GOKUL.P
CSE,3RDYEAR
SVCE.

URL SHORTNER
PYTHON PROJECT

Introduction

A URL shortener is a tool that takes a long URL and turns it into a short one that redirects to the intended page. URL shortener proves to be useful in many cases, such as requiring the user to only type small numbers of characters, as long URLs are difficult to memorize.

Pre-requisites

Setting Up the Django Project

In this step, you will be installing all the necessary tools for the application and setting up your Django project.

Once you have created your project directory and started your virtual environment, as covered in the prerequisites, install the necessary packages using `pip`, the Python package manager. This tutorial will install Django version 2.1.7 and Graphene-Django version 2.2.0 or higher:

```
pip intall django
```

You now have all the tools needed in your tool belt. Next, you will create a Django project using the `django-admin` command. A project is the default Django boilerplate—a set of folders and files with everything necessary to start the development of a web application. In this case, you will call your project and create it inside your current folder by specifying the `.` at the end:

```
django-admin startapp url
```

After creating your project, you will run the Django migrations. These files contain Python code generated by Django and are responsible for changing the application's structure according to the Django models. Changes might include the creation of a table, for example. By default, Django comes with its own set of migrations responsible for subsystems like Django Authentication, so it is necessary to execute them with the following command:

```
python manage.py migrate
```

This command uses the Python interpreter to invoke a Django script called `manage.py`, responsible for managing different aspects of your project, like creating apps or running migrations.

Once Django's database is ready to go, start its local development server:

```
python manage.py runserver
```

This command will take away the prompt in your terminal and start the server.

To stop the server and return to your terminal, press `CTRL+C`. Whenever you need to access the browser, make sure the preceding command is running.

STARTING WITH OUR PROJECT

First of all, We need to create our project by,

```
django-admin startproject urlshortner

cd urlshortner
```

Above Command, Creates A Django Project and then cd into that directory. After that, We also need to create an app inside of our project. App is sort of a container, where we will store our code. A project can have Multiple Apps and they can be interconnected

```
python manage.py startapp url
```

Above Command Creates an App named URL in our Project. Our File Structure Now will be —

```
urlShort

├── manage.py

├── url

│ ├── admin.py
│ ├── apps.py
│ ├── __init__.py
│ ├── migrations
│ │ └── __init__.py
│ ├── models.py
│ ├── tests.py
│ └── views.py
└── urlShort
 ├── asgi.py
 ├── __init__.py
 ├── __pycache__
 │ ├── __init__.cpython-37.pyc
 │ └── settings.cpython-37.pyc
 ├── settings.py
 ├── urls.py
 └── wsgi.py
```

Checking If all is Great…

You can check if all is working by just typing this in Command Line. But cd into the main folder, here urlShort.

```
python manage.py runserver
```

runserver will run a local server where our website will load. Move to url

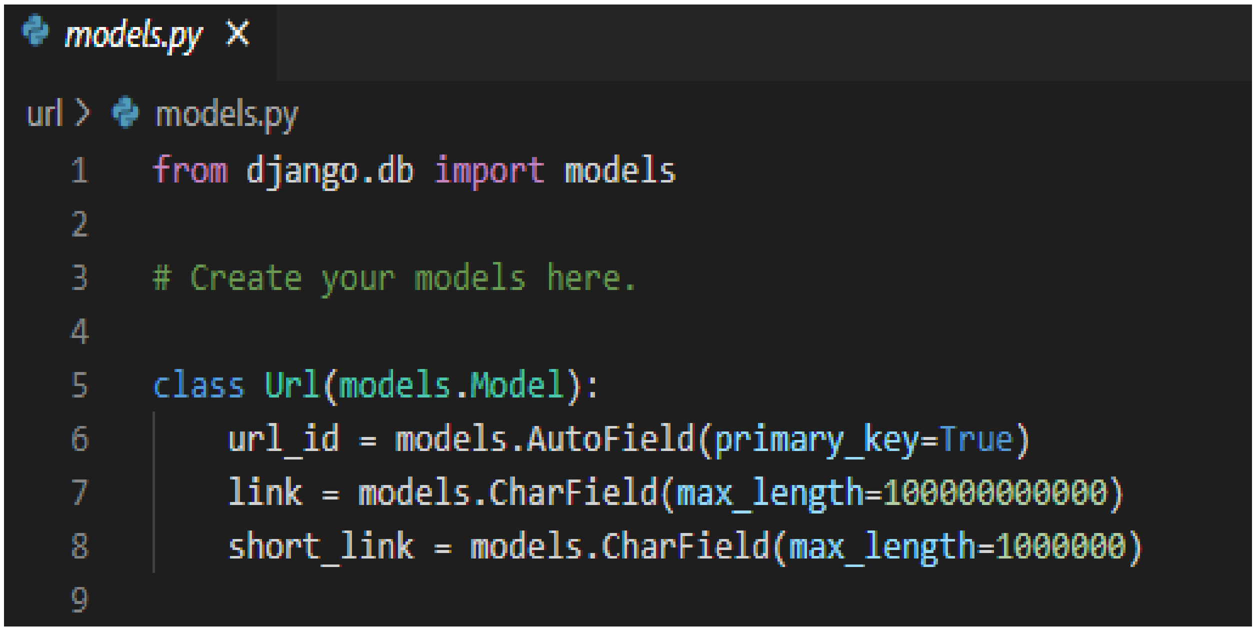
```
https://localhost:8000
```

Keep Your Console Window Open.

How to Build a URL Shortener Project?

Creating Django Models –

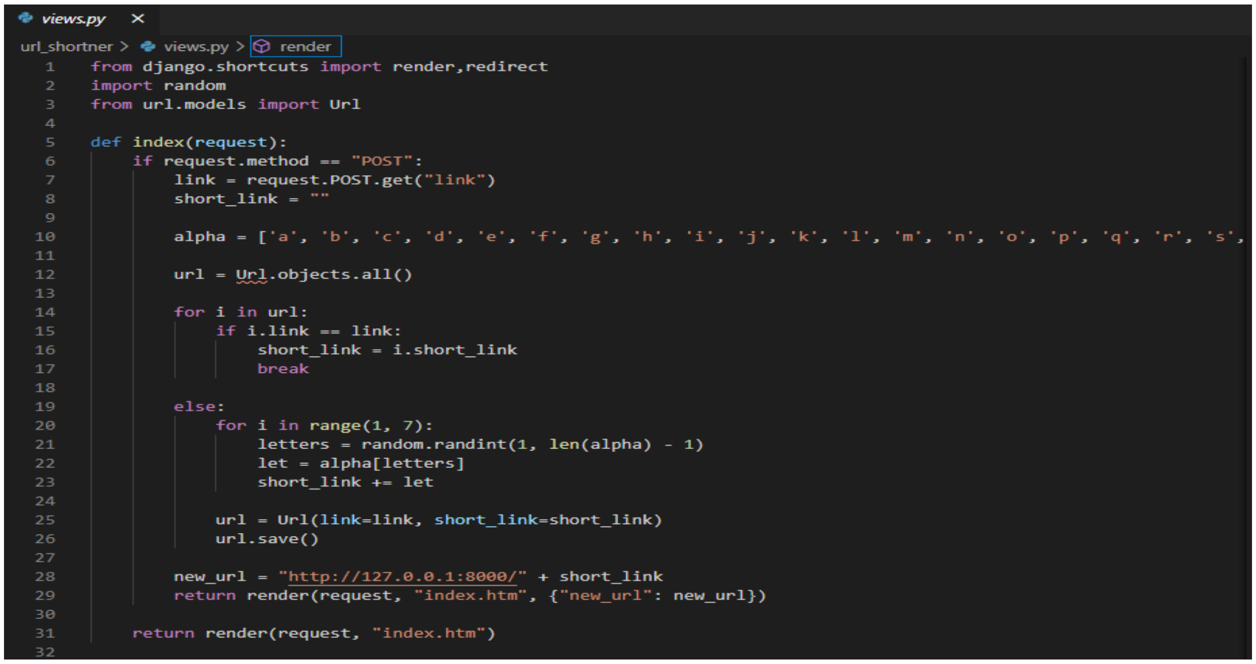
First of all, we need a Database to store our Shorten URL's. For That, We need to create a Schema for our Database Table in models.py.



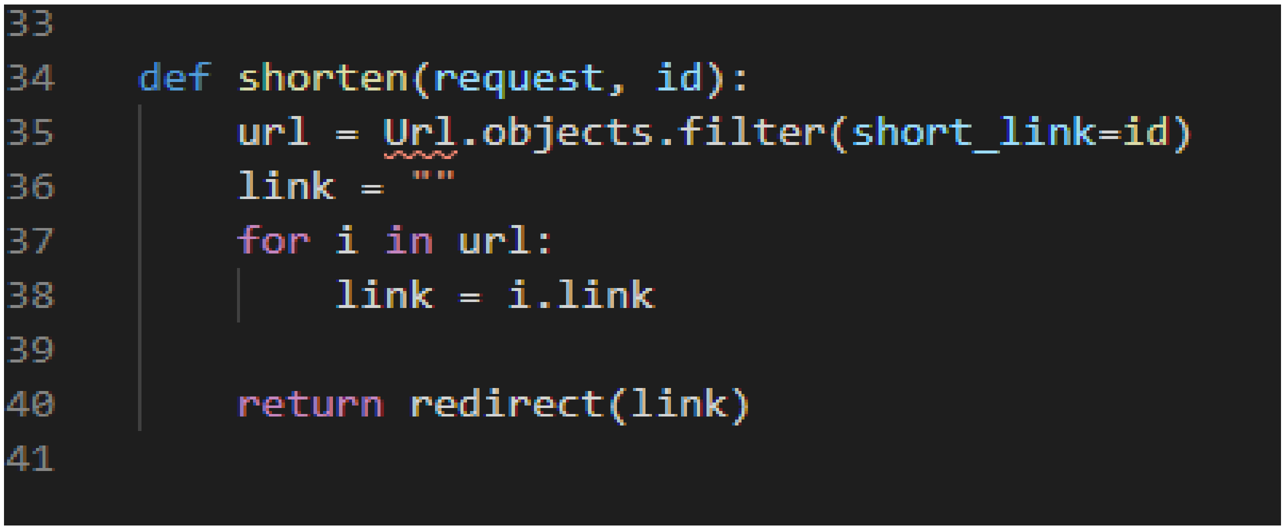
```python
from django.db import models

# Create your models here.

class Url(models.Model):
    url_id = models.AutoField(primary_key=True)
    link = models.CharField(max_length=100000000000)
    short_link = models.CharField(max_length=1000000)
```

Creating Views

Now, We will create the Interface of our App using *views.py*. Let's divide this part in Functions.
index()— This Function is where our *Main Algorithm* works. It takes a *url* from form after User submits it, then it generates a Random Slug which is then stored in Database with Original Url. It is also the function which render *index.html* (entrypoint of our app).



```python
from django.shortcuts import render,redirect
import random
from url.models import Url

def index(request):
    if request.method == "POST":
        link = request.POST.get("link")
        short_link = ""

        alpha = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's',

        url = Url.objects.all()

        for i in url:
            if i.link == link:
                short_link = i.short_link
                break

        else:
            for i in range(1, 7):
                letters = random.randint(1, len(alpha) - 1)
                let = alpha[letters]
                short_link += let

            url = Url(link=link, short_link=short_link)
            url.save()

        new_url = "http://127.0.0.1:8000/" + short_link
        return render(request, "index.htm", {"new_url": new_url})

    return render(request, "index.htm")
```

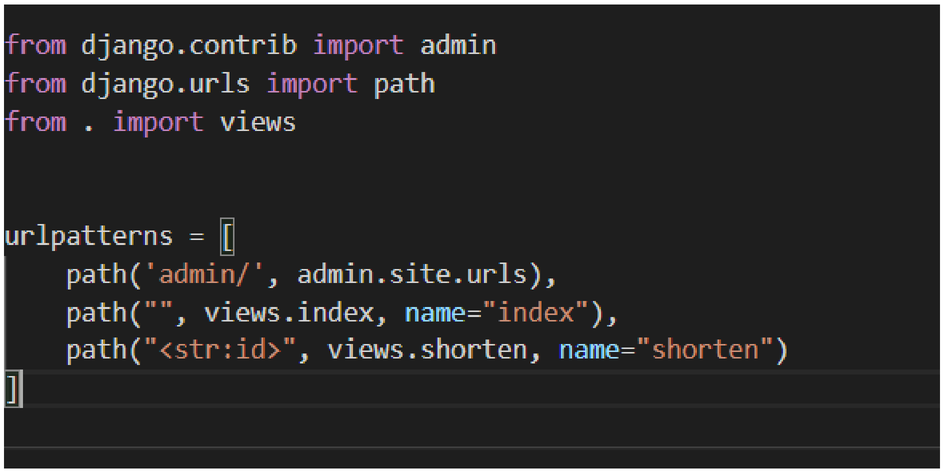


```
def shorten(request, id):
    url = Url.objects.filter(short_link=id)
    link = ""
    for i in url:
        link = i.link

    return redirect(link)
```

urlRedirect() — This Function tracks the slug to Original URL and redirects it to Original URL.

Creating Routes

Before Running This App, We need to specify URL paths in App's *urls.py*



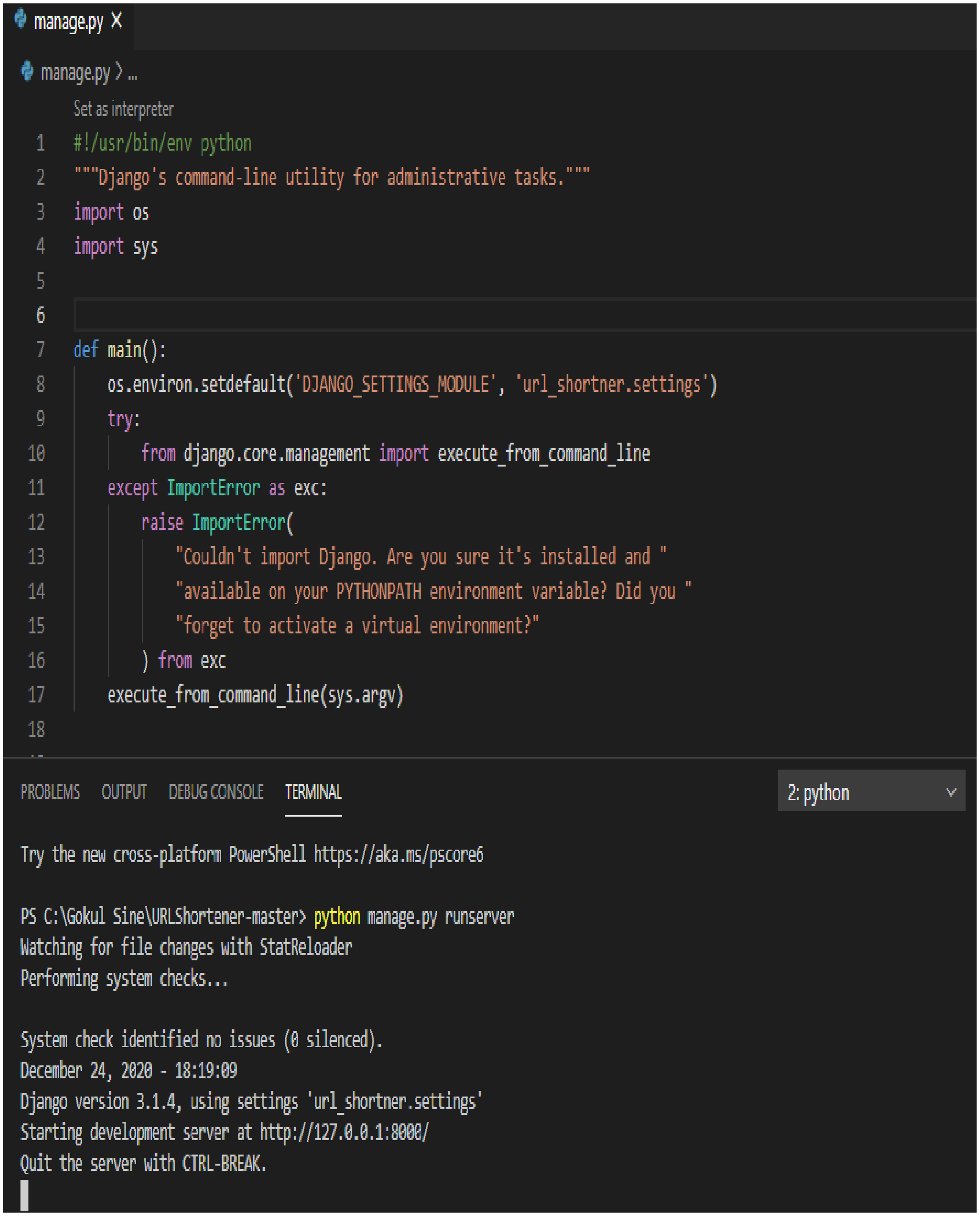
```python
from django.contrib import admin
from django.urls import path
from . import views


urlpatterns = [
    path('admin/', admin.site.urls),
    path("", views.index, name="index"),
    path("<str:id>", views.shorten, name="shorten")
]
```

Output

Run the Project

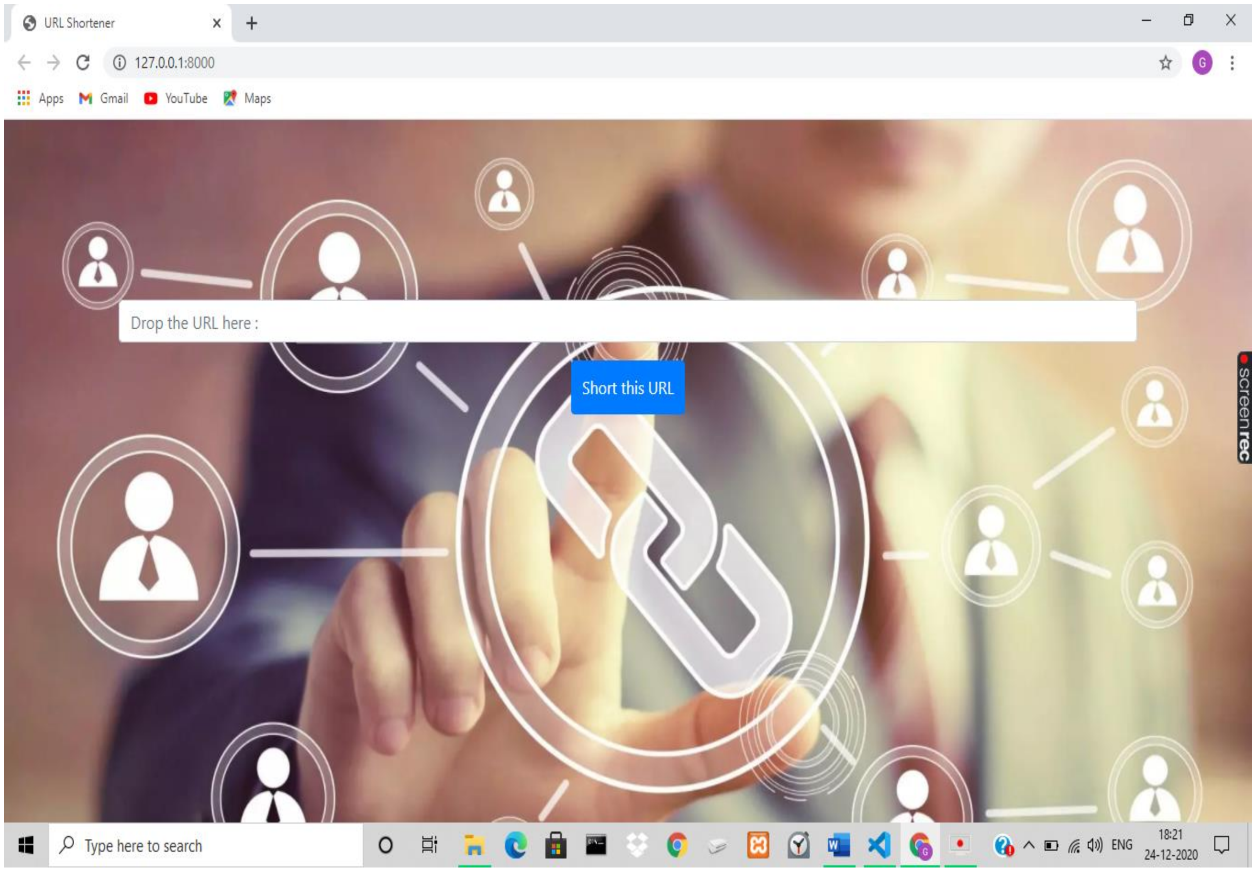
Open Console in Main Project Directory.

python manage.py runserver



```
#!/usr/bin/env python
"""Django's command-line utility for administrative tasks."""
import os
import sys


def main():
    os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'url_shortner.settings')
    try:
        from django.core.management import execute_from_command_line
    except ImportError as exc:
        raise ImportError(
            "Couldn't import Django. Are you sure it's installed and "
            "available on your PYTHONPATH environment variable? Did you "
            "forget to activate a virtual environment?"
        ) from exc
    execute_from_command_line(sys.argv)

```

```
Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Gokul Sine\URLShortener-master> python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
December 24, 2020 - 18:19:09
Django version 3.1.4, using settings 'url_shortner.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```



Conclusion

Thus, the URL Shortener is executed successfully .