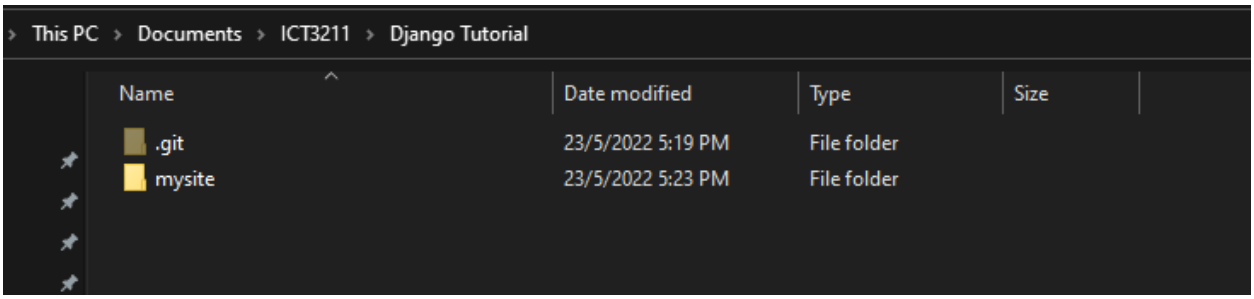
Setting up a project in Django

1. In the folder where you want to start your Django Project, type:

```
django-admin startproject yourprojectname
```

2. A folder with the name that you typed earlier for “yourprojectname” will appear. In my case, I typed mysite.



3. Change directory into mysite on your command terminal:

```
cd mysite
```

4. To test your newly created website:

```
python manage.py runserver
```

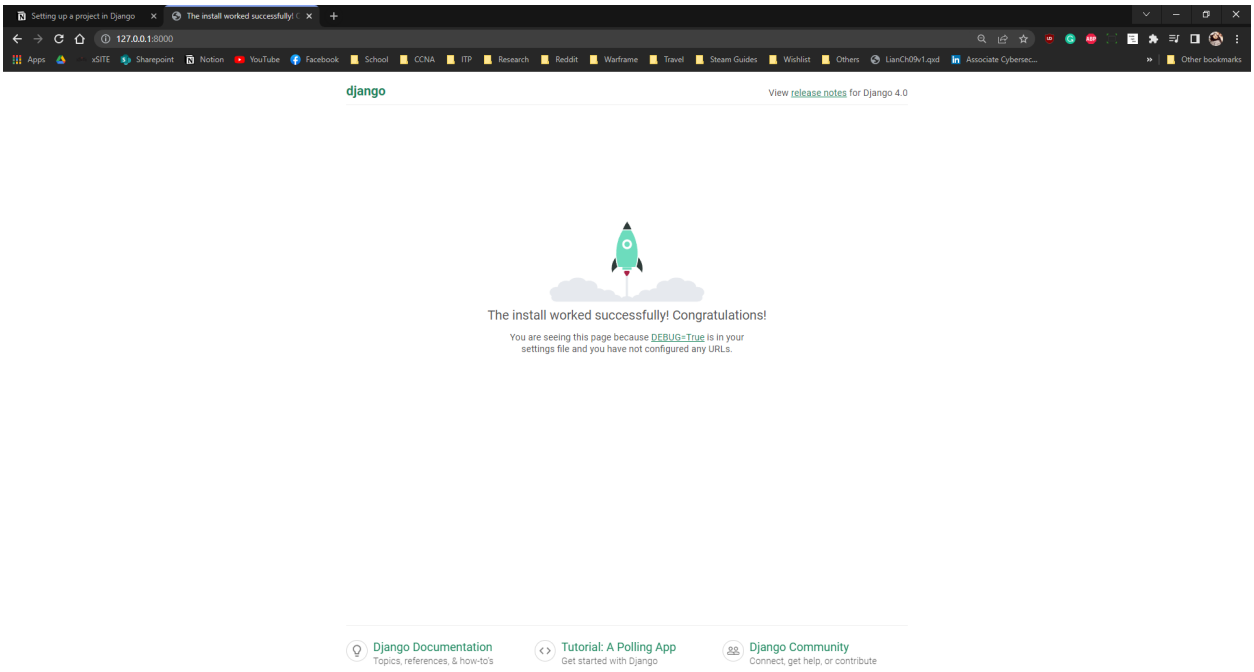
5. This starts a development server at “http://127.0.0.1:8000/”. Copy this url into your web browser.

```
D:\Documents\ICT3211\Django Tutorial\mysite>python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).

You have 18 unapplied migration(s). Your project may not work properly until you apply the migrations
 auth, contenttypes, sessions.
Run 'python manage.py migrate' to apply them.
May 23, 2022 - 17:26:37
Django version 4.0.4, using settings 'mysite.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
[23/May/2022 17:26:44] "GET / HTTP/1.1" 200 10697
[23/May/2022 17:26:44] "GET /static/admin/css/fonts.css HTTP/1.1" 200 423
[23/May/2022 17:26:44] "GET /static/admin/fonts/Roboto-Bold-webfont.woff HTTP/1.1" 200 86184
[23/May/2022 17:26:44] "GET /static/admin/fonts/Roboto-Regular-webfont.woff HTTP/1.1" 200 85876
[23/May/2022 17:26:44] "GET /static/admin/fonts/Roboto-Light-webfont.woff HTTP/1.1" 200 85692
```

6. The webpage should look like this:



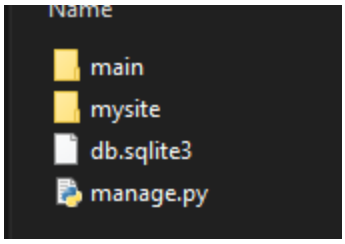
☝️ To run the dev server on another port, simply type the same command with the port number.
python manage.py runserver portnumber

7. At the command terminal, terminate your dev server by pressing Ctrl+C on your keyboard.

8. Next, to create an app, in the same directory in your command terminal, type:

```
python manage.py startapp appname
```

9. A folder should appear with the name you gave your app.



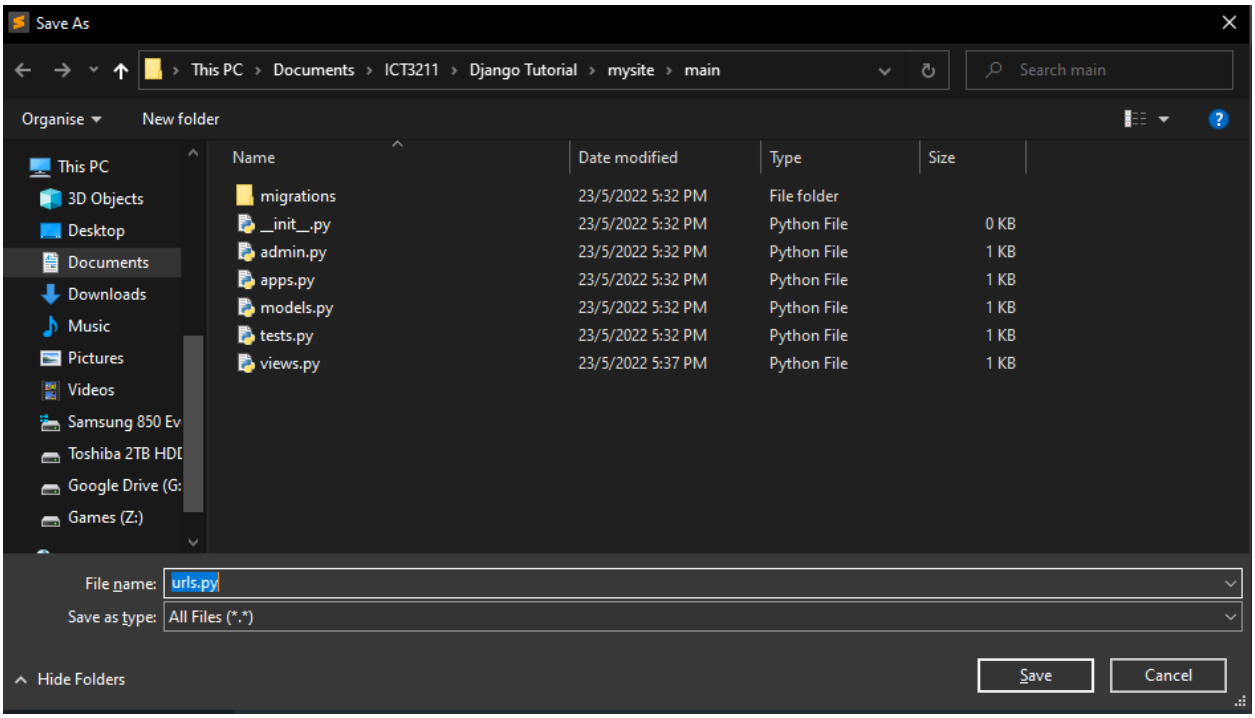
10. Next, we will go into our main folder. In here, you will find a few files of interest. First, we will go to views.py page.
    a. Import HttpResponse

```
#step 1
from django.http import HttpResponse
```

    b. Create a function

```
#step 2: create a function
def index(response)
    return HttpResponse("Hello World!") # html code goes in here
```

11. Next, create a new file, enter the name as urls.py and press save.



This file will define paths to different pages which we call *views.*

```
#define paths to different pages (views)

from django.urls import path
from . import views #import views from current dir

urlpatterns= [
path("", views.index, name="index"), # this serves as the index of the webpage
]
```

12. To link our app (main) to our project (mysite), go to urls.py in *mysite* folder. There should already be some code filled in.

```
"""mysite URL Configuration

The `urlpatterns` list routes URLs to views. For more information please see:
    https://docs.djangoproject.com/en/4.0/topics/http/urls/
Examples:
Function views
    1. Add an import:  from my_app import views
    2. Add a URL to urlpatterns:  path('', views.home, name='home')
Class-based views
    1. Add an import:  from other_app.views import Home
    2. Add a URL to urlpatterns:  path('', Home.as_view(), name='home')
Including another URLconf
    1. Import the include() function: from django.urls import include, path
    2. Add a URL to urlpatterns:  path('blog/', include('blog.urls'))
"""
from django.contrib import admin
from django.urls import path

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

What we want to do here is when a user enters a certain url, we want to access a certain webpage based on their input. Example: when a user enters /login or /signup and they are directed to login and signup pages.