SHRI VILEPARLE KELAVANI MANDAL'S
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC ACCREDITED with "A" GRADE (CGPA : 3.18)

COURSE CODE: DJS22ITL604 **DATE: 04-02-2025**

COURSE NAME: Full Stack Web Development Laboratory **CLASS: TYBTech**

NAME: Anish Sharma **DIV: IT1-1** **ROLL: I011**

DEPARTMENT OF INFORMATION TECHNOLOGY
EXPERIMENT NO. 02

CO/LO: CO1-Develop a full stack web application.

AIM / OBJECTIVE: Building a Simple React Application Create a simple React app with multiple components, manage state, and pass props Connecting React Front-End to Express.js Back-End.

THEORY:

React is a JavaScript library for building user interfaces. It allows you to create reusable UI components and manage the application's state efficiently. React follows a component-based architecture, where each UI piece is built as an independent, reusable component.

- **State Management**: React provides useState and useEffect hooks to manage state and lifecycle methods in functional components.
- **Props**: Props (short for properties) allow data to be passed between components.
- **Component Hierarchy**: Breaking UI into smaller components improves maintainability and reusability.

1. Building a Simple React Application a) Components in React

React applications are built using **components**, which are reusable pieces of UI. Components can be classified into two main types:

- **Functional Components**: Defined as JavaScript functions that return JSX (JavaScript XML). They are lightweight and mainly used for rendering UI.
- **Class Components**: Defined as JavaScript ES6 classes extending React.Component, allowing the use of lifecycle methods.

Example of a functional component:

```
function Greeting(props) {
  return <h1>Hello, {props.name}!</h1>;
}
```

b) Managing State in React

State is an object that stores dynamic data in a component. It is mainly used in **class components** via this.state and updated using setState(), but in functional components, the useState hook is commonly used.

Example using useState:

```
import { useState } from "react";

function Counter() {
  const [count, setCount] = useState(0);
 return (
    <div>
      <p>Count: {count}</p>
      <button onClick={() => setCount(count + 1)}>Increment</button> </div>
  ); }
```

c) Passing Props in React

Props (short for "properties") allow data to be passed from parent to child components. They are read-only and cannot be modified within the child component.

Example of passing props:

```
function WelcomeMessage({ user }) {
  return <h2>Welcome, {user}!</h2>;
}

function App() {
  return <WelcomeMessage user="Alice" />;
}
```

Express.js Overview

Express.js is a minimal and flexible Node.js web application framework that simplifies backend development. It helps create RESTful APIs for handling data and communicating with the frontend.

- **Routing**: Express allows defining multiple API routes to handle requests.
- **Middleware**: Middleware functions process incoming requests before sending responses.

- **CORS Handling**: Cross-Origin Resource Sharing (CORS) allows the frontend to communicate with the backend hosted on a different domain.

2. Connecting React Front-End to Express.js Back-End

React alone handles the front-end, but to fetch and manipulate data, a back-end server is required. Express.js is a minimal and flexible Node.js framework used to build APIs. The communication between React (front-end) and Express.js (back-end) is done via HTTP requests.

DEPARTMENT OF INFORMATION TECHNOLOGY

a) Setting Up Express.js Backend

Express.js is used to create a server that handles requests and sends responses.

Example of a simple Express server (server.js):

```
const  express  =  require('express');
const cors = require('cors'); const app
=      express();      app.use(cors());
app.use(express.json());
app.get('/api/message', (req, res) => {
  res.json({ message: 'Hello from Express.js backend!' });
});
app.listen(5000, () => {
  console.log('Server running on port 5000');
});
```

b) Fetching Data from Express API in React

React can fetch data from the Express.js server using the fetch API or libraries like axios.

Example:

```
import { useEffect, useState } from "react"; function
FetchMessage() {
  const [message, setMessage] = useState(""); useEffect(()
  => {
    fetch("http://localhost:5000/api/message")
      .then(response => response.json())
      .then(data => setMessage(data.message));
  }, []);
  return <h3>{message}</h3>;
}
```

PROCEDURE

Building an Administrator component which handles the database management process.

APP

```
"use client"

import { useState } from "react";
import YearSemesterSelector from "./components/YearSemesterSelector"; import
FileUpload from "./components/FileUpload"; import CourseTypeSelector from
"./components/CourseTypeSelector";
```

```
import SubjectInput from "./components/SubjectInput";
import MinorStudentInput from "./components/MinorStudentInput";
import { Button } from "@/components/ui/button";
 export default function StudentManagementApp() {
const [year, setYear] = useState("");   const [semester,
setSemester] = useState("");   const [courseType,
setCourseType] = useState("");   const [subjects,
setSubjects] = useState([]);   const [minorStudents,
setMinorStudents] = useState([]);   const [studentData,
setStudentData] = useState(null);
   const handleFileUpload = (data) =>
{      setStudentData(data);
  };    const clearAll =
() => {      setYear("");
setSemester("");
setCourseType("");
setSubjects([]);
setMinorStudents([]);
setStudentData(null);
  };
  return (
    <div className="container mx-auto p-4">       <h1 className="text-
2xl font-bold mb-4">Student Management App</h1>
      <YearSemesterSelector
year={year}          semester={semester}
onYearChange={setYear}
        onSemesterChange={setSemester}
      />
      {year && semester && <FileUpload onFileUpload={handleFileUpload} />}
      <CourseTypeSelector
courseType={courseType}
onCourseTypeChange={setCourseType}
      />
      <SubjectInput
subjects={subjects}
```

```
        onSubjectsChange={setSubjects}
      />
      <MinorStudentInput
 minorStudents={minorStudents}
        onMinorStudentsChange={setMinorStudents}
      />
      <Button onClick={clearAll} className="mt-4">
        Clear All
      </Button>
      {studentData && (
        <div className="mt-4">
          <h2 className="text-xl font-semibold mb-2">Uploaded Student Data:</h2>
          <pre className="bg-gray-100 p-4 rounded">{JSON.stringify(studentData, null,
2)}</pre>
        </div>
      )}
    </div>
  ); }
```

COMPONENTS:

```
import { Select, SelectContent, SelectItem, SelectTrigger, SelectValue } from
"@/components/ui/select";
 export default function CourseTypeSelector({ courseType, onCourseTypeChange })
{   return (
    <div className="mb-4">
      <Select value={courseType} onValueChange={onCourseTypeChange}>
        <SelectTrigger className="w-[180px]">
          <SelectValue placeholder="Select Course Type" />
</SelectTrigger>
        <SelectContent>
          <SelectItem value="Regular">Regular</SelectItem>
          <SelectItem value="DLE">DLE</SelectItem>
          <SelectItem value="ILE">ILE</SelectItem>
          <SelectItem value="ETD">ETD</SelectItem>
        </SelectContent>
      </Select>
    </div>
  );
}  import { useState } from "react"; import {
Button } from "@/components/ui/button"; import
{ Input } from "@/components/ui/input"; import
* as XLSX from "xlsx";
 export default function FileUpload({ onFileUpload })
{
```

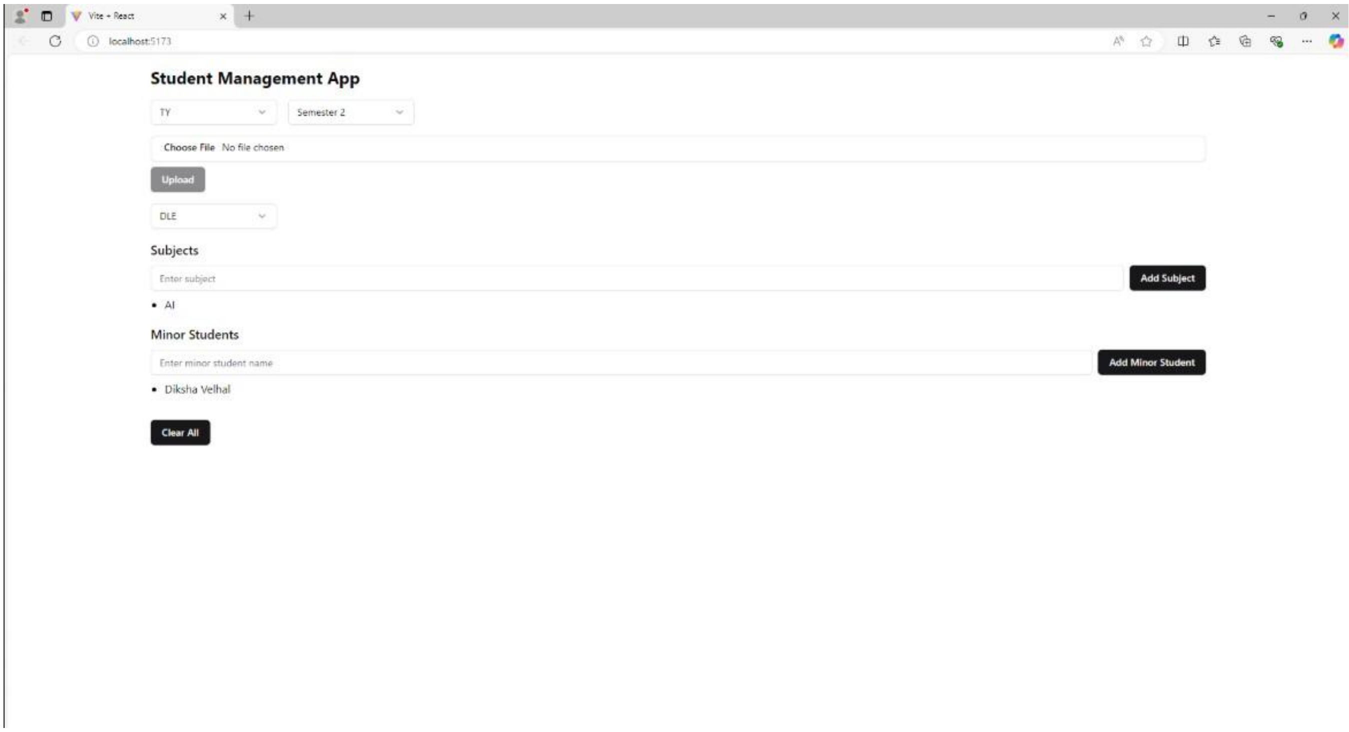
```
  const [file, setFile] = useState(null); const

  handleFileChange = (e) => {

    if (e.target.files) {
      setFile(e.target.files[0]);
    }
  };    const handleUpload = async
() => {      if (!file) return;
     const reader = new FileReader();     reader.onload =
(e) => {        const data = e.target?.result;       const
workbook = XLSX.read(data, { type: "array" });
const sheetName = workbook.SheetNames[0];       const
worksheet = workbook.Sheets[sheetName];       const json
= XLSX.utils.sheet_to_json(worksheet);
onFileUpload(json);
    };
reader.readAsArrayBuffer(file);
  };
  return (
    <div className="mb-4">
      <Input type="file" accept=".csv,.xlsx" onChange={handleFileChange}
className="mb2" />
      <Button onClick={handleUpload} disabled={!file}>
        Upload
      </Button>
    </div>
  );
}  import { useState } from "react"; import {
Input } from "@/components/ui/input"; import {
Button } from "@/components/ui/button";

export default function MinorStudentInput({ minorStudents, onMinorStudentsChange }) {
const [newMinorStudent, setNewMinorStudent] = useState("");
   const handleAddMinorStudent = () => {     if (newMinorStudent.trim()
!== "") {        onMinorStudentsChange([...minorStudents,
newMinorStudent.trim()]);
```

```
      setNewMinorStudent("");
    }   };

return (

<div

className=

"mb-4">
```

```
      <h3 className="text-lg font-semibold mb-2">Minor Students</h3>
      <div className="flex space-x-2 mb-2">
        <Input           type="text"
value={newMinorStudent}          onChange={(e) =>
setNewMinorStudent(e.target.value)}
placeholder="Enter minor student name"
className="flex-grow"
        />
        <Button onClick={handleAddMinorStudent}>Add Minor Student</Button>
</div>
      <ul className="list-disc pl-5">
        {minorStudents.map((student, index) => (
          <li key={index}>{student}</li>
        ))}
      </ul>
    </div>
  );
}  import { useState } from "react"; import {
Input } from "@/components/ui/input"; import {
Button } from "@/components/ui/button";
 export default function SubjectInput({ subjects, onSubjectsChange })
{   const [newSubject, setNewSubject] = useState("");
   const handleAddSubject = () => {     if
(newSubject.trim() !== "") {
onSubjectsChange([...subjects, newSubject.trim()]);
setNewSubject("");
    }
  };
  return (
    <div className="mb-4">
      <h3 className="text-lg font-semibold mb-2">Subjects</h3>
      <div className="flex space-x-2 mb-2">
        <Input           type="text"
value={newSubject}           onChange={(e) =>
setNewSubject(e.target.value)}
```

```
  placeholder="Enter subject"
  className="flex-grow"
/>
<Button onClick={handleAddSubject}>Add Subject</Button>
```

```
      </div>
      <ul className="list-disc pl-5">
        {subjects.map((subject, index) => (
          <li key={index}>{subject}</li>
        ))}
      </ul>
    </div>
  );
}  import { Select, SelectContent, SelectItem, SelectTrigger, SelectValue }
from "@/components/ui/select";
 export default function
YearSemesterSelector({   year,   semester,
onYearChange,   onSemesterChange,
}) {
return (
    <div className="flex space-x-4 mb-4">
      <Select value={year} onValueChange={onYearChange}>
        <SelectTrigger className="w-[180px]">
          <SelectValue placeholder="Select Year" />
        </SelectTrigger>
        <SelectContent>
          <SelectItem value="SY">SY</SelectItem>
          <SelectItem value="TY">TY</SelectItem>
          <SelectItem value="BE">BE</SelectItem>
        </SelectContent>
      </Select>
      {year && (
        <Select value={semester} onValueChange={onSemesterChange}>
          <SelectTrigger className="w-[180px]">
            <SelectValue placeholder="Select Semester" />
          </SelectTrigger>
          <SelectContent>
            <SelectItem value="Semester 1">Semester 1</SelectItem>
            <SelectItem value="Semester 2">Semester 2</SelectItem>
          </SelectContent>
        </Select>
      )}
    </div>
  );
}
```



Student Management App

TY

Semester 2

No file chosen

DLE

Subjects

Enter subject

- AI

Minor Students

Enter minor student name

- Diksha Velhal

CONCLUSION:

In this experiment we builda Simple React Application Create a simple React app with multiple components, manage state, and pass props Connecting React Front-End to Express.js Back-End.

BOOKS AND WEB RESOURCES:

[1] Fullstackopen, "Full Stack Open," Fullstackopen.com. [Online]. Available: https://fullstackopen.com/en

[2] freeCodeCamp, "The Complete Guide to Full Stack React Development," freeCodeCamp.org. [Online]. Available: https://www.freecodecamp.org/news/full-stack-react-application-architecture/

DigitalOcean, "How To Get Started with the MERN Stack," DigitalOcean.com. [Online]. Available: https://www.digitalocean.com/community/tutorials/getting-started-with-the-mern-stack

[3] Media, "Express.js Crash Course," *YouTube*. [Online]. Available: https://www.youtube.com/watch?v=L72fhGm1tfE.

[4] The Net Ninja, "Express.js Tutorial for Beginners," *YouTube*. [Online]. Available: https://youtu.be/98BzS5Oz5E4?si=mnOWtp0QHFRn7uf