Project Title: Library Management System

Subject: Software Engineering {B.Tech – CSE(DS)}

Course Instructor: Gaurav Buddhawar sir, Richa Kumari ma'am

BY Revant Pawar – 50012229

Abhiraj 500120514

Mohd hanzala -500124387

Batch 10

1. Problem Statement

Managing books in a library can become inefficient and error-prone when done manually. Common challenges include tracking issued and returned books, preventing duplicate records, updating book statuses, and ensuring real-time visibility of inventory. This system aims to automate the management of book records, allowing for efficient issuing, returning, adding, updating, and deleting of books.

2. Objectives

- Provide a user-friendly system for managing library books.
- Enable real-time updates of book status (Available, Issued).
- Allow addition, editing, and deletion of book records.
- Ensure smooth communication between frontend and backend.
- Offer error handling and status feedback for user actions.

3. Scope

This system covers:

- Displaying all books with details (title, author, status).
- Allowing users to issue and return books, updating their status.
- Adding new books to the system.
- Deleting books from the system.
- Updating book details directly via the backend.
- Providing a React-based frontend styled with Tailwind CSS, connected to a Node.js + Express backend.

4. System Requirements

Hardware:

- A computer with internet access
- Minimum 4GB RAM (8GB recommended)

Software:

- Node.js and npm
- React and Tailwind CSS
- Express.js backend
- Code editor (VS Code recommended)

5. System Design

- Frontend: React with Tailwind CSS
- Backend: Node.js with Express
- Data Storage: In-memory (can be extended to a database)

Key Components:

- BookList Component (displays books, buttons for actions)
- App Component (handles state, API communication)
- Express Routes (GET, POST, PUT, DELETE for books)

6. API Endpoints

- `GET /books` → Get list of books
- `POST /books` → Add a new book
- `PUT /books/:id` → Update an existing book
- `DELETE /books/:id` → Delete a book

7. Testing Plan

- Unit Testing: Verify backend routes work with valid and invalid data.
- Integration Testing: Ensure React components properly communicate with backend.
- UI Testing: Check responsiveness, button actions, and status updates.
- Boundary Testing: Handle empty inputs, duplicate IDs, or invalid book states.

Example test cases:

- Add a book with missing title → should return error.
- Issue a book already issued → should not reissue.
- Delete a non-existent book ID → should return 404.

8. Security Considerations

- Validate incoming request data (title, author).
- Sanitize inputs to prevent injection.
- Use proper HTTP status codes for responses.
- Consider adding authentication for admin actions (future scope).

9. Future Scope

- Integrate a database like MongoDB or MySQL for persistent storage.
- Add user authentication and roles (admin, librarian, member).
- Implement search and filter features.
- Add due dates and late fee calculations.
- Develop a mobile-friendly version or app.

10. Conclusion

The library management system streamlines basic book management tasks, allowing efficient handling of issuing, returning, adding, and deleting books. With a React-Tailwind frontend and a Node.js backend, the system is modular and can be extended with features like authentication and database integration in the future.

Code

Index.jsx

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

const cors = require('cors');

const logger = require('morgan')

const passport = require("passport");

const session = require("express-session");

const cookieParser = require("cookie-parser");


// Import routers

const authRouter = require("./routes/authRouter")

const bookRouter = require("./routes/bookRouter")

const authorRouter = require("./routes/authorRouter")
```

```
const borrowalRouter = require("./routes/borrowalRouter")
const genreRouter = require("./routes/genreRouter")
const userRouter = require("./routes/userRouter")
const reviewRouter = require("./routes/reviewRouter")

// Configure dotenv for environment variables in production
if (process.env.NODE_ENV !== "production") {
  require("dotenv").config();
}

// Setup express
const app = express();
const PORT = process.env.PORT || 8080

// Use morgan for loggings
app.use(logger("dev"))

// Set middleware to process form data
app.use(express.urlencoded({extended: false}));

// Connect to DB
const mongoose = require('mongoose');
mongoose.connect(process.env.MONGO_URI, {
  useNewUrlParser: true,
  useUnifiedTopology: true
})
```

```
  .then(() => {
    console.log('Connected to DB on MongoDB Atlas')
  })
  .catch((err) => console.log('DB connection error', err));

// Use CORS for Cross Origin Resource Sharing
app.use(cors({
  origin: "http://localhost:3000",
  credentials: true
}))

// Set middleware to manage sessions
app.use(
  session({
    secret: process.env.SESSION_SECRET,
    resave: true,
    saveUninitialized: true,
  })
);

// Parse cookies used for session management
app.use(cookieParser(process.env.SESSION_SECRET));

// Parse JSON objects in request bodies
app.use(express.json())
```

```
// Use passport authentication middleware
app.use(passport.initialize());
app.use(passport.session());

// Initialise passport as authentication middleware
const initializePassport = require("./passport-config");
initializePassport(passport);

// Implement routes for REST API
app.use("/api/auth", authRouter)
app.use("/api/book", bookRouter);
app.use("/api/author", authorRouter);
app.use("/api/borrowal", borrowalRouter);
app.use("/api/genre", genreRouter);
app.use("/api/user", userRouter);
app.use("/api/review", reviewRouter);

app.get('/', (req, res) => res.send('Welcome to Library Management System'));

app.listen(PORT, () => console.log(`Server listening on port ${PORT}!`));
```

userRouter.jsx

```
// Import required modules
const express = require("express")
const router = express.Router();
```

```
// Import functions from controller
const {
  getUser,
  getAllUsers,
  getAllMembers,
  addUser,
  updateUser,
  deleteUser
} = require('../controllers/userController')

router.get("/getAll", (req, res) => getAllUsers(req, res))

router.get("/getAllMembers", (req, res) => getAllMembers(req, res))

router.get("/get/:id", (req, res) => getUser(req, res))

router.post("/add", (req, res) => addUser(req, res))

router.put("/update/:id", (req, res) => updateUser(req, res))

router.delete("/delete/:id", (req, res) => deleteUser(req, res))

module.exports = router;
```

Author router

```
// Import required modules
const express = require("express")
```

```
const router = express.Router();

// Import functions from controller
const {
    getAuthor,
    getAllAuthors,
    addAuthor,
    updateAuthor,
    deleteAuthor
} = require('../controllers/authorController')

router.get("/getAll", (req, res) => getAllAuthors(req, res))

router.get("/get/:id", (req, res) => getAuthor(req, res))

router.post("/add", (req, res) => addAuthor(req, res))

router.put("/update/:id", (req, res) => updateAuthor(req, res))

router.delete("/delete/:id", (req, res) => deleteAuthor(req, res))

module.exports = router;
```

Borrow router

```
// Import required modules
const express = require("express")
const router = express.Router();
```

```
// Import functions from controller
const {
    getBorrowal,
    getAllBorrowals,
    addBorrowal,
    updateBorrowal,
    deleteBorrowal
} = require('../controllers/BorrowalController')

router.get("/getAll", (req, res) => getAllBorrowals(req,res))

router.get("/get/:id", (req, res) => getBorrowal(req, res))

router.post("/add", (req, res) => addBorrowal(req, res))

router.put("/update/:id", (req, res) => updateBorrowal(req, res))

router.delete("/delete/:id", (req, res) => deleteBorrowal(req, res))

module.exports = router;
```

Css Style

```
// @mui
import { alpha } from '@mui/material/styles';

// ----------------------------------------------------------------------
```

```
export function bgBlur(props) {
  const color = props?.color || '#000000';
  const blur = props?.blur || 6;
  const opacity = props?.opacity || 0.8;
  const imgUrl = props?.imgUrl;

  if (imgUrl) {
    return {
      position: 'relative',
      backgroundImage: `url(${imgUrl})`,
      '&:before': {
        position: 'absolute',
        top: 0,
        left: 0,
        zIndex: 9,
        content: """",
        width: '100%',
        height: '100%',
        backdropFilter: `blur(${blur}px)`,
        WebkitBackdropFilter: `blur(${blur}px)`,
        backgroundColor: alpha(color, opacity),
      },
    };
  }
```

```
  return {
    backdropFilter: `blur(${blur}px)`,
    WebkitBackdropFilter: `blur(${blur}px)`,
    backgroundColor: alpha(color, opacity),
  };
}

// ----------------------------------------------------------------------

export function bgGradient(props) {
  const direction = props?.direction || 'to bottom';
  const startColor = props?.startColor;
  const endColor = props?.endColor;
  const imgUrl = props?.imgUrl;
  const color = props?.color;

  if (imgUrl) {
    return {
      background: `linear-gradient(${direction}, ${startColor || color},
${endColor || color}), url(${imgUrl})`,
      backgroundSize: 'cover',
      backgroundRepeat: 'no-repeat',
      backgroundPosition: 'center center',
    };
  }
```

```
  return {
    background: `linear-gradient(${direction}, ${startColor}, ${endColor})`,
  };
}

// ---------------------------------------------------------------------

export function textGradient(value) {
  return {
    background: `-webkit-linear-gradient(${value})`,
    WebkitBackgroundClip: 'text',
    WebkitTextFillColor: 'transparent',
  };
}

// ---------------------------------------------------------------------

export function filterStyles(value) {
  return {
    filter: value,
    WebkitFilter: value,
    MozFilter: value,
  };
}

// ---------------------------------------------------------------------
```

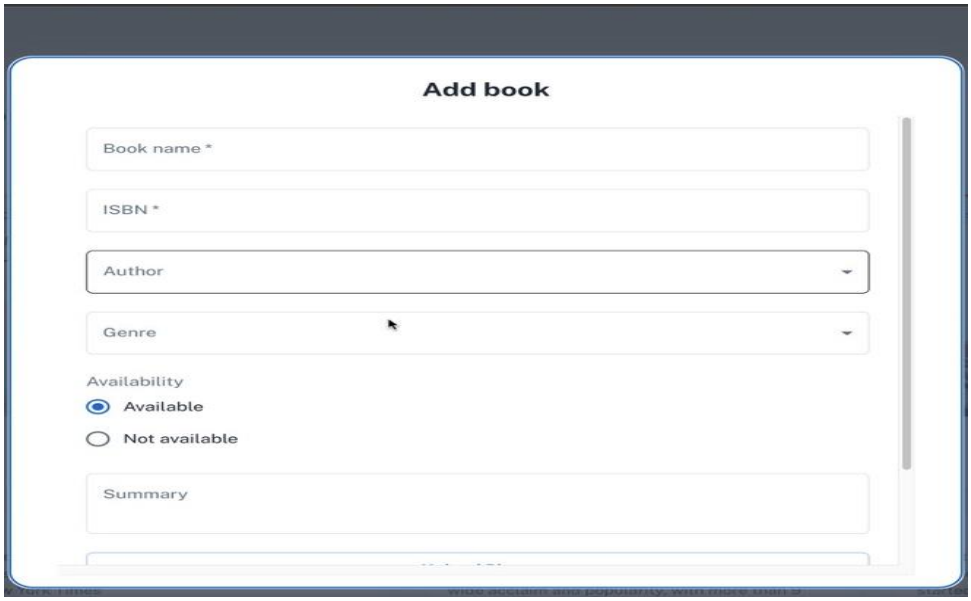
```
export const hideScrollbarY = {
  msOverflowStyle: 'none',
  scrollbarWidth: 'none',
  overflowY: 'scroll',
  '&::-webkit-scrollbar': {
    display: 'none',
  },
};

// ----------------------------------------------------------------------

export const hideScrollbarX = {
  msOverflowStyle: 'none',
  scrollbarWidth: 'none',
  overflowX: 'scroll',
  '&::-webkit-scrollbar': {
    display: 'none',
  },
};
```

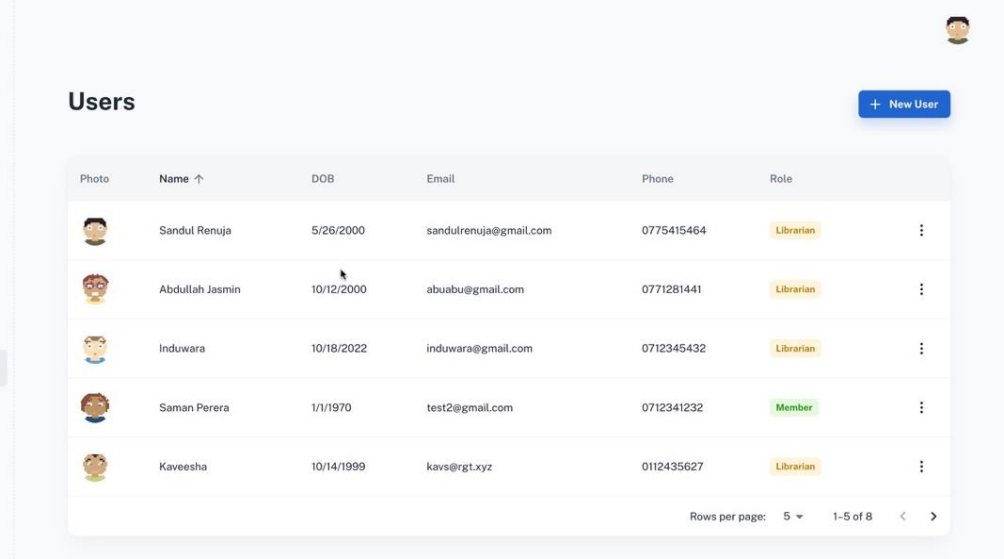
Library App .jsx

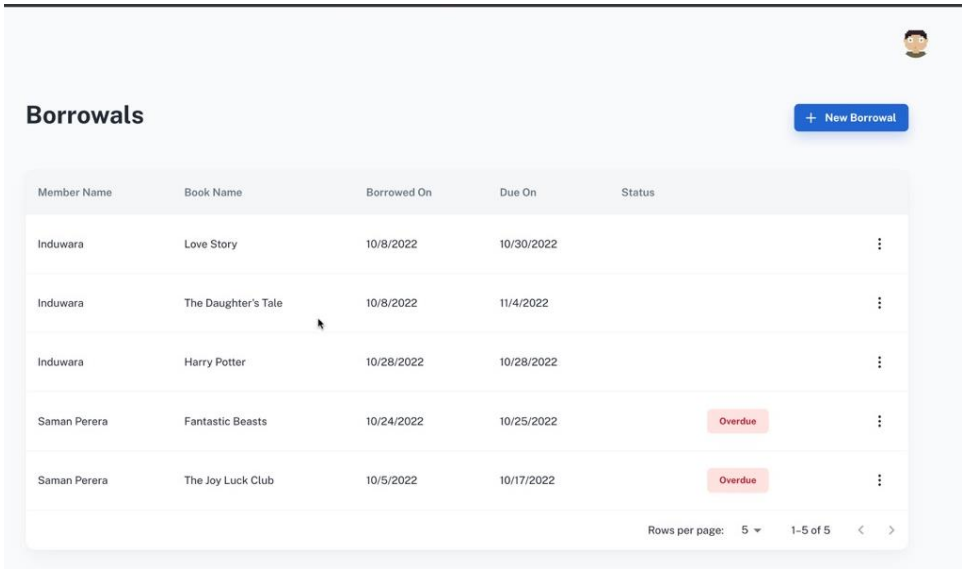
```
import { useState } from "react";
import { Navigate, Outlet } from "react-router-dom";
// @mui
import { styled } from "@mui/material/styles";
```

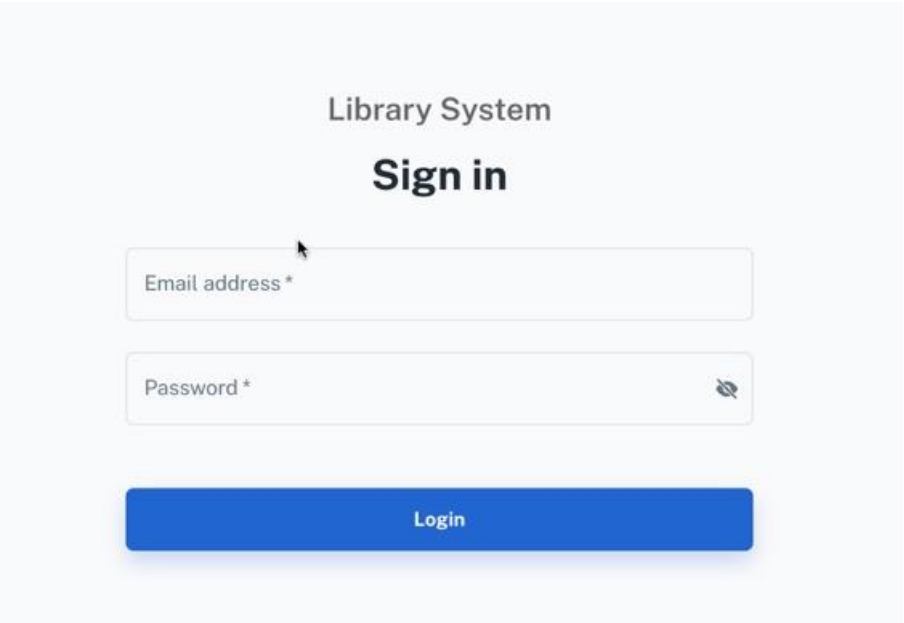
```
//
import Header from "./header";
import Nav from "./nav";
import { useAuth } from "../../hooks/useAuth";

// ---------------------------------------------------------------------

const APP_BAR_MOBILE = 64;
const APP_BAR_DESKTOP = 92;

const StyledRoot = styled('div')({
  display: 'flex',
  minHeight: '100%',
  overflow: 'hidden',
});

const Main = styled('div')(({ theme }) => ({
  flexGrow: 1,
  overflow: 'auto',
  minHeight: '100%',
  paddingTop: APP_BAR_MOBILE + 24,
  paddingBottom: theme.spacing(10),
  [theme.breakpoints.up('lg')]: {
    paddingTop: APP_BAR_DESKTOP + 24,
    paddingLeft: theme.spacing(2),
    paddingRight: theme.spacing(2),
```

```
  },
}));

// ----------------------------------------------------------------------

export default function LibraryApp() {
  const [open, setOpen] = useState(false);
  const {user} = useAuth();

  if (!user) {
    return <Navigate to={'/login'} replace/>
  }

  return (
    <StyledRoot>
      <Header onOpenNav={() => setOpen(true)}/>
      <Nav openNav={open} onCloseNav={() => setOpen(false)}/>
      <Main>
        <Outlet/>
      </Main>
    </StyledRoot>
  );
}
```


Add book
Book name *
ISBN *
Author
Genre
Availability
Available
Not available
Summary


Users
+ New User
Photo
Name
DOB
Email
Phone
Role
Sandul Renuja
5/26/2000
sandulrenuja@gmail.com
0775415464
Librarian
Abdullah Jasmin
10/12/2000
abuabu@gmail.com
0771281441
Librarian
Induwara
10/18/2022
induwara@gmail.com
0712345432
Librarian
Saman Perera
1/1/1970
test2@gmail.com
0712341232
Member
Kaveesha
10/14/1999
kavs@rgt.xyz
0112435627
Librarian
Rows per page: 5
1-5 of 8


Borrowals
+ New Borrowal
Member Name
Book Name
Borrowed On
Due On
Status
Induwara
Love Story
10/8/2022
10/30/2022
Induwara
The Daughter's Tale
10/8/2022
11/4/2022
Induwara
Harry Potter
10/28/2022
10/28/2022
Saman Perera
Fantastic Beasts
10/24/2022
10/25/2022
Overdue
Saman Perera
The Joy Luck Club
10/5/2022
10/17/2022
Overdue
Rows per page: 5
1-5 of 5


Library System
Sign in
Email address *
Password *
Login