



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

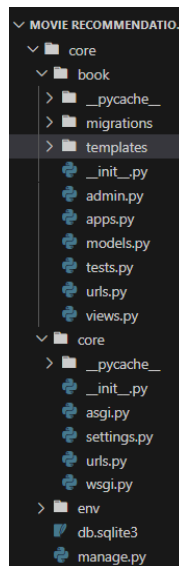
```
python manage.py startapp book
```

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

```

1 from django.db import models
2 from django.contrib.auth.models import User
3 # Create your models here.
4 class Emenitites(models.Model):
5     name = models.CharField(max_length=100)
6     def __str__(self):
7         return self.name
8
9 class Movie(models.Model):
10     movie_name = models.CharField(max_length=100)
11     movie_description = models.TextField()
12     movie_image = models.CharField(max_length=500)
13     price = models.IntegerField()
14     emenities = models.ManyToManyField(Emenitites)
15
16     def __str__(self):
17         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

```

1 from django.shortcuts import render, redirect
2 from .models import Emenitites, Movie
3 from django.http import JsonResponse
4 from django.contrib import messages
5 from django.contrib.auth import login, authenticate
6 from django.contrib.auth.decorators import login_required
7 from django.contrib.auth import logout
8
9 @login_required(login_url="/login/")
10 def home(request):
11     emenities = Emenitites.objects.all()

```

```
        context = {'emenities': emenities}
13     return render(request, 'home.html', context)
14
15 @login_required(login_url="/login/")
16 def api_movies(request):
17     movies_objs = Movie.objects.all()
18
19     price = request.GET.get('price')
20     if price:
21         movies_objs = movies_objs.filter(price__lte=price)
22
23     emenities = request.GET.get('emenities')
24     if emenities:
25         emenities = [int(e) for e in emenities.split(',') if
26 e.isdigit()]
27         movies_objs =
28 movies_objs.filter(emenities__in=emenities).distinct()
29
30     payload = [{'movie_name': movie_obj.movie_name,
31 'movie_description':
32 movie_obj.movie_description,
33 'movie_image': movie_obj.movie_image,
34 'price': movie_obj.price} for movie_obj in
35 movies_objs]
36
37     return JsonResponse(payload, safe=False)
38
39 def login_page(request):
```

Django Views Model Template Forms Jinja Python SQLite Flask Json Postman Interview Ques

```
38     username = request.POST.get('username')
39     password = request.POST.get('password')
40     user_obj =
41 User.objects.filter(username=username)
42
43     if not user_obj.exists():
44         messages.error(request, "Username not
45 found")
46         return redirect('/login/')
47
48     user_obj = authenticate(username=username,
49 password=password)
50
51     if user_obj:
```

```
        login(request, user_obj)
        return redirect('/')
50
51
52     messages.error(request, "Wrong Password")
53     return redirect('/login/')
54
55     except Exception as e:
56         messages.error(request, "Something went wrong")
57         return redirect('/register/')
58
59     return render(request, "login.html")
60
61 def register_page(request):
62     if request.method == "POST":
63         try:
64             username = request.POST.get('username')
65             password = request.POST.get('password')
66             user_obj =
67             User.objects.filter(username=username)
68
69             if user_obj.exists():
70                 messages.error(request, "Username is taken")
71                 return redirect('/register/')
72
73             user_obj =
74             User.objects.create(username=username)
75             user_obj.set_password(password)
76             user_obj.save()
77
78             messages.success(request, "Account created")
79             return redirect('/login/')
80
81         except Exception as e:
82             messages.error(request, "Something went wrong")
83             return redirect('/register/')
84
85     return render(request, "register.html")
86
87 def custom_logout(request):
88     logout(request)
89     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

```
1 {% load static %}
2 <!DOCTYPE html>
3 <html lang="en">
4
5 <head>
6     <meta charset="utf-8">
7     <meta name="viewport" content="width=device-width,
8         initial-scale=1">
9     <link
10         href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/
11         bootstrap.min.css" rel="stylesheet"
12         integrity="sha384-
13         EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTWfSpd3yD65VohhpuuCOmL
14         ASjC" crossorigin="anonymous">
15     <link rel="stylesheet"
16         href="https://cdnjs.cloudflare.com/ajax/libs/font-
17         awesome/6.4.0/css/all.min.css">
18     <title>Movie Recommendation System</title>
19 </head>
20
21 <body>
22     <nav class="navbar navbar-dark bg-warning shadow-lg">
23         <div class="container-fluid">
24             <a href="{% url 'home' %}" class="navbar-brand">
25                 <h2><b>Movie Recommendation System</b></h2></a>
26             <div class="d-flex">
27                 <a href="{% url 'login' %}" class="navbar-
28                 brand"><h4>Login</h4></a>
29                 <a href="{% url 'register' %}"
30                 class="navbar-brand"><h4>Register</h4></a>
31             </div>
32         </div>
33     </nav>
```

```

    <h1 class="text-center" style="margin-top: 2%;
    color:green; font-weight"> GeeksforGeks</h1>
25     {% block start %}{% endblock %}
26     <script
    src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bo
    otstrap.bundle.min.js"
27         integrity="sha384-
    MrcW6ZMFYlzcLA8Nl+NtUVF0sA7MsXsP1UyJoMp4YLEuNSfAP+JcXn/tWtIa
    xVXM"
28         crossorigin="anonymous"></script>
29 </body>
30
31 </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

```

1  {% extends "base.html" %}
2
3  {% block start %}
4  <div class="container mt-5 mx-auto col-md-4 card shadow p-5"
    style="background-color: #f7f7f7; border-radius: 20px;">
5      <div class="login-form">
6          {% if messages %}
7              {% for message in messages %}
8                  <div class="alert alert-success {{
    message.tags }} mt-4" role="alert">
9                      {{ message }}
10                     </div>
11                 {% endfor %}
12             {% endif %}
13             <form action="" method="post">
14                 {% csrf_token %}
15                 <h2 class="text-center" style="color: #333;">Log
    In ????</h2>
16                 <div class="form-group">
17                     <input type="text" class="form-control"
    name="username" placeholder="Username" required

```

```

style="background-color: #fff; border: 1px solid #ddd;
border-radius: 5px; padding: 10px;">
18         </div>
19         <div class="form-group mt-4">
20             <input type="password" class="form-control"
name="password" placeholder="Password" required
style="background-color: #fff; border: 1px solid #ddd;
border-radius: 5px; padding: 10px;">
21         </div>
22         <div class="form-group mt-4">
23             <button class="btn btn-success btn-
block">Log In ???</button>
24         </div>
25     </form>
26     <p class="text-center" style="color: #555;">Don't
have an account? <a href="{% url 'register' %}"
style="color: #007bff;">Create an Account </a></p>
27 </div>
28 </div>
29 {% 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

```

1  {% extends "base.html" %}
2
3  {% block start %}
4  <div class="container mt-5 mx-auto col-md-4 card shadow p-
5  5">
6      <div class="login-form">
7          {% if messages %}
8              {% for message in messages %}
9                  <div class="alert alert-success {{
message.tags }}" role="alert">
10                     {{ message }}
11                 </div>
12             {% endfor %}
13         {% endif %}

```



```

14         <form action="" method="post">
15             {% csrf_token %}
16             <h2 class="text-center">Register ???</h2>
17             <div class="form-group">
18                 <input type="text" class="form-control"
name="username" placeholder="Username" required>
19             </div>
20             <div class="form-group mt-4">
21                 <input type="password" class="form-control"
name="password" placeholder="Password" required>
22             </div>
23             <div class="form-group mt-4">
24                 <button class="btn btn-success btn-
block">Register ???</button>
25             </div>
26             <p class="text-center">Already have an account? <a
href="{% url 'login' %}">Log In </a></p>
27         </div>
28     </div>
29
30     <style>
31         .card { background-color: #f7f7f7; border-radius: 20px;
32         }
33         .login-form { padding: 20px; }
34         .btn-primary { background-color: #007bff; border-color:
#007bff; }
35         .btn-primary:hover { background-color: #0056b3; border-
color: #0056b3; }
36         .alert { background-color: #f44336; color: #fff; }
37     </style>
38     {% 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



```
<!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width,
initial-scale=1.0">
6   <title>Django Movies</title>
7   <script src="https://code.jquery.com/jquery-
3.6.0.min.js"></script>
8
9   <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/materialize/1
.0.0/css/materialize.min.css">
10  <script
src="https://cdnjs.cloudflare.com/ajax/libs/materialize/1.
0.0/js/materialize.min.js"></script>
11  <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/6.4.0/css/all.min.css">
12
13  <style>
14    body { background-color: #f5f5f5; }
15    .nav-wrapper { background-color: #ffc107; }
16    .brand-logo { margin-left: 20px; margin-top: -1%;
}
17    .container { margin-top: 20px; }
18    .card { margin: 10px; width: 100%; box-shadow: 0
4px 8px rgba(0, 0, 0, 0.2); }
19    .card-title { font-size: 1.2rem; color: #333; }
20    .range-field { padding: 20px 0; }
21    .input-field { margin-bottom: 20px; }
22    .gfg { margin-left: 40%; font-size: 45px; font-
weight: 700; color: green; }
23    .card-content { text-align: center; }
24    #web { margin-left: 85%; font-size: 20px; font-
weight: bold; padding: 5px 20px; border-radius: 9px;
background-color: rgb(235, 100, 100); }
25    .JT { margin-top: -10%; font-size: 23px; color:
black; font-weight: 400; }
26    .ex { margin-top: 2%; }
27    #oooo { background-color: red; margin-left:
1600px; }
28    .ex23 { font-size: 17px; }
29    .gfg1 { color: rgb(78, 71, 71); font-size: 25px;
font-weight: bold; }
30    .gfgd { color: gray; }
```

```

        .btn { padding: 0px 10px; font-weight: bold; }
32     </style>
33 </head>
34 <body>
35     <nav>
36         <div class="nav-wrapper">
37             <a href="/" class="brand-logo"><h4><b>Movie
Movie Recommendation System </b></h4></a>
38             <button class="btn btn-danger" id="oooo"> <a
href="{% url 'logout' %}">Logout </a></button>
39         </div>
40     </nav>
41     <h1 class="gfg"> GeeksforGeeks</h1>
42     <br>
43     <br>
44
45     <div class="container">
46         <div class="row">
47             <div class="col m5">
48                 <div class="input-field col s12">
49                     <select multiple
onchange="getMovies()" id="emenities"
onchange="getMovies()">
50                         <option value="" disabled
selected>Choose Your Preference</option>
51                         {% for emenitie in emenities %}
52                             <option value="
{{emenitie.id}}">{{emenitie.name}}</option>
53                         {% endfor %}
54                     </select>
55                     <label for="emenities"><h3 class="JT">
<i class="fas fa-suitcase"></i> Select Movie :</h3>
</label>
56                 </div>
57             </div>
58
59             <div class="col m4 ex">
60                 <label for="price"><h3 class="JT"><i
class="fas fa-clock"></i> Short By Price : </h3> </label>
61                 <p class="range-field">
62                     <input type="range"
onchange="getMovies()" id="price" min="100" max="10000"
value="10" >
63                 </p>
64             </div>

```

```

        </div>
66    </div>
67    <div class="container">
68        <div class="row" id="show_movies_here">
69        </div>
70    </div>
71
72    <script>
73        var show_movies_here =
document.getElementById("show_movies_here");
74
75        $(document).ready(function(){
76            $('select').formSelect();
77        });
78
79        function getMovies() {
80            var price =
document.getElementById("price").value;
81            var instance =
M.FormSelect.getInstance(document.getElementById('emenitie
s')));
82            var emenities = '';
83            var html = '';
84
85            if(instance){
86                emenities = instance.getSelectedValues();
87            }
88
89            fetch(`/api/movies?
emenities=${emenities}&price=${price}`)
90                .then(result => result.json())
91                .then(response => {
92                    for (var i = 0; i < response.length; i++)
93                    {
94                        html += `
95                            <div class="col s12 m4">
96                                <div class="card">
97                                    <div class="card-content">
98                                        <span
class="gfg1">${response[i].movie_name}</span>
99                                        <p
class="gfgd">${response[i].movie_description}</p>
100                                        <p class="ex23">
Price: <strong> ₹ ${response[i].price} </strong><p>
<br>

```

```

                                <button type="submit"
class="btn">Book</button>
102                                </div>
103                                </div>
104                                </div>
105                                `;
106                                }
107                                show_movies_here.innerHTML = html;
108                                });
109                                }
110                                getMovies()
111                                </script>
112                                </body>
113                                </html>

```

admin.py: Here we are registering our models.

Python

```

1  from django.contrib import admin
2  from book.models import *
3  # Register your models here.
4
5  admin.site.register(Ementities)
6  admin.site.register(Movie)

```

Deployment 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:

GeeksforGeks


Register

Already have an account? [Log In](#) 

register.html


GeeksforGeks

Log In


Don't have an account? [Create an Account](#) 

login.html

GeeksforGeeks

 Select Movie :

Choose Your Preference

 Short By Price :



Java DSA Movie

GeeksforGeeks is a leading platform for computer science enthusiasts, offering a plethora of coding resources and tutorials.

Price: ₹ 100

Python DSA Movie

GeeksforGeeks is a leading platform for computer science enthusiasts, offering a plethora of coding resources and tutorials.

Price: ₹ 100

C++ Movie

GeeksforGeeks is a leading platform for computer science enthusiasts, offering a plethora of coding resources and tutorials.

Price: ₹ 100

Communication Movie

GeeksforGeeks is a leading platform for computer science enthusiasts, offering a plethora of coding resources and tutorials.

Price: ₹ 100

Behaviour Movie

GeeksforGeeks is a leading platform for computer science enthusiasts, offering a plethora of coding resources and tutorials.

Price: ₹ 100

HTML Movie

GeeksforGeeks is a leading platform for computer science enthusiasts, offering a plethora of coding resources and tutorials.

Price: ₹ 100

home.html

Video demonstration

Are you ready to elevate your web development skills from foundational knowledge to advanced expertise? Explore our [Mastering Django Framework - Beginner to Advanced Course](#) on GeeksforGeeks, designed for aspiring developers and experienced programmers. This comprehensive course covers everything you need to know about Django, from the basics to advanced features. Gain practical experience through **hands-on projects** and real-world applications, mastering essential Django principles and techniques. Whether you're just starting or looking to refine your skills, this course will empower you to build sophisticated web applications efficiently. Ready to enhance your web development journey? Enroll now and unlock your potential with Django!

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



Corporate & Communications Address:-
A-143, 9th Floor, Sovereign Corporate
Tower, Sector- 136, Noida, Uttar Pradesh
(201305) | Registered Address:- K 061,
Tower K, Gulshan Vivante Apartment,
Sector 137, Noida, Gautam Buddh
Nagar, Uttar Pradesh, 201305



Company

- About Us
- Legal
- In Media
- Contact Us
- Advertise with us
- GFG Corporate Solution
- Placement Training Program
- GeeksforGeeks Community

Languages

- Python
- Java
- C++
- PHP
- GoLang
- SQL
- R Language
- Android Tutorial
- Tutorials Archive

DSA

- Data Structures
- Algorithms
- DSA for Beginners
- Basic DSA Problems
- DSA Roadmap

Data Science & ML

- Data Science With Python
- Data Science For Beginner
- Machine Learning
- ML Maths
- Data Visualisation

[Top 100 DSA Interview Problems](#)[DSA Roadmap by Sandeep Jain](#)[All Cheat Sheets](#)[Pandas](#)[NumPy](#)[NLP](#)[Deep Learning](#)

Web Technologies

[HTML](#)[CSS](#)[JavaScript](#)[TypeScript](#)[ReactJS](#)[NextJS](#)[Bootstrap](#)[Web Design](#)

Python Tutorial

[Python Programming Examples](#)[Python Projects](#)[Python Tkinter](#)[Web Scraping](#)[OpenCV Tutorial](#)[Python Interview Question](#)[Django](#)

Computer Science

[Operating Systems](#)[Computer Network](#)[Database Management System](#)[Software Engineering](#)[Digital Logic Design](#)[Engineering Maths](#)[Software Development](#)[Software Testing](#)

DevOps

[Git](#)[Linux](#)[AWS](#)[Docker](#)[Kubernetes](#)[Azure](#)[GCP](#)[DevOps Roadmap](#)

System Design

[High Level Design](#)[Low Level Design](#)[UML Diagrams](#)[Interview Guide](#)[Design Patterns](#)[OOAD](#)[System Design Bootcamp](#)[Interview Questions](#)

Interview Preparation

[Competitive Programming](#)[Top DS or Algo for CP](#)[Company-Wise Recruitment Process](#)[Company-Wise Preparation](#)[Aptitude Preparation](#)[Puzzles](#)

School Subjects

[Mathematics](#)[Physics](#)[Chemistry](#)[Biology](#)[Social Science](#)[English Grammar](#)[Commerce](#)[World GK](#)

GeeksforGeeks Videos

[DSA](#)[Python](#)[Java](#)[C++](#)[Web Development](#)[Data Science](#)[CS Subjects](#)