1.Write ReactJs code to use the states in the created Application.

App.js

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

Header.jsx

```
import React, { useState } from "react";
function Header() {
  const [count, setCount] = useState(0);
  return (
    <div>
      <h1>Counter: {count}</h1>
      <button onClick={() => setCount(count + 1)}>Increment</button>
      <button onClick={() => setCount(count - 1)}>Decrement</button>
      <button onClick={() => setCount(0)}>Reset</button>
    </div>
  );
}

export default Header;
```

2.Create Node JS Application for to stored student information in database

```
student-app/
│── server.js
│── models/
│   ├── studentModel.js
│── routes/
│   ├── studentRoutes.js
│── config/
│   ├── dbConfig.js
```

dbConfig.js

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

const connectDB = async () => {
    try {
        await mongoose.connect("mongodb://localhost:27017/studentsDB", {
            useNewUrlParser: true,
            useUnifiedTopology: true
        });
        console.log("MongoDB connected successfully!");
    } catch (error) {
        console.error("Database connection failed:", error);
    }
};

module.exports = connectDB;
```

studentModel.js

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

const studentSchema = new mongoose.Schema({
    name: String,
    age: Number,
    grade: String
});

const Student = mongoose.model("Student", studentSchema);

module.exports = Student;
```

studentRoutes.js

```
const express = require("express");
const Student = require("../models/studentModel");

const router = express.Router();
```

```
router.post("/add", async (req, res) => {
    try {
        const { name, age, grade } = req.body;
        const newStudent = new Student({ name, age, grade });
        await newStudent.save();
        res.status(201).send("Student added successfully!");
    } catch (error) {
        res.status(500).send(error.message);
    }
});

router.get("/all", async (req, res) => {
    try {
        const students = await Student.find();
        res.json(students);
    } catch (error) {
        res.status(500).send(error.message);
    }
});

module.exports = router;
```

server.js

```
const express = require("express");
const bodyParser = require("body-parser");
const cors = require("cors");
const connectDB = require("./config/dbConfig");
const studentRoutes = require("./routes/studentRoutes");

const app = express();
const PORT = 5000;

app.use(cors());
app.use(bodyParser.json());
app.use("/students", studentRoutes);

connectDB();

app.listen(PORT, () => {
    console.log(`Server running on http://localhost:${PORT}`);
});
```

3.Write ReactJs code for Client-side form validation.

App.js

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

Header.jsx

```
import React, { useState } from "react";
const FormValidation = () => {
  const [formData, setFormData] = useState({
    name: "",
    email: "",
    password: "",
  });

  const [errors, setErrors] = useState({});
  const [isSubmitted, setIsSubmitted] = useState(false);

  const validate = () => {
    let newErrors = {};

    if (!formData.name.trim()) {
      newErrors.name = "Name is required!";
    } else if (formData.name.length < 3) {
      newErrors.name = "Name must be at least 3 characters!";
    }

    const emailRegex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;
    if (!formData.email.trim()) {
      newErrors.email = "Email is required!";
    } else if (!emailRegex.test(formData.email)) {
      newErrors.email = "Invalid email format!";
    }

    if (!formData.password.trim()) {
      newErrors.password = "Password is required!";
    } else if (formData.password.length < 6) {
      newErrors.password = "Password must be at least 6 characters!";
    }

    setErrors(newErrors);
    return Object.keys(newErrors).length === 0
  };

  const handleChange = (e) => {
    setFormData({ ...formData, [e.target.name]: e.target.value });
  };

  const handleSubmit = (e) => {
    e.preventDefault();
    if (validate()) {
      setIsSubmitted(true);
    }
  };

  return (
    <div>
      <h2>Client-side Form Validation</h2>
      {isSubmitted && <p style={{ color: "green" }}>Form submitted successfully!</p>}
      <form onSubmit={handleSubmit}>
        <div>
          <label>Name:</label>
          <input type="text" name="name" value={formData.name} onChange={handleChange} />
          {errors.name && <p style={{ color: "red" }}>{errors.name}</p>}
        </div>

        <div>
          <label>Email:</label>
          <input type="email" name="email" value={formData.email} onChange={handleChange} />
          {errors.email && <p style={{ color: "red" }}>{errors.email}</p>}
        </div>

        <div>
          <label>Password:</label>
          <input type="password" name="password" value={formData.password} onChange={handleChange}
/>
          {errors.password && <p style={{ color: "red" }}>{errors.password}</p>}
        </div>

        <button type="submit" disabled={Object.keys(errors).length > 0}>Submit</button>
      </form>
    </div>
  );
};

export default FormValidation;
```

4.Create Node JS Application for login credentials.

```
login-app/
│── server.js
│── models/
│   ├── userModel.js
│── routes/
│   ├── authRoutes.js
│── config/
│   ├── dbConfig.js
│── .env
```

userModel.js

```
const mongoose = require("mongoose");
const bcrypt = require("bcryptjs");

const userSchema = new mongoose.Schema({
    username: String,
    email: { type: String, unique: true },
    password: String,
});

userSchema.pre("save", async function (next) {
    if (!this.isModified("password")) return next();
    this.password = await bcrypt.hash(this.password, 10);
    next();
});

const User = mongoose.model("User", userSchema);
module.exports = User;
```

authRoutes.js

```
const express = require("express");
const User = require("../models/userModel");
const bcrypt = require("bcryptjs");
const jwt = require("jsonwebtoken");
require("dotenv").config();

const router = express.Router();
const SECRET_KEY = process.env.JWT_SECRET;

router.post("/register", async (req, res) => {
    try {
        const { username, email, password } = req.body;

        if (!username || !email || !password) {
            return res.status(400).json({ error: "All fields are required!" });
        }
        const existingUser = await User.findOne({ email });
        if (existingUser) {
            return res.status(400).json({ error: "Email already exists!" });
        }

        const hashedPassword = await bcrypt.hash(password, 10);
        const newUser = new User({ username, email, password: hashedPassword });

        await newUser.save();
        res.status(201).json({ message: "User registered successfully!" });
    } catch (error) {
        res.status(500).json({ error: error.message });
    }
});

router.post("/login", async (req, res) => {
    try {
        const { email, password } = req.body;

        if (!email || !password) {
            return res.status(400).json({ error: "Both email and password are required!" });
        }

        const user = await User.findOne({ email });
        if (!user) {
            return res.status(404).json({ error: "User not found!" });
        }

        const isMatch = await bcrypt.compare(password, user.password);
        if (!isMatch) {
            return res.status(400).json({ error: "Invalid credentials!" });
        }

        const token = jwt.sign({ id: user._id }, SECRET_KEY, { expiresIn: "1h" });

        res.json({
            message: "Login successful!",
            token: `Bearer ${token}`
        });
    } catch (error) {
        res.status(500).json({ error: error.message });
    }
});

router.get("/user", async (req, res) => {
    try {
        const token = req.header("Authorization");
        if (!token) {
            return res.status(401).json({ error: "Access denied!" });
        }

        const decoded = jwt.verify(token.replace("Bearer ", ""), SECRET_KEY);
        const user = await User.findById(decoded.id).select("-password");

        res.json(user);
    } catch (error) {
        res.status(400).json({ error: "Invalid token!" });
    }
});

module.exports = router;
```

dbConfig.js

```
const mongoose = require("mongoose");
require("dotenv").config();

const connectDB = async () => {
    try {
        await mongoose.connect(process.env.MONGO_URI);
        console.log("MongoDB connected!");
    } catch (error) {
        console.error("DB connection failed:", error);
    }
};

module.exports = connectDB;
```

.env

```
MONGO_URI=mongodb://localhost:27017/loginDB
JWT_SECRET=my_secret_key
PORT=5000
```

Server.js

```
const express = require("express");
const connectDB = require("./config/dbConfig");
const authRoutes = require("./routes/authRoutes");
require("dotenv").config();

const app = express();
const PORT = process.env.PORT || 5000;

app.use(express.json());
app.use("/auth", authRoutes);

connectDB();

app.listen(PORT, () => console.log(`Server running on http://localhost:${PORT}`));
```

5.Write ReactJs code for Applying form components.

App.js

```
 import './App.css';
import Header   from './components/Header';
function App() {
  return (
    <div className="App">
      < FormComponent/>
    </div>
  );
}
export default App;
```

FormComponent.jsx

```
import React, { useState } from "react";

const FormComponent = () => {
  const [formData, setFormData] = useState({ name: "", email: "" });

  const handleChange = (e) => {
    setFormData({ ...formData, [e.target.name]: e.target.value });
  };

  const handleSubmit = (e) => {
    e.preventDefault();
    alert(`Name: ${formData.name}, Email: ${formData.email}`);
  };

  return (
    <div>
      <h2>Simple React Form</h2>
      <form onSubmit={handleSubmit}>
        <label>Name:</label>
        <input type="text" name="name" value={formData.name} onChange={handleChange} />
        <br />
        <label>Email:</label>
        <input type="email" name="email" value={formData.email} onChange={handleChange} />
        <br />
        <button type="submit">Submit</button>
      </form>
    </div>
  );
};

export default FormComponent;
```

6.Create Node JS Application to display student information.

```
student-app/
│── server.js
│── models/
│   ├── studentModel.js
│── routes/
│   ├── studentRoutes.js
│── config/
│   ├── dbConfig.js
```

dbConfig.js

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

const connectDB = async () => {
   try {
      await mongoose.connect("mongodb://localhost:27017/studentsDB");
      console.log("MongoDB connected successfully!");
   } catch (error) {
      console.error("Database connection failed:", error);
      process.exit(1);
   }
};

module.exports = connectDB;
```

studentModel.js

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

const studentSchema = new mongoose.Schema({
   name: { type: String, required: true },
   age: { type: Number, required: true },
   grade: { type: String, required: true }
}, { timestamps: true });

const Student = mongoose.model("Student", studentSchema);

module.exports = Student;
```

studentRoutes.js

```
const express = require("express");
const Student = require("../models/studentModel");

const router = express.Router();

router.get("/all", async (req, res) => {
   try {
      const students = await Student.find();
```

```
        res.json(students);
    } catch (error) {
        res.status(500).json({ error: error.message });
    }
});

module.exports = router;
```

server.js

```
const express = require("express");
const cors = require("cors");
const connectDB = require("./config/dbConfig");
const studentRoutes = require("./routes/studentRoutes");

const app = express();
const PORT = 5000;

app.use(cors());
app.use(express.json());
app.use("/students", studentRoutes);

connectDB();

app.get("/", (req, res) => {
    res.send("Welcome to the Student Display API!");
});

app.listen(PORT, () => {
    console.log(`Server running at http://localhost:${PORT}`);
});
```

7.Create Node JS Application to update, display and delete student information.

```
student-app/
│── server.js
│── models/
│   ├── studentModel.js
│── routes/
│   ├── studentRoutes.js
│── config/
│   ├── dbConfig.js
```

dbConfig.js

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

const connectDB = async () => {
    try {
        await mongoose.connect("mongodb://localhost:27017/studentsDB", {
            useNewUrlParser: true,
            useUnifiedTopology: true
        });
        console.log("MongoDB connected successfully!");
    } catch (error) {
        console.error("Database connection failed:", error);
    }
};

module.exports = connectDB;
```

studentModel.js

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

const studentSchema = new mongoose.Schema({
    name: String,
    age: Number,
    grade: String
});

const Student = mongoose.model("Student", studentSchema);

module.exports = Student;
```

studentRoutes.js

```
const express = require("express");
const Student = require("../models/studentModel");

const router = express.Router();
```

```
router.post("/add", async (req, res) => {
    try {
        const { name, age, grade } = req.body;
        const newStudent = new Student({ name, age, grade });
        await newStudent.save();
        res.status(201).send("Student added successfully!");
    } catch (error) {
        res.status(500).send(error.message);
    }
});

router.get("/all", async (req, res) => {
    try {
        const students = await Student.find();
        res.json(students);
    } catch (error) {
        res.status(500).send(error.message);
    }
});

module.exports = router;
```

server.js

```
const express = require("express");
const bodyParser = require("body-parser");
const cors = require("cors");
const connectDB = require("./config/dbConfig");
const studentRoutes = require("./routes/studentRoutes");

const app = express();
const PORT = 5000;

app.use(cors());
app.use(bodyParser.json());
app.use("/students", studentRoutes);

connectDB();

app.listen(PORT, () => {
    console.log(`Server running on http://localhost:${PORT}`);
});
```

8.Write ReactJs code to create Simple Login Form.

App.js

```
import './App.css';
import Header   from './components/Header';
function App() {
  return (
    <div className="App">
          < LoginForm />
    </div>
  );
}
export default App;
```

LoginForm.jsx

```
import React, { useState } from "react";

const LoginForm = () => {
  const [formData, setFormData] = useState({ email: "", password: "" });
  const handleChange = (e) => {
    setFormData({ ...formData, [e.target.name]: e.target.value });
  };

  const handleSubmit = (e) => {
    e.preventDefault();
    alert(`Email: ${formData.email}\nPassword: ${formData.password}`);
  };

  return (
    <div style={{ textAlign: "center", padding: "20px" }}>
      <h2>Login Form</h2>
      <form onSubmit={handleSubmit}>
        <div>
          <label>Email:</label>
          <input type="email" name="email" value={formData.email} onChange={handleChange} />
        </div>
        <br />
        <div>
          <label>Password:</label>
          <input type="password" name="password" value={formData.password} onChange={handleChange}
/>
        </div>
        <br />
        <button type="submit">Login</button>
      </form>
    </div>
  );
};

export default LoginForm;
```

9.Write ReactJsCreate a Single Page Application.

```
react-spa/
│── src/
│   ├── App.js
│   ├── index.js
│   ├── components/
│   │   ├── Navbar.js
│   │   ├── Home.js
│   │   ├── About.js
│   │   ├── Contact.js
```

App.js

```
import React from "react";
import { BrowserRouter as Router, Routes, Route } from "react-router-dom";
import Navbar from "./components/Navbar";
import Home from "./components/Home";
import About from "./components/About";
import Contact from "./components/Contact";

const App = () => {
  return (
    <Router>
      <Navbar />
      <Routes>
        <Route path="/" element={<Home />} />
        <Route path="/about" element={<About />} />
        <Route path="/contact" element={<Contact />} />
      </Routes>
    </Router>
  );
};
export default App;
```

Navbar.js

```
import React from "react";
import { Link } from "react-router-dom";

const Navbar = () => {
  return (
    <nav>
      <Link to="/">Home</Link> | <Link to="/about">About</Link> | <Link to="/contact">Contact</Link>
    </nav>
  );
};
export default Navbar;
```

Home.js

```
import React from "react";

const Home = () => {
  return <h1>Welcome to the Home Page</h1>;
};

export default Home;
```

About.js

```
import React from "react";

const About = () => {
  return <h1>About Us</h1>;
};

export default About;
```

Contact.js

```
import React from "react";

const Contact = () => {
  return <h1>Contact Us</h1>;
};

export default Contact;
```

10. Write ReactJs / NodeJs code to Applying Routing.

```
react-spa/
│── src/
│   ├── App.js
│   ├── index.js
│   ├── components/
│   │   ├── Navbar.js
│   │   ├── Home.js
│   │   ├── About.js
│   │   ├── Contact.js
```

App.js

```
import React from "react";
import { BrowserRouter as Router, Routes, Route } from "react-router-dom";
import Navbar from "./components/Navbar";
import Home from "./components/Home";
import About from "./components/About";
import Contact from "./components/Contact";

const App = () => {
  return (
    <Router>
      <Navbar />
      <Routes>
        <Route path="/" element={<Home />} />
        <Route path="/about" element={<About />} />
        <Route path="/contact" element={<Contact />} />
      </Routes>
    </Router>
  );
};
export default App;
```

Navbar.js

```
import React from "react";
import { Link } from "react-router-dom";

const Navbar = () => {
  return (
    <nav>
      <Link to="/">Home</Link> | <Link to="/about">About</Link> | <Link to="/contact">Contact</Link>
    </nav>
  );
};
export default Navbar;
```

Home.js
```
import React from "react";

const Home = () => {
  return <h1>Welcome to the Home Page</h1>;
};

export default Home;
```

About.js
```
import React from "react";

const About = () => {
  return <h1>About Us</h1>;
};

export default About;
```

Contact.js
```
import React from "react";

const Contact = () => {
  return <h1>Contact Us</h1>;
};

export default Contact;
```

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

```
post-demo/
│── server.js
│── client/
│   ├── App.js
```

Server.js

```
const express = require("express");
const cors = require("cors");
const bodyParser = require("body-parser");

const app = express();
app.use(cors());
app.use(bodyParser.json());

app.post("/submit", (req, res) => {
    const { name } = req.body;
    res.json({ message: `Hello, ${name}!` });
});

app.listen(5000, () => console.log("Server running on http://localhost:5000"));
```

App.js

```
import React, { useState } from "react";
import axios from "axios";

const App = () => {
  const [name, setName] = useState("");
  const [response, setResponse] = useState("");

  const handleSubmit = async () => {
    const res = await axios.post("http://localhost:5000/submit", { name });
    setResponse(res.data.message);
  };

  return (
    <div>
      <h2>POST Request Example</h2>
      <input type="text" placeholder="Enter name" value={name} onChange={(e) =>
setName(e.target.value)} />
      <button onClick={handleSubmit}>Submit</button>
      <p>{response}</p>
    </div>
  );
};

export default App;
```

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

```
post-demo/
│── server.js
│── client/
│   ├── App.js
```

Server.js

```
const express = require("express");
const cors = require("cors");
const app = express();
app.use(cors());

app.get("/students", (req, res) => {
    const students = [
        { id: 1, name: "John Doe", age: 22 },
        { id: 2, name: "Jane Smith", age: 20 },
    ];
    res.json(students);
});
app.listen(5000, () => console.log("Server running on http://localhost:5000"));
```

App.js

```
import React, { useState, useEffect } from "react";
import axios from "axios";

const App = () => {
  const [students, setStudents] = useState([]);
  useEffect(() => {
    axios.get("http://localhost:5000/students").then((response) => {
      setStudents(response.data);
    });
  }, []);

  return (
    <div>
      <h2>Student List</h2>
      <ul>
        {students.map((student) => (
          <li key={student.id}>
            {student.name} - {student.age} years old
          </li>
        ))}
      </ul>
    </div>
  );
};

export default App;
```

13. Create ReactJS Application for student registration form.

```
student-registration/
│── src/
│   ├── App.js
│   ├── components/
│   │   ├── RegistrationForm.js
```

RegistrationForm.js

```
import React, { useState } from "react";

const RegistrationForm = () => {
  const [formData, setFormData] = useState({
    name: "",
    age: "",
    email: "",
    course: ""
  });
  const handleChange = (e) => {
    setFormData({ ...formData, [e.target.name]: e.target.value });
  };

  const handleSubmit = (e) => {
    e.preventDefault();
    alert(`Student Registered!\nName: ${formData.name}\nAge: ${formData.age}\nEmail:
${formData.email}\nCourse: ${formData.course}`);
  };

  return (
    <div style={{ textAlign: "center", padding: "20px" }}>
      <h2>Student Registration Form</h2>
      <form onSubmit={handleSubmit}>
        <div>
          <label>Name:</label>
          <input type="text" name="name" value={formData.name} onChange={handleChange} />
        </div>

        <br />

        <div>
          <label>Age:</label>
          <input type="number" name="age" value={formData.age} onChange={handleChange} />
        </div>

        <br />

        <div>
          <label>Email:</label>
          <input type="email" name="email" value={formData.email} onChange={handleChange} />
        </div>

        <br />

        <div>
          <label>Course:</label>
          <input type="text" name="course" value={formData.course} onChange={handleChange} />
        </div>

        <br />

        <button type="submit">Register</button>
      </form>
    </div>
  );
};

export default RegistrationForm;
```

App.js

```
import './App.css';
import RegistrationForm from './components/RegistrationForm';

function App() {
  return (
    <div className="App">
      <RegistrationForm />
    </div>
  );
}

export default App;
```