Titles

Project Name – Online Library System

Build by – Shashikant Sahu

Batch – 1st March 2025

Table of Contents

Introduction

The **Online Library System** is a web-based application designed to provide users with a simple and intuitive platform to explore, search, and manage books. This system mimics the core functionalities of a traditional library, such as browsing categorized books, viewing detailed descriptions, and maintaining personal collections — all within a digital interface.

Built with modern technologies like **React**, **Redux Toolkit**, and **Tailwind CSS**, the platform ensures a responsive user experience while maintaining data persistence using **localStorage**. The system supports dynamic routing for individual book detail pages and offers the flexibility to add new books manually. Whether you're building a personal reading list or just exploring popular and categorized titles, this Online Library System serves as a foundational project for learning full-stack web development principles.

Organized File Structure

Creating a well-organized file structure is crucial for readability, maintainability, and collaboration in any web project. My project is already structured well, and here's a breakdown of each component:

Description of Files:

index.html

- The main HTML file used by Vite.
- It contains a where your React app is injected.
- You typically don't edit this much.

---

main.jsx

- Entry point of your React app.
- This is where the App component is rendered into the root element.

```
ReactDOM.createRoot(document.getElementById('root')).render(
  <React.StrictMode>
    <App />
  </React.StrictMode>
);
```

Home.jsx

- Displays the landing page of the library system.
- Includes:
  - A welcome message.
  - A list of book categories (e.g., Fiction, Sci-Fi).
  - A showcase of popular books (can be based on rating or hardcoded).
  - Links to browse or view details of books.

BrowseBooks.jsx

- Lets users search and browse all books.
- Features:
  - Search functionality for title/author.
  - Displays all books from Redux state.
  - Filters books based on user input.
  - Displays category buttons (e.g., Fiction, Romance) that can be linked to filtered views.
  - Each book has a “View Details” button linking to the book details page.

AddBook.jsx

- Provides a form for users to add a new book.
- Includes:
  - Form validation (e.g., required fields).
  - Fields: title, author, description, rating, image URL, category.
  - On submit: dispatches addBook() to Redux and saves it in localStorage.

BookDetails.jsx

- Shows detailed information about a specific book.
- Pulls the book by id from the Redux store via useParams().
- Displays:
  - Title, author, description, rating, and image.
- Has a “Back to Browse” link to return to the previous page.

BookCard.jsx *(optional but likely)*

- A reusable component used to render each book’s summary.
- Used by Home.jsx or BrowseBooks.jsx.
- Takes in props like book and renders image, title, author, short description, and "View Details" button.

Header.jsx

- Contains the navigation bar for the app.
- Typically includes links to:
  - Home
  - Browse Books
  - Add Book
- Ensures consistent top navigation across pages.

---

package.json

- Lists project dependencies like react, vite, and react-icons.
- Also contains scripts like npm run dev to start the app.

---

vite.config.js

- Configuration file for Vite.
- Usually left unchanged unless you need to customize the dev server, alias paths, etc.

README.md

- Documentation file explaining:
  - What the app does
  - How to install and run it.

.

Creativity and Presentation

Your **Online Library System** demonstrates thoughtful design, user experience, and modern development practices. Below is a detailed breakdown of the **technologies used** to build and present this project creatively:

Frontend Technologies

1. React.js
- Used for building dynamic UI components and managing state efficiently.
- Component-based architecture improves reusability and maintainability.

2. React Router DOM
- Enables navigation between different pages like Home, Browse Books, Add Book, and Book Details without page refresh.
- Dynamic routing allows book-specific detail views (/book-details/:id).

3. Redux Toolkit
- Centralized state management for handling books and app state.
- Clean integration with localStorage to persist data between sessions.

4. Tailwind CSS
- Provides utility-first styling.
- Helps create responsive, consistent, and modern designs with minimal custom CSS.
- Enhances the visual presentation of forms, buttons, layouts, and cards.

5. Vite
- Fast and modern development build tool.
- Offers lightning-fast hot reloads and optimized builds for production.

Storage and Data Handling

6. LocalStorage
- Used to persist book data added by users.
- Prevents loss of data on page refresh without requiring a backend or database.

1. Code

index.html code :-

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <link rel="icon" type="image/svg+xml" href="/vite.svg" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Online Library System</title>
  </head>
  <body>
    <div id="root"></div>
    <script type="module" src="/src/main.jsx"></script>
  </body>
</html>
```

main.jsx code :-

```
import { StrictMode } from "react";
import { createRoot } from "react-dom/client";
import "./index.css";
import App from "./App.jsx";
import { createBrowserRouter, RouterProvider } from "react-router-dom";
import BrowseBooks from "./components/BrowseBooks.jsx";
import AddBooks from "./components/AddBooks.jsx";
import Home from "./components/Home.jsx";
import Error from "./components/Error.jsx";
import BookDetails from "./components/BookDetails.jsx";
import BooksByCategory from "./components/BooksByCategory.jsx";

const appRouter = createBrowserRouter([
  {
    path: "/",
    element: <App />,
    children: [
      {
```

```
        path: "/",
        element: <Home />,
      },
      {
        path: "/browse-books",
        element: <BrowseBooks />,
      },
      {
        path: "/book-details/:id",
        element: <BookDetails />,
      },
      {
        path: "/add-books",
        element: <AddBooks />,
      },
          {
        path: "/books/:category",
        element: <BooksByCategory />,
      },
    ],
    errorElement: <Error />,
  },
]);

createRoot(document.getElementById("root")).render(
  <StrictMode>
    <RouterProvider router={appRouter} />
  </StrictMode>
);
```

App.jsx Code:-

```
import { Outlet } from "react-router-dom";
import "./App.css";
import Header from "./components/Header";
import { Provider } from "react-redux";
import { bookStore } from "./utils/bookStore";
```

```
function App() {
  return (
    <Provider store={bookStore}>
      <Header />
      <Outlet />
    </Provider>
  );
}

export default App;
```

Components/AddBooks.jsx Code:-

```
import { useState } from "react";
import { useDispatch } from "react-redux";
import { useNavigate } from "react-router-dom";
import { addBook } from "../utils/booksSlice";

function AddBooks() {
  const [formData, setFormData] = useState({
    image: "",
    title: "",
    author: "",
    category: "",
    description: "",
    rating: "",
  });
  const [errors, setErrors] = useState({});
  const dispatch = useDispatch();
  const navigate = useNavigate();

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

```
const validate = () => {
  const newErrors = {};

  if (!formData.image.trim()) {
    newErrors.image = "Image URL is required";
  }
  if (!formData.title.trim()) {
    newErrors.title = "Title is required";
  }
  if (!formData.author.trim()) {
    newErrors.author = "Author is required";
  }
  if (!formData.rating.trim()) {
    newErrors.rating = "Rating is required";
  }
  if (!formData.category.trim()) {
    newErrors.category = "Category is required";
  }
  if (!formData.description.trim()) {
    newErrors.description = "Description is required";
  }
  setErrors(newErrors);
  return Object.keys(newErrors).length === 0;
};

const handleSubmit = (e) => {
  e.preventDefault();
  if (validate()) {
    dispatch(addBook(formData));
    navigate("/browse-books");
  }
};

// console.log(formData);

return (
  <>
    <div className="bg-zinc-600 text-white h-auto w-[40%] m-auto mt-2 rounded-xl shadow-2xl">
```

```
<h1 className="font-bold text-3xl text-center mb-1 p-3 text-shadow-lg">
  Add a New Book Details
</h1>
<form
  onSubmit={handleSubmit}
  className="flex flex-col justify-center px-10"
>
  <label htmlFor="" className="">
    Book Image:
  </label>
  <input
    type="text"
    value={formData.image}
    onChange={handleChange}
    className=" w-full bg-white rounded-md p-1 mb-3 text-black"
    name="image"
    placeholder="Image Url"
  />
  {errors.image && (
    <p className="text-red-400 text-sm">{errors.image}</p>
  )}
  <label htmlFor="">Book Title:</label>
  <input
    type="text"
    value={formData.title}
    onChange={handleChange}
    className=" w-full bg-white rounded-md p-1 mb-3 text-black"
    name="title"
    placeholder="Enter Book Title"
  />
  {errors.title && (
    <p className="text-red-400 text-sm">{errors.title}</p>
  )}

  <label htmlFor="">Book Author:</label>
  <input
    type="text"
    value={formData.author}
```

```
  onChange={handleChange}
  className=" w-full bg-white rounded-md p-1 mb-3 text-black"
  name="author"
  placeholder="Enter Book Author"
/>
{errors.author && (
  <p className="text-red-400 text-sm">{errors.author}</p>
)}
<label htmlFor="">Book Rating:</label>
<input
  type="text"
  value={formData.rating}
  onChange={handleChange}
  className=" w-full bg-white rounded-md p-1 mb-3 text-black"
  name="rating"
  placeholder="Enter Book Rating"
/>
{errors.rating && (
  <p className="text-red-400 text-sm">{errors.rating}</p>
)}
<label htmlFor="">Book Category:</label>
<select
  value={formData.category}
  onChange={handleChange}
  className=" w-full bg-white rounded-md p-1 mb-3 text-black"
  name="category"
>
<option value="">Select a Category</option>
<option value="fiction">Fiction</option>
<option value="non-fiction">Non-Fiction</option>
<option value="sci-fiction">Sci-Fiction</option>
<option value="fantasy">Fantasy</option>
<option value="romance">Romance</option>
</select>
{errors.category && (
  <p className="text-red-400 text-sm">{errors.category}</p>
)}
```

```
          <label htmlFor="">Book Description:</label>
          <textarea
            name="description"
            value={formData.description}
            onChange={handleChange}
            className=" w-full bg-white rounded-md p-1 mb-3 text-black"
            cols="30"
            rows="3"
            placeholder="Enter Book Description"
          ></textarea>
          {errors.description && (
            <p className="text-red-400 text-sm">{errors.description}</p>
          )}

          <button
            type="submit"
            className="w-full bg-blue-500 font-bold rounded-lg p-1 mb-5 hover:bg-blue-900"
          >
            Submit
          </button>
        </form>
      </div>
    </>
  );
}
export default AddBooks;

;
```

Components/BooksDetails.jsx Code:-

```
import { useSelector } from "react-redux";
import { Link, useNavigate, useParams } from "react-router-dom";

function BookDetails() {
  const { id } = useParams();
  const navigate = useNavigate();
  const books = useSelector((state) => state.books.list);
```

```
const book = books.find((b) => b.id === id);

if (!book) {
  return (
    <div className="text-center p-10 text-2xl font-bold text-red-500">
      Book not found.
    </div>
  );
}
return (
  <>
    <h1 className="text-center m-5 p-2 text-2xl underline ">
      <em>{`Book Details with id: ${id}`}</em>
    </h1>
    <div className="m-10 bg-zinc-100 text-black rounded-xl shadow-xl p-6 flex">
      <div className=" w-[30%] flex justify-center items-center ">
        <img
          src={book.image}
          alt={book.title}
          className=" w-80 h-96 object-cover rounded mb-4"
        />
      </div>
      <div className="p-10 w-[70%]">
        <h2 className="text-3xl font-bold mb-4 text-center">{book.title}</h2>
        <p className="mb-2 text-lg">
          <strong>Author:</strong> {book.author}
        </p>
        <p className="mb-2 text-lg">
          <strong>Description:</strong> {book.description}
        </p>
        <p className="mb-4 text-lg">
          <strong>Rating:</strong> {book.rating || "N/A"}
        </p>

        <div className="text-center">
          <Link to="/browse-books">
            <button className="bg-zinc-600 hover:bg-zinc-800 text-white px-6 py-2 rounded-md">
              Back to Browse
```

```
            </button>
          </Link>
        </div>
      </div>
    </div>
  </>
 );
}
export default BookDetails;
```

Components/BooksByCategory.jsx Code:-

```
import { useSelector } from "react-redux";
import { Link, useParams } from "react-router-dom";

function BooksByCategory() {
 const { category } = useParams();
 const books = useSelector((state) => state.books.list);

 const filtered = books.filter(
  (book) => book.category.toLowerCase() === category.toLowerCase()
 );

 return (
  <>
   <div className="p-10">
    <h1 className="text-2xl font-bold mb-6 text-center">
     Books in <em className="text-blue-500 ">"{category}"</em> Category
    </h1>

    {filtered.length === 0 ? (
     <p className="text-center m-20 p-20 font-semibold">
      No books found in this category!
     </p>
    ) : (
     <div className="grid grid-cols-3">
```

```
          {filtered.map((book) => (
            <div
              key={book.id}
              className=" m-2 rounded-2xl p-2 shadow-2xl hover:scale-98 transition duration-200 ease-in"
            >
              <img
                src={book.image}
                alt={book.title}
                className="w-full h-62"
              />
              <h2 className="font-bold text-center">{book.title}</h2>
              <div className="flex justify-between">
                <h3 className="text-zinc-800 my-2">by: {book.author}</h3>
                <h3 className="text-zinc-800 my-2">Rating: {book.rating}</h3>
              </div>
              <p className="mx-1 mb-5">
                {book.description.split(" ").slice(0, 10).join(" ")}...
              </p>
              <Link to={`/book-details/${book.id}`}>
                <button className="w-full bg-zinc-600 hover:bg-zinc-800 text-white rounded-md mb-5 py-1 px-4
">
                  View Details
                </button>
              </Link>
            </div>
          ))}
        </div>
      )}
    </div>
  </>
 );
}
export default BooksByCategory;
```

Components/BrowseBooks.jsx Code:-

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

```
import { useSelector } from "react-redux";
import { Link } from "react-router-dom";

const categories = [
  "Fiction",
  "Non-Fiction",
  "Sci-Fiction",
  "Fantasy",
  "Romance",
];

const BrowseBooks = () => {
  const books = useSelector((state) => state.books.list);
  const [searchText, setSearchText] = useState("");
  const [filteredBooks, setFilteredBooks] = useState(books);

  function handleSearch() {
    console.log(searchText);
    const filterBooks = books.filter(
      (book) =>
        book.title.toLowerCase().includes(searchText.toLowerCase()) ||
        book.author.toLowerCase().includes(searchText.toLowerCase())
    );
    console.log("filtered Books ;", filterBooks);
    setFilteredBooks(filterBooks);
  }

  return (
    <>
      <h2 className="text-center font-bold my-4 p-5 text-3xl text-shadow-lg">
        Browse Books
      </h2>

      {/* //Search Bar  */}
      <div className="flex justify-center items-center gap-5 mb-5">
        <h3 className="font-bold text-xl">Search Books</h3>
        <input
          type="text"
```

```
    name=""
    className="w-[30%] border rounded-md p-1 text-black"
    placeholder="Enter a title/author of Book to Search..."
    onChange={(e) => {
      setSearchText(e.target.value);
    }}
  />
  <button
    className="bg-zinc-600 hover:bg-zinc-800 text-white px-5 py-1 rounded-md"
    onClick={handleSearch}
  >
    Search
  </button>
</div>

{/* Grouped by Category */}
{categories.map((category) => {
  const booksInCategory = filteredBooks.filter(
    (book) => book.category?.toLowerCase() === category.toLowerCase()
  );

  if (booksInCategory.length === 0) {
    return null;
  }

  return (
    <div key={category} className="mb-10">
      <h2 className="text-2xl text-center font-bold mb-4 px-5">
        {category}
      </h2>
      <div className="grid grid-cols-4 gap-6 px-5">
        {booksInCategory.map((book) => (
          <div
            key={books.id}
            className=" m-2 rounded-2xl p-2 shadow-2xl hover:scale-98 transition duration-200 ease-in"
          >
            <img
              src={book.image}
```

```
              alt={book.title}
              className="w-full h-62"
            />
            <h2 className="font-bold text-center">{book.title}</h2>
            <div className="flex justify-between">
              <h3 className="text-zinc-800 my-2">by: {book.author}</h3>
              <h3 className="text-zinc-800 my-2">
                Rating: {book.rating}
              </h3>
            </div>
            <p className="mx-1 mb-5">
              {book.description.split(" ").slice(0, 10).join(" ")}...
            </p>
            <Link to={`/book-details/${book.id}`}>
              <button className="w-full bg-zinc-600 hover:bg-zinc-800 text-white rounded-md mb-5 py-1 px-
4 ">
                View Details
              </button>
            </Link>
          </div>
        ))}
      </div>
    </div>
  );
})}

{/* No Books Fallback */}
{books.length === 0 && (
  <div className="text-center my-10">
    <h1 className="text-2xl font-bold mb-2">No Books Available</h1>
    <p>
      Add a Book from{" "}
      <Link to="/add-books" className="text-blue-400 underline">
        Add Book
      </Link>
    </p>
  </div>
)}
```

```
      <hr className="mb-30 text-white" />
    </>
  );
};
export default BrowseBooks;
```

Components/Error.jsx Code:-

```
import { useRouteError } from "react-router-dom";

function Error() {
  const error = useRouteError();
  //   console.error(error);
  return (
    <>
      <div className="h-screen flex flex-col justify-center items-center gap-5">
        <h1 className="text-red-700 text-5xl font-bold">Oops!</h1>
        <h1 className="text-xl">
          {error.status}--{error.statusText}
        </h1>
        <h2>{error.data}</h2>
        <p>Sorry, the page you're looking for does not exist.</p>
        <button className="bg-blue-500 hover:bg-blue-700 px-4 py-2 rounded-xl font-bold">
          <a href="/"> HomePage</a>
        </button>
      </div>
    </>
  );
}
export default Error;
```

Components/Header.jsx Code:-

```
function Header() {
  return (
    <>
```

```
      <div className="header bg-zinc-700 p-2 text-white flex justify-between items-center">
        <h1 className="mx-6 font-bold text-2xl text-blue-800 text-shadow-lg">
          Online Library System
        </h1>
        <nav className="navbar mx-8">
          <ul className="nav-list flex gap-6 text-lg">
            <li className="hover:text-blue-800 transition duration-300 ease-in-out">
              <a href="/">Home</a>
            </li>
            <li className="hover:text-blue-800 transition duration-300 ease-in-out">
              <a href="/browse-books">Browse Books</a>
            </li>
            <li className="hover:text-blue-800 transition duration-300 ease-in-out">
              <a href="/add-books">Add Book</a>
            </li>
          </ul>
        </nav>
      </div>
    </>
  );
}
export default Header;
```

Components/Home.jsx Code:-

```
import { useSelector } from "react-redux";
import { Link } from "react-router-dom";

const categories = [
  "Fiction",
  "Non-Fiction",
  "Sci-Fiction",
  "Fantasy",
  "Romance",
];
```

```
function Home() {
  const books = useSelector((state) => state.books.list);

  const popularBooks = books.slice(-4).reverse();

  return (
    <>
      <div className=" p-10">
        <h1 className="text-3xl font-bold text-center mb-6 text-shadow-lg">
          Welcome to Book Library
        </h1>
        <p className="text-center mb-10 text-lg">
          Explore books by category or browse our popular selections!
        </p>

        {/* Book Categories */}
        <div className="text-center p-5">
          <h2 className="text-2xl font-bold mb-6">Categories</h2>
          <div className="flex justify-center items-center flex-wrap gap-10 mb-4">
            {categories.map((category, index) => (
              <Link
                to={`/books/${category.toLowerCase()}`}
                key={index}
                className="bg-blue-600 hover:bg-blue-800 px-4 py-2 rounded-lg text-white"
              >
                {category}
              </Link>
            ))}
          </div>
        </div>

        {/* Popular Books */}
        <h2 className="text-2xl font-bold mb-4 text-center p-5">
          Popular Books
        </h2>
        <div>
          {popularBooks.length === 0 ? (
            <div className="text-center m-10 p-10 ">
```

```
            <p className=" text-xl font-semibold mb-4">No Books Added Yet.</p>
          </div>
        ) : (
          <div className="grid grid-cols-4">
            {popularBooks.map((book) => (
              <div
                key={book.id}
                className=" m-2 rounded-2xl p-2 shadow-2xl hover:scale-98 transition duration-200 ease-in"
              >
                <img
                  src={book.image}
                  alt={book.title}
                  className="w-full h-62"
                />
                <h2 className="font-bold text-center">{book.title}</h2>
                <div className="flex justify-between">
                  <h3 className="text-zinc-800 my-2">by: {book.author}</h3>
                  <h3 className="text-zinc-800 my-2">
                    Rating: {book.rating}
                  </h3>
                </div>
                <p className="mx-1 mb-5">
                  {book.description.split(" ").slice(0, 10).join(" ")}...
                </p>
                <Link to={`/book-details/${book.id}`}>
                  <button className="w-full bg-zinc-600 hover:bg-zinc-800 text-white rounded-md mb-5 py-1 px-
4 ">
                    View Details
                  </button>
                </Link>
              </div>
            ))}
          </div>
        )}
      </div>
      <hr className="mb-12 text-white" />
    </div>
  </>
```

```
  );
}
export default Home;
```

Components/booksSlice.js Code:-

```
import { createSlice } from "@reduxjs/toolkit";

const hardcodedBooks = [
  {
    id: "1",
    title: "The Great Gatsby",
    author: "F. Scott Fitzgerald",
    description: `The Great Gatsby is a 1925 novel by American writer F. Scott Fitzgerald. The novel is set in the Jazz Age on Long Island, near New York City, and depicts first-person narrator Nick Carraway's interactions with mysterious millionaire Jay Gatsby and Gatsby's obsession to reunite with his former lover, Daisy Buchanan. The novel presents a critical portrait of the American dream through its portrayal of the 1920s New York elite, exploring themes of wealth, class, love, and idealism.`,
    image: "https://images.unsplash.com/photo-1544716278-ca5e3f4abd8c",
    rating: 4.5,
    category: "Fiction",
  }, ////Hardcoded Data
  {
    id: "2",
    title: "A Brief History of Time",
    author: "Stephen Hawking",
    description:
      "An overview of cosmology from the Big Bang to black holes. Hawking writes in non-technical terms about the structure, origin, development and eventual fate of the universe. He talks about basic concepts like space and time, building blocks that make up the universe (such as quarks) and the fundamental forces that govern it (such as gravity). He discusses two theories, general relativity and quantum mechanics that form the foundation of modern physics. Finally, he talks about the search for a unified theory that describes everything in the universe in consistently.",
    image: "https://images.unsplash.com/photo-1512820790803-83ca734da794",
    rating: 4.8,
    category: "Non-Fiction",
  }, ////Hardcoded Data
  {
    id: "3",
    title: "The Pragmatic Programmer",
```

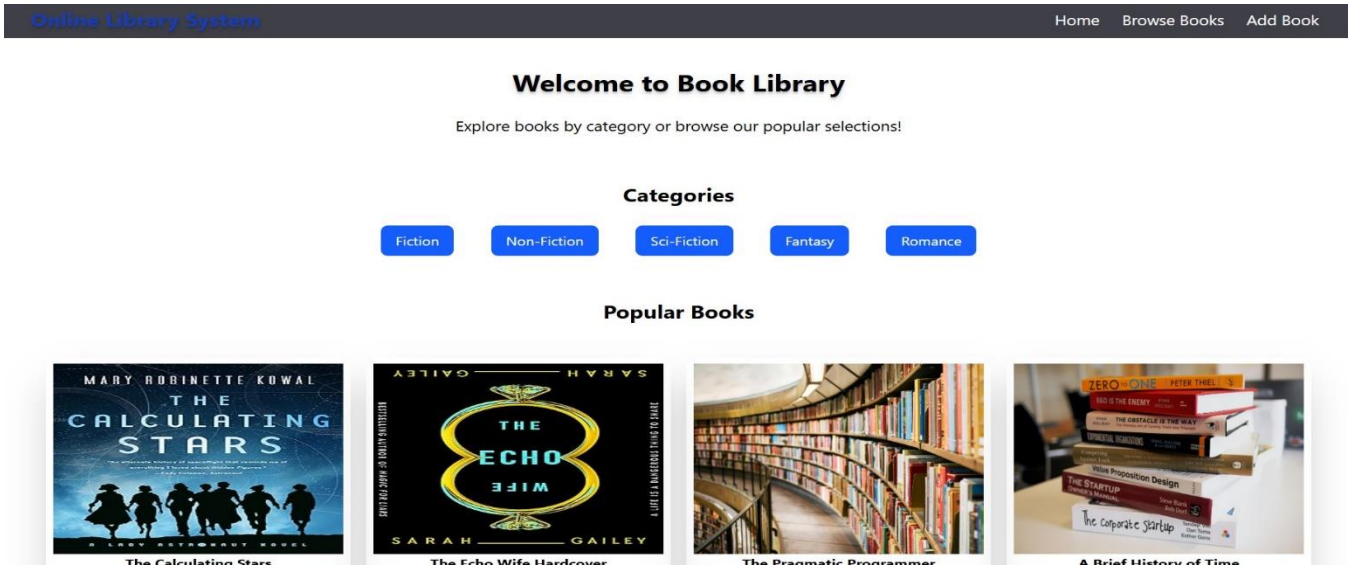
```
    author: "Andrew Hunt",
    description:
      "The Pragmatic Programmer: From Journeyman to Master is a book about computer programming and software engineering, written by Andrew Hunt and David Thomas and published in October 1999.[1][2][3] It is used as a textbook in related university courses.[4] It was the first in a series of books under the label The Pragmatic Bookshelf. A second edition, The Pragmatic Programmer: Your Journey to Mastery was released in 2019 for the book's 20th anniversary, with major revisions and new material which reflects new technology and other changes in the software engineering industry over the last twenty years",
    image: "https://images.unsplash.com/photo-1524995997946-a1c2e315a42f",
    rating: 4.7,
    category: "Fantasy",
  }, ////Hardcoded Data
  {
    id: "4",
    title: "The Echo Wife Hardcover",
    author: " Sarah Gailey",
    description:
      "Sarah Gailey's The Echo Wife is “a trippy domestic thriller which takes the extramarital affair trope in some intriguingly weird new directions.”--Entertainment Weekly\n\nI’m embarrassed, still, by how long it took me to notice. Everything was right there in the open, right there in front of me, but it still took me so long to see the person I had married.\n\nIt took me so long to hate him.\n\nMartine is a genetically cloned replica made from Evelyn Caldwell’s award-winning research. She’s patient and gentle and obedient. She’s everything Evelyn swore she’d never be.",
    image: "https://m.media-amazon.com/images/I/810MVy3iDPL._SY385_.jpg",
    rating: "4",
    category: "Sci-Fiction",
  }, ////Hardcoded Data
];


const loadUserBooks = () => {
  const saved = localStorage.getItem("userBooks");
  return saved ? JSON.parse(saved) : [];
}; ////fetch the data from the localStorage.


const booksSlice = createSlice({
  name: "books",
  initialState: {
    list: [...hardcodedBooks, ...loadUserBooks()],
  },
  reducers: {
    addBook: (state, action) => {
      const newBook = {
```

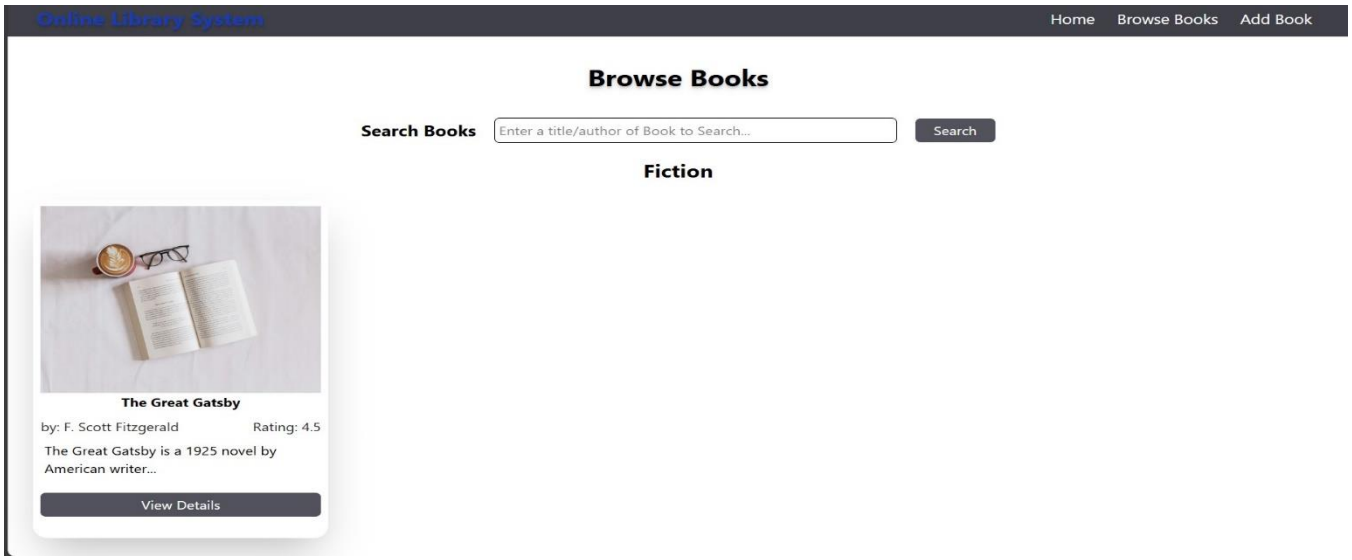
```
        ...action.payload,
        id: Date.now().toString(),
      };
      state.list.push(newBook);

      const userBooks = loadUserBooks(); // only store user Books
      userBooks.push(newBook);
      localStorage.setItem("userBooks", JSON.stringify(userBooks));
    },
  },
  // localStorage.removeItem("books"); // to clear the data from browser
});

export const { addBook } = booksSlice.actions;
export default booksSlice.reducer;
```

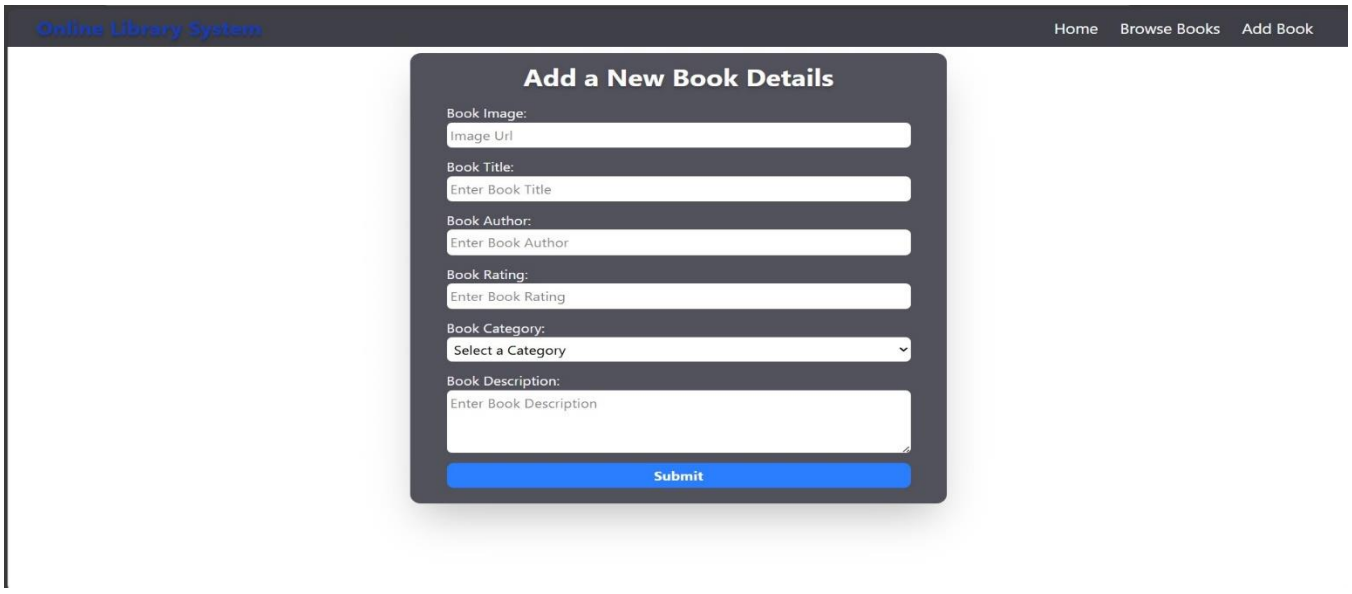
Components/booksStore.js Code:-

```
import { configureStore } from "@reduxjs/toolkit";
import booksReducer from './booksSlice.js'

export const bookStore = configureStore({
  reducer: {
    books: booksReducer,
  },
});
```

2. Screenshot :-







- **GitHub Link:- https://github.com/shashikant021/Online-Library-System**