SRI KRISHNA COLLEGE OF ENGINEERING AND TECHNOLOGY

(AN AUTONOMOUS INSTITUTION,
AFFILIATED TO ANNA UNIVERSITY, CHENNAI)

COIMBATORE – 641008

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

21IT301- WEB DEVELOPMENT USING REACT LABORATORY

CONTINUOUS ASSESSMENT RECORD

Submitted by

Name: ……..……..

Register No. : …….……..……

Degree & Branch: ……..…

Class & Semester: …….……..…

Academic Year: …….……

SRI KRISHNA COLLEGE OF ENGINEERING AND TECHNOLOGY

(AN AUTONOMOUS INSTITUTION,
AFFILIATED TO ANNA UNIVERSITY, CHENNAI)

COIMBATORE – 641008

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

21IT301- WEB DEVELOPMENT USING REACT LABORATORY

Continuous Assessment Record

Submitted by

Name : ……..…….……..…… **Register No. : .....……**

Class/Semester :…….. **Degree & Branch: ……..**

BONAFIDE CERTIFICATE

This is to certify that this record is the bonafide record of work done by Mr./Ms. _____ during the academic year 2022 – 2023.

Faculty In-charge **Head of the Department**

Submitted for the University practical examination held on _____

INTERNAL EXAMINER **EXTERNAL EXAMINER**

SRI KRISHNA COLLEGE OF ENGINEERING AND TECHNOLOGY

(AN AUTONOMOUS INSTITUTION,
AFFILIATED TO ANNA UNIVERSITY, CHENNAI)
COIMBATORE – 641008

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

21IT301- WEB DEVELOPMENT USING REACT LABORATORY

Record of laboratory work

EVEN SEMESTER - 2022-2023

Name of the Faculty	**Ms.N.Kalaivani** **Ms.D.V.Soundari** **Ms.S.Praseetha**

CONTINUOUS EVALUATION SHEET

REFERENCES RUBRICS TABLE

Criteria	**Range of Marks**			
	Excellent	**Good**	**Average**	**Below Average**
Objective & Description with sample data (20)	18-20	17-18	15-16	0-14
Formation of Program (30)	27-30	21-26	15-20	0-14
Execution and Testing (30)	27-30	21-26	15-20	0-14
Documentation (10)	9-10	7-8	5-6	0-4
Viva (10)	9-10	7-8	5-6	0-4
Overall Marks	**90-100**	**70-89**	**50-69**	**0-49**

INDEX

Ex.No	Name of the Experiment	Page No.	Marks	Signature
1	a. Study of React Installation and Terminal Commands			
	b. Demonstration of a Stateless Functional Component			
2	Demonstration of Stateful Class Component			
3	Implementation of Conditional Rendering using Class Component			
4	Implementation of Communication between Parent and child Components			
5	Designing a Registration Form with material UI			
6	Design a Custom Navigation bar using React			
7	Implementation of React component to handle HTTP requests			
8	Implementation of a Dropdown component using React			
9	Implementation of Routing in React			
10	Implementation of FORM validation in React			

SRI KRISHNA COLLEGE OF ENGINEERING AND TECHNOLOGY

COIMBATORE – 641008

(AN AUTONOMOUS INSTITUTION, AFFILIATED TO ANNA UNIVERSITY, CHENNAI)

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

21IT301- WEB DEVELOPMENT USING REACT LABORATORY

EVEN SEMESTER: 2022-2023

Components	EX1	EX2	EX3	EX4	EX5	EX6	EX7	EX8	EX9	EX10
Objective & Description with sample data (20 Marks)										
Formation of Program (30 Marks)										
Execution and Testing (30 Marks)										
Documentation (10 Marks)										
Viva (10 Marks)										
TOTAL										
AVERAGE										
Consolidated Mark (100)										
Faculty Signature										

1.a)Study of React Installation and Terminal Commands

Objective

To install the node js.

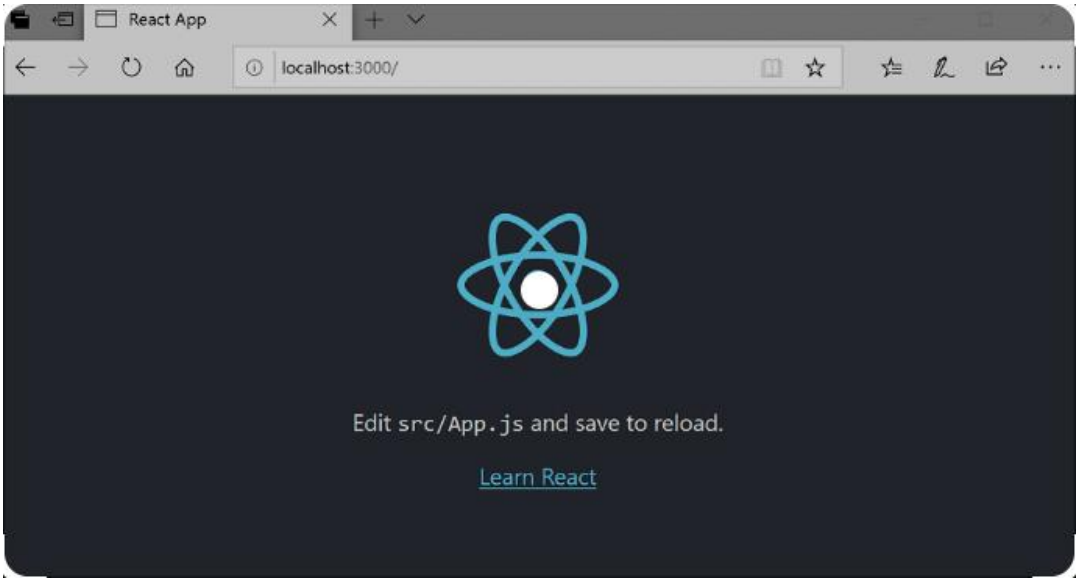
System and software tools required

- Visual Studio Code

Definitions /Theory/ Steps

1. Open the browser in your computer.
2. Download the node js from the browser.
3. And set up the node js .
4. Type the command “node -v” to see the version of node and “npm -v” to see the version of
   npm(Node Package Manager).
5. Open the terminal in VS Code and type the command “npx create-react-app my_react” to create the new react app.
6. To start the react app type the command “npm start”.
7. You can use the command “npm cache clean –force” to clear the cache in react app.
8. Now the react app will open in the browser.

React app:



1.b) Demonstration of a Stateless Functional Component

Objective

To Study the Stateless Functional Component.

System and software tools required

- Visual Studio Code

Description

A React functional component is a simple JavaScript function that accepts props and returns a React element. After the introduction of React Hooks, writing functional components has become the standard way of writing React components in modern applications.

Algorithm

1. Create the new js file name it as fcomponent.js.
2. Create the new function component and name it as demo in fcomponent.
3. Inside the function componentreturn the html element.
4. And export the function component demo.
5. Now import the function component from fcomponent file.
6. Inside the root.render use the component.
7. Now the output will display in the react app.

Program

fcomponent.js:

```
import React from 'react'
    function demo() {
        return(
<div>
<h1>Hello everyone, Welcome to react.</h1>
</div>
        )
    }
    export default demo
```

Index.js:

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import Fcomponent from './fcomponent';
const root = ReactDOM.createRoot(document.getElementById('root'));
```

```
root.render(
        <React.StrictMode>
        <Fcomponent/>
        </React.StrictMode>
          );
```

Sample Output

function Component:

Hello everyone ,Welcome to react.

Result

Thus the function component is successfully implemented.

2. Demonstration of a Statefull Class Component

Objective

To Study the Statefull Class Component.

System and software tools required

- Visual Studio Code

Description

A class component is a more featured way to define a React component. It also acts like a function that receives props. The component has to include the extends React.Component statement, this statement creates an inheritance to React.Component, and gives your component access to React.Component's functions.The component also requires a render() method, this method returns HTML

Algorithm

1.Create the new js file name it as ccomponent.js.

2.Create the new function component and name it as Demo in ccomponent.

3.Inside the render function of the class componentDemo return the html element.

4.And export the class component Demo.

5. Now import the class component from ccomponent file.

6.Inside the root.render use the component.

7.Now the output will display in the react app.

Program

ccomponent.js:

```
  import React from 'react'
        class Ccomponent extends React.Component {
        state = {  }
        render() {
            return (
<div>
<h1>Class Component:</h1>
<h2>Hello Everyone,Welcome To React</h2>
</div>
              );
          }
}
export default Ccomponent;
```

Index.js:

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import Ccomponent from './ccomponent';
const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
        <React.StrictMode>
                <Ccomponent/>
        </React.StrictMode>
);
```

Sample Output

Class Component:

Hello Everyone,Welcome To React

Result

Thus the Class component is successfully implemented.

3. Implementation of Conditional Rendering using Class Component

Objective

To Implement the Conditional Rendering using Class Component.

System and software tools required

- Visual Studio Code

Description

A conditional rendering is a piece of content that is displayed or rendered when a predefined condition is met. You can use conditional renderings to control the way visitors view and interact with your website.

Algorithm

1.Create the new js file name it as conditionalrendering.js.

2.Create the new function component and name it as Demo in conditionalrendering.

3.Create three functions,changeColor1, changeColor2 ,changeCount for changing colors of the Buttons and text.

4.Inside the render function of the class componentreturn the html Button elements.

5.And export the class component Conditionalrendering

6. Now import the class component from conditionalrendering file.

7.Inside the root.render use the component.

8.Now the output will display in the react app.

Program

conditionalrendering.js:

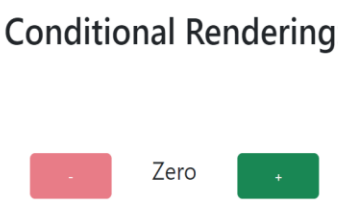
```
import React, { Component } from 'react';
import 'bootstrap/dist/css/bootstrap.css';
class Conditionalrendering extends Component {
    state = { count: 0 ,name:"Zero"}
      changeCount(){
        if(this.state.count===0 )
                return"Zero";
       else if(this.state.count===10)
                return"Ten";
      else
              return this.state.count;
```

```
            }
             changeColor1(){
             let str="btn btn-";
            if(this.state.count===0){
                    str+="danger";
              }
          else {
                str+="success";
            }
         return str;
       }
      changeColor2(){
          let str="btn btn-";
         if(this.state.count===10){
         str+="danger";
      }
       else {
          str+="success";
        }
        return str;
     }
     render() {
       return (<div style={{textAlign:'center',padding:50}}>
              <h1 style={{padding:100}}> Conditional Rendering:</h1>
             <button className={this.changeColor1()}style={{width:100,height:50}}
                   disabled={this.state.count===0} onClick={() =>
                  { this.setState({ count: this.state.count - 1 }) }}>-  </button>
               <span style={{padding:50,fontSize:27}}>{this.changeCount()}</span>
              <button className={this.changeColor2()}
                 style={{width:100,height:50}}disabled={this.state.count===10}
            onClick={() => { this.setState({ count: this.state.count + 1 }) }}>+</button>
     </div>);
   }
}
export default Conditionalrendering;
```

Index.js:

```
        import React from 'react';
        import ReactDOM from 'react-dom/client';
        import Conditional from './conditional';
        const root = ReactDOM.createRoot(document.getElementById('root'));
        root.render(
<React.StrictMode>
<Conditional/>
</React.StrictMode>);
```

Sample Output



Result:

Thus the conditional rendering is successfully implemented.

4.Implementation of Communication between Parent and child Components

Objective

To Implement the Communication between Parent and Child Component.

System and software tools required

- Visual Studio Code

Description

In React parent components can communicate to child components using a special property defined by React called as Props. All the components in React will be having this proper defined by default which will hold all the properties as key value pairs that are sent from the parent component.

Algorithm

1.Create the new js file names them as parent.js,child.js.

2.Create the new class component and name it as Child in child.js and return the html elements..

3.Inside the render function of the class componentit should return the html element.

4.And export the class component Child.

5. Now import the class component from child file.

6.Inside the render function of the parent's class component it should return the child component

7.Now the output will display in the react app.

Program

parent.js:

```
import React from 'react';
import Child from './child';
class Parent extends React.Component{
    state = {
        name: "",
    }
    handleCallback = (childData) =>{
        this.setState({name: childData})
    }
    render(){
        const {name} = this.state;
        return(
            <div style={{textAlign:'center'}}>
```

```
<h1>Child and Parent Components</h1>
                                <Child parentCallback = {this.handleCallback}/>
                                {name}
                        </div>
                )
        }
   }
   export default Parent;
```

Child.js:

```
  import React from 'react'
  class Child extends React.Component{

        onTrigger = (event) => {
                this.props.parentCallback(event.target.myname.value);
                event.preventDefault();
        }

        render(){
                return(
                <div>
                        <form onSubmit = {this.onTrigger}>
                                <input type = "text"
                                name = "myname" placeholder = "Enter Name"/>
                                <br></br><br></br>
                                <input type = "submit" value = "Submit"/>
                                <br></br><br></br>
                        </form>
                </div>
                )
        }
    }
   export default Child
```

Index.js:

```
            import React from 'react';
            import ReactDOM from 'react-dom/client';
            import Parent from './parentchild';
            const root = ReactDOM.createRoot(document.getElementById('root'));
            root.render(
<React.StrictMode>
<Parentt/>
</React.StrictMode>
            );
```

Sample Output

Result:

Thus the conditional rendering is successfully implemented.

5. Create material UI Card using React

Objective

To Create the material UI Card using React.

System and software tools required

- Visual Studio Code

Description

A react card component is a content container. It incorporates options for images, headers, and footers, a wide variety of content, contextual background colors, and excellent display options.

Algorithm

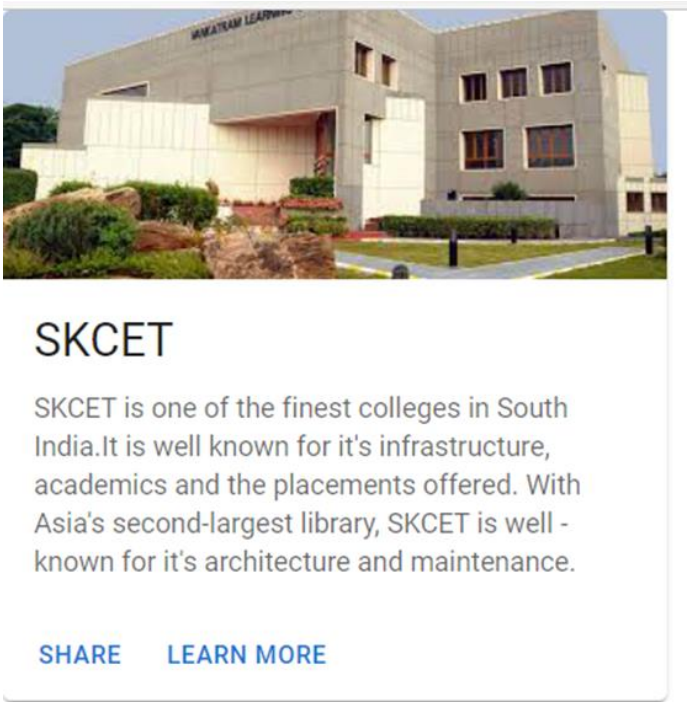
1. Create the UiCard.js.
2. Create the new function component and name it as a demo.
3. Inside the function component return the html element.
4. And export the function component demo.
5. Now import the function component from UiCrd file.
6. Inside the root.render use the component.
7. Now the output will display in the react app.

Program

```
import * as React from 'react';
import Card from '@mui/material/Card';
import CardActions from '@mui/material/CardActions';
import CardContent from '@mui/material/CardContent';
import CardMedia from '@mui/material/CardMedia';
import Button from '@mui/material/Button';
import Typography from '@mui/material/Typography';

export default function MediaCard() {
```

```
  return (
        <Card sx={{ maxWidth: 345 }}>
        <CardMedia
        component="img"
        height="140"
image=" /rjE3wp5woUA1wVtTHRbthQoZPY2ZeI4UKGhM//2Q=="
        alt="green iguana"
        />
        <CardContent>
        <Typography gutterBottom variant="h5" component="div">
        SKCET
        </Typography>
        <Typography variant="body2" color="text.secondary">
        SKCET is one of the finest colleges in South India.It is well known for it's infrastructure, academics and the placements offered. With Asia's second-largest library, SKCET is well -known for it's architecture and maintenance.
        </Typography>
        </CardContent>
        <CardActions>
        <Button size="small">Share</Button>
        <Button size="small">Learn More</Button>
        </CardActions>
        </Card>
  );
}
```

Sample Output:



RESULT

Thus the Program is Executed Successfully.

6. Design a Custom Navigation bar using React

Objective

To Design a Custom Navigation bar using React.

System and software tools required

- Visual Studio Code

Description

Bootstrap navbar is a horizontal navigation component which apart from traditional, text links, might embed icons, dropdowns, avatars or search forms.

Algorithm

1. Create the navbar.js file .
2. Create the new function component and name it as ResponsiveAppBar in navbar.
3. Inside the function component return the html element.
4. And export the function component ResponsiveAppBar.
5. Now import the function component from navbar file in App.js file.
6. Inside the root.render use the component.
7. Now the output will display in the react app.

Program

```
import * as React from 'react';
import AppBar from '@mui/material/AppBar';
import Box from '@mui/material/Box';
 import Toolbar from '@mui/material/Toolbar';
import IconButton from '@mui/material/IconButton';
import Typography from '@mui/material/Typography';
import Menu from '@mui/material/Menu';
 import MenuIcon from '@mui/icons-material/Menu';
import Container from '@mui/material/Container';
```

```
import Avatar from '@mui/material/Avatar';
import Button from '@mui/material/Button';
import Tooltip from '@mui/material/Tooltip';
import MenuItem from '@mui/material/MenuItem';
import AdbIcon from '@mui/icons-material/Adb';

const pages = ['Products', 'Pricing', 'Blog'];
const settings = ['Profile', 'Account', 'Dashboard', 'Logout'];

function ResponsiveAppBar() {
 const [anchorElNav, setAnchorElNav] = React.useState(null);
const [anchorElUser, setAnchorElUser] = React.useState(null);

 const handleOpenNavMenu = (event) => {
setAnchorElNav(event.currentTarget);
 };
 const handleOpenUserMenu = (event) => {
setAnchorElUser(event.currentTarget);
 };

 const handleCloseNavMenu = () => {
setAnchorElNav(null);
 };
 const handleCloseUserMenu = () => {
   setAnchorElUser(null);
          };
```

```
        return (
        <AppBar position="static">
<Container maxWidth="xl">
        <Toolbar disableGutters>
        <AdbIcon sx={{ display: { xs: 'none', md: 'flex' }, mr: 1 }} />
<Typography
   variant="h6" noWrap
   component="a"
   href="/" sx={{ mr:
        2,
        display: { xs: 'none', md: 'flex' },
        fontFamily: 'monospace',
        fontWeight: 700, letterSpacing:
        '.3rem', color:
        'inherit', textDecoration:
        'none',
        }}
        >
        SKCET
</Typography>

<Box sx={{ flexGrow: 1, display: { xs: 'flex', md: 'none' } }}>
<IconButton
size="large"
 aria-label="account of current user"
```

```
aria-controls="menu-appbar" ariahaspopup="true"
onClick={handleOpenNavMenu} color="inherit"
>
<MenuIcon />
</IconButton>
<Menu id="menuappbar"
anchorEl={anchorElNav}
anchorOrigin={{ vertical:
'bottom', horizontal:
'left',
 }}
 keepMounted
transformOrigin={{ vertical:
'top', horizontal:
'left',
 }}
 open={Boolean(anchorElNav)}
onClose={handleCloseNavMenu} sx={{ display: { xs: 'block',
md: 'none' },
 }}
>
 {pages.map((page) => (
<MenuItem key={page} onClick={handleCloseNavMenu}>
<Typography textAlign="center">{page}</Typography>
</MenuItem>
 ))}
```

```
</Menu>
</Box>
<AdbIcon sx={{ display: { xs: 'flex', md: 'none' }, mr: 1 }} />
<Typography
variant="h5" noWrap
component="a"
href="" sx={{ mr:
2,
 display: { xs: 'flex', md: 'none' },
flexGrow: 1, fontFamily: 'monospace',
fontWeight: 700, letterSpacing:
'.3rem',
color: 'inherit',
textDecoration: 'none',
 }}
>
 SKCET
</Typography>
<Box sx={{ flexGrow: 1, display: { xs: 'none', md: 'flex' } }}>
 {pages.map((page) => (
<Button
key={page}
 onClick={handleCloseNavMenu}
sx={{ my: 2, color: 'white', display: 'block' }}
>
 {page}
```

```
</Button>
 ))}
</Box>

<Box sx={{ flexGrow: 0 }}>
<Tooltip title="Open settings">
<IconButton onClick={handleOpenUserMenu} sx={{ p: 0 }}>
<Avatar alt="Remy Sharp" src="/static/images/avatar/2.jpg" />
</IconButton>
</Tooltip>
<Menu sx={{ mt:
        '45px' }} id="menuappbar"
   anchorEl={anchorElUser}
        anchorOrigin={{ vertical:
        'top', horizontal: 'right',
        }}
        keepMounted
   transformOrigin={{ vertical:
        'top', horizontal:
        'right',
        }}
    open={Boolean(anchorElUser)}
   onClose={handleCloseUserMenu}
        >
    {settings.map((setting) => (
<MenuItem key={setting} onClick={handleCloseUserMenu}>
```

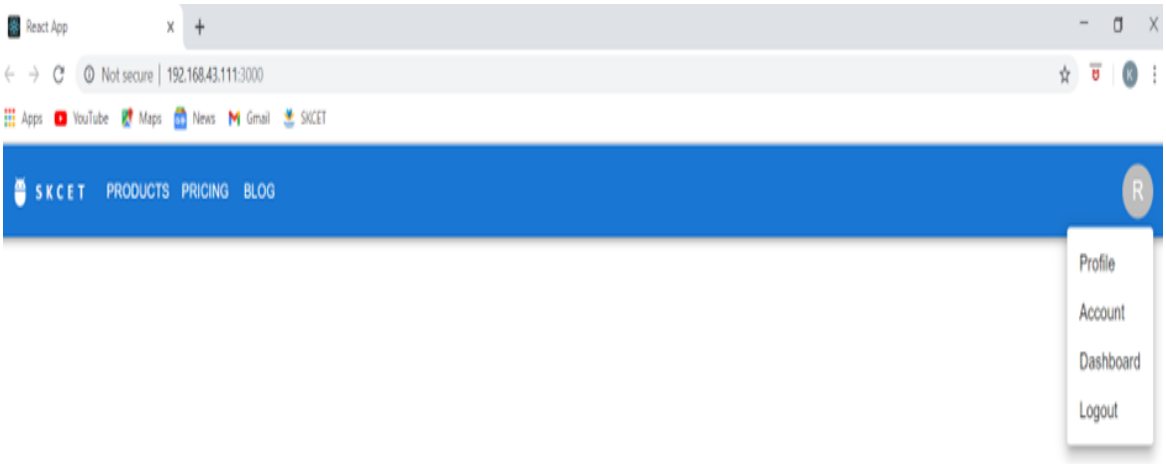
```
<Typography textAlign="center">{setting}</Typography>
</MenuItem>
        ))}
</Menu>
</Box>
        </Toolbar>
</Container>
</AppBar>
        );
}
export default ResponsiveAppBar;
```

App.js

```
import React from 'react'
import Hello from'./over';
import Card from './button';
import UpdatedComponent from './update';
import './style.css';

import ResponsiveAppBar from './navbar';
function App(){
  return(
<ResponsiveAppBar/>
  );
}
```

export default App

Output



Result

Thus the program is Executed Successfully.

7.Implementation of React component to handle HTTP requests

Objective

To implement a React component to handle HTTP requests.

System and software tools required

- Visual Studio Code

Description

Most useful React applications involve interacting with a server to load and persist data. To do this on the web, we use HTTP requests with the browser's built-in fetch API (or you may use some other open source library that's built on top of this API).

Algorithm

1. Create the fetch.js file .
2. Create the new function component and name it as FectchAPI in fetch.
3. Inside the function component return the html element.
4. And export the function component FetchAPI.
5. Now import the function component from FetchAPI file in fetch.js file.
6. Inside the root.render use the component.
7. Now the output will display in the react app.

Program

fetch.js:

```
import { useEffect, useState } from "react";
export default function FetchAPI(){
        const [user, setUser] = useState([]);
        useEffect(() => {
  fetch('https://jsonplaceholder.typicode.com/users')
        .then(res => res.json())
        .then(res => setUser(res))
})
```

```
        console.log(user)
        return(
        <div className="main">
        {user.map(u => (
<div>{u.name}, {u.id}, {u.email}</div>
        ))}
        </div>
        )
}
```

App.js:

```
import './App.css';
import FetchAPI from './fetch';
 function App() {
  return (<>
        <FetchAPI />
</>);
}
 export default App;
```

Output

```
Leanne Graham, 1, Sincere@april.biz
Ervin Howell, 2, Shanna@melissa.tv
Clementine Bauch, 3, Nathan@yesenia.net
Patricia Lebsack, 4, Julianne.OConner@kory.org
Chelsey Dietrich, 5, Lucio_Hettinger@annie.ca
Mrs. Dennis Schulist, 6, Karley_Dach@jasper.info
Kurtis Weissnat, 7, Telly.Hoeger@billy.biz
Nicholas Runolfsdottir V, 8, Sherwood@rosamond.me
Glenna Reichert, 9, Chaim_McDermott@dana.io
Clementina DuBuque, 10, Rey.Padberg@karina.biz
```

Result

Thus the program is executed successfully.

8. Implementation of Drop-down Component using React

Objective:
To implement a Drop-down component using React.

Algorithm

1.Create a react-app using npx create-react-app appname.

2.Give cd appname and npm start to run the app.

3.Create a Dropdown.js and Dropdown.css in the src folder.

4.Create a dropdown input UI in the Dropdown.js file.

5.Apply the required style in Dropdown.css file.

6.Create a dropdown menu UI in the Dropdown.js file.

7.Open/Close dropdown menu handler in Dropdown.js file.

8.Handle select/deselect dropdown item in Dropdown.js.

9.Next create a Multi Drop-down select.

10.Execute a Callback function with the selected values so that the App can do other stuff.

11.Import the Dropdown.js file in the App.js file and include the file inside tag.

Coding:

App.js

```
import Dropdown from "./main";
export default function App() {
  return (
<div className="App">
<Dropdown placeHolder="Select..." />
</div>
  );
}
```

Dropdown.css

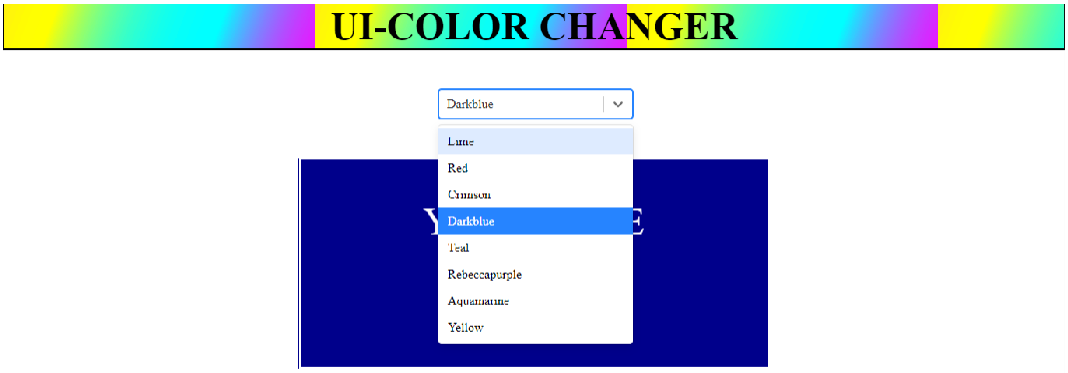
```
.dropdown-container {
  text-align: left;
  border: 2px solid rgb(57, 19, 196);
  position: relative;
  border-radius: 5px;
}
```

```
.dropdown-input {
  padding: 20px;
  display: flex;
  align-items: center;
  justify-content: space-between;
  user-select: none;
}
```

Dropdown.js

```
import * as React from 'react';
const App = () => {
 return (
<div>
<select>
<option value="fruit">Fruit</option
<option value="vegetable">Vegetable</option>
<option value="meat">Meat</option>
<option value="Groceries">Groceries</option>
<option value="Snacks">Snacks</option>
<option value="Footwears">Footwears</option>
</select>
</div>
 );
};
export default App;
```

Output:



Result:

Hance a Drop down component using React is successfully implemented.

9. Implementation of Routing in React

Objective

To implement Routing in React using react-router-dom.

Algorithm

1. 1.Create a react-app using npx create-react-app appname.
2. 2.Give cd appname and npm start to run the app everytime.
3. 3.Install react-router-dom using npm command.
4. 4.In App.js import BrowserRouter,Routes,Route,Router from react-router-dom.
5. 5.Inside function App() return statement open the BrowserRouter tag.
6. 6.Inside the BrowserRouter tag give their specified router inside the <router> tag.
7. 7.Inside the <router> tag specify the Route path and element using <Route Path> tag.
8. 8.Run the app using npm start.

App.js

```
import './mainpage.css';
import './returnpage.css';
import Mainpage from './mainpage'
import Returnpage from './returnpage'

import { BrowserRouter as Router, Route, Routes } from 'react-router-dom';
function App() {
  return(
    <Router>
      <div className='App'>
        <Routes>
          <Route exact path='/' element={<Mainpage />}></Route>
          <Route exact path='/mainpage' element={<Mainpage />}></Route>
          <Route exact path='/returnpage' element={<Returnpage />}></Route>
        </Routes>
      </div>
    </Router>
  )
}
export default App;
```

App.test.js

```
import React from 'react';
import ReactDOM from 'react-dom';
import App from './App';

it('renders without crashing', () => {
  const div = document.createElement('div');
  ReactDOM.render(<App />, div);
});
```

Index.css

```
html, body, #root, #root>div {
  height: 100%;
}

body {
  margin: 0;
  padding: 0;
  font-family: sans-serif;
  height: 100%;
}
```

Index.js

```
import React from 'react';
import ReactDOM from 'react-dom';
import App from './App';
import './index.css';

ReactDOM.render(
  <App />,
  document.getElementById('root')
);
```

Mainpage.js

```
//import React from 'react';
import "./mainpage.css"
import "./returnpage.css"
import { Link } from "react-router-dom";


const Mainpage = () => {
   return (<>
      <div>
      <h1>This is Mainpage</h1>
      <Link to="/returnpage" className="v2_21">Next Page</Link>

       </div>
       </>
     );
    };
export default Mainpage;
```

mainpage.css

```
.v2_21 {
  top: 100px;
  left: 50vh;
  position: absolute;
  font-family: Inter;
  font-weight: Bold;
  font-size: 30px;
  text-align: center;
}
```

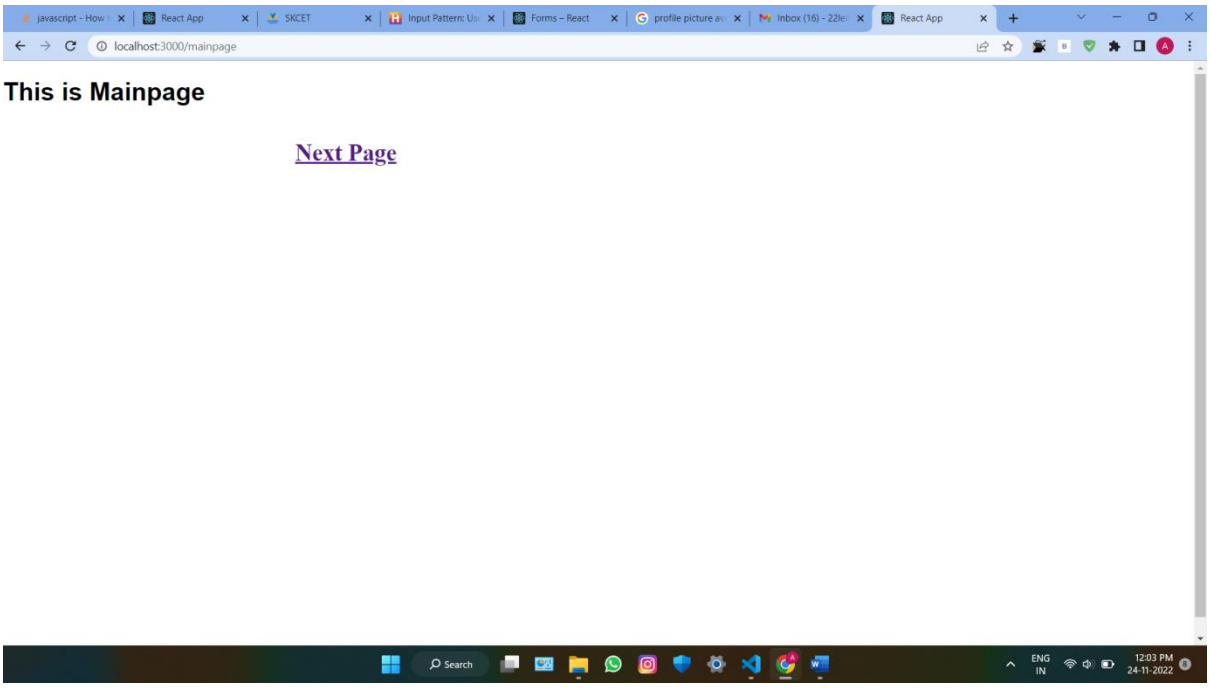
Returnpage.js

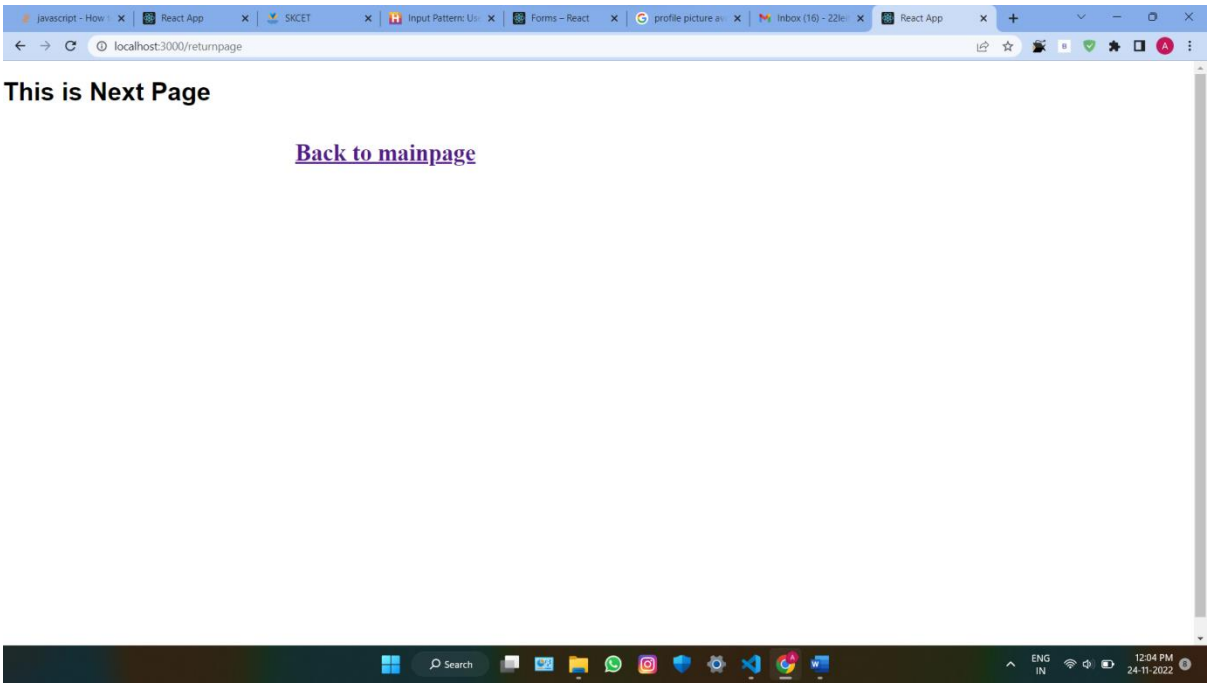
```
import "./mainpage.css"
import "./returnpage.css"
import { Link } from "react-router-dom";

const Returnpage = () => {
   return (<>
      <div>
        <h1>This is Next Page</h1>
       <Link to='/mainpage' class="v2_21">Back to mainpage</Link>

       </div>
       </>
     );
    };
export default Returnpage;
```

returnpage.css

```
.v2_21 {
  top: 100px;
  left: 50vh;
  position: absolute;
  font-family: Inter;
  font-weight: Bold;
  font-size: 30px;
  text-align: center;
}
```

Output:





Result:

Hence Routing is successfully implemented in React.

10. Implementation of form Validation in React

Objective:

To implement a form Validation in React.

Algorithm:

1.Create a react-app using npx create-react-app appname.
2.Give cd appname and npm start to run the app.
3.First, we need to update the App.jsx and App.css files.
4. The App Component will render a headline and the LoginForm Component it will create in a moment.
5. Create the login form component.
6. The login form utilizes the usestate hook to Store the state for the form.
7.The form State is defined, we have the onupdate field function, which is passed to a input field as an onchange handler.
8. Further, the onSubmitForm method will be executed when the form is Submitted.
9. Finally, the LoginForm Component render a form that comprises three fields- email, Password and Confirm password.
10.We need to install the clsx helper. Run the Command below in the terminal.

$ npm install clsx

11.Place all validators in the validators.js file.
12.We are going to put all the Validation logic in a custom hook called useloginForm Validator.
13. We have the ValidateForm function. It accepts an object with four properties.

form, field, errors, force TouchErrors.

14. Lastly, the SetErrors method is called with the Validation results and object with isValid flag and errors are returned.
15. After the validate Form function, we have the onBlur Field function. It checks if the field that was blurred is already dirty.
16.We also had to update both onUpdateField and onSubmitForm functions

Coding:

1. Form validation in react Form.js

```
import React from 'react'; import './form.css';
class RegisterForm extends React.Component { constructor() {
super(); this.state = {
fields: {},
errors: {}
}
this.handleChange = this.handleChange.bind(this); this.submituserRegistrationForm =
this.submituserRegistrationForm.bind(this);

};
handleChange(e) {
let fields = this.state.fields; fields[e.target.name] = e.target.value; this.setState({
fields
});
}
submituserRegistrationForm(e) { e.preventDefault();
if (this.validateForm()) { let fields = {}; fields["username"] = "";
fields["mobileno"] = "";
fields["password"] = ""; this.setState({fields:fields}); alert("Form submitted");
}

}

validateForm() {

let fields = this.state.fields; let errors = {};
let formIsValid = true; if (!fields["username"]) {
formIsValid = false;
errors["username"] = "*Please enter your username.";
}
if (typeof fields["username"] !== "undefined") {

if (!fields["username"].match(/^[a-zA-Z ]*$/)) { formIsValid = false;
errors["username"] = "*Please enter alphabet characters only.";
}
}
if (!fields["emailid"]) { formIsValid = false;
errors["emailid"] = "*Please enter your email-ID.";
}
if (!fields["mobileno"]) { formIsValid = false;
errors["mobileno"] = "*Please enter your mobile no.";
}
```

```
if (typeof fields["mobileno"] !== "undefined") { if (!fields["mobileno"].match(/^[0-9]{10}$/)) {
formIsValid = false;
errors["mobileno"] = "*Please enter valid mobile no.";
}
}

if (!fields["password"]) { formIsValid = false;
errors["password"] = "*Please enter your password.";
}

if (typeof fields["password"] !== "undefined") {
if (!fields["password"].match("^(?=.*[a-z])(?=.*[A-Z])(?=.*[0-
9])(?=.*[!@#\$%\^&\*])(?=.{8,})") {
formIsValid = false;
errors["password"] = "*Please enter secure and strong password.";
}
}

this.setState({ errors: errors
});
return formIsValid;
}
render() { return (
<div id="main-registration-container">
<div id="register">
<h3>Registration page</h3>

<form method="post" name="userRegistrationForm" onSubmit=
{this.submituserRegistrationForm} >
<label>Name</label>
<input type="text" name="username" value={this.state.fields.username}
onChange={this.handleChange} />
<div className="errorMsg">{this.state.errors.username}</div>
<label>Mobile No:</label>
<input type="text" name="mobileno" value={this.state.fields.mobileno}
onChange={this.handleChange}       />
<div className="errorMsg">{this.state.errors.mobileno}</div>
<label>Password</label>
<input type="password" name="password" value={this.state.fields.password}
onChange={this.handleChange} />
<div className="errorMsg">{this.state.errors.password}</div>
<input type="submit" className="button" value="Register"/>
</form>
</div>
</div>
```

```
);
}
}
export default RegisterForm;
```

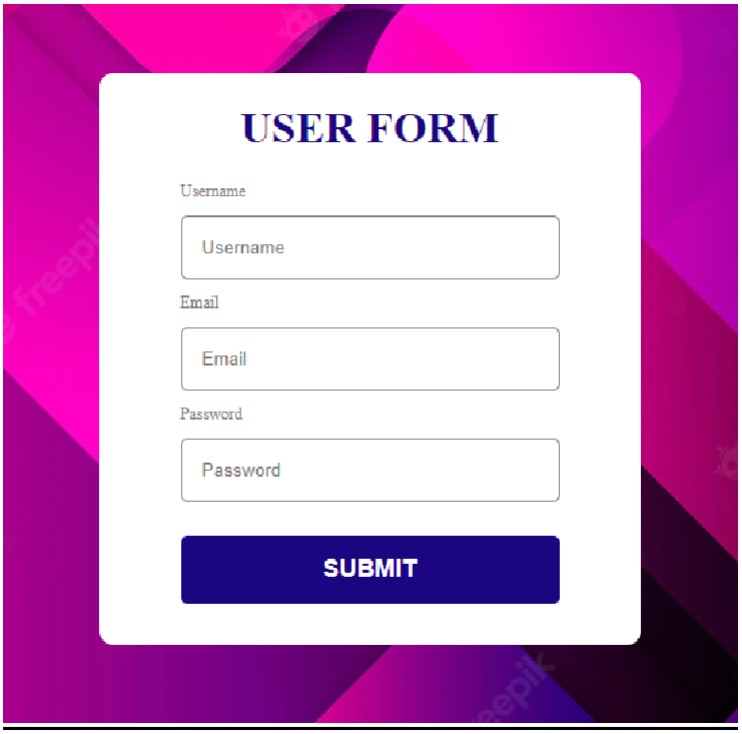
App.js

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

CSS

```
#register, #login {
width: 300px;

border: 1px solid #d6d7da; padding: 0px 15px 15px 15px; border-radius: 5px;
font-family: arial; line-height: 16px; color: #333333; font-size: 14px;
background: #ffffff; margin: 100px auto;
}

form label, form input { display: block;
margin-bottom: 10px; width: 90%
}

form input { padding: 10px;
border: solid 1px #BDC7D8;

}

.button {
background-color: #00BFFF; border-color: #3ac162; font-weight: bold; padding: 12px 15px;
color: #ffffff;
}

.errorMsg { color: #cc0000;
margin-bottom: 12px;
}

.sucessMsg { color: #6B8E23;
margin-bottom: 10px;
```

OUTPUT



Result:

Hence form validation is implemented in React.