# Library Management System

The Library Management System is a MERN (MongoDB, Express.js, React.js, Node.js) stack application designed to manage a library's book inventory and member information. This document provides an overview of the application, its features, how to run it, test the APIs, and interact with the frontend.

## Features:

### 1. Backend API Features:

- Add books to the library
- Add members to the library
- Allow members to borrow and return books
- Retrieve all available books with their availability status
- Retrieve all registered members
- Modify Member data

### 2. Frontend Features:

- View and search for books
- View and manage members (register new members, view member details)
- Functionality for members to borrow and return books

## Technology Stack:

| Frontend | Backend | Testing |
|---|---|---|
| React.js<br>Packages used(react,react-router-dom,axios) | Node.js<br>Express.js<br>MongoDB<br>Packages used(express,cors,mongoose) | Jest<br>Supertest<br>Packages Used(Jest,Supertest) |

# API Endpoints:

In this project I worked on the several API endpoints with the port number of 8000.

- “/login/userverfication”- Login authentication
- “/login/signup” - Registration member
- “/login/admin” - Admin signin
- “/login/userverification/books” - Show the list of books
- “/login/userverfication/books/edit/:id”- Update book detail
- “/login/userverfication/books/member/edit/:id”- Update member detail
- “/login/userverfication/books/member/delete” - delete member detail

# Frontend files:

### App.js

```
import React,{useState} from 'react';
import LoginPage from "./Components/login";
import Register from "./Components/registration";
import HomePage from "./Components/home";
import Book from "./Components/book";
import Adminlogin from './Components/adminlogin';
import AdminView from './Components/adminview';
import { BrowserRouter,Routes,Route } from 'react-router-dom';
import Edit from './Components/edit';

function App(){
  return(
    <BrowserRouter>
     <Routes>
      <Route path="/" element={<LoginPage/>}></Route>
        <Route path="/registration" element={<Register/>}></Route>
         <Route path="/home" element={<HomePage/>}></Route>
```

```
        <Route path="/book" element={<Book/>}></Route>
        <Route path="/adminlogin" element={<Adminlogin/>}></Route>
        <Route path="/adminview" element={<AdminView/>}></Route>
        <Route path="/edit" element={<Edit/>}></Route>
    </Routes>
  </BrowserRouter>
 )
}
export default App;
```

**Login.js**

/Components/login.js

```
import React, { useState } from "react";
import axios from "axios";
import { Link, useNavigate } from "react-router-dom";
import 'bootstrap/dist/css/bootstrap.min.css';
import 'bootstrap/dist/js/bootstrap.bundle.min.js';


function LoginPage() {
   const [name, setName] = useState("");
   const [email, setEmail] = useState("");
   const navigate = useNavigate();

   const fetchData = async (e) => {
      e.preventDefault();
      try {
         const response = await
axios.post("http://localhost:8000/login/userverfication", { name, email });
         localStorage.setItem('userData', JSON.stringify(response.data));
         navigate("/home");
      } catch (error) {
         console.error("An error occurred:", error);
      }
   }

   return (
      <div className="container d-flex justify-content-center align-
items-center vh-100">
         <div className="card p-4">
```

```
            <h1 className="text-center mb-4">Sign In Scientist</h1>
            <form onSubmit={fetchData}>
              <div className="mb-3">
                <input type="text" className="form-control"
name="name" placeholder="Username" value={name} onChange={(e)
=> setName(e.target.value)} />
              </div>
              <div className="mb-3">
                <input type="email" className="form-control"
name="email" placeholder="Email" value={email} onChange={(e) =>
setEmail(e.target.value)} />
              </div>
              <div className="mb-3">
                <button type="submit" className="btn btn-primary w-
100">Login</button>
              </div>
            </form>
            <div className="text-center">
              <h3>Create the New Generation Let Start With US</h3>
              <Link to="/registration" className="btn btn-success me-
2">Sign Up</Link>
              <Link to="/adminlogin" className="btn btn-info">Admin
Sign In</Link>
            </div>
          </div>
        </div>
    )
}

export default LoginPage;
```

**Home.js**

Components/home.js

```
import React, { useState, useEffect } from "react";
import axios from "axios";
import { Link, useLocation, useNavigate } from "react-router-dom";

function HomePage() {
  const [userData, setUserData] = useState(null);
  const [books, setBooks] = useState([]);
```

```
    const [searchTerm, setSearchTerm] = useState("");
    const [filteredBooks, setFilteredBooks] = useState([]);
    const navigate = useNavigate();

    useEffect(() => {
        const storedUserData = localStorage.getItem('userData');
        if (storedUserData) {
            setUserData(JSON.parse(storedUserData));
        }
    }, []);

    useEffect(() => {
        const fetchData = async () => {
            try {
                const response = await
axios.get("http://localhost:8000/login/userverfication/books");
                setBooks(response.data);
            } catch (error) {
                console.error("Error fetching books:", error);
            }
        };

        fetchData();
    }, []);

    useEffect(() => {
        const filtered = books.filter(book =>
            book.title.toLowerCase().includes(searchTerm.toLowerCase()) ||
            book.author.toLowerCase().includes(searchTerm.toLowerCase())
||
            book.genre.toLowerCase().includes(searchTerm.toLowerCase())
        );
        setFilteredBooks(filtered);
    }, [searchTerm, books]);

    const handleSearch = (e) => {
        setSearchTerm(e.target.value);
    };

    const handleBookClick = (book) => {
        navigate("/book", { state: { bookData: book } });
    };
```

```
  return (
    <div>
      <nav style={{ display: 'flex', justifyContent: 'space-between',
alignItems: 'center', padding: '1rem', backgroundColor: '#f8f9fa' }}>
        <div>
          <span>Welcome Back {userData ? userData.name :
"Guest"}</span>
        </div>
        <div style={{ flexGrow: 1, textAlign: 'center' }}>
          <input type="text" placeholder="Search..."
value={searchTerm} onChange={handleSearch} style={{ padding:
'0.5rem', borderRadius: '5px', border: 'none' }} />
          <button style={{ marginLeft: '1rem', padding: '0.5rem 1rem',
borderRadius: '5px', border: 'none', backgroundColor: '#007bff', color:
'#fff', cursor: 'pointer' }}>Search</button>
        </div>
        <div>
          <Link to="/"><button style={{ padding: '0.5rem 1rem',
borderRadius: '5px', border: 'none', backgroundColor: '#007bff', color:
'#fff', cursor: 'pointer' }}>LogOut</button></Link>
        </div>
      </nav>
      <main>
        <div className="card-container" style={{ display: 'flex',
flexWrap: 'wrap', gap: '1rem', padding: '1rem' }}>
          {filteredBooks.map((book) => (
            <div key={book.id} style={{ border: '1px solid #ccc',
borderRadius: '5px', padding: '1rem', width: '320px', minHeight: '200px',
cursor: 'pointer',textAlign:"center" }} onClick={() =>
handleBookClick(book)}>
              <h3>{book.title}</h3>
              <p>Author: {book.author}</p>
              <p>Genre: {book.genre}</p>
              <p>Availability: {book.availability_status}</p>
            </div>
          ))}
        </div>
      </main>
    </div>
  );
}
```

```
export default HomePage;
```

**Book.js**

Components/book.js

```
import React, { useState, useEffect } from "react";
import axios from "axios";
import { Link, useLocation, useNavigate } from "react-router-dom";

function Book() {
   const [userData, setUserData] = useState(null);
   const location = useLocation();
   const bookData = location.state ? location.state.bookData : null;
   const [isAvailable, setIsAvailable] = useState(false);
   const navigate = useNavigate();

   useEffect(() => {
      const storedUserData = localStorage.getItem('userData');
      if (storedUserData) {
         setUserData(JSON.parse(storedUserData));
      }

      if (bookData) {
         setIsAvailable(bookData.availability_status === "available");
      }
   }, [bookData]);

   const handlePurchase = async () => {
      try {
         if (!bookData) {
            throw new Error("No book data found");
         }
         const storedUserData = localStorage.getItem('userData');
         if (!storedUserData) {
            throw new Error("User data not found in local storage");
         }
         const userData = JSON.parse(storedUserData);
         await
axios.put(`http://localhost:8000/login/userverfication/member/edit/${user
Data._id}`, {
               $push: { books_borrowed: bookData }
```

```
            });

            await
axios.post(`http://localhost:8000/login/userverfication/books/edit/${book
Data._id}`, { availability_status: "Not Available" });

            navigate("/home");
        } catch (error) {
            console.error("Error purchasing book:", error);
        }
    };

    const handleReturn = async () => {
        try {
            if (!bookData) {
                throw new Error("No book data found");
            }

            const storedUserData = localStorage.getItem('userData');
            if (!storedUserData) {
                throw new Error("User data not found in local storage");
            }
            const userData = JSON.parse(storedUserData);

            const updatedBooksBorrowed =
userData.books_borrowed.filter(book => book._id !== bookData._id);
            await
axios.put(`http://localhost:8000/login/userverfication/member/edit/${user
Data._id}`, {
                books_borrowed: updatedBooksBorrowed
            });


            await
axios.post(`http://localhost:8000/login/userverfication/books/edit/${book
Data._id}`, { availability_status: "available" });


            const response = await
axios.get(`http://localhost:8000/login/userverfication/member/${userData.
_id}`);
            const updatedUserData = response.data;
```

```
            localStorage.setItem('userData', JSON.stringify(updatedUserData));
            setUserData(updatedUserData);
            navigate("/home");
        } catch (error) {
            console.error("Error returning book:", error);
        }
    };

    if (!bookData) {
        return <div>Loading...</div>;
    }

    return (
        <div style={{ display: "flex", justifyContent: "center", alignItems:
"center", height: "100vh" }}>
            <div style={{ textAlign: "center", border: "1px solid #ccc",
borderRadius: "5px", padding: "2rem" }}>
                <h1>Book Details</h1>
                <p>Title: {bookData.title}</p>
                <p>Author: {bookData.author}</p>
                <p>Genre: {bookData.genre}</p>
                <p>Availability: {bookData.availability_status}</p>
                <div style={{ marginTop: "1rem" }}>
                    {isAvailable ? (
                        <button onClick={handlePurchase} style={{ marginRight:
"1rem" }}>Purchase</button>
                    ) : (
                        <button onClick={handleReturn} style={{ marginRight:
"1rem" }}>Return</button>
                    )}
                    <Link to="/home">
                        <button>Cancel</button>
                    </Link>
                </div>
            </div>
        </div>
    );
}

export default Book;
```

**Register.js**

Components/register.js

```
import React, { useState } from "react";
import axios from "axios";
import { Link, useNavigate } from "react-router-dom";

function Register() {
   const [id, setId] = useState("");
   const [name, setName] = useState("");
   const [email, setEmail] = useState("");
   const navigate = useNavigate();

   const fetchData = async (e) => {
      e.preventDefault();
      try {
         const response = await
axios.post("http://localhost:8000/login/signup", { id, name, email });
         console.log(response.data);
         navigate("/");
      } catch (error) {
         console.error("An error occurred:", error);
      }
   }

   return (
      <div style={{ display: "flex", justifyContent: "center", alignItems:
"center", height: "100vh" }}>
         <div style={{ border: "1px solid #ccc", borderRadius: "5px",
padding: "2rem", width: "400px" }}>
            <h1 style={{ marginBottom: "1rem" }}>Sign up to Create the
New World</h1>
            <form onSubmit={fetchData}>
               <div style={{ marginBottom: "1rem" }}>
                  <input type="text" name="id" placeholder="ID" value={id}
onChange={(e) => setId(e.target.value)} className="form-control" />
               </div>

               <div style={{ marginBottom: "1rem" }}>
```

```
                    <input type="text" name="name"
placeholder="Username" value={name} onChange={(e) =>
setName(e.target.value)} className="form-control" />
                </div>
                <div style={{ marginBottom: "1rem" }}>
                    <input type="email" name="email" placeholder="Email"
value={email} onChange={(e) => setEmail(e.target.value)}
className="form-control" />
                </div>
                <div>
                    <button type="submit" className="btn btn-primary w-
100">Sign Up</button>
                </div>
            </form>
            <div style={{ marginTop: "1rem", textAlign: "center" }}>
                <p>If you already have an account, <Link to="/">Sign
In</Link></p>
            </div>
        </div>
    </div>
  )
}

export default Register;
```

**adminlogin.js**

Components/adminlogin.js

```
import React, { useState } from "react";
import axios from "axios";
import { Link, useNavigate } from "react-router-dom";

export default function Adminlogin() {
  const [name, setName] = useState("");
  const [email, setEmail] = useState("");
  const navigate = useNavigate();

  const fetchData = async (e) => {
    e.preventDefault();
    try {
      const response = await
axios.post("http://localhost:8000/login/admin", { name, email });
```

```
            console.log(response.data);
            navigate("/adminview", { state: { userData: response.data } });
        } catch (error) {
            console.error("An error occurred:", error);
        }
    }

    return (
        <div className="container d-flex justify-content-center align-
items-center vh-100">
            <div className="card p-4">
                <h1 className="text-center mb-4">Sign In Creator</h1>
                <form onSubmit={fetchData}>
                    <div className="mb-3">
                        <input type="text" className="form-control"
name="name" placeholder="Username" value={name} onChange={(e)
=> setName(e.target.value)} />
                    </div>
                    <div className="mb-3">
                        <input type="email" className="form-control"
name="email" placeholder="Email" value={email} onChange={(e) =>
setEmail(e.target.value)} />
                    </div>
                    <div className="mb-3">
                        <button type="submit" className="btn btn-primary w-
100">Login</button>
                    </div>
                </form>
            </div>
        </div>
    )
}
```

**Adminview.js**
Components/adminview.js

```
import React, { useState, useEffect } from "react";
import axios from "axios";
import { Link, useLocation, useNavigate } from "react-router-dom";

export default function AdminView() {
    const location = useLocation();
```

```
    const userData = location.state && location.state.userData;
    const [memberData, setMemberData] = useState(null);
    const navigate = useNavigate();

    useEffect(() => {
        const fetchMemberData = async () => {
            try {
                const response = await
axios.get("http://localhost:8000/login/userverfication/member");
                setMemberData(response.data);
            } catch (error) {
                console.error("Error fetching member data:", error);
            }
        };

        fetchMemberData();
    }, []);

    const handleEdit = (member) => {
        navigate(`/edit`, { state: { memberData: member } });
    };

    const handleDelete = async (memberId) => {
        try {
            await
axios.get(`http://localhost:8000/login/userverfication/member/delete/${m
emberId}`);
            setMemberData(prevMemberData =>
prevMemberData.filter(member => member._id !== memberId));
        } catch (error) {
            console.error("Error deleting member:", error);
        }
    };

    return (
        <div className="container d-flex justify-content-center align-
items-center vh-100">
            <div>
                <nav className="navbar navbar-light bg-light">
                    <span className="navbar-brand">Welcome Back
{userData ? userData.name : "Guest"}</span>
                </nav>
```

```
            <main className="mt-4">
              <h2 className="text-center">Member List</h2>
              <table className="table table-striped">
                <thead>
                  <tr>
                    <th>ID</th>
                    <th>Name</th>
                    <th>Email</th>
                    <th>Action</th>
                  </tr>
                </thead>
                <tbody>
                  {memberData && memberData.map(member => (
                    <tr key={member._id}>
                      <td>{member.id}</td>
                      <td>{member.name}</td>
                      <td>{member.email}</td>
                      <td>
                        <button className="btn btn-primary me-2"
onClick={() => handleEdit(member)}>Edit</button>
                        <button className="btn btn-danger"
onClick={() => handleDelete(member._id)}>Delete</button>
                      </td>
                    </tr>
                  ))}
                </tbody>
              </table>
            </main>
          </div>
        </div>
    );
}
```

**Edit.js**

Components/edit.js

```
import React, { useState } from 'react';
import axios from 'axios';
import { useLocation, useNavigate } from 'react-router-dom';

function Edit() {
  const location = useLocation();
```

```
  const memberData = location.state && location.state.memberData;
  const navigate = useNavigate();

  const [formData, setFormData] = useState({
    id: memberData?.id || '',
    name: memberData?.name || '',
    email: memberData?.email || '',
    books_borrowed: memberData?.books_borrowed.join('\n') || '',
  });

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

  const handleSubmit = async (e) => {
    e.preventDefault();

    try {
      const memberId = memberData._id;

      const booksBorrowedArray =
formData.books_borrowed.split('\n').map(bookId => bookId.trim());
      const updatedData = { ...formData, books_borrowed:
booksBorrowedArray };

      const response = await
axios.put(`http://localhost:8000/login/userverfication/member/edit/${me
mberId}`, updatedData);
      console.log('Member updated:', response.data);
      navigate("/adminview");
    } catch (error) {
      console.error('Error updating member:', error);
    }
  };

  return (
    <div className="container">
      <h2 className="mt-4">Edit Member</h2>
```

```
      <form onSubmit={handleSubmit}>
        <div className="mb-3">
          <label className="form-label">ID:</label>
          <input type="text" className="form-control" name="id"
value={formData.id} onChange={handleChange} />
        </div>
        <div className="mb-3">
          <label className="form-label">Name:</label>
          <input type="text" className="form-control" name="name"
value={formData.name} onChange={handleChange} />
        </div>
        <div className="mb-3">
          <label className="form-label">Email:</label>
          <input type="email" className="form-control" name="email"
value={formData.email} onChange={handleChange} />
        </div>
        <div className="mb-3">
          <label className="form-label">Books Borrowed:</label>
          <textarea className="form-control" name="books_borrowed"
value={formData.books_borrowed} onChange={handleChange}
rows="4" />
        </div>
        <button type="submit" className="btn btn-
primary">Save</button>
      </form>
    </div>
  );
}

export default Edit;
```

## Backend Code

### App.js

```
const express=require("express");
const mongoose=require("./models/db")
const { Book, Member }=require("./models/schema")
const lib=require("./routers/library")
const app = express();
app.use(express.json());
```

```
const cors = require('cors');

app.use(cors({ origin: "*" }));
app.use((req,res,next)=>{
    res.setHeader("Access-Control-Allow-Origin", "http://localhost:3000");
   res.setHeader("Access-Control-Allow-Methods", "GET, POST, PUT,
DELETE");
   next();
   })
app.use("/",lib);
app.listen(8000, () => {
    console.log("Server is running on port 8000");
});
```

**Library.js**

Router/library.js

```
const express=require("express");
const router = express.Router();
const { Book, Member,Admin } = require('../models/schema');
router.get("/login/userverfication/books",async(req,res)=>{
    try{
        const libraryBooks=await Book.find({});
        if(libraryBooks){
            res.json(libraryBooks)
        }else{
            res.send("No Books Availabe")
        }
    }catch(error){
        res.status(500).send(error.message);
    }
})
router.get("/login/userverfication/books/:id",async (req, res) => {
    try {
      const id = req.params.id;
      const book = await Book.findById(id);
      if (!book) {
        return res.status(404).json({message: "Student not found"});
      }
      res.json(book)
    } catch (error) {
      res.status(500).json({message: error.message});
    }
```

```
  });
router.post("/login/userverfication/books/edit/:id", async (req, res) => {
    try {
      const id = req.params.id;
      const updateData = req.body;
      const updatedBook = await Book.findByIdAndUpdate(id, updateData,
{
        new: true,
      });
      res.json(updatedBook)
    } catch (error) {

      res.status(500).json({message: error.message});
    }
  });
router.get("/login/userverfication/member",async(req,res)=>{
    try{
        const libraryMember=await Member.find({});
        if(libraryMember){
            res.json(libraryMember)
        }else{
            res.send("No Member details There")
        }
    }catch(error){
        res.status(500).send(error.message);
    }
})
router.get("/login/userverfication/member/:id",async (req, res) => {
    try {
      const id = req.params.id;
      const update = await Member.findById(id);
      if (!update) {
        return res.status(404).json({message: "Student not found"});
      }
      res.json(update)
    } catch (error) {
      res.status(500).json({message: error.message});
    }
  });
  router.put("/login/userverfication/member/edit/:id", async (req, res) => {
    try {
        const memberId = req.params.id;
```

```
        const updateData = req.body;

        const updatedMember = await
Member.findByIdAndUpdate(memberId, updateData, { new: true });

        res.json(updatedMember);
    } catch (error) {
        console.error("Error updating member:", error);
        res.status(500).json({ message: "Internal Server Error" });
    }
});




  router.get("/login/userverfication/member/delete/:id", async (req, res)
=> {
    try {
      const id = req.params.id;
      await Member.findByIdAndDelete(id);
      res.send("data has been deleted");
    } catch (error) {
      console.error(error);
      res.status(500).send("Error deleting the data");
    }
  });
router.post("/login/userverfication", async (req, res) => {
    const { name, email } = req.body;
    try {
        const existingUser = await Member.findOne({ name, email });
        if (!existingUser || existingUser.email !== email) {
            return res.status(400).send("Invalid username, password, or user
type");
        }
       res.json(existingUser);
    } catch (error) {
        res.status(500).send(error.message);
    }
});
router.post("/login/admin", async (req, res) => {
    const { name, email } = req.body;
    try {
```

```
        const existingUser = await Admin.findOne({ name, email });
        if (!existingUser || existingUser.email !== email) {
            return res.status(400).send("Invalid username, password, or user
type");
        }
       res.json(existingUser);
    } catch (error) {
        res.status(500).send(error.message);
    }
});

router.post('/login/signup', async (req, res) => {
    const { id,name, email } = req.body;
    try {
        const existingUser = await Member.findOne({ name });


        if (existingUser && existingUser.email === email) {
            return res.status(400).json({ message: "Username already exists.
Please choose a different one." });
        }

        await Member.create({ id,name,email })
        .then(employees =>res.json(employees))
        .catch(err=>res.json(err))
    } catch (error) {
        res.status(500).json({ message: error.message });
    }
});

module.exports = router;
```

**db.js**

Models/db.js

```
const mongoose = require("mongoose");
const {data,user} = require('../models/schema');

mongoose.connect("mongodb://127.0.0.1:27017/library").then(() => {
  console.log("Connected to MongoDB");
}).catch((err) => {
  console.error("Connection failed:", err);
```

```
});

module.exports = mongoose;
```

**Schema.js**

Models/schema.js

```
const mongoose = require("mongoose");

const bookSchema = new mongoose.Schema({
    title: String,
    author: String,
    genre: String,
    availability_status: {
        type: String,
        enum: ["available", "not available"]
    }
});

const Book = mongoose.model("Book", bookSchema);

const memberSchema = new mongoose.Schema({
    id: String,
    name: String,
    email: String,
    books_borrowed: [{
        book: {
            type: mongoose.Schema.Types.ObjectId,
            ref: 'Book'
        }
    }]
}, { versionKey: false });

const Member = mongoose.model("Member", memberSchema);

const adminSchema = new mongoose.Schema({
    name: String,
    email: {
        type: String,
        unique: true
    }
});
```

```
const Admin = mongoose.model("Admin", adminSchema);

module.exports = { Book, Member, Admin };
```

In conclusion, for testing the backend API endpoints of the Library Management System, we're using Jest along with Supertest. Jest offers comprehensive testing utilities, while Supertest facilitates HTTP assertions, allowing us to verify responses to requests. Together, these tools ensure the reliability and correctness of our API implementations, contributing to the robustness of the Library Management System.