Movie Recommendation System using Django

Last Updated : 24 Sep, 2024

In this article, we will guide you through the process of creating a comprehensive Movie Recommendation System with the added functionality of user authentication. By incorporating a login system, users can create accounts, personalize their preferences, and access movie recommendations tailored to their tastes. This combination of user authentication and movie recommendations enhances the overall user experience, making the system more engaging and user-friendly.

Movie Recommendation System Using Django

The initial step in our project involves setting up a user authentication system. Django, a high-level Python web framework, provides built-in features for handling user authentication seamlessly. By utilizing Django's authentication views and forms, users can easily register, log in, and manage their accounts.

To install Django follow these steps.

Starting the Project Folder

To start the project use this command

```
django-admin startproject core
cd core
```

To start the app use this command

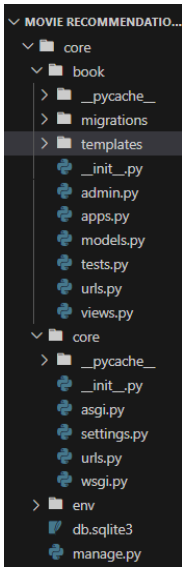
```
python manage.py startapp book
```

Now add this app to the ‘settings.py’

```
INSTALLED_APPS = [
    "django.contrib.admin",
```

```
    "django.contrib.auth",
    "django.contrib.contenttypes",
    "django.contrib.sessions",
    "django.contrib.messages",
    "django.contrib.staticfiles",
    "book",
]
```

File Structure



file structure

Note: A **movie recommendation system** is a great project that combines **Django with machine learning**. To delve further into advanced Django projects like this, the **Django Course - Basics to Advance** is the perfect resource.

Setting Necessary Files

models.py : In This Django code defines two models: `**Emenitites**` and `**Movie**`. The `**Emenitites**` model represents movie-related amenities with a single field for the amenity name. The `Movie` model represents individual movies with fields for the name, description, image URL, price, and a many-to-many relationship with amenities. This structure enables the creation of a Movie Recommendation System where movies can be associated with various amenities, offering a foundation for personalized recommendations based on user preferences and selected amenities.

Python

```python
from django.db import models
from django.contrib.auth.models import User
# Create your models here.
class Emenitites(models.Model):
    name = models.CharField(max_length=100)
    def __str__(self):
        return self.name

class Movie(models.Model):
    movie_name =models.CharField(max_length=100)
    movie_description = models.TextField()
    movie_image = models.CharField(max_length=500)
    price = models.IntegerField()
    emenities = models.ManyToManyField(Emenitites)

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

Views.py : In this Django code implements a Movie Recommendation System with user authentication. The `home` view displays a list of amenities, and the `**api_movies**` view provides movie data with filtering options. User authentication is managed by the `**login_page**`, `**register_page**`, and `**custom_logout**` views. The `**login_required**` decorator ensures authentication for the `**home**` and `**api_movies**` views. The system utilizes Django models for movie-related entities and user authentication, enhancing its functionality.

Python

```python
from django.shortcuts import render, redirect
from .models import Emenitites, Movie
from django.http import JsonResponse
from django.contrib import messages
from django.contrib.auth import login, authenticate
from django.contrib.auth.decorators import login_required
from django.contrib.auth import logout

@login_required(login_url="/login/")
def home(request):
    emenities = Emenitites.objects.all()
```

```
    context = {'emenities': emenities}
    return render(request, 'home.html', context)

@login_required(login_url="/login/")
def api_movies(request):
    movies_objs = Movie.objects.all()

    price = request.GET.get('price')
    if price:
        movies_objs = movies_objs.filter(price__lte=price)

    emenities = request.GET.get('emenities')
    if emenities:
        emenities = [int(e) for e in emenities.split(',') if e.isdigit()]
        movies_objs = movies_objs.filter(emenities__in=emenities).distinct()

    payload = [{'movie_name': movie_obj.movie_name,
                'movie_description': movie_obj.movie_description,
                'movie_image': movie_obj.movie_image,
                'price': movie_obj.price} for movie_obj in movies_objs]

    return JsonResponse(payload, safe=False)

def login_page(request):
```



```
            username = request.POST.get('username')
            password = request.POST.get('password')
            user_obj = User.objects.filter(username=username)

            if not user_obj.exists():
                messages.error(request, "Username not found")
                return redirect('/login/')

            user_obj = authenticate(username=username, password=password)

            if user_obj:
```

```
                login(request, user_obj)
                return redirect('/')

            messages.error(request, "Wrong Password")
            return redirect('/login/')

        except Exception as e:
            messages.error(request, "Something went wrong")
            return redirect('/register/')

    return render(request, "login.html")

def register_page(request):
    if request.method == "POST":
        try:
            username = request.POST.get('username')
            password = request.POST.get('password')
            user_obj = User.objects.filter(username=username)

            if user_obj.exists():
                messages.error(request, "Username is taken")
                return redirect('/register/')

            user_obj = User.objects.create(username=username)
            user_obj.set_password(password)
            user_obj.save()

            messages.success(request, "Account created")
            return redirect('/login/')

        except Exception as e:
            messages.error(request, "Something went wrong")
            return redirect('/register/')

    return render(request, "register.html")

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

Creating GUI

base.html : In this HTML template establishes the structure for a Movie Recommendation System webpage using Bootstrap and Font Awesome. It includes a navigation bar with links for "Home," "Login," and "Register." The main content is encapsulated in a customizable block. The page title is set to "Movie Recommendation System," contributing to a visually appealing layout.

HTML

```html
{% load static %}
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css" rel="stylesheet"
        integrity="sha384-EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTwFspd3yD65VohhpuuCOmLASjC" crossorigin="anonymous">
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.4.0/css/all.min.css">
    <title>Movie Recommendation System</title>
</head>

<body>
    <nav class="navbar navbar-dark bg-warning shadow-lg">
        <div class="container-fluid">
            <a href="{% url 'home' %}" class="navbar-brand"><h2><b>Movie Recommendation System</b></h2></a>
            <div class="d-flex">
                <a href="{% url 'login' %}" class="navbar-brand"><h4>Login</h4></a>
                <a href="{% url 'register' %}" class="navbar-brand"><h4>Register</h4></a>
            </div>
        </div>
    </nav>
```

```
    <h1 class="text-center" style="margin-top: 2%; color:green; font-weight"> GeeksforGeks</h1>
    {% block start %}{% endblock %}
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.bundle.min.js"
        integrity="sha384-MrcW6ZMFYlzcLA8Nl+NtUVF0sA7MsXsP1UyJoMp4YLEuNSfAP+JcXn/tWtIaxVXM"
        crossorigin="anonymous"></script>
</body>

</html>
```

login.html : In this file the below code maintains the structure of a login page that extends the base template. It includes form elements for username and password, along with Bootstrap styling. The unnecessary spaces and redundant styles are removed for conciseness.

Python

```python
{% extends "base.html" %}

{% block start %}
<div class="container mt-5 mx-auto col-md-4 card shadow p-5" style="background-color: #f7f7f7; border-radius: 20px;">
    <div class="login-form">
        {% if messages %}
            {% for message in messages %}
                <div class="alert alert-success {{ message.tags }} mt-4" role="alert">
                    {{ message }}
                </div>
            {% endfor %}
        {% endif %}
        <form action="" method="post">
            {% csrf_token %}
            <h2 class="text-center" style="color: #333;">Log In ????</h2>
            <div class="form-group">
                <input type="text" class="form-control" name="username" placeholder="Username" required
```

```
style="background-color: #fff; border: 1px solid #ddd; border-radius: 5px; padding: 10px;">
            </div>
            <div class="form-group mt-4">
                <input type="password" class="form-control" name="password" placeholder="Password" required style="background-color: #fff; border: 1px solid #ddd; border-radius: 5px; padding: 10px;">
            </div>
            <div class="form-group mt-4">
                <button class="btn btn-success btn-block">Log In ????</button>
            </div>
        </form>
        <p class="text-center" style="color: #555;">Don't have an account? <a href="{% url 'register' %}" style="color: #007bff;">Create an Account </a></p>
    </div>
</div>
{% endblock %}
```

register.html : The below HTML code extends a base template to create a registration page. It features a form for username and password, displays success messages, and includes a link to the login page. The styling uses Bootstrap classes and custom CSS for a visually appealing layout, including a shadowed card and themed buttons.

Python

```python
{% extends "base.html" %}

{% block start %}
<div class="container mt-5 mx-auto col-md-4 card shadow p-5">
    <div class="login-form">
        {% if messages %}
            {% for message in messages %}
                <div class="alert alert-success {{ message.tags }}" role="alert">
                    {{ message }}
                </div>
            {% endfor %}
        {% endif %}
```

```
        <form action="" method="post">
            {% csrf_token %}
            <h2 class="text-center">Register ????</h2>
            <div class="form-group">
                <input type="text" class="form-control" name="username" placeholder="Username" required>
            </div>
            <div class="form-group mt-4">
                <input type="password" class="form-control" name="password" placeholder="Password" required>
            </div>
            <div class="form-group mt-4">
                <button class="btn btn-success btn-block">Register ????</button>
            </div>
        </form>
        <p class="text-center">Already have an account? <a href="{% url 'login' %}">Log In </a></p>
    </div>
</div>

<style>
    .card { background-color: #f7f7f7; border-radius: 20px; }
    .login-form { padding: 20px; }
    .btn-primary { background-color: #007bff; border-color: #007bff; }
    .btn-primary:hover { background-color: #0056b3; border-color: #0056b3; }
    .alert { background-color: #f44336; color: #fff; }
</style>

{% endblock %}
```

home.html : This HTML code creates a Movie Recommendation System web page using Django. It uses Materialize CSS and jQuery for styling and functionality. Users can select movie preferences (amenities and price) to dynamically fetch and display movie recommendations from the Django API. The layout is clean, featuring a GeeksforGeeks header and a logout option.

HTML

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Django Movies</title>
    <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>

    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/materialize/1.0.0/css/materialize.min.css">
    <script src="https://cdnjs.cloudflare.com/ajax/libs/materialize/1.0.0/js/materialize.min.js"></script>
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.4.0/css/all.min.css">

    <style>
        body { background-color: #f5f5f5; }
        .nav-wrapper { background-color: #ffc107; }
        .brand-logo { margin-left: 20px; margin-top: -1%; }
        .container { margin-top: 20px; }
        .card { margin: 10px; width: 100%; box-shadow: 0 4px 8px rgba(0, 0, 0, 0.2); }
        .card-title { font-size: 1.2rem; color: #333; }
        .range-field { padding: 20px 0; }
        .input-field { margin-bottom: 20px; }
        .gfg { margin-left: 40%; font-size: 45px; font-weight: 700; color: green; }
        .card-content { text-align: center; }
        #web { margin-left: 85%; font-size: 20px; font-weight: bold; padding: 5px 20px; border-radius: 9px; background-color: rgb(235, 100, 100); }
        .JT { margin-top: -10%; font-size: 23px; color: black; font-weight: 400; }
        .ex { margin-top: 2%; }
        #oooo { background-color: red; margin-left: 1600px; }
        .ex23 { font-size: 17px; }
        .gfg1 { color: rgb(78, 71, 71); font-size: 25px; font-weight: bold; }
        .gfgd { color: gray; }
```

```
        .btn { padding: 0px 10px; font-weight: bold; }
    </style>
</head>
<body>
    <nav>
        <div class="nav-wrapper">
            <a href="/" class="brand-logo"><h4><b>Movie Recommendation System </b></h4></a>
            <button class="btn btn-danger" id="oooo"> <a href="{% url 'logout' %}">Logout </a></button>
        </div>
    </nav>
    <h1 class="gfg"> GeeksforGeeks</h1>
    <br>
    <br>

    <div class="container">
        <div class="row">
            <div class="col m5">
                <div class="input-field col s12">
                    <select multiple onchange="getMovies()" id="emenities" onchange="getMovies()">
                        <option value="" disabled selected>Choose Your Preference</option>
                        {% for emenitie in emenities %}
                            <option value="{{emenitie.id}}">{{emenitie.name}}</option>
                        {% endfor %}
                    </select>
                    <label for="emenities"><h3 class="JT"><i class="fas fa-suitcase"></i> Select Movie   :</h3> </label>
                </div>
            </div>

            <div class="col m4 ex">
                <label for="price"><h3 class="JT"><i class="fas fa-clock"></i> Short By Price : </h3> </label>
                <p class="range-field">
                    <input type="range" onchange="getMovies()" id="price" min="100" max="10000" value="10" >
                </p>
            </div>
```

```
        </div>
    </div>
    <div class="container">
        <div class="row" id="show_movies_here">
        </div>
    </div>

    <script>
        var show_movies_here = document.getElementById("show_movies_here");

        $(document).ready(function(){
            $('select').formSelect();
        });

        function getMovies() {
            var price = document.getElementById("price").value;
            var instance = M.FormSelect.getInstance(document.getElementById('emenities'));
            var emenities = '';
            var html = '';

            if(instance){
                emenities = instance.getSelectedValues();
            }

            fetch(`/api/movies?emenities=${emenities}&price=${price}`)
            .then(result => result.json())
            .then(response => {
                for (var i = 0; i < response.length; i++) {
                    html += `
                        <div class="col s12 m4">
                            <div class="card">
                                <div class="card-content">
                                    <span class="gfg1">${response[i].movie_name}</span>
                                    <p class="gfgd">${response[i].movie_description}</p>
                                    <p class="ex23">Price: <strong> ₹ ${response[i].price} </strong><p>
                                     <br>
```

```
                                        <button type="submit" class="btn">Book</button>
                                    </div>
                                </div>
                            </div>
                        `;
                    }
                    show_movies_here.innerHTML = html;
                });
            }
            getMovies()
        </script>
</body>
</html>
```

admin.py: Here we are registering our models.

Python

```python
from django.contrib import admin
from book.models import *
# Register your models here.

admin.site.register(Emenitites)
admin.site.register(Movie)
```

Deployement of the Project

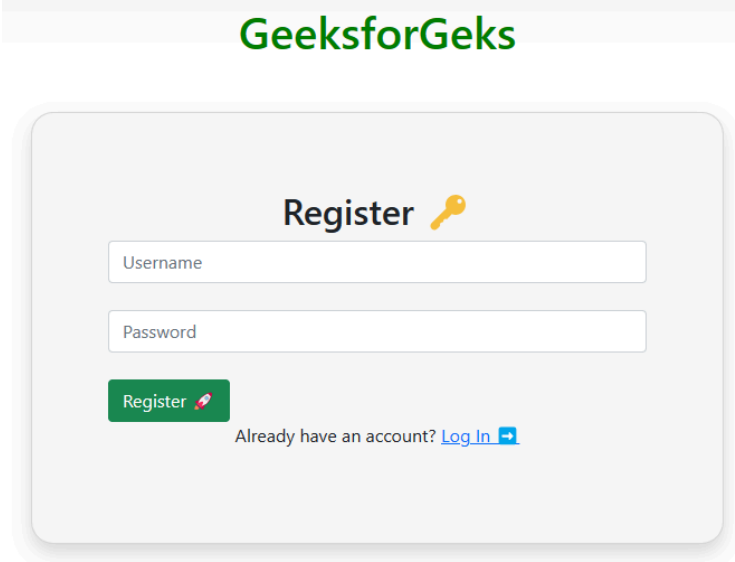
Run these commands to apply the migrations:

```
python3 manage.py makemigrations
python3 manage.py migrate
```

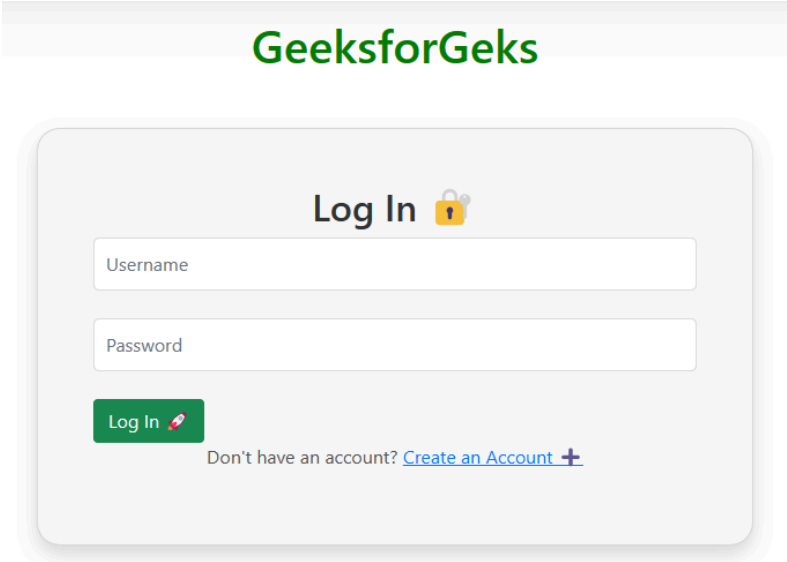
Run the server with the help of following command:

```
python3 manage.py runserver
```

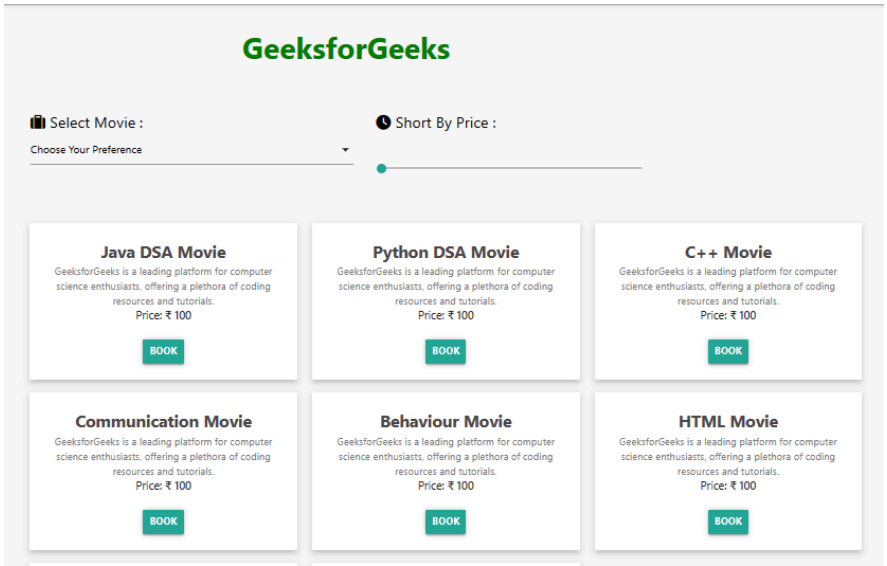
Output:



register.html



login.html



home.html

Video demonstration

D kasot... 14

Previous Article

Setting Up a Virtual Environment in Django

Next Article

Similar Reads

Movie Recommendation System with Node and Express.js

Building a movie recommendation system with Node and Express will help you create personalized suggestions and recommendations according to the genre...

3 min read

Movie recommendation based on emotion in Python

Movies that effectively portray and explore emotions resonate deeply with audiences because they tap into our own emotional experiences and...

4 min read

Python IMDbPY - Retrieving movie using movie ID

In this article we will see how we can retrieve the data of movie using its movie ID, movie id is the unique id given to each movie by IMDb. We can use...

1 min read

Python IMDbPY – Getting released year of movie from movie object

In this article we will see how we can get the released year of movie from the movie object, we can get movie object with the help of search_movie and...

2 min read

Blog Post Recommendation using Django

In this article, we will guide you through the creation of a blog post recommendation system using Django. Our article covers the integration of a...

10 min read

Recipe Recommendation System Using Python

In this article, we will explore how to build a Recipe Recommendation System using Streamlit and OpenAI. We will create the GUI using the Streamlit library...

2 min read

Book Recommendation System using Node and Express.js

The Book Recommendation System aims to enhance the user's reading experience by suggesting books tailored to their interests and preferences....

4 min read

Movie Ticket Booking using Django

In this article, we will create a Movie Ticket Booking system using Django. To begin, we will create the first homepage where we can view all the movies that...

8 min read

Project Idea | Songs Recommendation System in Android

Project Title: Songs Recommendation System in Android Introduction: We all know that in today's era internet is expanding very much and as a result, the dat...

2 min read

Project Idea | Recommendation System based on Graph Database

The main objective of this project is to build an efficient recommendation engine based on graph database(Neo4j). The system aims to be a one stop destination...

1 min read

Article Tags : Django Geeks Premier League Project Python +3 More

Practice Tags : python