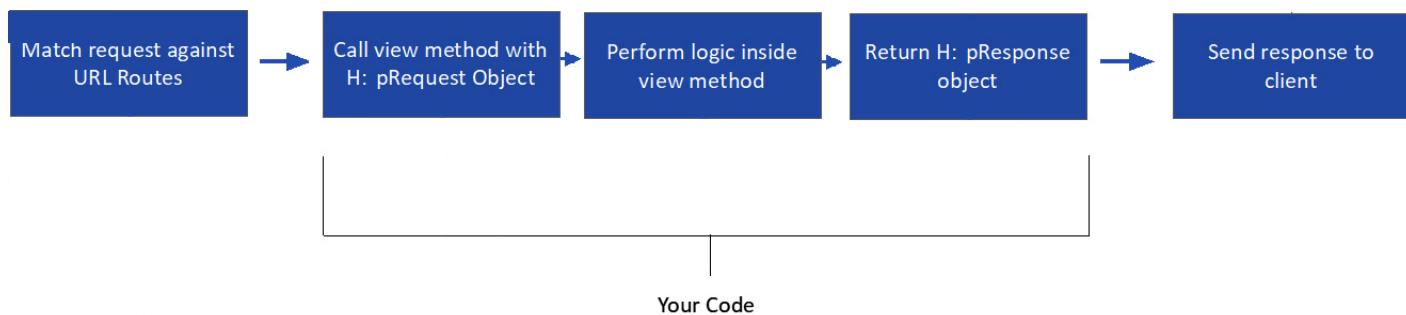


This lab session covers the basics of the Django URL and HTML template. By the end of this session, students should have a very good understanding of how to define URL patterns and create an HTML template and how both work together to process an HTTP request. They should also have already practiced with the Git system and how to use basic commands to push and manage their repository.

The following diagram shows the direction of the transmission of HTTP requests and HTTP responses, between a browser and a web server



Inside the framework, the flow of HTTP request and response is illustrated in the following figure. The sections indicated as **Your Code** are for the code that you write, and the first and last steps are taken care of by Django. Django does the URL matching for you, calls your view code, and then handles passing the response back to the client. Actually, you are going to build and configure a link between a URL and a view where the URL matching is taking place by Django.



Pre-lab Preparation:

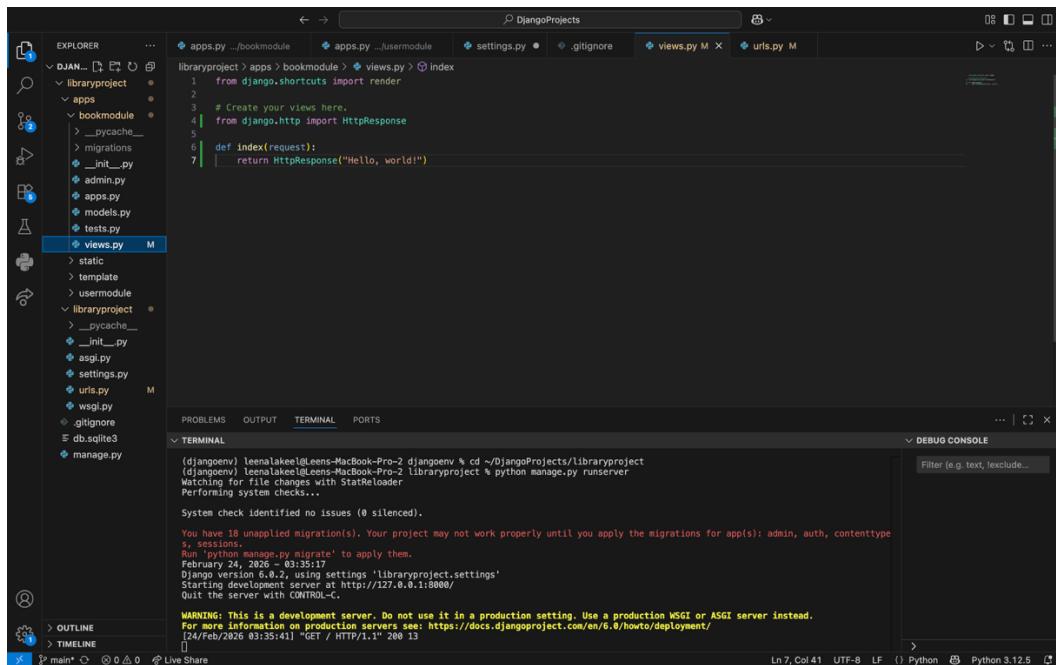
1. Complete the previous lab (Lab3): must have the initial Django project with apps (bookmodule & usermodule) and the necessary configurations.
2. Good understanding of HTTP protocol, mainly the request, response, and HTTP status codes.
3. Basic knowledge about HTTP tags.

Lab Activities: Build and configure a simple link between a URL and a view (simple pattern), along with a simple HTML template.

Task 1: Build your first view function and corresponding URL mapping in the core/urls.py

Step 1: from the root folder, navigate to apps/bookmodule/view.py, then remove `# Create your views here.` from views.py and instead insert this content:

```
from django.http import HttpResponse
def index(request):
    return HttpResponse("Hello, world!")
```



Step 2: Open DjangoProjects/urls.py, file (that controls all URL mapping). Import your bookmodule views into the DjangoProjects/urls.py file by adding this line after the other existing imports:

```
...
import apps.bookmodule.views
```

Step 3: Again inside DjangoProjects/urls.py, add a map to the index view to the urlpatterns list by adding a call to the path function¹, with an empty string and a reference to the index function:

```
urlpatterns = [
    path('admin/', admin.site.urls),
    path('', apps.bookmodule.views.index) #add only this line of code
```

¹ If the paths aren't sufficient for defining your URL patterns, you can also use regular expressions. To do so, use `re_path()` instead of `path()`.

]

The screenshot shows the Visual Studio Code interface with a Django project named 'libraryproject'. The Explorer sidebar on the left lists files and folders like 'urls.py', 'wsgi.py', '.gitignore', and 'db.sqlite3'. The 'urls.py' file in the center contains code for a book module. The 'TERMINAL' tab at the bottom shows the command `python manage.py runserver` being run, and the server is listening on port 8000.

```

6 Examples:
7 Function views
8 1. Add an import: from my_app import views
9 2. Add a URL to urlpatterns: path('', views.home, name='home')
10 Class-based views
11 1. Add an import: from other_app.views import Home
12 2. Add a URL to urlpatterns: path('', Home.as_view(), name='home')
13 Including another URLconf
14 1. Import the include() function: from django.urls import include, path
15 2. Add a URL to urlpatterns: path('blog/', include('blog.urls'))
16 ....
17 from django.contrib import admin
18 from django.urls import path
19 import apps.bookmodule.views
20
21 urlpatterns = [
22     path('admin/', admin.site.urls),
23     path('', apps.bookmodule.views.index),
24 ]

```

```

(djangoenv) leenalakeel@Leens-MacBook-Pro-2 djangoenv % cd ~/DjangoProjects/libraryproject
(djangoenv) leenalakeel@Leens-MacBook-Pro-2 libraryproject % 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 for app(s): admin, auth, contenttype, sessions.
Run 'python manage.py migrate' to apply them.
February 24, 2026 - 03:35:17
Django version 6.0.2, using settings 'libraryproject.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.

WARNING: This is a development server. Do not use it in a production setting. Use a production WSGI or ASGI server instead.
For more information on production servers see: https://docs.djangoproject.com/en/6.0/howto/deployment/
[24/Feb/2026 03:35:41] "GET / HTTP/1.1" 200 13

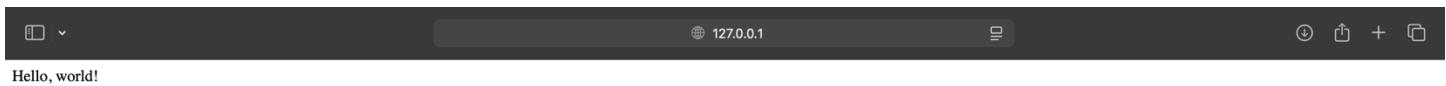
```

Step 4: Run the server to test that everything works as intended. From the ROOT location of your Django project, do the following (don't forget to activate the virtual python environment):

> python manage.py runserver

Important Note: From now on, the development server watches your Django project directory and will restart automatically every time you save a file so that any code changes you make are automatically reloaded into the server. No need to re-run the server again. However, you still have to manually refresh your browser to see changes there (you may press F5 to refresh the browser current webpage).

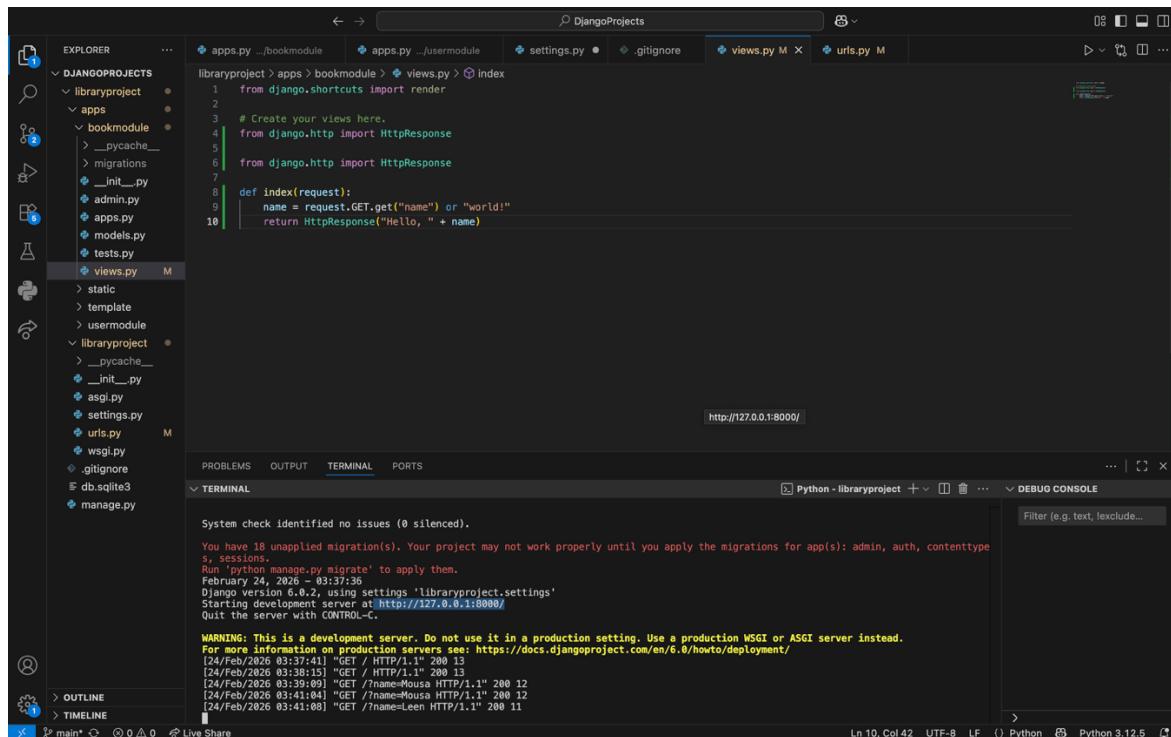
Step 5: Open a web browser and navigate to <http://127.0.0.1:8000>. You should see “Hello, world!”



Task 2: Add parameters with HTTP requests²

Step 1: From the root folder, navigate to apps/bookmodule/views.py, then, in views.py, modify content as follows:

```
from django.http import HttpResponseRedirect
def index(request):
    name = request.GET.get("name") or "world!" #add this line
    return HttpResponseRedirect("Hello, " + name) #replace the word "world!" with the variable name
```



Step 2: Now from the browser navigate to the following urls:

<http://127.0.0.1:8000>. You should see “Hello, world!”

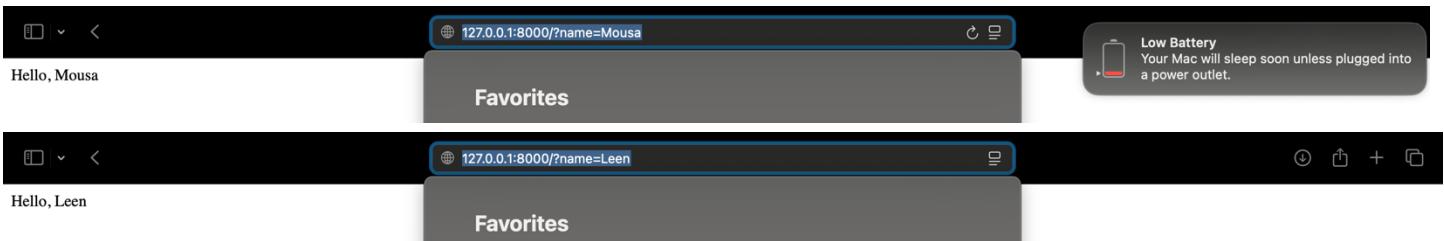
<http://127.0.0.1:8000?name=world!>. You should see “Hello, world!”

<http://127.0.0.1:8000?name=Mousa>. You should see “Hello, Mousa”



² It is possible to have more than one parameter, like <http://127.0.0.1:8000?id=10&name=world!&age=18>

Note: this method is GET HTTP request, which is not recommended for security reasons, and use POST HTTP request instead.

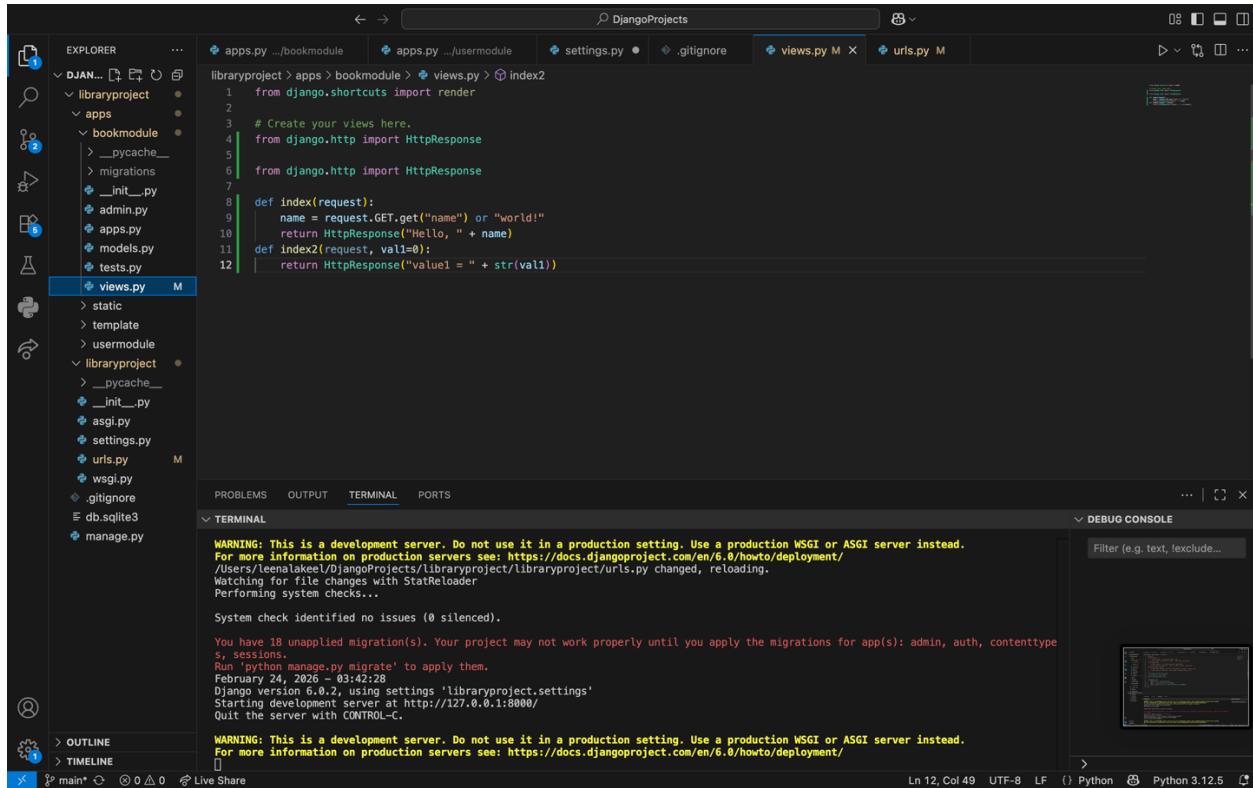


Task 3: Build your second view function and corresponding URL mapping with parameters within URL path

Step 1: from the root folder, navigate to apps/bookmodule/view.py, then, in views.py, insert this additional content (index2 view) beside index view:

```
from django.http import HttpResponse
def index(request):
    name = request.GET.get("name") or "world!"
    return HttpResponse("Hello, " + name)

def index2(request, val1 = 0):    #add the view function (index2)
    return HttpResponse("value1 = " + str(val1))
```



Step 2: Again inside DjangoProjects/urls.py, add a map to the index2 view to the urlpatterns list by adding a call to the path function, with an '`index2/<pythontype:parameter1>`'³ string and a reference

³ For more than one parameter, it will be like

'index2/<pythontype:parameter1>/<pythontype:parameter2>/<pythontype:parameter3>'''

to the index2 function. Note that `<pythontype: parameter1>` can be rewritten as `<parameter1>` without specifying the data type of the parameter variable:

```
urlpatterns = [
    path('admin/', admin.site.urls),
    path('', apps.bookmodule.views.index),
    path('index2/<int:val1>', apps.bookmodule.views.index2) #add only this line
]
```

```
urlpatterns = [
    path('admin/', admin.site.urls),
    path('', apps.bookmodule.views.index),
    path('index2/<int:val1>', apps.bookmodule.views.index2),
]
```

```
WARNING: This is a development server. Do not use it in a production setting. Use a production WSGI or ASGI server instead.
For more information on production servers see: https://docs.djangoproject.com/en/6.0/howto/deployment/
/Users/leenalakeel/DjangоПроекты/libraryproject/libraryproject/urls.py changed, reloading.
Watching for file changes with StatReloader
Performing system checks...
System check identified no issues (0 silenced).

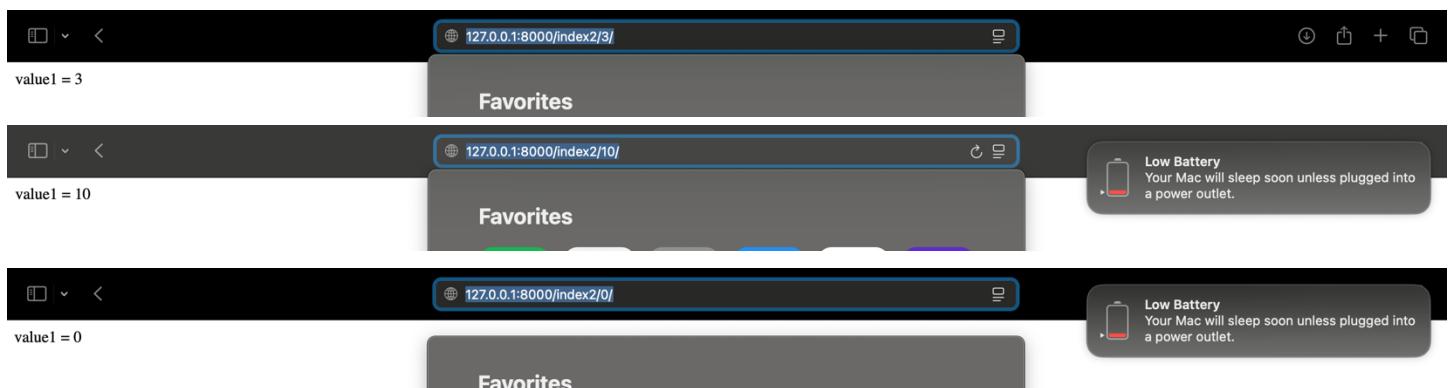
You have 18 unapplied migration(s). Your project may not work properly until you apply the migrations for app(s): admin, auth, contenttype, sessions.
Run 'python manage.py migrate' to apply them.
February 24, 2026 - 03:42:28
Django version 6.0.2, using settings 'libraryproject.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
```

Step 3: Now from the browser navigate to the following urls:

`http://127.0.0.1:8000/index2/3/`. You should see “`value1 = 3`”

`http://127.0.0.1:8000/index2/10/`. You should see “`value1 = 10`”

`http://127.0.0.1:8000/index2/0/`. You should see “`value1 = 0`”



Task 4: Create a simple HTML template

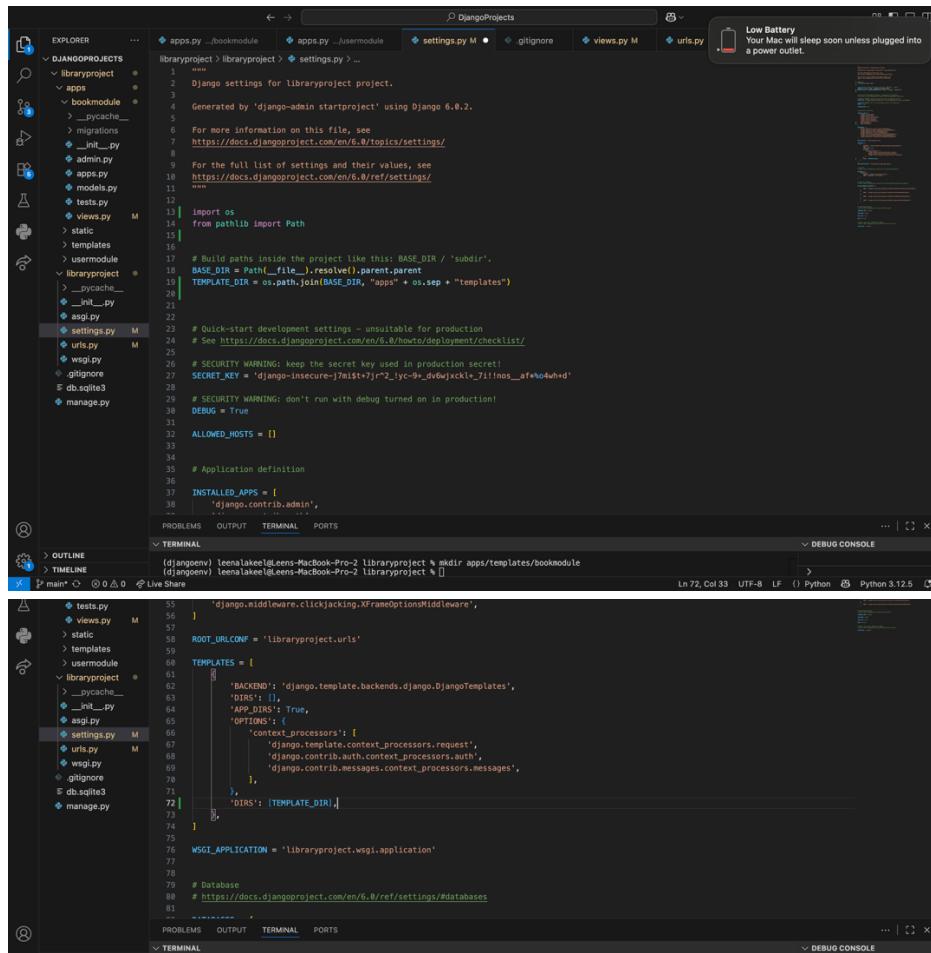
Step 1: Create a templates directory for bookmodule app, by doing the following:

```
> mkdir apps/templates/bookmodule
```

Step 2: Update main/settings.py with template location

```
import os

...
BASE_DIR = Path(__file__).resolve().parent.parent
TEMPLATE_DIR = os.path.join(BASE_DIR, "apps" + os.sep + "templates")
TEMPLATES = [
    ...
    'DIRS': [TEMPLATE_DIR],
    ...
]
```



The screenshot shows the PyCharm IDE interface with the Django project structure. The Explorer panel on the left lists files and folders like `libraryproject`, `apps`, `bookmodule`, `migrations`, `admin.py`, `apps.py`, `models.py`, `tests.py`, `views.py`, `static`, `templates`, and `usermodule`. The `settings.py` file is open in the editor, showing the code for setting up the template directory. The code is as follows:

```
...
# Build paths inside the project like this: BASE_DIR / 'subdir'.
BASE_DIR = Path(__file__).resolve().parent.parent
TEMPLATE_DIR = os.path.join(BASE_DIR, "apps" + os.sep + "templates")

# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/2.2/howto/deployment/checklist/
# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = 'django-insecure-y7misi+7r2_lys-9_d6w/xck1_7llnos_afw0whhd'

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True

ALLOWED_HOSTS = []

# Application definition

INSTALLED_APPS = [
    'django.contrib.admin',
    ...
]

# Templates
TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
        'DIRS': [TEMPLATE_DIR],
    },
]
...
WSGI_APPLICATION = 'libraryproject.wsgi.application'

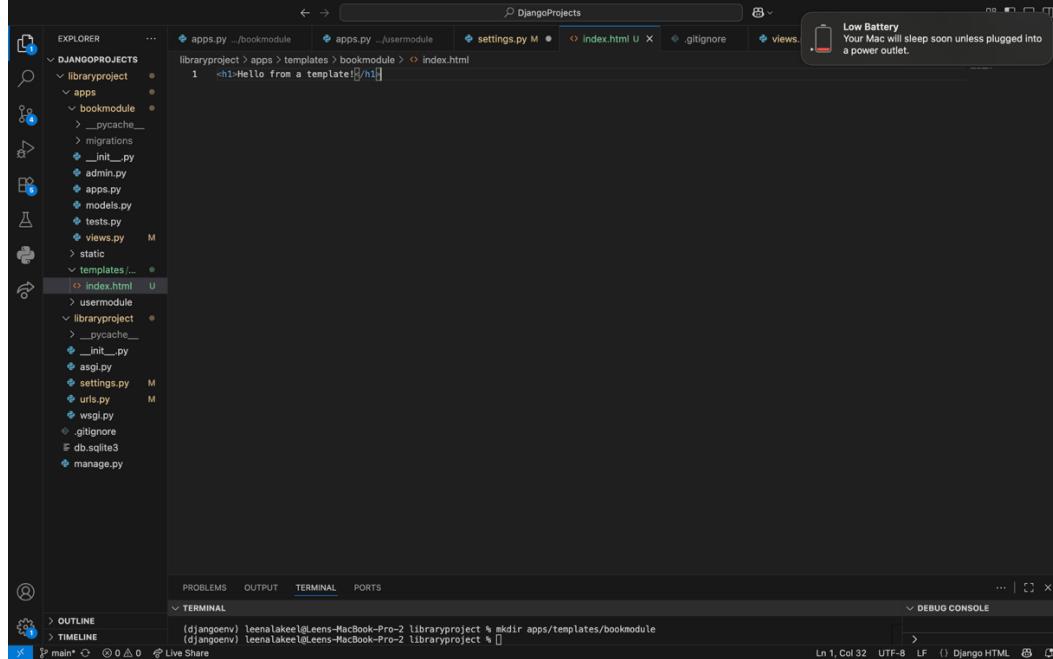
# Database
# https://docs.djangoproject.com/en/2.2/ref/settings/#databases
...
```

Note that many options are available to tell Django how to find templates, which can be set in the `TEMPLATES` setting. The easiest one is to create a `templates` directory inside the `bookmodule` directory. Django will look in this (and in other apps' `templates` directories)

because APP_DIRS is True in the settings.py file. Anyway, this task is intended to utilize the one template inside the apps folder and make it accessible to all apps.

Step 3: Create a file with the name "index.html" within the templates directory 'apps/templates/bookmodule', and add the following html content:

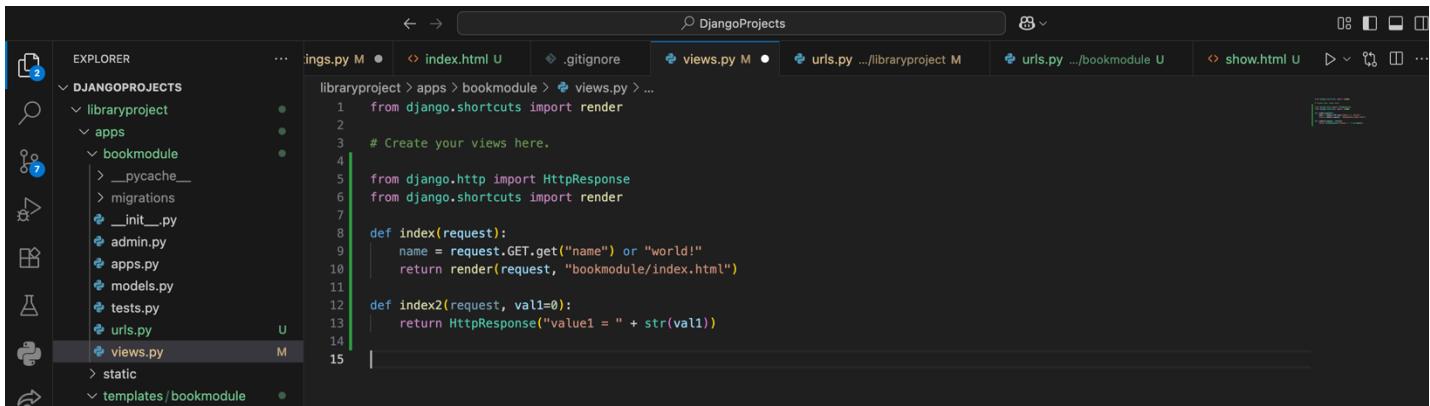
```
<h1>Hello from a template!</h1>
```



Step 4: Open views.py in the bookmodule directory. We no longer manually create HttpResponseRedirect, so you may remove the HttpResponseRedirect import line and add this line after the other existing imports:
`from django.shortcuts import render`

Step 5: Update the index function, in apps/bookmodule/views.py, so that, instead of returning HttpResponseRedirect, it's returning a call to render, passing in request and the template name:
`def index(request):`

```
    name = request.GET.get("name") or "world!"  
    return render(request, "bookmodule/index.html")      #Change HttpResponseRedirect to render function
```



The screenshot shows the VS Code interface with the 'DjangoProjects' workspace open. The left sidebar displays the project structure under 'EXPLORER'. The main editor area shows the contents of 'views.py' in the 'bookmodule' app:

```

from django.shortcuts import render
# Create your views here.

def index(request):
    name = request.GET.get("name") or "world!"
    return render(request, "bookmodule/index.html")

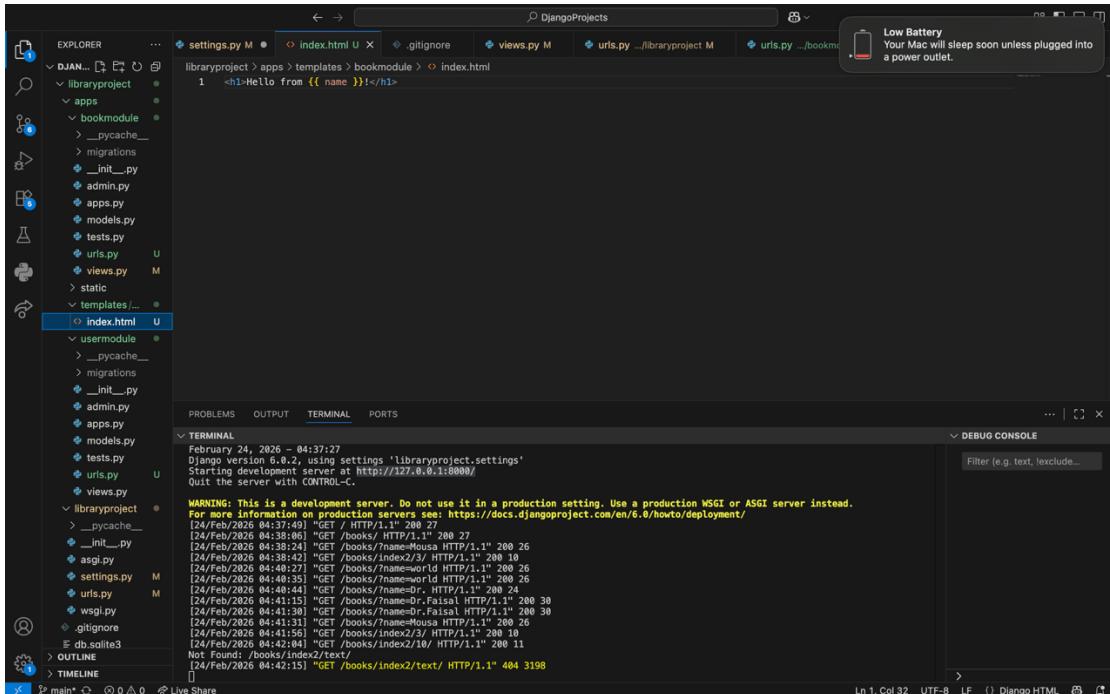
def index2(request, val=0):
    return HttpResponse("value1 = " + str(val))

```

Task 5: Rendering variables in the HTML template that processes a context

Step 1: open the file “index.html”, and update the file so that it contains a place to render the name variable:

```
<h1>Hello from {{ name }}!</h1>
```



The screenshot shows the VS Code interface with the 'DjangoProjects' workspace open. The left sidebar displays the project structure under 'EXPLORER'. The main editor area shows the contents of 'index.html' in the 'bookmodule' app:

```
<h1>Hello from {{ name }}!</h1>
```

The terminal window at the bottom shows the development server logs:

```

February 24, 2026 - 04:37:27
Django version 6.0.2, using settings 'libraryproject.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.

WARNING: This is a development server. Do not use it in a production setting. Use a production WSGI or ASGI server instead.
For more information on production servers see: https://docs.djangoproject.com/en/6.0/howto/deployment/
[24/Feb/2026 04:37:49] "GET / HTTP/1.1" 200 27
[24/Feb/2026 04:38:24] "GET /books/?name=Mous HTTP/1.1" 200 27
[24/Feb/2026 04:38:42] "GET /books/index2/3 HTTP/1.1" 200 10
[24/Feb/2026 04:48:27] "GET /books/?name=world HTTP/1.1" 200 26
[24/Feb/2026 04:48:44] "GET /books/?name=Dr. HTTP/1.1" 200 26
[24/Feb/2026 04:49:44] "GET /books/?name=Dr. Faisal HTTP/1.1" 200 26
[24/Feb/2026 04:41:15] "GET /books/?name=Dr.Faisal HTTP/1.1" 200 30
[24/Feb/2026 04:41:30] "GET /books/?name=Dr.Faisal HTTP/1.1" 200 30
[24/Feb/2026 04:41:50] "GET /books/index2/2 HTTP/1.1" 200 10
[24/Feb/2026 04:41:56] "GET /books/index2/3 HTTP/1.1" 200 10
[24/Feb/2026 04:42:04] "GET /books/index2/10 HTTP/1.1" 200 11
Not Found: /books/index2/text/
[24/Feb/2026 04:42:15] "GET /books/index2/text/ HTTP/1.1" 404 3198

```

Step 2: Update the index function, in apps/bookmodule/views.py, and add the context dictionary as the third argument to the render function. Change your render line to this:

```
def index(request):
    name = request.GET.get("name") or "world!"
    return render(request, "bookmodule/index.html" , {"name": name}) #your render line
```

```

settings.py M
index.html U
.gitignore
views.py M X
urls.py .../libraryproject M
urls.py .../bookm...
Low Battery
Your Mac will sleep soon unless plugged into a power outlet.

EXPLORER
Django... E U S
libraryproject
apps
bookmodule
__pycache__
migrations
__init__.py
admin.py
apps.py
models.py
tests.py
urls.py
views.py M
views.py M
static
templates/...
index.html U
usermodule
__pycache__
migrations
__init__.py
admin.py
apps.py
models.py
tests.py
urls.py
views.py
libraryproject
__pycache__
__init__.py
asgi.py
settings.py M
urls.py M
wsgi.py
.gitignore
db.sqlite3
OUTLINE
TIMELINE

PROBLEMS OUTPUT TERMINAL PORTS
TERMINAL
February 24, 2026 - 04:37:27
Django version 6.0.2, using settings 'libraryproject.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.

WARNING: This is a development server. Do not use it in a production setting. Use a production WSGI or ASGI server instead.
For more information on production servers see: https://docs.djangoproject.com/en/6.0/howto/deployment/
[24/Feb/2026 04:37:27] "GET / HTTP/1.1" 200 27
[24/Feb/2026 04:38:06] "GET /books/ HTTP/1.1" 200 26
[24/Feb/2026 04:38:24] "GET /books/?name=Mousa HTTP/1.1" 200 26
[24/Feb/2026 04:38:42] "GET /books/index2/3 HTTP/1.1" 200 10
[24/Feb/2026 04:40:27] "GET /books/?name=world HTTP/1.1" 200 26
[24/Feb/2026 04:40:35] "GET /books/?name=world HTTP/1.1" 200 26
[24/Feb/2026 04:40:48] "GET /books/?name=Dr. Faisal HTTP/1.1" 200 26
[24/Feb/2026 04:41:15] "GET /books/?name=Dr. Faisal HTTP/1.1" 200 30
[24/Feb/2026 04:41:30] "GET /books/?name=Dr. Faisal HTTP/1.1" 200 10
[24/Feb/2026 04:41:31] "GET /books/?name=Mousa HTTP/1.1" 200 26
[24/Feb/2026 04:41:56] "GET /books/index2/3 HTTP/1.1" 200 10
[24/Feb/2026 04:42:04] "GET /books/index2/10/ HTTP/1.1" 200 11
Not Found: /books/index2/text/
[24/Feb/2026 04:42:15] "GET /books/index2/text/ HTTP/1.1" 404 3198

```

Ln 10, Col 68 UTF-8 LF () Python Python 3.12.5

Task 6: Define patterns globally (DjangoProjects/urls.py) with specific urls.py file for each app/module

Note that it is common to keep one URL configuration per application in your Django project. Here, we will create a separate URL configuration for each app and add it to our project-level URL configuration.

Step 1: Create a new urls.py file for each app module(bookmodule and usermodule) in the same folder as the views.py file.

Step 2: move urls of bookmodule to apps/bookmodule/urls.py, and include newly created urls.py in core/urls.py using include module.

DjangoProjects/urls.py

```

from django.contrib import admin
from django.urls import include, path
urlpatterns = [
    path('admin/', admin.site.urls),
    path('books/', include("apps.bookmodule.urls")), #include urls.py of bookmodule app
    path('users/', include("apps.usermodel.urls")) #include urls.py of usermodule app
]

```

apps/bookmodule/urls.py

```

from django.urls import path
from . import views

```

```
urlpatterns = [
    path('', views.index),
    path('index2/<int:val1>', views.index2)
]
```

apps/usermodule/urls.py

```
from django.urls import path
urlpatterns = [
]
```

The screenshot shows the VS Code interface with the DjangoProjects workspace. The Explorer sidebar shows the project structure: libraryproject > bookmodule > urls.py. The main editor tab displays the following code:

```
10 Class-based views
11     1. Add an import: from other_app.views import Home
12     2. Add a URL to urlpatterns: path('', Home.as_view(), name='home')
13 Including another URLconf
14     1. Import the include() function: from django.urls import include, path
15     2. Add a URL to urlpatterns: path('blog/', include('blog.urls'))
16 """
17 from django.contrib import admin
18 from django.urls import path, include
19 import apps.bookmodule.views
20
21 urlpatterns = [
22     path('admin/', admin.site.urls),
23     path('', apps.bookmodule.views.index),
24     path('index2/<int:val1>', apps.bookmodule.views.index2),
25     path('books/', include("apps.bookmodule.urls")),
26     path('users/', include("apps.usermodule.urls")),
27 ]
```

The screenshot shows the VS Code interface with the DjangoProjects workspace. The Explorer sidebar shows the project structure: libraryproject > bookmodule > urls.py. The main editor tab displays the following code:

```
1 from django.urls import path
2 from . import views
3
4 urlpatterns = [
5     path('', views.index),
6     path('index2/<int:val1>', views.index2),
7 ]
```

The screenshot shows the VS Code interface with the DjangoProjects workspace. The Explorer sidebar shows the project structure: libraryproject > usermodule > urls.py. The main editor tab displays the following code:

```
1 from django.urls import path
2
3 urlpatterns = [
4 ]
```

At the bottom of the screen, the Terminal tab is open, showing the output of the command "python manage.py runserver". The output includes:

```
February 24, 2026 - 04:37:27
Django version 6.0.2, using settings 'libraryproject.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
```

A warning message is also present at the bottom of the terminal output:

```
WARNING: This is a development server. Do not use it in a production setting. Use a production WSGI or ASGI server instead.
```

Step 3: Now from the browser navigate to the following urls:

<http://127.0.0.1:8000/books/>. You should see “Hello, world!”

<http://127.0.0.1:8000/books/?name=world>!. You should see “Hello, world!”

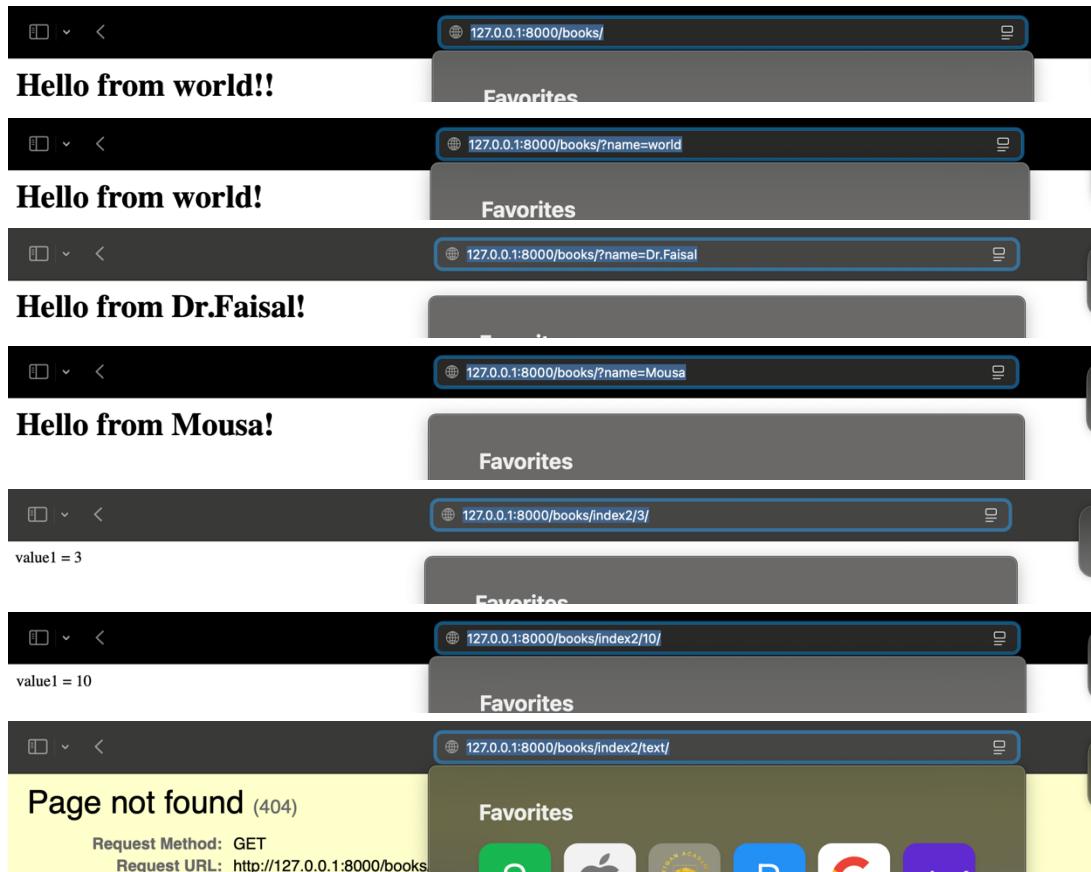
<http://127.0.0.1:8000/books/?name=Dr.Faisal>. You should see “Hello, Dr. Faisal”

<http://127.0.0.1:8000/books/?name=Mousa>. You should see “Hello, Mousa”

[http://127.0.0.1:8000/books/index2/3/](http://127.0.0.1:8000/books/index2/3).. You should see “value1 = 3”

[http://127.0.0.1:8000/books/index2/10/](http://127.0.0.1:8000/books/index2/10). You should see “value1 = 10”

[http://127.0.0.1:8000/books/index2/text/](http://127.0.0.1:8000/books/index2/text). You should see “error, expected val1 to be integer”

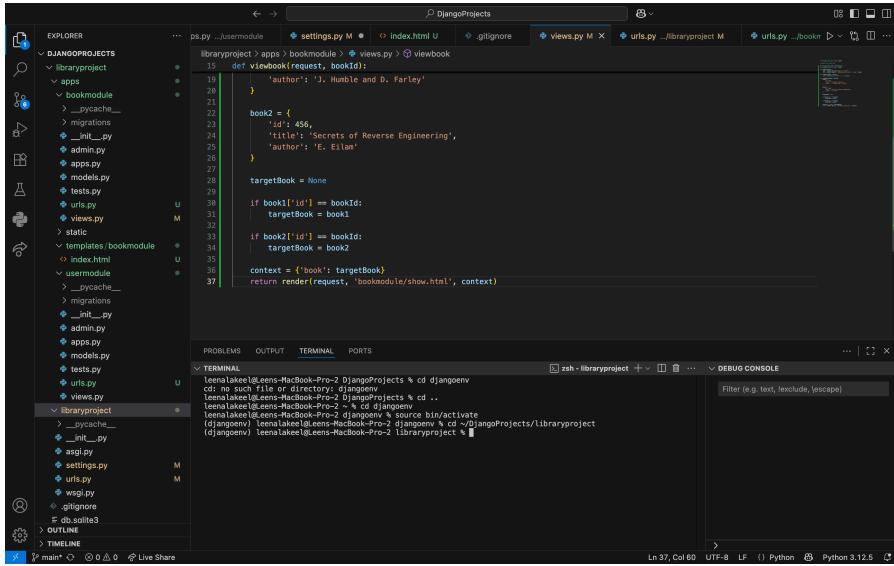


Task7: Create a URL, view, and HTML to display one book details

Step 1: Create a new function at the location: 'root/apps/bookmodule/views.py' that accept an ID argument:

```
def viewbook(request, bookId):
    # assume that we have the following books somewhere (e.g. database)
    book1 = {'id':123, 'title':'Continuous Delivery', 'author':'J. Humble and D. Farley'}
    book2 = {'id':456, 'title':'Secrets of Reverse Engineering', 'author':'E. Eilam'}
    targetBook = None
    if book1['id'] == bookId: targetBook = book1
```

```
if book2['id'] == bookId: targetBook = book2
context = {'book':targetBook} # book is the variable name accessible by the template
return render(request, 'bookmodule/show.html', context)
```

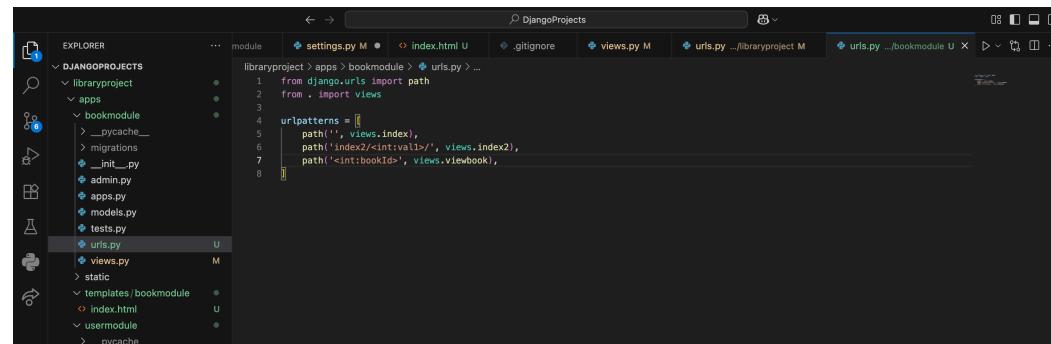


The screenshot shows the VS Code interface with the 'views.py' file open in the editor. The code implements a view function 'viewbook' that takes a request and a book ID. It checks if the book ID matches the one in the query parameters. If it does, it sets the target book to the book from the database. Then, it creates a context dictionary containing the target book and returns it to the template 'show.html'.

```
15 def viewbook(request, bookId):
16     book1 = {
17         'author': 'J. Humble and D. Farley'
18     }
19     book2 = {
20         'id': 456,
21         'title': 'Secrets of Reverse Engineering',
22         'author': 'E. Eilan'
23     }
24
25     targetBook = None
26
27     if book1['id'] == bookId:
28         targetBook = book1
29
30     if book2['id'] == bookId:
31         targetBook = book2
32
33     context = {'book': targetBook}
34
35     return render(request, 'bookmodule/show.html', context)
```

Step 2: In 'root/apps/bookmodule/urls.py', add the following path function to urlpatterns:

```
path('<int:bookId>', views.viewbook)
```



The screenshot shows the VS Code interface with the 'urls.py' file open in the editor. The code defines a URL pattern for the 'viewbook' view, which maps to the path '/bookmodule/<int:bookId>'.

```
1 from django.urls import path
2 from . import views
3
4 urlpatterns = [
5     path('', views.index),
6     path('index2<int:values>', views.index2),
7     path('<int:bookId>', views.viewbook),
8 ]
```

Step 3: Create the template at the location: 'root/apps/templates/bookmodule/show.html' and put some code in it.

```
<h4>ID: {{ book.id }}</h4>
<h4>Title: {{ book.title }}</h4>
<h4>Authors</h4>
<p>{{ book.author }}</p>
<h4>Description</h4>
<p>Description here</p>
```

```

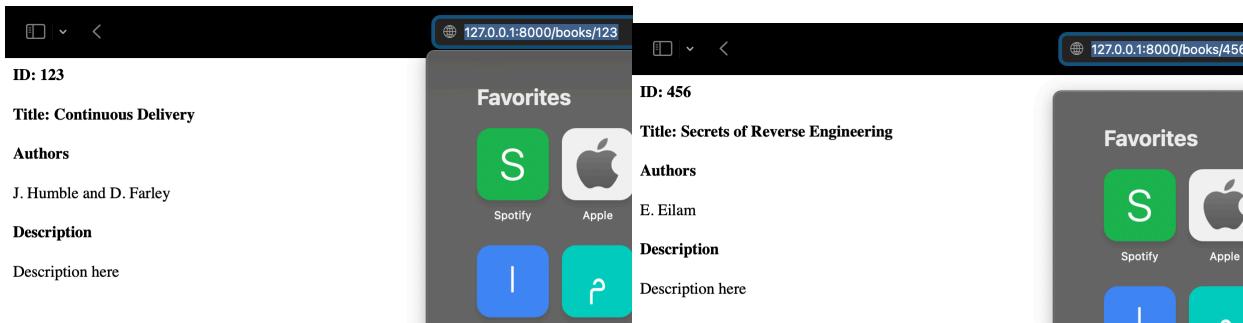
Djangoprojects
  libraryproject
    apps
      bookmodule
        __pycache__
        migrations
        __init__.py
        admin.py
        apps.py
        models.py
        tests.py
        urls.py
        views.py
      static
      templates/bookmodule
        index.html
        show.html
      usermodule
        __pycache__
        migrations
        __init__.py
        admin.py

```

Step 4: Now from the browser navigate to the following urls:

<http://127.0.0.1:8000/books/123>

<http://127.0.0.1:8000/books/456>



Additional content: To redirect to another view or URL, use the function `redirect`, and for details and examples visit the following link:

<https://docs.djangoproject.com/en/5.1/topics/http/shortcuts/#redirect>