**🔹 Authentication (আমি কে?)**

👉 ব্যবহারকারী কে সেটা যাচাই করার প্রক্রিয়া।  
মানে—

* তুমি কি আসলেই এই সিস্টেমের **valid user**?
* লগইন করার সময় তোমার **username + password/email/OTP/fingerprint** মিলিয়ে দেখে।

✅ Example:

* তুমি Facebook এ লগইন করছো।
* সিস্টেম প্রথমে চেক করবে তোমার email/phone + password ঠিক আছে কিনা।
* যদি ঠিক থাকে → তুমি **authenticated**।

**🔹 Authorization (আমি কী করতে পারব?)**

👉 ব্যবহারকারী **কি কাজ করতে পারবে আর কি পারবে না** সেটা নির্ধারণ করার প্রক্রিয়া।  
মানে—

* একজন user system-এ ঢুকতে পারলেও, তার কাছে কি কি access আছে সেটা দেখা।

✅ Example:

* Facebook এ লগইন করার পর তুমি শুধু তোমার profile edit করতে পারবে, কিন্তু অন্য কারো profile edit করতে পারবে না।
* Admin role থাকলে তুমি হয়তো অন্যদের account ban করতে পারবে।
* সাধারণ user সেটা করতে পারবে না।

**🔥 মূল পার্থক্য**

| **বিষয়** | **Authentication** | **Authorization** |
| --- | --- | --- |
| মানে | User কে চেক করা (identity verify) | User কী করতে পারবে সেটা চেক করা (permissions) |
| কবে হয় | **Before** system এ ঢোকা | **After** system এ ঢোকার পর |
| প্রমাণ | Password, OTP, Fingerprint | Roles, Permissions, Access control |
| Example | লগইন করা | Admin panel এ ঢুকতে পারা |

**🚪 সহজ উদাহরণ (বাসা analogy):**

* **Authentication** → দরজায় কলিং বেল দিলে guard জিজ্ঞেস করল "আপনি কে?" (identity verify করল)।
* **Authorization** → ভেতরে ঢোকার পর আপনি drawing room এ যেতে পারবেন, কিন্তু master bedroom এ অনুমতি নাই (permission control)
* একদম ঠিক আছে ভাই 👍  
  আমি তোমাকে **Register (Sign Up) + Login** পুরোটা সহজভাবে ব্যাখ্যা করি,  
  যাতে তুমি খাতায় নোট নিতে পারো।

**Register (Sign Up)**

**১) forms.py**

(আমরা নিজের ModelForm বানাই – UserCreationForm ব্যবহার করলেও পারতে)

from django import forms

from django.contrib.auth.models import User

from django.contrib.auth.forms import UserCreationForm

class RegisterForm(UserCreationForm):

class Meta:

model = User

fields = ["username", "email", "password1", "password2"]

**২) views.py**

from django.shortcuts import render, redirect

from .forms import RegisterForm

from django.contrib.auth import login

def register\_user(request):

form = RegisterForm()

if request.method == "POST":

form = RegisterForm(request.POST)

if form.is\_valid():

user = form.save()

login(request, user) *# রেজিস্টার হলে লগইন করে দিবে*

return redirect("home")

return render(request, "register.html", {"form": form})

**৩) register.html**

{% extends "base.html" %}

{% load crispy\_forms\_tags %}

{% block content %}

<div class="container">

<div class="card p-4">

<h3 class="text-center">Sign Up</h3>

<form method="post">{% csrf\_token %}

{{ form|crispy }}

<button class="btn btn-primary w-100 mt-3">Create Account</button>

</form>

<p class="text-center mt-2">Already have account? <a href="{% url 'login' %}">Login</a></p>

</div>

</div>

{% endblock %}

**Login/CustomForm**

**username diye[Default]**

**১) forms.py (optional)**

👉 চাইলে AuthenticationForm সরাসরি ব্যবহার করতে পারো।

from jango.contrib.auth.forms import AuthenticationForm

অথবা কাস্টমাইজ করতে চাইলে:

from jango import forms

from jango.contrib.auth.forms import AuthenticationForm

class CustomLoginForm(AuthenticationForm):

username = forms.CharField(widget=forms.TextInput(attrs={‘placeholder’: ‘Username’}))

password = forms.CharField(widget=forms.PasswordInput(attrs={‘placeholder’: ‘Password’}))

* **২) views.py**
* AuthenticationForm():

def user\_login(request):

    form = AuthenticationForm()

    if request.method == "P“ST":”        form = AuthenticationForm( data=request.POST)

        if form.is\_valid():

            username = form.cleaned\_data.get('u‘ername')’            password = form.cleaned\_data.get('p‘ssword')’            user = authenticate(request, username=username, password=password)

            if user is not None:

                login(request, user)

                messages.success(request, "L“gged in successfully!")”                return redirect("h“me")”        messages.error(request, "I“valid username or password.")”    return render(request, "s“udent/login.html",”{"f“rm":”form, "s“gnup\_bar":”True})

from djjangoontrib.auth.forms import AuthenticationForm

from djjangoontrib.auth import login

def login\_user(request):

form = CustomLoginForm() *# এখানে CustomLoginForm ব্যবহার করছি*

if request.method == "P“ST":” form = CustomLoginForm(data=request.POST)

if form.is\_valid():

user = form.get\_user()

login(request, user)

return redirect("h“me")” return render(request, "l“gin.html",”{"f“rm":”form})

**৩) login.html**

{% extends "b“se.html" ”}

{% load crispy\_forms\_tags %}

{% block content %}

<div class="c”ntainer">” <div class="c”rd p-4">” <h3 class="t”xt-center">”ogin</h3>

<form method="p”st">”% csrf\_token %}

{{ form|crispy }}

<button class="b”n btn-success w-100 mt-3">”ogin</button>

</form>

<p class="t”xt-center mt-2">”on’t have account? <a href="{” url 'r‘gister' ’}">”ign Up</a></p>

</div>

</div>

{% endblock %}

**🔹 urls.py**

from djjangorls import path

from . import views

urlpatterns = [

path("r“gister/",”views.register\_user, name="r”gister")”

path("l“gin/",”views.login\_user, name="l”gin")”

]

**🔹 settings.py (optional)**

LOGIN\_REDIRECT\_URL = "h“me" ” *# লগইন/রেজিস্টার শেষে home এ যাবে*

LOGOUT\_REDIRECT\_URL = "l“gin"

”

**Email diye**

**ঠিক আছে ভাই 😎, তাহলে আমরা শুধু Email দিয়ে login হবে এমন সহজ ও safe ভার্সন বানাই।**

1️⃣ forms.py

from django import forms

class EmailLoginForm(forms.Form):

email = forms.EmailField(

label="Email Address",

required=True,

widget=forms.EmailInput(attrs={

'class': 'form-control',

'placeholder': 'Enter your registered email',

})

)

password = forms.CharField(

label="Password",

required=True,

widget=forms.PasswordInput(attrs={

'class': 'form-control',

'placeholder': 'Enter your password',

})

)

2️⃣ views.py

from django.shortcuts import render, redirect

from django.contrib.auth import authenticate, login

from django.contrib.auth.models import User

from django.contrib import messages

from .forms import EmailLoginForm

def login\_with\_email(request):

if request.method == "POST":

form = EmailLoginForm(request.POST)

if form.is\_valid():

email = form.cleaned\_data['email']

password = form.cleaned\_data['password']

try:

*# Email দিয়ে user খুঁজে বের করা*

user\_obj = User.objects.get(email=email)

*# authenticate requires username, তাই username নিয়ে check*

user = authenticate(username=user\_obj.username, password=password)

if user is not None:

login(request, user)

messages.success(request, "✅ Login successful!")

return redirect("home")

else:

messages.error(request, "❌ Wrong password")

except User.DoesNotExist:

messages.error(request, "❌ Email not found")

else:

form = EmailLoginForm()

return render(request, "student/login\_with\_email.html", {"form": form})

3️⃣ login\_with\_email.html

{% extends 'base.html' %}

{% load crispy\_forms\_tags %}

{% block content %}

<div class="d-flex justify-content-center align-items-center vh-100">

  <div class="card shadow-lg p-4 rounded-4" style="max-width: 400px; width: 100%;">

    <h3 class="text-center mb-4">🔐 Login</h3>

    <form method="post">

      {% csrf\_token %}

      {{ form|crispy }}

      <button class="btn btn-secondary w-100 mt-3">Login</button>

    </form>

    <p class="text-center mt-3 mb-0">

      Don’t have account?

      <a href="{% url 'signup' %}" class="text-decoration-none">Sign Up</a>

    </p>

  </div>

</div>

{% endblock %}

⚡ কী হবে?

* শুধু email + password দিয়েই login হবে।

**Login Required**

login\_required সহজভাবে বললে 👉  
এটা Django-র একটা **decorator** যেটা ব্যবহার করলে কোনো view কেবলমাত্র **লগইন করা ইউজার**-রা অ্যাক্সেস করতে পারবে।

**উদাহরণ ছাড়া বুঝতে অসুবিধা হয়, তাই ধাপে ধাপে বলি:**

🔹 ধরো তোমার একটা view আছে:

def profile(request):

return render(request, "profile.html")

এখন যেকেউ /profile/ URL লিখলেই পেজটা দেখতে পারবে – এমনকি লগইন না করলেও।

**এখন login\_required ব্যবহার করলে:**

from django.contrib.auth.decorators import login\_required

@login\_required

def profile(request):

return render(request, "profile.html")

👉 এখন /profile/ দেখতে চাইলে আগে লগইন করতে হবে।  
যদি ইউজার লগইন করা না থাকে, Django স্বয়ংক্রিয়ভাবে তাকে **লগইন পেইজে রিডাইরেক্ট** করবে।

**কাস্টমাইজ (লগইন URL ঠিক করা)**

ডিফল্টে এটা /accounts/login/ এ রিডাইরেক্ট করে।  
তুমি চাইলে নিজে নির্দিষ্ট করতে পারো:

@login\_required(login\_url="/login/")

def profile(request):

return render(request, "profile.html")

👉 **সহজ ভাষায়**:  
login\_required = *“এই পেজে ঢুকতে হলে আগে লগইন করতেই হবে”* 🚪🔑

**OR[Easy and Better]**

সাধারণত আমরা এভাবে লিখি:

@login\_required(login\_url="/login/")

def profile(request):

...

কিন্তু প্রতিবার login\_url="/login/" লিখতে ঝামেলা লাগে।

**✅ সমাধান**

তুমি যদি settings.py-তে **LOGIN\_URL** সেট করে দাও,  
তাহলে আর view-তে আলাদা করে login\_url লিখতে হবে না।

**settings.py-তে যোগ করো:**

*# settings.py*

*# যখন login\_required detect করবে যে ইউজার লগইন না,*

*# তখন কোন URL-এ পাঠাবে:*

LOGIN\_URL = "/login/"

*# সফল লগইনের পর কোন পেজে পাঠাবে:*

LOGIN\_REDIRECT\_URL = "/"

*# লগআউটের পর কোন পেজে পাঠাবে:*

LOGOUT\_REDIRECT\_URL = "/login/"

**এখন view.py তে শুধু এভাবে লিখলেই হবে:**

from django.contrib.auth.decorators import login\_required

@login\_required

def profile(request):

return render(request, "profile.html")

👉 এখানে আর login\_url="/login/" আলাদা করে লেখার দরকার নেই।  
কারণ Django স্বয়ংক্রিয়ভাবে **settings.py → LOGIN\_URL** ব্যবহার করবে।

মানে দাঁড়ালো –  
🔑 শুধু @login\_required লিখলেই হবে, বাকি কাজ **settings.py** করবে। 🚀

**How authenticated user can do CRUD operations**

**1) Model (প্রথমে টেবিলটা ঠিক করো)**

Student মডেলে user রাখতে হবে যাতে কে create করেছে সেটা ট্র্যাক হয়।

*# models.py*

from django.conf import settings

from django.db import models

class Student(models.Model):

user = models.ForeignKey(settings.AUTH\_USER\_MODEL, on\_delete=models.CASCADE)

name = models.CharField(max\_length=100)

photo = models.ImageField(upload\_to="students/", blank=True, null=True)

*# আর দরকারি ফিল্ডগুলো...*

def \_\_str\_\_(self):

return self.name

**মাইগ্রেশন চালাও**  
python manage.py makemigrations → python manage.py migrate

**2) Form (user ফিল্ডটা ফর্মে দেখাবে না)**

user আমরা view থেকে সেট করবো, তাই ফর্মে রাখবে না।

*# forms.py*

from django import forms

from .models import Student

class StudentForm(forms.ModelForm):

class Meta:

model = Student

fields = ["name", "photo"] *# ইচ্ছে হলে exclude = ["user"] ব্যবহার করতে পারো*

**3) URL**

*# urls.py*

from django.urls import path

from . import views

urlpatterns = [

path("students/create/", views.create\_student, name="create\_student"),

]

**4) View (লাইন–বাই–লাইন বোঝা)**

*# views.py*

from django.shortcuts import render, redirect

from django.contrib import messages

from django.contrib.auth.decorators import login\_required

from . import forms

@login\_required

def create\_student(request):

if request.method == "POST": *# 1) ফর্ম সাবমিট হলে*

form = forms.StudentForm(request.POST, request.FILES) *# 2) ডাটা ধরো (FILES লাগবে ছবি/ফাইলের জন্য)*

if form.is\_valid(): *# 3) সব ভ্যালিড?*

student = form.save(commit=False) *# 4) এখনই DB-তে না, হাতে নিয়ন্ত্রণ নিলাম*

student.user = request.user *# 5) লগইন করা ইউজার সেট করলাম*

student.save() *# 6) এবার সেভ করলাম*

*# যদি ফর্মে ManyToMany ফিল্ড থাকে, তখন:*

*# form.save\_m2m()*

messages.success(request, "Student created successfully!") *# 7) ফিডব্যাক মেসেজ*

return redirect("home") *# 8) কাজ শেষে কোথায় যাবে*

else:

form = forms.StudentForm() *# 9) GET হলে খালি ফর্ম*

return render(request, "student/create\_student.html", {"form": form}) *# 10) ফর্ম দেখাও*

**কেন commit=False?**  
কারণ user ফিল্ডটা আগে সেট করতে হবে; সরাসরি form.save() করলে user ছাড়া সেভ হবে না। তাই আগে অবজেক্টটা ধরি, user বসাই, তারপর save()।

**5) Template (CSRF + enctype ভুলবে না)**

*<!-- templates/student/create\_student.html -->*

{% load crispy\_forms\_tags %} *<!-- থাকলে ব্যবহার করো, না থাকলে normal form.as\_p -->*

*<!-- success/error messages দেখাতে -->*

{% if messages %}

<ul>

{% for message in messages %}

<li>{{ message }}</li>

{% endfor %}

</ul>

{% endif %}

<form method="post" enctype="multipart/form-data">

{% csrf\_token %}

{{ form|crispy }} *<!-- crispy না থাকলে: {{ form.as\_p }} -->*

<button type="submit">Create</button>

</form>

enctype="multipart/form-data" না দিলে ছবি/ফাইল যাবে না।

**6) Settings (login auto-redirect কাজ করার জন্য)**

*# settings.py*

LOGIN\_URL = "/login/"

LOGIN\_REDIRECT\_URL = "/"

LOGOUT\_REDIRECT\_URL = "/login/"

এগুলো থাকলে @login\_required ইউজার লগইন না থাকলে LOGIN\_URL এ পাঠাবে—view-তে আলাদা login\_url লিখতে হবে না।

**7) Common pitfalls (ছোটখাটো ভুল যেন না হয়)**

* Student.user যদি null=False (ডিফল্ট), তাহলে **অবশ্যই** commit=False নিয়ে request.user সেট করতে হবে।
* টেমপ্লেটে enctype না দিলে request.FILES পেলেও ছবি সেভ হবে না।
* messages দেখাতে টেমপ্লেটে messages লুপ থাকতে হবে।
* যদি ফর্মে **ManyToMany** ফিল্ড থাকে, student.save()–এর পর **form.save\_m2m()** কল দাও।

এভাবেই পুরো ফ্লোটা কাজ করবে:  
URL → Login check (@login\_required) → Form submit → commit=False → student.user = request.user → Save → Success message → Redirect ✅

আর কিছু লাগলে বলো—যেমন login/logout view, বা home টেমপ্লেটের ছোট সেটআপও দিয়ে দিতে পারি।

**Jodi Update Ke korse set korte Chaile**

আচ্ছা ভাই, একদম clear 👍  
তুমি student list template দেখাচ্ছো — যেখানে photo, name, email, phone, edit/delete আছে।  
এখন তুমি চাইছো **update করলে কে update করলো সেটাও list এ show হোক**।

**✅ Step 1: Model এ updated\_by ফিল্ড থাকতে হবে**

আগে যেমন বলেছি:

*# models.py*

updated\_by = models.ForeignKey(

settings.AUTH\_USER\_MODEL,

on\_delete=models.SET\_NULL,

null=True,

blank=True,

related\_name="updated\_students"

)

**✅ Step 2: Update view এ updated\_by সেট করো**

@login\_required

def update\_student(request, pk):

student = get\_object\_or\_404(Student, pk=pk)

if request.method == "POST":

form = StudentForm(request.POST, request.FILES, instance=student)

if form.is\_valid():

student = form.save(commit=False)

student.updated\_by = request.user *# <-- কে update করলো সেট হলো*

student.save()

messages.success(request, "Student updated successfully!")

return redirect("student\_list") *# student\_list বা অন্য url এ redirect*

else:

form = StudentForm(instance=student)

return render(request, "student/update\_student.html", {"form": form})

**✅ Step 3: Template এ show করো**

তুমি যে list template পাঠালে, তার ভেতরে updated\_by দেখানোর জন্য শুধু extra লাইন যোগ করতে হবে।  
তাহলে তোমার template হবে এভাবে:

{% extends 'base.html' %}

{% block content %}

<div class="container mt-3">

<h1 class="mb-4 fw-bold text-center">All Student List</h1>

<div class="d-flex flex-wrap justify-content-start">

{% for student in students %}

<div class="card student-card m-3">

<a href="{% url 'student\_detail' student.id %}">

<img src="{{ student.photo.url }}" class="card-img-top student-img" alt="Student Photo" />

</a>

<div class="card-body text-center">

<h5 class="card-title fw-bold">{{ student.name }}</h5>

<p class="text-muted mb-1">{{ student.email }}</p>

<p class="mb-1">

<a class="text-decoration-none" href="tel:{{ student.phone }}">{{ student.phone }}</a>

</p>

<!-- 👇 কে update করেছে সেটা দেখানো -->

{% if student.updated\_by %}

<p class="small text-success mb-2">Last updated by: {{ student.updated\_by.username }}</p>

{% else %}

<p class="small text-muted mb-2">Never updated</p>

{% endif %}

<div class="d-flex justify-content-center gap-2">

<a href="{% url 'update\_student' student.id %}" class="btn btn-sm btn-outline-primary edit-btn">✏️ Edit</a>

<a href="{% url 'delete\_student' student.id %}" class="btn btn-sm btn-outline-danger delete-btn">🗑 Delete</a>

</div>

</div>

</div>

{% endfor %}

</div>

</div>

{% endblock %}

👉 এখন প্রতিটি student card এর নিচে **“Last updated by: username”** দেখাবে।  
যদি কোনো ইউজার এখনও আপডেট না করে, তাহলে **“Never updated”** দেখাবে।

**Reset password**

**📒 Password Reset Notes (Django)**

**🔹 1. Password Change vs Password Reset**

* **Password Change** → যখন user login করা থাকে তখন নিজের password change করতে পারে।
* **Password Reset** → যখন user password ভুলে যায়, তখন email এর মাধ্যমে নতুন password সেট করে।

তুমি এখানে **Password Reset system** implement করেছ।

**🔹 Password Reset Process (4 Steps in Django)**

Django এর built-in **auth views** ব্যবহার করলে password reset flow ৪টা ধাপে সম্পন্ন হয়।

**1️⃣ Password Reset Request (User email দেয়)**

path("password-reset/",

auth\_views.PasswordResetView.as\_view(),

name="password\_reset"),

* এখানে user তার **registered email** দিবে।
* Django ওই email এ **password reset link** পাঠাবে।
* Default template → registration/password\_reset\_form.html

**2️⃣ Password Reset Done (Email sent confirmation)**

path("password-reset/done/",

auth\_views.PasswordResetDoneView.as\_view(),

name="password\_reset\_done"),

* User কে দেখানো হবে: “We’ve sent you an email to reset your password.”
* Default template → registration/password\_reset\_done.html

**3️⃣ Password Reset Confirm (New password set করা)**

path("reset/<uidb64>/<token>/",

auth\_views.PasswordResetConfirmView.as\_view(),

name="password\_reset\_confirm"),

* User email এ যেই **reset link** পাবে, সেখানে uidb64 এবং token থাকবে।
* ওই লিংক ক্লিক করলে নতুন password set করার form আসবে।
* Default template → registration/password\_reset\_confirm.html

**4️⃣ Password Reset Complete (Success message)**

path("reset/done/",

auth\_views.PasswordResetCompleteView.as\_view(),

name="password\_reset\_complete"),

* Password successfully reset হলে success page show করবে।
* Default template → registration/password\_reset\_complete.html

**🔹 Templates Required (Default Paths)**

তুমি চাইলে নিজে template বানাতে পারো। Django ডিফল্ট ভাবে নিচের template খোঁজে:

1. registration/password\_reset\_form.html
2. registration/password\_reset\_done.html
3. registration/password\_reset\_confirm.html
4. registration/password\_reset\_complete.html

**🔹 Email Backend Setup**

Password reset কাজ করার জন্য Django কে email পাঠাতে হবে। এজন্য settings.py এ **Email Configuration** দিতে হবে। Example:

EMAIL\_BACKEND = "django.core.mail.backends.console.EmailBackend"

DEFAULT\_CHARSET = "utf-8"

👉 Gmail ব্যবহার করলে অবশ্যই **App Password** generate করতে হবে।

**🔹 Final Flow (Summary)**

1. User → password-reset/ → Email দেয়
2. Django → Email এ Reset Link পাঠায়
3. User → Link এ ক্লিক করে → নতুন Password set করে
4. Password reset complete → Success message

**1️⃣ password\_reset\_form.html**

(User email দিয়ে reset request করবে)

{% extends "base.html" %}

{% block content %}

<div class="container mt-5">

<div class="card shadow-lg p-4">

<h3 class="text-center mb-3">🔑 Reset Your Password</h3>

<p class="text-muted text-center">Enter your registered email address, we’ll send you instructions.</p>

<form method="post">

{% csrf\_token %}

{{ form.email.label\_tag }}

{{ form.email }}

<button type="submit" class="btn btn-primary w-100 mt-3">Send Reset Link</button>

</form>

</div>

</div>

{% endblock %}

**2️⃣ password\_reset\_done.html**

(Email sent confirmation)

{% extends "base.html" %}

{% block content %}

<div class="container mt-5">

<div class="alert alert-info shadow">

✅ We have sent you an email with instructions to reset your password.

Please check your inbox.

</div>

<a href="{% url 'login' %}" class="btn btn-success">Back to Login</a>

</div>

{% endblock %}

**3️⃣ password\_reset\_confirm.html**

(User নতুন password দিবে)

{% extends "base.html" %}

{% block content %}

<div class="container mt-5">

<div class="card shadow-lg p-4">

<h3 class="text-center mb-3">🔐 Set a New Password</h3>

{% if validlink %}

<form method="post">

{% csrf\_token %}

{{ form.as\_p }}

<button type="submit" class="btn btn-primary w-100 mt-3">Reset Password</button>

</form>

{% else %}

<div class="alert alert-danger">

❌ The password reset link is invalid, possibly because it has already been used.

</div>

{% endif %}

</div>

</div>

{% endblock %}

**4️⃣ password\_reset\_complete.html**

(Success message after reset complete)

{% extends "base.html" %}

{% block content %}

<div class="container mt-5">

<div class="alert alert-success shadow">

🎉 Your password has been successfully reset!

You can now login with your new password.

</div>

<a href="{% url 'login' %}" class="btn btn-primary">Go to Login</a>

</div>

{% endblock %}

**🔹 Extra Tips**

* যদি তুমি **Crispy Forms** ব্যবহার কর, তাহলে শুধু {{ form|crispy }} লিখে দাও → automatically সুন্দর design হয়ে যাবে।
* Templates extend করার জন্য base.html এ Bootstrap CDN থাকতে হবে।

**Search Bar**

def TaskList(request):

task = Task.objects.filter(user=request.user) *# user এর task filter*

search\_input = request.GET.get('search-area') or '' *# search input নাও*

if search\_input:

task = task.filter(title\_\_icontains=search\_input) *# title অনুযায়ী filter*

search\_input = search\_input *# এখানে technically লাগে না, শুধু context এ পাঠাতে হবে*

return render(request, 'todo\_app\_list.html', {'tasks': task})

**🔹 কিছু ছোট টিপস:**

1. search\_input = search\_input **লাজিম না**, সরাসরি context এ পাঠাতে পারো।
2. Template এ value="{{ search\_input }}" দিলে search box এ আগের লেখা থাকবে।

**✅ Updated clean version:**

def TaskList(request):

tasks = Task.objects.filter(user=request.user)

search\_input = request.GET.get('search-area', '') *# safer default*

if search\_input:

tasks = tasks.filter(title\_\_icontains=search\_input)

return render(request, 'todo\_app\_list.html', {'tasks': tasks, 'search\_input': search\_input})

* এখন template এ search box এ আগের query auto দেখাবে।
* Task list properly filtered হবে।

<form *action*="" *method*="get" *class*="flex items-center space-x-0">

        <input *type*="text" *name*="search-area" *placeholder*="Search..."

*value*="{{search\_input}}"

*class*="px-3 py-2 rounded-l-md border border-gray-300 focus:outline-none focus:ring-2 focus:ring-blue-400 text-black w-40 md:w-64 transition">

        <button *type*="submit"

*class*="bg-white text-blue-600 px-4 py-2 rounded-r-md font-semibold hover:bg-gray-200 transition">

          Search

        </button>

 </form>