code for frontend is :-  
  
import React from "react";  
import { BrowserRouter as Router, Route, Switch } from "react-router-dom";  
import PrivateRoute from "./PrivateRoute";  
import Dashboard from "./components/Dashboard";  
import LoginPage from "./components/LoginPage";  
import SignupPage from "./components/SignupPage";  
function App() {  
 return (  
 <Router>  
 <Switch>  
 <PrivateRoute exact path="/" component={Dashboard} />  
 <Route path="/login" component={LoginPage} />  
 <Route path="/signup" component={SignupPage} />  
 </Switch>  
 </Router>  
 );  
}  
import React from "react";  
import { Redirect, Route } from "react-router-dom";  
import { isAuthenticated } from "./auth";  
const PrivateRoute = ({ component: Component, ...rest }) => (  
 <Route  
 {...rest}  
 render={(props) =>  
 isAuthenticated() ? (  
 <Component {...props} />  
 ) : (  
 <Redirect to="/login" />  
 )  
 }  
 />  
);  
export const isAuthenticated = () => {  
 };  
export const login = (token) => {  
 };  
export const logout = () => {  
 };  
import React from "react";  
import { Link } from "react-router-dom";  
import { getUser } from "./api"; // Implement an API function to get user data  
const Dashboard = () => {  
 const user = getUser(); // Fetch user data from an API  
 return (  
 <div>  
 <h1>Welcome to the Library Management System</h1>  
 <h2>Categories</h2>  
 <ul>  
 <li>  
 <Link to="/category/1">Sci-Fi</Link>  
 </li>  
 <li>  
 <Link to="/category/2">Fiction</Link>  
 </li>  
 <li>  
 <Link to="/category/3">Comedy</Link>  
 </li>  
 </ul>

{user && (  
 <div>  
 <h2>User Information</h2>  
 <p>Name: {user.name}</p>  
 <p>Phone: {user.phone}</p>  
 <p>Email: {user.email}</p>  
 <p>Address: {user.address}</p>  
 <p>College: {user.college}</p>  
 <h2>Assigned Books</h2>  
 <ul>  
 {user.books.map((book) => (  
 <li key={book.id}>{book.title}</li>  
 ))}  
 </ul>  
 </div>  
 )}  
 </div>  
 );  
}  
import React from "react";  
import { useFormik } from "formik";  
import \* as Yup from "yup";  
import { login } from "./auth";  
const LoginPage = () => {  
 const formik = useFormik({  
 initialValues: {  
 email: "",  
 password: "",  
 },  
 validationSchema: Yup.object({  
 email: Yup.string().email("Invalid email address").required("Required"),  
 password: Yup.string().required("Required"),  
 }),  
 onSubmit: (values) => {  
 // Implement login functionality and set token in the auth module  
 login("dummy\_token");  
 },  
 });  
return (  
 <div>  
 <h1>Login</h1>  
 <form onSubmit={formik.handleSubmit}>  
 <label htmlFor="email">Email</label>  
 <input  
 type="email"  
 id="email"  
 {...formik.getFieldProps("email")}  
 />  
 {formik.touched.email && formik.errors.email ? (  
 <div>{formik.errors.email}</div>  
 ) : null}  
 <label htmlFor="password">Password</label>  
 <input  
 type="password"  
 id="password"  
 {...formik.getFieldProps("password")}  
 />  
 {formik.touched.password && formik.errors.password ? (  
 <div>{formik.errors.password}</div>  
 ) : null}  
 <button type="submit">Login</button>  
 </form>  
 </div>  
 );  
}  
import React from "react";  
import { useFormik } from "formik";  
import \* as Yup from "yup";  
import { signup } from "./api"; // Implement an API function to handle user signup  
const SignupPage = () => {  
 const formik = useFormik({  
 initialValues: {  
 name: "",  
 phone: "",  
 email: "",  
 address: "",  
 college: "",  
 },  
 validationSchema: Yup.object({  
 name: Yup.string().required("Required"),  
 phone: Yup.string().required("Required"),  
 email: Yup.string().email("Invalid email address").required("Required"),  
 address: Yup.string().required("Required"),  
 college: Yup.string().required("Required"),  
 }),  
 onSubmit: (values) => {  
 // Implement signup functionality using an API function  
 signup(values);  
 },  
 });

return (  
 <div>  
 <h1>Signup</h1>  
 <form onSubmit={formik.handleSubmit}>  
 <label htmlFor="name">Name</label>  
 <input  
 type="text"  
 id="name"  
 {...formik.getFieldProps("name")}  
 />  
 {formik.touched.name && formik.errors.name ? (  
 <div>{formik.errors.name}</div>  
 ) : null}  
   
 <label htmlFor="phone">Phone</label>  
 <input  
 type="text"  
 id="phone"  
 {...formik.getFieldProps("phone")}  
 />  
 {formik.touched.phone && formik.errors.phone ? (  
 <div>{formik.errors.phone}</div>  
 ) : null}  
 <label htmlFor="email">Email</label>  
 <input  
 type="email"  
 id="email"  
 {...formik.getFieldProps("email")}  
 />  
 {formik.touched.email && formik.errors.email ? (  
 <div>{formik.errors.email}</div>  
 ) : null}  
 <label htmlFor="address">Address</label>  
 <input  
 type="text"  
 id="address"  
 {...formik.getFieldProps("address")}  
 />  
 {formik.touched.address && formik.errors.address ? (  
 <div>{formik.errors.address}</div>  
 ) : null}  
 <label htmlFor="college">College</label>  
 <input  
 type="text"  
 id="college"  
 {...formik.getFieldProps("college")}  
 />  
 {formik.touched.college && formik.errors.college ? (  
 <div>{formik.errors.college}</div>  
 ) : null}  
   
 <button type="submit">Signup</button>  
 </form>  
 </div>  
 );  
}

code for backend is :-  
  
'rest\_framework',  
 'rest\_framework\_simplejwt',  
 ]  
REST\_FRAMEWORK = {  
 'DEFAULT\_AUTHENTICATION\_CLASSES': [  
 'rest\_framework\_simplejwt.authentication.JWTAuthentication',  
 ],  
}  
SIMPLE\_JWT = {  
 'AUTH\_HEADER\_TYPES': ('Bearer',),  
}  
python manage.py startapp authentication  
from rest\_framework import serializers  
from django.contrib.auth.models import User  
from rest\_framework\_simplejwt.serializers import TokenObtainPairSerializer  
class UserSerializer(serializers.ModelSerializer):  
 class Meta:  
 model = User  
 fields = ['id', 'username', 'email', 'password']  
 extra\_kwargs = {'password': {'write\_only': True}}  
 def create(self, validated\_data):  
 user = User.objects.create\_user(  
 username=validated\_data['username'],  
 email=validated\_data['email'],  
 password=validated\_data['password']  
 )  
 return user  
class MyTokenObtainPairSerializer(TokenObtainPairSerializer):  
@classmethod  
 def get\_token(cls, user):  
 token = super().get\_token(user)  
 token['username'] = user.username  
 token['email'] = user.email  
return token  
from rest\_framework.views import APIView  
from rest\_framework.response import Response  
from rest\_framework.permissions import AllowAny  
from .serializers import UserSerializer, MyTokenObtainPairSerializer  
class CreateUserView(APIView):  
 permission\_classes = [AllowAny]  
 def post(self, request, format='json'):  
 serializer = UserSerializer(data=request.data)  
 if serializer.is\_valid():  
 serializer.save()  
 return Response(serializer.data, status=201)  
 return Response(serializer.errors, status=400)  
class MyTokenObtainPairView(TokenObtainPairView):  
 serializer\_class = MyTokenObtainPairSerializer  
from django.urls import path  
from authentication.views import CreateUserView, MyTokenObtainPairView  
urlpatterns = [  
 path('api/register/', CreateUserView.as\_view(), name='register'),  
 path('api/token/', MyTokenObtainPairView.as\_view(), name='token\_obtain\_pair'),  
]  
python manage.py makemigrations  
python manage.py migrate