🧪 Complete Testing Guide for Contentra CMS

Step 1: Reset and Seed Database

```bash
# Clear cache and reset database
php artisan config:clear
php artisan cache:clear
php artisan route:clear

# Fresh migration with seeder
php artisan migrate:fresh --seed
```

Expected Output:

```
✅ Roles, Permissions, and Test Users created successfully!

Test Credentials:
Super Admin: superadmin@contentra.test / password
Admin: admin@contentra.test / password
Editor: editor@contentra.test / password
Author: author@contentra.test / password
Viewer: viewer@contentra.test / password
```

Step 2: Test Web Authentication

2.1 Login Test

1. Visit `http://localhost:8000/login`
2. Login with: `admin@contentra.test` / `password`
3. Should redirect to `/home`
4. Visit `http://localhost:8000/admin/dashboard`
5. Should see dashboard with user counts

2.2 Permission Test

1. Logout and login as: `viewer@contentra.test` / `password`

2. Try to access: `http://localhost:8000/admin/roles/create`
3. **Expected:** 403 Forbidden error (viewers can't create roles)

Step 3: Test API Authentication

3.1 Register New User

Request:

```http
POST http://localhost:8000/api/register
Content-Type: application/json

{
  "name": "Test User",
  "email": "test@example.com",
  "password": "password123",
  "password_confirmation": "password123"
}
```

Expected Response (201):

```json
{
  "message": "User registered successfully",
  "user": {
    "id": 6,
    "name": "Test User",
    "email": "test@example.com",
    "roles": [
      {
        "name": "viewer",
        "permissions": [...]
      }
    ]
  },
  "token": "1|xxxxxxxxxxxx"
}
```

3.2 Login

Request:

```http
POST http://localhost:8000/api/login
Content-Type: application/json

{
  "email": "admin@contentra.test",
  "password": "password"
}
```

Expected Response (200):

```json
{
  "message": "Login successful",
  "user": {
    "id": 2,
    "name": "Admin User",
    "email": "admin@contentra.test",
    "roles": [...]
  },
  "token": "2|xxxxxxxxxxxx"
}
```

📝 **SAVE THIS TOKEN** - You'll need it for subsequent requests!

3.3 Get Current User

Request:

```http
GET http://localhost:8000/api/user
Authorization: Bearer YOUR_TOKEN_HERE
```

Expected Response (200):

```json
```

```
{
  "user": {
    "id": 2,
    "name": "Admin User",
    "email": "admin@contentra.test",
    "roles": [
      {
        "name": "admin",
        "permissions": [...]
      }
    ]
  }
}
```

Step 4: Test Permission Enforcement

4.1 Test WITH Admin Token (Should Work)

Get an admin token first:

```
http

POST http://localhost:8000/api/login
Content-Type: application/json

{
  "email": "admin@contentra.test",
  "password": "password"
}
```

Then try to view users:

```
http

GET http://localhost:8000/api/users
Authorization: Bearer ADMIN_TOKEN_HERE
```

Expected: ✅ Returns list of users

4.2 Test WITH Viewer Token (Should Fail)

Get a viewer token:

```http
POST http://localhost:8000/api/login
Content-Type: application/json

{
  "email": "viewer@contentra.test",
  "password": "password"
}
```

Then try to view users:

```http
GET http://localhost:8000/api/users
Authorization: Bearer VIEWER_TOKEN_HERE
```

Expected Response (403):

```json
{
  "message": "Forbidden. You do not have permission to access this resource.",
  "required_permission": "view-users"
}
```

Step 5: Test All API Endpoints

5.1 Users API

List Users

```http
GET http://localhost:8000/api/users
Authorization: Bearer ADMIN_TOKEN
```

Create User

```http
```

```
POST http://localhost:8000/api/users
Authorization: Bearer ADMIN_TOKEN
Content-Type: application/json

{
  "name": "New User",
  "email": "newuser@test.com",
  "password": "password123",
  "roles": ["editor"]
}
```

Update User

```
http

PUT http://localhost:8000/api/users/6
Authorization: Bearer ADMIN_TOKEN
Content-Type: application/json

{
  "name": "Updated Name",
  "roles": ["author"]
}
```

Delete User

```
http

DELETE http://localhost:8000/api/users/6
Authorization: Bearer ADMIN_TOKEN
```

5.2 Roles API

List Roles

```
http

GET http://localhost:8000/api/roles
Authorization: Bearer ADMIN_TOKEN
```

Create Role

```http
POST http://localhost:8000/api/roles
Authorization: Bearer ADMIN_TOKEN
Content-Type: application/json

{
  "name": "content-manager",
  "permissions": ["view-content", "create-content", "edit-content"]
}
```

Update Role

```http
PUT http://localhost:8000/api/roles/6
Authorization: Bearer ADMIN_TOKEN
Content-Type: application/json

{
  "name": "content-manager-updated",
  "permissions": ["view-content", "create-content"]
}
```

Delete Role

```http
DELETE http://localhost:8000/api/roles/6
Authorization: Bearer ADMIN_TOKEN
```

5.3 Permissions API

List Permissions

```http
GET http://localhost:8000/api/permissions
Authorization: Bearer ADMIN_TOKEN
```

Create Permission

```http
```

```
POST http://localhost:8000/api/permissions
Authorization: Bearer ADMIN_TOKEN
Content-Type: application/json

{
  "name": "custom-permission"
}
```

Step 6: Permission Matrix Testing

Test each role's capabilities:

Endpoint	Super Admin	Admin	Editor	Author	Viewer
GET /api/users	✅	✅	❌	❌	❌
POST /api/users	✅	✅	❌	❌	❌
GET /api/roles	✅	✅	❌	❌	❌
POST /api/roles	✅	✅	❌	❌	❌
GET /api/permissions	✅	✅	❌	❌	❌

Step 7: Logout Test

```
http

POST http://localhost:8000/api/logout
Authorization: Bearer YOUR_TOKEN
```

Expected Response (200):

```
json

{
  "message": "Logged out successfully"
}
```

Then try to use the same token:

```http
GET http://localhost:8000/api/user
Authorization: Bearer SAME_TOKEN
```

Expected: 401 Unauthenticated

Common Issues & Solutions

Issue: "Unauthenticated" even with token

Solution: Make sure you're using `Authorization: Bearer TOKEN` header

Issue: "Permission not found"

Solution: Run `php artisan cache:clear` and `php artisan config:clear`

Issue: Permissions not working

Solution:

```bash
php artisan permission:cache-reset
php artisan config:clear
```

Issue: Can't access admin panel

Solution: Make sure your user has roles assigned with 'web' guard

---

Next Steps

Once all tests pass, you're ready for **Sprint 2: Content Modeling**! 🎉

This includes:

- Content Type Builder
- Dynamic field definitions
- Schema storage

- Content entry CRUD

Would you like me to proceed with Content Modeling implementation?