Vidyavardhini's College of Engineering & Technology

Department of Computer Science and Engineering (Data Science)

EXPERIMENT ASSESSMENT

ACADEMIC YEAR 2025-26

Course: Web Computing and Networking Lab

Course code: CSL501

Year: TE SEM: V

Experiment No. 5
AIM:- React: Installation and Configuration, JSX, Components, Props, State, Forms, Events, Routers, Refs, Keys.
Name: Tejas Ravindra Bhatankar
Roll Number: 10
Date of Performance: 11/9/2025
Date of Submission: 18/9/2025

Evaluation

Performance Indicator	Max. Marks	Marks Obtained
Performance	5	
Understanding	5	
Journal work and timely submission.	10	

Total 20

Performance Indicator	Exceed Expectations (EE)	Meet Expectations (ME)	Below Expectations (BE)
Performance	5	3	2
Understanding	5	3	2
Journal work and	10	8	4

timely submission.			

Checked by

Name of Faculty : Mrs.Kranti Gule

Signature :

Date :18/9/2025

Vidyavardhini's College of Engineering & Technology

Department of Computer Science and Engineering (Data Science)

Experiment No. 5

Aim: React: Installation and Configuration, JSX, Components, Props, State, Forms, Events, Routers, Refs, Keys.

Objective:

1) To produce the most effective possible rendering performance.

2) Rather than being engaged on the whole web app, React JS allows a developer to break down the complex UI into simpler components.

Theory:

ReactJS is a declarative, efficient, and flexible JavaScript library for building reusable UI components. It is an open-source, component-based front end library responsible only for the view layer of the application. It was created by Jordan Walke, who was a software engineer at Facebook. It was initially developed and maintained by Facebook and was later used in its products like WhatsApp & Instagram. Facebook developed ReactJS in 2011 in its newsfeed section, but it was released to the public in the month of May 2013.

Today, most of the websites are built using MVC (model view controller) architecture. In MVC architecture, React is the 'V' which stands for view, whereas the architecture is provided by the Redux or Flux.

A ReactJS application is made up of multiple components, each component responsible for outputting a small, reusable piece of HTML code. The components are the heart of all React applications. These Components can be nested with other components to allow complex applications to be built of simple building blocks. ReactJS uses virtual DOM based mechanism to fill data in HTML DOM. The virtual DOM works fast as it only changes individual DOM elements instead of reloading complete DOM every time

Installation Reactjs on Windows:

Step 1: Install Node.js installer for windows. Once downloaded open NodeJS without disturbing other settings, click on the Next button until it's completely installed.

Step 2: Open command prompt to check whether it is completely installed or not type the command –>

node -v

If the installation went well it will give you the version you have installed

Step 3: Now in the terminal run the below command:

npm install -g create-react-app

It will globally install react app for you. To check everything went well run the command

create-react-app --version

If everything went well it will give you the installed version of react app

Step 4:Now Create a new folder where you want to make your react app using the below command:

mkdir newfolder.

Move inside the same folder using the below command:

cd newfolder (your folder name)

Step 5: Now inside this folder run the command –>

create-react-app reactfirst YOUR_APP_NAME

Step 6: Now open the IDE of your choice for eg. Visual studio code and open the folder where you have installed the react app newolder (in the above example) inside the folder you will see your app's name reactapp (In our example). Use the terminal and move inside your app name folder.Use command cd reactapp (your app name)

Step 7: To start your app run the below command :

npm start

Code & Output:

1) React Components, State , Props and Events

```
import React, { useState, useRef } from 'react';
import './App.css';
```

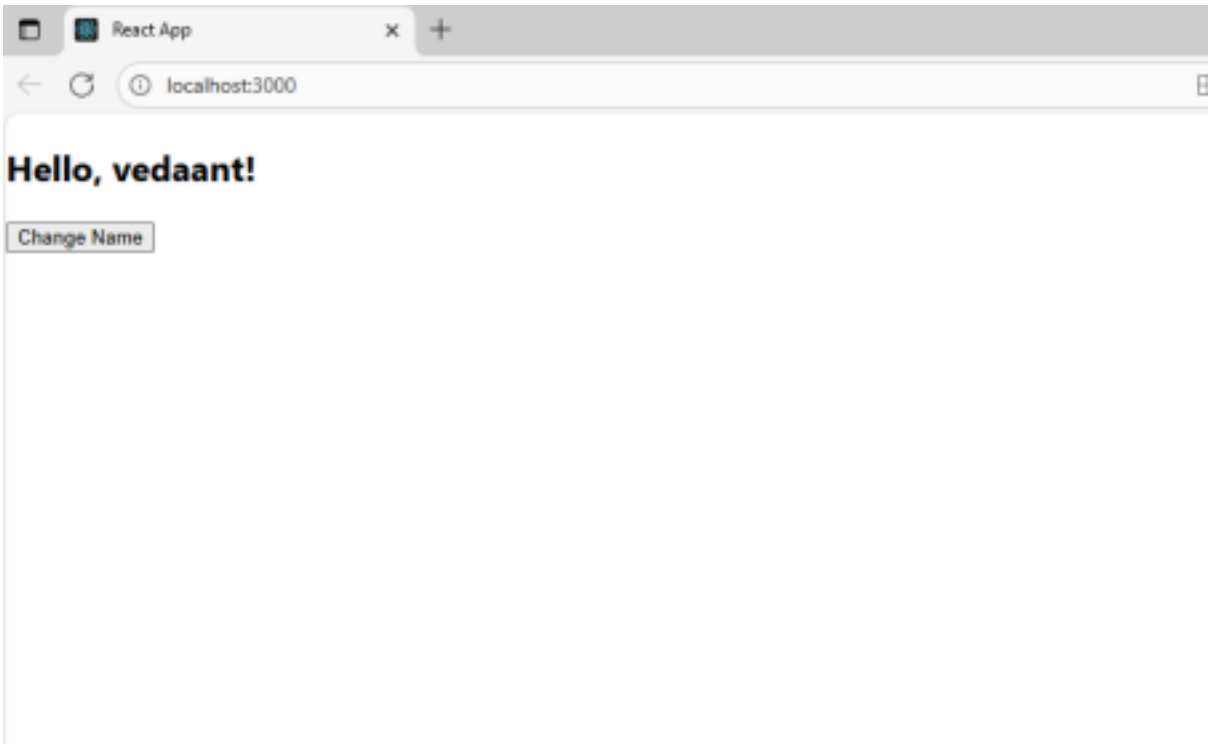


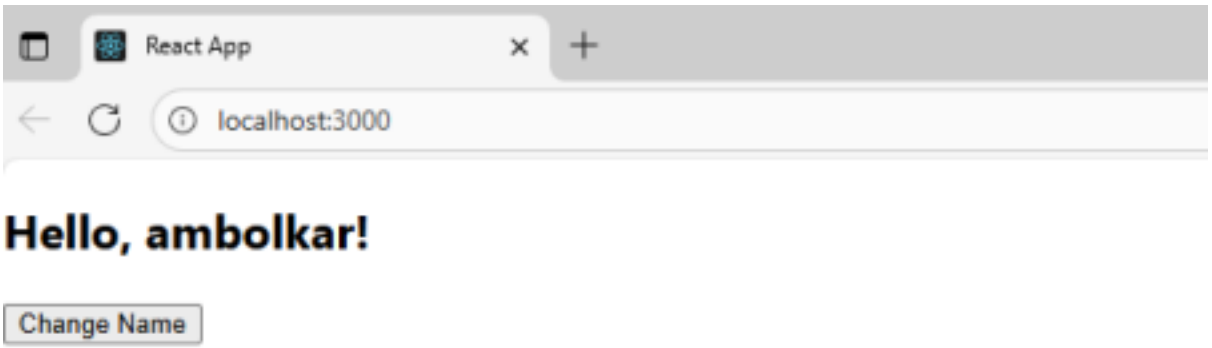
```
function App() {
 const [name, setName] = useState("vedaant");
 const nameInputRef = useRef(null);

 const handleChangeName = () => {
 const newName = nameInputRef.current.value.trim();
 if (newName) {
 setName(newName.toLowerCase());
 nameInputRef.current.value = '';
 }
 };

 const handleKeyPress = (e) => {
 if (e.key === 'Enter') {
 handleChangeName();
 }
 };

 return (
 <div className="app">
```

```
<div className="container">
<h1>Hello, {name}!</h1>

<div className="input-section">
<input
type="text"
ref={nameInputRef}
placeholder="Enter new name"
className="name-input"
onKeyPress={handleKeyPress}
/>

<button
className="change-button"
onClick={handleChangeName}
>
Change Name
</button>
</div>
</div>
</div>
);
}

export default App;
```

Output:

Vidyavardhini's College of Engineering & Technology
Department of Computer Science and Engineering (Data Science)





2) React Form and Router.

```
import React, { useState, useRef, useEffect } from 'react';

import './App.css';


function App() {

  const [name, setName] = useState("VEDAANT");

  const [welcomeMessage, setWelcomeMessage] = useState("");

  const nameInputRef = useRef(null);


  useEffect(() => {
```

```
  // Set initial welcome message
  setWelcomeMessage(`Welcome, ${name}!`);
}, [name]);

const handleUpdate = () => {
  if (nameInputRef.current.value.trim()) {
    setName(nameInputRef.current.value.trim());
    nameInputRef.current.value = '';
  }
};

const handleKeyPress = (e) => {
  if (e.key === 'Enter') {
    handleUpdate();
  }
};

return (
```

```
<div className="app">
  <header className="app-header">
    <h1>localhost:3000</h1>

    <div className="welcome-section">
      <div className="url-display">
        <span className="url-text">localhost:3000 says</span>
      </div>
```



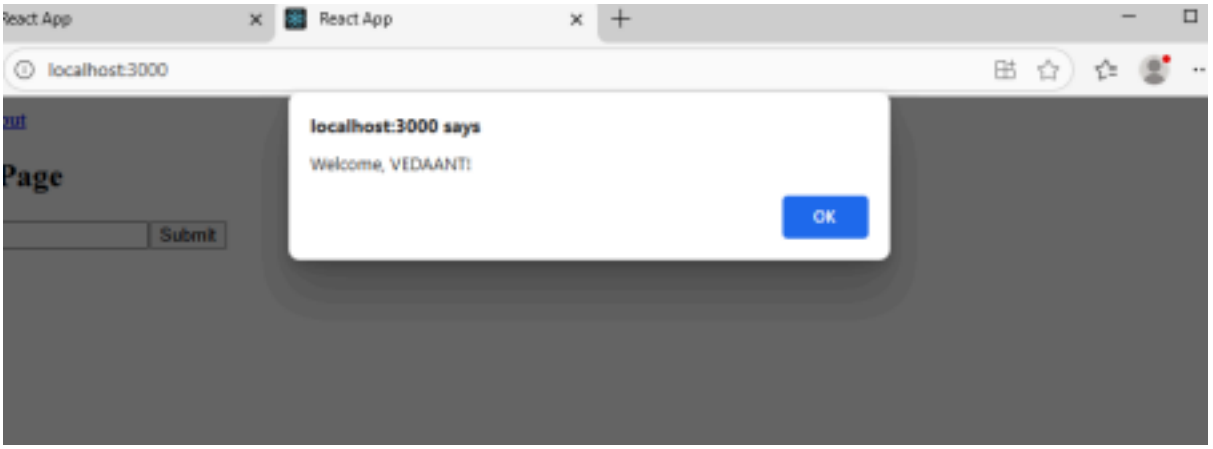
```
      <div className="message-display">
        <span className="welcome-text">{welcomeMessage}</span>
      </div>

      <div className="input-group">
        <input
          type="text"
          ref={nameInputRef}
          placeholder="Enter new name"
```

```
            className="name-input"
            onKeyPress={handleKeyPress}
          />

          <button
            className="update-button"
            onClick={handleUpdate}
          >
```



```
            Update Name
          </button>
        </div>
      </div>
    </header>
  </div>
 );
}

export default App;
```

Output:



3) React Refs

```
import React, { useState, useRef } from 'react';
import './App.css';

function App() {
 const [saying, setSaying] = useState("VEDAANT");
 const [savedSayings, setSavedSayings] = useState([]);
 const nameRef = useRef(null);

 const handleSave = () => {
 if (saying.trim() && nameRef.current.value.trim()) {
 const newSaying = {
 name: nameRef.current.value,
 saying: saying
```

```
};
setSavedSayings([...savedSayings, newSaying]);
nameRef.current.value = '';
setSaying('VEDAANT');
}
};

const handleKeyPress = (e) => {
if (e.key === 'Enter') {
handleSave();
```



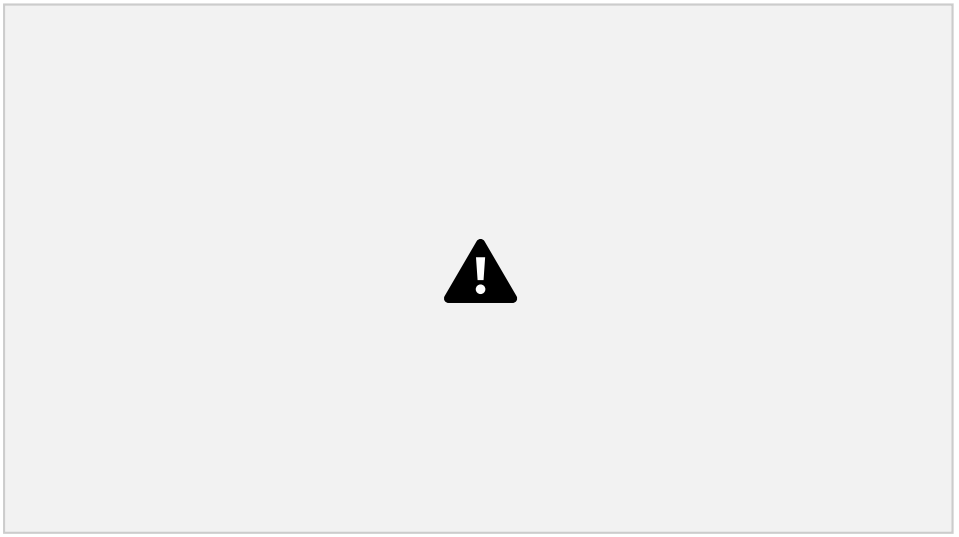
```
}
};

return (
<div className="app">
<header className="app-header">
<h1>React Ref Example</h1>

<div className="input-section">
<input
type="text"
ref={nameRef}
placeholder="Enter name"
className="name-input"
onKeyPress={handleKeyPress}
/>
```

```
<div className="saying-display">
<span className="saying-text">{saying}</span>
</div>

<button
className="save-button"
onClick={handleSave}
>
Save Saying
```



```
</button>
</div>

<div className="saved-sayings">
{savedSayings.map((item, index) => (
<div key={index} className="saying-item">
<strong>{item.name}</strong> says: {item.saying}
</div>
))}
</div>
</header>
</div>
);
}

export default App;
```

Output:



4) React keys

APP.JSX

```
import React from 'react';

import './App.css';
```

```
function App() {
 return (
 <div className="app">
 <header className="header">
 <div className="header-top">
 <span className="site-url">localhost:3000/about</span>
```



```
 <button className="info-button">View site information</button>
</div>

 <nav className="navigation">
 <h2 className="nav-title">Navigation Menu</h2>
 <ul className="nav-menu">
 <li className="nav-item">
 <a href="#home" className="nav-link">Home</a>
 </li>
 <li className="nav-item">
 <a href="#about" className="nav-link active">About</a>
 </li>
```

<li className="nav-item">

Services

<li className="nav-item">

Contact

<li className="nav-item">

Blog

</nav>

</header>

<main className="main-content">

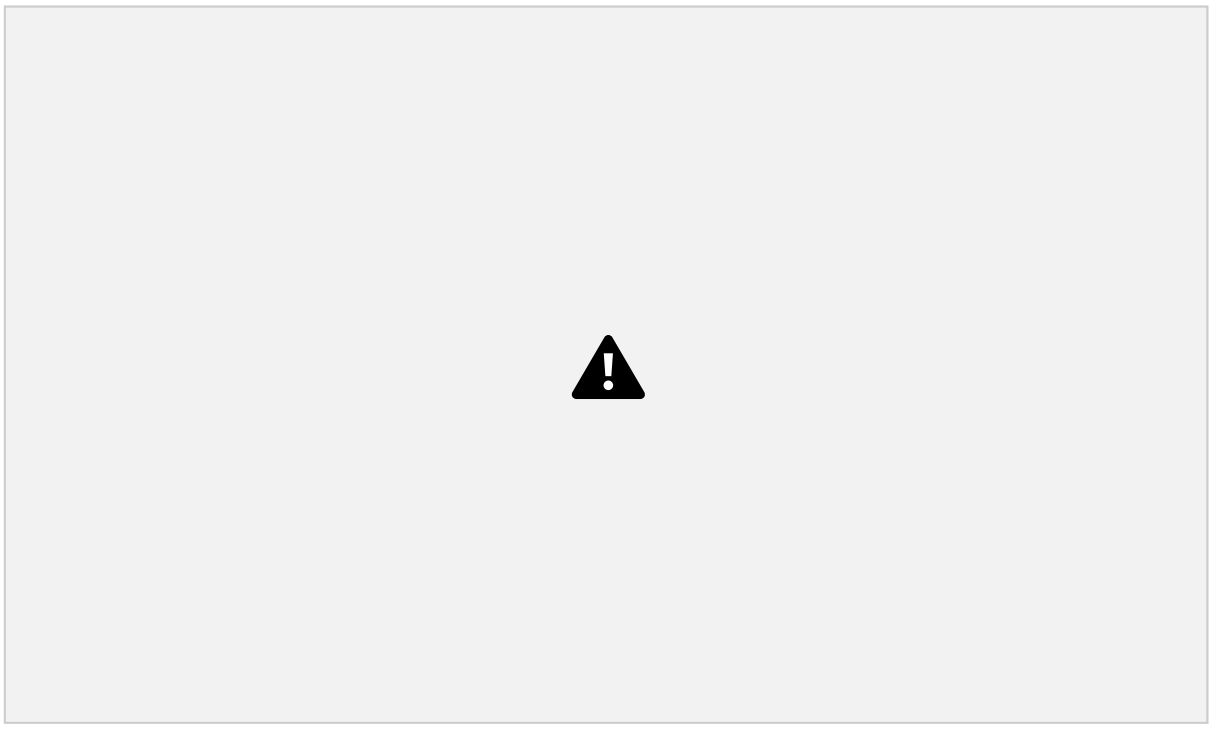
<h1>About Us</h1>

Welcome to our about page. This is a demonstration of a React application with navigation.

</main>

```
</div>

);

}


export default App;
```

Output



Conclusion:

1.How does the installation and configuration of React provide a foundation for building applications?

ANS=>Installing and configuring React sets up the environment needed to build modern, interactive web applications. Tools like **Node.js** and **npm (Node Package Manager)** allow developers to install React and related dependencies easily. The **create-react-app** command

provides a ready-made project structure with preconfigured settings for bundling, compiling JSX, and running a development server. This foundation ensures developers can focus on building components and features rather than worrying about setup complexities.

2.Why is JSX preferred over plain JavaScript for creating UI elements in React? ANS=>**JSX (JavaScript XML)** allows developers to write HTML-like syntax directly within JavaScript, making the UI code more readable and easier to visualize. It helps combine logic and structure in a single place, enabling developers to quickly define how components should look and behave. JSX also offers better error checking and integration with React's rendering system compared to plain JavaScript.

3.What role do components play in building reusable and modular applications? ANS=>Components are the **building blocks** of a React application. Each component represents a small, independent piece of the UI that can be reused across different parts of the app. By dividing the interface into modular components, developers can maintain, test, and

update parts of the application easily. This modularity improves scalability and code reusability.

4.How do props differ from state, and why are both essential in React development? ANS=>**Props (properties):** Used to pass data **from parent to child components**. They are **read-only** and help make components reusable by allowing customization.

- **State:** Represents **data that can change** within a component over time (e.g., user input, UI updates). It is **managed internally** by the component.

Both are essential because **props** enable data flow and customization, while **state** allows interactivity and dynamic behavior within the application.

CSL501: Web Computing and Network Lab