Backend: Django REST Framework

1. Project Setup

```
# create & activate virtualenv
python -m venv venv
source venv/bin/activate

# install dependencies
pip install django djangorestframework djangorestframework-simplejwt mysqlclient

# start project & app
django-admin startproject backend
cd backend
python manage.py startapp books
```

In backend/settings.py:

```
INSTALLED_APPS = [
  ...,
  'rest_framework',
  'rest_framework_simplejwt.token_blacklist',
  'books',
]

DATABASES = {
  'default': {
    'ENGINE':  'django.db.backends.mysql',
    'NAME':   'your_db',
    'USER':   'your_user',
    'PASSWORD':'your_password',
```

```
    'HOST':   'localhost',
    'PORT':   '3306',
  }
}

REST_FRAMEWORK = {
  'DEFAULT_AUTHENTICATION_CLASSES': (
    'rest_framework_simplejwt.authentication.JWTAuthentication',
  ),
  'DEFAULT_PERMISSION_CLASSES': (
    'rest_framework.permissions.IsAuthenticated',
  ),
}
```

2. Models

books/models.py

```
from django.db import models
from django.contrib.auth.models import User

class Book(models.Model):
  owner       = models.ForeignKey(User, on_delete=models.CASCADE,
related_name='books')
  title       = models.CharField(max_length=200)
  author      = models.CharField(max_length=100)
  publication_date = models.DateField()
  genre       = models.CharField(max_length=50)
  description   = models.TextField(blank=True)

  def __str__(self):
```

```
        return self.title
```

python manage.py makemigrations

python manage.py migrate

3. Serializers

books/serializers.py

```
from rest_framework import serializers
from django.contrib.auth.models import User
from .models import Book


class UserSerializer(serializers.ModelSerializer):
    password = serializers.CharField(write_only=True, min_length=8)
    class Meta:
        model = User
        fields = ['id','username','email','password']


    def create(self, validated_data):
        user = User.objects.create_user(**validated_data)
        return user


class BookSerializer(serializers.ModelSerializer):
    owner = serializers.ReadOnlyField(source='owner.username')
    class Meta:
        model  = Book
        fields = ['id','owner','title','author','publication_date','genre','description']
```

4. Views & Permissions

books/permissions.py

```
from rest_framework import permissions
```

```
class IsOwnerOrReadOnly(permissions.BasePermission):
    def has_object_permission(self, request, view, obj):
        if request.method in permissions.SAFE_METHODS:
            return True
        return obj.owner == request.user
books/views.py
from rest_framework import viewsets, generics, status
from rest_framework.response import Response
from rest_framework.permissions import AllowAny
from rest_framework_simplejwt.views import TokenObtainPairView
from .models import Book
from .serializers import UserSerializer, BookSerializer
from .permissions import IsOwnerOrReadOnly

# Registration endpoint
class RegisterView(generics.CreateAPIView):
    queryset = User.objects.all()
    serializer_class = UserSerializer
    permission_classes = [AllowAny]

# JWT login is handled by SimpleJWT's view:
class LoginView(TokenObtainPairView):
    permission_classes = [AllowAny]

# CRUD for Book
class BookViewSet(viewsets.ModelViewSet):
    queryset       = Book.objects.all()
    serializer_class = BookSerializer
```

```
    permission_classes = [IsOwnerOrReadOnly]

    def perform_create(self, serializer):
        serializer.save(owner=self.request.user)
```

5. URLs

backend/urls.py

```
from django.urls import path, include
from rest_framework.routers import DefaultRouter
from books.views import RegisterView, LoginView, BookViewSet

router = DefaultRouter()
router.register(r'books', BookViewSet, basename='book')

urlpatterns = [
    path('api/auth/register/', RegisterView.as_view(), name='register'),
    path('api/auth/login/',    LoginView.as_view(),    name='login'),
    path('api/', include(router.urls)),
]
```

Frontend: React.js

1. Project Setup

```
npx create-react-app frontend
cd frontend
npm install axios react-router-dom
```

2. Auth Context (React Context API)

src/context/AuthContext.js

```
jsx
CopyEdit
```

```
import React, { createContext, useState, useEffect } from 'react';
import axios from 'axios';

export const AuthContext = createContext();

export function AuthProvider({ children }) {
 const [authTokens, setAuthTokens] = useState(
  () => JSON.parse(localStorage.getItem('tokens'))
 );
 const [user, setUser] = useState(() =>
  authTokens ? parseJwt(authTokens.access) : null
 );

 useEffect(() => {
  axios.defaults.headers.common['Authorization'] = authTokens
   ? `Bearer ${authTokens.access}`
   : '';
 }, [authTokens]);

 const login = async (credentials) => {
  const resp = await axios.post('/api/auth/login/', credentials);
  setAuthTokens(resp.data);
  setUser(parseJwt(resp.data.access));
  localStorage.setItem('tokens', JSON.stringify(resp.data));
 };

 const register = async (data) => {
  await axios.post('/api/auth/register/', data);
```

```
  };

  const logout = () => {
    setAuthTokens(null);
    setUser(null);
    localStorage.removeItem('tokens');
    delete axios.defaults.headers.common['Authorization'];
  };

  return (
    <AuthContext.Provider value={{ user, login, register, logout }}>
      {children}
    </AuthContext.Provider>
  );
}

// simple base64 JWT parse
function parseJwt(token) {
  return JSON.parse(atob(token.split('.')[1]));
}
```

3. Routing & Protected Routes

src/App.js

```
import { BrowserRouter, Routes, Route, Navigate } from 'react-router-dom';
import { AuthProvider, AuthContext } from './context/AuthContext';
import Login    from './pages/Login';
import Register  from './pages/Register';
import BookList  from './pages/BookList';
import BookDetail from './pages/BookDetail';
```

```
import BookForm  from './pages/BookForm';

function PrivateRoute({ children }) {
 const { user } = React.useContext(AuthContext);
 return user ? children : <Navigate to="/login" />;
}

export default function App() {
 return (
  <AuthProvider>
   <BrowserRouter>
    <Routes>
     <Route path="/login"  element={<Login />} />
     <Route path="/register" element={<Register />} />

     <Route path="/books" element={
      <PrivateRoute><BookList/></PrivateRoute>
     }/>
     <Route path="/books/add" element={
      <PrivateRoute><BookForm/></PrivateRoute>
     }/>
     <Route path="/books/:id" element={
      <PrivateRoute><BookDetail/></PrivateRoute>
     }/>
     <Route path="/books/:id/edit" element={
      <PrivateRoute><BookForm editMode/></PrivateRoute>
     }/>
```

```
        <Route path="*" element={<Navigate to="/books"/>} />
      </Routes>
    </BrowserRouter>
  </AuthProvider>
 );
}
```

4. Example Page: BookList

src/pages/BookList.js

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

export default function BookList() {
 const [books, setBooks] = useState([]);

 useEffect(() => {
  axios.get('/api/books/')
     .then(res => setBooks(res.data));
 }, []);

 return (
  <div>
   <h1>Your Books</h1>
   <Link to="/books/add">Add New Book</Link>
   <ul>
    {books.map(book => (
     <li key={book.id}>
      <Link to={`/books/${book.id}`}>{book.title}</Link>
```

```
        </li>
      ))}
    </ul>
  </div>
 );
}
```

5. Other Pages

- **Login.js: form calls login({ username, password }) from context.**
- **Register.js: form calls register({ username, email, password }).**
- **BookDetail.js: fetch /api/books/:id/ and show details, with "Edit" & "Delete" buttons.**
- **BookForm.js: reusable for Add and Edit. On submit, POST or PUT to /api/books/ or /api/books/:id/.**

---

6. Final Touches

- **CORS: if React runs on a different origin, add django-cors-headers.**
- **Pagination: DRF offers page sizes you can configure in REST_FRAMEWORK.**
- **Validation & Notifications: use React form libraries or custom state to show errors/success.**
- **Styling: add your CSS or a component library (Tailwind, Material-UI, etc.).**