EXPERIMENT-3.2

- **AIM: -** To design and implement a simple Library Management UI that allows users to search for books, add new books, and remove existing books, demonstrating core full-stack development concepts.
- **THEORY: -** Full Stack Development integrates both frontend (UI/UX) and backend (server, database).

✓ The frontend (React/HTML + CSS) enables interaction like search, add, and remove.

✓ The backend (Node.js + Express) handles data storage and retrieval.

✓ A database (MongoDB / in-memory for demo) stores book records (title, author, id).

✓ REST APIs (GET, POST, DELETE) facilitate communication between frontend and backend.

✓ Search functionality is implemented via string matching on book titles/authors.

- **CODE: -**

1. **BACKEND→**

```
// backend/index.js
const express = require("express"); const cors = require("cors");
const app = express(); app.use(cors()); app.use(express.json());

let books = [
{ id: 1, title: "Harry Potter", author: "J.K. Rowling" },
{ id: 2, title: "The Alchemist", author: "Paulo Coelho" },
];

    // Get all books app.get("/books", (req, res) => {
res.json(books);

    });

    // Add a new book app.post("/books", (req, res) => { const {
title, author } = req.body;
    const newBook = { id: books.length + 1, title, author };
books.push(newBook);
```

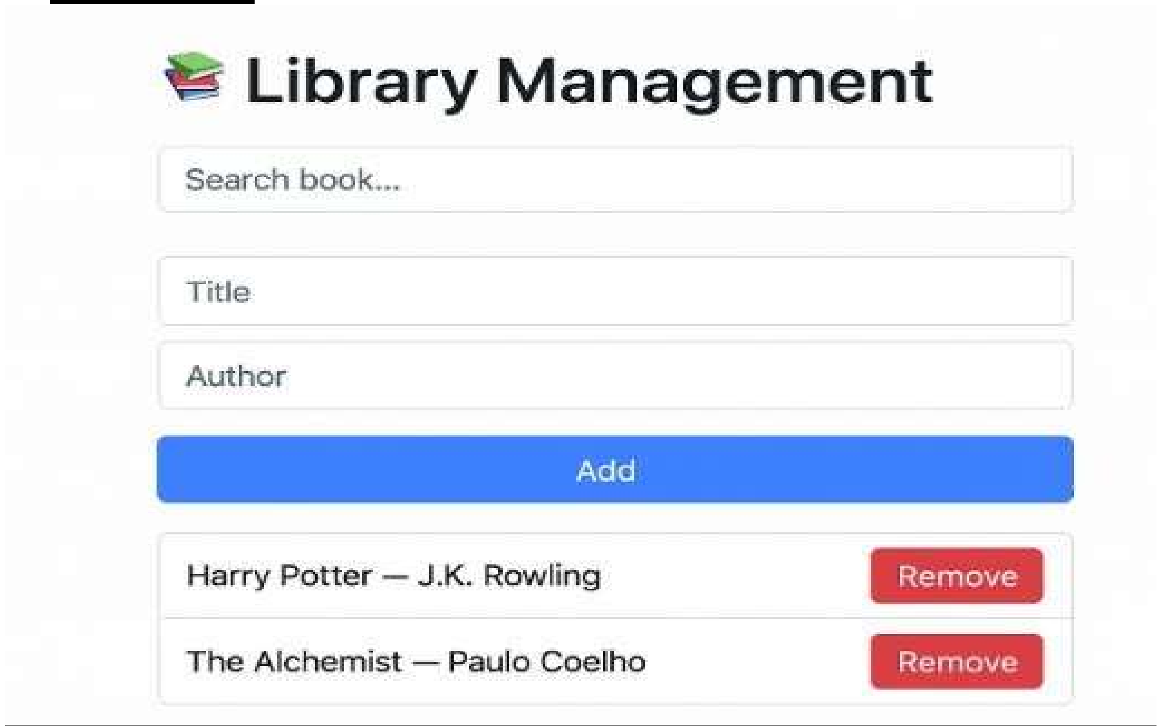
```
    res.json(newBook);
    });

    // Delete a book app.delete("/books/:id", (req, res) => { const { id } = req.params;
    books = books.filter((book) => book.id !== parseInt(id)); res.json({ message: "Book removed" });
    });

    app.listen(5000, () => console.log("Server running on port 5000"));

```

FRONTEND→

```
// frontend/App.js
import React, { useState, useEffect } from "react";

function App() {
const [books, setBooks] = useState([]); const [search, setSearch] = useState(""); const [title, setTitle] = useState(""); const [author, setAuthor] = useState("");

useEffect(() => { fetch("http://localhost:5000/books")
.then(res => res.json())
.then(data => setBooks(data));
}, []);

const addBook = () => { fetch("http://localhost:5000/books", { method: "POST",
headers: { "Content-Type": "application/json" }, body: JSON.stringify({ title, author }),
})
.then(res => res.json())
.then(data => setBooks([...books, data]));
};

const removeBook = (id) => { fetch(`http://localhost:5000/books/${id}`, { method:
"DELETE" })
.then(() => setBooks(books.filter(book => book.id !== id)));
};

return (
<div className="p-6 max-w-lg mx-auto">
<h1 className="text-2xl font-bold mb-4">µ–H‛ l_ µ_' Library Management</h1>

{/* Search */}
<input type="text"
placeholder="Search book..." className="border p-2 w-full mb-4" value={search}
onChange={(e) => setSearch(e.target.value)}
/>

{/* Add Book */}
<div className="flex gap-2 mb-4">
<input type="text" placeholder="Title" className="border p-2"

value={title} onChange={(e) => setTitle(e.target.value)} />
<input type="text" placeholder="Author" className="border p-2"
value={author} onChange={(e) => setAuthor(e.target.value)} />
<button onClick={addBook} className="bg-blue-500 text-white px-3 rounded">
Add
</button>
</div>

{/* Book List */}
<ul>
{books
.filter(b => b.title.toLowerCase().includes(search.toLowerCase()))
.map((book) => (
<li key={book.id} className="flex justify-between items-center border-b py-2">
<span>{book.title} — {book.author}</span>
<button onClick={() => removeBook(book.id)} className="bg-red-500 text--white px-2 rounded">
Remove
</button>
</li>
))}
</ul>
</div>
);
```

```
}

export default App;

```

- **OUTPUT→**



Library Management

Search book...

Title

Author

Add

Harry Potter — J.K. Rowling Remove

The Alchemist — Paulo Coelho Remove

LEARNING OUTCOMES→

- ✓ Understood integration of frontend and backend in a full-stack app.
- ✓ Learned how to implement CRUD operations (Create, Read, Delete) in REST APIs.
- ✓ Practiced state management in React with dynamic updates.
- ✓ Understood how to handle search filters in frontend UI.
- ✓ Gained experience in designing a realistic library management prototype.