BookVerse – Day 3 Routing and Backend Integration

This report covers Day 3 enhancements for the 'BookVerse' React-based project. The application integrates routing, backend API fetching using json-server, dynamic book details, and conditional navigation for authors. Day 3 introduced backend integration and route-based navigation, extended functionality by adding author-specific routing and back navigation.

1. Project Overview

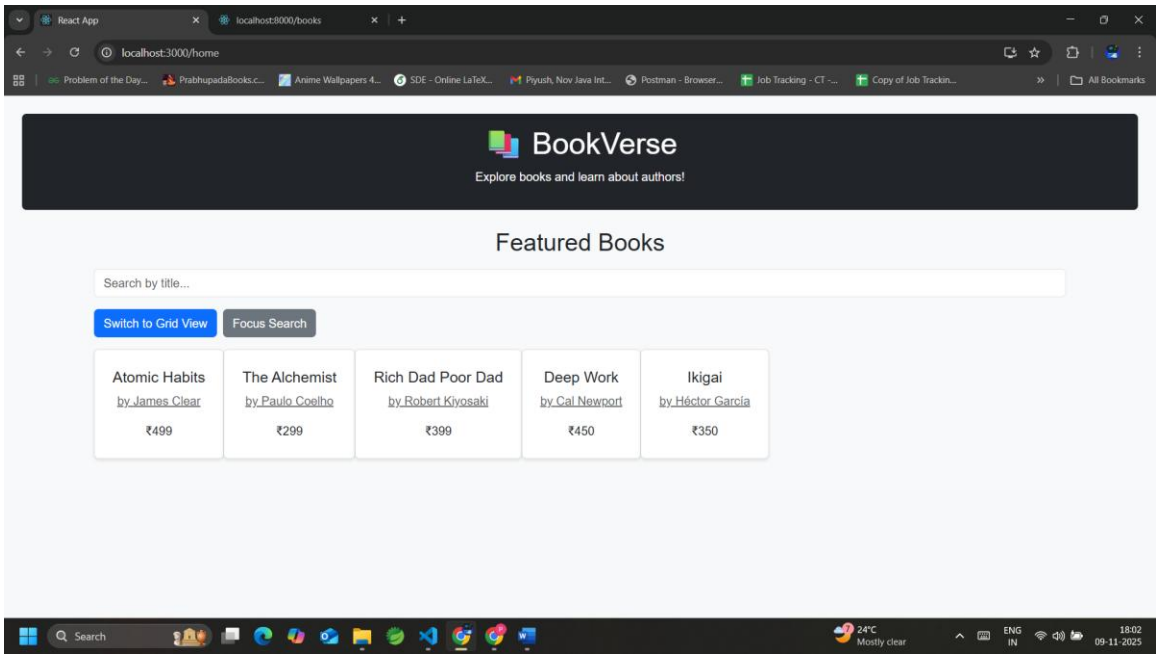
Title: BookVerse – React Routing & Backend Integration with Conditional Navigation

Objectives:

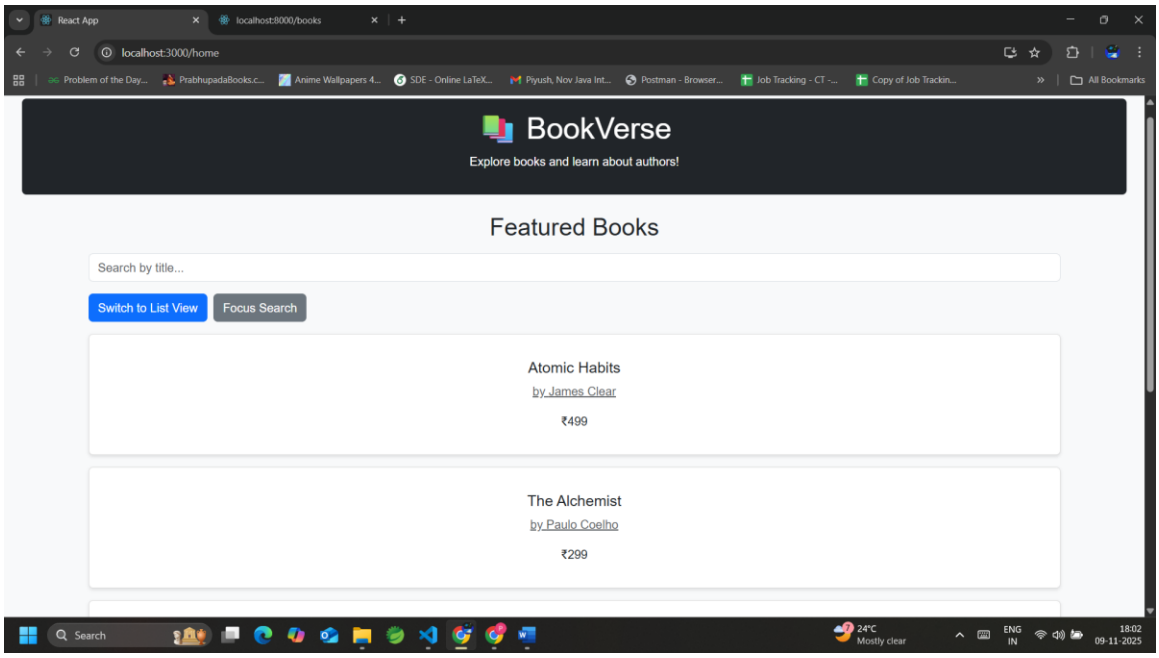
- Fetch book data dynamically from a backend server (json-server)
- Display Book Details and Author Details using React Router
- Implement loading states using Higher Order Components (HOCs)
- Add conditional navigation for AuthorInfo and BookDetails based on click targets
- Provide smooth transitions and clear user experience

2. Output Screenshots (Placeholders)

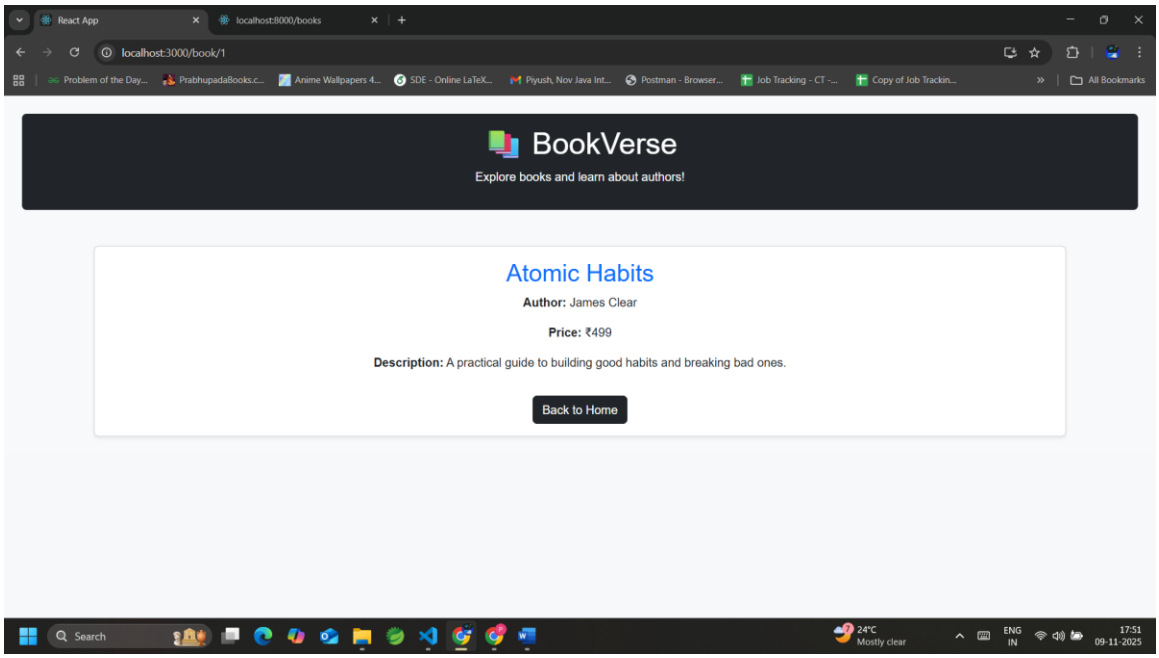
[Screenshot 1: Home Page – Book List Grid View Placeholder]



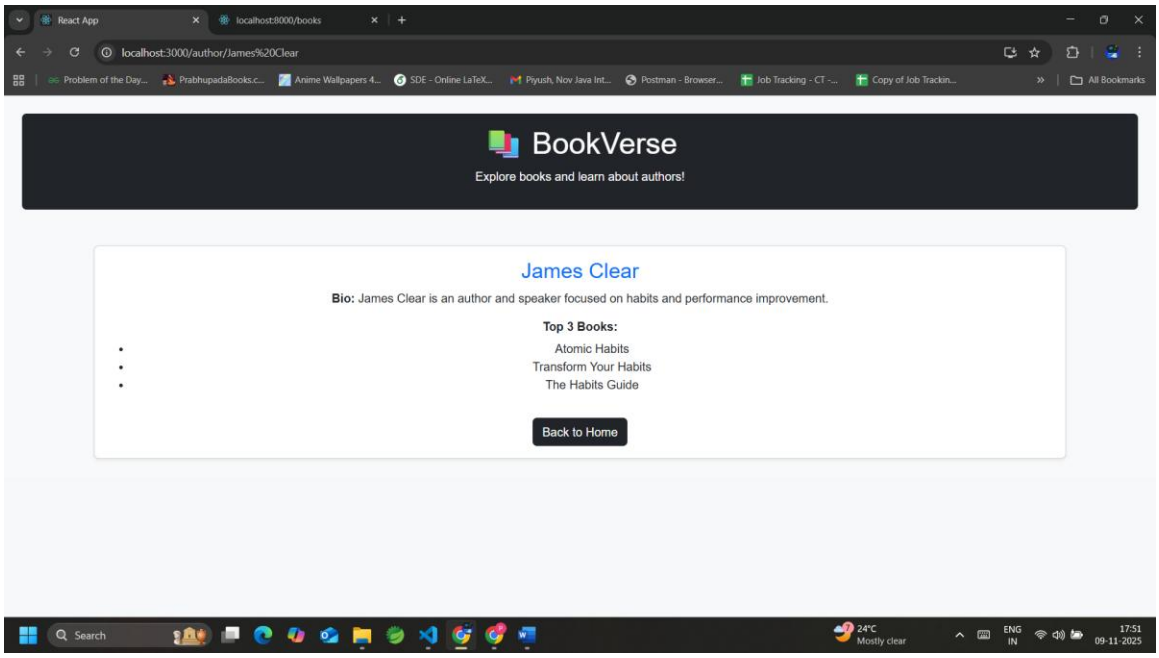
[Screenshot 2: Home Page – List View Placeholder]



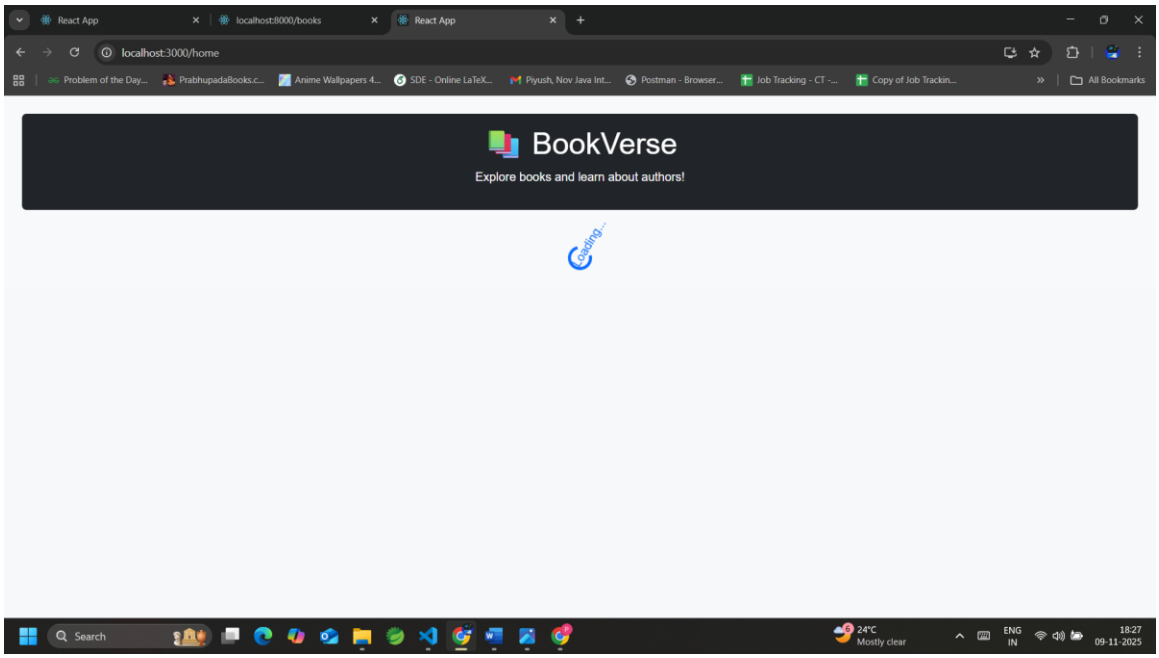
[Screenshot 3: Book Details Page Placeholder]



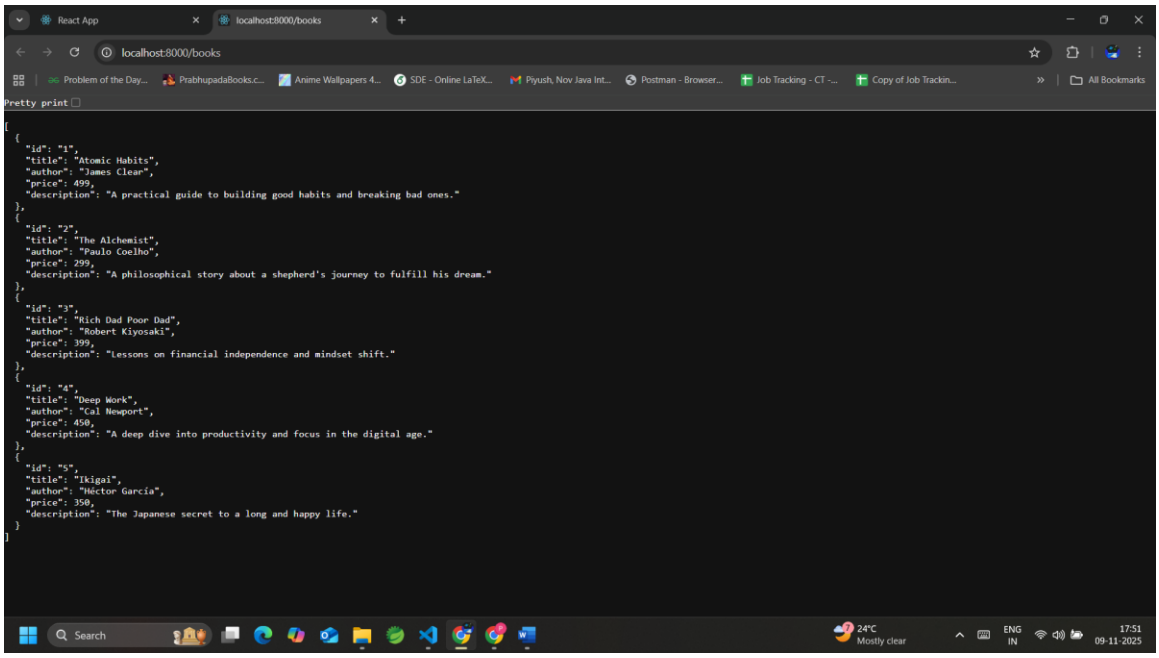
[Screenshot 4: Author Info Page with Back Button Placeholder]



[Screenshot 5: Routing Transition Animation Placeholder]



[Screenshot 6: books.json on json-server]



3. Components and Updated Code Snippets

3.1 App.jsx (Routing Setup)

This component defines the routing structure for the entire application, including routes for Home (BookList), BookDetails, and AuthorInfo. It also manages loading state passed to components.

```
// App.jsx
// Adds routing between Home (BookList) and Book Details page with fade
transition

import React, { useState } from "react";
import { BrowserRouter as Router, Routes, Route, Navigate } from "react-
router-dom";
import BookList from "./components/BookList";
import BookDetails from "./components/BookDetails";
import AuthorInfo from "./components/AuthorInfo";
import "bootstrap/dist/css/bootstrap.min.css";
import "./App.css";

const App = () => {
  const [loading, setLoading] = useState(false);

  return (
    <Router>
      <div className="App bg-light p-4 fade-in">
        <header className="bg-dark text-white p-3 rounded mb-4">
```

```
          <h1>📚 BookVerse</h1>
          <p>Explore books and learn about authors!</p>
        </header>

        <Routes>
          <Route path="/" element={<Navigate to="/home" />} />
          <Route path="/home" element={<BookList loading={loading}
setLoading={setLoading} />} />
          <Route path="/book/:id" element={<BookDetails loading={loading}
setLoading={setLoading} />} />
          <Route path="/author/:authorName" element={<AuthorInfo />} />
        </Routes>
      </div>
    </Router>
  );
};

export default App;
```

3.2 BookCard.jsx (Conditional Navigation)

BookCard handles display of individual book information. The component differentiates between clicks on the author name (navigating to AuthorInfo) and clicks elsewhere on the card (navigating to BookDetails).

```
// BookCard.jsx
import React from "react";
import PropTypes from "prop-types";
import { useNavigate } from "react-router-dom";

const BookCard = ({ id, title, author, price, layout }) => {
    const navigate = useNavigate();

    // Navigate to book details page
    const handleCardClick = () => navigate(`/book/${id}`);

    // Navigate to author info page (stop card click)
    const handleAuthorClick = (e) => {
        e.stopPropagation(); // prevent card navigation
        navigate(`/author/${encodeURIComponent(author)}`);
    };

    return (
        <div
```

```
            className={`book-card card mb-3 ${layout === "grid" ? "p-3" : 
"p-2"}`}
            onClick={handleCardClick}
            style={{ cursor: "pointer" }}
        >
            <div className="card-body">
                <h5 className="card-title">{title}</h5>
                <p
                    className="card-text text-muted"
                    onClick={handleAuthorClick}
                    style={{ color: "#007bff", textDecoration: 
"underline", cursor: "pointer" }}
                >
                    by {author}
                </p>
                <p className="card-text">₹{price}</p>
            </div>
        </div>
    );
};

BookCard.propTypes = {
    id: PropTypes.number.isRequired,
    title: PropTypes.string.isRequired,
    author: PropTypes.string.isRequired,
    price: PropTypes.number.isRequired,
    layout: PropTypes.string.isRequired,
};

export default BookCard;
```

3.3 BookList.jsx (Book Display & Search)

BookList fetches book data from the backend API, handles layout toggling (grid/list), and integrates a search feature. It renders BookCards for each book, passing data as props.

[Code Snippet Placeholder – BookList.jsx API Fetch and Map Logic]

3.4 BookDetails.jsx (Book Information Page)

BookDetails fetches detailed information for a specific book using its ID from the route. It uses loading states to show a spinner during API calls, and displays title, author, price, and description.

```
import React, { useState, useEffect, useRef } from "react";
import BookCard from "./BookCard";
import SearchBox from "./SearchBox";
```

```
import AuthorInfo from "./AuthorInfo";
import RenderMessage from "./RenderMessage";
import { Link } from "react-router-dom";
import withLoader from "./withLoader";
import "bootstrap/dist/css/bootstrap.min.css";

const BookList = ({ loading, setLoading }) => {
    const [books, setBooks] = useState([]);
    const [layout, setLayout] = useState("grid");
    const [searchTerm, setSearchTerm] = useState("");
    const [selectedAuthor, setSelectedAuthor] = useState(null);
    const searchRef = useRef(null);

    useEffect(() => {
        setLoading(true);
        fetch("http://localhost:8000/books")
            .then((res) => res.json())
            .then((data) => {
                setBooks(data);     // not data.books ❌
                setLoading(false);
            });
    }, [setLoading]);

    const toggleLayout = () => setLayout(layout === "grid" ? "list" :
"grid");
    const focusSearch = () => searchRef.current &&
searchRef.current.focus();

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

    return (
        <div className="container">
            <h2 className="mt-4">Featured Books</h2>
            <SearchBox searchTerm={searchTerm}
setSearchTerm={setSearchTerm} inputRef={searchRef} />

            <div className="d-flex gap-2 mb-3">
                <button className="btn btn-primary"
onClick={toggleLayout}>
                    Switch to {layout === "grid" ? "List" : "Grid"} View
                </button>
                <button className="btn btn-secondary"
onClick={focusSearch}>Focus Search</button>
```

```
            </div>

            {filteredBooks.length === 0 && (
                <RenderMessage render={() => "No books found. Try a
different search!"} />
            )}

            <div className={`book-list d-${layout === "grid" ? "grid" :
"flex"} flex-wrap`}>
                {filteredBooks.map((book) => (
                    <BookCard
                        key={book.id}
                        id={book.id}
                        title={book.title}
                        author={book.author}
                        price={book.price}
                        layout={layout}
                    />
                ))}
            </div>

            {/* <AuthorInfo author={selectedAuthor} /> */}
        </div>
    );
};

export default withLoader(BookList);
```

3.5 AuthorInfo.jsx (Author Page with Back Button)

AuthorInfo retrieves author data from local JSON or static data file based on the author's name passed in the URL. A 'Back to Home' button enables easy navigation back to the main list.

```
// AuthorInfo.jsx
import React, { Component } from "react";
import { useParams, useNavigate } from "react-router-dom";
import authorsData from "../data/authorsData";
import "bootstrap/dist/css/bootstrap.min.css";

/* Wrapper so class component can use hooks */
function AuthorInfoWrapper() {
    const { authorName } = useParams();
    const navigate = useNavigate();
    return <AuthorInfo authorName={authorName} navigate={navigate} />;
```

```
}

class AuthorInfo extends Component {
    constructor(props) {
        super(props);
        this.state = { author: null };
    }

    componentDidMount() {
        const { authorName } = this.props;
        if (authorName) {
            this.loadAuthorInfo(decodeURIComponent(authorName));
        }
    }

    loadAuthorInfo = (authorName) => {
        const info = authorsData[authorName] || null;
        this.setState({ author: info });
    };

    render() {
        const { author } = this.state;
        const { navigate } = this.props;

        if (!author) {
            return (
                <div className="alert alert-warning mt-5 text-center">
                    Author not found.
                    <br />
                    <button className="btn btn-secondary mt-3" onClick={()
=> navigate("/home")}>
                        Back to Home
                    </button>
                </div>
            );
        }

        return (
            <div className="container mt-5">
                <div className="card shadow-sm">
                    <div className="card-body">
                        <h3 className="card-title text-
primary">{decodeURIComponent(this.props.authorName)}</h3>
                        <p><strong>Bio:</strong> {author.bio}</p>
                        <h6><strong>Top 3 Books:</strong></h6>
```

```
                        <ul>
                            {author.topBooks.map((book, idx) => (
                                <li key={idx}>{book}</li>
                            ))}
                        </ul>
                        <button className="btn btn-dark mt-3" onClick={()
=> navigate("/home")}>
                            Back to Home
                        </button>
                    </div>
                </div>
            </div>
        );
    }
}

export default AuthorInfoWrapper;
```

3.6 withLoader.jsx (Higher Order Component for Loading State)

This Higher Order Component (HOC) wraps around other components to show a loading spinner while data is being fetched from the API. It improves reusability and consistency of UI feedback.

```
// withLoader.jsx
// Higher Order Component to show a loading spinner during data fetch

import React from "react";

const withLoader = (WrappedComponent) => {
    return function WithLoaderComponent({ loading, ...props }) {
        if (loading) {
            return (
                <div className="text-center mt-5">
                    <div className="spinner-border text-primary"
role="status">
                        <span className="sr-only">Loading...</span>
                    </div>
                </div>
            );
        }
        return <WrappedComponent {...props} />;
    };
};
```

```
export default withLoader;
```

3.7 RenderMessage.jsx (Reusable Message Renderer)

RenderMessage is a simple utility component that renders a custom message dynamically through a function passed as a prop. It is used to show fallback messages such as 'No books found'.

```
// RenderMessage.jsx
// Render Props component to display dynamic UI content

import React from "react";

const RenderMessage = ({ render }) => {
    return <div className="alert alert-info text-center mt-
3">{render()}</div>;
};

export default RenderMessage;
```

4. Summary

The Day 3 implementation completes the major frontend functional flow of BookVerse. Users can browse books, search and filter them, view detailed pages for each book, view author profiles, and return home seamlessly. Backend integration ensures scalability, while structured routing and loading management enhance usability and performance.