Experiment 3

Student Name: Ayush Ranjan **UID: 23BCS10187**
Branch: CSE **Section/Group: KRG_2_B**
Semester: 5th **Date of Performance: 28/8/2025**
Subject Name: Full Stack- I **Subject Code: 23CSP-339**

1. Aim:
To build an interactive library management interface using React components with full CRUD (Create, Read, Update, Delete) functionality.

2. Objective:
- Design a book listing component.
- Implement search functionality.
- Add a form for new book entries.
- Enable update and delete capabilities for each book.
- Manage state using React hooks.

3. Code:

App.js:

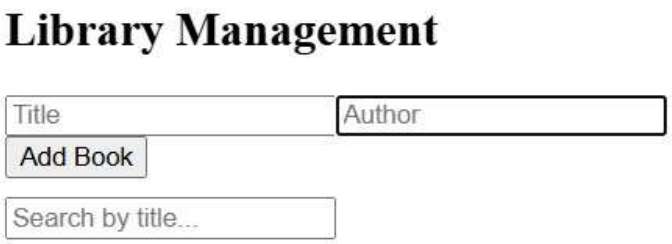
```
import React, { useState, useEffect } from 'react';
function App() {
  const [books, setBooks] = useState([]);
  const [formData, setFormData] = useState({ title: '', author: '' });
  const [searchTerm, setSearchTerm] = useState('');
  const [editingBookId, setEditingBookId] = useState(null);

  useEffect(() => {
    fetch('http://localhost:3001/books')
      .then(res => res.json())
      .then(data => setBooks(data));
  }, []);

  const handleChange = e => {
```

```
 setFormData({ ...formData, [e.target.name]: e.target.value });
 };

 const handleSubmit = e => {
  e.preventDefault();
  if (editingBookId) {
   fetch(`http://localhost:3001/books/${editingBookId}`, {
    method: 'PUT',
    headers: { 'Content-Type': 'application/json' },
    body: JSON.stringify(formData),
   })
    .then(res => res.json())
    .then(updatedBook => {
     setBooks(books.map(book => (book.id === editingBookId ? updatedBook :
book)));
     setEditingBookId(null);
     setFormData({ title: '', author: '' });
    });
  } else {
   fetch('http://localhost:3001/books', {
    method: 'POST',
    headers: { 'Content-Type': 'application/json' },
    body: JSON.stringify(formData),
   })
    .then(res => res.json())
    .then(newBook => {
     setBooks([...books, newBook]);
     setFormData({ title: '', author: '' });
    });
  }
 };

 const handleEdit = book => {
  setEditingBookId(book.id);
  setFormData({ title: book.title, author: book.author });
 };

 const handleDelete = id => {
```

```
fetch(`http://localhost:3001/books/${id}`, {
    method: 'DELETE',


}).then(() => {
    setBooks(books.filter(book => book.id !== id));
  });
 };

 const filteredBooks = books.filter(book =>
  book.title.toLowerCase().includes(searchTerm.toLowerCase())
 );

 return (
  <div style={{ padding: '20px' }}>
   <h2>Library Management</h2>
   <form onSubmit={handleSubmit}>
    <input
     name="title"
     placeholder="Title"
     value={formData.title}
     onChange={handleChange}
     required
    />
    <input
     name="author"
     placeholder="Author"
     value={formData.author}
     onChange={handleChange}
     required
    />
    <button type="submit">{editingBookId ? 'Update' : 'Add'} Book</button>
   </form>
   <input
    placeholder="Search by title..."
    value={searchTerm}
    onChange={e => setSearchTerm(e.target.value)}
    style={{ marginTop: '10px' }}
```

```
/>
    <ul>
      {filteredBooks.map(book => (
       <li key={book.id}>
         <strong>{book.title}</strong> by {book.author}
         <button onClick={() => handleEdit(book)}>Edit</button>


<button onClick={() => handleDelete(book.id)}>Delete</button>
        </li>
      ))}
    </ul>
  </div>
 );
}

export default App;
```

4. Output:



5. Learning Outcomes:

a) Learned to create and manage React functional components.
b) Gained experience using useState and useEffect hooks for state management and side effects.
c) Practiced handling forms with controlled components for adding and updating data.
d) Learned to implement real-time search and CRUD operations with a mock API.
e) Understood how to build a dynamic and responsive user interface using React.