Workflow for Building a Django Form

1. Create the Form Class

Create a `forms.py` file in your app (if it doesn't already exist). Define a form class using Django's `forms.Form` or `forms.ModelForm` (if tied to a model).

python

```python
from django import forms

class ReservationForm(forms.Form):
    name = forms.CharField(max_length=100)
    email = forms.EmailField()
    date = forms.DateField(widget=forms.SelectDateWidget)
    time = forms.TimeField(widget=forms.TimeInput(format='%H:%M'))
    party_size = forms.IntegerField()
```

2. Add or Update the Template (`form.html`)

Create an HTML file for rendering the form in your app's `templates/` directory. Use Django's templating syntax to render the form.

html

```html
<!-- templates/form.html -->
<h1>Make a Reservation</h1>
<form method="post" action="{% url 'reservation_submit' %}">
    {% csrf_token %}
    {{ form.as_p }}
    <button type="submit">Submit</button>
</form>
```

3. Create a Success Template (`success.html`)

Add a `success.html` template to display a confirmation message after the form is successfully submitted.

html

```html
<!-- templates/success.html -->
<h1>Reservation Successful</h1>
<p>Thank you, your reservation has been submitted!</p>
```

4. Add a View for the Form

Create a view function (or class-based view) in your app's `views.py` to handle GET and POST requests for the form.

```python
from django.shortcuts import render, redirect
from .forms import ReservationForm

def reservation_view(request):
    if request.method == 'POST':
        form = ReservationForm(request.POST)
        if form.is_valid():
            # Process form data (e.g., save to database, send email, etc.)
            return redirect('success')
    else:
        form = ReservationForm()

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

def success_view(request):
    return render(request, 'success.html')
```

5. Add URL Patterns

Define URLs in your app's `urls.py` to map the form view and success page.

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

urlpatterns = [
    path('reservation/', views.reservation_view, name='reservation_submit'),
    path('success/', views.success_view, name='success'),
]
```

6. Use ModelForm for Database Integration

If your form data corresponds to a database model, use `forms.ModelForm`.

```python
from .models import Reservation

class ReservationForm(forms.ModelForm):
    class Meta:
        model = Reservation
        fields = ['name', 'email', 'date', 'time', 'party_size']
```

Example view:

```python
if form.is_valid():
    form.save()  # Saves the form data to the database
```

7. Create a Model

Define a model for your reservation data if you're using `ModelForm`.

```python
from django.db import models

class Reservation(models.Model):
    name = models.CharField(max_length=100)
    email = models.EmailField()
    date = models.DateField()
    time = models.TimeField()
    party_size = models.IntegerField()
```

8. Style the Form Using Bootstrap and SCSS

Use Bootstrap classes and SCSS to style your form fields.

python

```
class ReservationForm(forms.Form):
    name = forms.CharField(widget=forms.TextInput(attrs={'class': 'form-control'}))
    email = forms.EmailField(widget=forms.EmailInput(attrs={'class': 'form-control'}))
    date = forms.DateField(widget=forms.SelectDateWidget(attrs={'class': 'form-select'}))
    time = forms.TimeField(widget=forms.TimeInput(attrs={'class': 'form-control'}))
    party_size = forms.IntegerField(widget=forms.NumberInput(attrs={'class': 'form-control'}))
```

SCSS Example:

scss

```
// _form.scss
.form-control {
    border-radius: 5px;
}
```

9. Test the Workflow

Run the server and navigate to the form URL to test it.

Example command:

bash

```
python manage.py runserver
```

10. Include the Form in the Base Template

If you want the form to appear on multiple pages, include it in a base template.

Example:

html

```
{% include 'form.html' %}
```