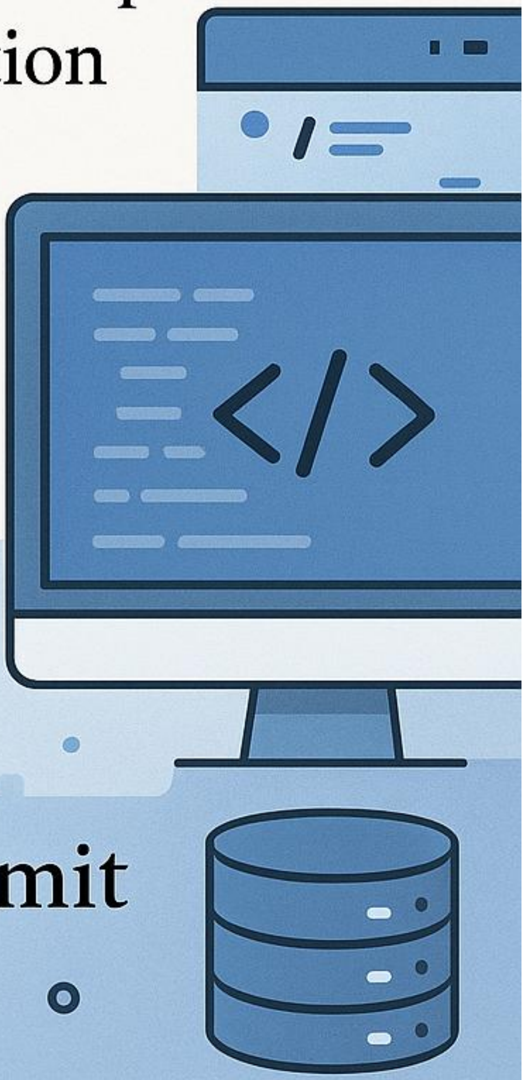
django

Mastering Django

A Comprehensive Guide from Environment Setup to CRUD Application Development



Badhan Roy Amit

Topic: Django

Introduction

Django is a high-level web framework written in Python that helps developers build web applications quickly and efficiently. It was created by experienced programmers to simplify many common tasks involved in web development. As an open-source and free framework, Django allows developers to focus on building their applications without having to start from scratch each time.

The main purpose of Django is to make it easier to develop complex websites that rely on databases. It is designed around important software engineering principles such as reusability, modularity, minimal code repetition, and low interdependence between different parts of a system. A core idea in Django's design is "Don't Repeat Yourself" (DRY), which promotes writing clean and efficient code. Python is used throughout Django, including for defining settings, handling files, and creating data models.

Additionally, Django offers a built-in administrative interface that lets developers manage content by creating, reading, updating, and deleting data. This admin interface is generated automatically by inspecting the application's models, which saves time and effort in developing backend management tools.

Several prominent websites rely on Django for their platforms. Examples include **Instagram**, **Mozilla**, **Disqus**, **Bitbucket**, **Nextdoor**, and **Clubhouse**, demonstrating Django's capability to handle both large-scale and complex web applications.

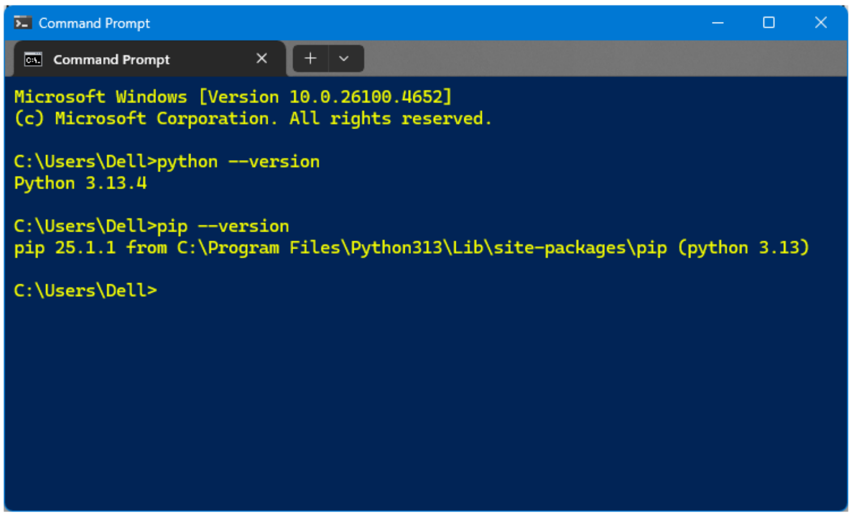
Step by Step Implementation with Codes and Outputs

To gain a practical understanding of how Django operates, we will develop a small project that demonstrates its core features. This project, titled **"Boitoroni,"** will serve as an online repository containing PDF books across various genres. The platform will offer both free and premium PDFs to users.

However, in this initial stage, we will not implement any functionalities related to purchasing premium books. Instead, our focus will be on illustrating how an administrator can upload new book information into the system. Additionally, users will have the ability to browse and read the free PDFs available on the platform. Through this project, we aim to explore how Django facilitates the development of web applications and manages essential tasks such as data handling and user interaction. So, Let's get started.

1. **The first step** is to verify whether Python and pip are installed on the system. To accomplish this, we open the Command Prompt and execute the command "**python --version**" to check the current Python installation. If Python is not installed, it can be downloaded and installed from https://www.python.org/downloads/.

   On our device, Python version 3.13.4 is already installed. Next, we run the command "**pip –version"** to confirm the presence of pip, which is the package manager for Python. Fortunately, pip version 25.1.1 is available on our system, ensuring that we can proceed with installing the necessary dependencies for our Django project.



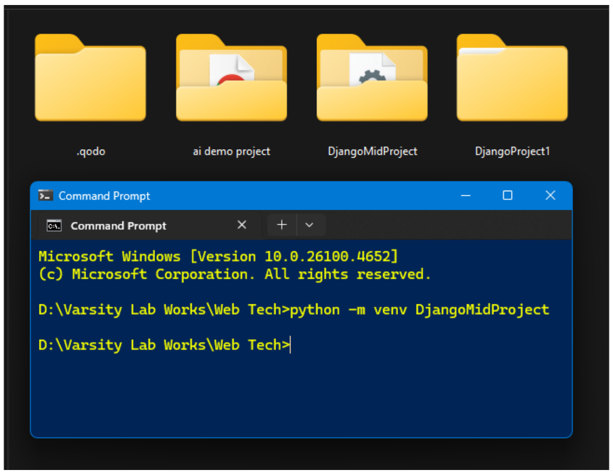
```
Microsoft Windows [Version 10.0.26100.4652]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Dell>python --version
Python 3.13.4

C:\Users\Dell>pip --version
pip 25.1.1 from C:\Program Files\Python313\Lib\site-packages\pip (python 3.13)

C:\Users\Dell>
```

2. The second step involves creating a virtual environment for our project. It is highly recommended to maintain a separate virtual environment for each Django project to ensure proper dependency management and avoid conflicts between different projects. One commonly used tool for creating virtual environments is venv, which comes bundled with Python.

   To set up the virtual environment, we must first navigate to the directory where we intend to develop the project. In our case, this involves moving to the following path on the D drive: "*D:\Varsity Lab Works\Web Tech*". Once within this directory, we proceed to create the virtual environment. The name assigned to the virtual environment is arbitrary and can be chosen according to preference. For this project, we will name the virtual environment "**DjangoMidProject**".

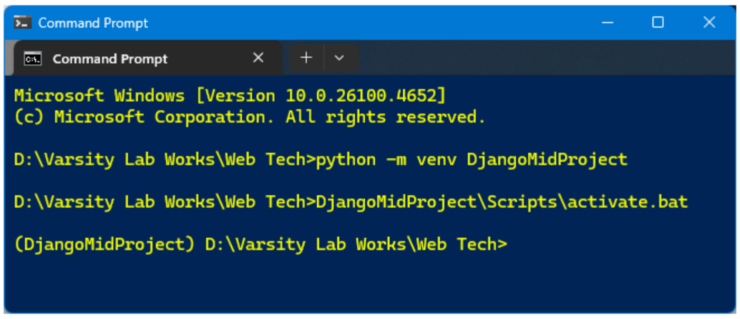
   Therefore, within our chosen directory, we open the Command Prompt and execute the following command: ***python -m venv DjangoMidProject***

   This command generates a folder named "**DjangoMidProject**" in the specified directory, containing all the necessary files to manage our isolated Python environment.



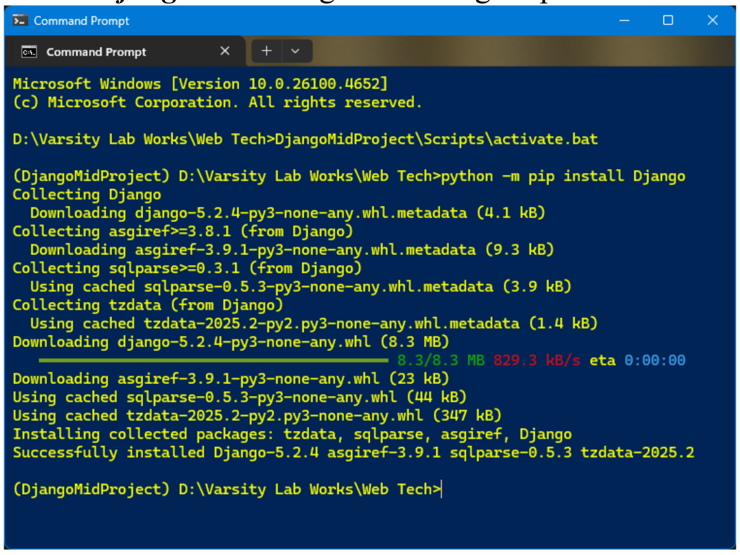
```
Microsoft Windows [Version 10.0.26100.4652]
(c) Microsoft Corporation. All rights reserved.

D:\Varsity Lab Works\Web Tech>python -m venv DjangoMidProject

D:\Varsity Lab Works\Web Tech>
```

   Now, we have to activate the environment, by typing this command: ***DjangoMidProject\Scripts\activate.bat.*** Once activated, we will see the ***(DjangoMidProject) D:\Varsity Lab Works\Web Tech>*** as result in the command prompt.



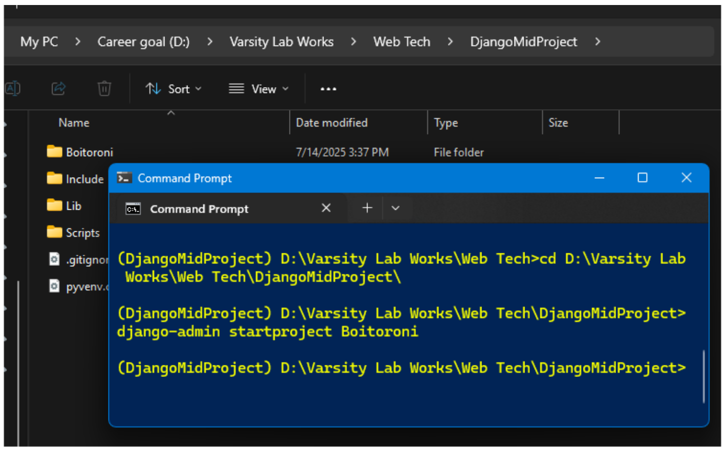
```
Microsoft Windows [Version 10.0.26100.4652]
(c) Microsoft Corporation. All rights reserved.

D:\Varsity Lab Works\Web Tech>python -m venv DjangoMidProject

D:\Varsity Lab Works\Web Tech>DjangoMidProject\Scripts\activate.bat

(DjangoMidProject) D:\Varsity Lab Works\Web Tech>
```

3. Now we have to install Django in the virtual Environment. To do this we have to write ***python -m pip install Django.*** We will get following output of installation.



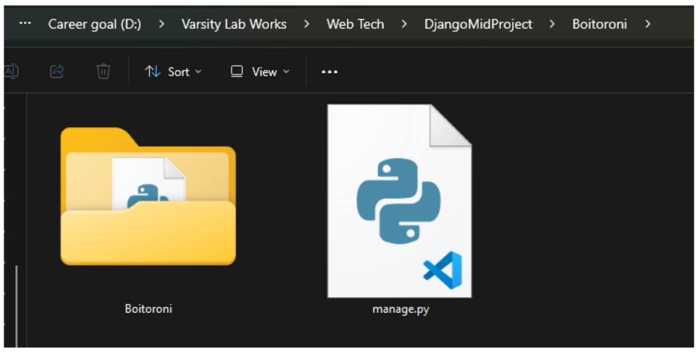
```
Microsoft Windows [Version 10.0.26100.4652]
(c) Microsoft Corporation. All rights reserved.

D:\Varsity Lab Works\Web Tech>DjangoMidProject\Scripts\activate.bat

(DjangoMidProject) D:\Varsity Lab Works\Web Tech>python -m pip install Django
Collecting Django
  Downloading django-5.2.4-py3-none-any.whl.metadata (4.1 kB)
Collecting asgiref>=3.8.1 (from Django)
  Downloading asgiref-3.9.1-py3-none-any.whl.metadata (9.3 kB)
Collecting sqlparse>=0.3.1 (from Django)
  Using cached sqlparse-0.5.3-py3-none-any.whl.metadata (3.9 kB)
Collecting tzdata (from Django)
  Using cached tzdata-2025.2-py2.py3-none-any.whl.metadata (1.4 kB)
Downloading django-5.2.4-py3-none-any.whl (8.3 MB)
   ---------------------------------------- 8.3/8.3 MB 829.3 kB/s eta 0:00:00
Downloading asgiref-3.9.1-py3-none-any.whl (23 kB)
Using cached sqlparse-0.5.3-py3-none-any.whl (44 kB)
Using cached tzdata-2025.2-py2.py3-none-any.whl (347 kB)
Installing collected packages: tzdata, sqlparse, asgiref, Django
Successfully installed Django-5.2.4 asgiref-3.9.1 sqlparse-0.5.3 tzdata-2025.2

(DjangoMidProject) D:\Varsity Lab Works\Web Tech>
```

4. The next step involves creating our Django project, which we will title **"Boitoroni."** To begin, we need to navigate to the existing project directory named **DjangoMidProject**. Once inside this folder, we open the Command Prompt (control panel) and execute the command: ***django-admin startproject Boitoroni*** in the control panel. Executing this command will generate a new folder named **"Boitoroni"** within the **DjangoMidProject** directory. This newly created folder will contain the essential project files and directories required to start development. Notably, the **DjangoMidProject** folder already contains system-generated folders such as **Include**, **Lib**, and **Scripts**, which are part of the virtual environment setup.



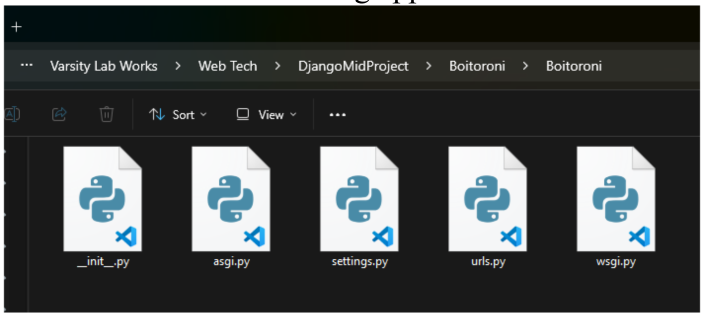
```
(DjangoMidProject) D:\Varsity Lab Works\Web Tech>cd D:\Varsity Lab
 Works\Web Tech\DjangoMidProject\

(DjangoMidProject) D:\Varsity Lab Works\Web Tech\DjangoMidProject>
django-admin startproject Boitoroni

(DjangoMidProject) D:\Varsity Lab Works\Web Tech\DjangoMidProject>
```

5. Inside this "Boitoroni" folder we will find *manage.py* and another "Boitoroni" folder like this



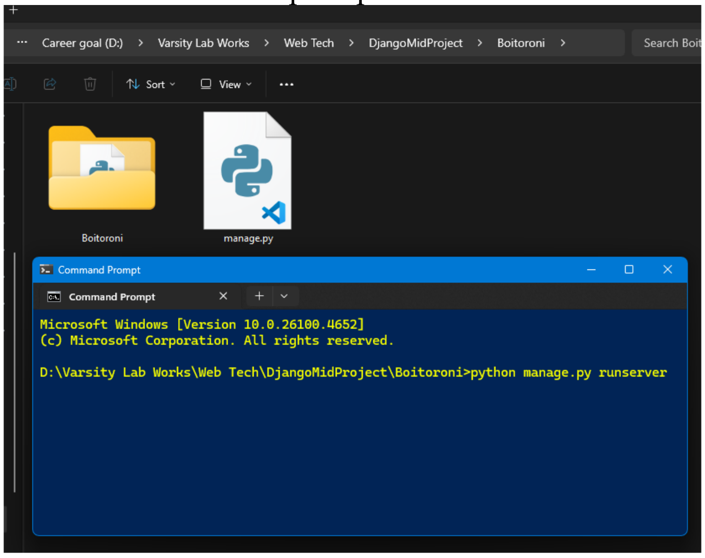
··· Career goal (D:) > Varsity Lab Works > Web Tech > DjangoMidProject > Boitoroni >
Sort
View
Boitoroni
manage.py

Inside this "Boitoroni" Folder, we will find many python files like following. These are all files and folders with a specific meaning, we will learn about some of them later in this Documentation, but for now, it is more important to know that this is the location of our project, and that We can start building applications in it.



··· Varsity Lab Works > Web Tech > DjangoMidProject > Boitoroni > Boitoroni
Sort
View
__init__.py
asgi.py
settings.py
urls.py
wsgi.py

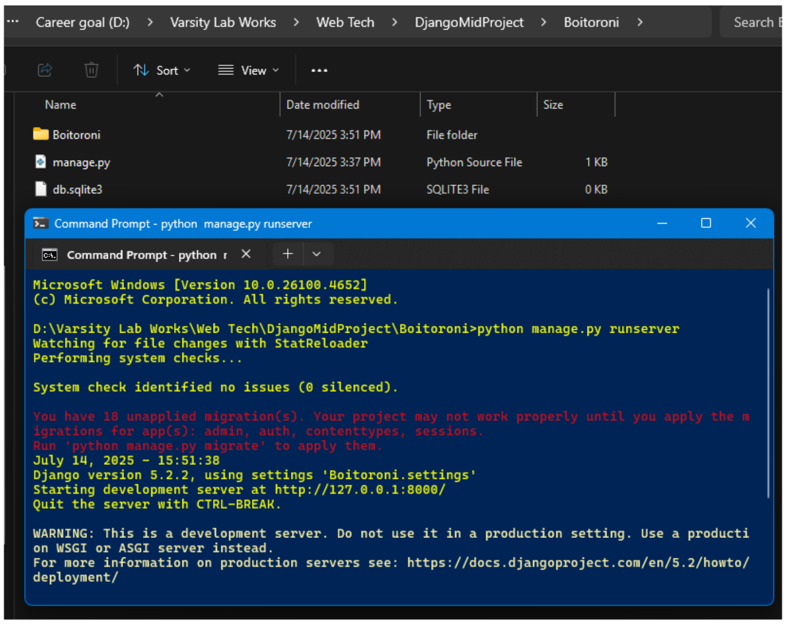
6. Now we Have a Django Project. Now we have to navigate back to the parent "Boitoroni" Folder where *manage.py* exists. There we have to run this command python manage.py runserver in the command prompt.



··· Career goal (D:) > Varsity Lab Works > Web Tech > DjangoMidProject > Boitoroni >
Search Boit
Sort
View
Boitoroni
manage.py
Command Prompt
Command Prompt

```
Microsoft Windows [Version 10.0.26100.4652]
(c) Microsoft Corporation. All rights reserved.

D:\Varsity Lab Works\Web Tech\DjangoMidProject\Boitoroni>python manage.py runserver
```

Running this, we've got the following result. Also, there a database named as "db.sqlite3" has been created.

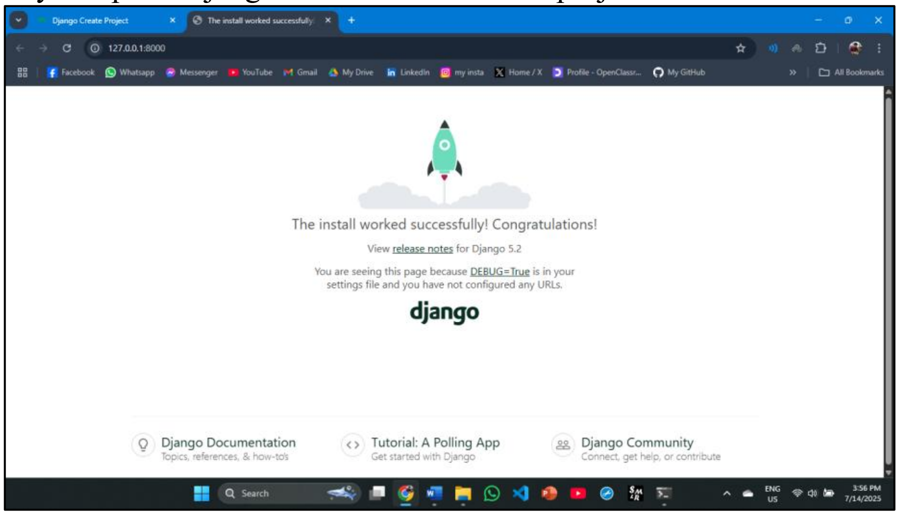


Career goal (D:) > Varsity Lab Works > Web Tech > DjangoMidProject > Boitoroni >
Sort
View

Name	Date modified	Type	Size
Boitoroni	7/14/2025 3:51 PM	File folder	
manage.py	7/14/2025 3:37 PM	Python Source File	1 KB
db.sqlite3	7/14/2025 3:51 PM	SQLITE3 File	0 KB

```
Microsoft Windows [Version 10.0.26100.4652]
(c) Microsoft Corporation. All rights reserved.

D:\Varsity Lab Works\Web Tech\DjangoMidProject\Boitoroni>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 m
igrations for app(s): admin, auth, contenttypes, sessions.
Run 'python manage.py migrate' to apply them.
July 14, 2025 - 15:51:38
Django version 5.2.2, using settings 'Boitoroni.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.

WARNING: This is a development server. Do not use it in a production setting. Use a producti
on WSGI or ASGI server instead.
For more information on production servers see: https://docs.djangoproject.com/en/5.2/howto/
deployment/
```

Also, we have got and address of the deployment server http://127.0.0.1:8000/ by clicking on it we got a Congratulations message on browser that confirms that we have successfully setup the Django Environment for our project.

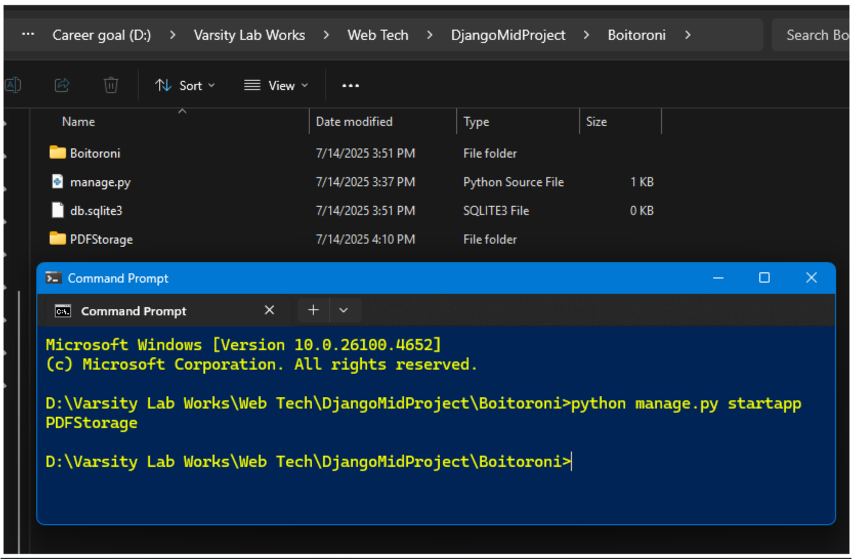


The install worked successfully! Congratulations!

View release notes for Django 5.2

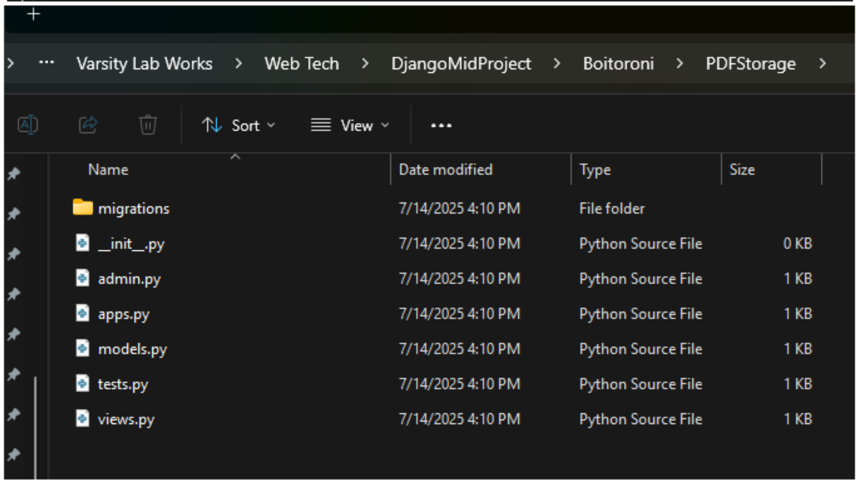
You are seeing this page because DEBUG=True is in your settings file and you have not configured any URLs.

7. Now we are going to Create an App. An app is a web application that has a specific meaning in your project, like a home page, a contact form, or a members database.
   In this tutorial we will create an app that allows us to list and register PDFs in a database.
   But first, let's just create a simple Django app that displays "Hello World!".
   We will name Our app PDFStorage.
   To do so, we have to navigate to the selected location where we want to store the app, we have to navigate back to the parent "Boitoroni" Folder where *manage.py* exists, and run the command: *python manage.py startapp PDFStorage*. A folder named as "PDFStorage" will be created consisting some python files.


Career goal (D:) > Varsity Lab Works > Web Tech > DjangoMidProject > Boitoroni >

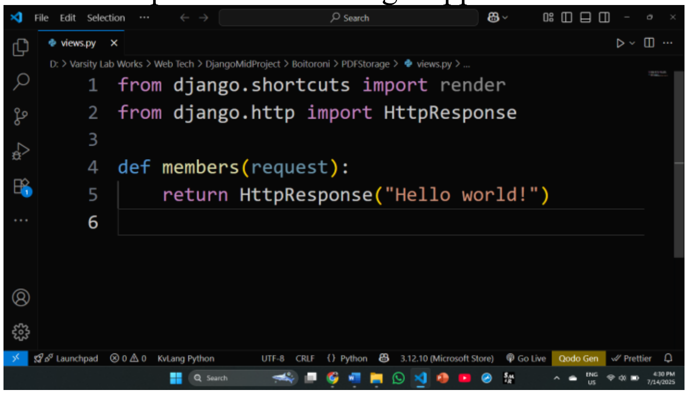
Name	Date modified	Type	Size
Boitoroni	7/14/2025 3:51 PM	File folder	
manage.py	7/14/2025 3:37 PM	Python Source File	1 KB
db.sqlite3	7/14/2025 3:51 PM	SQLITE3 File	0 KB
PDFStorage	7/14/2025 4:10 PM	File folder	

```
Microsoft Windows [Version 10.0.26100.4652]
(c) Microsoft Corporation. All rights reserved.

D:\Varsity Lab Works\Web Tech\DjangoMidProject\Boitoroni>python manage.py startapp
PDFStorage

D:\Varsity Lab Works\Web Tech\DjangoMidProject\Boitoroni>
```


Varsity Lab Works > Web Tech > DjangoMidProject > Boitoroni > PDFStorage >

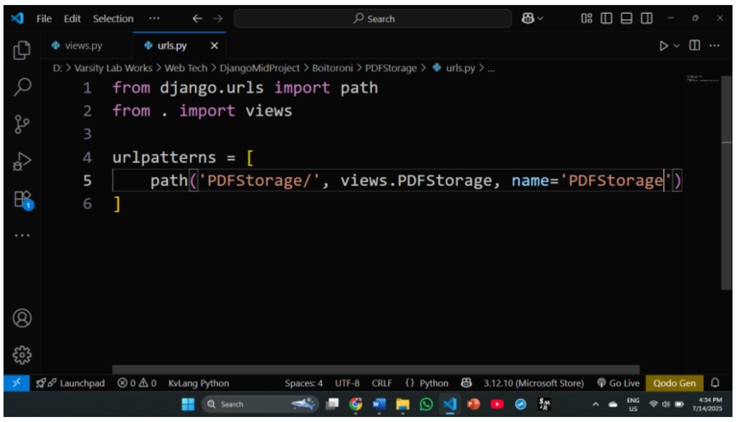
Name	Date modified	Type	Size
migrations	7/14/2025 4:10 PM	File folder	
__init__.py	7/14/2025 4:10 PM	Python Source File	0 KB
admin.py	7/14/2025 4:10 PM	Python Source File	1 KB
apps.py	7/14/2025 4:10 PM	Python Source File	1 KB
models.py	7/14/2025 4:10 PM	Python Source File	1 KB
tests.py	7/14/2025 4:10 PM	Python Source File	1 KB
views.py	7/14/2025 4:10 PM	Python Source File	1 KB

8. Now we have to work with views. Django views are Python functions that take http requests and return http response, like HTML documents. A web page that uses Django is full of views with different tasks and missions. Views are usually put in a file called views.py located on our app's folder.
In our case we have a views.py in our PDFStorage app can be seen on above screenshot. So in that file, we have to replace the following snippet.


views.py
D: > Varsity Lab Works > Web Tech > DjangoMidProject > Boitoroni > PDFStorage > views.py > ...

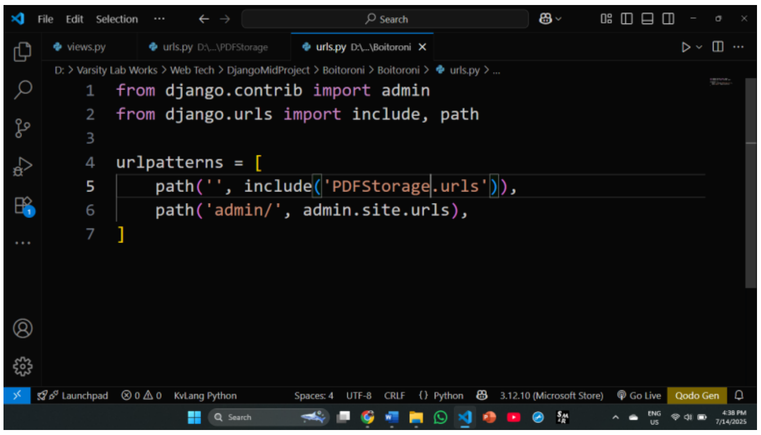
```python
from django.shortcuts import render
from django.http import HttpResponse

def members(request):
    return HttpResponse("Hello world!")

```

9. The 8th Procedure will Show “Hello World!” as response but still we have to perform some tasks. Now we have to setup the connections of the pages. To do that we have to Create a file named urls.py in the same folder as the views.py file, and type this code in it:



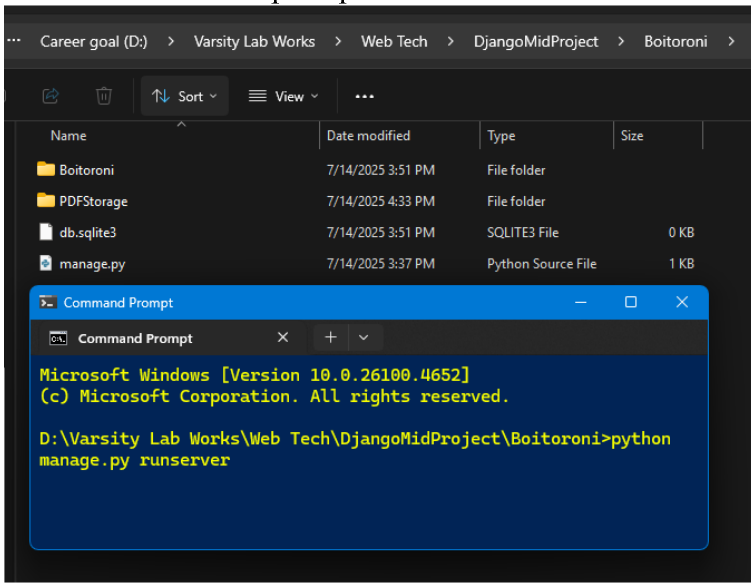
```python
from django.urls import path
from . import views

urlpatterns = [
    path('PDFStorage/', views.PDFStorage, name='PDFStorage')
]
```

There is a file called urls.py on the Boitoroni folder. We have to add the include module in the import statement, and also add a path() function in the urlpatterns[] list, with arguments that will route users that comes in via 127.0.0.1:8000/.



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

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

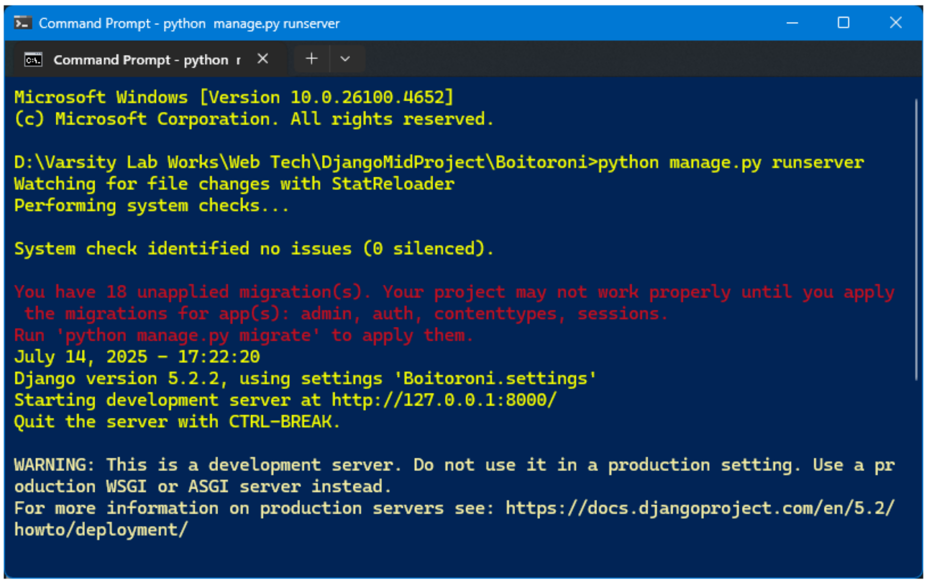
10. Conection establishing done. Now to check the app is running or not, we have to navigate to "Boitoroni" folder where manage.py exists, and run the command : *python manage.py runserver* in command prompt.



Career goal (D:) > Varsity Lab Works > Web Tech > DjangoMidProject > Boitoroni >

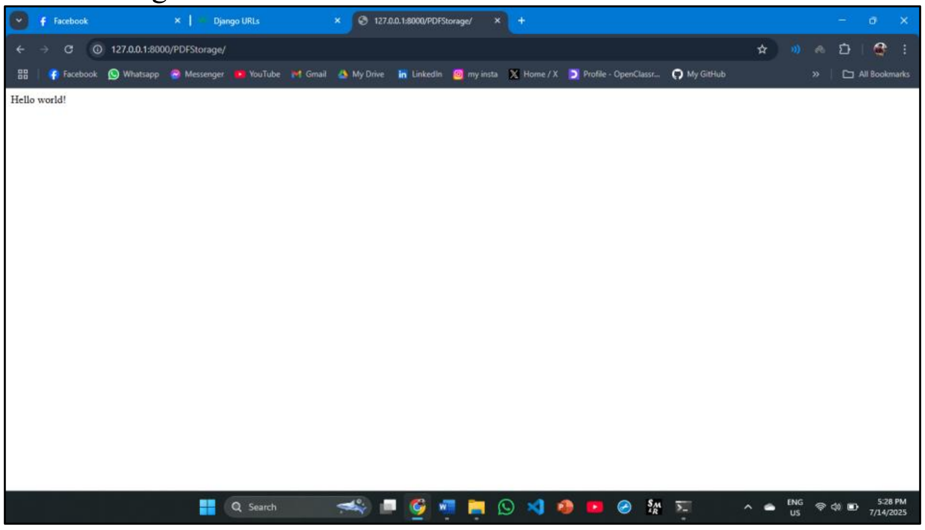
Name	Date modified	Type	Size
Boitoroni	7/14/2025 3:51 PM	File folder	
PDFStorage	7/14/2025 4:33 PM	File folder	
db.sqlite3	7/14/2025 3:51 PM	SQLITE3 File	0 KB
manage.py	7/14/2025 3:37 PM	Python Source File	1 KB

```
Microsoft Windows [Version 10.0.26100.4652]
(c) Microsoft Corporation. All rights reserved.

D:\Varsity Lab Works\Web Tech\DjangoMidProject\Boitoroni>python
manage.py runserver
```



```
Microsoft Windows [Version 10.0.26100.4652]
(c) Microsoft Corporation. All rights reserved.

D:\Varsity Lab Works\Web Tech\DjangoMidProject\Boitoroni>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, contenttypes, sessions.
Run 'python manage.py migrate' to apply them.
July 14, 2025 - 17:22:20
Django version 5.2.2, using settings 'Boitoroni.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.

WARNING: This is a development server. Do not use it in a production setting. Use a pr
oduction WSGI or ASGI server instead.
For more information on production servers see: https://docs.djangoproject.com/en/5.2/
howto/deployment/
```

Now by running this *http://127.0.0.1:8000/PDFStorage/* on browser, we will get the Hello World message.



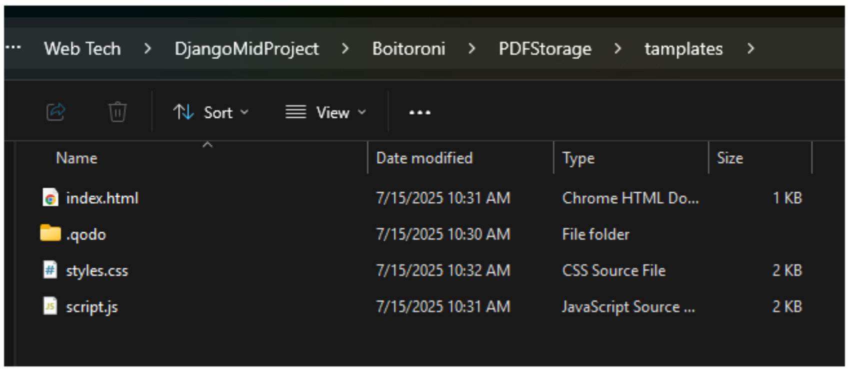
Hello world!

11. At this stage, our Django app is successfully running. The next objective is to begin constructing the website by creating an HTML file that will define the structure and content of the web pages.

In Django, HTML files are typically placed within a **template's directory**, which the framework uses to locate and render dynamic web pages. To establish this structure for our project, we will proceed as follows:

- **Create a folder named templates** inside the existing **PDFStorage** app directory. This folder will serve as the designated location for all HTML files related to this app.
- **Within the templates folder, create an HTML file** named **index.html**. This file will serve as the initial webpage for our app and will later be configured to display content such as the list of available PDFs or other dynamic data.

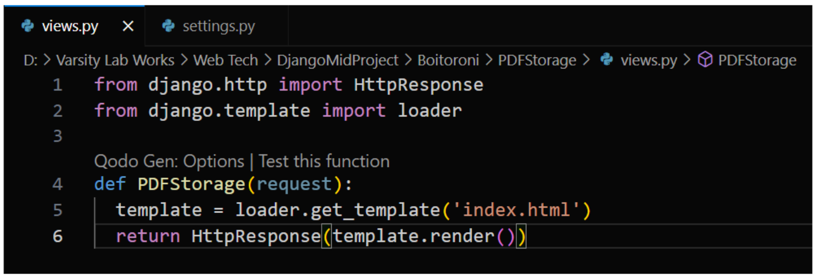
By following this structure, Django will be able to identify and render the HTML templates appropriately when handling user requests and responses.



··· Web Tech > DjangoMidProject > Boitoroni > PDFStorage > templates >
Sort
View

Name	Date modified	Type	Size
index.html	7/15/2025 10:31 AM	Chrome HTML Do...	1 KB
.qodo	7/15/2025 10:30 AM	File folder	
styles.css	7/15/2025 10:32 AM	CSS Source File	2 KB
script.js	7/15/2025 10:31 AM	JavaScript Source ...	2 KB

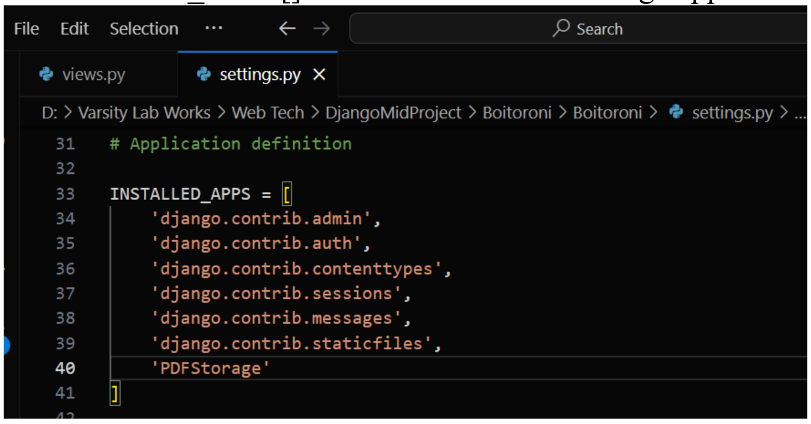
12. Now we have to modify the views.py located in PDFStorage folder as we now want to display the index.html page instead of Hello World. In there, we replaced the following snippets.



views.py
settings.py
D: > Varsity Lab Works > Web Tech > DjangoMidProject > Boitoroni > PDFStorage > views.py > PDFStorage
Qodo Gen: Options | Test this function

```python
from django.http import HttpResponse
from django.template import loader

def PDFStorage(request):
    template = loader.get_template('index.html')
    return HttpResponse(template.render())
```

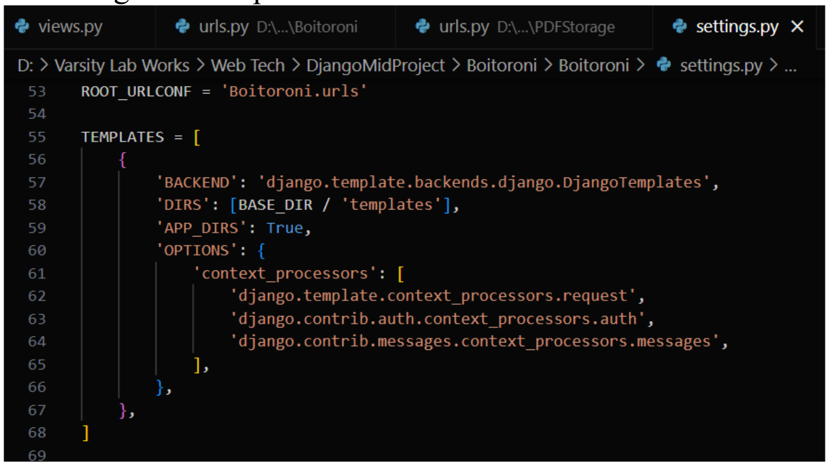
13. To be able to work with more complicated stuff than "Hello World!", We have to tell Django that a new app is created.
This is done in the settings.py file in the Boitoroni folder.
There in the INSTALLED_APPS[] list and add the PDFStorage app like this:



File Edit Selection ···
Search
views.py
settings.py
D: > Varsity Lab Works > Web Tech > DjangoMidProject > Boitoroni > Boitoroni > settings.py > ...

```python
# Application definition

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

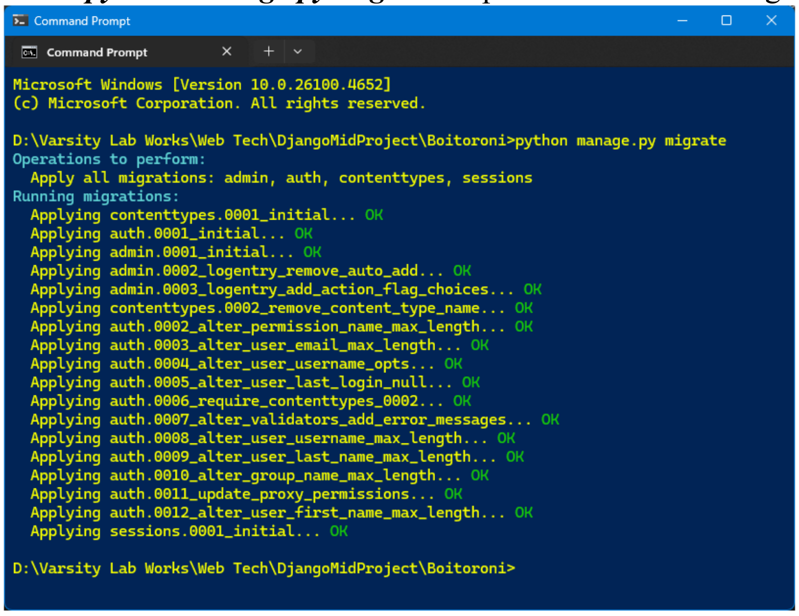
Also did some changes in Templates



views.py
urls.py D:\...\Boitoroni
urls.py D:\...\PDFStorage
settings.py
D: > Varsity Lab Works > Web Tech > DjangoMidProject > Boitoroni > Boitoroni > settings.py > ...

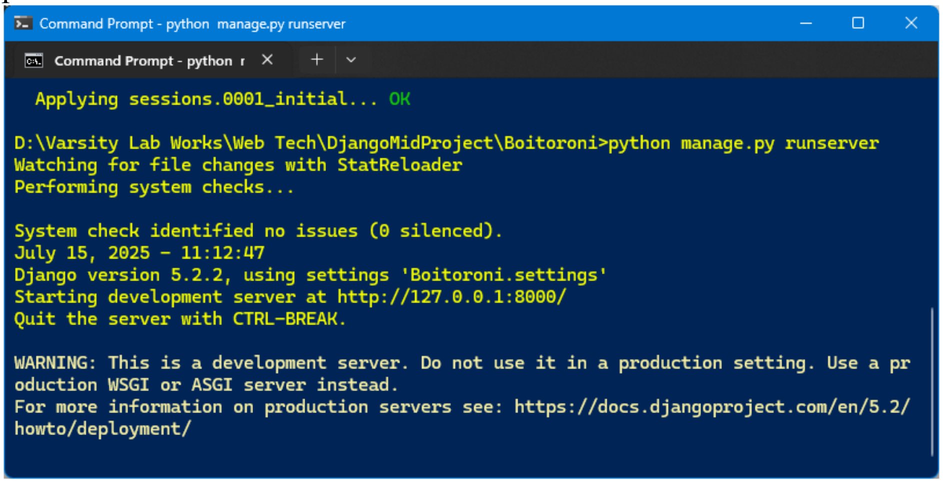
```python
ROOT_URLCONF = 'Boitoroni.urls'

TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [BASE_DIR / 'templates'],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]
```

14. Now we have to navigate to the parent Boitoroni folder where manage.py exists and run the command: ***python manage.py migrate.*** It produced the following output.



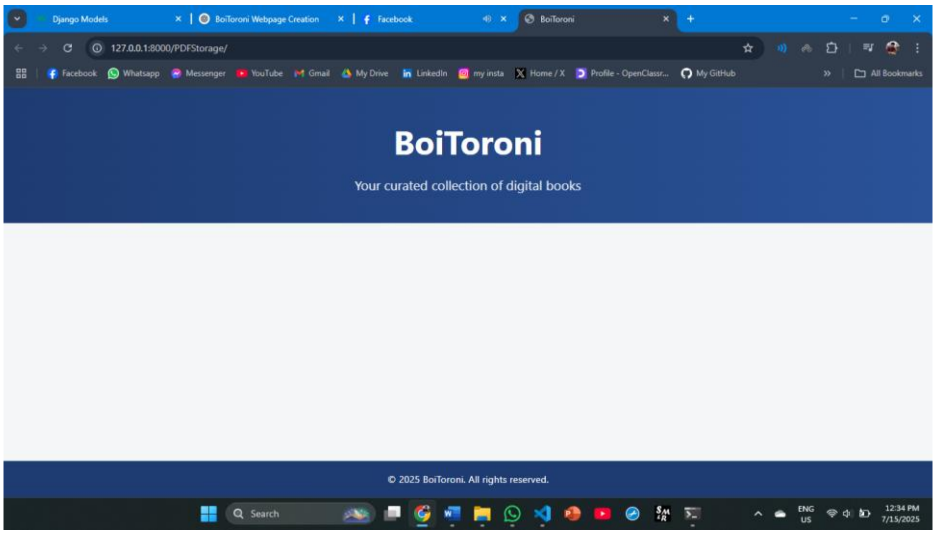
```
Microsoft Windows [Version 10.0.26100.4652]
(c) Microsoft Corporation. All rights reserved.

D:\Varsity Lab Works\Web Tech\DjangoMidProject\Boitoroni>python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, sessions
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying admin.0002_logentry_remove_auto_add... OK
  Applying admin.0003_logentry_add_action_flag_choices... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0002_alter_permission_name_max_length... OK
  Applying auth.0003_alter_user_email_max_length... OK
  Applying auth.0004_alter_user_username_opts... OK
  Applying auth.0005_alter_user_last_login_null... OK
  Applying auth.0006_require_contenttypes_0002... OK
  Applying auth.0007_alter_validators_add_error_messages... OK
  Applying auth.0008_alter_user_username_max_length... OK
  Applying auth.0009_alter_user_last_name_max_length... OK
  Applying auth.0010_alter_group_name_max_length... OK
  Applying auth.0011_update_proxy_permissions... OK
  Applying auth.0012_alter_user_first_name_max_length... OK
  Applying sessions.0001_initial... OK

D:\Varsity Lab Works\Web Tech\DjangoMidProject\Boitoroni>
```

15. Now in there we have to run the server by the command: ***python manage.py runserver***. it will provide a link.


Command Prompt - python manage.py runserver
Command Prompt - python r

```
  Applying sessions.0001_initial... OK

D:\Varsity Lab Works\Web Tech\DjangoMidProject\Boitoroni>python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
July 15, 2025 - 11:12:47
Django version 5.2.2, using settings 'Boitoroni.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.

WARNING: This is a development server. Do not use it in a production setting. Use a pr
oduction WSGI or ASGI server instead.
For more information on production servers see: https://docs.djangoproject.com/en/5.2/
howto/deployment/
```

16. By visiting http://127.0.0.1:8000/PDFStorage/ we got the following webpage opened on browser.

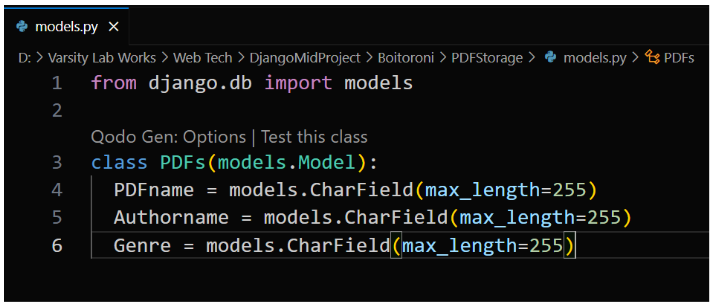


BoiToroni

Your curated collection of digital books

© 2025 BoiToroni. All rights reserved.

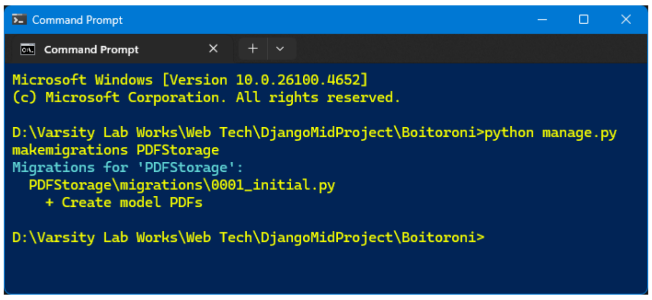
17. Up until now in this tutorial, output has been static data from Python or HTML templates. Now we will see how Django allows us to work with data, without having to change or upload files in the process. In Django, data is created in objects, called Models, and is actually tables in a database.
To create a model, we navigate to the models.py file in the **/PDFStorage/** folder. Opened it, and add a **PDFs** table by creating a **PDFs** class, and describe the table fields in it:


models.py
D: > Varsity Lab Works > Web Tech > DjangoMidProject > Boitoroni > PDFStorage > models.py > PDFs
Qodo Gen: Options | Test this class

```python
from django.db import models

class PDFs(models.Model):
    PDFname = models.CharField(max_length=255)
    Authorname = models.CharField(max_length=255)
    Genre = models.CharField(max_length=255)
```

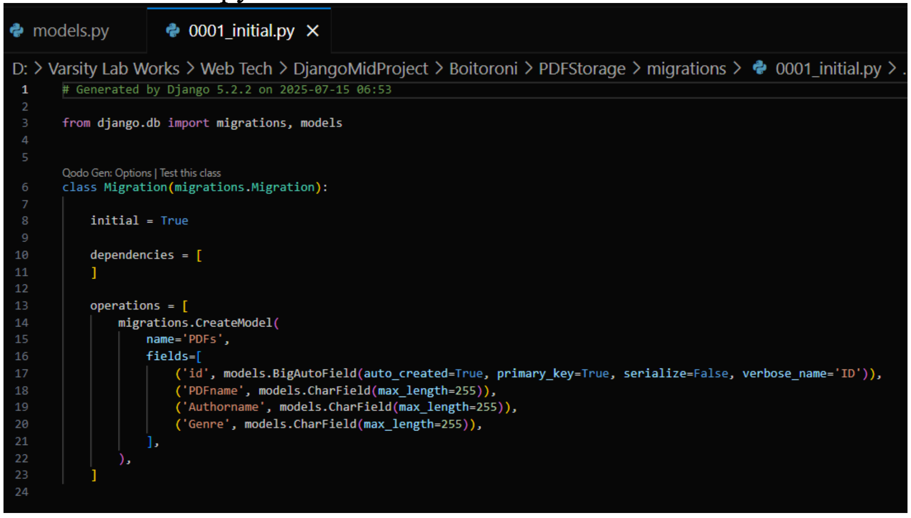
Here,
The first field, ***PDFname***, is a Text field, and will contain the name of the Pdfs.
The second field, ***Authorname***, is also a Text field, with the Author's name.
Ther Third Field, ***Genre,*** is also a Text Field, with the Genre's name
All of the 3 fields are set up to have a maximum of 255 characters.

18. Now, again navigated to Boitoroni folder where manage.py exists, we opened command prompt and wrote: python manage.py makemigrations PDFStorage. The following result came out.



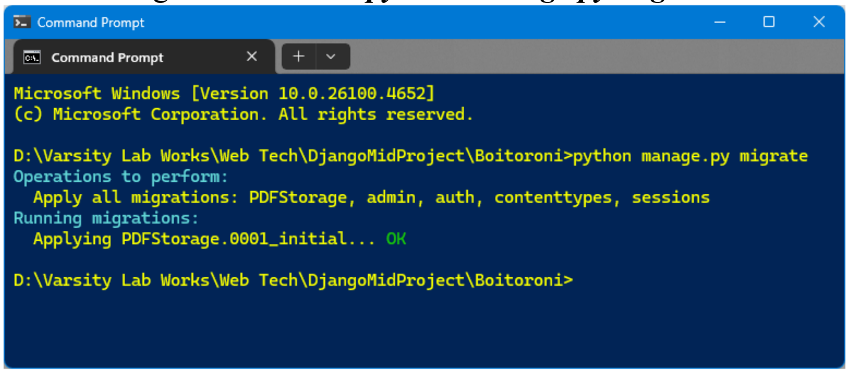
```
Microsoft Windows [Version 10.0.26100.4652]
(c) Microsoft Corporation. All rights reserved.

D:\Varsity Lab Works\Web Tech\DjangoMidProject\Boitoroni>python manage.py
makemigrations PDFStorage
Migrations for 'PDFStorage':
  PDFStorage\migrations\0001_initial.py
    + Create model PDFs

D:\Varsity Lab Works\Web Tech\DjangoMidProject\Boitoroni>
```

19. Now Django creates a file describing the changes and stores the file in the /migrations/ folder. Here we got 0001_initial.py file that consists our table we created above in models.py


models.py
0001_initial.py
D: > Varsity Lab Works > Web Tech > DjangoMidProject > Boitoroni > PDFStorage > migrations > 0001_initial.py >
Qodo Gen: Options | Test this class

```python
# Generated by Django 5.2.2 on 2025-07-15 06:53

from django.db import migrations, models


class Migration(migrations.Migration):

    initial = True

    dependencies = [
    ]

    operations = [
        migrations.CreateModel(
            name='PDFs',
            fields=[
                ('id', models.BigAutoField(auto_created=True, primary_key=True, serialize=False, verbose_name='ID')),
                ('PDFname', models.CharField(max_length=255)),
                ('Authorname', models.CharField(max_length=255)),
                ('Genre', models.CharField(max_length=255)),
            ],
        ),
    ]
```

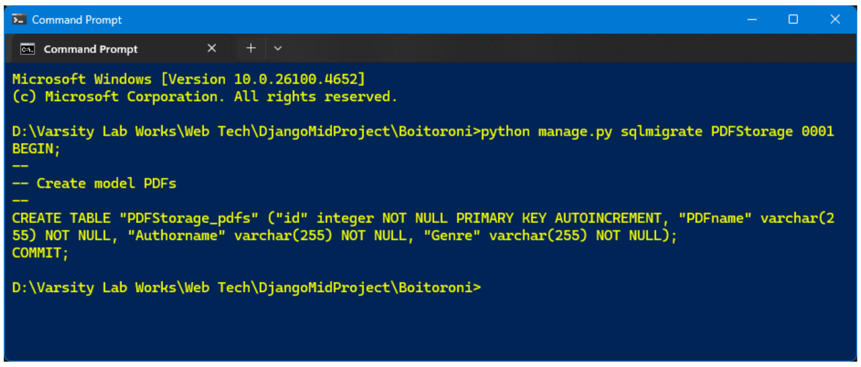
Here,
We should Note that Django inserts an id field for our tables, which is an auto increment number (first record gets the value 1, the second record 2 etc.), this is the default behavior of Django, We can override it by describing our own id field. But for now, we are keeping it as it is.
The table is not created yet, we will have to run one more command, then Django will create and execute an SQL statement, based on the content of the new file in the /migrations/ folder.
We need to run the migrate command: ***python manage.py migrate***



```
Microsoft Windows [Version 10.0.26100.4652]
(c) Microsoft Corporation. All rights reserved.

D:\Varsity Lab Works\Web Tech\DjangoMidProject\Boitoroni>python manage.py migrate
Operations to perform:
  Apply all migrations: PDFStorage, admin, auth, contenttypes, sessions
Running migrations:
  Applying PDFStorage.0001_initial... OK

D:\Varsity Lab Works\Web Tech\DjangoMidProject\Boitoroni>
```

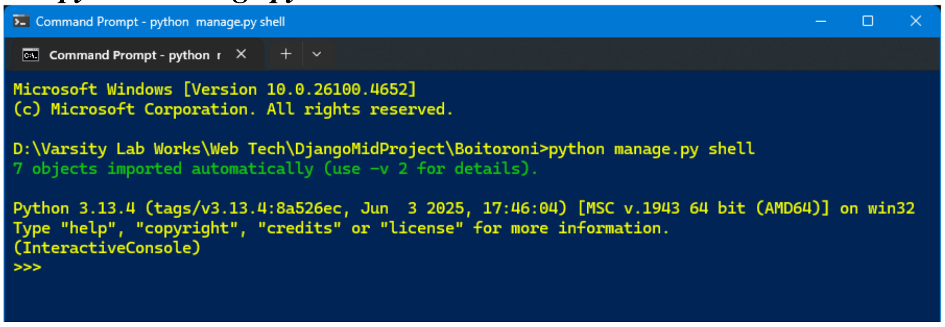
Now we Have a PDFs Table is the database.

20. To Check the SQL statement that were executed from the migration above. All we have to do is to run this command: ***python manage.py sqlmigrate PDFStorage 0001***, with the migration number.



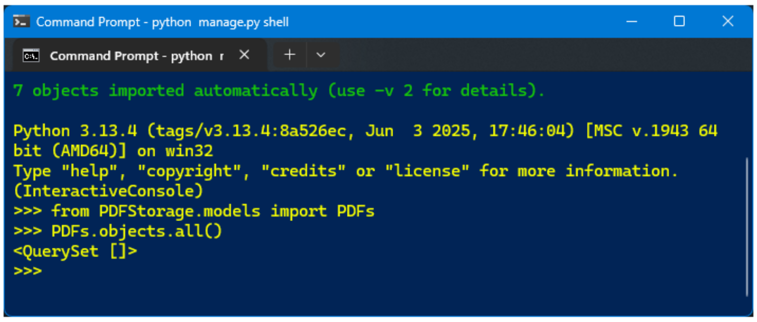
```
Microsoft Windows [Version 10.0.26100.4652]
(c) Microsoft Corporation. All rights reserved.

D:\Varsity Lab Works\Web Tech\DjangoMidProject\Boitoroni>python manage.py sqlmigrate PDFStorage 0001
BEGIN;
--
-- Create model PDFs
--
CREATE TABLE "PDFStorage_pdfs" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT, "PDFname" varchar(2
55) NOT NULL, "Authorname" varchar(255) NOT NULL, "Genre" varchar(255) NOT NULL);
COMMIT;

D:\Varsity Lab Works\Web Tech\DjangoMidProject\Boitoroni>
```

21. The PDFs table created in the previous step is empty. We will use the Python interpreter (Python shell) to add some PDF information to it. To open a Python shell, we type this command: ***python manage.py shell***



```
Microsoft Windows [Version 10.0.26100.4652]
(c) Microsoft Corporation. All rights reserved.

D:\Varsity Lab Works\Web Tech\DjangoMidProject\Boitoroni>python manage.py shell
7 objects imported automatically (use -v 2 for details).

Python 3.13.4 (tags/v3.13.4:8a526ec, Jun  3 2025, 17:46:04) [MSC v.1943 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
(InteractiveConsole)
>>>
```

At the bottom, after the three >>> write: >>> from PDFStorage.models import PDFs
Then Hit [enter] and write this to look at the empty PDFs table: >>> PDFs.objects.all(). This gave us an empty QuerySet object, like this:

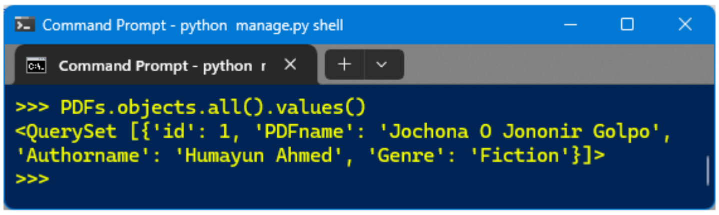


Command Prompt - python manage.py shell
Command Prompt - python r

```
7 objects imported automatically (use -v 2 for details).

Python 3.13.4 (tags/v3.13.4:8a526ec, Jun  3 2025, 17:46:04) [MSC v.1943 64
bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
(InteractiveConsole)
>>> from PDFStorage.models import PDFs
>>> PDFs.objects.all()
<QuerySet []>
>>>
```

A QuerySet is a collection of data from a database.

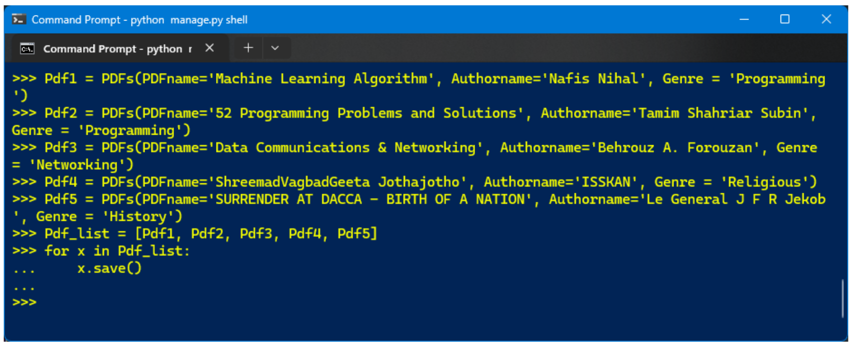
22. Now we add a record to the table, by executing these two lines:
>>> ***Pdfs = PDFs(PDFname='Jochona O Jononir Golpo', Authorname='Humayun Ahmed', Genre = 'Fiction')***
>>> ***Pdfs.save()***
Then we Execute this command to see if the Member table got a member:
>>> ***PDFs.objects.all().values()***
The data we entered shows as result.



Command Prompt - python manage.py shell
Command Prompt - python r

```
>>> PDFs.objects.all().values()
<QuerySet [{'id': 1, 'PDFname': 'Jochona O Jononir Golpo',
'Authorname': 'Humayun Ahmed', 'Genre': 'Fiction'}]>
>>>
```

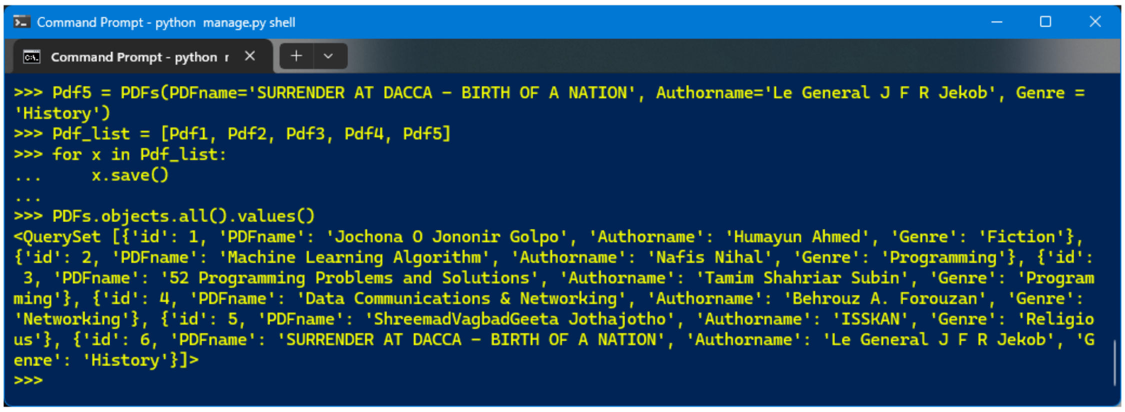
23. So, we successfully added a data to our database. Now let's add 5 more data to our database.



Command Prompt - python manage.py shell
Command Prompt - python r

```
>>> Pdf1 = PDFs(PDFname='Machine Learning Algorithm', Authorname='Nafis Nihal', Genre = 'Programming
')
>>> Pdf2 = PDFs(PDFname='52 Programming Problems and Solutions', Authorname='Tamim Shahriar Subin',
Genre = 'Programming')
>>> Pdf3 = PDFs(PDFname='Data Communications & Networking', Authorname='Behrouz A. Forouzan', Genre
= 'Networking')
>>> Pdf4 = PDFs(PDFname='ShreemadVagbadGeeta Jothajotho', Authorname='ISSKAN', Genre = 'Religious')
>>> Pdf5 = PDFs(PDFname='SURRENDER AT DACCA - BIRTH OF A NATION', Authorname='Le General J F R Jekob
', Genre = 'History')
>>> Pdf_list = [Pdf1, Pdf2, Pdf3, Pdf4, Pdf5]
>>> for x in Pdf_list:
...     x.save()
...
>>>
```

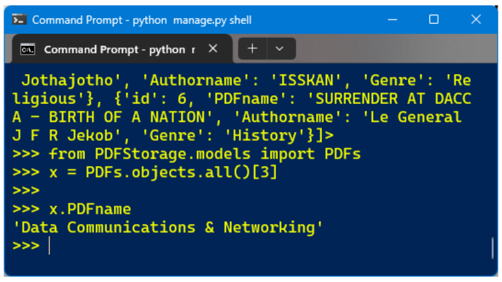
24. Now, to see all the values in database, we ran this command:
>>> ***PDFs.objects.all().values()***
All the data will be shown from the database like this:



Command Prompt - python manage.py shell
Command Prompt - python r

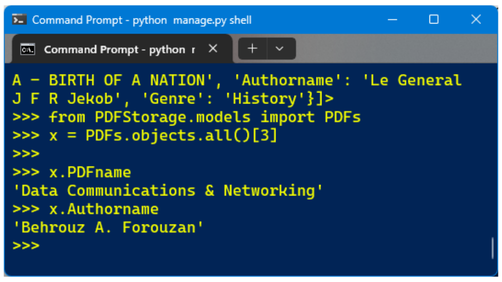
```
>>> Pdf5 = PDFs(PDFname='SURRENDER AT DACCA - BIRTH OF A NATION', Authorname='Le General J F R Jekob', Genre =
'History')
>>> Pdf_list = [Pdf1, Pdf2, Pdf3, Pdf4, Pdf5]
>>> for x in Pdf_list:
...     x.save()
...
>>> PDFs.objects.all().values()
<QuerySet [{'id': 1, 'PDFname': 'Jochona O Jononir Golpo', 'Authorname': 'Humayun Ahmed', 'Genre': 'Fiction'},
{'id': 2, 'PDFname': 'Machine Learning Algorithm', 'Authorname': 'Nafis Nihal', 'Genre': 'Programming'}, {'id':
 3, 'PDFname': '52 Programming Problems and Solutions', 'Authorname': 'Tamim Shahriar Subin', 'Genre': 'Program
ming'}, {'id': 4, 'PDFname': 'Data Communications & Networking', 'Authorname': 'Behrouz A. Forouzan', 'Genre':
'Networking'}, {'id': 5, 'PDFname': 'ShreemadVagbadGeeta Jothajotho', 'Authorname': 'ISSKAN', 'Genre': 'Religio
us'}, {'id': 6, 'PDFname': 'SURRENDER AT DACCA - BIRTH OF A NATION', 'Authorname': 'Le General J F R Jekob', 'G
enre': 'History'}]>
>>>
```

25. Now let's see how to update a value in the database. To do so, we first have to get the record we want to update. In Our case, we are going to change the Author name of the 3rd book. We will replace 'Tamim Shahriar Subin' with 'Tahmid Rafi'. To do so, we wrote the following commands in the Command prompt.
>>> from PDFStorage.models import PDFs
>>> x = PDFs.objects.all()[3]
x will now represent the member at index 3, but to make sure, let us see if that is correct:
>>> x.PDFname



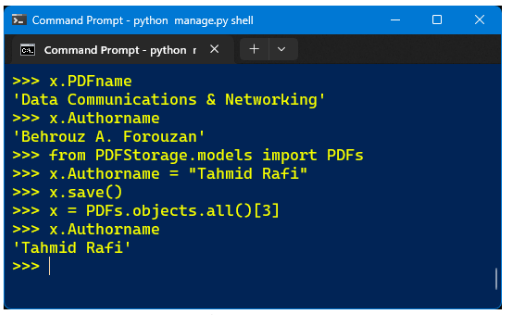
```
Command Prompt - python  manage.py shell
 Jothajotho', 'Authorname': 'ISSKAN', 'Genre': 'Re
ligious'}, {'id': 6, 'PDFname': 'SURRENDER AT DACC
A - BIRTH OF A NATION', 'Authorname': 'Le General
J F R Jekob', 'Genre': 'History'}]>
>>> from PDFStorage.models import PDFs
>>> x = PDFs.objects.all()[3]
>>>
>>> x.PDFname
'Data Communications & Networking'
>>> |
```

>>>x.Authorname



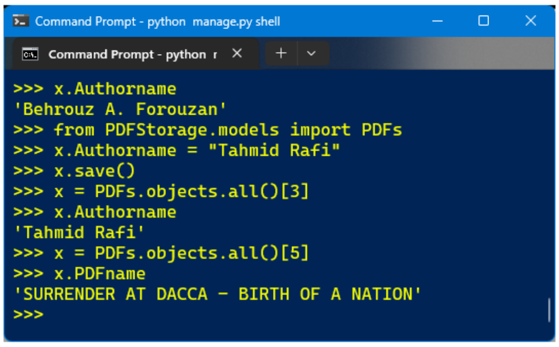
```
Command Prompt - python  manage.py shell
A - BIRTH OF A NATION', 'Authorname': 'Le General
J F R Jekob', 'Genre': 'History'}]>
>>> from PDFStorage.models import PDFs
>>> x = PDFs.objects.all()[3]
>>>
>>> x.PDFname
'Data Communications & Networking'
>>> x.Authorname
'Behrouz A. Forouzan'
>>>
```

Now to Change the value in Author name, we wrote the following command.
>>> x.Authorname = " Tahmid Rafi"
>>> x.save()
The Entry will be updated. Now lets check the entry again running following command.
>>> x = PDFs.objects.all()[3]
>>> x.Authorname
We will get the updated Result.



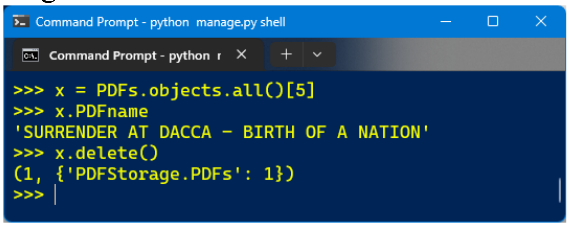
```
Command Prompt - python  manage.py shell
>>> x.PDFname
'Data Communications & Networking'
>>> x.Authorname
'Behrouz A. Forouzan'
>>> from PDFStorage.models import PDFs
>>> x.Authorname = "Tahmid Rafi"
>>> x.save()
>>> x = PDFs.objects.all()[3]
>>> x.Authorname
'Tahmid Rafi'
>>> |
```

26. Now let's assume we have to delete 5th Index entry from the database. To do so, we first have to get the 5th record we want to Delete. To do so, we wrote following command:
>>> from PDFStorage.models import PDFs
>>> x = PDFs.objects.all()[5]
x will now represent the PDF information at index 5, but to make sure, let us see if that is correct:
>>>x.PDFname


Command Prompt - python manage.py shell
Command Prompt - python r

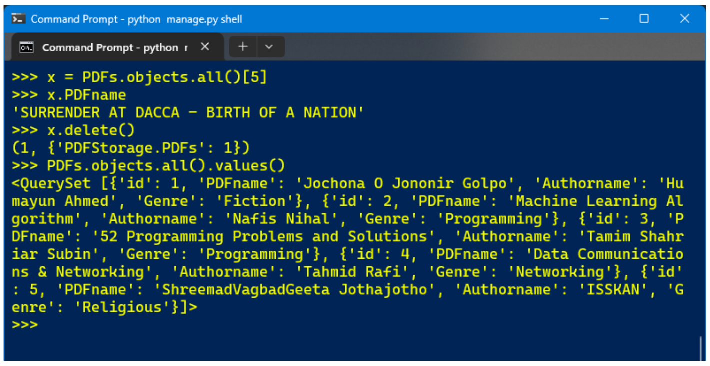
```
>>> x.Authorname
'Behrouz A. Forouzan'
>>> from PDFStorage.models import PDFs
>>> x.Authorname = "Tahmid Rafi"
>>> x.save()
>>> x = PDFs.objects.all()[3]
>>> x.Authorname
'Tahmid Rafi'
>>> x = PDFs.objects.all()[5]
>>> x.PDFname
'SURRENDER AT DACCA - BIRTH OF A NATION'
>>>
```

Now we can delete the record writing this command:
>>> x.delete()
We will get the following result.


Command Prompt - python manage.py shell
Command Prompt - python r

```
>>> x = PDFs.objects.all()[5]
>>> x.PDFname
'SURRENDER AT DACCA - BIRTH OF A NATION'
>>> x.delete()
(1, {'PDFStorage.PDFs': 1})
>>> |
```

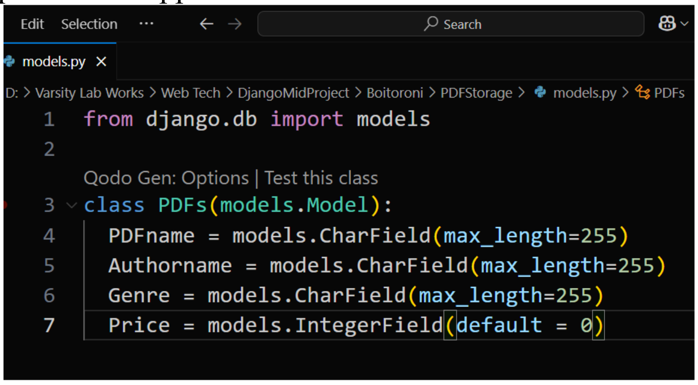
It tells us how many items were deleted, and from which Model. Now let's view the whole database again to confirm the deletion.


Command Prompt - python manage.py shell
Command Prompt - python r

```
>>> x = PDFs.objects.all()[5]
>>> x.PDFname
'SURRENDER AT DACCA - BIRTH OF A NATION'
>>> x.delete()
(1, {'PDFStorage.PDFs': 1})
>>> PDFs.objects.all().values()
<QuerySet [{'id': 1, 'PDFname': 'Jochona O Jononir Golpo', 'Authorname': 'Hu
mayun Ahmed', 'Genre': 'Fiction'}, {'id': 2, 'PDFname': 'Machine Learning Al
gorithm', 'Authorname': 'Nafis Nihal', 'Genre': 'Programming'}, {'id': 3, 'P
DFname': '52 Programming Problems and Solutions', 'Authorname': 'Tamim Shahr
iar Subin', 'Genre': 'Programming'}, {'id': 4, 'PDFname': 'Data Communicatio
ns & Networking', 'Authorname': 'Tahmid Rafi', 'Genre': 'Networking'}, {'id'
: 5, 'PDFname': 'ShreemadVagbadGeeta Jothajotho', 'Authorname': 'ISSKAN', 'G
enre': 'Religious'}]>
>>>
```

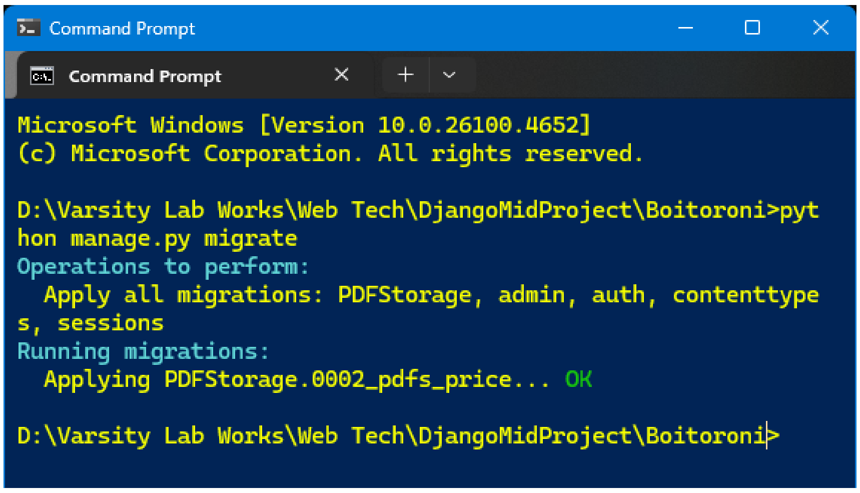
Yes, the entry of the pdf named "SURRENDER AT DACCA – BIRTH OF A NATION" successfully deleted.

27. We can update models too. Suppose we will need to add another field to our database. To do so, we have to follow 19th step. In our case we will add "Price" field to our database. To do so, we have to go again in models.py located in PDFStorage folder. There we will update the snippet.


Edit
Selection
Search
models.py
D: > Varsity Lab Works > Web Tech > DjangoMidProject > Boitoroni > PDFStorage > models.py > PDFs
Qodo Gen: Options | Test this class

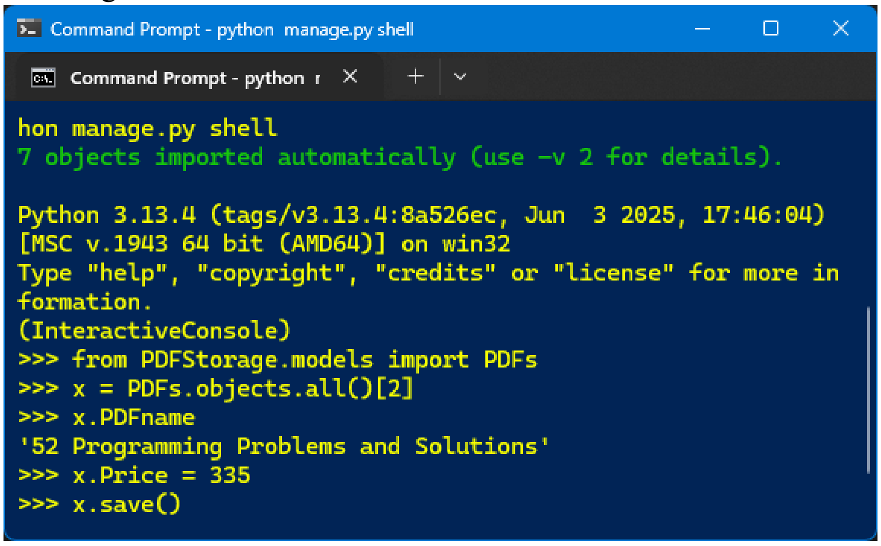
```python
from django.db import models

class PDFs(models.Model):
    PDFname = models.CharField(max_length=255)
    Authorname = models.CharField(max_length=255)
    Genre = models.CharField(max_length=255)
    Price = models.IntegerField(default = 0)
```

Now we make the migration once again with this command: ***python manage.py migrate***



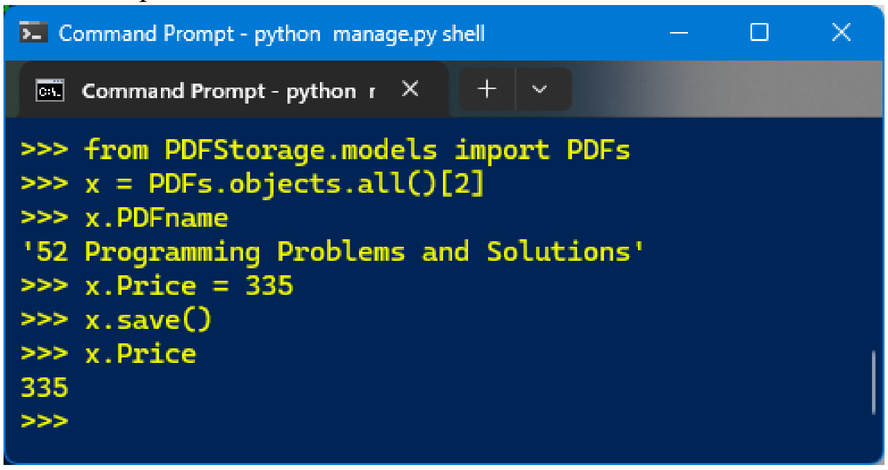
```
Microsoft Windows [Version 10.0.26100.4652]
(c) Microsoft Corporation. All rights reserved.

D:\Varsity Lab Works\Web Tech\DjangoMidProject\Boitoroni>pyt
hon manage.py migrate
Operations to perform:
  Apply all migrations: PDFStorage, admin, auth, contenttype
s, sessions
Running migrations:
  Applying PDFStorage.0002_pdfs_price... OK

D:\Varsity Lab Works\Web Tech\DjangoMidProject\Boitoroni>
```

28. Now let's insert a data into the field by updating the table in the same way in python shell we did on 25th Step. We will update the price = 335 of index = 2, ID = 3 pdf named "52 Programming Problems and Solutions'"



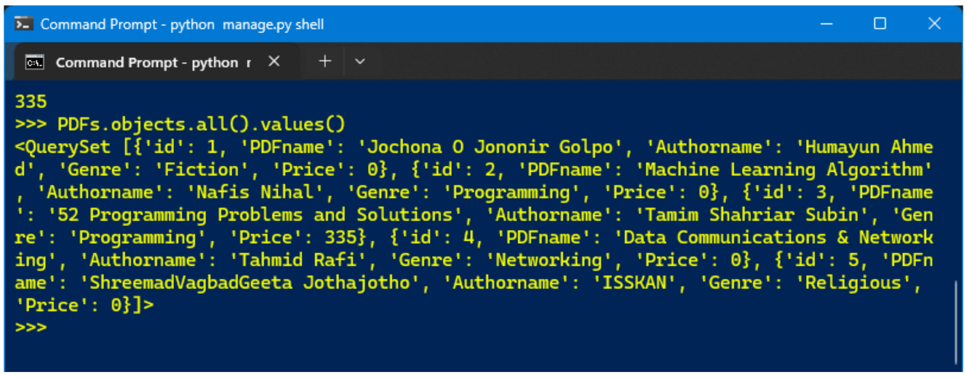
```
hon manage.py shell
7 objects imported automatically (use -v 2 for details).

Python 3.13.4 (tags/v3.13.4:8a526ec, Jun  3 2025, 17:46:04)
[MSC v.1943 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more in
formation.
(InteractiveConsole)
>>> from PDFStorage.models import PDFs
>>> x = PDFs.objects.all()[2]
>>> x.PDFname
'52 Programming Problems and Solutions'
>>> x.Price = 335
>>> x.save()
```

29. Now let's view the price



```
>>> from PDFStorage.models import PDFs
>>> x = PDFs.objects.all()[2]
>>> x.PDFname
'52 Programming Problems and Solutions'
>>> x.Price = 335
>>> x.save()
>>> x.Price
335
>>>
```

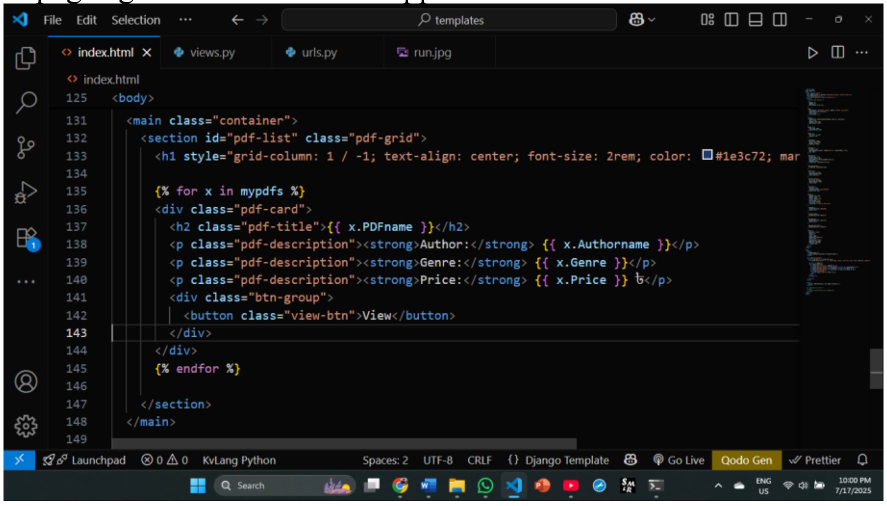
Now Let's display the whole database.



```
335
>>> PDFs.objects.all().values()
<QuerySet [{'id': 1, 'PDFname': 'Jochona O Jononir Golpo', 'Authorname': 'Humayun Ahme
d', 'Genre': 'Fiction', 'Price': 0}, {'id': 2, 'PDFname': 'Machine Learning Algorithm'
, 'Authorname': 'Nafis Nihal', 'Genre': 'Programming', 'Price': 0}, {'id': 3, 'PDFname
': '52 Programming Problems and Solutions', 'Authorname': 'Tamim Shahriar Subin', 'Gen
re': 'Programming', 'Price': 335}, {'id': 4, 'PDFname': 'Data Communications & Network
ing', 'Authorname': 'Tahmid Rafi', 'Genre': 'Networking', 'Price': 0}, {'id': 5, 'PDFn
ame': 'ShreemadVagbadGeeta Jothajotho', 'Authorname': 'ISSKAN', 'Genre': 'Religious',
'Price': 0}]>
>>>
```

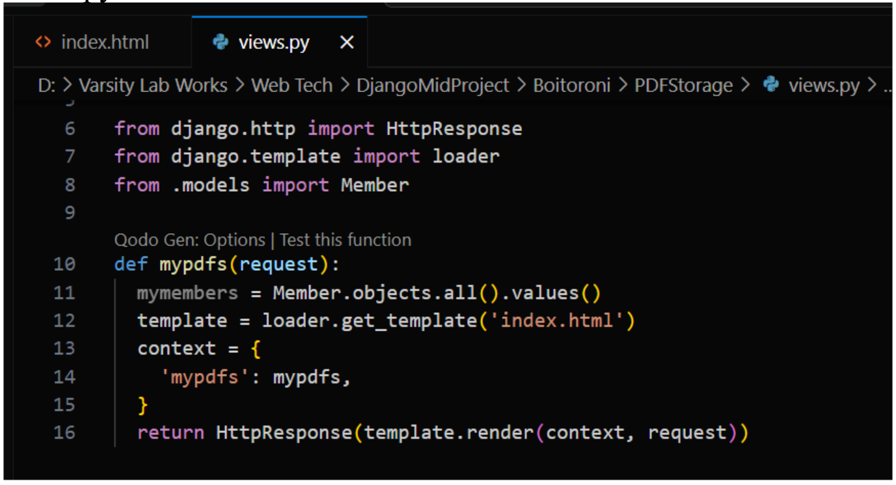
Since we set the default price = 0 and we set 335 as price value on “52 Programming Problems and Solutions” book, all the prices are 0 here except that book.

30. So, we have successfully constructed a database and manipulated the data. Now let’s make it displayable for the users in our Boitoroni Webpage. Now lets get back to index.html page again and add some snippet there.



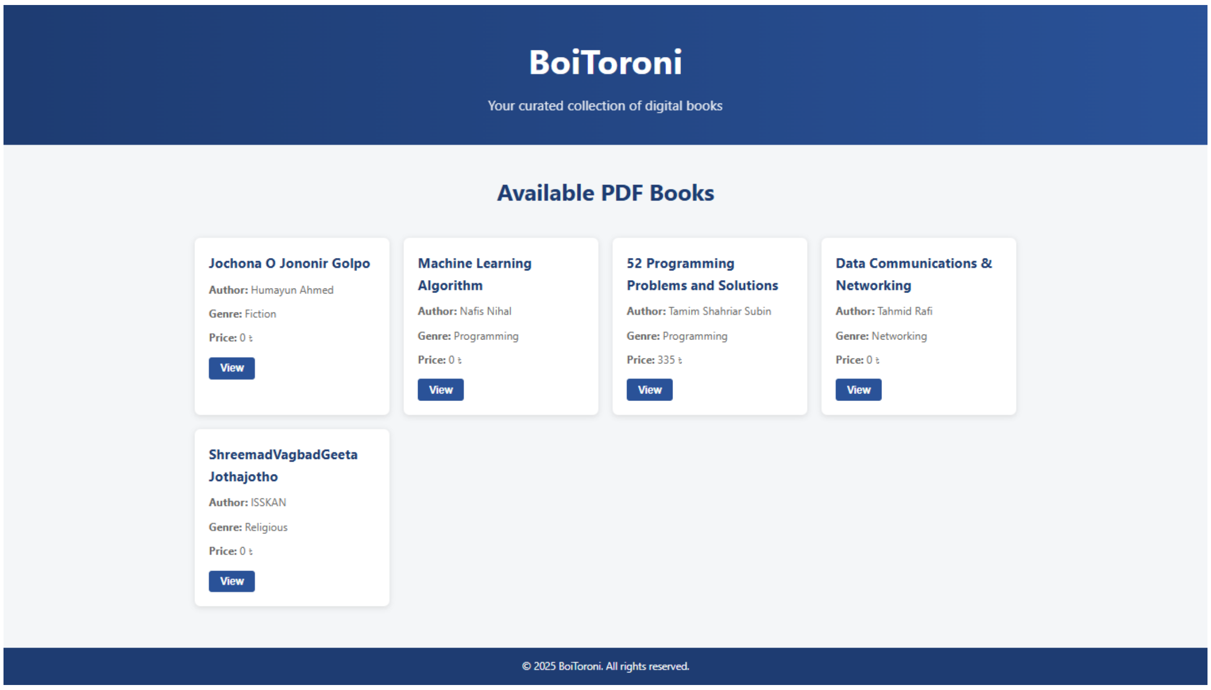
```html
<body>
  <main class="container">
    <section id="pdf-list" class="pdf-grid">
      <h1 style="grid-column: 1 / -1; text-align: center; font-size: 2rem; color: #1e3c72; mar

      {% for x in mypdfs %}
      <div class="pdf-card">
        <h2 class="pdf-title">{{ x.PDFname }}</h2>
        <p class="pdf-description"><strong>Author:</strong> {{ x.Authorname }}</p>
        <p class="pdf-description"><strong>Genre:</strong> {{ x.Genre }}</p>
        <p class="pdf-description"><strong>Price:</strong> {{ x.Price }} ৳</p>
        <div class="btn-group">
          <button class="view-btn">View</button>
        </div>
      </div>
      {% endfor %}

    </section>
  </main>
```

Then in views.py file we edited the code.



```python
from django.http import HttpResponse
from django.template import loader
from .models import Member

def mypdfs(request):
    mymembers = Member.objects.all().values()
    template = loader.get_template('index.html')
    context = {
      'mypdfs': mypdfs,
    }
    return HttpResponse(template.render(context, request))
```

Now again running the command: python manage.py runserver we get the following result. Visiting this link: http://127.0.0.1:8000/PDFStorage/


BoiToroni

Your curated collection of digital books

Available PDF Books

Title	Author	Genre	Price
Jochona O Jononir Golpo	Humayun Ahmed	Fiction	0 ৳
Machine Learning Algorithm	Nafis Nihal	Programming	0 ৳
52 Programming Problems and Solutions	Tamim Shahriar Subin	Programming	335 ৳
Data Communications & Networking	Tahmid Rafi	Networking	0 ৳
ShreemadVagbadGeeta Jothajotho	ISSKAN	Religious	0 ৳

© 2025 BoiToroni. All rights reserved.

Conclusion: Through this project, we have successfully explored the foundational aspects of Django development by building a basic web application titled **Boitoroni**. From setting up the development environment and creating the Django project to designing views, templates, and implementing models for PDF management, each step has demonstrated Django's robust capabilities in streamlining web development.

This hands-on implementation offered practical exposure to data handling, template rendering, and URL routing, while also illustrating how CRUD operations can be performed efficiently using Django's ORM. Although the project currently supports static PDF display and basic database manipulation, it lays a strong groundwork for further development, including dynamic features like user authentication, search functionality, and file uploads.

In conclusion, this exercise not only solidified our understanding of Django's core components but also emphasized the framework's scalability and its ability to power real-world web applications with minimal boilerplate.