1. **Create screens/pages using ReactJS required for the REST API created at Assignment 7. Use react-router-dom for easy navigation between pages. Use AXIOS to connect with the REST API. Do required proxy settings to avoid CORS error.**

**Ans-**

**App.jsx**

import { createBrowserRouter, RouterProvider } from "react-router-dom";

import Layout from "./layout/Layout";

import Home from "./pages/Home";

import StudentDetails from "./pages/StudentDetails";

import AddStudentDetails from "./pages/AddStudentDetails";

import UpdateStudentDetails from "./pages/UpdateStudentDetails";

import DeleteStudentDetails from "./pages/DeleteStudentDetails";

import UpdateStudent from "./pages/UpdateStudent";

import SearchStudentDetails from "./pages/SearchStudentDetails";

function App() {

const router = createBrowserRouter([

{

path: "/",

element: <Layout />,

children: [

{ path: "/", element: <Home /> },

{ path: "/studentdetails", element: <StudentDetails /> },

{ path: "/addstudentdetails", element: <AddStudentDetails /> },

{

path: "/updatestudentdetails",

element: <UpdateStudentDetails />,

},

{

path: "/updatestudent/:id",

element: <UpdateStudent />,

},

{

path: "/deletestudentdetails",

element: <DeleteStudentDetails />,

},

{

path: "/searchstudentdetails",

element: <SearchStudentDetails />,

},

],

},

]);

return <RouterProvider router={router} />;

}

export default App;

**Layout.jsx**

import { Outlet, useLocation } from "react-router-dom";

import ResponsiveAppBar from "../components/Navbar";

import Footer from "../components/Footer";

const Layout = () => {

const location = useLocation();

return (

<>

<ResponsiveAppBar />

<Outlet />

<Footer

fixed={

location.pathname === "/" ||

location.pathname === "/updatestudentdetails" ||

location.pathname === "/studentdetails" ||

location.pathname === "/deletestudentdetails" ||

location.pathname === "/searchstudentdetails"

? true

: false

}

/>

</>

);

};

export default Layout;

**Navbar.jsx**

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 Button from "@mui/material/Button";

import MenuItem from "@mui/material/MenuItem";

import AssignmentIcon from "@mui/icons-material/Assignment";

import { useNavigate } from "react-router-dom";

const pages = [

"Student Details",

"Add Student Details",

"Update Student Details",

"Delete Student Details",

"Search Student Details",

];

function ResponsiveAppBar() {

const [anchorElNav, setAnchorElNav] = React.useState(null);

const navigate = useNavigate();

const handleOpenNavMenu = (event) => {

setAnchorElNav(event.currentTarget);

};

const handleNavigate = (event) => {

setAnchorElNav(null);

const pageToNavigate = event.target.innerText.toLowerCase();

navigate(`/${pageToNavigate.split(" ").join("")}`);

};

return (

<AppBar position="static">

<Container maxWidth="xl">

<Toolbar disableGutters>

<AssignmentIcon 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",

}}

>

Assignment-9

</Typography>

<Box sx={{ flexGrow: 1, display: { xs: "flex", md: "none" } }}>

<IconButton

size="large"

aria-label="account of current user"

aria-controls="menu-appbar"

aria-haspopup="true"

onClick={handleOpenNavMenu}

color="inherit"

>

<MenuIcon />

</IconButton>

<Menu

id="menu-appbar"

anchorEl={anchorElNav}

anchorOrigin={{

vertical: "bottom",

horizontal: "left",

}}

keepMounted

transformOrigin={{

vertical: "top",

horizontal: "left",

}}

open={Boolean(anchorElNav)}

onClose={handleNavigate}

sx={{

display: { xs: "block", md: "none" },

}}

>

{pages.map((page, index) => (

<MenuItem key={index} onClick={handleNavigate}>

<Typography textAlign="center">{page}</Typography>

</MenuItem>

))}

</Menu>

</Box>

<AssignmentIcon 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",

}}

>

Assignment-9

</Typography>

<Box sx={{ flexGrow: 1, display: { xs: "none", md: "flex" } }}>

{pages.map((page) => (

<Button

key={page}

onClick={handleNavigate}

sx={{ my: 2, color: "white", display: "block" }}

>

{page}

</Button>

))}

</Box>

{/\* <Box sx={{ flexGrow: 0 }}>

<Avatar alt="user" src="/avatars/user.png" />

</Box> \*/}

</Toolbar>

</Container>

</AppBar>

);

}

export default ResponsiveAppBar;

**Footer.jsx**

/\* eslint-disable react/prop-types \*/

import { AppBar, Toolbar, Typography, Container } from "@mui/material";

import moment from "moment";

function Footer({ fixed = false }) {

const footerStyle = {

padding: "20px",

marginTop: "auto",

backgroundColor: "black",

bottom: 0,

position: fixed ? "fixed" : "relative",

height: "6rem",

};

const year = moment().format("YYYY");

return (

<AppBar

position="static"

style={footerStyle}

>

<Toolbar>

<Container maxWidth="md">

<Typography

variant="body1"

color="inherit"

sx={{

display: "flex",

justifyContent: "center",

alignItems: "center",

}}

>

&copy; {year} 21BCSE30 | ANIKET PATRA &nbsp;&nbsp;All rights

reserved.

</Typography>

</Container>

</Toolbar>

</AppBar>

);

}

export default Footer;

**studentService.js**

import axios from "axios";

class StudentService {

constructor() {

this.api = import.meta.env.VITE\_URL;

}

async getStudentDetails() {

try {

const res = await axios.get(`${this.api}/allstudentinfos/`);

return { status: true, data: res.data };

} catch (error) {

return { status: false, data: error };

}

}

async addStudentDetails(studentDetails) {

try {

const res = await axios.post(

`${this.api}/allstudentinfos/`,

studentDetails

);

return { status: true, data: res.data };

} catch (error) {

return { status: false, data: error };

}

}

async getStudentDetailsByRoll(rollNo) {

rollNo = parseInt(rollNo.value);

try {

const res = await axios.get(

`http://localhost:5000/allstudentinfos/${rollNo}`

);

if (

res.data.studentInfo.length > 0 &&

res.data.studentAcademicInfo.length > 0

) {

return { status: true, data: res.data };

} else {

return { status: false, data: "Student Data Not Availaible!!" };

}

} catch (error) {

return { status: false, data: error };

}

}

async updateStudentDetailsByRoll(StudentDetails, rollNo) {

rollNo = parseInt(rollNo);

try {

const res = await axios.put(

`http://localhost:5000/allstudentinfos/${rollNo}`,

StudentDetails

);

return { status: true, data: res.data };

} catch (error) {

return { status: false, data: error };

}

}

async deleteStudentDetailsByRoll(rollNo) {

rollNo = parseInt(rollNo.value);

try {

const res = await axios.delete(

`http://localhost:5000/allstudentinfos/${rollNo}`

);

return { status: true, data: res.data };

} catch (error) {

return { status: false, data: error };

}

}

}

const studentService = new StudentService();

export default studentService;

**AddStudentDetails.jsx**

import MyForm from "../components/MyForm";

const AddStudentDetails = () => {

return <MyForm />;

};

export default AddStudentDetails;

**MyForm.jsx**

/\* eslint-disable react/prop-types \*/

import { useEffect, useState } from "react";

import {

TextField,

Button,

Grid,

Typography,

Paper,

Snackbar,

Alert,

} from "@mui/material";

import studentService from "../service/StudentService";

function MyForm({ type = "Add", data = {} }) {

const initialFormData = {

rollno: data?.rollno,

name: data?.name,

email: data?.email,

mobile: data?.mobile,

city: data?.city,

state: data?.state,

pin: data?.pin,

program: data?.program,

branch: data?.branch,

cgpa: data?.cgpa,

};

const DummyData = {

rollno: "",

name: "",

email: "",

mobile: "",

city: "",

state: "",

pin: "",

program: "",

branch: "",

cgpa: "",

};

const [formData, setFormData] = useState(initialFormData);

const [Data, setData] = useState(data);

useEffect(() => {

setData(data);

}, [Data]);

const [open, setOpen] = useState(false);

const [success, setSuccess] = useState(false);

const handleClose = (event, reason) => {

if (reason === "clickaway") {

return;

}

setOpen(false);

};

const studentdetails = {

name: formData.name,

rollno: parseInt(formData.rollno),

mobile: formData.mobile,

email: formData.email,

address: {

city: formData.city,

state: formData.state,

pin: formData.pin,

},

studentDetails: [

{

rollno: parseInt(formData.rollno),

program: formData.program,

branch: formData.branch,

cgpa: formData.cgpa,

},

],

};

const handleSubmit = async (e) => {

e.preventDefault();

if (type.toLowerCase() === "update") {

const updatedData = {

studentDetails: [

{

rollno: formData.rollno,

name: formData.name,

email: formData.email,

mobile: formData.mobile,

address: {

city: formData.city,

state: formData.state,

pin: formData.pin,

},

},

],

studentAcademicDetails: [

{

rollno: formData.rollno,

program: formData.program,

branch: formData.branch,

cgpa: formData.cgpa,

},

],

};

await studentService.updateStudentDetailsByRoll(

updatedData,

formData.rollno

);

setOpen(true);

setSuccess(true);

return;

}

const res = await studentService.addStudentDetails(studentdetails);

if (res.status) {

setFormData(DummyData);

setOpen(true);

setSuccess(true);

} else {

setSuccess(false);

}

};

const handleChange = (e) => {

const { name, value } = e.target;

setFormData((prevData) => ({

...prevData,

[name]: value,

}));

};

return (

<Paper elevation={3} style={{ padding: 20 }}>

<Snackbar

open={open}

autoHideDuration={6000}

onClose={handleClose}

anchorOrigin={{ vertical: "top", horizontal: "right" }}

>

<Alert

onClose={handleClose}

severity={success ? "success" : "error"}

variant="filled"

sx={{ width: "100%" }}

>

{success

? type.toLowerCase() === "update"

? "Successfully Updated the Student Details!!"

: "Successfully Added the Student Setails!!"

: type.toLowerCase() === "update"

? "Error in Updating Student Details!!"

: "Error in Adding the Student Details!!"}

</Alert>

</Snackbar>

<Typography variant="h6" gutterBottom>

{type} Student Details

</Typography>

<form onSubmit={handleSubmit}>

<Grid container spacing={2}>

<Grid item xs={12} sm={6}>

<TextField

name="rollno"

label="Roll No"

variant="outlined"

fullWidth

value={formData.rollno}

onChange={handleChange}

/>

</Grid>

<Grid item xs={12} sm={6}>

<TextField

name="name"

label="Name"

variant="outlined"

fullWidth

value={formData.name}

onChange={handleChange}

/>

</Grid>

<Grid item xs={12} sm={6}>

<TextField

name="email"

label="Email"

variant="outlined"

fullWidth

value={formData.email}

onChange={handleChange}

/>

</Grid>

<Grid item xs={12} sm={6}>

<TextField

name="mobile"

label="Mobile No"

variant="outlined"

fullWidth

value={formData.mobile}

onChange={handleChange}

/>

</Grid>

<Grid item xs={12} sm={6}>

<TextField

name="program"

label="Program"

variant="outlined"

fullWidth

value={formData.program}

onChange={handleChange}

/>

</Grid>

<Grid item xs={12} sm={6}>

<TextField

name="branch"

label="Branch"

variant="outlined"

fullWidth

value={formData.branch}

onChange={handleChange}

/>

</Grid>

<Grid item xs={12} sm={6}>

<TextField

name="cgpa"

label="CGPA"

variant="outlined"

fullWidth

value={formData.cgpa}

onChange={handleChange}

/>

</Grid>

<Grid item xs={12} sm={6}>

<TextField

name="city"

label="City"

variant="outlined"

fullWidth

value={formData.city}

onChange={handleChange}

/>

</Grid>

<Grid item xs={12} sm={6}>

<TextField

name="state"

label="State"

variant="outlined"

fullWidth

value={formData.state}

onChange={handleChange}

/>

</Grid>

<Grid item xs={12} sm={6}>

<TextField

name="pin"

label="Pin Code"

variant="outlined"

fullWidth

value={formData.pin}

onChange={handleChange}

/>

</Grid>

</Grid>

<Button

type="submit"

variant="contained"

color="primary"

style={{

marginTop: 20,

}}

>

Submit

</Button>

</form>

</Paper>

);

}

export default MyForm;

**UpdateStudentDetails**

import {

Stack,

Paper,

TextField,

Button,

Alert,

Snackbar,

} from "@mui/material";

import studentService from "../service/StudentService";

import { useState } from "react";

import { useNavigate } from "react-router-dom";

const UpdateStudentDetails = () => {

const [rollNo, setRollNo] = useState(null);

const [open, setOpen] = useState(false);

const [success, setSuccess] = useState(false);

const navigate = useNavigate();

const handleUpdate = async () => {

if (rollNo?.value !== undefined) {

const res = await studentService.getStudentDetailsByRoll(rollNo);

if (res.status) {

if (res.status) {

setOpen(true);

setSuccess(true);

setTimeout(() => {

navigate(`/updatestudent/${parseInt(rollNo.value)}`);

}, 2000);

}

} else {

setOpen(true);

setSuccess(false);

}

} else {

setOpen(true);

setSuccess(false);

}

};

const getRollNo = (e) => {

const { value } = e.target;

setRollNo((prevData) => ({

...prevData,

value,

}));

};

const handleClose = (event, reason) => {

if (reason === "clickaway") {

return;

}

setOpen(false);

};

return (

<Paper

elevation={2}

sx={{

display: "flex",

justifyContent: "center",

alignItems: "center",

marginTop: "5rem",

width: {

xs: "100%",

sm: "90%",

md: "60%",

},

marginLeft: {

xs: "0%",

sm: "5%",

md: "20%",

},

}}

>

<Snackbar

open={open}

autoHideDuration={2000}

onClose={handleClose}

anchorOrigin={{ vertical: "top", horizontal: "right" }}

>

<Alert

onClose={handleClose}

severity={success ? "success" : "error"}

variant="filled"

sx={{ width: "100%" }}

>

{success

? "Navigating to Update Student Page!!"

: "Student details not Availaible!!"}

</Alert>

</Snackbar>

<Stack spacing={2} marginTop={"2rem"} marginBottom={"2rem"}>

<h1>Enter the Roll No of Student to Update Details</h1>

<TextField

label="Roll No"

variant="outlined"

id="outlined-basic"

onChange={getRollNo}

/>

<Button variant="contained" onClick={handleUpdate}>

Update

</Button>

</Stack>

</Paper>

);

};

export default UpdateStudentDetails;

**StudentDetails.jsx**

/\* eslint-disable react/prop-types \*/

import { DataGrid } from "@mui/x-data-grid";

import Box from "@mui/material/Box";

import { useEffect, useState } from "react";

import studentService from "../service/StudentService";

const columns = [

{ field: "rollno", headerName: "Roll No", type: "number", width: 90 },

{

field: "name",

headerName: "Student Name",

width: 200,

},

{

field: "email",

headerName: "Email",

width: 250,

},

{

field: "mobile",

headerName: "Mobile No.",

width: 200,

},

{

field: "address",

headerName: "Address",

width: 300,

},

{

field: "program",

headerName: "Program",

width: 150,

},

{

field: "branch",

headerName: "Branch",

width: 310,

},

{

field: "cgpa",

headerName: "CGPA",

width: 100,

},

];

const StudentDetails = () => {

const [rows, setRows] = useState([]);

const fetchData = async () => {

const res = await studentService.getStudentDetails();

if (res.status) {

const students = res.data;

setRows(

students.map((student) => {

return {

rollno: student.rollno,

name: student.name,

email: student.email,

mobile: student.mobile,

address: `${student.address.state}, ${student.address.city} PIN - ${student.address.pin}`,

program: student.studentDetails[0].program,

branch: student.studentDetails[0].branch,

cgpa: student.studentDetails[0].cgpa,

};

})

);

}

};

useEffect(() => {

fetchData();

}, [rows]);

return <StudentDataGrid rows={rows} />;

};

function StudentDataGrid({ rows = [] }) {

const generateRowId = () => {

let id = 0;

return () => {

id += 1;

return id;

};

};

const generateId = generateRowId();

const rowsWithIds = rows.map((row) => ({ ...row, id: generateId() }));

return (

<Box sx={{ height: 400, width: "100%" }}>

<DataGrid

rows={rowsWithIds}

columns={columns}

initialState={{

pagination: {

paginationModel: {

pageSize: 5,

},

},

}}

pageSizeOptions={[5]}

disableRowSelectionOnClick

/>

</Box>

);

}

export default StudentDetails;

**DeleteStudentDetails.jsx**

import {

Stack,

Paper,

TextField,

Button,

Alert,

Snackbar,

} from "@mui/material";

import studentService from "../service/StudentService";

import { useState } from "react";

const DeleteStudentDetails = () => {

const [rollNo, setRollNo] = useState(null);

const [open, setOpen] = useState(false);

const [success, setSuccess] = useState(false);

const handleDelete = async () => {

if (rollNo?.value !== undefined) {

const res = await studentService.getStudentDetailsByRoll(rollNo);

if (res.status) {

const res = await studentService.deleteStudentDetailsByRoll(rollNo);

if (res.status) {

setOpen(true);

setSuccess(true);

}

} else {

setOpen(true);

setSuccess(false);

}

} else {

setOpen(true);

setSuccess(false);

}

};

const getRollNo = (e) => {

const { value } = e.target;

setRollNo((prevData) => ({

...prevData,

value,

}));

};

const handleClose = (event, reason) => {

if (reason === "clickaway") {

return;

}

setOpen(false);

};

return (

<Paper

elevation={2}

sx={{

display: "flex",

justifyContent: "center",

alignItems: "center",

marginTop: "5rem",

width: {

xs: "100%",

sm: "90%",

md: "60%",

},

marginLeft: {

xs: "0%",

sm: "5%",

md: "20%",

},

}}

>

<Snackbar

open={open}

autoHideDuration={6000}

onClose={handleClose}

anchorOrigin={{ vertical: "top", horizontal: "right" }}

>

<Alert

onClose={handleClose}

severity={success ? "success" : "error"}

variant="filled"

sx={{ width: "100%" }}

>

{success

? "Successfully Deleted the Student Setails!!"

: "Student details not Availaible!!"}

</Alert>

</Snackbar>

<Stack spacing={2} marginTop={"2rem"} marginBottom={"2rem"}>

<h1>Enter the Roll No of Student to Delete Details</h1>

<TextField

label="Roll No"

variant="outlined"

id="outlined-basic"

onChange={getRollNo}

/>

<Button variant="contained" color="error" onClick={handleDelete}>

Delete

</Button>

</Stack>

</Paper>

);

};

export default DeleteStudentDetails;

**SearchStudentDetails.jsx**

/\* eslint-disable react/prop-types \*/

import { useState } from "react";

import Table from "@mui/material/Table";

import TableBody from "@mui/material/TableBody";

import TableCell from "@mui/material/TableCell";

import TableContainer from "@mui/material/TableContainer";

import TableHead from "@mui/material/TableHead";

import TableRow from "@mui/material/TableRow";

import Paper from "@mui/material/Paper";

import { Button, Stack, TextField, Alert, Snackbar } from "@mui/material";

import SearchIcon from "@mui/icons-material/Search";

import studentService from "../service/StudentService";

function createData(

rollno,

name,

email,

mobile,

address,

program,

branch,

cgpa

) {

return { rollno, name, email, mobile, address, program, branch, cgpa };

}

function BasicTable({ rows = [] }) {

return (

<TableContainer component={Paper}>

<Table sx={{ minWidth: 650 }} aria-label="simple table">

<TableHead>

<TableRow>

<TableCell>Roll No</TableCell>

<TableCell align="right">Name</TableCell>

<TableCell align="right">Email</TableCell>

<TableCell align="right">Mobile</TableCell>

<TableCell align="right">Address</TableCell>

<TableCell align="right">Program</TableCell>

<TableCell align="right">Branch</TableCell>

<TableCell align="right">CGPA</TableCell>

</TableRow>

</TableHead>

<TableBody>

{rows.map((row) => (

<TableRow

key={row.rollno}

sx={{ "&:last-child td, &:last-child th": { border: 0 } }}

>

<TableCell component="th" scope="row">

{row.rollno}

</TableCell>

<TableCell align="right">{row.name}</TableCell>

<TableCell align="right">{row.email}</TableCell>

<TableCell align="right">{row.mobile}</TableCell>

<TableCell align="right">{row.address}</TableCell>

<TableCell align="right">{row.program}</TableCell>

<TableCell align="right">{row.branch}</TableCell>

<TableCell align="right">{row.cgpa}</TableCell>

</TableRow>

))}

</TableBody>

</Table>

</TableContainer>

);

}

const SearchStudentDetails = () => {

const [rollNo, setRollNo] = useState(null);

const [open, setOpen] = useState(false);

const [success, setSuccess] = useState(false);

const [studentData, setStudentData] = useState({

rollno: "",

name: "",

email: "",

mobile: "",

address: "",

program: "",

branch: "",

cgpa: "",

});

const handleRollNo = (e) => {

setRollNo(e.target.value);

};

const handleSearch = async () => {

const roll = parseInt(rollNo);

if (!isNaN(roll)) {

const res = await studentService.getStudentDetailsByRoll({ value: roll });

if (res.status) {

const studentDetails = res.data.studentInfo[0];

const studentAcademicDetails = res.data.studentAcademicInfo[0];

setStudentData({

rollno: studentDetails.rollno,

name: studentDetails.name,

email: studentDetails.email,

mobile: studentDetails.mobile,

address: `${studentDetails.address.city}, ${studentDetails.address.state}, PIN - ${studentDetails.address.pin}`,

program: studentAcademicDetails.program,

branch: studentAcademicDetails.branch,

cgpa: studentAcademicDetails.cgpa,

});

setOpen(true);

setSuccess(true);

} else {

setOpen(true);

setSuccess(false);

}

}

};

const handleClose = (event, reason) => {

if (reason === "clickaway") {

return;

}

setOpen(false);

};

return (

<>

<Snackbar

open={open}

autoHideDuration={6000}

onClose={handleClose}

anchorOrigin={{ vertical: "top", horizontal: "right" }}

>

<Alert

onClose={handleClose}

severity={success ? "success" : "error"}

variant="filled"

sx={{ width: "100%" }}

>

{success

? "Successfully Fetched the Student Setails!!"

: "Student details not Availaible!!"}

</Alert>

</Snackbar>

<Stack

marginBottom={"2rem"}

spacing={"0.4rem"}

sx={{

width: {

xs: "100%",

sm: "100%",

md: "50%",

},

display: "flex",

justifyContent: "center",

alignItems: "center",

flexDirection: "column",

}}

marginLeft={{

xs: "0rem",

sm: "0rem",

md: "20rem",

}}

marginTop={"2rem"}

>

<TextField variant="outlined" label="Search" onChange={handleRollNo} />

<Button variant="contained" color="success" onClick={handleSearch}>

<SearchIcon />

Search

</Button>

</Stack>

{success ? (

<BasicTable

rows={[

createData(

studentData.rollno,

studentData.name,

studentData.email,

studentData.mobile,

studentData.address,

studentData.program,

studentData.branch,

studentData.cgpa

),

]}

/>

) : (

<BasicTable />

)}

</>

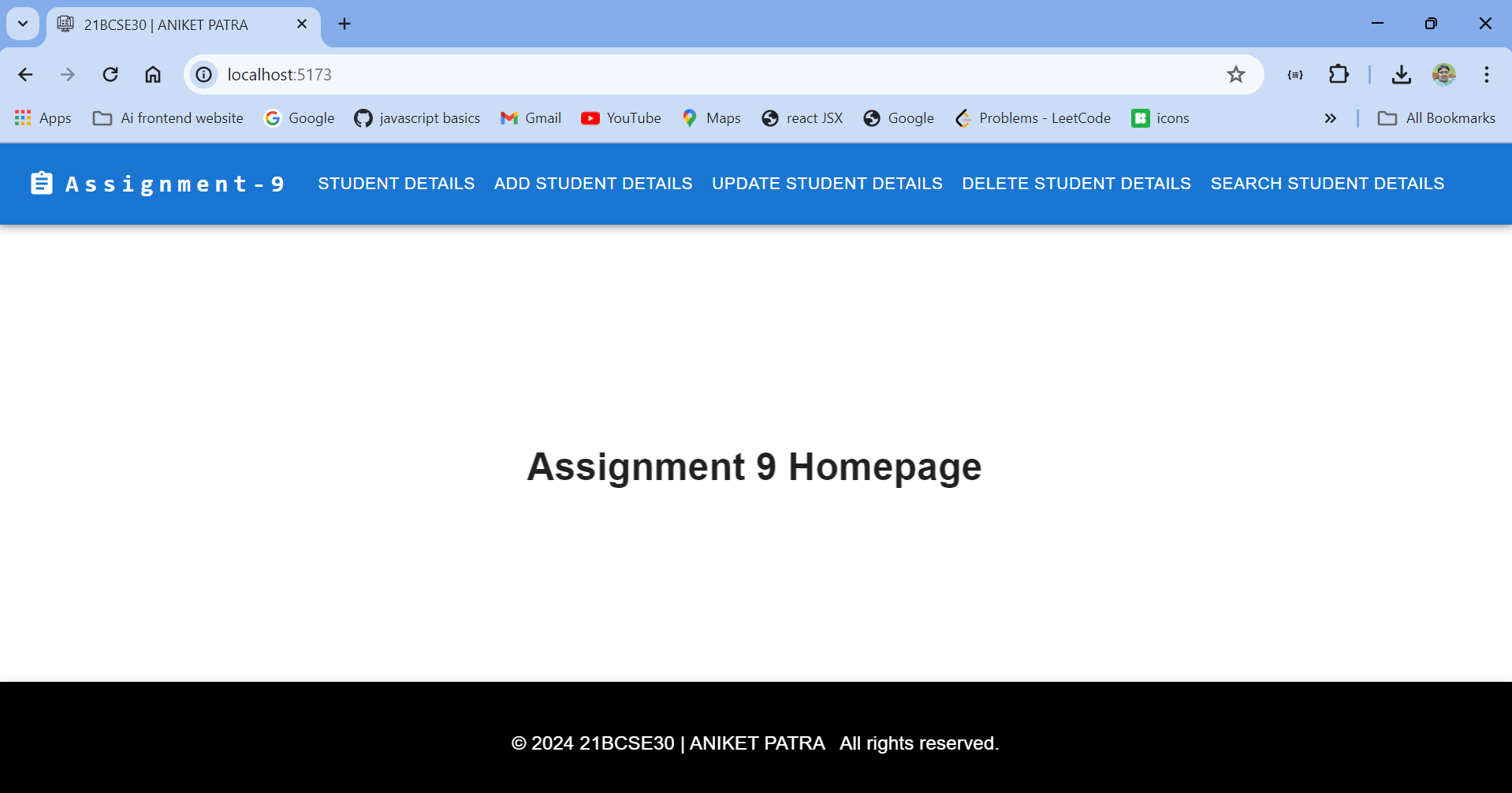
);

};

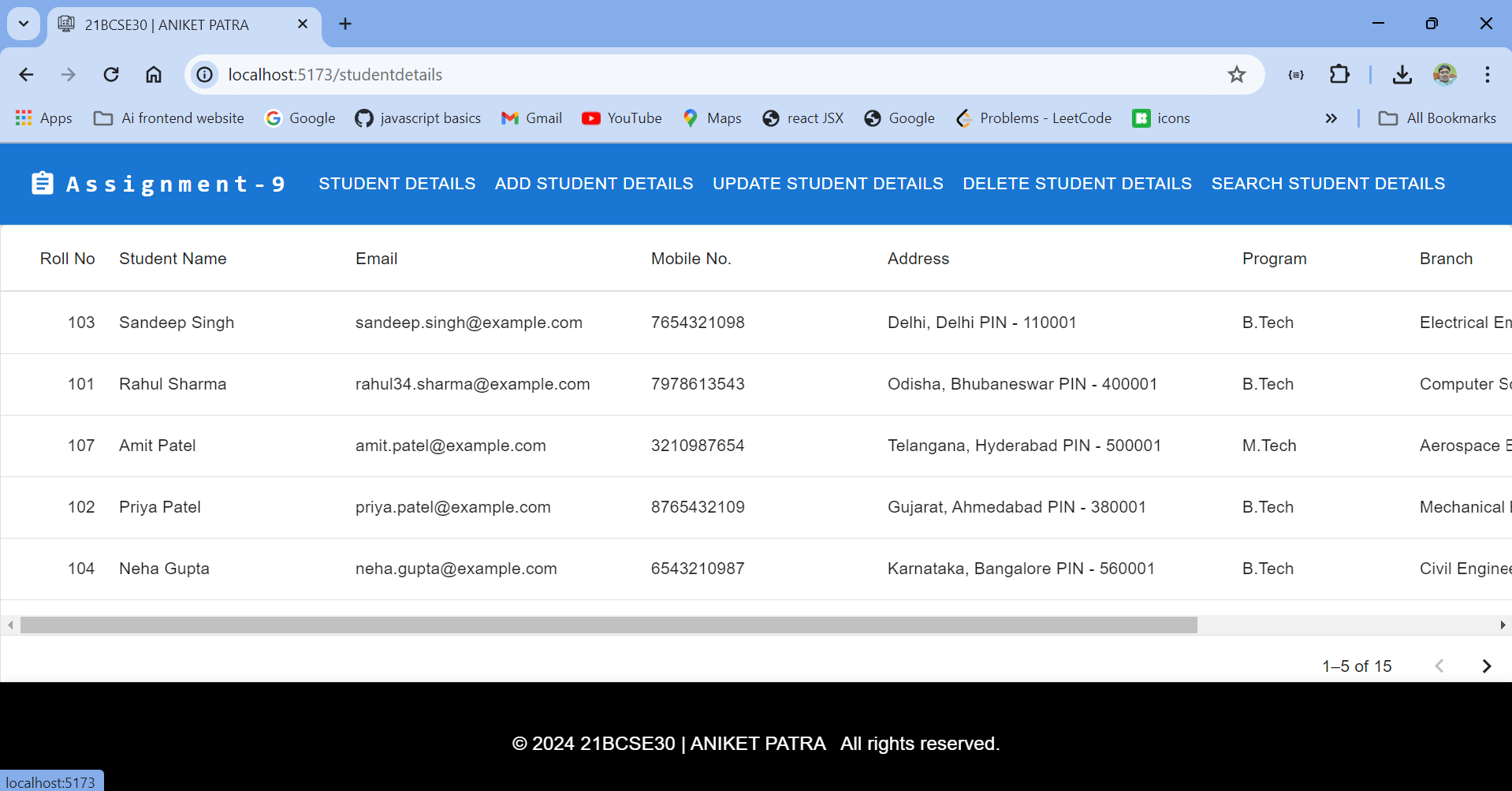
export default SearchStudentDetails;

**Output:-**

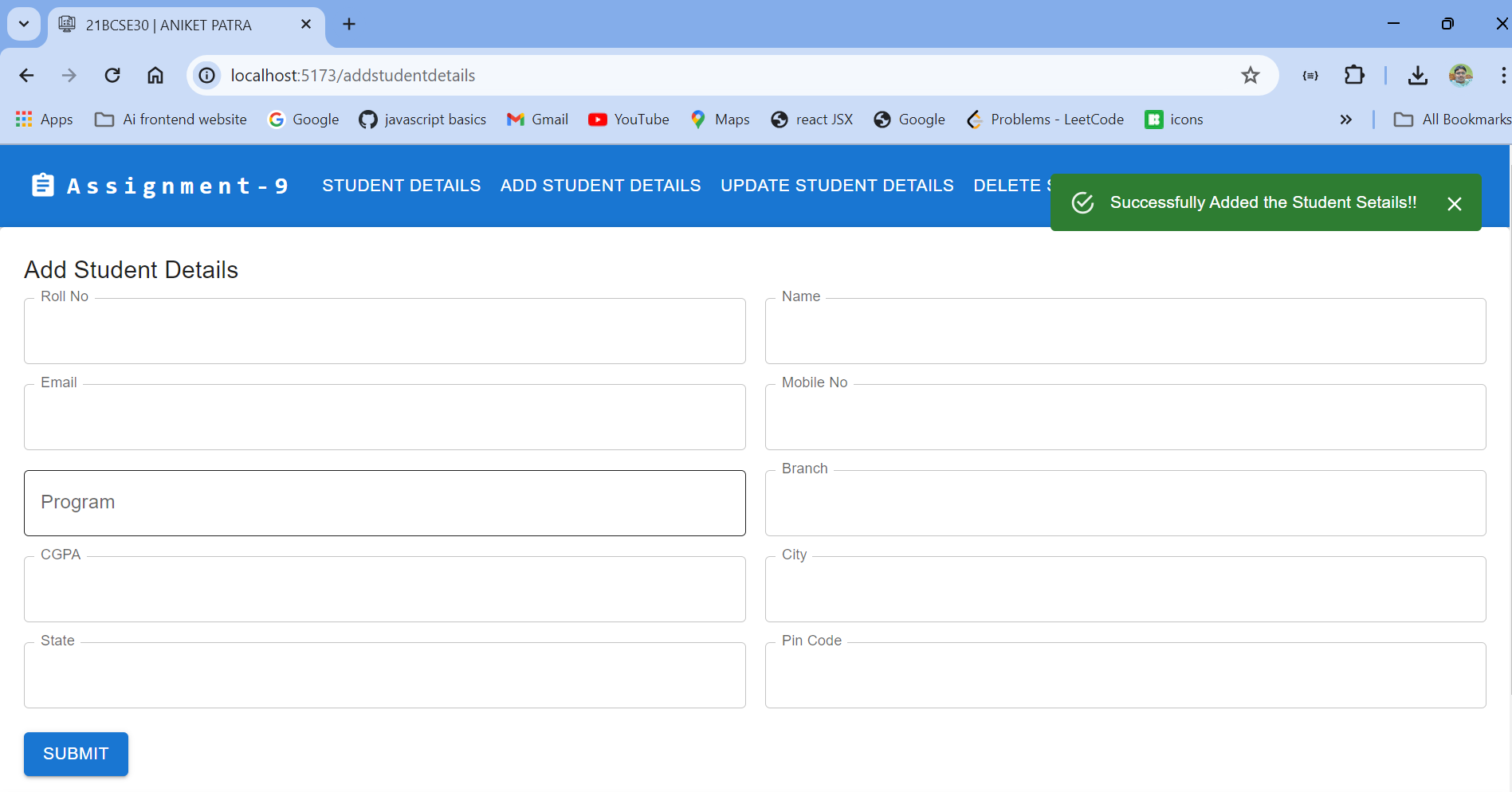
**Home.jsx**

****

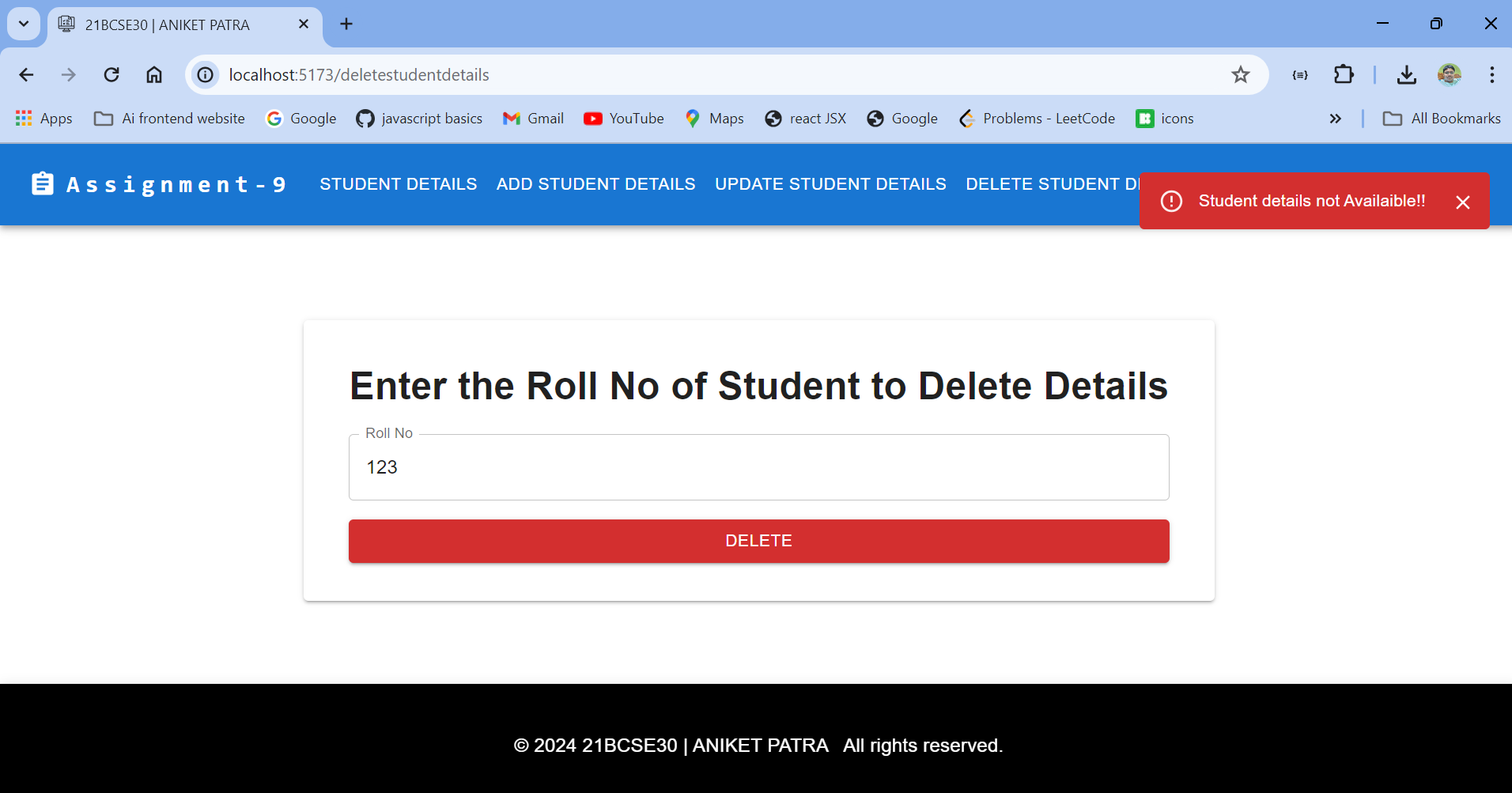
**Student Details**

****

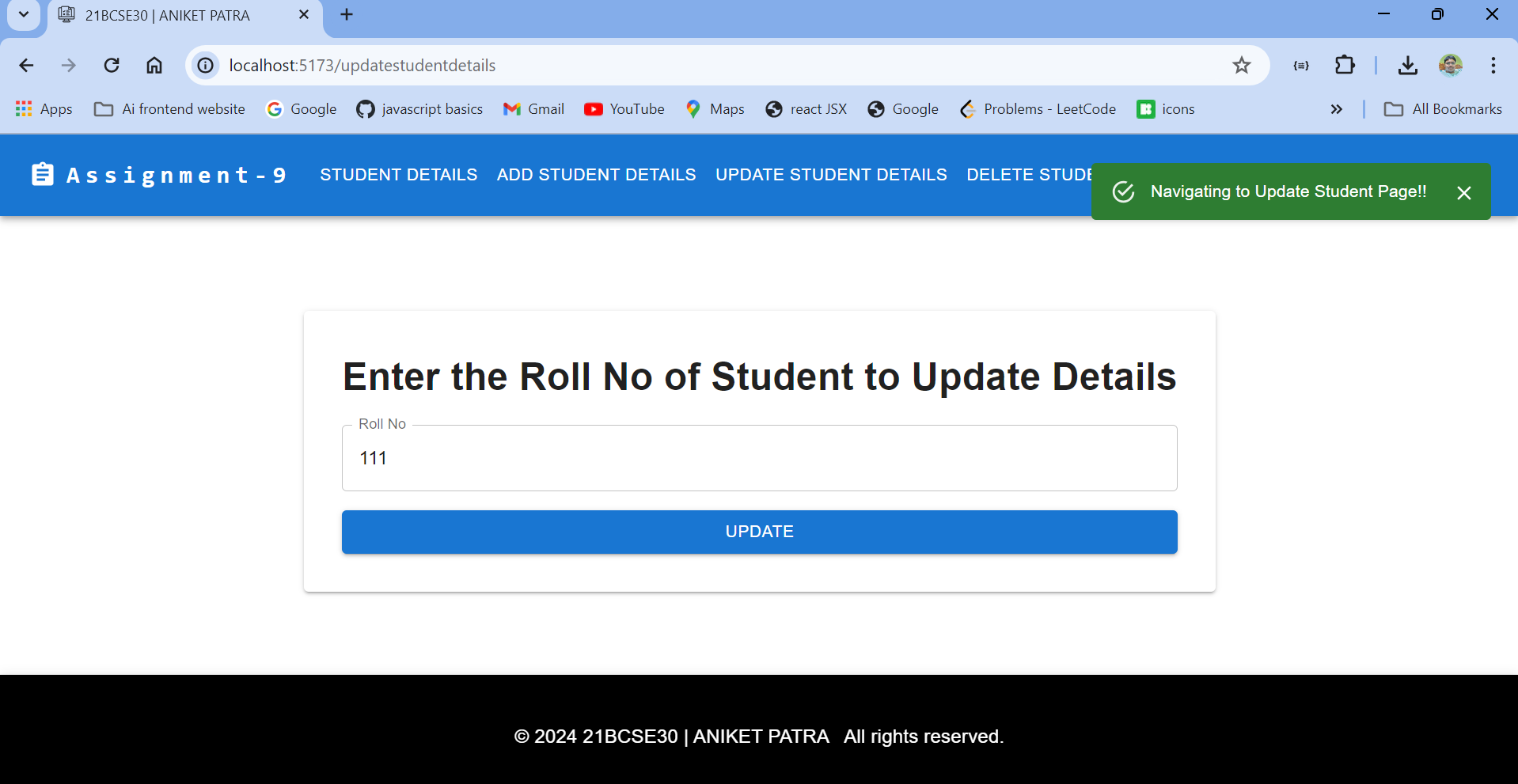
**Add Student Details**

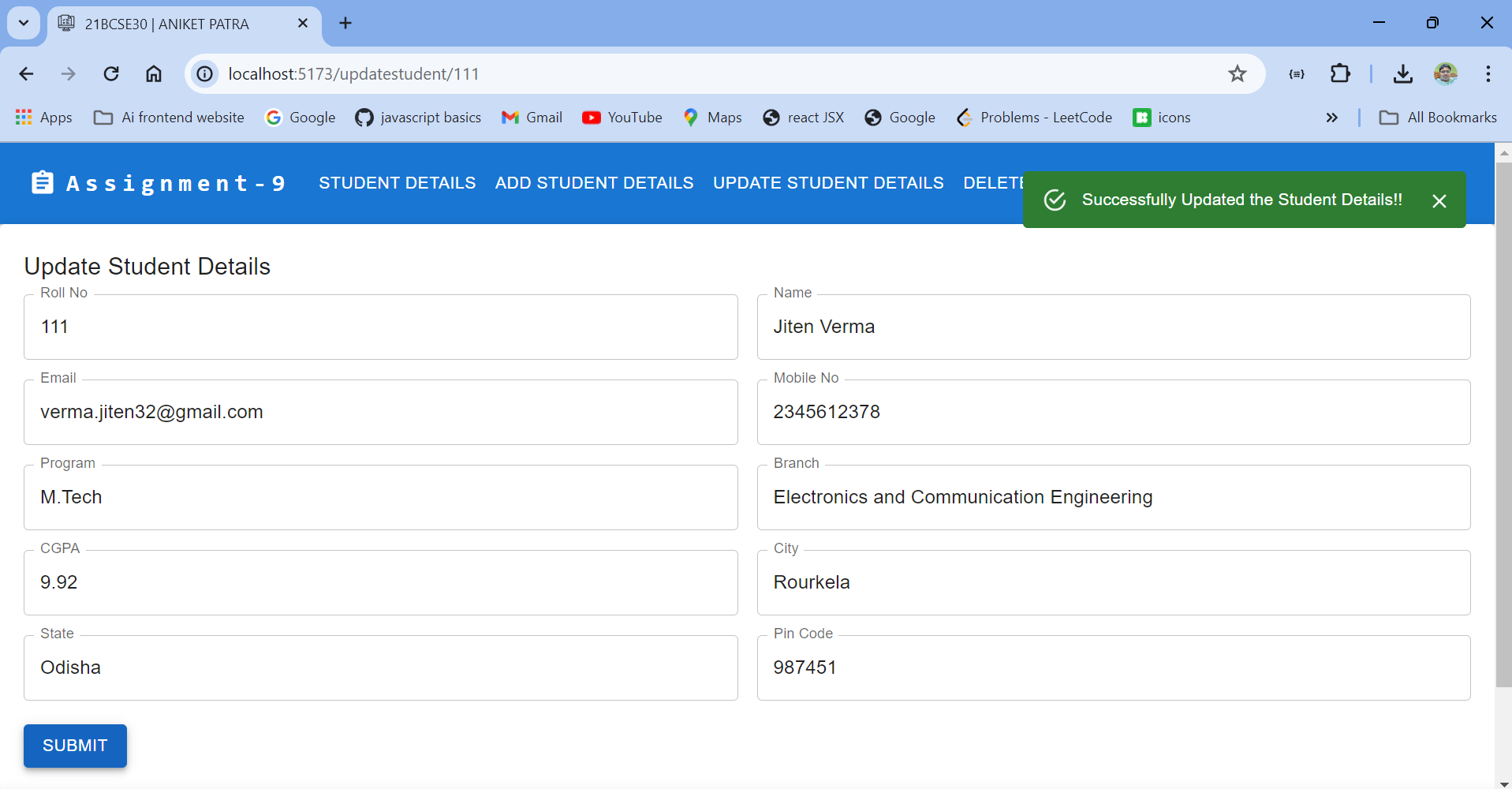
****

**DeleteStudent Details**

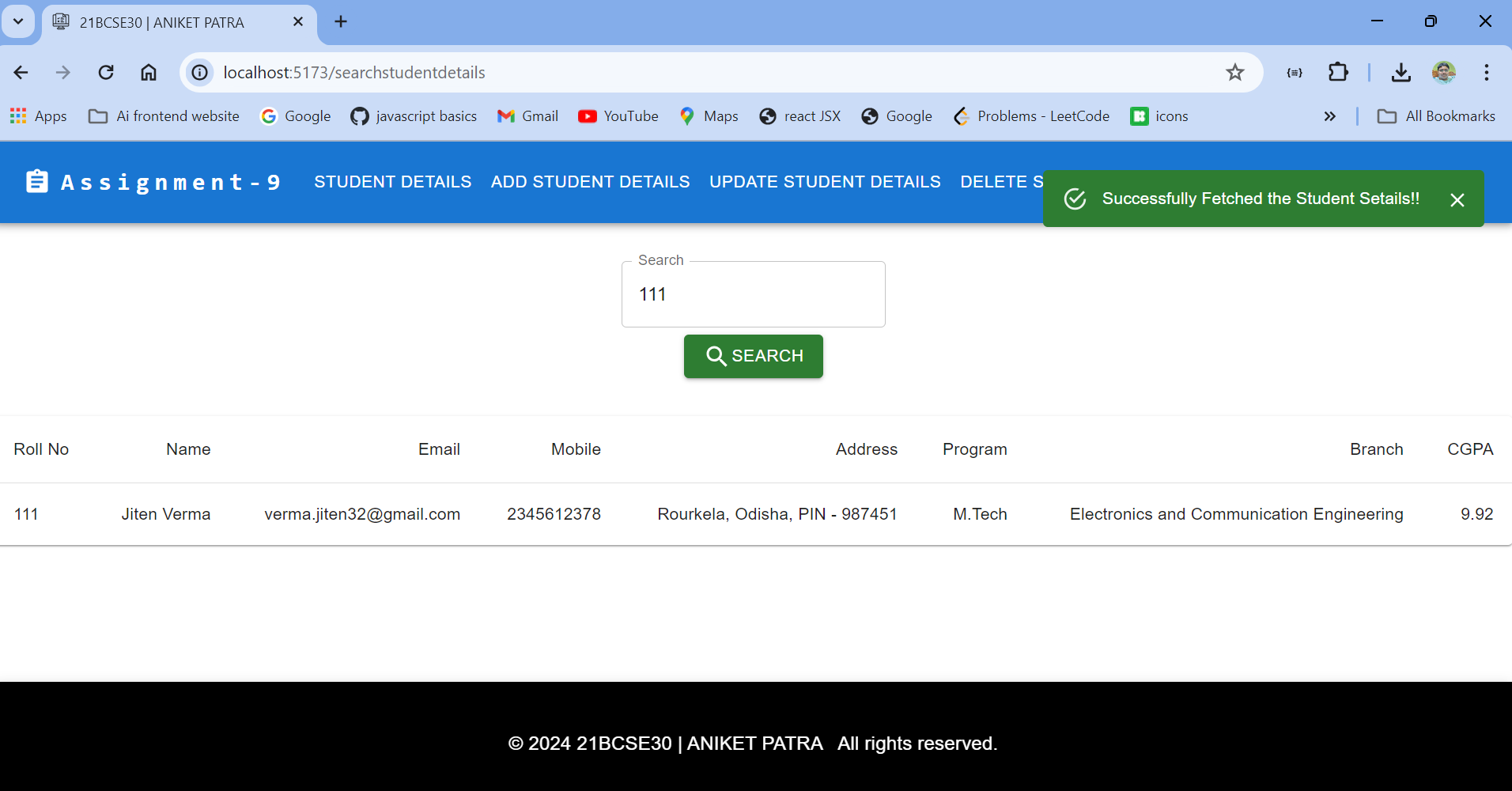
****

**UpdateStudent Details**

****

****

**SearchStudent**

****