Django Watchlist Projekt: Schritt-für-Schritt-Anleitung

In diesem Projekt erstellen wir eine Django-Webanwendung, bei der Benutzer sich registrieren, anmelden und ihre persönliche Watchlist von Lieblingsfilmen verwalten können. Jeder Benutzer sieht nur seine eigenen Einträge und kann diese bearbeiten.

---

1. Projekt-Setup

Projekt erstellen und App hinzufügen

```
django-admin startproject watchlist_project
cd watchlist_project
python manage.py startapp movies
```

App registrieren

In `settings.py`:

```
INSTALLED_APPS = [
    ...,
    'movies',
]
```

Führe die Migrationen aus:

```
python manage.py migrate
```

---

2. Benutzerregistrierung und Authentifizierung

Benutzerregistrierungsformular

`forms.py` in der `movies` App

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

class UserRegistrationForm(forms.ModelForm):
    password = forms.CharField(widget=forms.PasswordInput)
    confirm_password = forms.CharField(widget=forms.PasswordInput)

    class Meta:
        model = User
        fields = ['username', 'email', 'password']

    def clean(self):
        cleaned_data = super().clean()
        password = cleaned_data.get('password')
        confirm_password = cleaned_data.get('confirm_password')

        if password != confirm_password:
            raise forms.ValidationError("Passwörter stimmen nicht überein.")
```

View für Registrierung

`views.py` **in der** `movies` **App**

```python
from django.shortcuts import render, redirect
from django.contrib.auth.models import User
from django.contrib.auth import login, authenticate
from .forms import UserRegistrationForm
from django.contrib import messages

def register(request):
    if request.method == 'POST':
        form = UserRegistrationForm(request.POST)
        if form.is_valid():
            user = form.save(commit=False)
            user.set_password(form.cleaned_data['password'])
            user.save()
            messages.success(request, 'Registrierung erfolgreich! Bitte melden Sie sich an.')
            return redirect('login')
    else:
        form = UserRegistrationForm()
    return render(request, 'movies/register.html', {'form': form})
```

Template für Registrierung

`templates/movies/register.html`

```html
<h2>Registrierung</h2>
<form method="post">
    {% csrf_token %}
    {{ form.as_p }}
    <button type="submit">Registrieren</button>
</form>
```

Login und Logout

`views.py`

```python
from django.contrib.auth import login, logout
from django.contrib.auth.forms import AuthenticationForm

def login_view(request):
    if request.method == 'POST':
        form = AuthenticationForm(data=request.POST)
        if form.is_valid():
            user = form.get_user()
            login(request, user)
            return redirect('movies:watchlist')
    else:
        form = AuthenticationForm()
    return render(request, 'movies/login.html', {'form': form})

def logout_view(request):
    logout(request)
    return redirect('login')
```

`templates/movies/login.html`

```
<h2>Anmeldung</h2>
<form method="post">
    {% csrf_token %}
    {{ form.as_p }}
    <button type="submit">Anmelden</button>
</form>
```

---

3. Watchlist-Funktionalität

Modell für Filme

`models.py`

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

class Movie(models.Model):
    user = models.ForeignKey(User, on_delete=models.CASCADE)
    title = models.CharField(max_length=255)
    director = models.CharField(max_length=255)
    release_year = models.PositiveIntegerField()
    genre = models.CharField(max_length=100)
    imdb_url = models.URLField(blank=True, null=True)  # IMDb-URL optional


    def __str__(self):
        return self.title
```

Führe die Migrationen aus:

```
python manage.py makemigrations
python manage.py migrate
```

Watchlist-Ansicht

`views.py`

```
from django.contrib.auth.decorators import login_required
from .models import Movie

@login_required(login_url='/login/')
def watchlist_view(request):
    movies = Movie.objects.filter(user=request.user)
    return render(request, 'movies/watchlist.html', {'movies': movies})
```

`templates/movies/watchlist.html`

```
<h2>Deine Watchlist</h2>
<ul>
    {% for movie in movies %}
        <li>
            <strong>{{ movie.title }}</strong> ({{ movie.release_year }}) - {{ movie.genre }}
            <br>Regisseur: {{ movie.director }}
            {% if movie.imdb_url %}
```

```
                <br><a href="{{ movie.imdb_url }}" target="_blank">{{ movie.title }} - IMDb-Seite</a>
            {% endif %}
        </li>
    {% endfor %}
</ul>
<a href="{% url 'movies:add_movie' %}">Film hinzufügen</a>
```

Film hinzufügen

`forms.py`

```
from .models import Movie

class MovieForm(forms.ModelForm):
    class Meta:
        model = Movie
        fields = ['title', 'director', 'release_year', 'genre', 'imdb_url']  # IMDb-URL hinzugefügt
```

`views.py`

```
@login_required(login_url='/login/')
def add_movie(request):
    if request.method == 'POST':
        form = MovieForm(request.POST)
        if form.is_valid():
            movie = form.save(commit=False)
            movie.user = request.user
            movie.save()
            return redirect('movies:watchlist')
    else:
        form = MovieForm()
    return render(request, 'movies/add_movie.html', {'form': form})
```

`templates/movies/add_movie.html`

```
<h2>Film hinzufügen</h2>
<form method="post">
    {% csrf_token %}
    {{ form.as_p }}
    <button type="submit">Hinzufügen</button>
</form>
```

---

4. URLs und Navigation

`urls.py` in der `movies` App

```
from django.urls import path
from . import views

app_name = 'movies'

urlpatterns = [
    path('register/', views.register, name='register'),
    path('login/', views.login_view, name='login'),
    path('logout/', views.logout_view, name='logout'),
    path('watchlist/', views.watchlist_view, name='watchlist'),
    path('add/', views.add_movie, name='add_movie'),
]
```

Navigation hinzufügen

```
<nav>
    <a href="{% url 'movies:watchlist' %}">Meine Watchlist</a> |
    <a href="{% url 'movies:logout' %}">Abmelden</a>
</nav>
```

Zusammenfassung

1. **Benutzerregistrierung und Login**: Sichere Authentifizierung.
2. **Individuelle Watchlist**: Benutzer sehen nur ihre eigenen Filme.
3. **CRUD für Filme**: Filme hinzufügen und anzeigen.

Dieses Projekt bietet eine komplette Lösung für eine benutzerdefinierte Watchlist mit Django.