**Practical 1:**

**Create a new react application which shows the implementation of Table.**

Applying Table component

First of all, create a new react application and start it using below command.

create-react-app myapp

cd myapp npm start

|  |
| --- |
| Next, install the bootstrap and react-bootstrap library using below command, |
| npm install --save bootstrap react-bootstrap |
| Next, open *App.css* (src/App.css) and remove all CSS classes. |
| // remove all css classes |
| Next, create a simple table  component, *SimpleTable* (src/SimpleTable.js) and render a table as shown below – |

App.js

import { Table } from 'react-bootstrap'; function SimpleTable() { return (

<Table striped bordered hover>

<thead>

<tr>

<th>#</th>

<th>Name</th>

<th>Age</th>

|  |
| --- |
| <th>Email</th>  </tr>  </thead>  <tbody>  <tr>  <td>1</td>  <td>John</td>  <td>25</td>  <td>john.example@tutorialspoint.com</td>  </tr>  <tr>  <td>1</td>  <td>Peter</td>  <td>15</td>  <td>peter.example@tutorialspoint.com</td>  </tr>  <tr>  <td>1</td>  <td>Olivia</td>  <td>23</td>  <td>olivia.example@tutorialspoint.com</td>  </tr>  </tbody>  </Table>  );  } export default SimpleTable; |

index.js

import './App.css'

|  |
| --- |
| import "bootstrap/dist/css/bootstrap.min.css"; import SimpleTable from './SimpleTable' function App() { return (  <div className="container">  <div style={{ padding: "10px" }}>  <div>  <SimpleTable />  </div>  </div>  </div>  );  } export default App; |

**Practical 2:**

**implementation of Counter.**

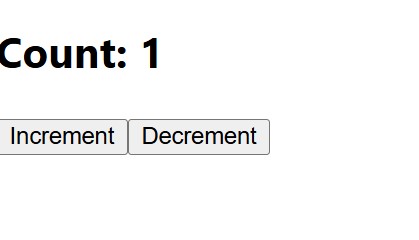
**Counter Example:**

App.js

|  |
| --- |
| import React, { useState } from 'react';    function Counter() { const [count, setCount] = useState(0); // Initialize count state to 0    const increment = () => { setCount(count + 1); // Update count state  };    const decrement = () => { setCount(count - 1); // Update count state  };    return (  <div>  <h2>Count: {count}</h2>  <button onClick={increment}>Increment</button>  <button onClick={decrement}>Decrement</button> </div>  );  }    export default Counter; |

index.js

|  |
| --- |
| import React from 'react'; import ReactDOM from 'react-dom/client'; import './index.css'; import Counter from './App'; import reportWebVitals from './reportWebVitals';    const root =  ReactDOM.createRoot(document.getElementById('root')); root.render( <Counter/>  );    reportWebVitals(); |



**Practical 3:**

**implementation of List.**

App.js

|  |
| --- |
| import React from 'react';    function ShoppingList() { const products = [  { id: 1, title: 'Apples' },  { id: 2, title: 'Bananas' },  { id: 3, title: 'Milk' },  ];    return (  <div>  <h2>Shopping List</h2>  <ul>  {products.map(product => (  <li key={product.id}>{product.title}</li>  ))}  </ul> </div>  );  }    export default ShoppingList; |

index.js

import React from 'react';

|  |
| --- |
| import ReactDOM from 'react-dom/client'; import './index.css'; import ShoppingList from './App'; import reportWebVitals from './reportWebVitals';    const root =  ReactDOM.createRoot(document.getElementById('root')); root.render(  <ShoppingList/>  );  reportWebVitals(); |

Output:



**Practical 4:**

**implementation of Router which helps to route to pages**

Add React Router

To add React Router in your application, run this in the terminal from the root directory of the application:

npm i -D react-router-dom

or

npm install react-router-dom

**Create React App using following command:** npx create-react-app routingeg

Within the src folder, we'll create a folder named pages with several files: src\:

* Layout.js
* Home.js
* Blogs.js
* Contact.js
* NoPage.js

Each file will contain a very basic React component.

Use React Router to route to pages based on URL:

App.js:

import { Route, Routes, BrowserRouter } from "react-routerdom";

|  |
| --- |
| import "./App.css" import Home from "./Home"; import About from "./About"; import Contact from "./Contact"; import Navigate from "./Navigate"; function App() { return (  <div className="App">  <BrowserRouter>  <Routes>  <Route path="/" element={<Navigate />}>  <Route index element={<Home />} />    <Route path="About" element={<About />} />    <Route path="Contact" element={<Contact />} />  </Route>  </Routes>  </BrowserRouter>  </div>  );  } export default App; |

**Practical 5**

**Create and validate the user form in React.**

Create new React App with name Practical4b by using following command:

npx create-react-app practical4b

Here we are renaming App.js to UserForm.js

|  |
| --- |
| import React, { useState } from 'react';    function UserForm() { const [formData, setFormData] = useState({ username: '', email: '', password: '',  }); const [errors, setErrors] = useState({});    const handleChange = (e) => { const { name, value } = e.target; setFormData((prevData) => ({ ...prevData, [name]: value }));  };    const validateForm = () => { let newErrors = {}; if (!formData.username) { newErrors.username = 'Username is required';  }  if (!formData.email) { newErrors.email = 'Email is required';  } else if (!/\S+@\S+\.\S+/.test(formData.email)) { |

|  |
| --- |
| newErrors.email = 'Email address is invalid';  }  // Add more validation rules for password, etc. setErrors(newErrors); return Object.keys(newErrors).length === 0; // Return true if no errors  };    const handleSubmit = (e) => {  e.preventDefault(); // Prevent default form submission if (validateForm()) {  // Form is valid, proceed with submission (e.g., API call) console.log('Form submitted:', formData);  } else { console.log('Form has errors');  }  };    return (  <form onSubmit={handleSubmit}>  <div>  <label htmlFor="username">Username:</label>  <input type="text" id="username" name="username" value={formData.username} onChange={handleChange}  />  {errors.username && <p style={{ color: 'red'  }}>{errors.username}</p>} |
| </div>  <div>  <label htmlFor="email">Email:</label>  <input type="email" id="email" name="email" value={formData.email} onChange={handleChange}  />  {errors.email && <p style={{ color: 'red'  }}>{errors.email}</p>}  </div>  {/\* Add password field and other inputs similarly \*/}  <button type="submit">Register</button>  </form>  );  }    export default UserForm; |

index.js

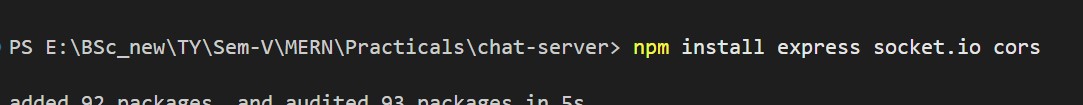
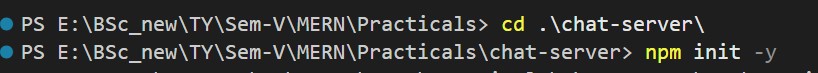
|  |
| --- |
| import React from 'react'; import ReactDOM from 'react-dom/client'; import './index.css'; import reportWebVitals from './reportWebVitals'; import UserForm from './UserForm';    const root =  ReactDOM.createRoot(document.getElementById('root')); root.render( |
| <UserForm/>  );  reportWebVitals(); |

Run the app by using following command:

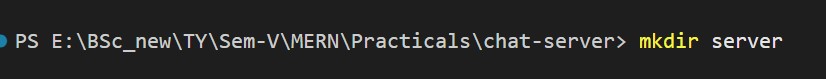
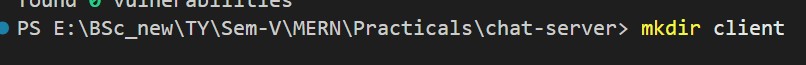
**Practical 6**

**Create Chat application by using Socket.IO**

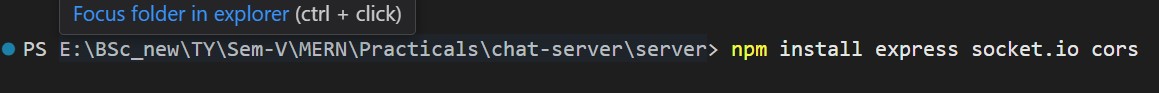
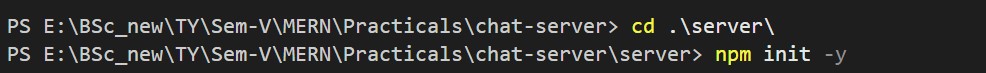
Create Folder chat-server



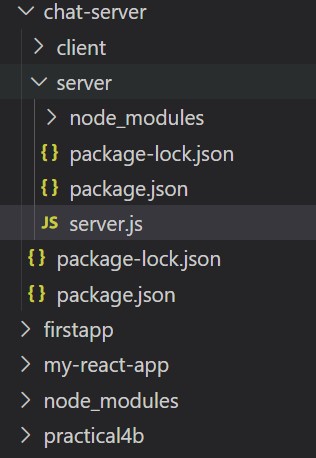
Create folders client and server



Be inside the server folder

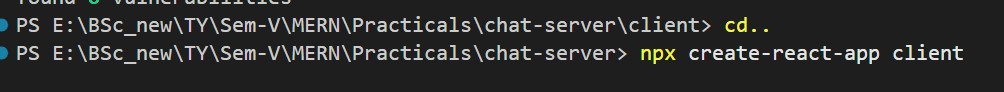


Create a file in server with name server.js

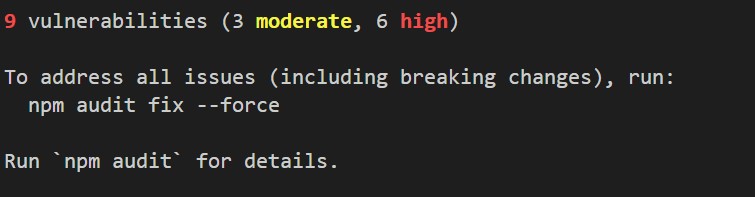


|  |
| --- |
| // server/server.js const express = require('express'); const http = require('http'); const socketIo = require('socket.io'); const cors = require('cors');    const app = express(); const server = http.createServer(app); const io = socketIo(server, { cors: { origin: '\*', // Allow all origins for development, restrict in production methods: ['GET', 'POST']  }  });    app.use(cors()); // Use CORS middleware for Express routes if needed |
| io.on('connection', (socket) => { console.log('A user connected:', socket.id);    socket.on('message', (message) => { console.log('Message received:', message); io.emit('message', message); // Broadcast message to all connected clients  });    socket.on('disconnect', () => { console.log('User disconnected:', socket.id);  });  });    const PORT = process.env.PORT || 5000; server.listen(PORT, () => { console.log(`Server running on port ${PORT}`);  }); |

Create a react app with name client using the following



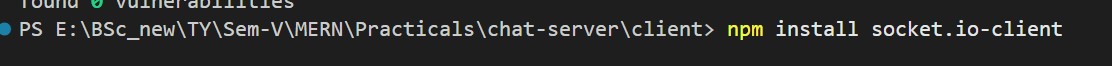
If some warning occurs like this



Then the following command



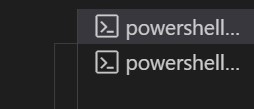
Install dependencies



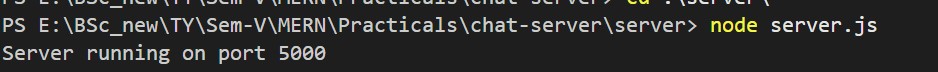
In under client src🡪 App.js

|  |
| --- |
| // client/src/App.js import React, { useState, useEffect } from 'react'; import io from 'socket.io-client';    const socket = io('http://localhost:5000'); // Connect to your server    function App() { const [message, setMessage] = useState(''); const [messages, setMessages] = useState([]);    useEffect(() => { socket.on('message', (msg) => { setMessages((prevMessages) => [...prevMessages, msg]);  });    return () => { socket.off('message');  };  }, []);    const sendMessage = (e) => { e.preventDefault(); if (message.trim()) { |
| socket.emit('message', message); setMessage('');  }  };  return ( <div>  <h1>Real-time Chat</h1>  <div>  {messages.map((msg, index) => (  <p key={index}>{msg}</p>  ))}  </div>  <form onSubmit={sendMessage}>  <input type="text" value={message} onChange={(e) => setMessage(e.target.value)} placeholder="Type a message..."  />  <button type="submit">Send</button>  </form>  </div>  );  }    export default App; |

Open two terminal

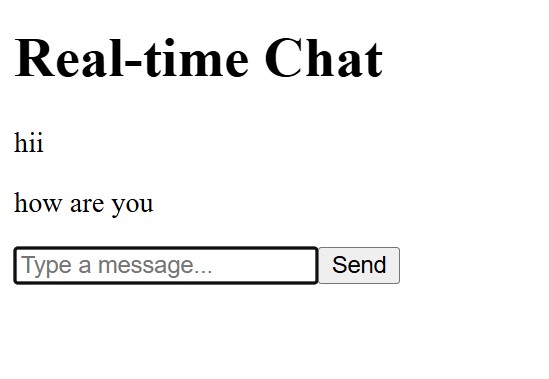


In one terminal

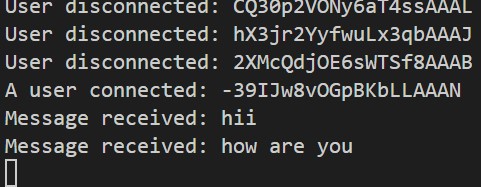


In second terminal





We can check message in server terminal



ERROR in [eslint] Failed to load config "react-app" to extend from.

npm install --save-dev eslint-config-react-app

**Practical 7:**

**Fetch the API data and render the list on React app.**

Create new React App with name Practical5 by using following command:

npx create-react-app practical5

Here we are renaming App.js to FetchData.js import React, { useEffect, useState } from 'react'

//import './App.css'; function FetchData() { const [records, setrecords] = useState([]) useEffect(()=>{ fetch('https://jsonplaceholder.typicode.com/users')

.then(response => response.json())

.then(data => setrecords(data))

.catch(err => console.log(err))

},[] ) return (

<div>

<ul> {records.map((list,index)=>(

<li key={index} > {list.id} | {[list.name}](http://list.name/)</li> ))}

</ul>

</div> );

}export default FetchData;

Run the app by using following command:



**Practical 8:**

**In Angular create TODO list with different components.** [Todolist.js](http://todolist.js/) import React,{useState} from "react" function ToDoList(){

const [tasks,setTasks] = useState(["Eat Breakfast","Take Shower"]); const [newTask,setNewTask] = useState("");

function handleInputChange(event) { setNewTask(event.target.value);

}

function addTask(){ if(newTask.trimEnd()!==""){ setTasks(t=>[...tasks,newTask]); setNewTask("");

}

} function deleteTask(index){ const updatedTasks = tasks.filter((\_, i) => i !== index); setTasks(updatedTasks);

}

function moveTaskUp(index){

if(index>0){ const updatedTasks=[...tasks];

[updatedTasks[index],updatedTasks[index-1]]= [updatedTasks[index-1],updatedTasks[index]]; setTasks(updatedTasks);

}

} function moveTaskDown(index){ if(index < tasks.length - 1){ const updatedTasks=[...tasks];

[updatedTasks[index],updatedTasks[index+1]]= [updatedTasks[index+1],updatedTasks[index]]; setTasks(updatedTasks);

}

} return(

<div className="to-do-list">

<h1>To-Do-List</h1>

<div> <input type="text" placeholder="Enter a task.." value={newTask} onChange={handleInputChange}></input>

<button className="add-button" onClick={addTask}></button>

</div>

<ol>

{tasks.map((task,index)=>

<li key={index}>

<span className="text">{task}</span>

<button className="delete-button" onClick={()=>deleteTask(index)}>

Delete

</button>

<button

className="move-button" onClick={()=>moveTaskUp(index)}>

Up

</button> <button className="move-button" onClick={()=>moveTaskDown(index)}>

Down

</button>

</li>

)}

</ol>

</div>);

}

export default ToDoList

[App.js](http://app.js/) import ToDoList from "./ToDoList.js"; function App(){ return(<ToDoList/>)

} export default App [Index.js](http://index.js/) import React from 'react'; import ReactDOM from 'react-dom/client'; import './indexx.css'; import App from './App'; import reportWebVitals from './reportWebVitals';

const root =

ReactDOM.createRoot(document.getElementById('root')); root.render(

<React.StrictMode>

<App />

</React.StrictMode>

);

// If you want to start measuring performance in your app, pass a function

// to log results (for example: reportWebVitals(console.log))

// or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitals reportWebVitals();

**Practical 9:**

**Fetch the API data and render the list on Angular app.** [App.js](http://app.js/) angular.module('todoApp', [])

.controller('TodoController', function($scope) {

$scope.todos = [

{ text: 'Learn AngularJS', done: false },

{ text: 'Build a To-Do list', done: true }

];

$scope.addTodo = function() { if ($scope.newTodoText) {

$scope.todos.push({ text: $scope.newTodoText, done: false });

$scope.newTodoText = ''; // Clear input

}

};

$scope.removeTodo = function(todoToRemove) {

$scope.todos = $scope.todos.filter(function(todo) { return todo !== todoToRemove;

});

};

$scope.remaining = function() { var count = 0; angular.forEach($scope.todos, function(todo) { count += todo.done ? 0 : 1;

}); return count;

};

});

Index.html

<!DOCTYPE html>

<html lang="en" ng-app="todoApp">

<head>

<meta charset="UTF-8">

<title>AngularJS To-Do List</title>

<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/ang ular.min.js"></script>

<script src="app.js"></script>

</head>

<body>

<div ng-controller="TodoController">

<h2>My To-Do List</h2>

<input type="text" ng-model="newTodoText" placeholder="Add a new task">

<button ng-click="addTodo()">Add Task</button>

<ul>

<li ng-repeat="todo in todos">

<input type="checkbox" ng-model="todo.done">

<span ng-class="{ 'done': todo.done }">{{ todo.text

}}</span>

<button ng-

click="removeTodo(todo)">Remove</button>

</li>

</ul>

<p>Remaining tasks: {{ remaining() }}</p>

</div>

<script defer src="https://static.cloudflareinsights.com/beacon.min.js/vcd15 cbe7772f49c399c6a5babf22c1241717689176015" integrity="sha512-

ZpsOmlRQV6y907TI0dKBHq9Md29nnaEIPlkf84rnaERnq6z vWvPUqr2ft8M1aS28oN72PdrCzSjY4U6VaAw1EQ==" data-cfbeacon='{"version":"2024.11.0","token":"499e684b7b104387 8977050a0a606794","r":1,"server\_timing":{"name":{"cfCach eStatus":true,"cfEdge":true,"cfExtPri":true,"cfL4":true,"cfOri gin":true,"cfSpeedBrain":true},"location\_startswith":null}}' crossorigin="anonymous"></script>

</body>

</html>