Step 1: Create Django To-Do App

Generating the parent project framework:

Once we installed Django in our system, we will get ‘django-admin’ command line tool, which can be used to create Django projects.

django-admin startproject todoProject

Creating app:

Now, create app using the django-admin command-line tool.

python manage.py startapp todoApp

Configure Project:
Add application in settings.py, so that Django aware about application.

In Settings.py:

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'todoApp',
]
```

The second file that we need to modify in the project folder is urls.py, which controls the URL lookup at the project level.

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

urlpatterns = [
    path('admin/', admin.site.urls),
    path('todoProject/',include('todoApp.urls')),
]
```

Now, create app-level URL configuration file now. Open a new file in editor and save it under the name urls.py:

```
urlpatterns=[


]
```

Now, working project and app setup are completed and can test our work so far by starting the Django development server:

```
python manage.py runserver
```

Step 2: Design To-Do Data:

Each type of user data will require its own **data model**. To-do list app will contain just two basic types of data:

1. **A ToDoList with a title:** Number of lists can be added
2. **A ToDoItem that is linked to a particular list:** Again, there's no limit to the number of ToDoItem objects. Each ToDoItem will have its own title, a longer description, a created date, and a due date.

```
from django.db import models
from django.utils import timezone
from django.urls import reverse

#todoApp/model
def one_week_hence():
  return timezone.now() + timezone.timedelta(days=7)

class ToDoList(models.Model):
  title = models.CharField(max_length=100, unique=True)

  def get_absolute_url(self):
    return reverse("list", args=[self.id])

  def __str__(self):
    return self.title

class ToDoItem(models.Model):
  title = models.CharField(max_length=100)
  description = models.TextField(null=True, blank=True)
  created_date = models.DateTimeField(auto_now_add=True)
  due_date = models.DateTimeField(default=one_week_hence)
  todo_list = models.ForeignKey(ToDoList, on_delete=models.CASCADE)

  def get_absolute_url(self):
    return reverse(
      "item-update", args=[str(self.todo_list.id), str(self.id)]
    )

  def __str__(self):
    return f"{self.title}: due {self.due_date}"

  class Meta:
    ordering = ["due_date"]
```

Create data base by activating migrations:

```
python manage.py makemigrations
```

```
python manage.py migrate
```

Step 3: Add Sample To-Do Data

Create new superuser now to use the admin interface:

```
python manage.py createsuperuser
```

Register the models with the admin app, which can be done by editing the file admin.py:

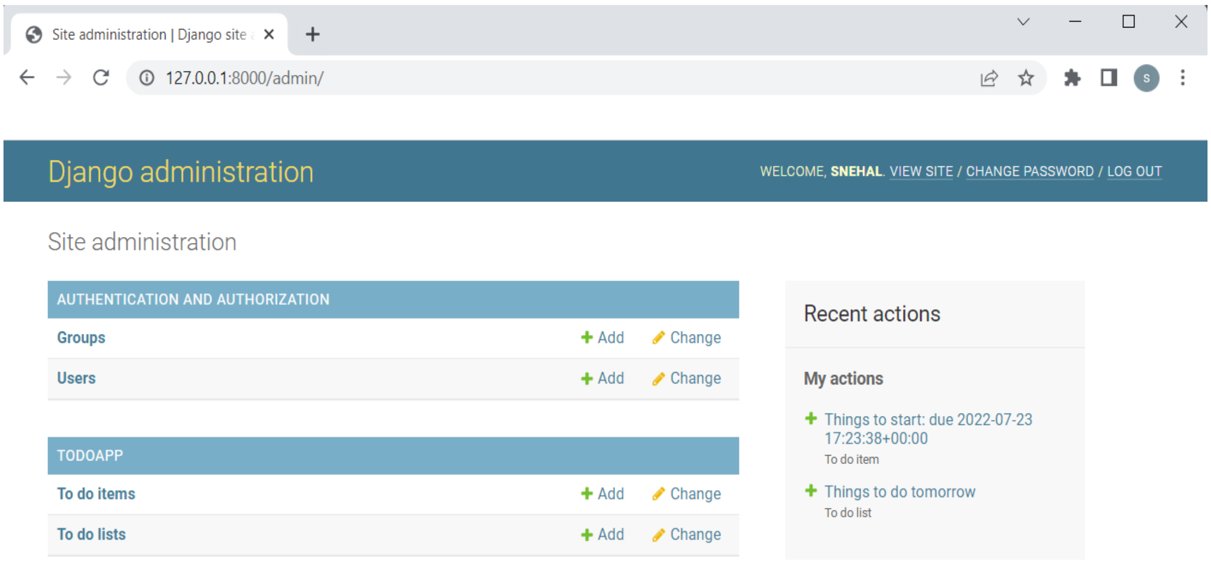
```
from django.contrib import admin
from .models import ToDoItem, ToDoList

# Register your models here.
admin.site.register(ToDoItem)
admin.site.register(ToDoList)
```

Launch the development server and start exploring.

```
python manage.py runserver
```

Now open a web browser and go to the address http://127.0.0.1:8000/admin/.

Enter newly minted credentials, and the admin landing page appears:


Site administration | Django site
127.0.0.1:8000/admin/
Django administration
WELCOME, SNEHAL. VIEW SITE / CHANGE PASSWORD / LOG OUT

Site administration

AUTHENTICATION AND AUTHORIZATION		
Groups	+ Add	Change
Users	+ Add	Change

TODOAPP		
To do items	+ Add	Change
To do lists	+ Add	Change

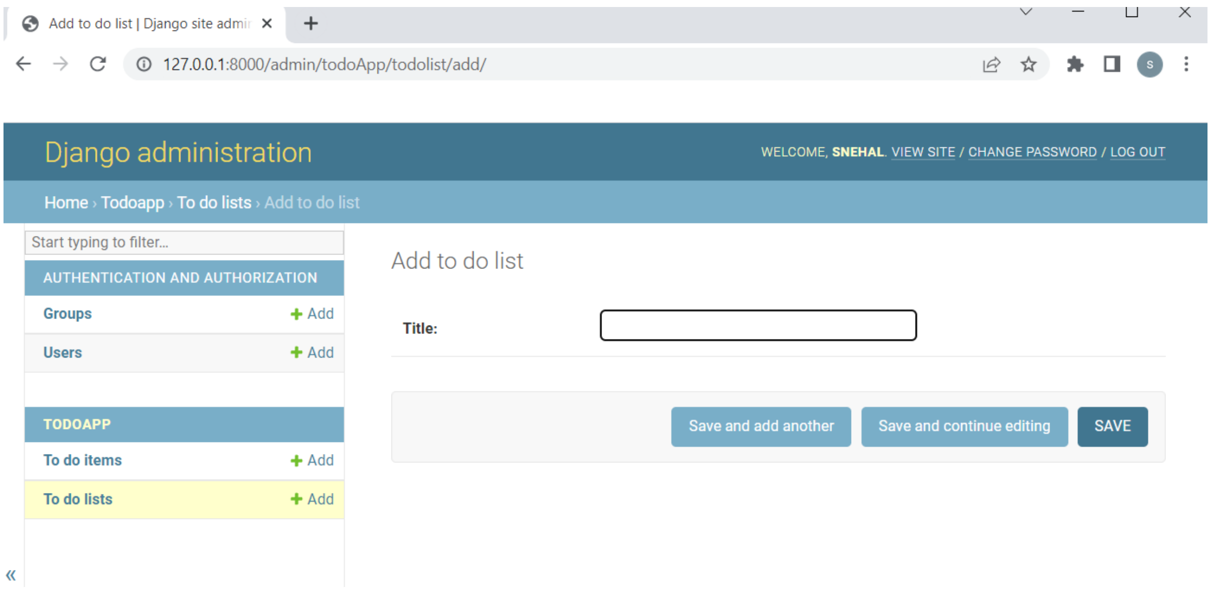
Recent actions

My actions

+ Things to start: due 2022-07-23 17:23:38+00:00
  To do item
+ Things to do tomorrow
  To do list

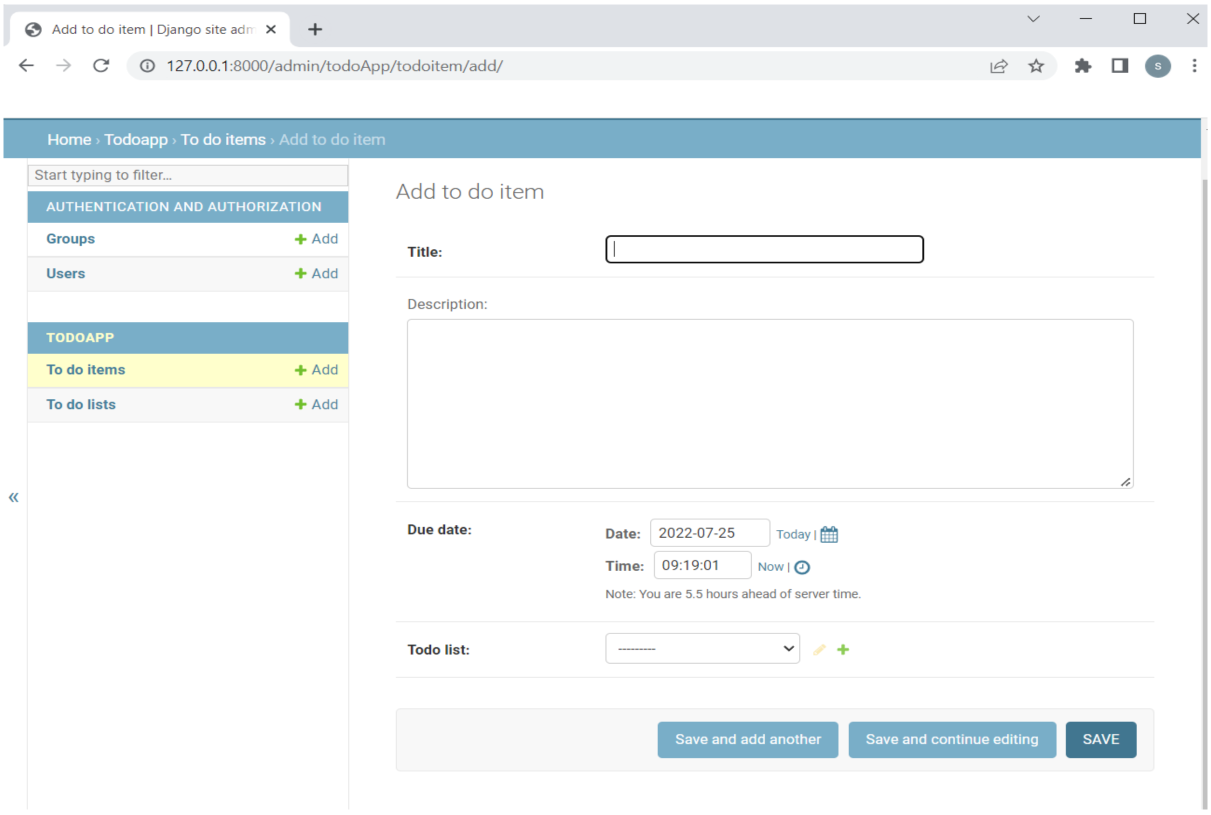
Start a To-Do List
At the left of the main Django administration page, click on To do lists. On the next screen, click on the button at the top right that says *ADD TO DO LIST*.

The new list now appears on a page headed *Select to do list to change*. A new form appears:


Add to do list | Django site admin
127.0.0.1:8000/admin/todoApp/todolist/add/
Django administration
WELCOME, SNEHAL. VIEW SITE / CHANGE PASSWORD / LOG OUT
Home › Todoapp › To do lists › Add to do list
Start typing to filter...
AUTHENTICATION AND AUTHORIZATION
Groups + Add
Users + Add
TODOAPP
To do items + Add
To do lists + Add
Save and add another
Save and continue editing
SAVE

Add to do list

Title:


Add to do item | Django site adm
127.0.0.1:8000/admin/todoApp/todoitem/add/
Home › Todoapp › To do items › Add to do item
Start typing to filter...
AUTHENTICATION AND AUTHORIZATION
Groups + Add
Users + Add
TODOAPP
To do items + Add
To do lists + Add
Today |
Now |
Save and add another
Save and continue editing
SAVE

Add to do item

Title:

Description:

Due date:
Date: 2022-07-25
Time: 09:19:01
Note: You are 5.5 hours ahead of server time.

Todo list: ---------

Step 4: Create the Django Views for application in views.py

ListView to Display a List of To-Do Items:

```
from django.shortcuts import render
from django.views.generic import ListView
from django.urls import reverse,reverse_lazy

from django.views.generic import (
    ListView,
    CreateView,
    UpdateView,
    DeleteView,
)
from .models import ToDoList, ToDoItem

# todo_list/todoApp/views.py

class ListListView(ListView):
    model = ToDoList
    template_name = "todoApp/index.html"

class ItemListView(ListView):
    model = ToDoItem
    template_name = "todoApp/todo_list.html"

    def get_queryset(self):
        return ToDoItem.objects.filter(todo_list_id=self.kwargs["list_id"])

    def get_context_data(self):
        context = super().get_context_data()
        context["todo_list"] = ToDoList.objects.get(id=self.kwargs["list_id"])
        return context
```

Step 5: Create and Update Model Objects in Django

```
from django.urls import reverse,reverse_lazy

from django.views.generic import (
    ListView,
    CreateView,
    UpdateView,
    DeleteView,
)
class ListCreate(CreateView):
    model = ToDoList
    fields = ["title"]

    def get_context_data(self):
        context = super(ListCreate, self).get_context_data()
        context["title"] = "Add a new list"
        return context
```

```
class ItemCreate(CreateView):
    model = ToDoItem
    fields = [
        "todo_list",
        "title",
        "description",
        "due_date",
    ]

    def get_initial(self):
        initial_data = super(ItemCreate, self).get_initial()
        todo_list = ToDoList.objects.get(id=self.kwargs["list_id"])
        initial_data["todo_list"] = todo_list
        return initial_data

    def get_context_data(self):
        context = super(ItemCreate, self).get_context_data()
        todo_list = ToDoList.objects.get(id=self.kwargs["list_id"])
        context["todo_list"] = todo_list
        context["title"] = "Create a new item"
        return context

    def get_success_url(self):
        return reverse("list", args=[self.object.todo_list_id])


class ItemUpdate(UpdateView):
    model = ToDoItem
    fields = [
        "todo_list",
        "title",
        "description",
        "due_date",
    ]

    def get_context_data(self):
        context = super(ItemUpdate, self).get_context_data()
        context["todo_list"] = self.object.todo_list
        context["title"] = "Edit item"
        return context

    def get_success_url(self):
        return reverse("list", args=[self.object.todo_list_id])
```

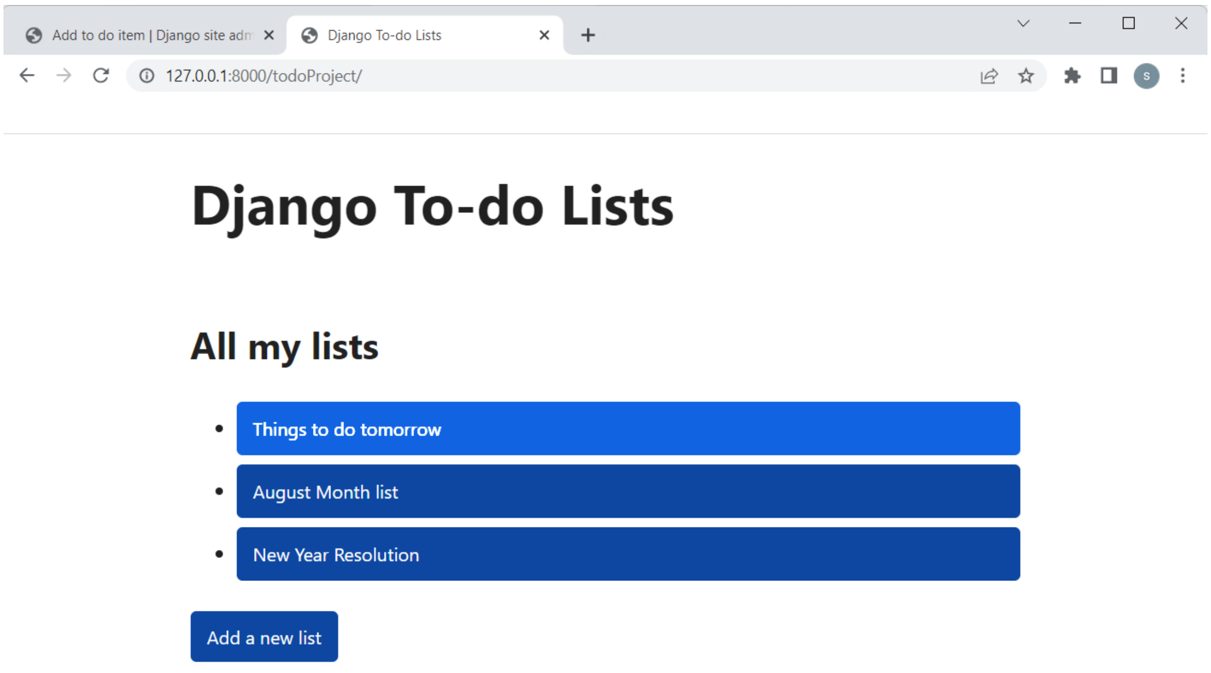
Step 6: Delete To-Do Lists and Items

```
rom django.urls import reverse,reverse_lazy

from django.views.generic import (
    ListView,
    CreateView,
    UpdateView,
    DeleteView,
)
from .models import ToDoList, ToDoItem
class ListDelete(DeleteView):
    model = ToDoList
    # You have to use reverse_lazy() instead of reverse(),
    # as the urls are not loaded when the file is imported.
    success_url = reverse_lazy("index")

class ItemDelete(DeleteView):
    model = ToDoItem

    def get_success_url(self):
        return reverse_lazy("list", args=[self.kwargs["list_id"]])

    def get_context_data(self, **kwargs):
        context = super().get_context_data(**kwargs)
        context["todo_list"] = self.object.todo_list
        return context
```

Define URL-pattern at application level for our view inside urls.py

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

urlpatterns = [
    path("", views.ListListView.as_view(), name="index"),
    path("list/<int:list_id>/",views.ItemListView.as_view(), name="list"),
    # CRUD patterns for ToDoLists
    path("list/add/", views.ListCreate.as_view(), name="list-add"),
    path(
        "list/<int:pk>/delete/", views.ListDelete.as_view(), name="list-delete"
    ),
    # CRUD patterns for ToDoItems
    path(
        "list/<int:list_id>/item/add/",
        views.ItemCreate.as_view(),
        name="item-add",
    ),
    path(
        "list/<int:list_id>/item/<int:pk>/",
        views.ItemUpdate.as_view(),
        name="item-update",
    ),
    path(
        "list/<int:list_id>/item/<int:pk>/delete/",
        views.ItemDelete.as_view(),
        name="item-delete",
    ),
]
```

Step 7: Use Django To-Do List App


Add to do item | Django site adm
Django To-do Lists
127.0.0.1:8000/todoProject/

Django To-do Lists

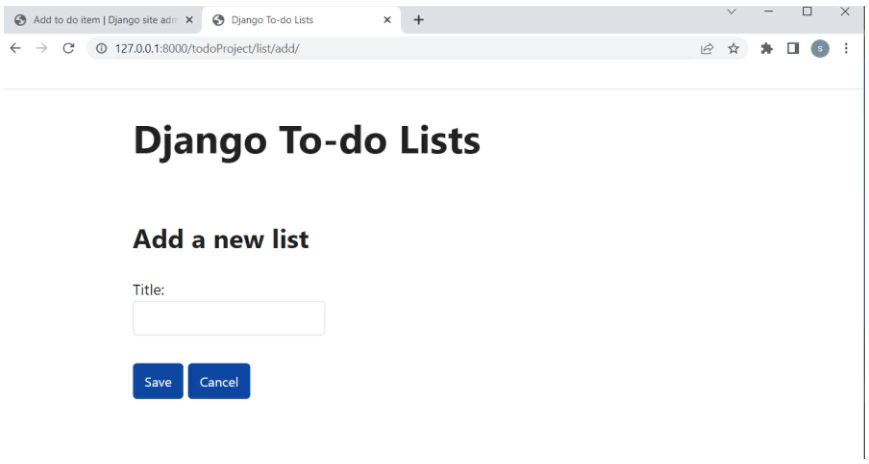
All my lists

- Things to do tomorrow
- August Month list
- New Year Resolution

Add a new list

The app heading *Django To-do Lists* will appear on every page.

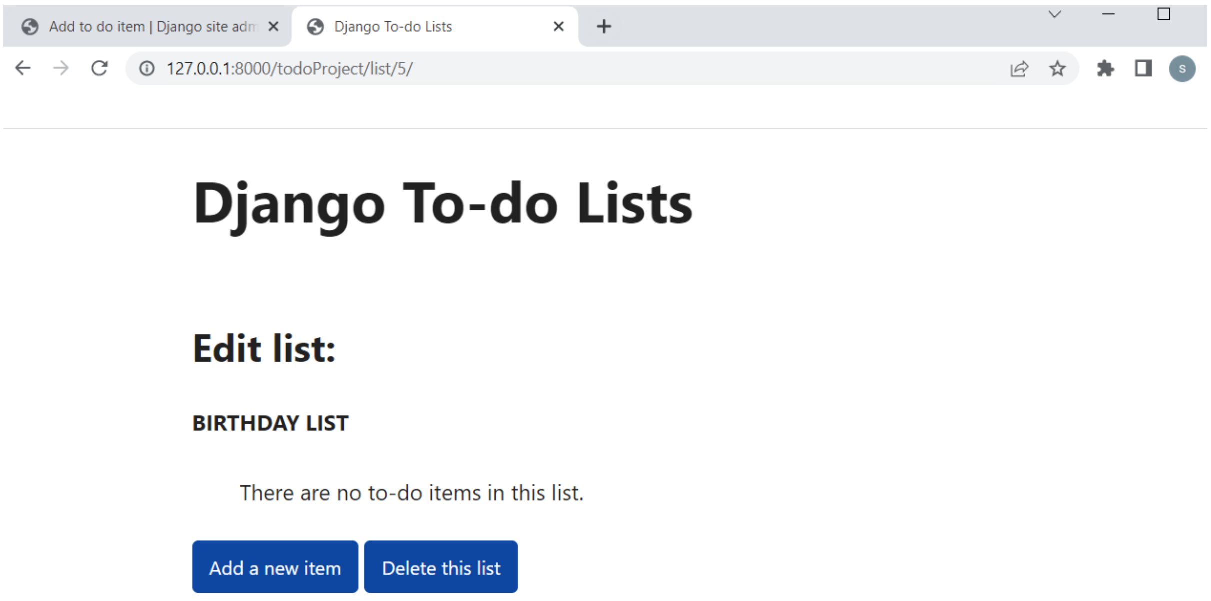
- Click on *Add a new list*. A new screen appears, offering a blank text box for the new list's title.
- Give new list a name and press *Save*. You're taken to the *Edit List* page, with the message *There are no to-do items in this list.*
- Click on *Add a new item*. The *Create a new item* form appears. Fill in a title and a description, and notice that the default due date is exactly one week ahead. We can change it as per the requirement.


Add to do item | Django site adm
Django To-do Lists
127.0.0.1:8000/todoProject/list/add/

Django To-do Lists

Add a new list

Title:

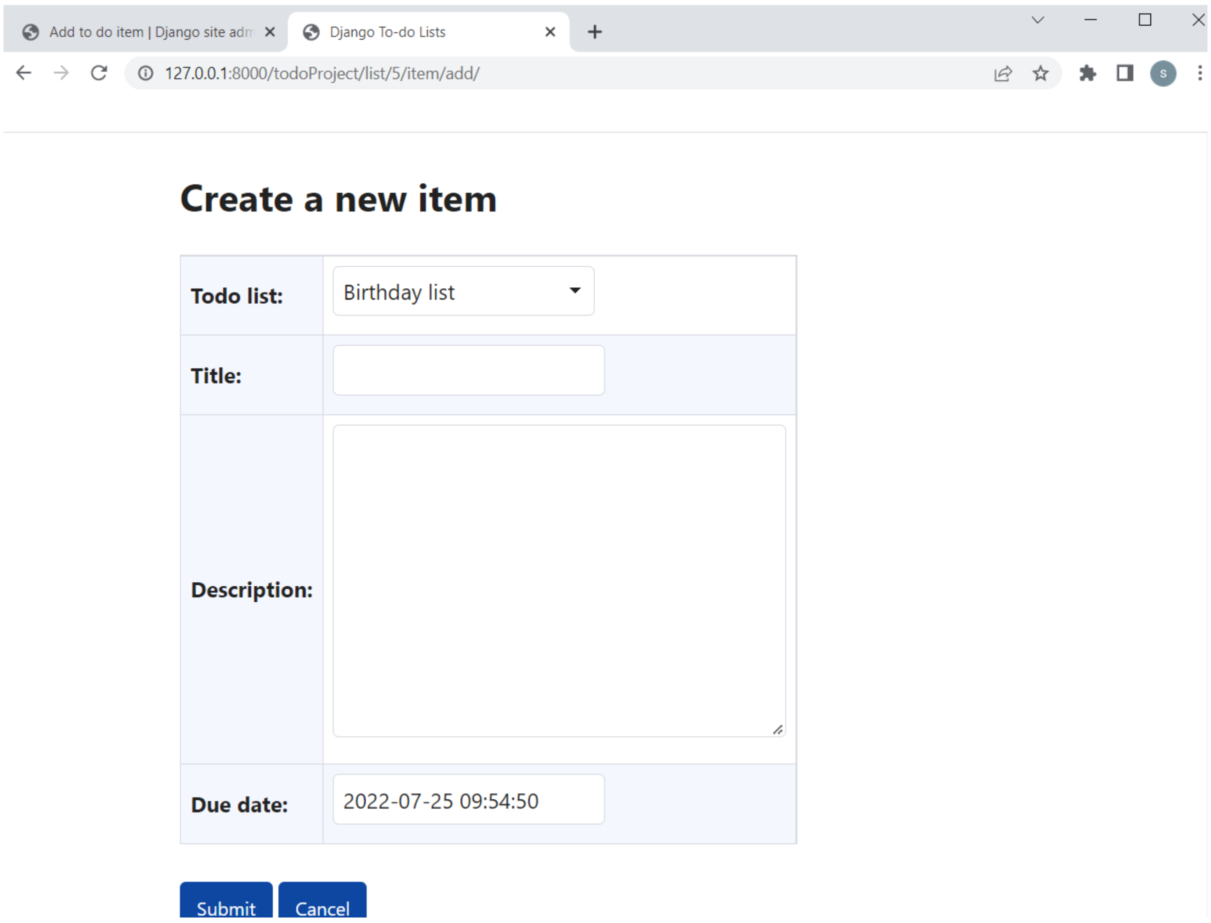

Add to do item | Django site adm
Django To-do Lists
127.0.0.1:8000/todoProject/list/5/

Django To-do Lists

Edit list:

BIRTHDAY LIST

There are no to-do items in this list.


Add to do item | Django site adm
Django To-do Lists
127.0.0.1:8000/todoProject/list/5/item/add/

Create a new item

Todo list:	Birthday list
Title:	
Description:	
Due date:	2022-07-25 09:54:50