLEWARE] DeprecationW
arning: 'onAfterSetupMiddleware' option is deprecated. Please use the 'setup
Middlewares' option.
(Use `node --trace-deprecation ...` to show where the warning was created)
(node:22168) [DEP_WEBPACK_DEV_SERVER_ON_BEFORE_SETUP_MIDDLEWARE] Deprecation
Warning: 'onBeforeSetupMiddleware' option is deprecated. Please use the 'set
upMiddlewares' option.
Starting the development server...
Compiled successfully!

You can now view react-elements-components in the browser.

  Local:            http://localhost:3000
  On Your Network:  http://192.168.29.250:3000

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

webpack compiled successfully
Compiling...
Compiled successfully!

You can now view react-elements-components in the browser.

  Local:            http://localhost:3000
  On Your Network:  http://192.168.29.250:3000

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

webpack compiled successfully
```

Starting the server using npm



Initially we can see this page by default



React Elements & Components

Welcome to React Elements!

Hello, Alice! This is a Functional Component.

Hello, Bob! This is a Class Component.

© 2024 React Elements & Components

Example page for React Elements & Components

Date:

Experiment - 9

Aim:

Develop a Single Page Application (SPA)

Description:

What is a Single Page Application (SPA)?

- A web application that dynamically updates content without reloading the entire page.
- Uses JavaScript frameworks like React, Angular, or Vue.js.
- Improves performance and user experience by fetching only necessary data.

Key Features of a SPA

- Uses client-side routing (e.g., React Router).
- Loads only required components when navigating between views.
- Reduces server requests by managing state on the client side.

Steps to Develop an SPA with React

1. **Install Node.js and npm**
    - Ensure Node.js is installed (node -v and npm -v to verify).
2. **Create a React App**
3. npx create-react-app my-spa
4. cd my-spa
5. npm start
    - This sets up a new React project and runs the development server.
6. **Install React Router for Navigation**
7. npm install react-router-dom
    - Enables client-side routing without full-page reloads.
8. **Define Routes in the Application**
    - Use BrowserRouter, Routes, and Route components from react-router-dom.

- Example routes: Home, About, Contact.

9. **Implement Components for Different Views**
   - Create separate functional components for each page.
   - Example: Home.js, About.js, Contact.js.
10. **Navigation with React Router**
    - Use Link components instead of <a> tags to prevent full-page reloads.
11. **Manage State Efficiently**
    - Use useState and useEffect for handling application state.
    - For complex state management, consider Redux or Context API.
12. **Deploy the Application**
    - Build the project using:
    - npm run build
    - Deploy to platforms like Vercel, Netlify, or Firebase Hosting.

Program:

Output: