**EX NO: 7**

**DATE: 07.03.2025**

**IMPLEMENT THE SYSTEM AS PER THE DETAILED DESIGN**

**AIM :**

To implement the system as per the detailed design.

**PROCEDURE:**

1. Backend Setup (Express + MongoDB)

* Initialize a new Node.js project:  
  npm init -y
* Install dependencies:  
  npm install express mongodb cors
* Create server.js and write API logic
* Start the backend server:  
  node server.js

2. Frontend Setup (React + Vite)

* Create a new Vite React project:  
  npm create vite@latest
* Navigate to the project folder:  
  cd frontend
* Install dependencies:  
  npm install
* Install React Router and Axios:  
  npm install react-router-dom axios
* Modify main.jsx to set up React Router.
* Start the frontend development server:  
  npm run dev

3. Run the Full Stack Application

* Start the backend:  
  cd backend && node server.js
* Start the frontend:  
  cd frontend && npm run dev

**PROGRAM:**

**//server.js**

const express = require('express');

const { MongoClient, ServerApiVersion, ObjectId } = require('mongodb');

const cors = require('cors');

const app = express();

const PORT = 5000;

app.use(cors());

app.use(express.json());

const uri = "";

const client = new MongoClient(uri, {

serverApi: {

version: ServerApiVersion.v1,

strict: true,

deprecationErrors: true,

}

});

// API Endpoints for Students

app.get('/students', async (req, res) => {

try {

await client.connect();

const students = await client.db("sis").collection("students").find().toArray();

res.json(students);

} catch (error) {

console.error('Error fetching students:', error);

res.status(500).json({ message: 'Error fetching students', error });

} finally {

await client.close();

}

});

app.post('/students', async (req, res) => {

try {

await client.connect();

const newStudent = req.body;

await client.db("sis").collection("students").insertOne(newStudent);

res.json(newStudent);

} catch (error) {

console.error('Error adding student:', error);

res.status(500).json({ message: 'Error adding student', error });

} finally {

await client.close();

}

});

app.put('/students/:id', async (req, res) => {

try {

await client.connect();

const studentId = req.params.id;

const updatedStudent = req.body;

await client.db("sis").collection("students").updateOne({ \_id: new ObjectId(studentId) }, { $set: updatedStudent });

res.json({ message: 'Student updated' });

} catch (error) {

console.error('Error updating student:', error);

res.status(500).json({ message: 'Error updating student', error });

} finally {

await client.close();

}

});

app.delete('/students/:id', async (req, res) => {

try {

await client.connect();

const studentId = req.params.id;

await client.db("sis").collection("students").deleteOne({ \_id: new ObjectId(studentId) });

res.json({ message: 'Student deleted' });

} catch (error) {

console.error('Error deleting student:', error);

res.status(500).json({ message: 'Error deleting student', error });

} finally {

await client.close();

}

});

// API Endpoints for Faculty

app.get('/faculty', async (req, res) => {

try {

await client.connect();

const faculty = await client.db("sis").collection("faculty").find().toArray();

res.json(faculty);

} catch (error) {

console.error('Error fetching faculty:', error);

res.status(500).json({ message: 'Error fetching faculty', error });

} finally {

await client.close();

}

});

app.post('/faculty', async (req, res) => {

try {

await client.connect();

const newFaculty = req.body;

await client.db("sis").collection("faculty").insertOne(newFaculty);

res.json(newFaculty);

} catch (error) {

console.error('Error adding faculty:', error);

res.status(500).json({ message: 'Error adding faculty', error });

} finally {

await client.close();

}

});

app.delete('/faculty/:id', async (req, res) => {

try {

await client.connect();

const facultyId = req.params.id;

await client.db("sis").collection("faculty").deleteOne({ \_id: new ObjectId(facultyId) });

res.json({ message: 'Faculty deleted' });

} catch (error) {

console.error('Error deleting faculty:', error);

res.status(500).json({ message: 'Error deleting faculty', error });

} finally {

await client.close();

}

});

// API Endpoints for Courses

app.get('/courses', async (req, res) => {

try {

await client.connect();

const courses = await client.db("sis").collection("courses").find().toArray();

res.json(courses);

} catch (error) {

console.error('Error fetching courses:', error);

res.status(500).json({ message: 'Error fetching courses', error });

} finally {

await client.close();

}

});

app.post('/courses', async (req, res) => {

try {

await client.connect();

const newCourse = req.body;

await client.db("sis").collection("courses").insertOne(newCourse);

res.json(newCourse);

} catch (error) {

console.error('Error adding course:', error);

res.status(500).json({ message: 'Error adding course', error });

} finally {

await client.close();

}

});

app.delete('/courses/:id', async (req, res) => {

try {

await client.connect();

const courseId = req.params.id;

await client.db("sis").collection("courses").deleteOne({ \_id: new ObjectId(courseId) });

res.json({ message: 'Course deleted' });

} catch (error) {

console.error('Error deleting course:', error);

res.status(500).json({ message: 'Error deleting course', error });

} finally {

await client.close();

}

});

// API Endpoints for Marks

app.get('/marks', async (req, res) => {

try {

await client.connect();

const marks = await client.db("sis").collection("marks").find().toArray();

res.json(marks);

} catch (error) {

console.error('Error fetching marks:', error);

res.status(500).json({ message: 'Error fetching marks', error });

} finally {

await client.close();

}

});

app.post('/marks', async (req, res) => {

try {

await client.connect();

const newMarks = req.body;

await client.db("sis").collection("marks").insertOne(newMarks);

res.json(newMarks);

} catch (error) {

console.error('Error adding marks:', error);

res.status(500).json({ message: 'Error adding marks', error });

} finally {

await client.close();

}

});

app.delete('/marks/:id', async (req, res) => {

try {

await client.connect();

const marksId = req.params.id;

await client.db("sis").collection("marks").deleteOne({ \_id: new ObjectId(marksId) });

res.json({ message: 'Marks deleted' });

} catch (error) {

console.error('Error deleting marks:', error);

res.status(500).json({ message: 'Error deleting marks', error });

} finally {

await client.close();

}

});

app.listen(PORT, () => {

console.log(`Server is running on http://localhost:${PORT}`);

});

**//App.jsx**

import { BrowserRouter as Router, Routes, Route } from 'react-router-dom';

import Login from './Login';

import Admin from './admin/Admin';

import FacultyInfo from './admin/FacultyInfo';

import StudentInfo from './admin/StudentInfo';

import ReportInfo from './admin/ReportInfo';

import CourseDetails from './admin/CourseDetails';

import Faculty from './faculty/Faculty';

import Faculty1 from './faculty/Faculty1';

import Student1 from './faculty/Student1';

import Course1 from './faculty/Course1';

import Student from './student/Student';

import Report1 from './student/Report1';

function App() {

return (

<Router>

<Routes>

<Route path="/" element={<Login />} />

<Route path="/admin" element={<Admin />} />

<Route path="/faculty" element={<Faculty />} />

<Route path="/student" element={<Student />} />

<Route path="/student-info" element={<StudentInfo />} />

<Route path="/faculty-info" element={<FacultyInfo />} />

<Route path="/course-details" element={<CourseDetails />} />

<Route path="/report-info" element={<ReportInfo />} />

<Route path="/student1" element={<Student1 />} />

<Route path="/faculty1" element={<Faculty1 />} />

<Route path="/course1" element={<Course1 />} />

<Route path="/report1" element={<Report1 />} />

</Routes>

</Router> );}

export default App;

**//Login.jsx**

import { useState } from 'react';

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

import './css/Login.css';

function Login() {

const [username, setUsername] = useState('');

const [password, setPassword] = useState('');

const [role, setRole] = useState('admin');

const navigate = useNavigate();

const handleLogin = () => {

if (username === role && password === role) {

navigate(`/${role}`);

} else {

alert('Invalid credentials');} };

return (

<div className="login-container">

<h2>LOGIN</h2>

<input

type="text"

placeholder="Username"

value={username}

onChange={(e) => setUsername(e.target.value)}

/>

<input

type="password"

placeholder="Password"

value={password}

onChange={(e) => setPassword(e.target.value)}

/>

<select value={role} onChange={(e) => setRole(e.target.value)}>

<option value="admin">Admin</option>

<option value="faculty">Faculty</option>

<option value="student">Student</option>

</select>

<button onClick={handleLogin}>Login</button>

</div> );}

export default Login;

**//Admin.jsx**

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

import '../css/Home.css';

function Admin() {

const navigate = useNavigate();

return (

<div className="admin-container">

<h2>Welcome Admin</h2>

<div className="admin-folders">

<div className="folder student-info" onClick={() => navigate('/student-info')}>

<span>Student Information</span>

</div>

<div className="folder faculty-info" onClick={() => navigate('/faculty-info')}>

<span>Faculty Information</span>

</div>

<div className="folder course-details" onClick={() => navigate('/course-details')}>

<span>Course Details</span>

</div>

<div className="folder report" onClick={() => navigate('/report-info')}>

<span>Report</span>

</div>

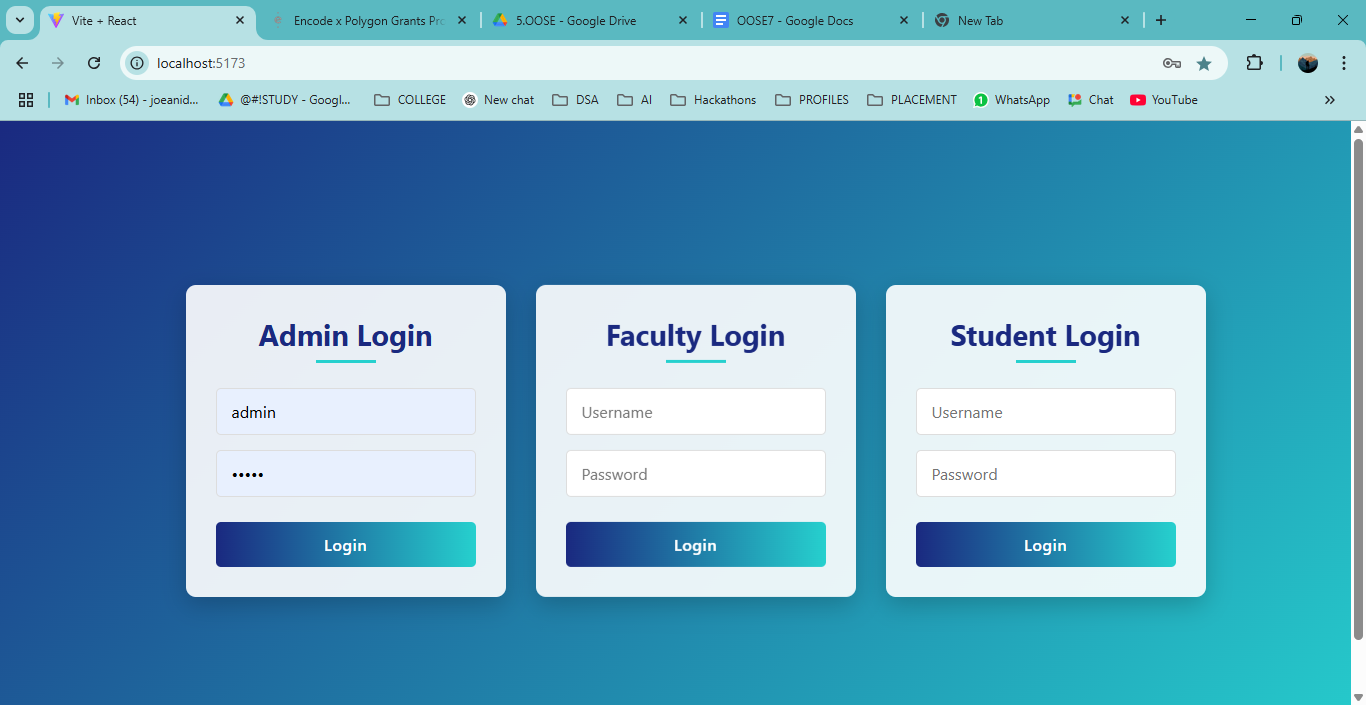
</div>

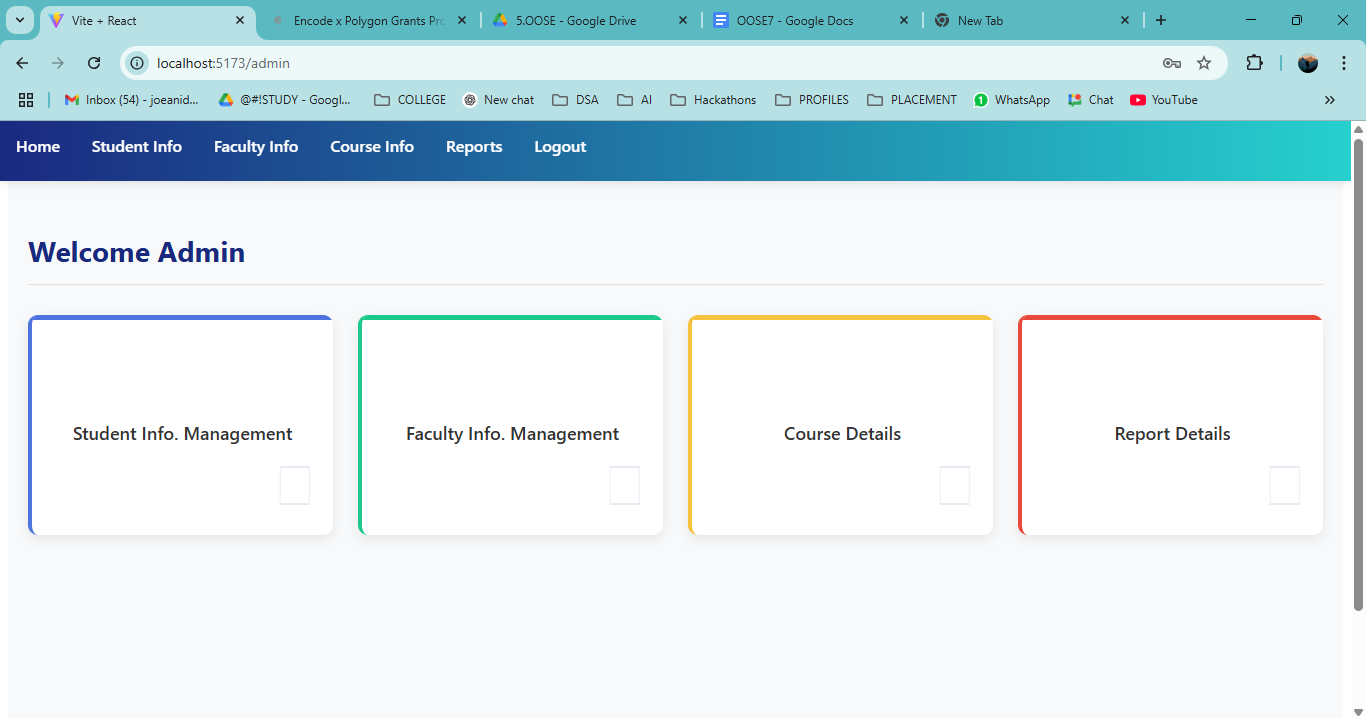
</div>

);}

export default Admin;

**OUTPUT:**

****

****

**RESULT:**