online-library-management-system/

├── backend/

│ ├── config/

│ │ └── db.js

│ ├── controllers/

│ │ ├── bookController.js

│ │ ├── transactionController.js

│ │ └── userController.js

│ ├── models/

│ │ ├── Book.js

│ │ ├── Transaction.js

│ │ └── User.js

│ ├── routes/

│ │ ├── bookRoutes.js

│ │ ├── transactionRoutes.js

│ │ └── userRoutes.js

│ ├── middleware/

│ │ └── authMiddleware.js

│ ├── server.js

│ └── .env

├── frontend/

│ ├── public/

│ ├── src/

│ │ ├── components/

│ │ │ ├── AdminDashboard.js

│ │ │ ├── BookList.js

│ │ │ ├── LoginForm.js

│ │ │ ├── RegisterForm.js

│ │ │ ├── TransactionHistory.js

│ │ │ └── UserDashboard.js

│ │ ├── context/

│ │ │ ├── AuthContext.js

│ │ │ ├── AuthProvider.js

│ │ ├── pages/

│ │ │ ├── AdminPage.js

│ │ │ ├── HomePage.js

│ │ │ ├── LoginPage.js

│ │ │ ├── RegisterPage.js

│ │ │ └── UserPage.js

│ │ ├── services/

│ │ │ ├── api.js

│ │ ├── App.js

│ │ ├── index.js

│ │ └── styles.css

│ └── .env

└── README.md

Backend Development—------

**const mongoose = require('mongoose');**

**const connectDB = async () => {**

**try {**

**await mongoose.connect(process.env.MONGO\_URI, {**

**useNewUrlParser: true,**

**useUnifiedTopology: true**

**});**

**console.log('MongoDB connected...');**

**} catch (err) {**

**console.error(err.message);**

**process.exit(1);**

**}**

**};**

**module.exports = connectDB;**

**Model—--**

**const mongoose = require('mongoose');**

**const bookSchema = new mongoose.Schema({**

**name: { type: String, required: true },**

**author: { type: String, required: true },**

**isAvailable: { type: Boolean, default: true }**

**});**

**module.exports = mongoose.model('Book', bookSchema);**

* **User (backend/models/User.js)**

**const mongoose = require('mongoose');**

**const userSchema = new mongoose.Schema({**

**username: { type: String, required: true, unique: true },**

**name: { type: String, required: true },**

**email: { type: String, required: true, unique: true },**

**contactNumber: { type: String, required: true },**

**password: { type: String, required: true },**

**isAdmin: { type: Boolean, default: false }**

**});**

**module.exports = mongoose.model('User', userSchema);**

**Transaction (backend/models/Transaction.js)**

**const mongoose = require('mongoose');**

**const transactionSchema = new mongoose.Schema({**

**user: { type: mongoose.Schema.Types.ObjectId, ref: 'User', required: true },**

**book: { type: mongoose.Schema.Types.ObjectId, ref: 'Book', required: true },**

**dueDate: { type: Date, required: true },**

**transactionType: { type: String, enum: ['borrowed', 'returned'], required: true }**

**});**

**module.exports = mongoose.model('Transaction', transactionSchema);**

**const Book = require('../models/Book');**

**const addBook = async (req, res) => {**

**const { name, author } = req.body;**

**const newBook = new Book({ name, author });**

**await newBook.save();**

**res.status(201).json(newBook);**

**};**

**const getAllBooks = async (req, res) => {**

**const books = await Book.find();**

**res.json(books);**

**};**

**const deleteBook = async (req, res) => {**

**await Book.findByIdAndDelete(req.params.id);**

**res.status(204).json({ message: 'Book removed' });**

**};**

**module.exports = { addBook, getAllBooks, deleteBook };**

**const User = require('../models/User');**

**const bcrypt = require('bcryptjs');**

**const jwt = require('jsonwebtoken');**

**const registerUser = async (req, res) => {**

**const { username, name, email, contactNumber, password } = req.body;**

**const user = new User({ username, name, email, contactNumber, password });**

**const salt = await bcrypt.genSalt(10);**

**user.password = await bcrypt.hash(password, salt);**

**await user.save();**

**res.status(201).json(user);**

**};**

**const loginUser = async (req, res) => {**

**const { username, password } = req.body;**

**const user = await User.findOne({ username });**

**if (user && (await bcrypt.compare(password, user.password))) {**

**const token = jwt.sign({ id: user.\_id }, process.env.JWT\_SECRET, { expiresIn: '1h' });**

**res.json({ token });**

**} else {**

**res.status(400).json({ message: 'Invalid credentials' });**

**}**

**};**

**module.exports = { registerUser, loginUser };**

**const Transaction = require('../models/Transaction');**

**const Book = require('../models/Book');**

**const borrowBook = async (req, res) => {**

**const { user, book, dueDate } = req.body;**

**const transaction = new Transaction({ user, book, dueDate, transactionType: 'borrowed' });**

**await transaction.save();**

**await Book.findByIdAndUpdate(book, { isAvailable: false });**

**res.status(201).json(transaction);**

**};**

**const returnBook = async (req, res) => {**

**const { user, book } = req.body;**

**const transaction = new Transaction({ user, book, dueDate: new Date(), transactionType: 'returned' });**

**await transaction.save();**

**await Book.findByIdAndUpdate(book, { isAvailable: true });**

**res.status(201).json(transaction);**

**};**

**const getUserTransactions = async (req, res) => {**

**const transactions = await Transaction.find({ user: req.params.userId }).populate('book');**

**res.json(transactions);**

**};**

**module.exports = { borrowBook, returnBook, getUserTransactions };**

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

**const { addBook, getAllBooks, deleteBook } = require('../controllers/bookController');**

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

**router.post('/', addBook);**

**router.get('/', getAllBooks);**

**router.delete('/:id', deleteBook);**

**module.exports = router;**

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

**const { registerUser, loginUser } = require('../controllers/userController');**

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

**router.post('/register', registerUser);**

**router.post('/login', loginUser);**

**module.exports = router;**

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

**const { borrowBook, returnBook, getUserTransactions } = require('../controllers/transactionController');**

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

**router.post('/borrow', borrowBook);**

**router.post('/return', returnBook);**

**router.get('/:userId', getUserTransactions);**

**module.exports = router;**

**const jwt = require('jsonwebtoken');**

**const User = require('../models/User');**

**const protect = async (req, res, next) => {**

**let token;**

**if (req.headers.authorization && req.headers.authorization.startsWith('Bearer')) {**

**try {**

**token = req.headers.authorization.split(' ')[1];**

**const decoded = jwt.verify(token, process.env.JWT\_SECRET);**

**req.user = await User.findById(decoded.id).select('-password');**

**next();**

**} catch (error) {**

**res.status(401).json({ message: 'Not authorized, token failed' });**

**}**

**}**

**if (!token) {**

**res.status(401).json({ message: 'Not authorized, no token' });**

**}**

**};**

**module.exports = { protect };**

**Server—----**

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

**const dotenv = require('dotenv');**

**const connectDB = require('./config/db');**

**const bookRoutes = require('./routes/bookRoutes');**

**const userRoutes = require('./routes/userRoutes**