Task Manager application where users can create, read, update, and delete tasks.

Step 1: Set Up the Django Project

1. **Install Django** (if not already installed):

   ```
   pip install django
   ```

2. **Create a Django Project**:

   ```
   django-admin startproject taskmanager
   cd taskmanager
   ```

3. **Create a Django App**:

   ```
   python manage.py startapp tasks
   ```

4. **Add the App to `INSTALLED_APPS`**: Open `taskmanager/settings.py` and add `'tasks'` to the `INSTALLED_APPS` list:

   ```
   INSTALLED_APPS = [
       'django.contrib.admin',
       'django.contrib.auth',
       'django.contrib.contenttypes',
       'django.contrib.sessions',
       'django.contrib.messages',
       'django.contrib.staticfiles',
       'tasks',  # Add your app here
   ]
   ```

Step 2: Create the Task Model

1. **Define the Model**: Open `tasks/models.py` and create a `Task` model:

   ```
   from django.db import models

   class Task(models.Model):
       title = models.CharField(max_length=200)
       description = models.TextField(blank=True)
       completed = models.BooleanField(default=False)
       created_at = models.DateTimeField(auto_now_add=True)

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

2. **Migrate the Database**: Run the following commands to create and apply migrations:

   ```
   python manage.py makemigrations
   python manage.py migrate
   ```

Step 3: Set Up the Admin Panel

1. **Register the Model in the Admin Panel**: Open `tasks/admin.py` and register the `Task` model:

   ```
   from django.contrib import admin
   from .models import Task

   admin.site.register(Task)
   ```

2. **Create a Superuser**: Run the following command to create an admin user:

   ```
   python manage.py createsuperuser
   ```

   Follow the prompts to set up the superuser.

3. **Access the Admin Panel**: Run the server:

   ```
   python manage.py runserver
   ```

   Go to `http://127.0.0.1:8000/admin/` and log in with your superuser credentials. You’ll see the `Task` model there, and you can add tasks manually.

Step 4: Create Views for CRUD Operations

1. **Create Views**: Open `tasks/views.py` and add the following views:

   ```
   from django.shortcuts import render, redirect, get_object_or_404
   from .models import Task
   from .forms import TaskForm

   # List all tasks
   def task_list(request):
       tasks = Task.objects.all()
       return render(request, 'tasks/task_list.html', {'tasks': tasks})

   # Create a new task
   def task_create(request):
       if request.method == 'POST':
           form = TaskForm(request.POST)
           if form.is_valid():
               form.save()
               return redirect('task_list')
       else:
           form = TaskForm()
       return render(request, 'tasks/task_form.html', {'form': form})

   # Update a task
   def task_update(request, pk):
       task = get_object_or_404(Task, pk=pk)
       if request.method == 'POST':
           form = TaskForm(request.POST, instance=task)
   ```

```
        if form.is_valid():
            form.save()
            return redirect('task_list')
    else:
        form = TaskForm(instance=task)
    return render(request, 'tasks/task_form.html', {'form': form})

# Delete a task
def task_delete(request, pk):
    task = get_object_or_404(Task, pk=pk)
    if request.method == 'POST':
        task.delete()
        return redirect('task_list')
    return render(request, 'tasks/task_confirm_delete.html', {'task': task})
```

2. **Create a Form**: Create a new file `tasks/forms.py` and define a form for the `Task` model:

```
from django import forms
from .models import Task

class TaskForm(forms.ModelForm):
    class Meta:
        model = Task
        fields = ['title', 'description', 'completed']
```

Step 5: Set Up URLs

1. **Define URLs for the App**: Create a new file `tasks/urls.py` and add the following code:

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

urlpatterns = [
    path('', views.task_list, name='task_list'),
    path('create/', views.task_create, name='task_create'),
    path('update/<int:pk>/', views.task_update, name='task_update'),
    path('delete/<int:pk>/', views.task_delete, name='task_delete'),
]
```

2. **Include App URLs in the Project**: Open `taskmanager/urls.py` and include the `tasks` app URLs:

```
from django.contrib import admin
from django.urls import path, include

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include('tasks.urls')),  # Include tasks URLs
]
```

Step 6: Create Templates

1. **Create a Templates Directory**: Inside the `tasks` directory, create a folder named `templates`. Inside the `templates` folder, create another folder named `tasks`.

   The structure should look like this:

   ```
   tasks/
   ├── templates/
   │   └── tasks/
   │       ├── task_list.html
   │       ├── task_form.html
   │       └── task_confirm_delete.html
   ```

2. **Create `task_list.html`**: This template will display the list of tasks and links to create, update, and delete tasks.

   ```
   <!DOCTYPE html>
   <html lang="en">
   <head>
       <meta charset="UTF-8">
       <meta name="viewport" content="width=device-width, initial-scale=1.0">
       <title>Task List</title>
   </head>
   <body>
       <h1>Task List</h1>
       <a href="{% url 'task_create' %}">Create New Task</a>
       <ul>
           {% for task in tasks %}
               <li>
                   {{ task.title }} - {{ task.completed|yesno:"Completed,Not
   Completed" }}
                   <a href="{% url 'task_update' task.pk %}">Edit</a>
                   <form action="{% url 'task_delete' task.pk %}" method="post"
   style="display:inline;">
                       {% csrf_token %}
                       <button type="submit">Delete</button>
                   </form>
               </li>
           {% endfor %}
       </ul>
   </body>
   </html>
   ```

3. **Create `task_form.html`**: This template will display the form for creating and updating tasks.

   ```
   <!DOCTYPE html>
   <html lang="en">
   <head>
       <meta charset="UTF-8">
   ```

```
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Task Form</title>
</head>
<body>
    <h1>{% if form.instance.pk %}Update Task{% else %}Create Task{% endif %}
</h1>
    <form method="post">
        {% csrf_token %}
        {{ form.as_p }}
        <button type="submit">Save</button>
    </form>
    <a href="{% url 'task_list' %}">Cancel</a>
</body>
</html>
```

4. **Create `task_confirm_delete.html`**: This template will confirm the deletion of a task.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Delete Task</title>
</head>
<body>
    <h1>Delete Task</h1>
    <p>Are you sure you want to delete "{{ task.title }}"?</p>
    <form method="post">
        {% csrf_token %}
        <button type="submit">Yes, delete</button>
    </form>
    <a href="{% url 'task_list' %}">Cancel</a>
</body>
</html>
```

Step 7: Run the Application

1. **Run the Server**:

```
python manage.py runserver
```

2. **Access the Application**: Go to `http://127.0.0.1:8000/` in your browser. You'll see the task list, and you can create, update, and delete tasks.

Final Project Structure

```
taskmanager/
├── taskmanager/
```

```
│   ├── __init__.py
│   ├── settings.py
│   ├── urls.py
│   ├── wsgi.py
│   └── asgi.py
├── tasks/
│   ├── migrations/
│   ├── __init__.py
│   ├── admin.py
│   ├── apps.py
│   ├── models.py
│   ├── tests.py
│   ├── views.py
│   ├── forms.py
│   ├── urls.py
│   ├── templates/
│   │   └── tasks/
│   │       ├── task_list.html
│   │       ├── task_form.html
│   │       └── task_confirm_delete.html
└── manage.py
```

Summary

- You created a Django project and app.
- You defined a `Task` model and performed migrations.
- You set up views, forms, and URLs for CRUD operations.
- You created templates to display the task list, form, and delete confirmation.

This is a fully functional CRUD application in Django!