Here’s a **step-by-step Instruction Manual & Construction Guide** for your Django + React “Royal Road–style” novel platform. You can copy/paste these bits into your project (and read the narrative explanations to understand how it all fits together).

## 1. Prerequisites

* **Python 3.8+** (we’ll use a venv)
* **Node.js 12+ & npm**
* Git (optional)
* Basic familiarity with the terminal and code editors

## 2. Project Layout

royalroad\_clone/ ← Django project root

├── manage.py

├── royalroad\_clone/ ← Django settings + URLs + WSGI

│ ├── settings.py

│ ├── urls.py

│ └── wsgi.py

├── core/ ← Django “app” with your models/serializers/views

│ ├── models.py

│ ├── serializers.py

│ ├── views.py

│ ├── permissions.py

│ └── urls.py ← if you split your API URLs here

└── rr-frontend/ ← React front‐end

├── package.json

└── src/

├── api.js ← axios instance + refresh logic

├── index.js ← wraps <App/> in BrowserRouter

├── App.js ← top‐level <Routes/> & <NavBar/> + <Footer/>

├── NavBar.js ← nav links + login/logout

├── Hero.js ← optional banner component

├── Login.js ← JWT login form

├── StoryFront.js ← public home, filtering, new‐story button

├── StoryList.js ← admin list view (similar grid)

├── StoryForm.js ← create/edit story form

├── StoryPage.js ← view story, chapters, add‐chapter form

├── ChapterForm.js ← create chapter form

├── ChapterPage.js ← read a chapter, comments, rating

├── CommentForm.js ← post comments

└── Footer.js ← site footer

## 3. Backend Setup (Django + DRF)

### a) Virtualenv & Dependencies

cd ~/projects

python3 -m venv rr-env

source rr-env/bin/activate

pip install django djangorestframework djangorestframework-simplejwt django-cors-headers django-filter

### b) Start Project

django-admin startproject royalroad\_clone

cd royalroad\_clone

python manage.py startapp core

### c) settings.py

* **Install apps**:
* INSTALLED\_APPS = [
* …,
* 'corsheaders',
* 'rest\_framework',
* 'rest\_framework\_simplejwt',
* 'django\_filters',
* 'core',
* ]
* **Middleware**:
* MIDDLEWARE = [
* …,
* 'corsheaders.middleware.CorsMiddleware',
* …
* ]
* **REST\_FRAMEWORK**:
* REST\_FRAMEWORK = {
* 'DEFAULT\_AUTHENTICATION\_CLASSES': [
* 'rest\_framework\_simplejwt.authentication.JWTAuthentication',
* ],
* 'DEFAULT\_PERMISSION\_CLASSES': [
* 'rest\_framework.permissions.IsAuthenticatedOrReadOnly',
* ],
* 'DEFAULT\_FILTER\_BACKENDS': [
* 'django\_filters.rest\_framework.DjangoFilterBackend'
* ],
* }
* **CORS**:
* CORS\_ALLOWED\_ORIGINS = ['http://localhost:3000']
* CORS\_ALLOW\_ALL\_ORIGINS = True # dev only

### d) models.py

Define your models and M2M through table:

from django.db import models

from django.contrib.auth.models import User

class Tag(models.Model):

name = models.CharField(max\_length=50, unique=True)

class Story(models.Model):

STATUS\_CHOICES = [('ONGOING','Ongoing'),('COMPLETED','Completed')]

author = models.ForeignKey(User, on\_delete=models.CASCADE, related\_name='stories')

title = models.CharField(max\_length=255)

summary = models.TextField()

status = models.CharField(max\_length=10, choices=STATUS\_CHOICES, default='ONGOING')

tags = models.ManyToManyField(Tag, through='StoryTag', related\_name='stories')

created\_at = models.DateTimeField(auto\_now\_add=True)

updated\_at = models.DateTimeField(auto\_now=True)

class StoryTag(models.Model):

story = models.ForeignKey(Story, on\_delete=models.CASCADE)

tag = models.ForeignKey(Tag, on\_delete=models.CASCADE)

class Chapter(models.Model):

story = models.ForeignKey(Story, on\_delete=models.CASCADE, related\_name='chapters')

title = models.CharField(max\_length=255)

content = models.TextField()

position = models.PositiveIntegerField()

created\_at = models.DateTimeField(auto\_now\_add=True)

updated\_at = models.DateTimeField(auto\_now=True)

class Meta: ordering = ['position']

class Comment(models.Model):

user = models.ForeignKey(User, on\_delete=models.CASCADE)

chapter = models.ForeignKey(Chapter, on\_delete=models.CASCADE, related\_name='comments')

content = models.TextField()

created\_at = models.DateTimeField(auto\_now\_add=True)

class Rating(models.Model):

user = models.ForeignKey(User, on\_delete=models.CASCADE)

chapter = models.ForeignKey(Chapter, on\_delete=models.CASCADE, related\_name='ratings')

value = models.PositiveSmallIntegerField()

created\_at = models.DateTimeField(auto\_now\_add=True)

class Meta: unique\_together = ('user','chapter')

### e) serializers.py

(See the previous complete answer to copy/paste.)

### f) permissions.py

from rest\_framework.permissions import BasePermission

class IsOwnerOnly(BasePermission):

def has\_object\_permission(self, request, view, obj):

return obj.author == request.user

### g) views.py

(Use the final version with queryset = Story.objects.all() plus your two custom actions in RatingViewSet.)

### h) urls.py

from django.contrib import admin

from django.urls import path, include

from rest\_framework.routers import DefaultRouter

from rest\_framework\_nested.routers import NestedDefaultRouter

from core.views import TagViewSet, StoryViewSet, ChapterViewSet, CommentViewSet, RatingViewSet

router = DefaultRouter()

router.register(r'tags', TagViewSet)

router.register(r'stories', StoryViewSet, basename='stories')

router.register(r'comments',CommentViewSet)

router.register(r'ratings', RatingViewSet)

stories\_router = NestedDefaultRouter(router, 'stories', lookup='story')

stories\_router.register('chapters', ChapterViewSet, basename='story-chapters')

urlpatterns = [

path('admin/', admin.site.urls),

path('api/', include(router.urls)),

path('api/', include(stories\_router.urls)),

path('api/token/', include('rest\_framework\_simplejwt.urls')),

]

### i) Migrate & Create Superuser

python manage.py makemigrations

python manage.py migrate

python manage.py createsuperuser

## 4. Frontend Setup (React + Axios)

### a) Create React App

cd royalroad\_clone

npx create-react-app rr-frontend

cd rr-frontend

npm install axios react-router-dom jwt-decode

### b) src/api.js

import axios from 'axios';

const API = axios.create({

baseURL: 'http://127.0.0.1:8000/api/',

});

API.interceptors.request.use(cfg => {

const t = localStorage.getItem('access');

if (t) cfg.headers.Authorization = `Bearer ${t}`;

return cfg;

});

export default API;

(Optionally add refresh‐token logic as shown earlier.)

### c) src/index.js

Wrap in a Router:

import React from 'react';

import ReactDOM from 'react-dom/client';

import { BrowserRouter } from 'react-router-dom';

import './index.css';

import App from './App';

const root = ReactDOM.createRoot(document.getElementById('root'));

root.render(

<React.StrictMode>

<BrowserRouter>

<App />

</BrowserRouter>

</React.StrictMode>

);

### d) src/App.js

import { Routes, Route } from 'react-router-dom';

import NavBar from './NavBar';

import Footer from './Footer';

import Login from './Login';

import StoryFront from './StoryFront';

import StoryList from './StoryList';

import StoryForm from './StoryForm';

import StoryPage from './StoryPage';

import ChapterPage from './ChapterPage';

export default function App() {

return (

<>

<NavBar />

<main className="container mx-auto p-4">

<Routes>

<Route path="/login" element={<Login />} />

<Route path="/" element={<StoryFront />} />

<Route path="/stories" element={<StoryList />} />

<Route path="/stories/new" element={<StoryForm />} />

<Route path="/stories/:id" element={<StoryPage />} />

<Route path="/stories/:storyId/chapters/:chapterId" element={<ChapterPage />} />

</Routes>

</main>

<Footer />

</>

);

}

### e) React Components

* **NavBar.js**: shows Home / Login / Logout, reads access token, uses jwtDecode.
* **Login.js**: posts to api/token/, stores access & refresh.
* **StoryFront.js**: fetches /stories/ & /tags/, filters by tag, shows grid + “+ New Story” if logged in.
* **StoryList.js**: same grid but for admin/users; inline <StoryForm onNewStory={…}/>.
* **StoryForm.js**: form to create (or edit) a story; optional onNewStory.
* **StoryPage.js**: GET /stories/{id}/, /stories/{id}/chapters/, GET /ratings/story/{id}/average/, allows rating & (if you’re the author) shows <ChapterForm/>.
* **ChapterForm.js**: GET /stories/{storyId}/chapters/ to determine next position, POST nested to create.
* **ChapterPage.js**: GET /stories/{s}/chapters/{c}/, GET /comments/?chapter={c}, GET /ratings/chapter/{c}/, allows commenting & rating.
* **CommentForm.js**: simple POST to /comments/.

(You already have these—just ensure the imports, prop‐names, and URL paths match exactly.)

## 5. Running

* **Backend**:
* source rr-env/bin/activate
* python manage.py runserver
* **Frontend**:
* cd rr-frontend
* npm start
* Open <http://localhost:3000> in your browser.

## 6. How It Works

1. **Django REST API**
   * Models → DRF ModelSerializers → ModelViewSets
   * JWT auth via SimpleJWT
   * CORS & filtering by tag
   * Nested chapters under stories
   * Custom @actions for ratings
2. **React SPA**
   * api.js hooks into axios for JWT
   * index.js wraps in BrowserRouter
   * App.js defines all routes + shared NavBar/Footer
   * Pages fetch data on useEffect, store in state
   * Forms POST via API.post(...), update UI + redirect

## 7. Final Tips

* **Styling**: we used Tailwind classes in JSX (install it or replace with your CSS).
* **Artwork**: put banner images in public/assets/… and reference by <img src=“…/assets/…”>.
* **Next Features**: paging, full‐text search, user profiles, notifications.

With those files in place and servers restarted, you’ll have:

* ✔️ public browse & filter by tag
* ✔️ signup/login (via admin or extend with registration)
* ✔️ create/edit your own stories & chapters
* ✔️ comments & 5-star ratings per chapter
* ✔️ average ratings on stories

Feel free to let me know if any step needs more detail!