1. Write a ReactJS code to use all the states in the created Application.

Create index.html

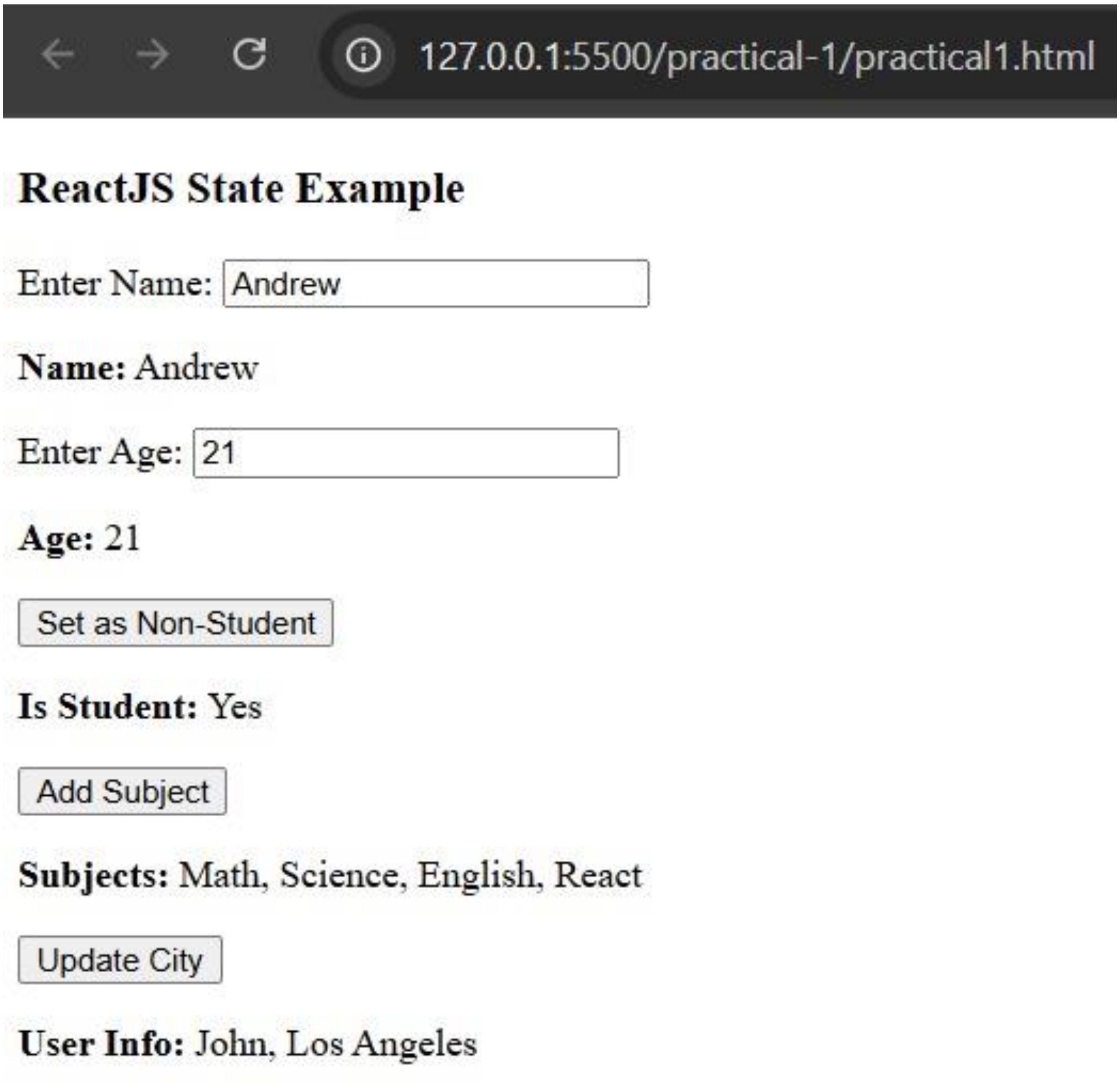
```
<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="UTF-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <title>ReactJS State Example</title>
 <script src="https://unpkg.com/react@18/umd/react.development.js"></script>
 <script src="https://unpkg.com/react-dom@18/umd/react-dom.development.js"></script>
 <script src="https://unpkg.com/babel-standalone@6/babel.min.js"></script>
</head>
<body>
 <h2>Practical - ReactJS State Management</h2>
 <div id="root"></div>
 <script type="text/babel" src="./index.js"></script>
</body>
</html>
```

<u>Create index.js</u>

```
const { useState } = React;
// React Functional Component
function StateExample() {
 // 1 String State
 const [name, setName] = useState("");
 // 2 Number State
 const [age, setAge] = useState(18);
 // 3 Boolean State
 const [isStudent, setIsStudent] = useState(false);
 // 4 Array State
 const [subjects, setSubjects] = useState(["Math", "Science", "English"]);
 // 5 Object State
 const [user, setUser] = useState({ name: "John", city: "New York" });
 return (
 <div>
 <h3>ReactJS State Example</h3>
 {/* String State */}
 <label>Enter Name: </label>
 <input type="text" value={name} onChange={(e) => setName(e.target.value)}
/>
 <p><b>Name:</b> {name}</p>
 {/* Number State */}
 <label>Enter Age: </label>
 <input type="number" value={age} onChange={(e) => setAge(e.target.value)}
/>
```

```
 <p><b>Age:</b> {age}</p>
 {/* Boolean State */}
 <button onClick={() => setIsStudent(!isStudent)}>
 {isStudent ? "Set as Non-Student" : "Set as Student"}
 </button>
 <p><b>Is Student:</b> {isStudent ? "Yes" : "No"}</p>
 {/* Array State */}
 <button onClick={() => setSubjects([...subjects, "React"])}>Add
Subject</button>
 <p><b>Subjects:</b> {subjects.join(", ")}</p>
 {/* Object State */}
<button onClick={() => setUser({ ...user, city: "Los Angeles" })}>Update
City</button>
 <p><b>User Info:</b> {user.name}, {user.city}</p>
 </div>
 );
}
// Rendering the Component
const root = ReactDOM.createRoot(document.getElementById("root"));
root.render(<StateExample />);
```

Output :-



ReactJS State Example

Enter Name: Andrew

Name: Andrew

Enter Age: 21

Age: 21

Is Student: Yes

Subjects: Math, Science, English, React

User Info: John, Los Angeles

2. Write a ReactJS code for to client-side form validation.

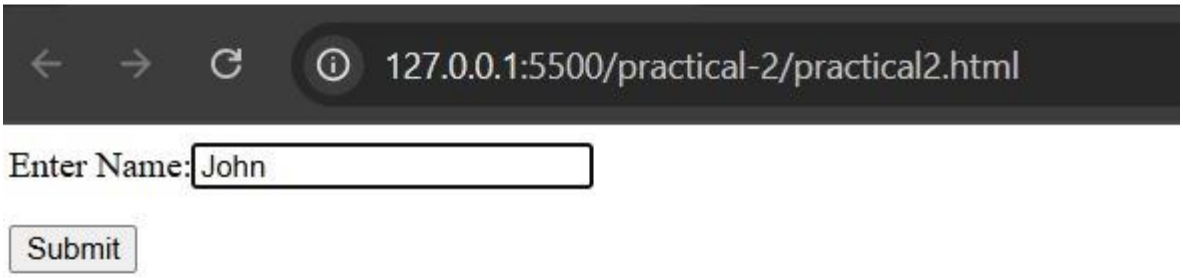
Create index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width,
initial-scale=1.0">
<title>React Form Validation</title>
<script
src="https://unpkg.com/react@18/umd/react.development.js
"></script>
<script src="https://unpkg.com/react-dom@18/umd/react-
dom.development.js"></script>
<script src="https://unpkg.com/babel-
standalone@6/babel.min.js"></script>
</head>
<body>
<h2>Practical 2: Client-side Form Validation</h2>
<div id="root"></div>
<script type="text/babel" src="formValidation.js"></script>
</body>
</html>
```

Create index.js

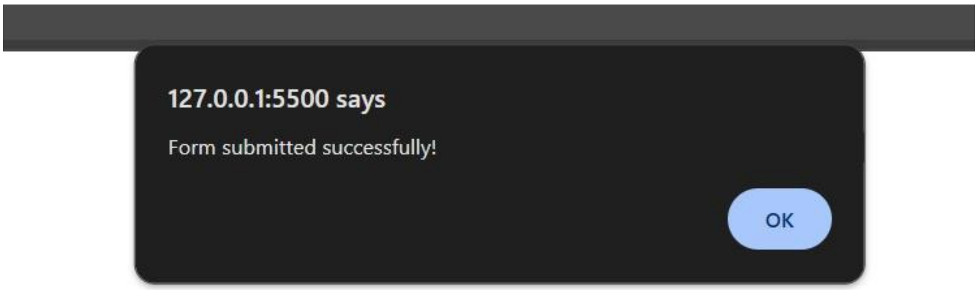
```
function FormValidation() {
const [name, setName] = React.useState("");
const [error, setError] = React.useState("");
function handleSubmit(e) {
e.preventDefault();
if (!name.trim()) {
setError("Name is required!");
} else {
setError("");
alert("Form submitted successfully!");
}
}
return (
<form onSubmit={handleSubmit}>
<label>Enter Name:</label>
<input type="text" value={name} onChange={(e) =>
setName(e.target.value)} />
<p style={{ color: "red" }}>{error}</p>
<button type="submit">Submit</button>
</form>
);
}
const root =
ReactDOM.createRoot(document.getElementById("root"));
root.render(<FormValidation />);
```

Output :-



127.0.0.1:5500/practical-2/practical2.html

Enter Name: John



127.0.0.1:5500 says

Form submitted successfully!

3. Write ReactJs code for Applying form Components.

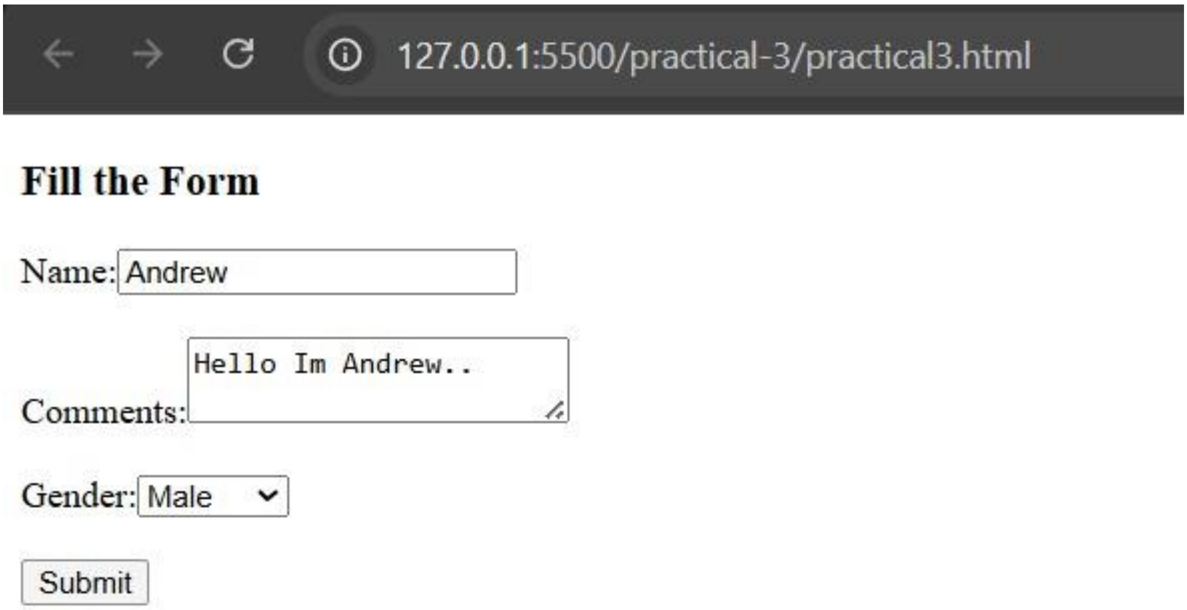
Create index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Practical 3: Applying Form Components in ReactJS</title>
<script src="https://unpkg.com/react@18/umd/react.development.js"></script>
<script src="https://unpkg.com/react-dom@18/umd/react-dom.development.js"></script>
<script src="https://unpkg.com/babel-standalone@6/babel.min.js"></script>
</head>
<body>
<h2>Practical 3: Applying Form Components in ReactJS</h2>
<div id="root"></div>
<script type="text/babel" src="formValidation.js"></script>
</body>
</html>
```

Create index.js

```
function SimpleForm() {
const [name, setName] = React.useState("");
const [comments, setComments] = React.useState("");
const [gender, setGender] = React.useState("male");
const handleSubmit = (event) => {
event.preventDefault();
alert(`Name: ${name} \nComments: ${comments} \nGender: ${gender}`);
};
return (
<div>
<h3>Fill the Form</h3>
<form onSubmit={handleSubmit}>
<label>
Name:
<input
type="text"
value={name}
onChange={(e) => setName(e.target.value)}
required
/>
</label>
<br /><br />
<label>
Comments:
<textarea
value={comments}
onChange={(e) => setComments(e.target.value)}
required
/>
</label>
<br /><br />
<label>
Gender:
<select value={gender} onChange={(e) =>
setGender(e.target.value)}>
<option value="male">Male</option>
<option value="female">Female</option>
<option value="other">Other</option>
</select>
</label>
<br /><br />
<button type="submit">Submit</button>
</form>
</div>
);
const root = ReactDOM.createRoot(document.getElementById("root"));
root.render(<SimpleForm />);
```

Output :-



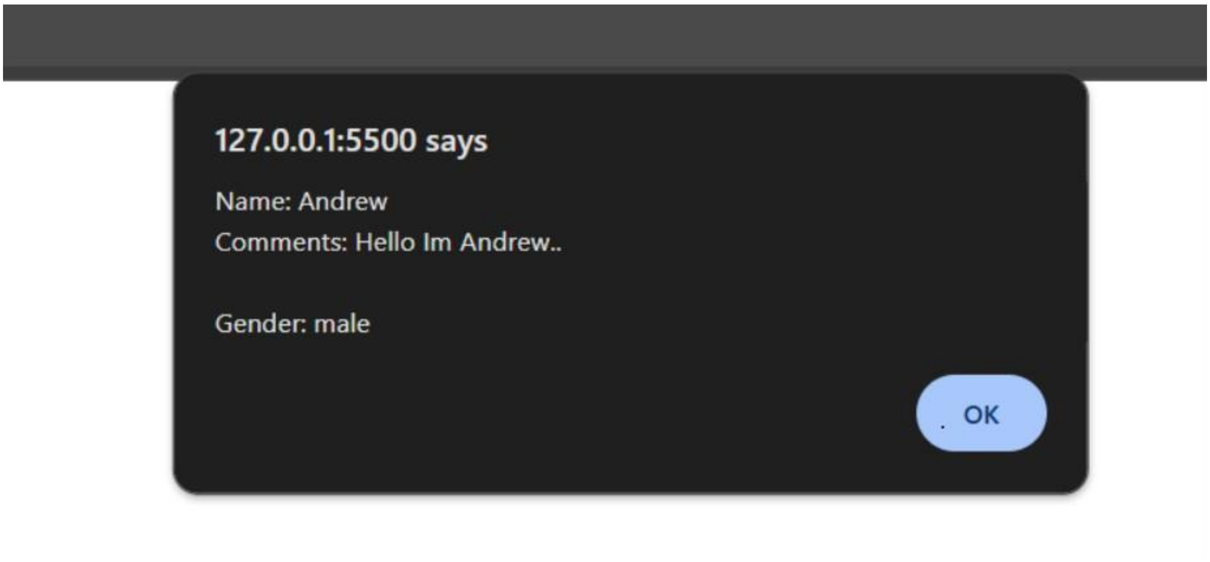
127.0.0.1:5500/practical-3/practical3.html

Fill the Form

Name: Andrew

Comments: Hello Im Andrew..

Gender: Male



127.0.0.1:5500 says

Name: Andrew
Comments: Hello Im Andrew..

Gender: male

4. Write ReactJs code to create student Registration Form.

Create index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Student Registration Form</title>
<script src="https://unpkg.com/react@18/umd/react.development.js"></script>
<script src="https://unpkg.com/react-dom@18/umd/reactdom.development.js"></script>
<script src="https://unpkg.com/babel-standalone@6/babel.min.js"></script>
</head>
<body>
<h2>Practical 4: Student Registration Form</h2>
<div id="root"></div>
<script type="text/babel" src="app.js"></script>
</body>
</html>
```

Create a JavaScript file (app.js)

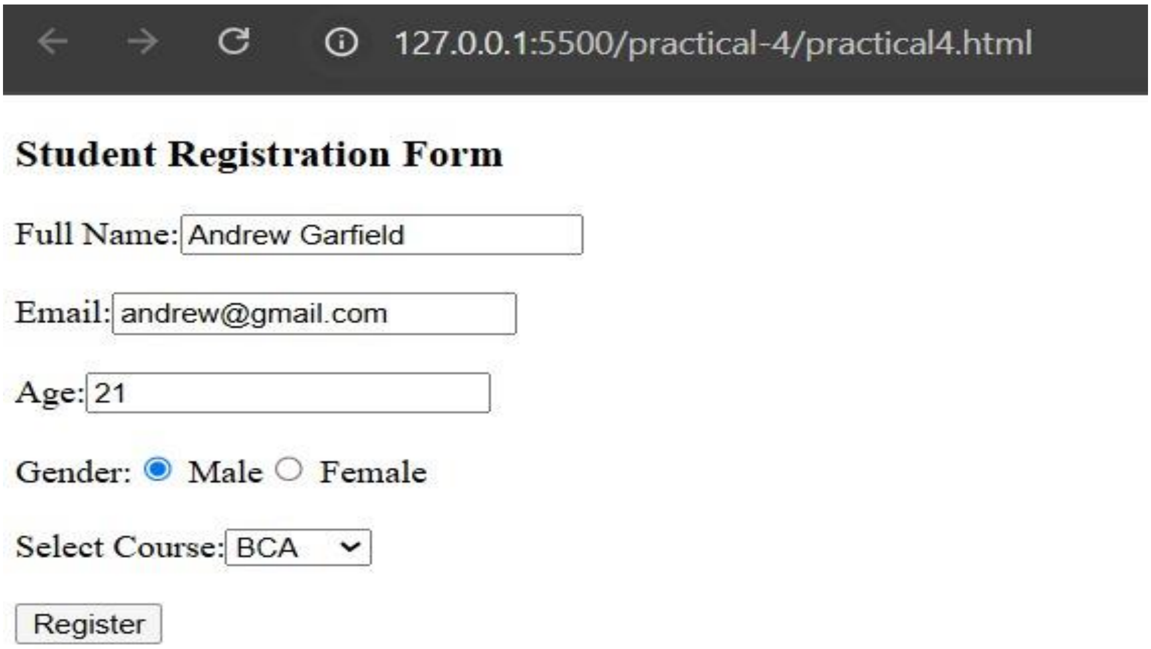
```
function StudentRegistrationForm() {
const [name, setName] = React.useState("");
const [email, setEmail] = React.useState("");
const [age, setAge] = React.useState("");
const [gender, setGender] = React.useState("");
const [course, setCourse] = React.useState("BCA");
const handleSubmit = (event) => {
event.preventDefault();
alert(`Student Registered!\n\nName: ${name}\nEmail:
${email}\nAge:
${age}\nGender: ${gender}\nCourse: ${course}`);
};
return (
<div>
<h3>Student Registration Form</h3>
<form onSubmit={handleSubmit}>
<label>
Full Name:
<input
type="text"
value={name}
```

```
onChange={(e) => setName(e.target.value)}
required
/>
</label>
<br /><br />
<label>
Email:
<input
type="email"
value={email}
onChange={(e) => setEmail(e.target.value)}
required
/>
</label>
<br /><br />
<label>
Age:
<input
type="number"
value={age}
onChange={(e) => setAge(e.target.value)}
required
```

```
/>
</label>
<br /><br />
<label>
Gender:
<input
type="radio"
name="gender"
value="Male"
onChange={(e) => setGender(e.target.value)}
required
/> Male
<input
type="radio"
name="gender"
value="Female"
onChange={(e) => setGender(e.target.value)}
required
/> Female
</label>
<br /><br />
<label>
```

```
Select Course:
<select value={course} onChange={(e) =>
setCourse(e.target.value)}>
<option value="BCA">BCA</option>
<option value="MCA">MCA</option>
<option value="BSc IT">BSc IT</option>
<option value="MSc IT">MSc IT</option>
</select>
</label>
<br /><br />
<button type="submit">Register</button>
</form>
</div>
);
}
const root =
ReactDOM.createRoot(document.getElementById("root"));
root.render(<StudentRegistrationForm />);
```

Output :-



127.0.0.1:5500/practical-4/practical4.html

Student Registration Form

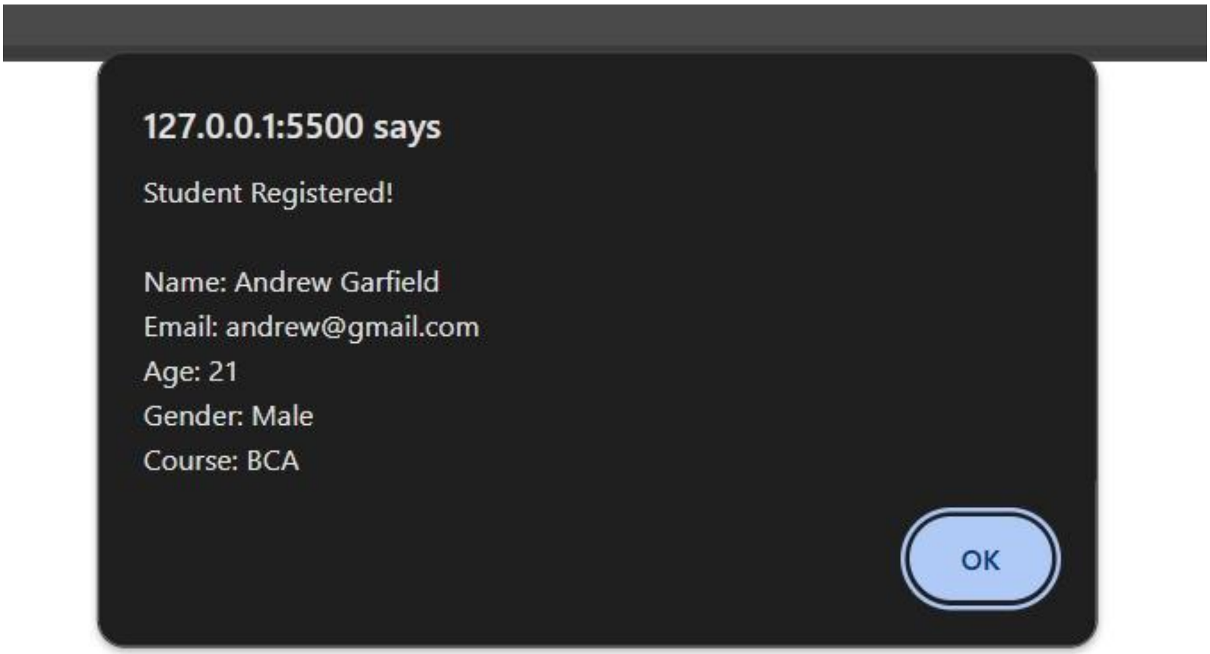
Full Name: Andrew Garfield

Email: andrew@gmail.com

Age: 21

Gender: ● Male ○ Female

Select Course: BCA



127.0.0.1:5500 says

```
Student Registered!

Name: Andrew Garfield
Email: andrew@gmail.com
Age: 21
Gender: Male
Course: BCA
```

5. Write ReactJs code to create Simple Login Form.

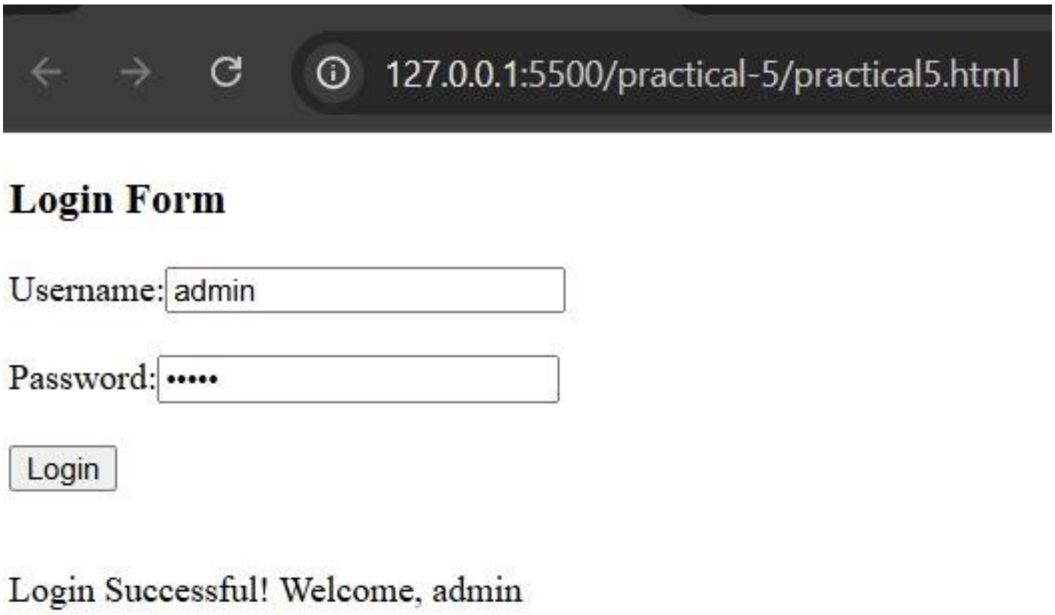
Create index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Practical 5: Simple Login Form</title>
<script
src="https://unpkg.com/react@18/umd/react.development.js"></script>
<script src="https://unpkg.com/react-
dom@18/umd/reactdom.development.js"></script>
<script src="https://unpkg.com/babel-standalone@6/babel.min.js"></script>
</head>
<body>
<h2>Practical 5: Simple Login Form</h2>
<div id="root"></div>
<script type="text/babel" src="demo.js"></script>
</body>
</html>
```

Create a JavaScript file (app.js)

```
function LoginForm() {
const [username, setUsername] = React.useState("");
const [password, setPassword] = React.useState("");
const [message, setMessage] = React.useState("");
const handleSubmit = (event) => {
event.preventDefault();
// Simple validation check
if (username === "admin" && password === "12345") {
setMessage("Login Successful! Welcome, " + username);
} else {
setMessage("Invalid Credentials. Try again.");
}
};
return (
<div>
<h3>Login Form</h3>
<form onSubmit={handleSubmit}>
<label>
Username:
<input
type="text"
value={username}
onChange={(e) => setUsername(e.target.value)}
required
/>
</label>
<br /><br />
<label>
Password:
<input
type="password"
value={password}
onChange={(e) => setPassword(e.target.value)}
required
/>
</label>
<br /><br />
<button type="submit">Login</button>
</form>
<br />
<p>{message}</p>
</div>
);
}
const root = ReactDOM.createRoot(document.getElementById("root"));
root.render(<LoginForm />);
```

Output :-



127.0.0.1:5500/practical-5/practical5.html

Login Form

Username: admin

Password: •••••

Login Successful! Welcome, admin

6. Write ReactJs Create a Single Page Application.

Create index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Practical 6: Single Page Application</title>
<script src="https://unpkg.com/react@18/umd/react.development.js"></script>
<script src="https://unpkg.com/react-dom@18/umd/reactdom.development.js"></script>
<script src="https://unpkg.com/babel-standalone@6/babel.min.js"></script>
</head>
<body>
<h2>Practical 6: Single Page Application</h2>
<div id="root"></div>
<script type="text/babel" src="demo.js"></script>
</body>
</html>
```

Create a JavaScript file (app.js)

```
const { BrowserRouter, Routes, Route, Link } = ReactRouterDOM;
// Home Component
function Home() {
return <h2>Welcome to the Home Page</h2>;
}
// About Component
function About() {
return <h2>This is the About Page</h2>;
}
// Contact Component
function Contact() {
return <h2>Contact Us at: example@example.com</h2>;
}
// App Component with Navigation
function App() {
return (
<BrowserRouter>
<div>
<nav>
<ul>
<li><Link to="/">Home</Link></li>
<li><Link to="/about">About</Link></li>
<li><Link to="/contact">Contact</Link></li>
</ul>
</nav>
<Routes>
<Route path="/" element={<Home />} />
<Route path="/about" element={<About />} />
<Route path="/contact" element={<Contact />} />
</Routes>
</div>
</BrowserRouter>
);
}
const root = ReactDOM.createRoot(document.getElementById("root"));
root.render(<App />);
```

7. Write ReactJs / NodeJs code to Applying Routing.

Create index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>React Routing</title>
<script
src="https://unpkg.com/react@18/umd/react.development.js"></script>
<script src="https://unpkg.com/react-dom@18/umd/react-
dom.development.js"></script>
<script src="https://unpkg.com/babel-standalone@6/babel.min.js"></script>
<script src="https://unpkg.com/react-router-dom/umd/react-router-
dom.min.js"></script>
</head>
<body>
<h2>Practical 7: Applying Routing</h2>
<div id="root"></div>
<script type="text/babel" src="routing.js"></script>
</body>
</html>
```

Create a JavaScript file (app.js)

```
const { BrowserRouter, Routes, Route, Link } = ReactRouterDOM;
function Home() {
return <h3>Welcome to Home Page</h3>;
}
function About() {
return <h3>This is the About Page</h3>;
}
function App() {
return (
<BrowserRouter>
<nav>
<Link to="/">Home</Link> |
<Link to="/about">About</Link>
</nav>
<Routes>
<Route path="/" element={<Home />} />
<Route path="/about" element={<About />} />
</Routes>
</BrowserRouter>
);
}
const root = ReactDOM.createRoot(document.getElementById("root"));
root.render(<App />);
```

8. Write ReactJs / NodeJs code to demonstrate the use of POST Method.

Create index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>React POST Method</title>
<script src="https://unpkg.com/react@18/umd/react.development.js"></script>
<script src="https://unpkg.com/react-dom@18/umd/reactdom.development.js"></script>
<script src="https://unpkg.com/babel-standalone@6/babel.min.js"></script>
</head>
<body>
<h2>Practical 8: POST Method in React</h2>
<div id="root"></div>
<script type="text/babel" src="post_request.js"></script>
<body>
<html>
```

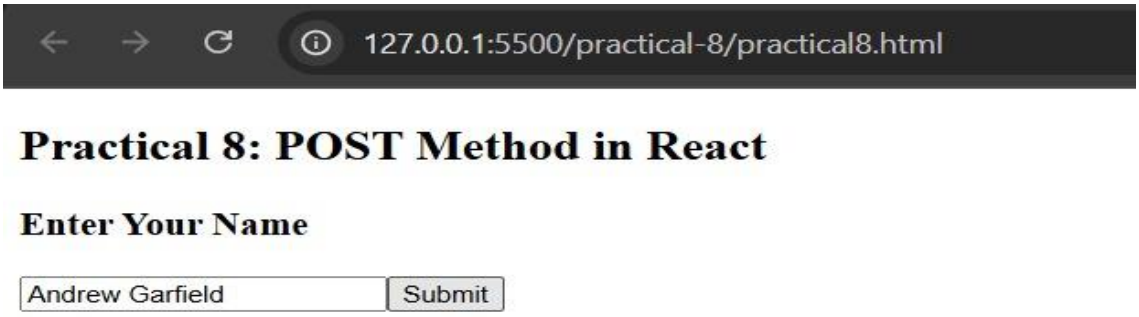
ReactJS File (post_request.js)

```
function App() {
const [name, setName] = React.useState("");
const handleSubmit = (e) => {
e.preventDefault();
fetch("http://localhost:5000/submit", {
method: "POST",
headers: { "Content-Type": "application/json" },
body: JSON.stringify({ name }),
})
.then(response => response.json())
.then(data => alert("Response from Server: " + data.message));
};
return (
<div>
<h3>Enter Your Name</h3>
<form onSubmit={handleSubmit}>
<input
type="text"
value={name}
onChange={(e) => setName(e.target.value)}
placeholder="Enter name"
required
/>
<button type="submit">Submit</button>
</form>
</div>
);
}
const root =
ReactDOM.createRoot(document.getElementById("root"));
root.render(<App />);
```

Create a Node.js Backend (server.js)

```
const express = require("express");
const cors = require("cors");
const app = express();
app.use(express.json()); // Parse JSON requests
app.use(cors()); // Enable CORS for frontend-backend communication
app.post("/submit", (req, res) => {
const { name } = req.body;
res.json({ message: `Hello, ${name}! Your data has been received.` });
});
app.listen(5000, () => console.log("Server running on port 5000"));
```

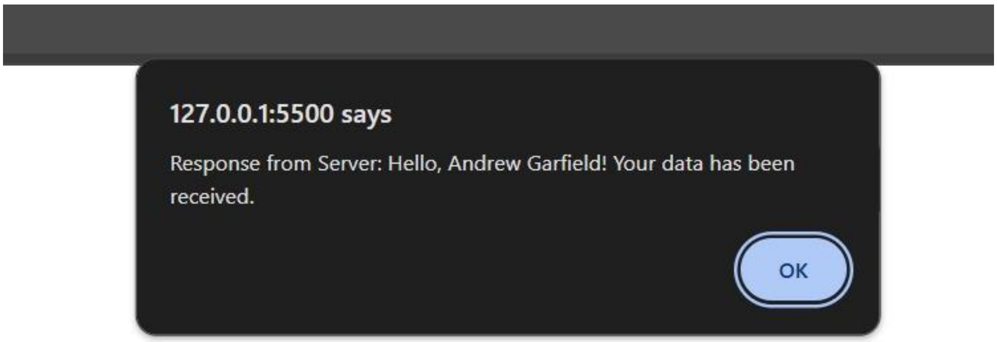
Output :-



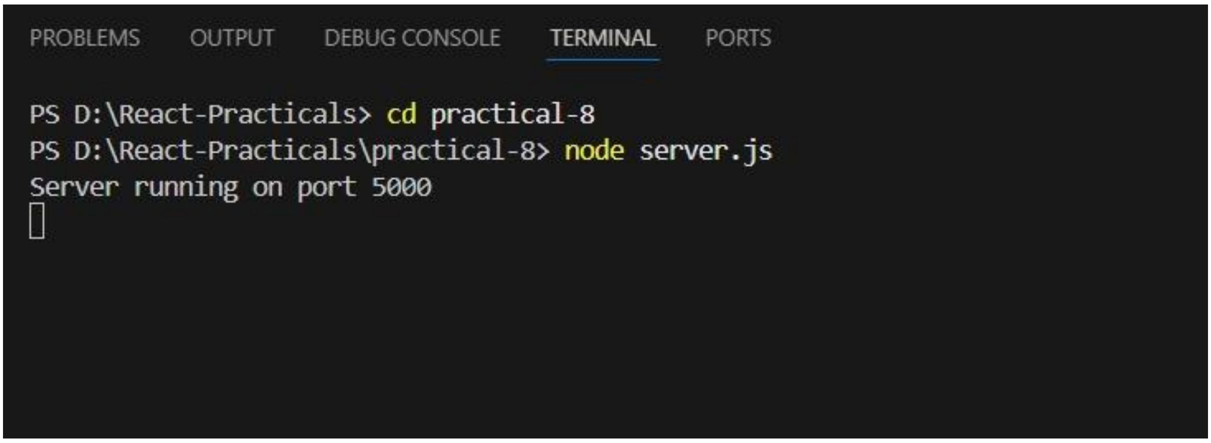
Practical 8: POST Method in React

Enter Your Name

Andrew Garfield


127.0.0.1:5500 says

Response from Server: Hello, Andrew Garfield! Your data has been received.



```
PS D:\React-Practicals> cd practical-8
PS D:\React-Practicals\practical-8> node server.js
Server running on port 5000
```

9. Write ReactJs / NodeJs code to demonstrate the use of GET Method

Create index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>React GET Method</title>
<script src="https://unpkg.com/react@18/umd/react.development.js"></script>
<script src="https://unpkg.com/react-dom@18/umd/reactdom.development.js"></script>
<script src="https://unpkg.com/babel-standalone@6/babel.min.js"></script>
</head>
<body>
<h2>Practical 9: GET Method in React</h2>
<div id="root"></div>
<script type="text/babel" src="get_request.js"></script>
</body>
</html>
```

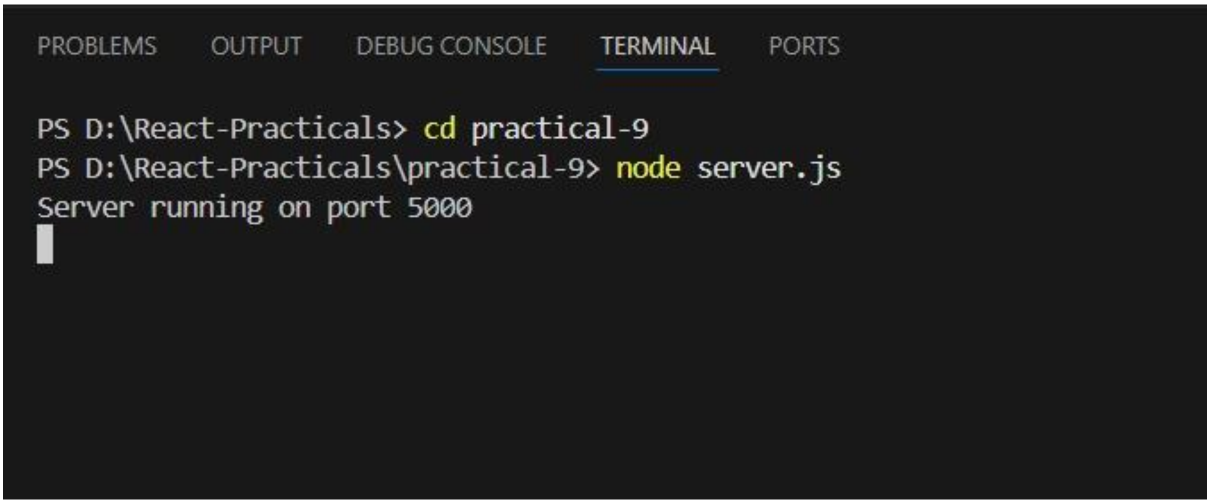
ReactJS File (get_request.js)

```
function App() {
const [data, setData] = React.useState([]);
React.useEffect(() => {
fetch("http://localhost:5000/data")
.then(response => response.json())
.then(data => setData(data));
}, []);
return (
<div>
<h3>Fetched Data from Server</h3>
<ul>
{data.map((item, index) => (
<li key={index}>{item}</li>
))}
</ul>
</div>
);
}
const root =
ReactDOM.createRoot(document.getElementById("root"));
root.render(<App />);
```

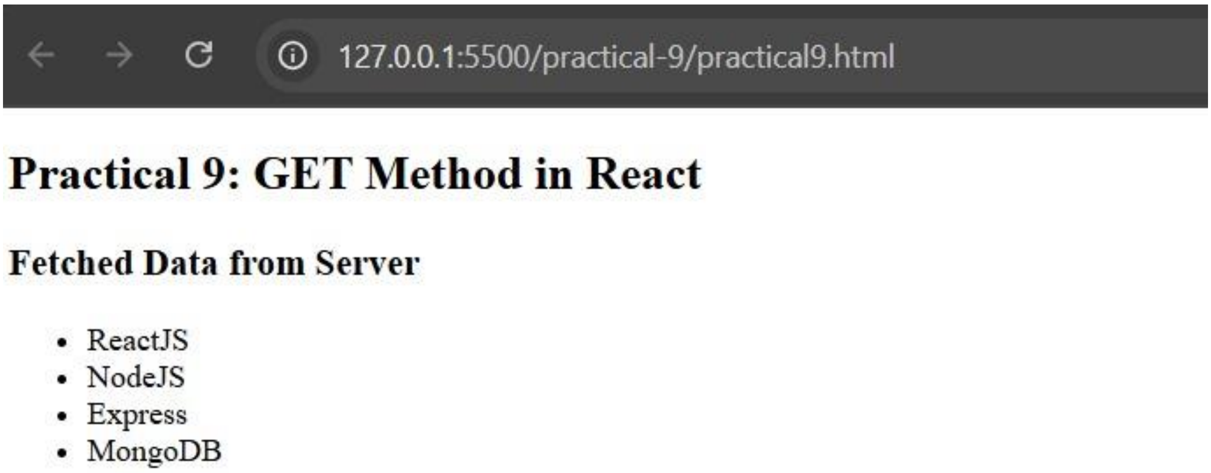
Create a Node.js Backend (server.js)

```
const express = require("express");
const cors = require("cors");
const app = express();
app.use(cors()); // Enable CORS for frontend-backend communication
app.get("/data", (req, res) => {
const sampleData = ["ReactJS", "NodeJS", "Express", "MongoDB"];
res.json(sampleData);
});
app.listen(5000, () => console.log("Server running on port 5000"));
```

Output :-



```
PS D:\React-Practicals> cd practical-9
PS D:\React-Practicals\practical-9> node server.js
Server running on port 5000
```



127.0.0.1:5500/practical-9/practical9.html

Practical 9: GET Method in React

Fetched Data from Server

- ReactJS
- NodeJS
- Express
- MongoDB

10. To demonstrate REST API in Node.js

Create a Node.js Project and Install Dependencies

Open your terminal and run:

```
mkdir node-rest-api
cd node-rest-api
npm init -y
npm install express cors body-parser
```

Create the REST API Server (server.js)

```
const express = require("express");
const cors = require("cors");
const app = express();
app.use(cors());
app.use(express.json());
let students = [
{ id: 1, name: "Alice", course: "BCA" },
{ id: 2, name: "Bob", course: "MCA" }
];
// GET: Fetch all students
app.get("/students", (req, res) => {
res.json(students);
});
// POST: Add a new student
app.post("/students", (req, res) => {
```

```
const newStudent = { id: students.length + 1, ...req.body };
students.push(newStudent);
res.status(201).json(newStudent);
});
// PUT: Update student details
app.put("/students/:id", (req, res) => {
const { id } = req.params;
const index = students.findIndex(s => s.id == id);
if (index !== -1) {
students[index] = { ...students[index], ...req.body };
res.json(students[index]);
} else {
res.status(404).send("Student not found");
}
});
// DELETE: Remove a student
app.delete("/students/:id", (req, res) => {
const { id } = req.params;
students = students.filter(s => s.id != id);
res.send("Student deleted");
});
app.listen(5000, () => console.log("Server running on port 5000"));
```

11. Create NodeJs Application for to stored student information in database.

13.Create NodeJs Application to display student information.

14. Create NodeJs Application to upadate, display, and delete student information.

Create a New Node.js Project and Install Dependencies

Run the following commands in the terminal:

```
mkdir student-management

cd student-management

npm init -y

npm install express mongoose cors body-parser
```

Create a MongoDB Database

- If MongoDB is installed locally, start the database using

```
mongod --dbpath /path/to/database
```

Create server.js File

```
const express = require("express");

const mongoose = require("mongoose");

const cors = require("cors");

const app = express();

app.use(cors());

app.use(express.json());

// Connect to MongoDB (Change the connection string if using MongoDB Atlas)
```

```
mongoose.connect("mongodb://127.0.0.1:27017/studentsDB", {
useNewUrlParser: true,
useUnifiedTopology: true
}).then(() => console.log("Connected to MongoDB"))
.catch(err => console.error("MongoDB Connection Error:", err));
// Define Student Schema
const studentSchema = new mongoose.Schema({
name: String,
course: String,
age: Number
});
const Student = mongoose.model("Student", studentSchema);
// API Routes
// GET: Fetch all students
app.get("/students", async (req, res) => {
const students = await Student.find();
res.json(students);
});
// POST: Add a new student
app.post("/students", async (req, res) => {
const newStudent = new Student(req.body);
await newStudent.save();
res.status(201).json(newStudent);
});
// PUT: Update student details
app.put("/students/:id", async (req, res) => {
```

```
const updatedStudent = await Student.findByIdAndUpdate(req.params.id, req.body, {
new: true });
res.json(updatedStudent);
});
// DELETE: Remove a student
app.delete("/students/:id", async (req, res) => {
await Student.findByIdAndDelete(req.params.id);
res.send("Student deleted");
});
app.listen(5000, () => console.log("Server running on port 5000"));
```