Part B

1.Develop a React application that dynamically displays the capital city of a selected country. Implement the country selection dropdown as a separate, reusable component. The main application component should integrate this dropdown component and, upon selection of a country, display the corresponding capital city.

CountryDropdown.js

```
function CountryDropdown(props) {
   function handleChange(event) {
     props.onSelect(event.target.value);
   }

   return (
     <select onChange={handleChange}>
       <option value="">Select a Country</option>
       {props.countries.map(function (country) {
         return (
           <option key={country.name} value={country.name}>
             {country.name}
           </option>
         );
       })}
     </select>
   );
 }
 export default CountryDropdown;
```

App.js

```
import React, { useState } from "react";
import CountryDropdown from "./CountryDropdown";

function App() {
  const [capital, setCapital] = useState("");

  const countries = [
    { name: "United States", capital: "Washington, D.C." },
    { name: "India", capital: "New Delhi" },
    { name: "France", capital: "Paris" },
    { name: "Germany", capital: "Berlin" },
    { name: "Japan", capital: "Tokyo" },
  ];

  function handleCountrySelect(countryName) {
    const country = countries.find(function (c) {
      return c.name === countryName;
    });
    setCapital(country ? country.capital : "");
  }

  return (
    <div>
      <h1>Country and Capital Finder</h1>
      <CountryDropdown countries={countries}
onSelect={handleCountrySelect} />
      {capital && <p>Capital: {capital}</p>}
    </div>
  );
}
```

```
export default App;
```

2. Develop a react application and: i) Build your own Button component and render it three times. On user click, it should alert which button was clicked. (React JS). ii) Use the useState React hook to track how many times a button is clicked, and display the number

Button.js

```
function Button(props) {
   function handleClick() {
     props.onClick(props.label);
   }

   return (
     <button onClick={handleClick}>
       {props.label}
     </button>
   );
 }
 export default Button;
```

App.js

```
import React, { useState } from "react";
import Button from "./Button";

function App() {
```

```
  const [clickCounts, setClickCounts] = useState({ Button1: 0, Button2:
0, Button3: 0 });

  function handleButtonClick(label) {
    alert(label + " was clicked!");
    setClickCounts(function (prevCounts) {
      return {
        ...prevCounts,
        [label]: prevCounts[label] + 1,
      };
    });
  }

  return (
    <div>
      <h1>Button Click Tracker</h1>
      <Button label="Button1" onClick={handleButtonClick} />
      <Button label="Button2" onClick={handleButtonClick} />
      <Button label="Button3" onClick={handleButtonClick} />
      <div>
        <p>Button1 Clicks: {clickCounts.Button1}</p>
        <p>Button2 Clicks: {clickCounts.Button2}</p>
        <p>Button3 Clicks: {clickCounts.Button3}</p>
      </div>
    </div>
  );
}

export default App;
```

3. Create a custom component for rendering each joke present in an array. Using the map function, map through each object in the array. Use the custom component to render each object

Joke.js

```
function Joke(props) {
   return (
     <div>
       <h3>{props.joke.setup}</h3>
       <p>{props.joke.punchline}</p>
     </div>
   );
 }
 export default Joke;
```

App.js

```
import Joke from "./Joke";
function App() {
  const jokes = [
    { id: 1, setup: "Why don't scientists trust atoms?", punchline:
"Because they make up everything!" },
    { id: 2, setup: "Why did the scarecrow win an award?", punchline:
"Because he was outstanding in his field!" },
    { id: 3, setup: "What do you call fake spaghetti?", punchline: "An
impasta!" },
    { id: 4, setup: "How do you organize a space party?", punchline: "You
planet!" },
  ];
```

```
  return (
    <div>
      <h1>Joke List</h1>
      {jokes.map(function (joke) {
        return <Joke key={joke.id} joke={joke} />;
      })}
    </div>
  );
}

export default App;
```

4. create a multi page React application with a navigation bar component and routes using react-router-dom.

npm install react-router-dom

Home.js

```
function Home() {
    return <h2>Welcome to the Home Page!</h2>;
  }
  export default Home;
```

About.js

```
function About() {
    return <h2>About Us</h2>;
  }
export default About;
```

Contact.js

```
function Contact() {
   return <h2>Contact Us</h2>;
 }
 export default Contact;
```

Navbar.js

```
import { Link } from "react-router-dom";
function Navbar() {
   return (
     <nav>
       <ul>
         <li>
           <Link to="/">Home</Link>
         </li>
         <li>
           <Link to="/about">About</Link>
         </li>
         <li>
           <Link to="/contact">Contact</Link>
         </li>
       </ul>
     </nav>
   );
 }
 export default Navbar;
```

App.js

```
import { BrowserRouter as Router, Routes, Route} from
"react-router-dom";
import Home from "./Home";
import About from "./About";
import Contact from "./Contact";
import Navbar from "./Navbar";
function App() {
  return (
    <Router>
      <div>
        <Navbar />
        <Routes>
          <Route path="/" element={<Home />} />
          <Route path="/about" element={<About />} />
          <Route path="/contact" element={<Contact />} />
        </Routes>
      </div>
    </Router>
  );
}

export default App;
```

5. Create an HTTP server listening on port 1337, which sends Hello, World! to the browser and using Express.

1. **HTTP Server**

   Create a folder
   Open terminal in vs code

Run command:

npm init -y

Create a file Index.js

Index.js

```
const http = require('http'); // Import the HTTP module

// Create the server
const server = http.createServer((req, res) => {
  res.writeHead(200, { 'Content-Type': 'text/plain' }); // Set response headers
  res.end('Hello, World!'); // Send "Hello, World!" as the response
});

// Make the server listen on port 1337
server.listen(1337, () => {
  console.log('Server is running on http://localhost:1337'); // Log message
});
```

Run Command:

node Index.js

Go to Browser **http://localhost:1337**

2. **Using Express**

Create a folder

Open terminal in vs code

Run command:

npm init -y

npm install express

Create a file Server.js

Server.js

```
const express = require('express'); // Import Express
const app = express(); // Create an instance of Express

// Define a route for the root URL ("/")
app.get('/', (req, res) => {
   res.send('Hello, World!'); // Send "Hello, World!" as the
response
});

// Make the app listen on port 1337
app.listen(1337, () => {
   console.log('Server is running on http://localhost:1337'); // Log
message
});
```

Run Command:

node Server.js

Go to Browser **http://localhost:1337**

6

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

function App() {
  const [users, setUsers] = useState([]);
  const [name, setName] = useState("");

  useEffect(function () {
    fetchUsers();
  }, []);

  function fetchUsers() {
    fetch("https://67a1bf3b5bcfff4fabe34c24.mockapi.io/students/users")
      .then(function (response) {
        return response.json();
      })
      .then(function (data) {
        setUsers(data);
      })
      .catch(function (error) {
        console.error("Error fetching users:", error);
      });
  }

  function handleInputChange(event) {
    setName(event.target.value);
  }

  function addUser() {
```

```
    if (name.trim() === "") {
      alert("Name cannot be empty");
      return;
    }

    fetch("https://67a1bf3b5bcfff4fabe34c24.mockapi.io/students/users",
{
      method: "POST",
      headers: {
        "Content-Type": "application/json",
      },
      body: JSON.stringify({ name: name }),
    })
      .then(function (response) {
        return response.json();
      })
      .then(function (newUser) {
        setUsers(function (prevUsers) {
          return prevUsers.concat(newUser);
        });
        setName("");
      })
      .catch(function (error) {
        console.error("Error adding user:", error);
      });
  }

  function deleteUser(id) {
    fetch("https://67a1bf3b5bcfff4fabe34c24.mockapi.io/students/users/"
+ id, {
```

```
    method: "DELETE",
  })
    .then(function () {
      setUsers(function (prevUsers) {
        return prevUsers.filter(function (user) {
          return user.id !== id;
        });
      });
    })
    .catch(function (error) {
      console.error("Error deleting user:", error);
    });
}

return (
  <div className="app">
    <h1>User Management</h1>

    <div className="add-user">
      <input
        type="text"
        placeholder="Enter name"
        value={name}
        onChange={handleInputChange}
      />
      <button onClick={addUser}>Add User</button>
    </div>

    <ul className="user-list">
      {users.map(function (user) {
```

```
          return (
            <li key={user.id}>
              {user.name}
              <button onClick={function () { deleteUser(user.id);
}}>Delete</button>
            </li>
          );
        })}
      </ul>
    </div>
  );
}

export default App;
```

Practice Program (not to be written in record)

Steps:

- Create the Backend Folder
- npm init –y
- npm install express mongoose cors
- create server.js and paste the backend code
- node server.js
- Outside the backend folder, create a new folder (e.g., frontend).
- npx create-react-app
- Replace App.js code
- npm start

```
// server.js (Node.js backend)
const express = require('express');
const mongoose = require('mongoose');
const cors = require('cors');

const app = express();

// Middleware
app.use(express.json());
app.use(cors());

// MongoDB connection
const mongoURI = 'mongodb://127.0.0.1:27017/yourDatabaseName';
mongoose.connect(mongoURI, { useNewUrlParser: true, useUnifiedTopology: true })
  .then(() => console.log('Connected to MongoDB'))
  .catch(err => console.error('MongoDB connection error:', err));

// Define a Mongoose schema
const userSchema = new mongoose.Schema({
  name: { type: String, required: true },
  age: { type: Number, required: true },
});
```

```
const User = mongoose.model('User', userSchema);

// API routes
app.post('/api/users', async (req, res) => {
  try {
    const { name, age } = req.body;
    const newUser = new User({ name, age });
    await newUser.save();
    res.status(201).json(newUser);
  } catch (err) {
    res.status(500).json({ error: 'Failed to save user' });
  }
});

// Start server
const PORT = 5000;
app.listen(PORT, () => console.log(`Server running on port ${PORT}`));


// React Frontend (App.js)
import React, { useState } from 'react';

const App = () => {
  const [formData, setFormData] = useState({ name: '', age: '' });
  const [message, setMessage] = useState('');

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

  const handleSubmit = async (e) => {
    e.preventDefault();
    try {
      const response = await fetch('http://localhost:5000/api/users', {
        method: 'POST',
        headers: {
          'Content-Type': 'application/json',
        },
        body: JSON.stringify(formData),
```

```
    });

    if (!response.ok) {
      throw new Error('Failed to save user');
    }

    const data = await response.json();
    setMessage(`User saved: ${data.name}, Age: ${data.age}`);
    setFormData({ name: '', age: '' });
  } catch (err) {
    setMessage('Failed to save user');
  }
};

return (
  <div className="App">
    <h1>Save User to MongoDB</h1>
    <form onSubmit={handleSubmit}>
      <div>
        <label>Name:</label>
        <input
          type="text"
          name="name"
          value={formData.name}
          onChange={handleChange}
          required
        />
      </div>
      <div>
        <label>Age:</label>
        <input
          type="number"
          name="age"
          value={formData.age}
          onChange={handleChange}
          required
        />
      </div>
      <button type="submit">Save</button>
    </form>
```

```
      {message && <p>{message}</p>}
    </div>
  );
};

export default App;

// Steps to Run:
// 1. Install Node.js and MongoDB Compass.
// 2. Create a MongoDB database named `yourDatabaseName`.
// 3. Install dependencies:
//    - Backend: `npm install express mongoose cors`
// 4. Run the backend server: `node server.js`.
// 5. Start the React app: `npm start`.
// 6. Access the form at http://localhost:3000 and enter data to save to MongoDB.
```