**Module 4 Cheat Sheet: Consolidate and Deploy Your Django App**

| **Package/ Method** | **Description** | **Code Example** |
| --- | --- | --- |
| **ListView:** | Displays a list of objects. | **class MyListView(ListView):**  **model = MyModel**  **template\_name = 'my\_template.html'**  **context\_object\_name = 'object\_list'  # default: object\_list** |
| **DetailView** | Displays details of a single object. | **class MyDetailView(DetailView):**  **model = MyModel**  **template\_name = 'my\_template.html'**  **context\_object\_name = 'object'  # default: object**  **pk\_url\_kwarg = 'my\_model\_id'  # default: pk** |
| **CreateView** | Displays a form to create a new object. | **class MyCreateView(CreateView):**  **model = MyModel**  **template\_name = 'my\_template.html'**  **fields = '\_\_all\_\_'  # or specify a list of fields** |
| **UpdateView** | Displays a form to update an existing object. | **class MyUpdateView(UpdateView):**  **model = MyModel**  **template\_name = 'my\_template.html'**  **fields = '\_\_all\_\_'  # or specify a list of fields**  **pk\_url\_kwarg = 'my\_model\_id'  # default: pk** |
| **DeleteView** | Displays a confirmation page to delete an object. | **class MyDeleteView(DeleteView):**  **model = MyModel**  **template\_name = 'my\_template.html'**  **success\_url = '/success-url/'**  **pk\_url\_kwarg = 'my\_model\_id'  # default: pk** |
| **Basic View Function** | Function-based view that returns “Hello, World!”    From Django.http import HttpResponse | **def my\_view(request):**  **# Your view logic here**  **return HttpResponse("Hello, World!")** |
| **Render a Template** | Function-based view to render a template with context.    From django.shortcuts import render | **def my\_template\_view(request):**  **context = {'variable': value}**  **return render(request, 'my\_template.html', context)** |
| **Redirect to a URL** | Function-based view to redirect to a specific URL.    From django.shortcuts import redirect | **def my\_redirect\_view(request):**  **return redirect('url\_name\_or\_path')** |
| **Handle a Form Submission** | Function-based view to handle form submission.    From django.shortcuts import render | **def my\_form\_view(request):**  **if request.method == 'POST':**  **# Process the form data here**  **else:**  **# Display the form**  **return render(request, 'my\_form\_template.html', context)** |
| **Handle URL Parameters** | Function-based view that accesses URL parameters. | **def my\_param\_view(request, param):**  **# Access the 'param' value from the URL** |
| **Protecting Views (Restrict Access) using @login\_required Decorator** | Function-based view protected with login\_required decorator.    From django.contrib.auth.decorators import login\_required | **@login\_required**  **def my\_protected\_view(request):**  **# Your view logic here** |
| **Bootstrap CSS** | Link to include Bootstrap CSS in the base template. | **Add the following link to the <head> section of your base template (usually base.html):**    **<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" rel="stylesheet">** |
| **Bootstrap JavaScript** | Script tag to include Bootstrap JavaScript library. | **Include the Bootstrap JavaScript library at the end of the <body> section to enable certain features (for example, dropdowns, modals):**    **<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.min.js"></script>** |
| **Bootstrap classes and components** | Create visually appealing and responsive web pages without having to write CSS styles manually. | **<a href="#" class="btn btn-primary">Click Me</a>** |
| **Configuration – Static files** | Django settings for static files configuration. | **In your Django settings (settings.py), define the following settings:**    **STATIC\_URL = '/static/studio/edx.org-next/'  # URL to access static files**  **STATICFILES\_DIRS = [os.path.join(BASE\_DIR, 'static')]  # Directory to look for static files** |
| **Configuration – Installed apps** | Defines a list of all the applications installed in the project. | **Add 'django.contrib.staticfiles' to your INSTALLED\_APPS in settings.py:**    **INSTALLED\_APPS = [**  **# ...**  **'django.contrib.staticfiles',**  **# ...**  **]** |
| **Configuration – App Dirs** | A configuration option used within the TEMPLATES setting. When set to TRUE, Django will look for template files within the app directories. | **Make sure the APP\_DIRS setting is set to True in the TEMPLATES list. This allows Django to look for static files within the apps' directories.**    **TEMPLATES = [**  **{**  **# ...**  **'APP\_DIRS': True,**  **# ...**  **},**  **]** |
| **Usage – Static content** | Code to style the HTML templates and provide interactivity to web pages. | **<link href="{% static 'your\_app/css/style.css' %}" rel="stylesheet">**  **<script src="{% static 'your\_app/js/script.js' %}"></script>**  **<img src="{% static 'your\_app/img/logo.png' %}" alt="Logo">** |
| **Collecting static files** | When deploying your project, you need to collect all static files into a single location. | **python manage.py collectstatic**  **STATIC\_ROOT = os.path.join(BASE\_DIR, 'staticfiles')** |