

Dr. AMBEDKAR INSTITUTE OF TECHNOLOGY
(An Autonomous Institution aided by Government of Karnataka)
Near Jnana Bharathi Campus, Bangalore – 560056

SCHOOL OF COMPUTING
INFORMATION SCIENCE & ENGINEERING PROGRAMME



LAB MANUAL
Full Stack Development
Sub code: 22ISU601
EVEN SEMESTER 2025-2026
BE - ISE – VI Sem

Co-ordinator

Smt P. Malathi
Asst Professor,
ISE Programme, Dr. AIT.

Head of the Programme

Dr Prabha R
Professor & HOP,
ISE Programme, Dr. AIT.

LIST OF PROGRAMS

1. Installation of Python, Django and Visual Studio code editors to be demonstrated.
2. Creation of virtual environment, Django project and App should be demonstrated.
3. Django app that displays tables of squares of pairs of numbers input in the URL.
4. Develop a Django app that displays number of vowels and consonants and also list of vowels and consonants for any input sentence specified in the URL.
5. Develop a Django Project to check whether a given number is prime or not.
6. Develop a Django app that performs student registration to a course. It should also display list of students registered for any selected course. Create students and course as models with enrolment as ManyToMany field.
7. For student and course models created in Lab experiment for Module2, register admin interfaces, perform migrations and illustrate data entry through admin forms.
8. For students enrolment developed in Module 2, create a generic class view which displays list of students and detail view that displays student details for any selected student in the list.
9. Develop example Django app that performs CSV and PDF generation for any models created in previous laboratory component
10. Develop a registration page for student enrolment as done in Module 2 but without page refresh using AJAX
11. Develop a search application in Django using AJAX that displays courses enrolled by a student being searched.

Experiment-01

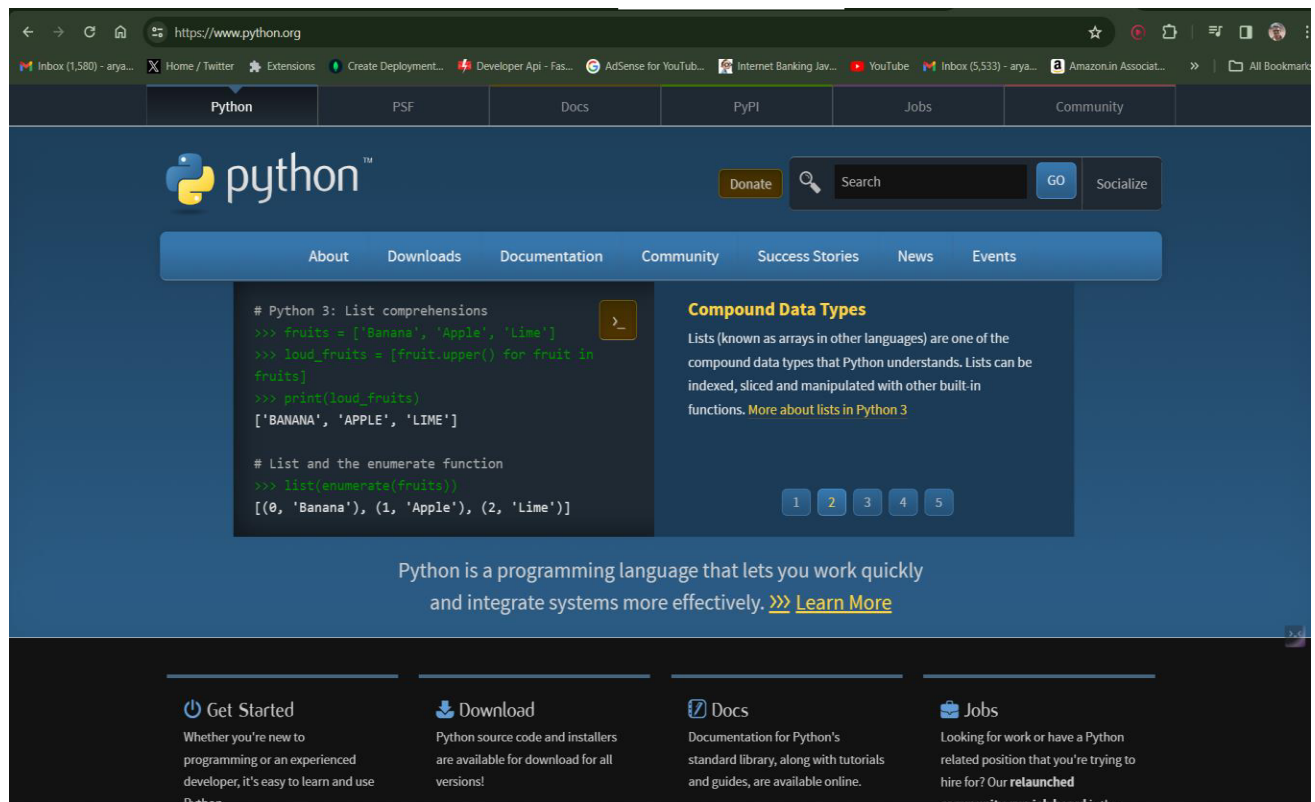
Installation of Python, Django and Visual Studio code editors can be demonstrated.

Python

- Python is a popular high-level, general-use programming language.
- Python is a programming language that enables rapid development as well as more effective system integration.
- Python has two main different versions: Python 2 and Python 3. Both are really different.

Here are the steps you can follow to download Python: **Steps to Download & Install Python**

Visit the link <https://www.python.org> to download the latest release or [python](#).



Step - 1: Select the Python's version to download.

Click on the download button to download the exe file of Python.

The image shows the Python.org homepage. At the top, there's a navigation bar with links to Python, PSF, Docs, PyPI, Jobs, and Community. Below this is a search bar and a 'Donate' button. The main heading is 'Download the latest version for Windows' with a 'Download Python 3.12.2' button. Below this, there's text about looking for Python on different OSes and links to prereleases and Docker images. A large illustration of two parachutes carrying boxes is on the right. Below the main heading, there's a section titled 'Active Python Releases' with a link to the Python Developer's Guide. A table lists the active releases:

Python version	Maintenance status	First released	End of support	Release schedule
3.13	prerelease	2024-10-01 (planned)	2029-10	PEP 719
3.12	bugfix	2023-10-02	2028-10	PEP 693
3.11	bugfix	2022-10-24	2027-10	PEP 664
3.10	security	2021-10-04	2026-10	PEP 610

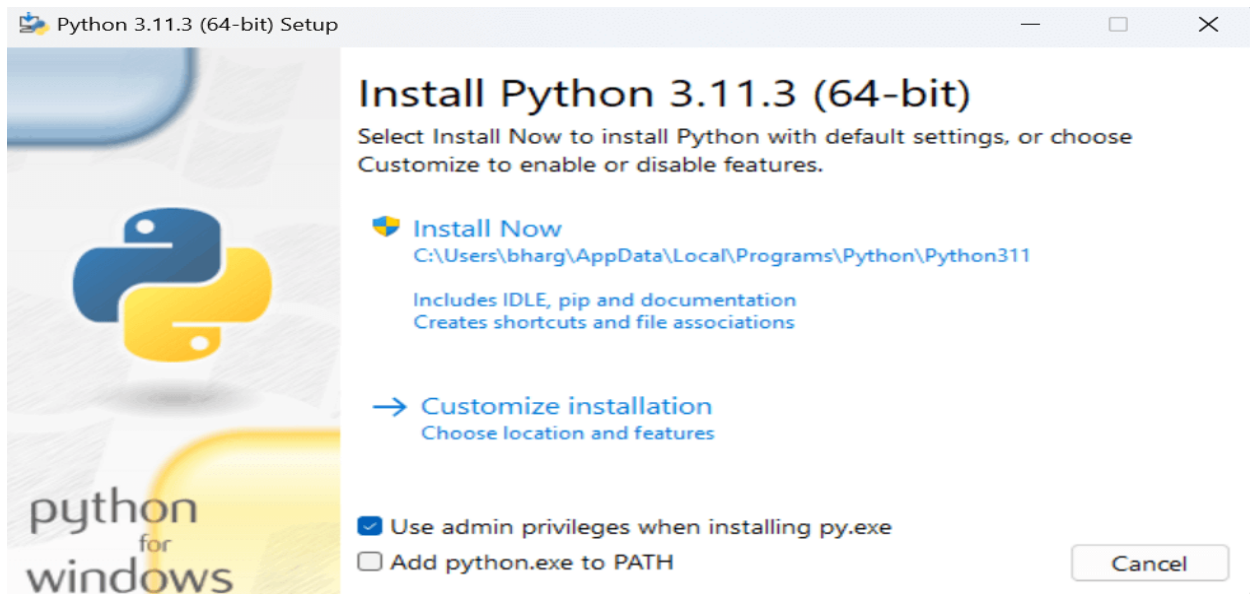
If in case you want to download the specific version of Python. Then, you can scroll down further below to see different versions from 2 and 3 respectively. Click on download button right next to the version number you want to download.

The image shows the 'Looking for a specific release?' section on Python.org. It has a heading 'Looking for a specific release?' and a subheading 'Python releases by version number:'. Below this is a table with columns 'Release version', 'Release date', and 'Click for more'. The table lists several releases with 'Download' and 'Release Notes' links. Below the table is a link to 'View older releases'. At the bottom, there's a 'Sponsors' section with logos for Google, Meta, fastly, and Bloomberg Engineering. The footer contains links for 'Licenses', 'Sources', 'Alternative', and 'History'.

Release version	Release date	Click for more
Python 3.11.9	April 2, 2024	Download Release Notes
Python 3.10.14	March 19, 2024	Download Release Notes
Python 3.9.19	March 19, 2024	Download Release Notes
Python 3.8.19	March 19, 2024	Download Release Notes
Python 3.11.8	Feb. 6, 2024	Download Release Notes
Python 3.12.2	Feb. 6, 2024	Download Release Notes
Python 3.12.1	Dec. 8, 2023	Download Release Notes
Python 3.11.8	Dec. 8, 2023	Download Release Notes

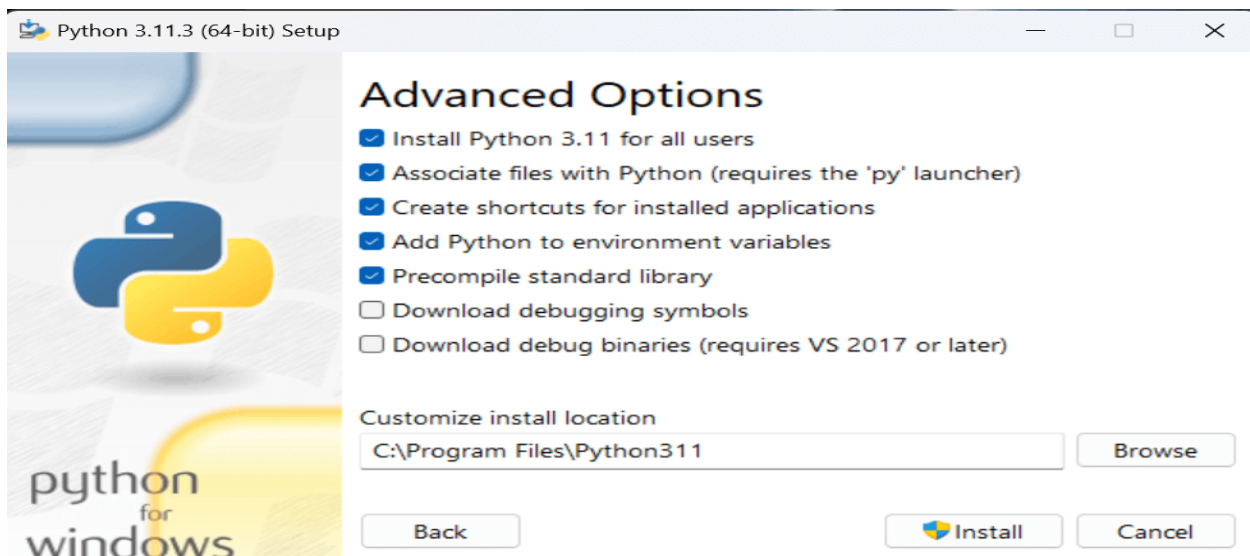
Step - 2: Click on the Install Now

Double-click the executable file, which is downloaded.

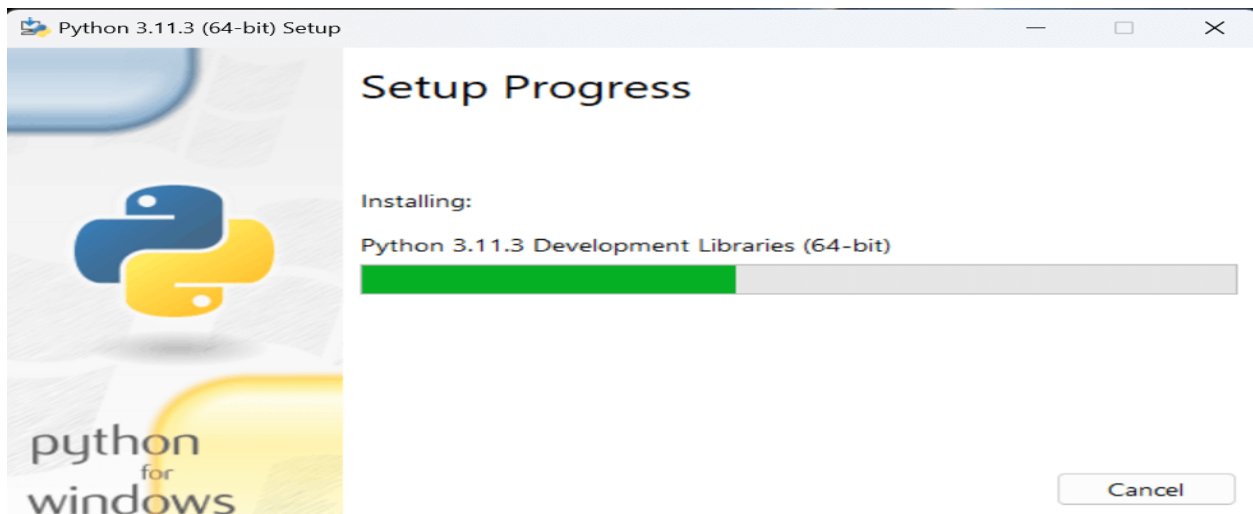


The following window will open. Click on the Add Path check box, it will set the Python path automatically.

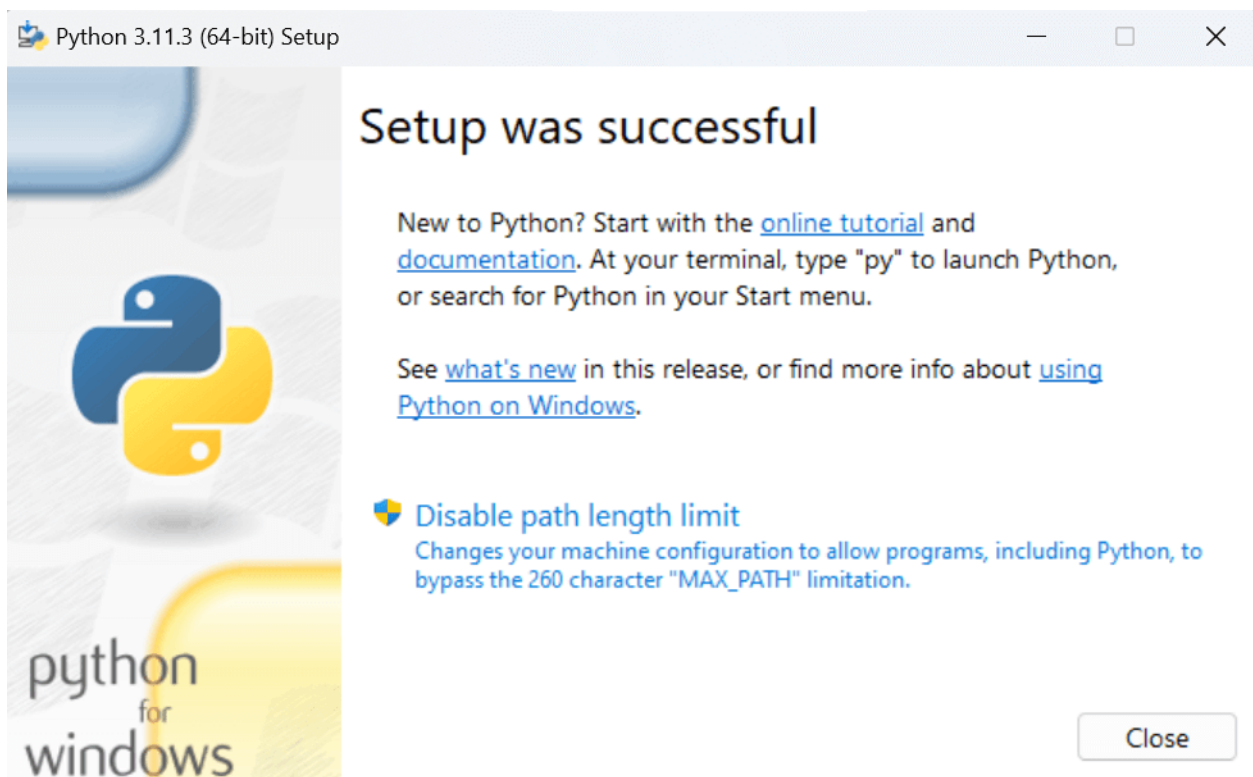
Now, Select Customize installation and proceed. We can also click on the customize installation to choose desired location and features. Other important thing is installing launcher for the all user must be checked.



Step - 3 Installation in Process



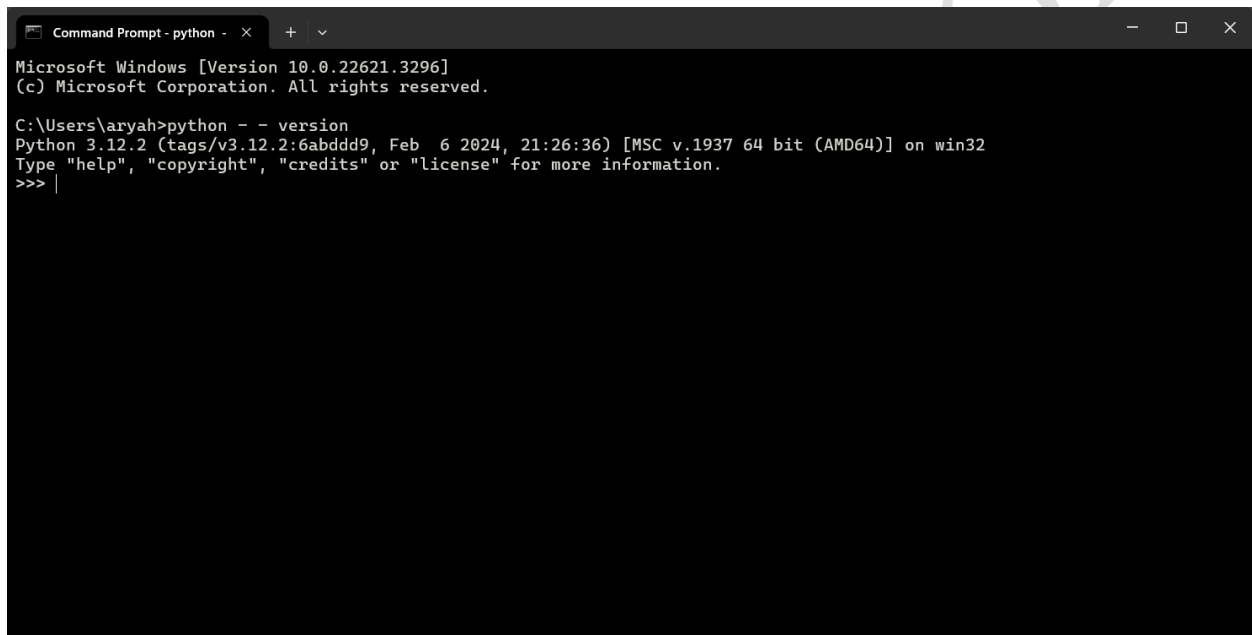
The set up is in progress. All the python libraries, packages, and other python default files will be installed in our system. Once the installation is successful, the following page will appear saying " Setup was successful ".



Step - 4: Verifying the Python Installation

To verify whether the python is installed or not in our system, we have to do the following.

- Go to "Start" button, and search " cmd ".
- Then type, " python - - version ".
- If python is successfully installed, then we can see the version of the python installed.
- If not installed, then it will print the error as " 'python' is not recognized as an internal or external command, operable program or batch file. ".

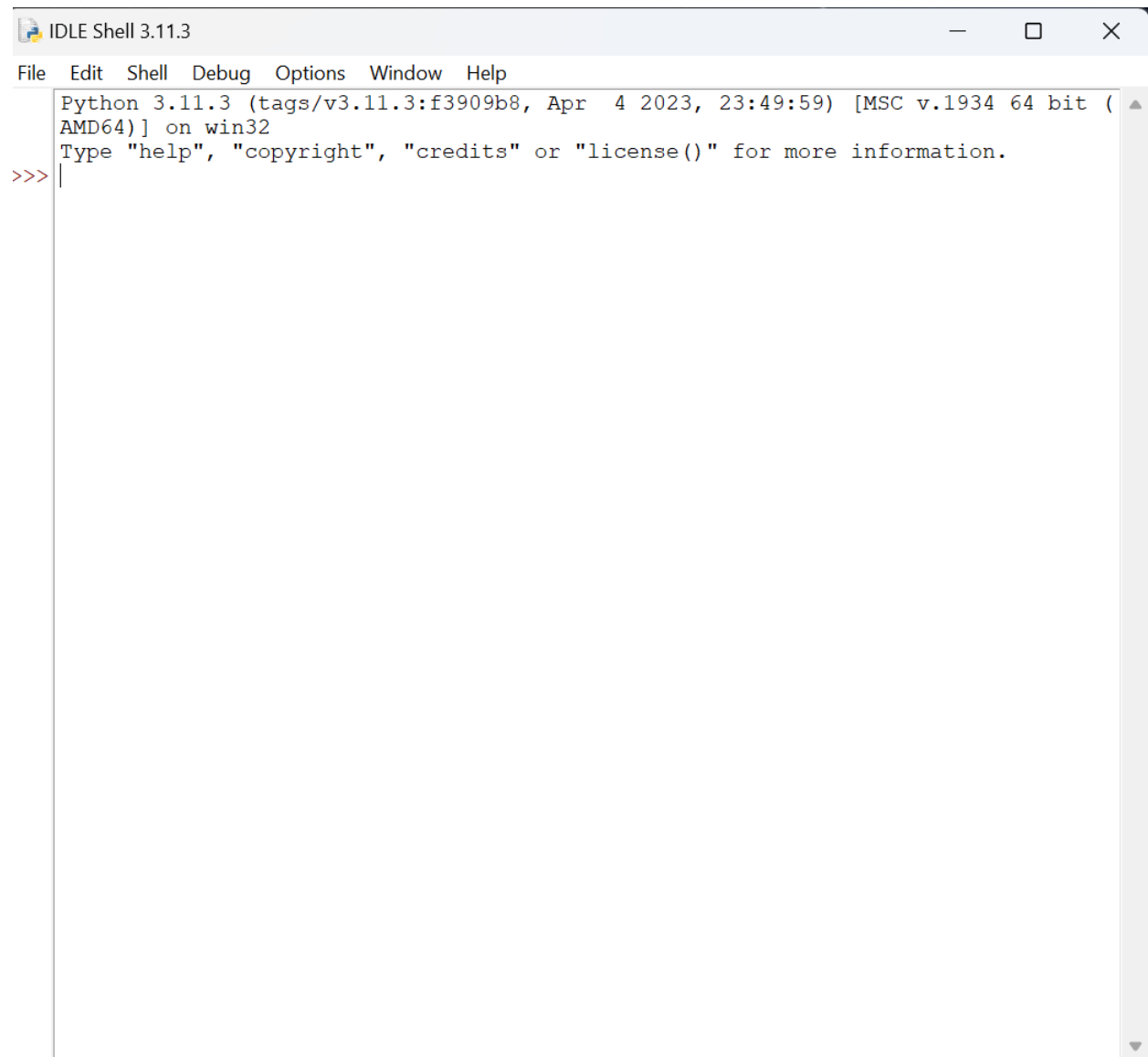


```
Command Prompt - python - x + v
Microsoft Windows [Version 10.0.22621.3296]
(c) Microsoft Corporation. All rights reserved.

C:\Users\aryah>python --version
Python 3.12.2 (tags/v3.12.2:6abddd9, Feb  6 2024, 21:26:36) [MSC v.1937 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> |
```

We are ready to work with the Python.

Step - 5: Opening idle

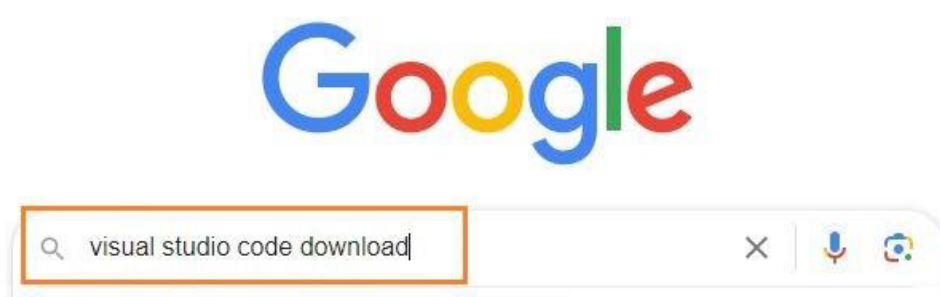


The screenshot shows the IDLE Shell 3.11.3 window. The title bar reads "IDLE Shell 3.11.3". The menu bar includes "File", "Edit", "Shell", "Debug", "Options", "Window", and "Help". The main text area displays the following information: "Python 3.11.3 (tags/v3.11.3:f3909b8, Apr 4 2023, 23:49:59) [MSC v.1934 64 bit (AMD64)] on win32". Below this, it says "Type 'help', 'copyright', 'credits' or 'license()' for more information." At the bottom left of the text area, the prompt ">>>" is followed by a vertical cursor line.

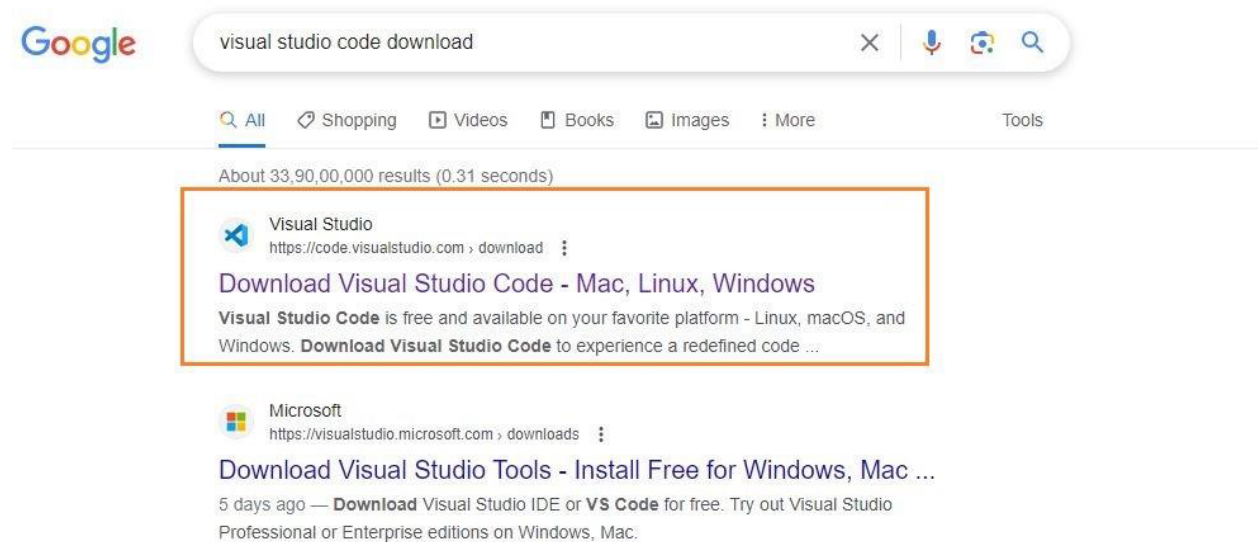
Now, to work on our first python program, we will go the interactive interpreter prompt(idle). To open this, go to "Start" and type idle. Then, click on open to start working on idle.

Here are the steps you can follow: Steps to Download & Install VS Code

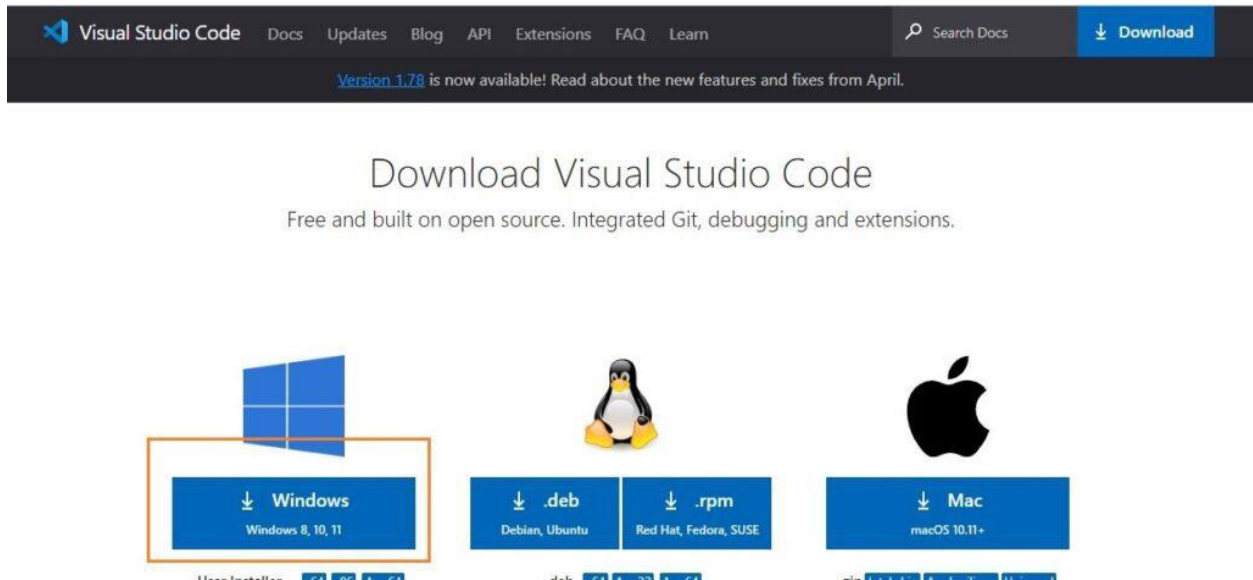
Step – 1: Open Google and type Visual Studio Code download in the search bar.



Step – 2: Click on the below highlighted link for any OS.

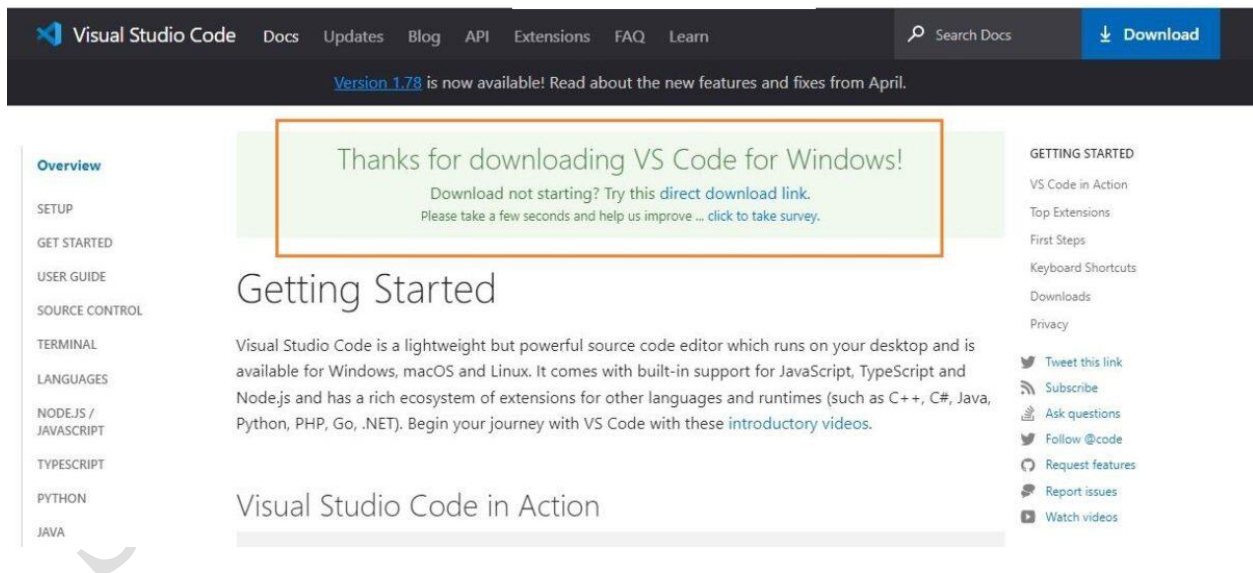


Step – 3: Now, select the respective OS. In this case we are selecting Windows.

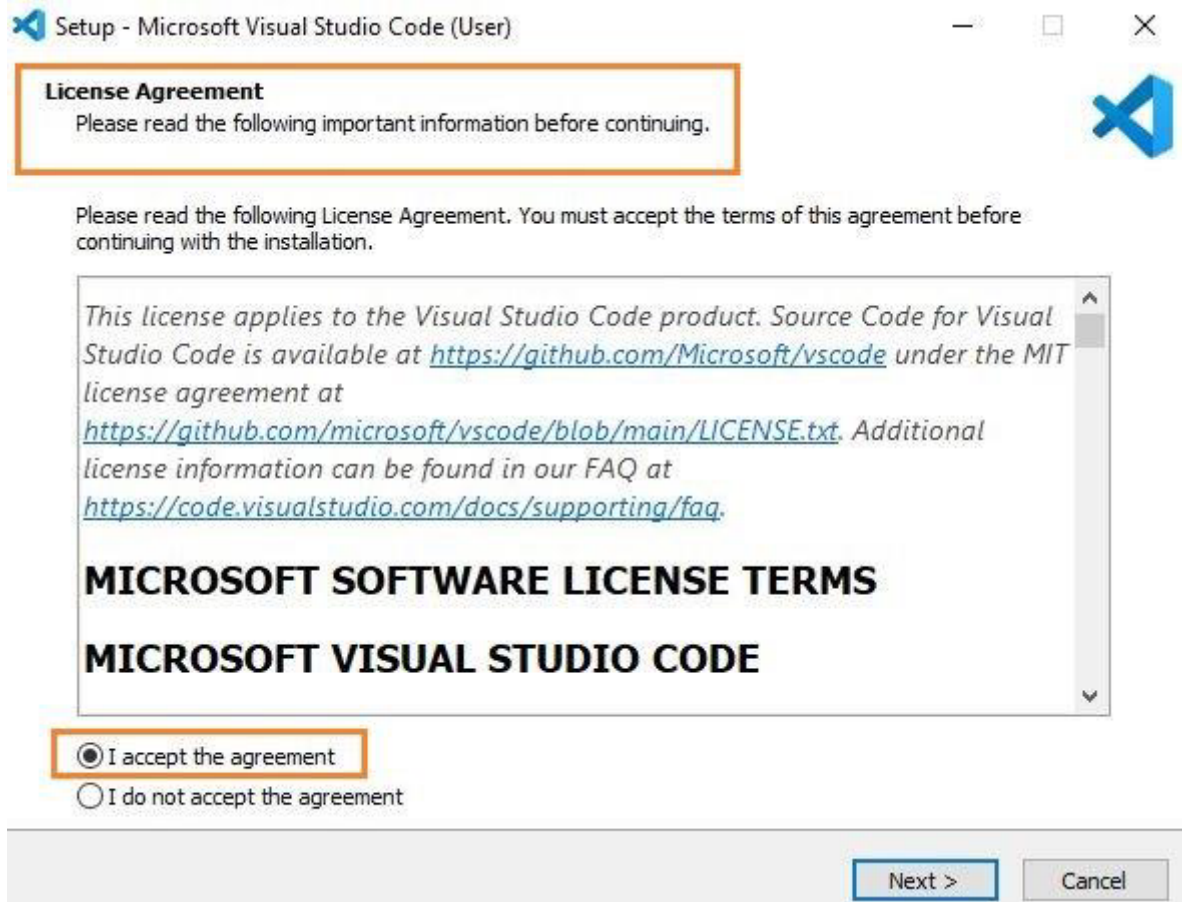


Step – 4: The file will be downloaded onto your system. Open the file and then click on Install.

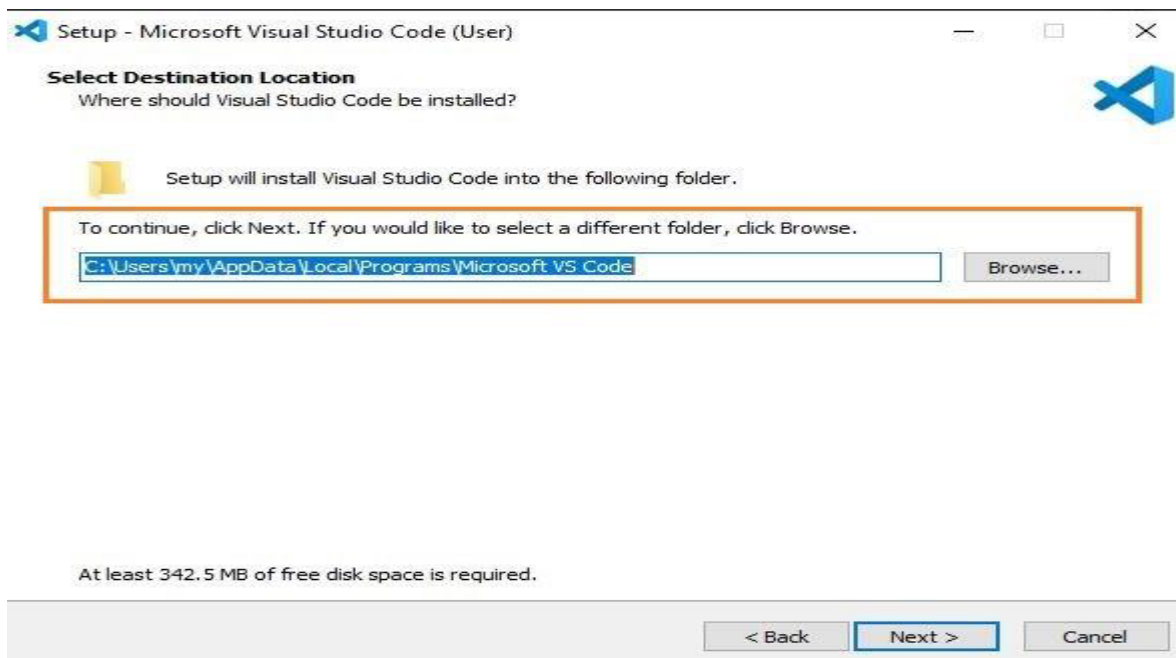
After downloading the VS Code file, the official site will display a Thanks message for downloading the file.



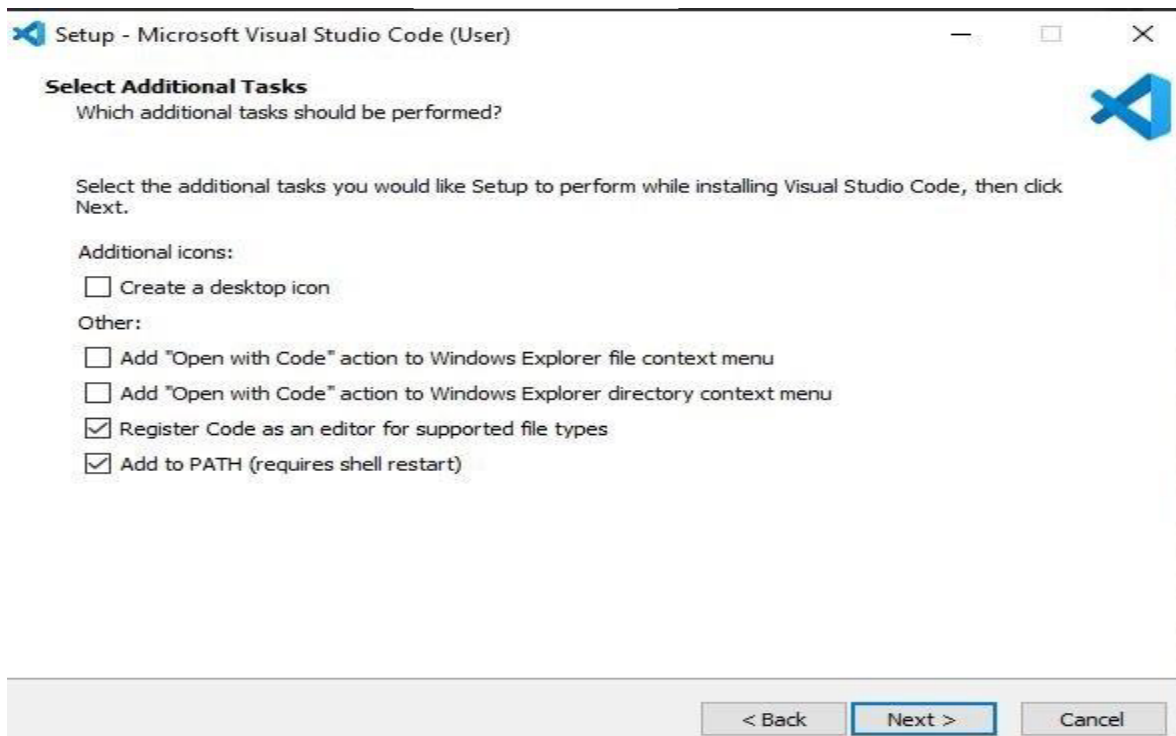
Step – 5: Now accept the license agreement.



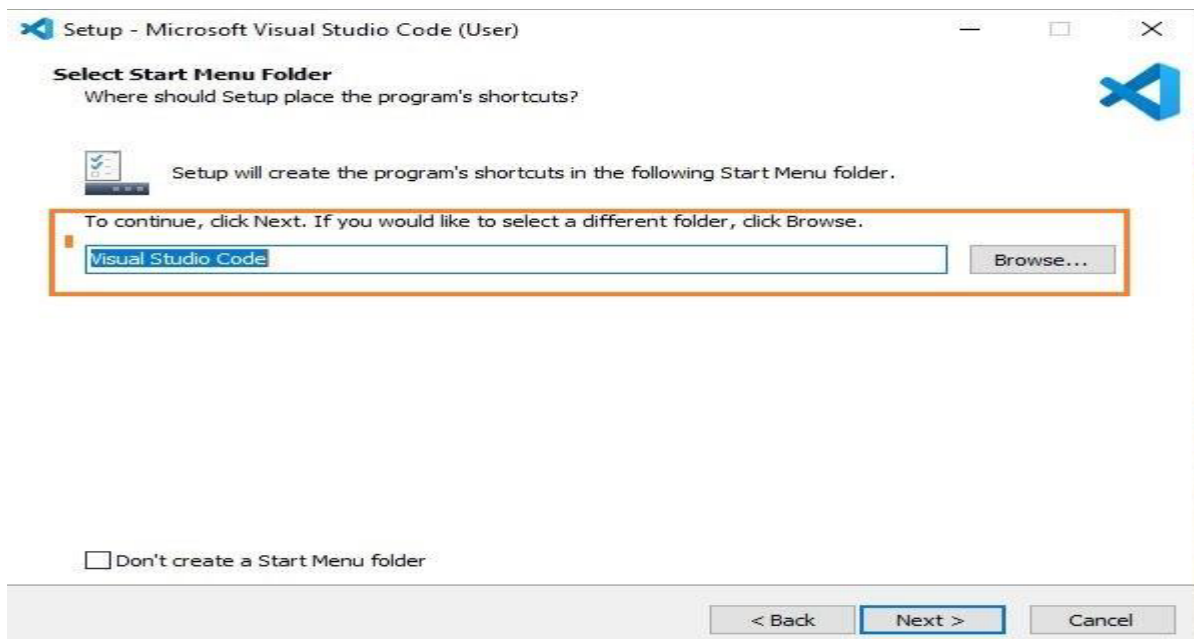
Step – 6: Then it prompts for the file location, where you want to save the VS Code file.
Browse the location and then click on Next.



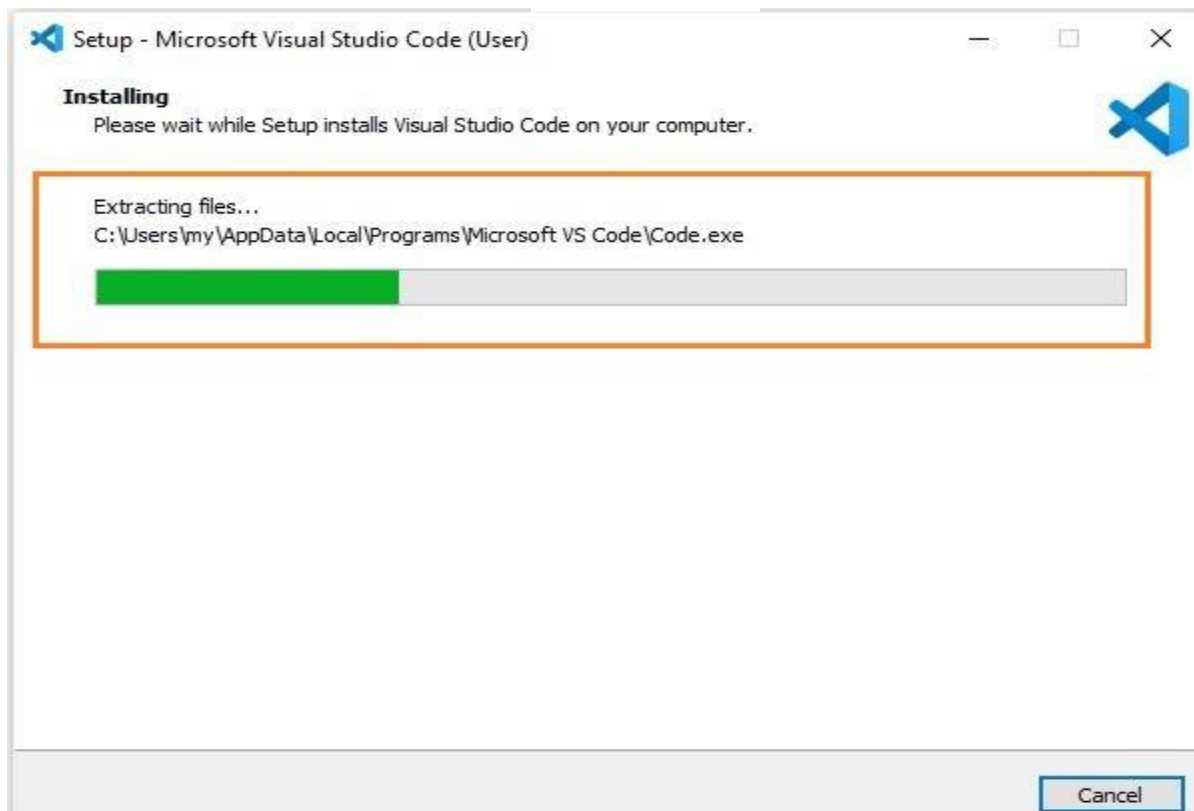
Step – 7: Next, you see the prompt for the additional task which we want the VS Code to perform. At this step, choose the default settings and then click on next.



Step – 8: The next prompt is how you want the VS Code on your startup. Change according to your convenience and click on Next.



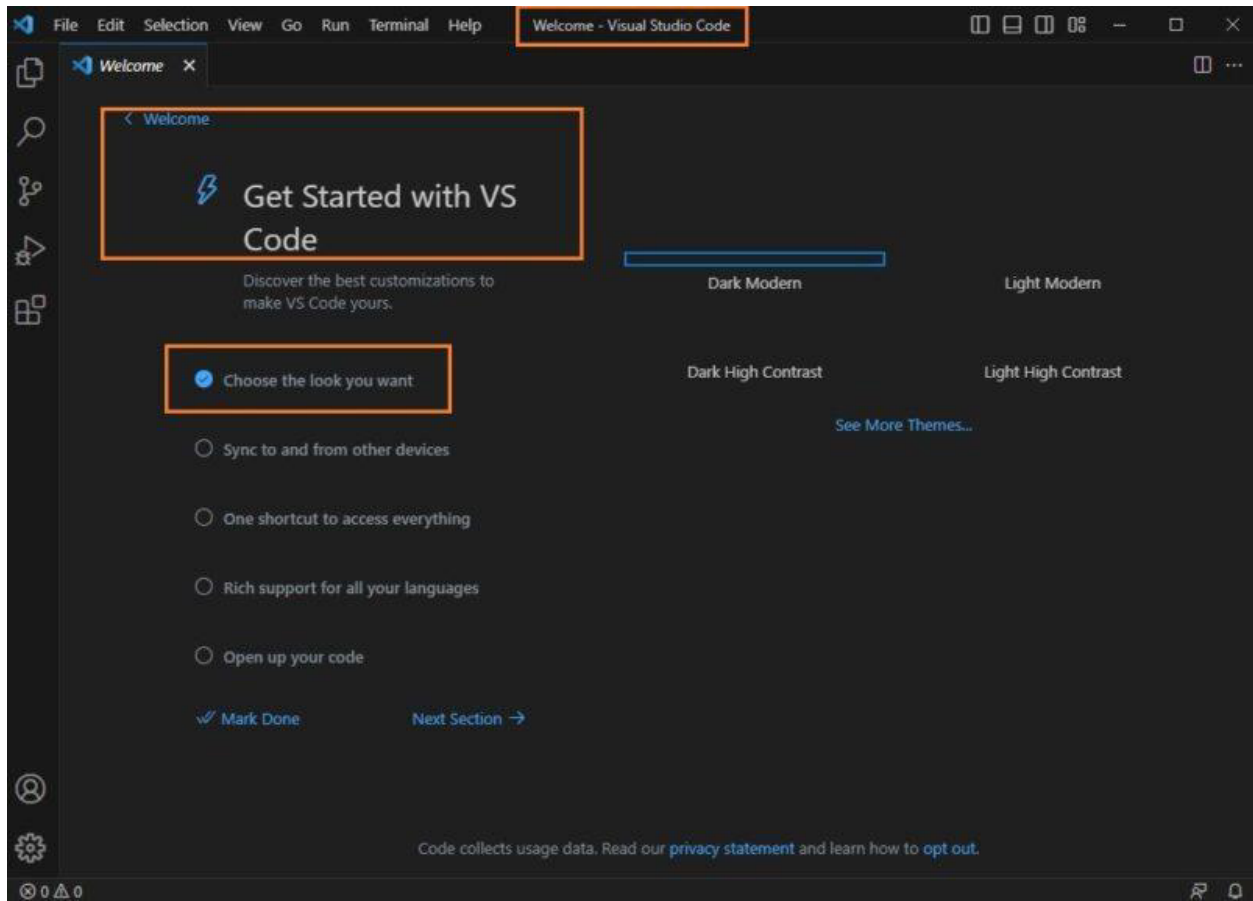
Step – 9: The installation of VS Code will now begin.



Step – 10: At this step, we have completed installing VS Code, click on Finish.



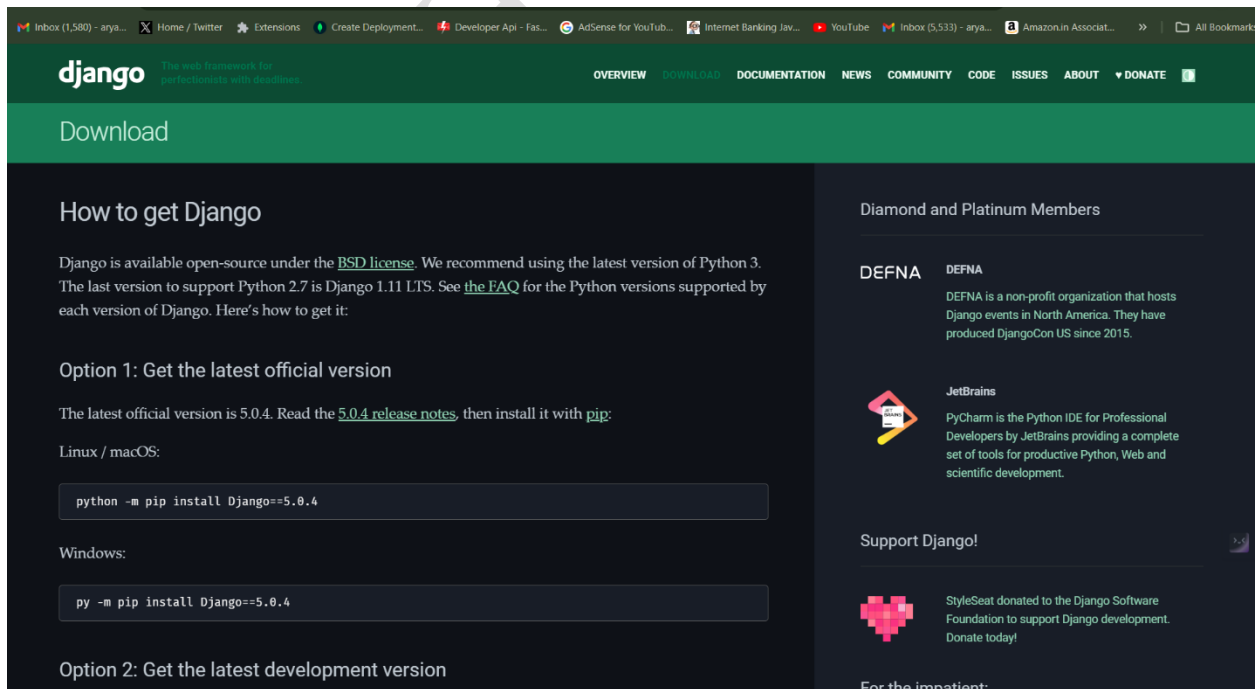
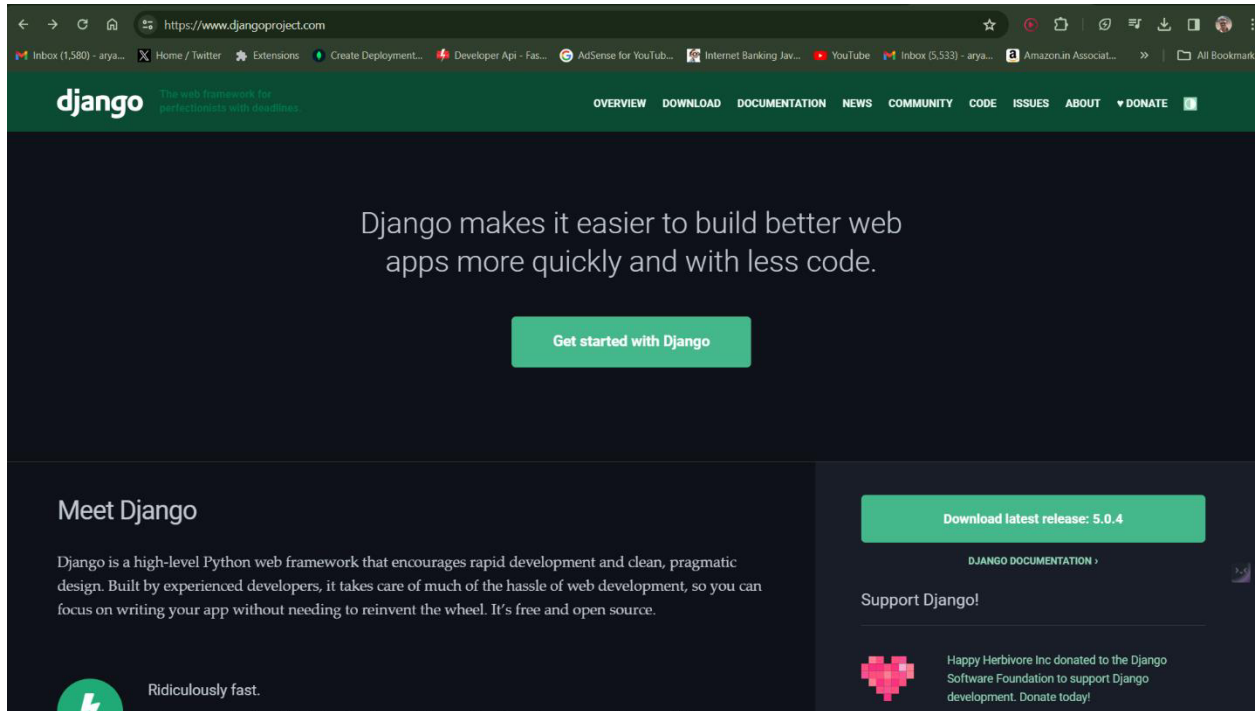
Step – 11: Now that VS Code installation is successful, the page appears as below:



We can change the look as per our choice and continue working on it.

Here are the steps you can follow: Steps to Install Django

To install Django, first visit to **Django official site** (<https://www.djangoproject.com>) and download Django by clicking on the download section.

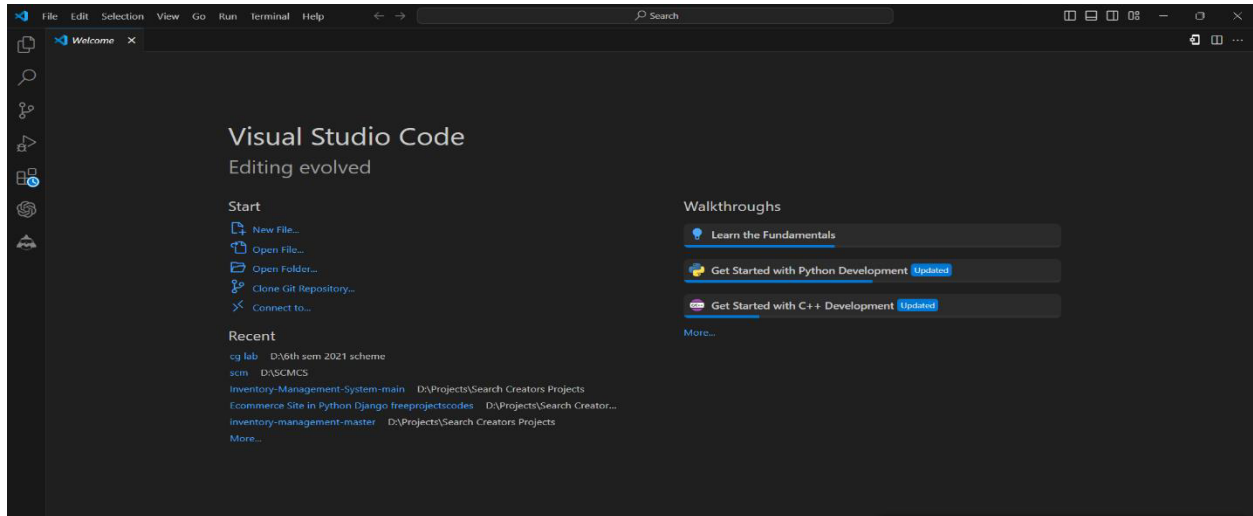


Experiment-02

Creation of virtual environment, Django project and App should be demonstrated.

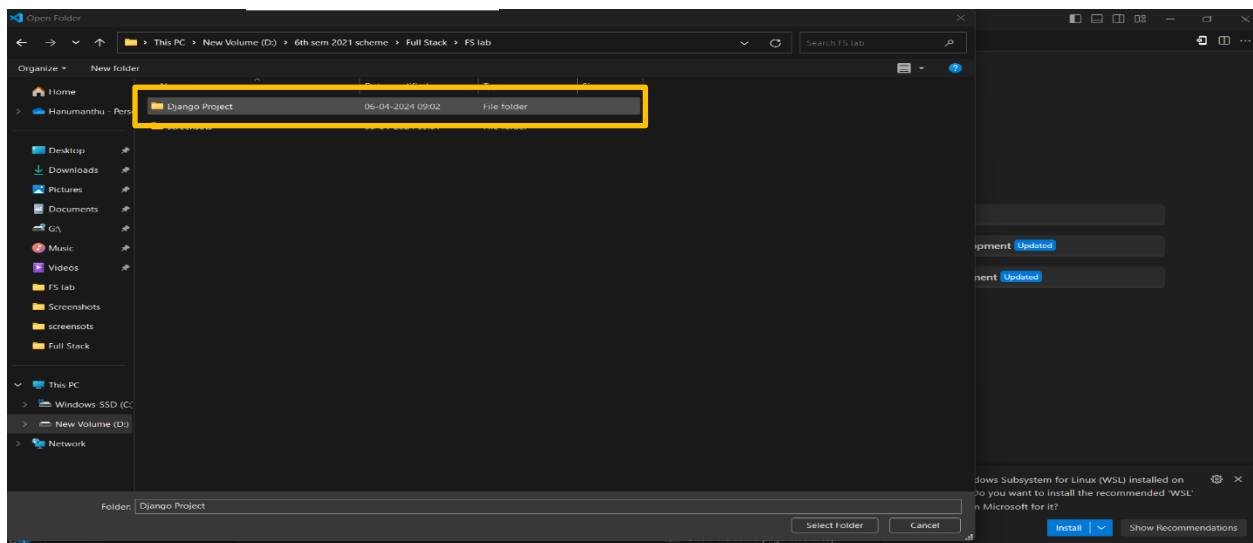
Here are the steps you can follow to create virtual environment: Steps to Create Virtual Environment, Django Project and App

Step-01: Open Visual Studio Code



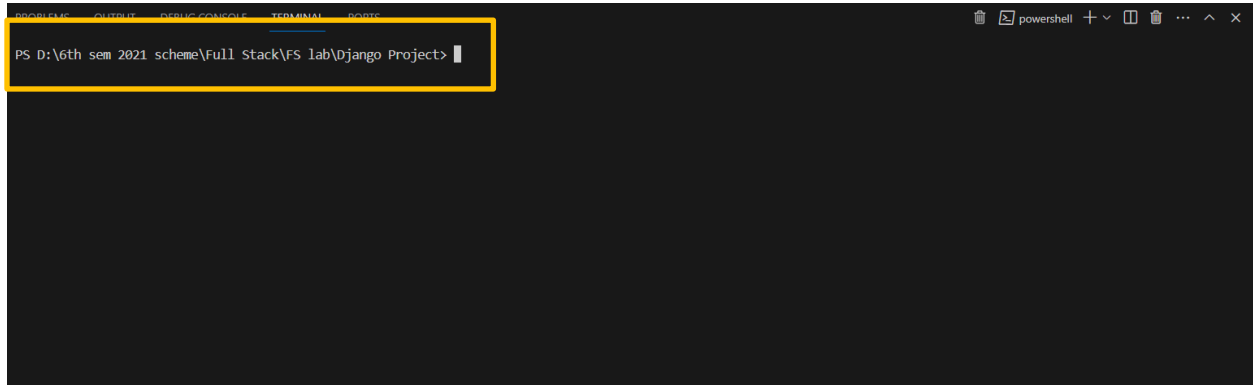
Step-02: Create a new folder for your project

In VS Code, go to **File > Open...** and create a new folder or select an existing folder where you want to create your Django project.



Step-03: Open the integrated terminal

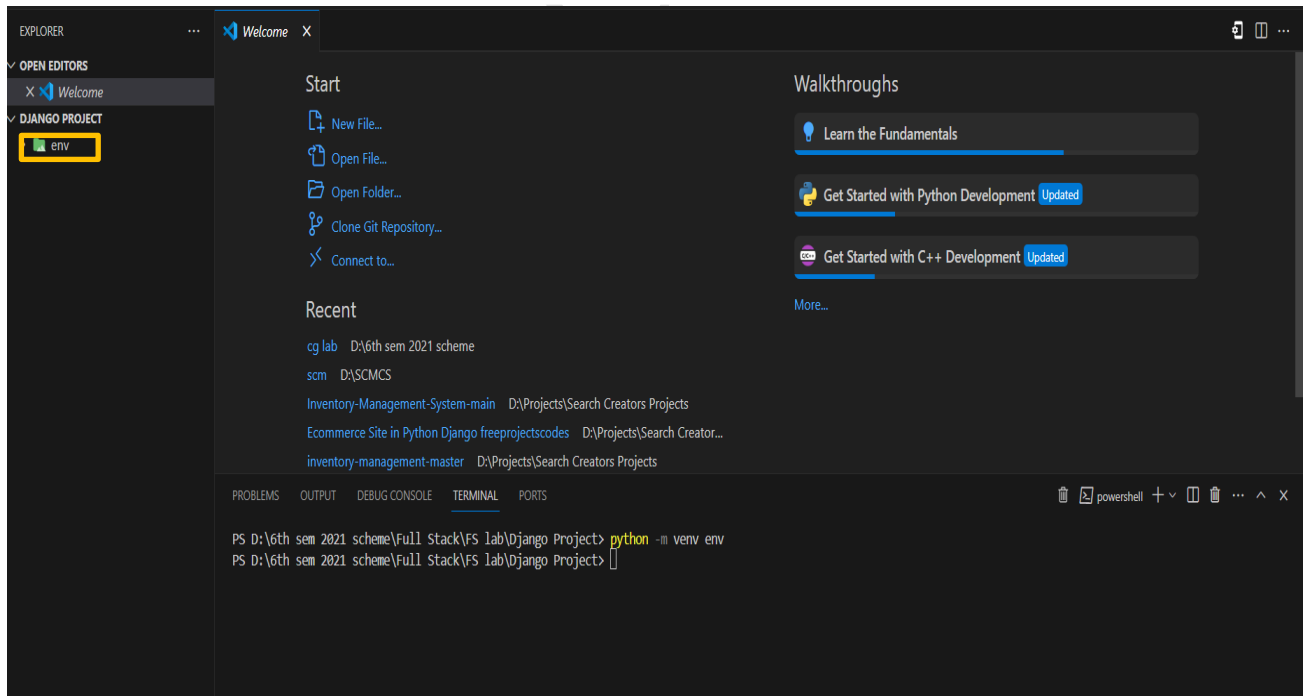
In VS Code, go to **View > Terminal** or use the keyboard shortcut **Ctrl+`** (Windows/Linux) or **Cmd+`** (macOS) to open the integrated terminal.



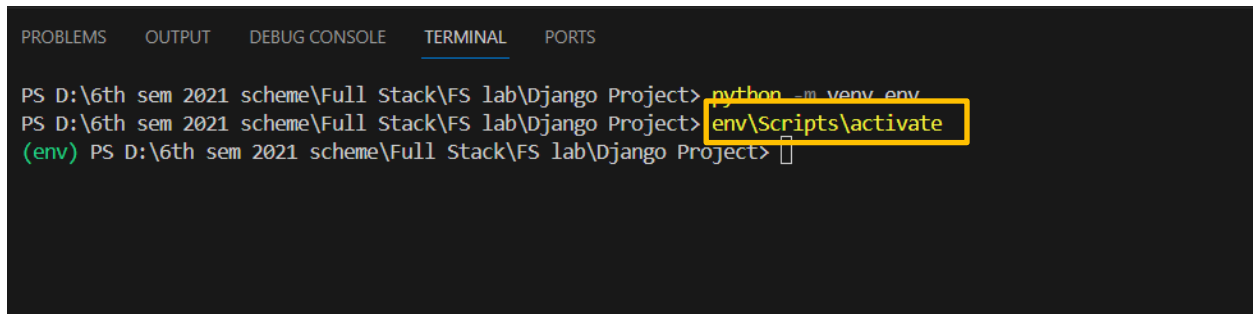
Step-03: Create a virtual environment

In the terminal, run the following command to create a new virtual environment:

python -m venv env



Step-04: Activate the virtual environment

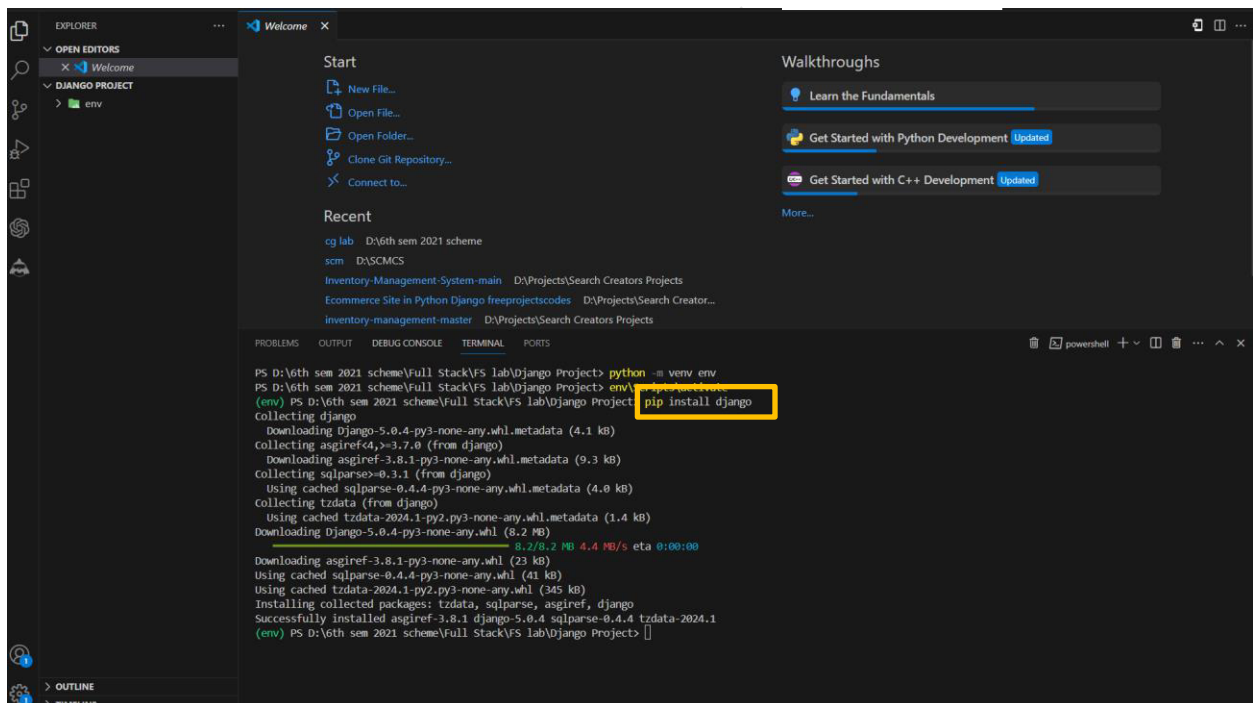


```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS D:\6th sem 2021 scheme\Full Stack\F5 lab\Django Project> python -m venv env
PS D:\6th sem 2021 scheme\Full Stack\F5 lab\Django Project> env\Scripts\activate
(env) PS D:\6th sem 2021 scheme\Full Stack\F5 lab\Django Project> 
```

Step-05: Install Django

With the virtual environment active, install Django by running the following command in the terminal: **pip install django**



```
EXPLORER Welcome
OPEN EDITORS
Welcome
DIANGO PROJECT
env

Start
New File...
Open File...
Open Folder...
Clone Git Repository...
Connect to...

Recent
cg lab D:\6th sem 2021 scheme
scm D:\SCMCS
Inventory-Management-System-main D:\Projects\Search Creators Projects
Ecommerce Site in Python Django freeprojectscodes D:\Projects\Search Creator...
Inventory-management-master D:\Projects\Search Creators Projects

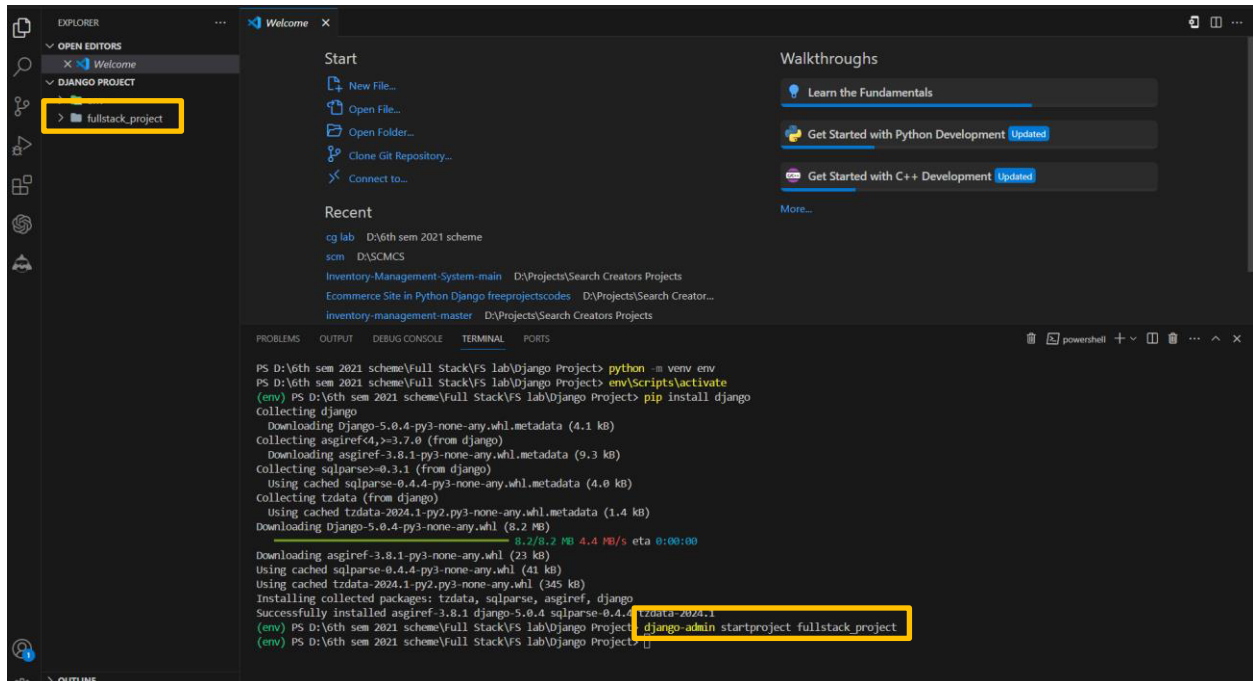
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS D:\6th sem 2021 scheme\Full Stack\F5 lab\Django Project> python -m venv env
PS D:\6th sem 2021 scheme\Full Stack\F5 lab\Django Project> env\Scripts\activate
(env) PS D:\6th sem 2021 scheme\Full Stack\F5 lab\Django Project> pip install django
Collecting django
  Downloading Django-5.0.4-py3-none-any.whl.metadata (4.1 kB)
Collecting asgiref<4,=>3.7.0 (from django)
  Downloading asgiref-3.8.1-py3-none-any.whl.metadata (9.3 kB)
Collecting sqlparse<0.3.1 (from django)
  Using cached sqlparse-0.4.4-py3-none-any.whl.metadata (4.0 kB)
Collecting tzdata (from django)
  Using cached tzdata-2024.1-py2.py3-none-any.whl.metadata (1.4 kB)
Downloading Django-5.0.4-py3-none-any.whl (8.2 MB)
8.2/8.2 MB 4.4 MB/s eta 0:00:00
Downloading asgiref-3.8.1-py3-none-any.whl (23 kB)
Using cached sqlparse-0.4.4-py3-none-any.whl (41 kB)
Using cached tzdata-2024.1-py2.py3-none-any.whl (345 kB)
Installing collected packages: tzdata, sqlparse, asgiref, django
Successfully installed asgiref-3.8.1 django-5.0.4 sqlparse-0.4.4 tzdata-2024.1
(env) PS D:\6th sem 2021 scheme\Full Stack\F5 lab\Django Project> 
```

Step-06: Create a new Django project

Run the following command to create a new Django project:

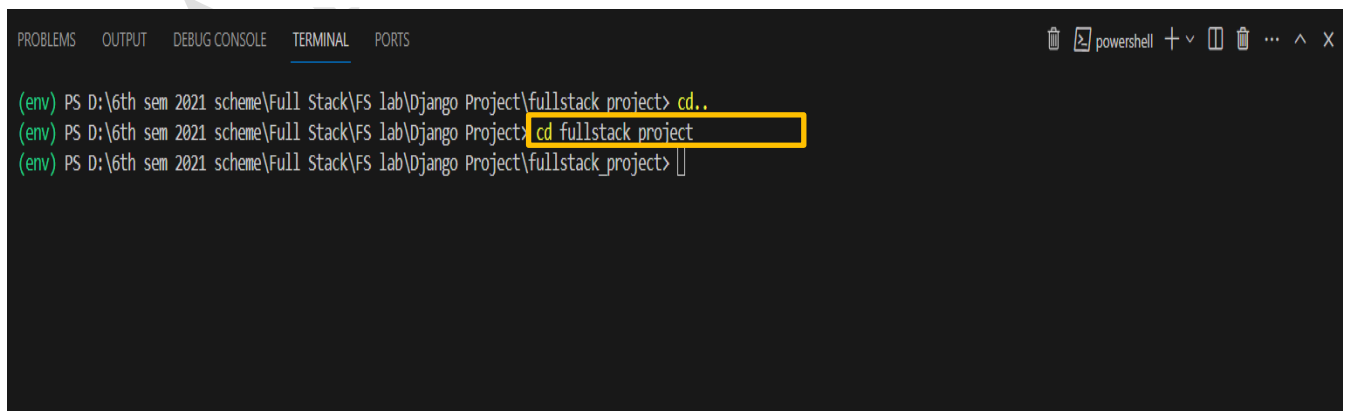
django-admin startproject myproject [change Project name as You Desired]



Step-07: Create a new Django app

In the VS Code terminal, navigate to your project directory:

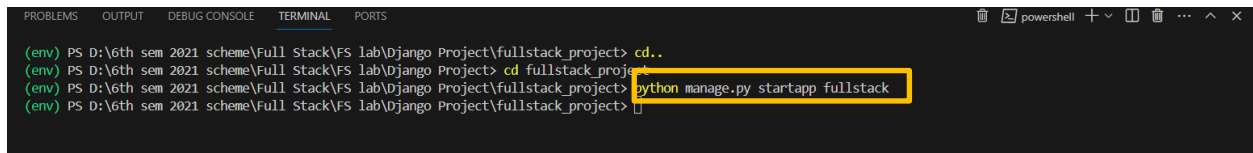
cd myproject [change Project name as You Desired]



Create a new Django app by running the following command

python manage.py startapp myapp

[Replace myapp with the name you want to give to your app.]

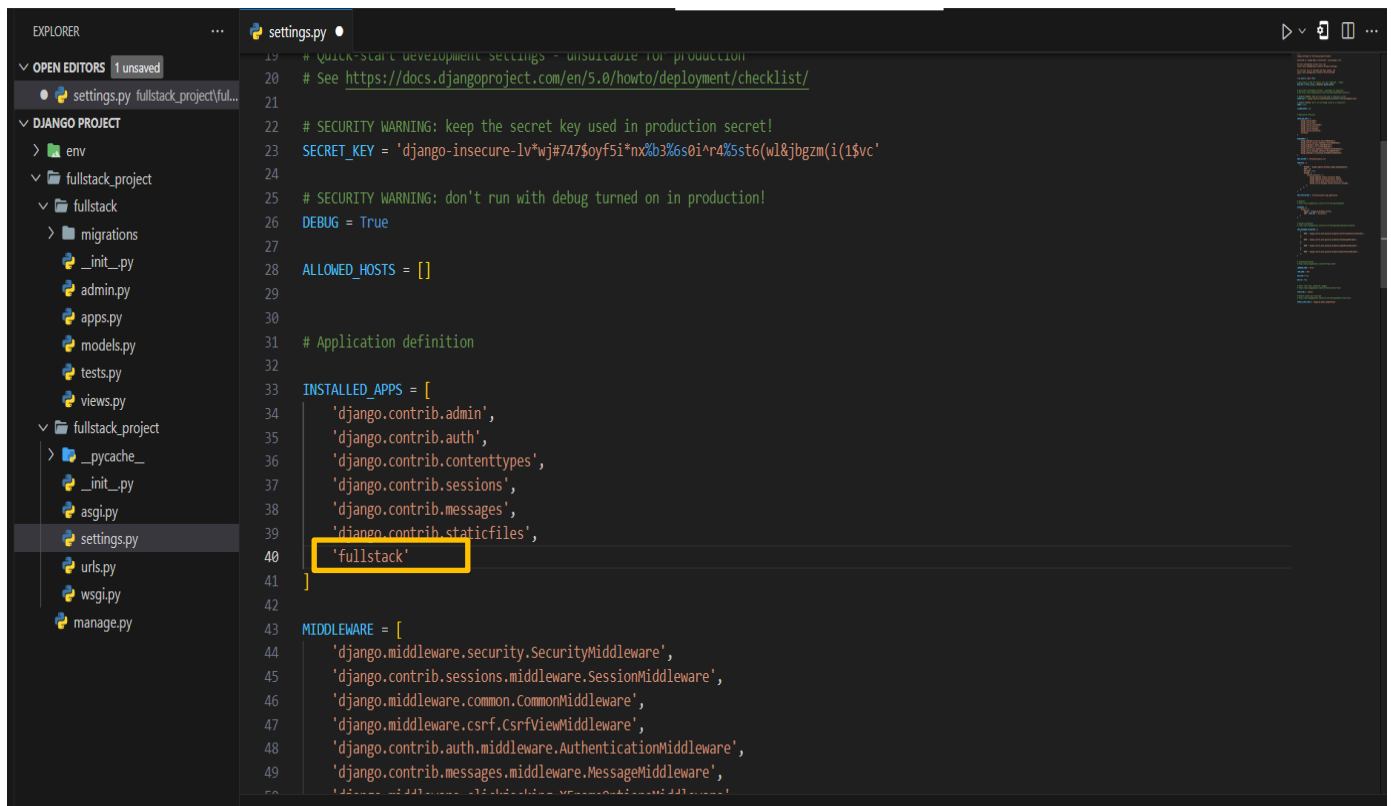


```
(env) PS D:\6th sem 2021 scheme\Full Stack\FS lab\Django Project\fullstack_project> cd..
(env) PS D:\6th sem 2021 scheme\Full Stack\FS lab\Django Project> cd fullstack_project
(env) PS D:\6th sem 2021 scheme\Full Stack\FS lab\Django Project\fullstack_project> python manage.py startapp fullstack
(env) PS D:\6th sem 2021 scheme\Full Stack\FS lab\Django Project\fullstack_project>
```

Step-08: Add the app to the INSTALLED_APPS list

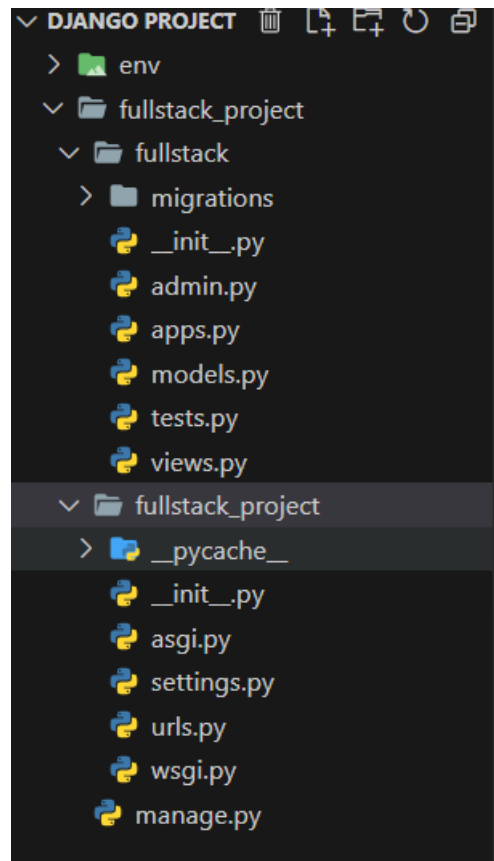
In the VS Code file explorer, locate the settings.py file (usually located in the myproject directory) and open it.

Locate the **INSTALLED_APPS** list and add the name of your new app to the list



```
19 # Quick-start development settings - unsuitable for production
20 # See https://docs.djangoproject.com/en/5.0/howto/deployment/checklist/
21
22 # SECURITY WARNING: keep the secret key used in production secret!
23 SECRET_KEY = 'django-insecure-lv*wj#747$oyf5i*nx%b3%so0i*r4%5st6(wl&jbgzm(i(1$vc'
24
25 # SECURITY WARNING: don't run with debug turned on in production!
26 DEBUG = True
27
28 ALLOWED_HOSTS = []
29
30 # Application definition
31
32 INSTALLED_APPS = [
33     'django.contrib.admin',
34     'django.contrib.auth',
35     'django.contrib.contenttypes',
36     'django.contrib.sessions',
37     'django.contrib.messages',
38     'django.contrib.staticfiles',
39     'fullstack'
40 ]
41
42
43 MIDDLEWARE = [
44     'django.middleware.security.SecurityMiddleware',
45     'django.contrib.sessions.middleware.SessionMiddleware',
46     'django.middleware.common.CommonMiddleware',
47     'django.middleware.csrf.CsrfViewMiddleware',
48     'django.contrib.auth.middleware.AuthenticationMiddleware',
49     'django.contrib.messages.middleware.MessageMiddleware',
50     'django.middleware.clickjacking.XFrameOptionsMiddleware',
51 ]
```

Here is the Django project structure that you wanted to create



after creating your Django app, the next step is to **create database migrations** for your app's models (if you have any defined). Here's how you can do that using the **python manage.py makemigrations** command in Visual Studio Code (VS Code)

```
(env) PS D:\6th sem 2021 scheme\Full Stack\FS lab\Django Project\fullstack_project> python manage.py makemigrations
No changes detected
```

Step-08: Apply the migrations

Once you've reviewed the migration files and are ready to apply the changes to your database, run the following command in the terminal:

```
(env) PS D:\6th sem 2021 scheme\Full Stack\FS lab\Django Project\fullstack_project> 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
```

Step-09: Run Your Project

Here's how you can run the Django development server using the **python manage.py runserver** command in Visual Studio Code (VS Code)

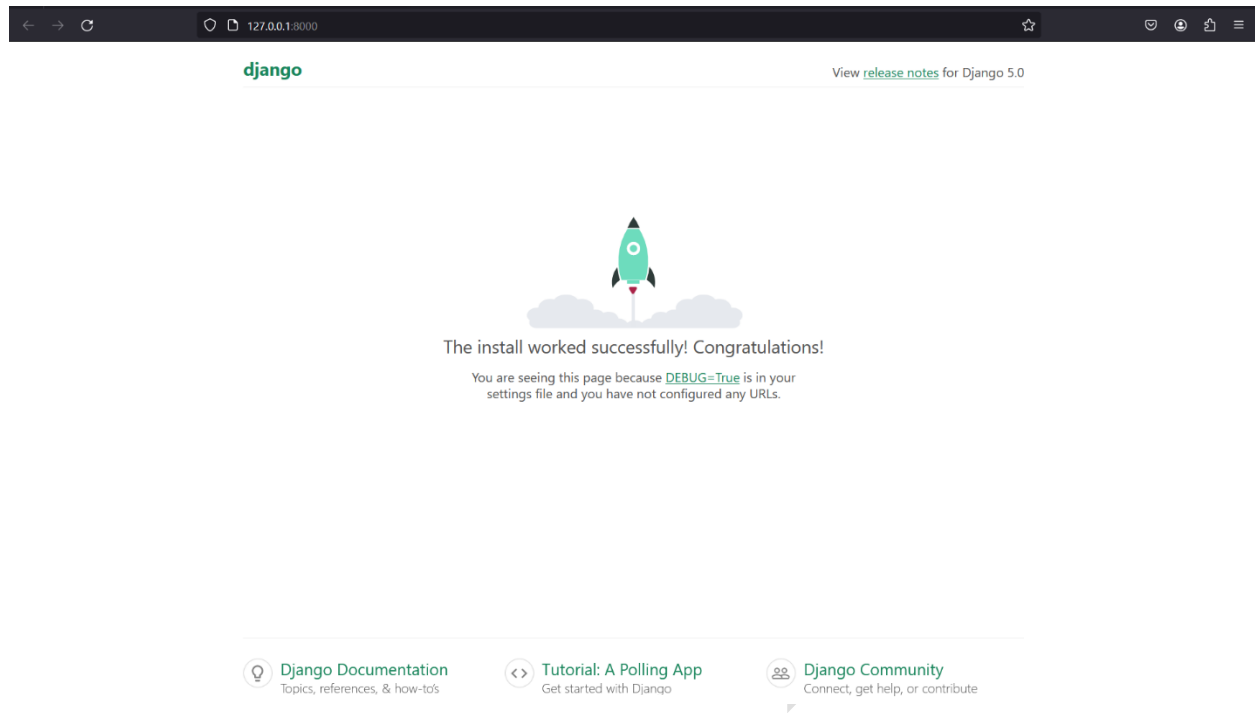
```
(env) PS D:\6th sem 2021 scheme\Full Stack\FS lab\Django Project\fullstack_project> python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
April 06, 2024 - 11:23:06
Django version 5.0.4, using settings 'fullstack_project.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

Check the server output

The terminal will display the URL where the development server is running, typically <http://127.0.0.1:8000/>. It will also show any startup logs or warnings, if any. Open the development server in your browser Copy the URL from the terminal output (e.g., <http://127.0.0.1:8000/>) and paste it into your web browser's address bar.

Here Displays the Development Server After Running Server



3. Django app that displays tables of squares of pairs of numbers input in the URL.

Views.py

```
from datetime import date

from django.http import HttpResponse from django.shortcuts import render
from django.template import Context, Template def create_table_of_squares(request,s,n):
result=""
for i in range(1,n+1): result+="

" +str(s)+"*" +str(i)+"="+str((s*i))+"</p>"
return HttpResponse(result)


```

URLS.py

```
from django.contrib import admin
from django.urls import path, re_path
from app1.views import create_table_of_squares urlpatterns = [
path('admin/', admin.site.urls),
path('cts/<int:s>/<int:n>', create_table_of_squares),
]
```

← → ↺ 127.0.0.1:8000/cts/3/6

Table of squares

3*1=3

3*2=6

3*3=9

3*4=12

4. Develop a Django app that displays number of vowels and consonants and also list of vowels and consonants for any input sentence specified in the URL.

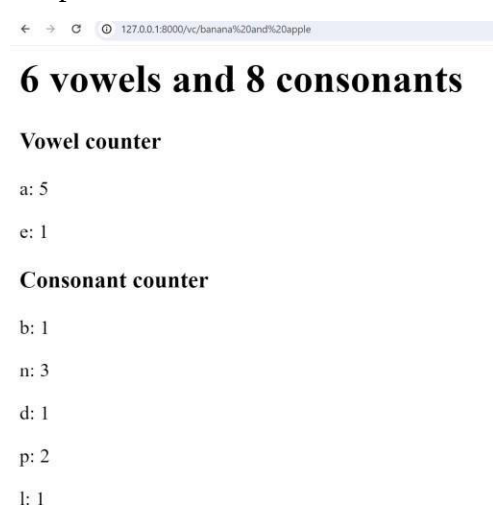
Views.py

```
def vc(request,sentence): vowel_cnt=0
cons_cnt=0 vowel_dict=dict() cons_dict=dict()
for letter in sentence: if letter.isalpha():
if letter in "aeiouAEIOU": vowel_cnt=vowel_cnt+1
vowel_dict[letter]=vowel_dict.get(letter,0)+1 else:
cons_cnt=cons_cnt+1 cons_dict[letter]=cons_dict.get(letter,0)+1
result="<h1>%d Vowels and %d Consonants</h1>" % (vowel_cnt,cons_cnt) result+="<h2>Vowel
Counter</h2>"
for key,value in vowel_dict.items(): result+="<p>%s:%d</p>"%(key,value)
result+="<h2>Consonant Counter</h2>" for key,value in cons_dict.items():
result+="<p>%s:%d</p>"%(key,value) return HttpResponse(result)
```

URLS.py

```
from django.contrib import admin
from django.urls import path, re_path from app2.views import vc urlpatterns = [
path('vc/<str:sentence>', vc),
]
```

Output:



5. Develop a Django Project to check whether a given number is prime or not

Step 1: Create Django Project and App

```
django-admin startproject primecheck cd primecheck
```

```
python manage.py startapp checker
```

Step 2: primecheck/settings.py

```
Add 'checker', to INSTALLED_APPS: INSTALLED_APPS = [
```

```
...
```

```
'checker',
```

```
]
```

Step 3: primecheck/urls.py

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

```
urlpatterns = [
```

```
path('admin/', admin.site.urls), path("", views.home, name='home'),
```

```
]
```

Step 4: checker/views.py

```
from django.shortcuts import render def is_prime(n):
```

```
if n <= 1:
```

```
return False if n <= 3:
```

```
return True
```

```
if n % 2 == 0 or n % 3 == 0: return False
```

```
i = 5
```

```
while i * i <= n:
```

```
if n % i == 0 or n % (i + 2) == 0: return False
```

```
i += 6
```

```
return True
```

```
def home(request): result = None number = None
```

```
if request.method == 'POST':
```

```
number = request.POST.get('number') try:
```

```
number = int(number) result = is_prime(number)
```

```
except ValueError:
```

```
result = "Invalid input. Please enter a valid integer."
```

```
return render(request, 'checker/home.html', {'result': result, 'number': number})
```

Step 5: Create Template checker/templates/checker/home.html

Create the folders if they don't exist. Then add this HTML:

```
<!DOCTYPE html>
<html>
<head>
<title>Prime Number Checker</title>
</head>
<body>
<h1>Prime Number Checker</h1>
<form method="post">
{% csrf_token %}
<label>Enter a number:</label>
<input type="text" name="number" required>
<button type="submit">Check</button>
</form>
```

```
{% if result is not None %}
<h2>
{% if result == True %}
{{ number }} is a prime number.
{% elif result == False %}
{{ number }} is not a prime number.
{% else %}
{{ result }}
{% endif %}
</h2>
{% endif %}
</body>
</html>
```

6. Develop a Django app that performs student registration to a course. It should also display list of students registered for any selected course. Create students and course as models with enrolment as ManyToMany field.

```
#settings.py
INSTALLED_APPS = [
    'django.contrib.admin', 'django.contrib.auth', 'django.contrib.contenttypes', 'django.contrib.sessions',
    'django.contrib.messages', 'django.contrib.staticfiles', 'courses',]
```

```
#urls.py
from django.contrib import admin
from django.urls import path, include
urlpatterns = [
    path("admin/", admin.site.urls),]
```

```
#courses/views.py
```

```
from django.shortcuts import render, redirect,
get_object_or_404 from .models import Student, Course
from .forms import StudentRegistrationForm, CourseSelectionForm
```

```
def register_student(request): if request.method == 'POST':
student = get_object_or_404(Student, id=student_id) if request.method == 'POST':
    form = CourseSelectionForm(request.POST)
    if form.is_valid():
        course = form.cleaned_data['course'] course.students.add(student)
        return redirect('course_students', course_id=course.id)
    else:
        form = CourseSelectionForm()
        return render(request, 'courses/select_course.html', {'form': form, 'student': student})
```

```
def course_students(request, course_id):
course = get_object_or_404(Course, id=course_id) students = course.students.all()

#Courses/urls.py
from django.urls import path from . import views
urlpatterns = [
```

```

path('register/', views.register_student, name='register_student'),
path('select_course/<int:student_id>/', views.select_course,

#courses/models.py
from django.db import models class
class Student(models.Model):
    first_name = models.CharField(max_length=50) last_name = models.CharField(max_length=50)
    email = models.EmailField(unique=True)

    def str (self):
        return f'{self.first_name} {self.last_name}'

```

```

class Course(models.Model):
    name = models.CharField(max_length=100) description = models.TextField()
    students = models.ManyToManyField(Student, related_name='courses')
    def str (self):
        return f'{self.first_name} {self.last_name}'

```

```

class Course(models.Model):
    name = models.CharField(max_length=100) description = models.TextField()
    students = models.ManyToManyField(Student, related_name='courses')
    def str (self):
        return f'{self.first_name} {self.last_name}'

```

```

class Course(models.Model):
    name = models.CharField(max_length=100) description = models.TextField()
    students = models.ManyToManyField(Student, related_name='courses')

```

```

#courses/forms.py

```

```

# courses/admin.py

```

```

from django.contrib import admin from .models import Student, Course
class StudentAdmin(admin.ModelAdmin):
    list_display = ('first_name', 'last_name', 'email')
courses/admin.py
from django.contrib import admin from .models import Student, Course

```

```
class StudentAdmin(admin.ModelAdmin):
list_display = ('first_name', 'last_name', 'email')
admin.site.register(Course, CourseAdmin)

#templates\courses\course_students.html
<!-- courses/templates/courses/course_students.html -->
```

```
<!DOCTYPE html>
<html>
<head>
<title>Students Enrolled in {{ course.name }}</title>
</head>
<body>
<h1>Students Enrolled in {{ course.name }}</h1>
<ul>
{% for student in students %}
<li>{{ student.first_name }} {{ student.last_name }}</li>
{% endfor %}
```

```
#templates/courses/register_student.html
<!-- courses/templates/courses/register_student.html -->
<!DOCTYPE html>
<html>
<head>
<title>Register Student</title>
</head>
<body>
<h1>Register Student</h1>
```

```
#courses/templates/courses/select_course.html
<!-- courses/templates/courses/select_course.html -->
<!DOCTYPE html>
<html>
```

<head>

<title>Select Course</title>

</head>

<body>

<h1>Select Course for {{ student.first_name }} {{ student.last_name }}</h1>

7. For student and course models created in Lab experiment for Module2, register admin interfaces, perform migrations and illustrate data entry through admin forms.

Settings.py

```
INSTALLED_APPS = [  
"django.contrib.admin", "django.contrib.auth", "django.contrib.contenttypes",  
"django.contrib.sessions",  
"django.contrib.messages",  
"django.contrib.staticfiles",  
'courses', # Add your app here  
]
```

#urls.py

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

```
urlpatterns = [  
path("admin/", admin.site.urls),  
path("", include('courses.urls')), # Include the app URLs  
]
```

courses/views.py

```
from django.shortcuts import render, redirect from .models import Student, Course  
from .forms import StudentRegistrationForm, CourseSelectionForm
```

```
def register_student(request): if request.method == 'POST':  
form = StudentRegistrationForm(request.POST) if form.is_valid():  
student = form.save()  
return redirect('select_course', student_id=student.id)  
else:  
form = StudentRegistrationForm()  
return render(request, 'courses/register_student.html', {'form': form})  
def select_course(request, student_id):
```

```
student = Student.objects.get(id=student_id) if request.method == 'POST':  
form = CourseSelectionForm(request.POST) if form.is_valid():  
course = form.cleaned_data['course'] course.students.add(student)
```

```

return redirect('course_students', course_id=course.id)
else:
form = CourseSelectionForm()
return render(request, 'courses/select_course.html', {'form': form, 'student': student})

def course_students(request, course_id): course = Course.objects.get(id=course_id) students =
course.students.all()
return render(request, 'courses/course_students.html', {'course': course, 'students': students})

# courses/urls.py
from django.urls import path from . import views

urlpatterns = [
path('register/', views.register_student, name='register_student'),
path('select_course/<int:student_id>/', views.select_course,
name='select_course'),
path('course_students/<int:course_id>/', views.course_students, name='course_students'),
]
# courses/models.py
from django.db import models class Student(models.Model):
first_name = models.CharField(max_length=50) last_name = models.CharField(max_length=50)
email = models.EmailField(unique=True)

def str (self):
return f"{self.first_name} {self.last_name}"

class Course(models.Model):
name = models.CharField(max_length=100) description = models.TextField()
students = models.ManyToManyField(Student, related_name='courses')

def str (self): return self.name

# courses/forms.py
from django import forms
from .models import Student, Course

```

```

class StudentRegistrationForm(forms.ModelForm):
    class Meta:
        model = Student
        fields = ['first_name', 'last_name', 'email']

class CourseSelectionForm(forms.Form):
    course = forms.ModelChoiceField(queryset=Course.objects.all(), required=True)

# courses/admin.py
from django.contrib import admin
from .models import Student, Course

class StudentAdmin(admin.ModelAdmin):
    list_display = ('first_name', 'last_name', 'email')
    search_fields = ('first_name', 'last_name', 'email')

class CourseAdmin(admin.ModelAdmin):
    list_display = ('name', 'description')
    search_fields = ('name',)
    filter_horizontal = ('students',)

admin.site.register(Student, StudentAdmin)
admin.site.register(Course, CourseAdmin)

#courses/templates/courses/course_students.html
<!-- courses/templates/courses/course_students.html -->
<!DOCTYPE html>
<html>
<head>
<title>Students Enrolled in {{ course.name }}</title>
</head>
<body>
<h1>Students Enrolled in {{ course.name }}</h1>
<ul>
{% for student in students %}
<li>{{ student.first_name }} {{ student.last_name }}</li>
{% endfor %}
</ul>
<a href="{% url 'register_student' %}">Register another student</a>
</body>

```

```
</html>
```

```
#courses/templates/courses/register_student.html
```

```
<!-- courses/templates/courses/register_student.html -->
```

```
<!DOCTYPE html>
```

```
<html>
```

```
<head>
```

```
<title>Register Student</title>
```

```
</head>
```

```
<body>
```

```
<h1>Register Student</h1>
```

```
<form method="post">
```

```
{% csrf_token %}
```

```
{{ form.as_p }}
```

```
<button type="submit">Register</button>
```

```
</form>
```

```
</body>
```

```
</html>
```

```
#courses/templates/courses/select_course.html
```

```
<!-- courses/templates/courses/select_course.html -->
```

```
<!DOCTYPE html>
```

```
<html>
```

```
<head>
```

```
<title>Select Course</title>
```

```
</head>
```

```
<body>
```

```
<h1>Select Course for {{ student.first_name }} {{ student.last_name }}</h1>
```

```
<form method="post">
```

```
{% csrf_token %}
```

```
{{ form.as_p }}
```

```
<button type="submit">Enroll</button>
```

```
</form>
```

```
</body>
```

```
</html>
```

8. For students enrolment developed in Module 2, create a generic class view which displays list of students and detail view that displays student details for any selected student in the list.

```
#Settings.py INSTALLED_APPS = [
"django.contrib.admin", "django.contrib.auth", "django.contrib.contenttypes",
"django.contrib.sessions", "django.contrib.messages", "django.contrib.staticfiles", 'registration',
]

# urls.py
from django.contrib import admin
from django.urls import path, include
urlpatterns = [
path('admin/', admin.site.urls),
path('registration/', include('registration.urls')), # Include the app URLs
]

# Registration/views.py
from django.shortcuts import render, redirect
from django.views.generic import ListView, DetailView from .models import Student
from .forms import StudentForm, ProjectForm

def register_student(request): if request.method == 'POST':
student_form = StudentForm(request.POST) project_form = ProjectForm(request.POST)
if student_form.is_valid() and project_form.is_valid(): project = project_form.save()
student = student_form.save(commit=False) student.project = project

student.save()
return redirect('student_list')
else:
student_form = StudentForm() project_form = ProjectForm()
return render(request, 'registration/register_student.html', {'student_form': student_form,
'project_form': project_form})

class StudentListView(ListView): model = Student
template_name = 'registration/student_list.html' context_object_name = 'students'

class StudentDetailView(DetailView): model = Student
template_name = 'registration/student_detail.html' context_object_name = 'student'
```

```

# Registration/urls.py
from django.urls import path from . import views
urlpatterns = [
    path('register/', views.register_student, name='register_student'), path('students/',
    views.StudentListView.as_view(), name='student_list'), path('students/<int:pk>',
    views.StudentDetailView.as_view(),
    name='student_detail'),
]

# Registration/models.py from django.db import models
class Project(models.Model):
    topic = models.CharField(max_length=200) languages_used =
    models.CharField(max_length=200) duration = models.CharField(max_length=100)
    def str (self): return self.topic

class Student(models.Model):
    first_name = models.CharField(max_length=50) last_name = models.CharField(max_length=50)
    email = models.EmailField(unique=True)
    project = models.OneToOneField(Project, on_delete=models.CASCADE, null=True,
    blank=True)

    def str (self):
        return f"{self.first_name} {self.last_name}"

# Registration/forms.py from django import forms
from .models import Student, Project

class ProjectForm(forms.ModelForm): class Meta:
    model = Project
    fields = ['topic', 'languages_used', 'duration']

class StudentForm(forms.ModelForm): class Meta:
    model = Student
    fields = ['first_name', 'last_name', 'email']

#Registration/templates/registration/register_student.html

```

```
<!-- registration/templates/registration/register_student.html -->
```

```
<!DOCTYPE html>
```

```
<html>
```

```
<head>
```

```
<title>Register Student</title>
```

```
</head>
```

```
<body>
```

```
<h1>Register Student</h1>
```

```
<form method="post">
```

```
{% csrf_token %}
```

```
{{ student_form.as_p }}
```

```
{{ project_form.as_p }}
```

```
<button type="submit">Register</button>
```

```
</form>
```

```
</body>
```

```
</html>
```

```
#Registration/templates/registration/student_detail.html
```

```
<!-- registration/templates/registration/student_detail.html -->
```

```
<!DOCTYPE html>
```

```
<html>
```

```
<head>
```

```
<title>Student Details</title>
```

```
</head>
```

```
<body>
```

```
<h1>Student Details</h1>
```

```
<p><strong>First Name:</strong> {{ student.first_name }}</p>
```

```
<p><strong>Last Name:</strong> {{ student.last_name }}</p>
```

```
<p><strong>Email:</strong> {{ student.email }}</p>
```

```
<p><strong>Project Topic:</strong> {{ student.project.topic }}</p>
```

```
<p><strong>Languages Used:</strong> {{ student.project.languages_used }}</p>
```

```
<p><strong>Duration:</strong> {{ student.project.duration }}</p>
```

```
<a href="{% url 'student_list' %}">Back to Student List</a>
```

```
</body>
```

```
</html>
```

```
#Registration/templates/registration/student_list.html
<!-- registration/templates/registration/student_list.html -->
<!DOCTYPE html>
<html>
<head>
<title>Student List</title>
</head>
<body>
<h1>Registered Students</h1>
<ul>
{% for student in students %}
<li>
<a href="{% url 'student_detail' student.pk %}">
{{ student.first_name }} {{ student.last_name }} - {{ student.project.topic }}
</a>
</li>
{% endfor %}
</ul>
<a href="{% url 'register_student' %}">Register another student</a>
</body>
</html>
```

9. Develop example Django app that performs CSV and PDF generation for any models created in previous laboratory component.

#Settings.py

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

#Student_project/urls.py

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

```
urlpatterns = [  
    path('admin/', admin.site.urls),  
    path('registration/', include('registration.urls')), # Include the app URLs  
]
```

Registration/views.py import csv

```
from django.http import HttpResponse from django.shortcuts import render  
from django.views.generic import ListView, DetailView from .models import Student, Project  
from .forms import StudentForm, ProjectForm from reportlab.lib.pagesizes import letter from  
reportlab.pdfgen import canvas
```

```
def register_student(request): if request.method == 'POST':  
    student_form = StudentForm(request.POST) project_form = ProjectForm(request.POST)  
    if student_form.is_valid() and project_form.is_valid(): project = project_form.save()  
    student = student_form.save(commit=False) student.project = project
```

```
    student.save()  
    return redirect('student_list')  
else:  
    student_form = StudentForm() project_form = ProjectForm()  
    return render(request, 'registration/register_student.html', {'student_form': student_form,  
        'project_form': project_form})
```

```
class StudentListView(ListView): model = Student  
    template_name = 'registration/student_list.html' context_object_name = 'students'
```

```

class StudentDetailView(DetailView): model = Student
template_name = 'registration/student_detail.html' context_object_name = 'student'

def generate_csv(request):
    response = HttpResponse(content_type='text/csv')
    response['Content-Disposition'] = 'attachment; filename="students.csv"'

    writer = csv.writer(response)
    writer.writerow(['First Name', 'Last Name', 'Email', 'Project Topic', 'Languages Used',
    'Duration'])
    students = Student.objects.all().select_related('project')
    for student in students:
        writer.writerow([student.first_name, student.last_name, student.email, student.project.topic,
        student.project.languages_used, student.project.duration])
    return response

def generate_pdf(request):
    response = HttpResponse(content_type='application/pdf')
    response['Content-Disposition'] = 'attachment; filename="students.pdf"'

    buffer = BytesIO()
    p = canvas.Canvas(buffer, pagesize=letter)
    p.setFont("Helvetica", 12)

    students = Student.objects.all().select_related('project')
    y = 750
    p.drawString(30, y, "First Name")
    p.drawString(130, y, "Last Name")

    p.drawString(230, y, "Email")
    p.drawString(330, y, "Project Topic")
    p.drawString(430, y, "Languages Used")
    p.drawString(530, y, "Duration")

    for student in students:
        y -= 30
        p.drawString(30, y, student.first_name)
        p.drawString(130, y, student.last_name)
        p.drawString(230, y, student.email)
        p.drawString(330, y, student.project.topic)
        p.drawString(430, y, student.project.languages_used)
        p.drawString(530, y, student.project.duration)

```

```
p.showPage() p.save()
```

```
buffer.seek(0)
```

```
return FileResponse(buffer, as_attachment=True, filename='students.pdf')
```

```
# Registration/urls.py
```

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

```
urlpatterns = [
```

```
path('register/', views.register_student, name='register_student'), path('students/',
```

```
views.StudentListView.as_view(), name='student_list'), path('students/<int:pk>/',
```

```
views.StudentDetailView.as_view(),
```

```
name='student_detail'),
```

```
path('students/csv/', views.generate_csv, name='generate_csv'), path('students/pdf/',
```

```
views.generate_pdf, name='generate_pdf'),
```

```
]
```

```
# Registration/models.py from django.db import models
```

```
class Project(models.Model):
```

```
topic = models.CharField(max_length=200) languages_used =
```

```
models.CharField(max_length=200) duration = models.CharField(max_length=100)
```

```
def str (self): return self.topic
```

```
class Student(models.Model):
```

```
first_name = models.CharField(max_length=50) last_name = models.CharField(max_length=50)
```

```
email = models.EmailField(unique=True)
```

```
project = models.OneToOneField(Project, on_delete=models.CASCADE, null=True,
```

```
blank=True)
```

```
def str (self):
```

```
return f'{self.first_name} {self.last_name}'
```

```
# Registration/forms.py from django import forms
```

```
from .models import Student, Project
```

```
class ProjectForm(forms.ModelForm): class Meta:
```

```
model = Project
```

```
fields = ['topic', 'languages_used', 'duration']
```

```
class StudentForm(forms.ModelForm): class Meta:
```

```
model = Student
```

```
fields = ['first_name', 'last_name', 'email']
```

```
#Registration/templates/registration/register_student.html
```

```
<!-- registration/templates/registration/register_student.html -->
```

```
<!DOCTYPE html>
```

```
<html>
```

```
<head>
```

```
<title>Register Student</title>
```

```
</head>
```

```
<body>
```

```
<h1>Register Student</h1>
```

```
<form method="post">
```

```
{% csrf_token %}
```

```
{{ student_form.as_p }}
```

```
{{ project_form.as_p }}
```

```
<button type="submit">Register</button>
```

```
</form>
```

```
</body>
```

```
</html>
```

```
#Registration/templates/registration/student_detail.html
```

```
<!-- registration/templates/registration/student_detail.html -->
```

```
<!DOCTYPE html>
```

```
<html>
```

```
<head>
```

```
<title>Student Details</title>
```

```
</head>
```

```
<body>
```

```
<h1>Student Details</h1>
```

```
<p><strong>First Name:</strong> {{ student.first_name }}</p>
```

```
<p><strong>Last Name:</strong> {{ student.last_name }}</p>
```

```
<p><strong>Email:</strong> {{ student.email }}</p>
```

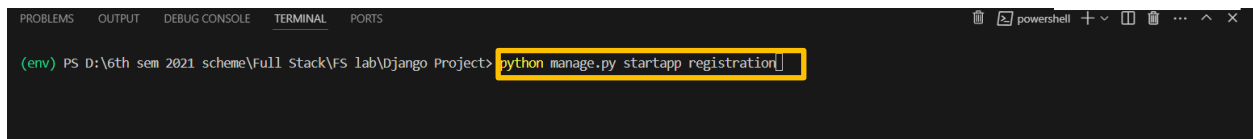
```
<p><strong>Project Topic:</strong> {{ student.project.topic }}</p>
<p><strong>Languages Used:</strong> {{ student.project.languages_used }}</p>
<p><strong>Duration:</strong> {{ student.project.duration }}</p>
<a href="{% url 'student_list' %}">Back to Student List</a>
</body>
</html>
```

```
#Registration/templates/registration/student_list.html<!--
registration/templates/registration/student_list.html -->
<!DOCTYPE html>
<html>
<head>
<title>Student List</title>
</head>
<body>
<h1>Registered Students</h1>
<ul>
{% for student in students %}
<li>
<a href="{% url 'student_detail' student.pk %}">
{{ student.first_name }} {{ student.last_name }} - {{ student.project.topic }}
</a>
</li>
{% endfor %}
</ul>
<a href="{% url 'register_student' %}">Register another student</a>
<br>
<a href="{% url 'generate_csv' %}">Download CSV</a>
<br>
<a href="{% url 'generate_pdf' %}">Download PDF</a>
</body>
</html>
```

10. Develop a registration page for student enrolment as done in Module 2 but without page refresh using AJAX.

Develop a registration page for student enrolment as done in Module 2 but without page refresh using AJAX.

Step-01: This app will be created in the Django project we made earlier.

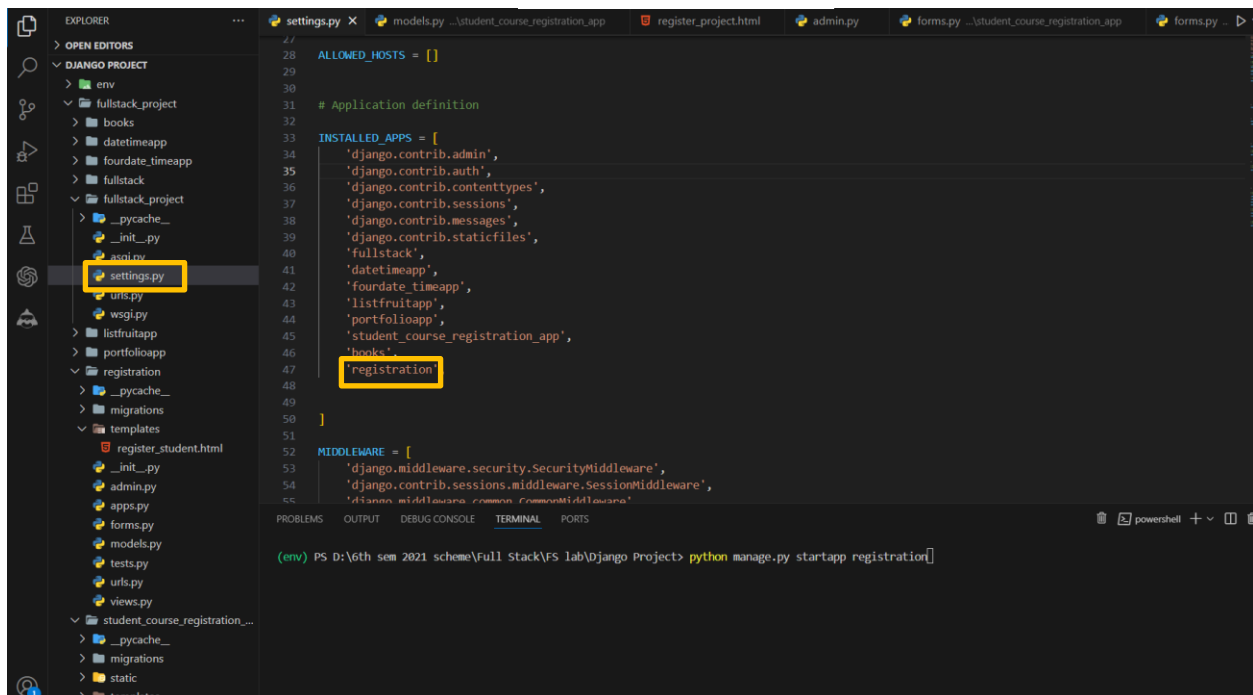


```
(env) PS D:\6th sem 2021 scheme\Full Stack\FS lab\Django Project> python manage.py startapp registration
```

Step-02: Add the app to INSTALLED_APPS

Open the settings.py file in your project's directory (e.g., `fullstack_project/settings.py`).

Locate the `INSTALLED_APPS` list and add the name of your new app to the list:



```
41
42
43 ALLOWED_HOSTS = []
44
45 # Application definition
46
47 INSTALLED_APPS = [
48     'django.contrib.admin',
49     'django.contrib.auth',
50     'django.contrib.contenttypes',
51     'django.contrib.sessions',
52     'django.contrib.messages',
53     'django.contrib.staticfiles',
54     'fullstack',
55     'datetimeapp',
56     'fourdate_timeapp',
57     'listfruitapp',
58     'portfolioapp',
59     'student_course_registration_app',
60     'books',
61     'registration'
62 ]
63
64 MIDDLEWARE = [
65     'django.middleware.security.SecurityMiddleware',
66     'django.contrib.sessions.middleware.SessionMiddleware',
67     'django.middleware.common.CommonMiddleware',
68 ]
69
70
```

Step-03: Create models

- Open the **models.py** file inside the registration app and define your models:

from django.db import models

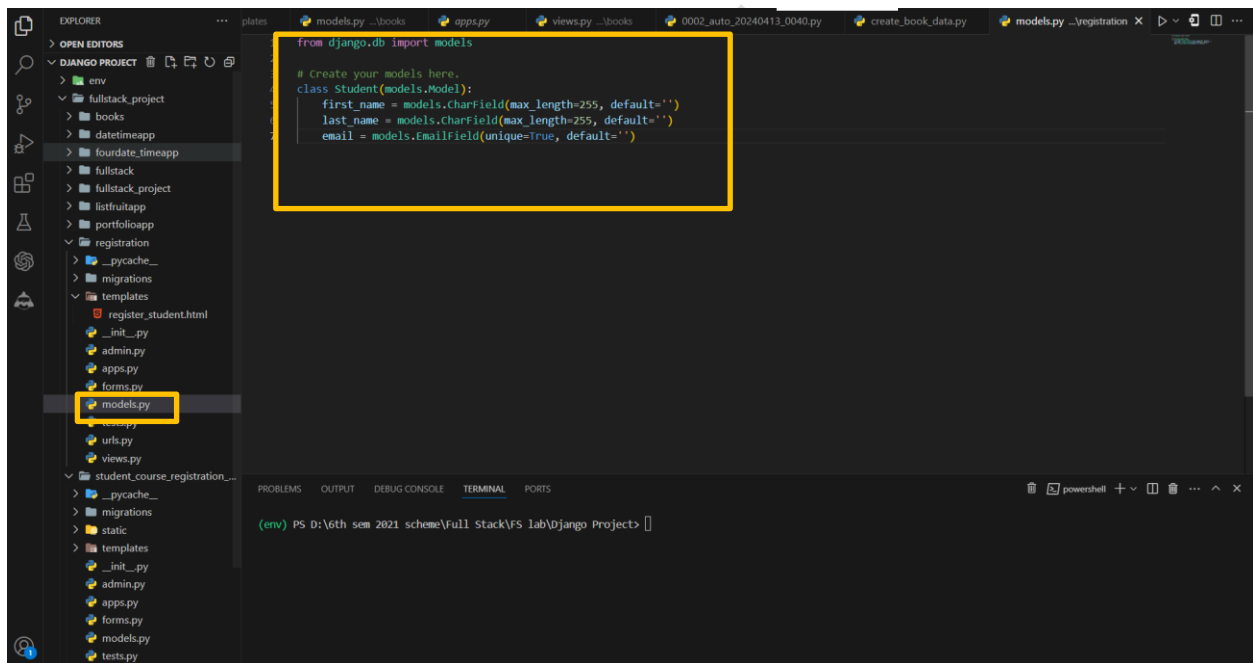
Create your models here.

class Student(models.Model):

first_name = models.CharField(max_length=255, default='')

last_name = models.CharField(max_length=255, default='')

email = models.EmailField(unique=True, default='')



Step-04: Create a `views.py` file in your app and define the views

from django.shortcuts import render, redirect

from django.http import JsonResponse

from .forms import StudentForm

def student(request):

if request.method == 'POST':

form = StudentForm(request.POST)

if form.is_valid():

student = form.save()

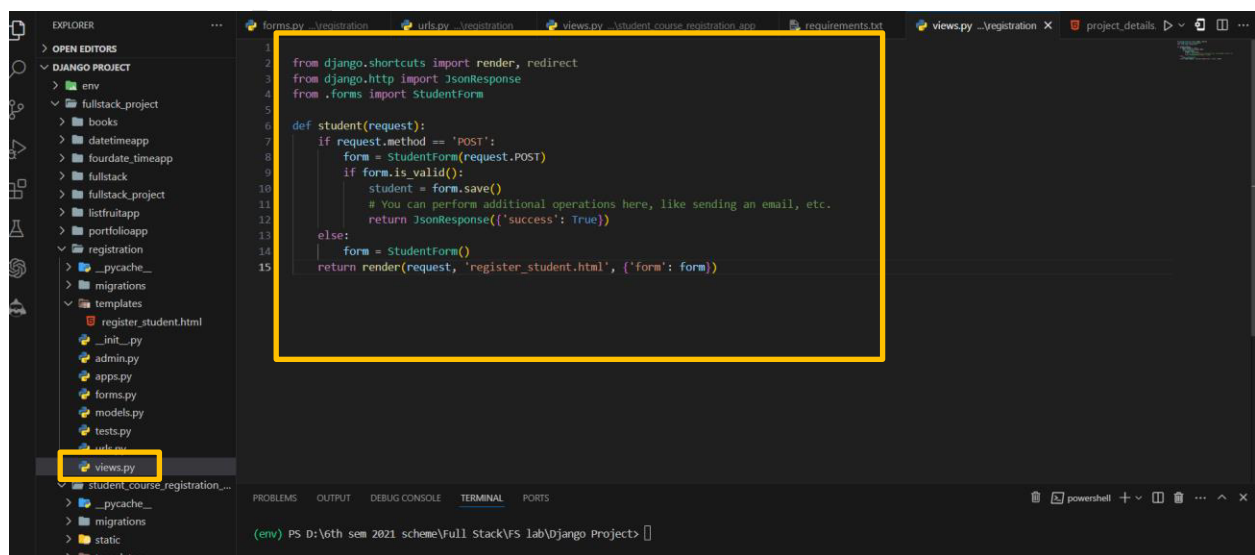
You can perform additional operations here, like sending an email, etc.

return JsonResponse({'success': True})

else:

form = StudentForm()

return render(request, 'register_student.html', {'form': form})



Step-05: Create a **forms.py** file in your app and define the forms

from django import forms

from .models import Student

class StudentForm(forms.ModelForm):

class Meta:

model = Student

fields = ['first_name', 'last_name', 'email']

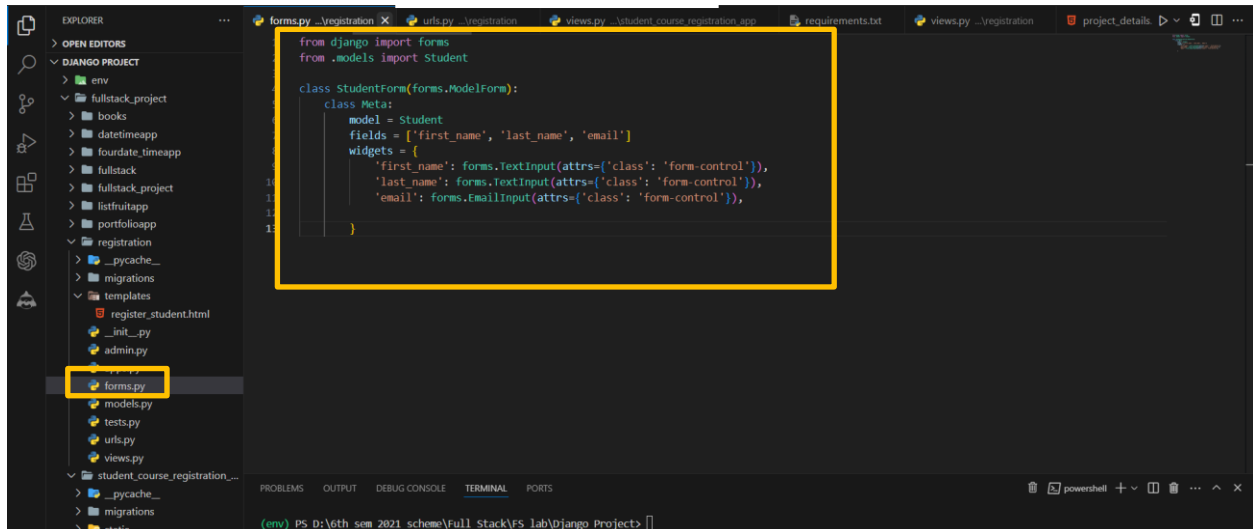
widgets = {

'first_name': forms.TextInput(attrs={'class': 'form-control'}),

'last_name': forms.TextInput(attrs={'class': 'form-control'}),

'email': forms.EmailInput(attrs={'class': 'form-control'}),

}



Step-06: In your app, create a templates directory and an html file

register_student.html

```
{% load static %}

<!DOCTYPE html>

<html>

<head>

    <title>Student Registration</title>

    <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css">

    <script src="{% static 'js/jquery.min.js' %}"></script>

</head>

<body>

    <div class="container">

        <h1>Student Registration</h1>

        <form id="student-form" method="post">

            {% csrf_token %}

            {% for field in form %}

                <div class="form-group">

                    { field.label_tag }

                    { field }

                    % if field.help_text %

                        <small class="form-text text-muted">{{ field.help_text }}</small>

                    {% endif %}

                    {% for error in field.errors %}

                        <div class="alert alert-danger">{{ error }}</div>

                    {% endfor %}

                </div>

            {% endfor %}

        </form>

    </div>

</body>

</html>
```

```
        {% endfor %}

    </div>

    {% endfor %}

    <button type="submit" class="btn btn-primary">Submit</button>

</form>

<div id="response"></div>

</div>

<script>

$(document).ready(function() {

    $('#student-form').on('submit', function(e) {

        e.preventDefault();

        var formData = $(this).serialize();

        $.ajax({

            type: 'POST',

            url: '{% url "student" %}',

            data: formData,

            success: function(response) {

                if (response.success) {

                    $('#response').html('<div class="alert alert-success">Student registered successfully.</div>');

                    $('#student-form')[0].reset();

                } else {

                    $('#response').html('<div class="alert alert-danger">An error occurred. Please try again.</div>');

                }

            },


```

```

        error: function(xhr, errormsg, err) {

            $('#response').html('<div class="alert alert-danger">An error occurred: ' +
xhr.status + ' ' + xhr.responseText + '</div>');

        }

    });

});

});

</script>

</body>

</html>

```

```

({% load static %}
<!DOCTYPE html>
<html>
<head>
<title>Student Registration</title>
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css">
<script src="{% static 'js/jquery.min.js' %}"></script>
</head>
<body>
<div class="container">
<h1>Student Registration</h1>
<form id="student-form" method="post">
    {% csrf token %}
    {% for field in form %}
        <div class="form-group">
            {{ field.label_tag }}
            {{ field }}
            {% if field.help_text %}
                <small class="form-text text-muted">{{ field.help_text }}</small>
            {% endif %}
            {% for error in field.errors %}
                <div class="alert alert-danger">{{ error }}</div>
            {% endfor %}
        </div>
    {% endfor %}
    <button type="submit" class="btn btn-primary">Submit</button>
</form>
<div id="response"></div>
</div>

<script>
$(document).ready(function() {
    $('#student-form').on('submit', function(e) {
        e.preventDefault();
        var formData = $(this).serialize();
        $.ajax({
            type: 'POST',
            url: '{% url "student" %}',
            data: formData

```

Step-07: Update app URLs

project/urls.py

from django.urls import path

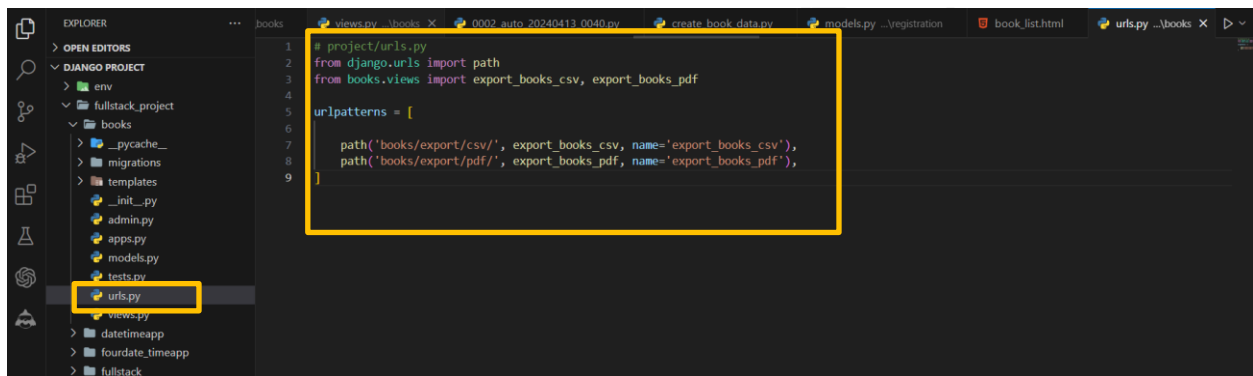
from books.views import export_books_csv, export_books_pdf

urlpatterns = [

 path('books/export/csv/', export_books_csv, name='export_books_csv'),

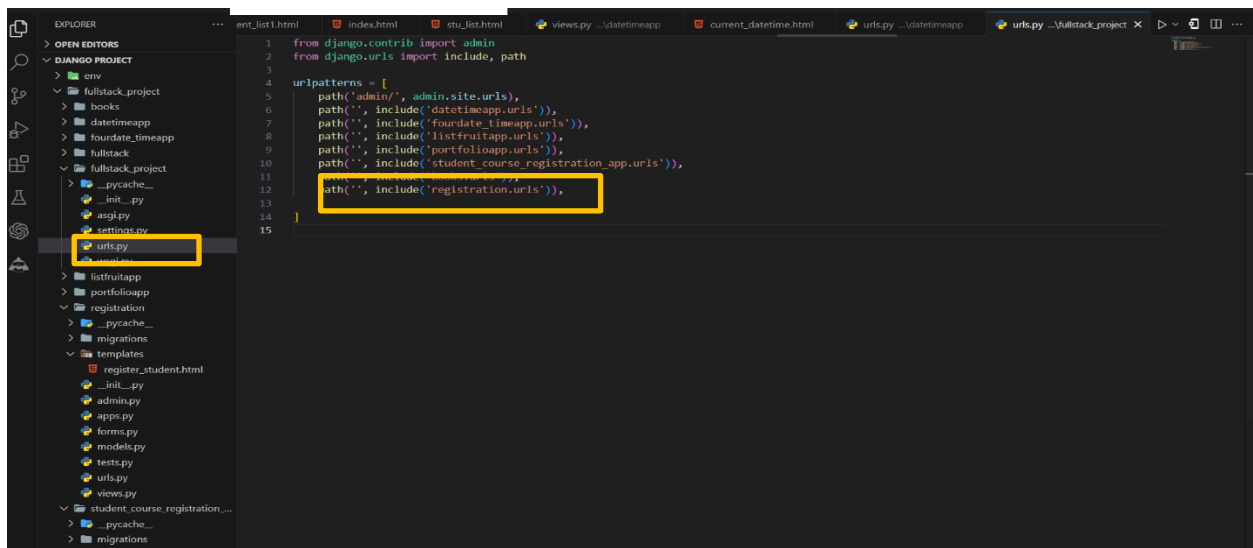
 path('books/export/pdf/', export_books_pdf, name='export_books_pdf'),

]



Step-08: Update project URLs

- Open the **urls.py** file inside your project and include the URLs from the registration app:

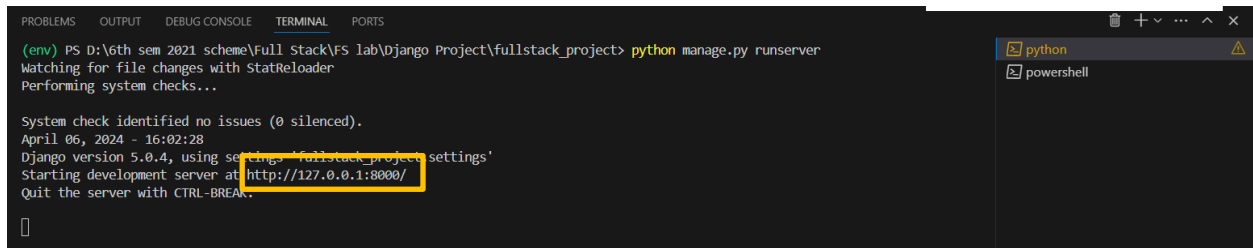


Step-09: Run the development server

- In the VS Code terminal, navigate to your Django project's directory (the one containing `manage.py`).
- Run the development server

`python manage.py runserver`

- Open your web browser and visit <http://127.0.0.1:8000/>.



The screenshot shows a VS Code terminal window with the following output:

```
(env) PS D:\6th sem 2021 scheme\Full Stack\FS lab\Django Project\fullstack_project> python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
April 06, 2024 - 16:02:28
Django version 5.0.4, using settings 'fullstack_project.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
[]
```

The URL `http://127.0.0.1:8000/` is highlighted with a yellow box. To the right of the terminal, there are two tabs: 'python' and 'powershell'.

- Type or copy this <http://127.0.0.1:8000/student/>



The screenshot shows a web browser window with the address bar displaying `127.0.0.1:8000/student/`. The page title is "Student Registration". The form contains three input fields labeled "First name:", "Last name:", and "Email:". Below these fields is a blue "Submit" button.

Fig: Student Registration Screen

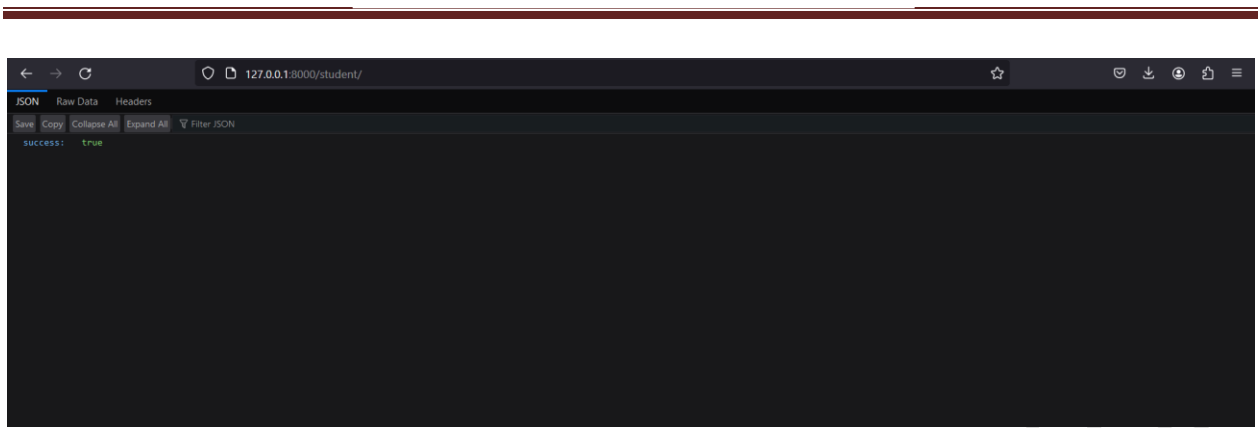


Fig: Student Registration Success Message Screen

Develop a search application in Django using AJAX that displays courses enrolled by a student being searched.

Step-01: Use an existing app i.e. student_course_registration_app

```
[08/Apr/2024 00:21:00] "GET /student-list/1/ HTTP/1.1" 200 1333
[08/Apr/2024 00:21:02] "GET /student-list/2/ HTTP/1.1" 200 1326
PS D:\6th sem 2021 scheme\Full Stack\FS lab\Django Project\fullstack_project> python manage.py makemigrations student_course_registration_app
```

Step-02: Create View Function in a student_course_registration_app views.py

```
def search_students(request):
```

```
    is_ajax = request.headers.get('X-Requested-With') == 'XMLHttpRequest'
```

```
    if is_ajax:
```

```
        query = request.GET.get('query', '')
```

```
        students = Student.objects.filter(first_name__icontains=query) |
```

```
        Student.objects.filter(last_name__icontains=query)
```

```
        results = []
```

```
        for student in students:
```

```
            student_data = {
```

```
                'id': student.id,
```

```
                'name': f'{student.first_name} {student.last_name}',
```

```
                'email': student.email,
```

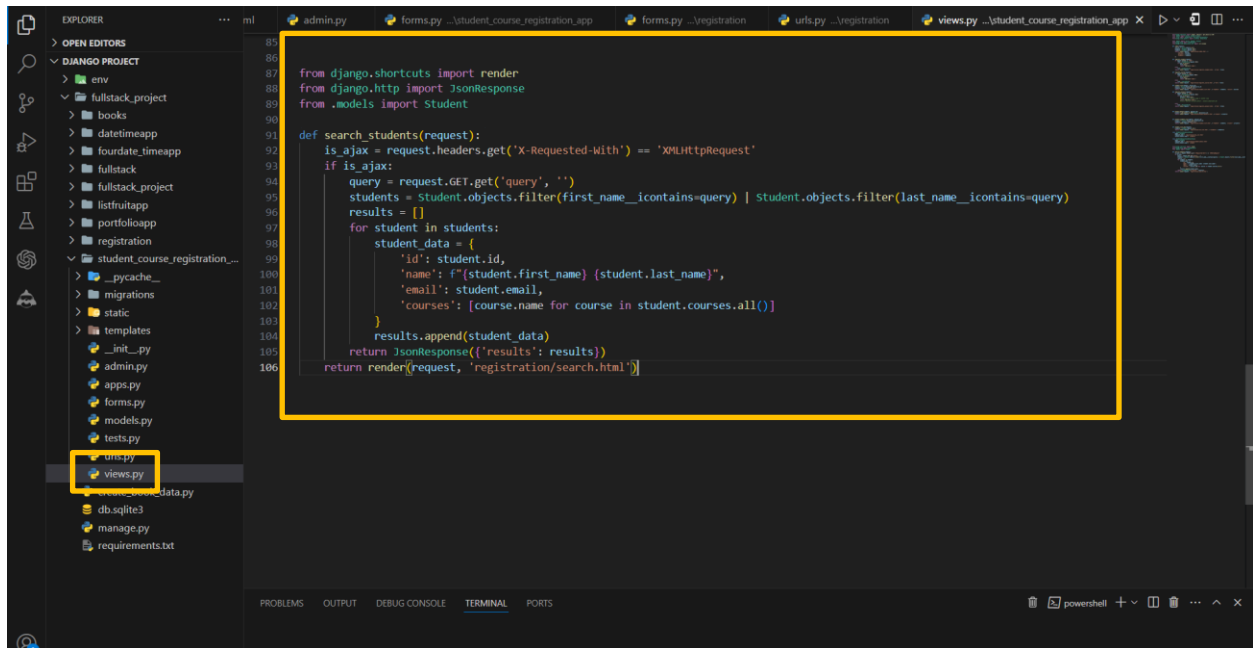
```
                'courses': [course.name for course in student.courses.all()]
```

```
            }
```

```
            results.append(student_data)
```

```
return JsonResponse({'results': results})
```

```
return render(request, 'registration/search.html')
```



Step-02: Create Templates in a student_course_registration_app

Registration/search.html

```
{% load static %}
```

```
<!DOCTYPE html>
```

```
<html>
```

```
<head>
```

```
<title>Search Students</title>
```

```
<link rel="stylesheet"
```

```
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css">
```

```
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
```

```
<script>
```

```
$.ajaxSetup({
```

```
headers: {  
    'X-CSRFToken': '{{ csrf_token }}'  
}  
});
```

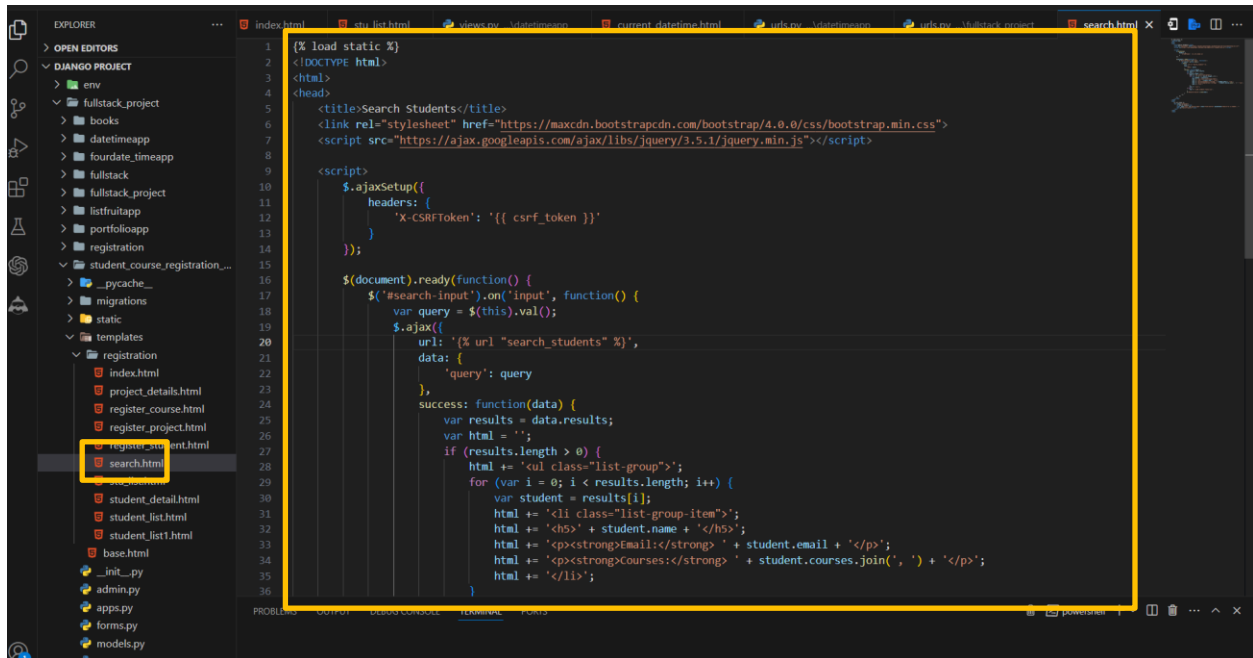
```
$(document).ready(function() {  
    $('#search-input').on('input', function() {  
        var query = $(this).val();  
        $.ajax({  
            url: '{% url "search_students" %}',  
            data: {  
                'query': query  
            },  
            success: function(data) {  
                var results = data.results;  
                var html = "";  
                if (results.length > 0) {  
                    html += '<ul class="list-group">';  
                    for (var i = 0; i < results.length; i++) {  
                        var student = results[i];  
                        html += '<li class="list-group-item">';  
                        html += '<h5>' + student.name + '</h5>';  
                        html += '<p><strong>Email:</strong>' + student.email + '</p>';  
                        html += '<p><strong>Courses:</strong>' + student.courses.join(', ') + '</p>';  
                        html += '</li>';  
                    }  
                }  
            }  
        });  
    });  
});
```

```
        }
        html += '</ul>';
    } else {
        html += '<p>No students found.</p>';
    }
    $('#search-results').html(html);
}

});

});

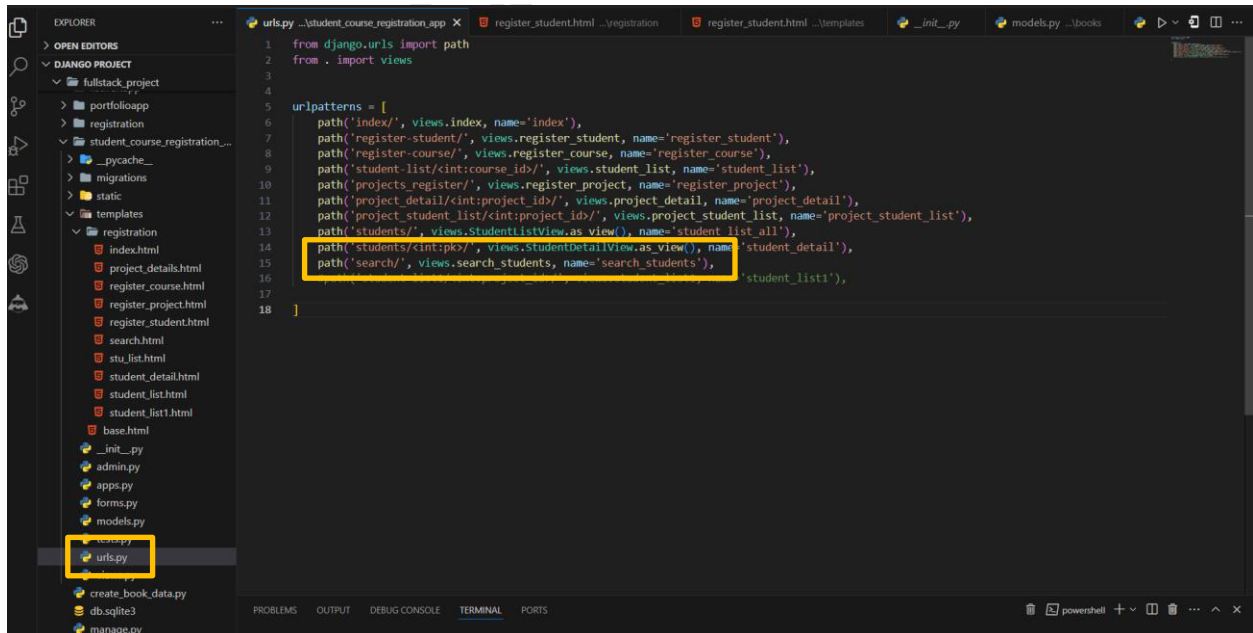
});
</script>
</head>
<body>
    {% csrf_token %}
    <div class="container">
        <h1>Search Students
        <input type="text" id="search-input" class="form-control" placeholder="Search for a
student...">
        <div id="search-results"></div>
    </div>
</body>
</html>
```



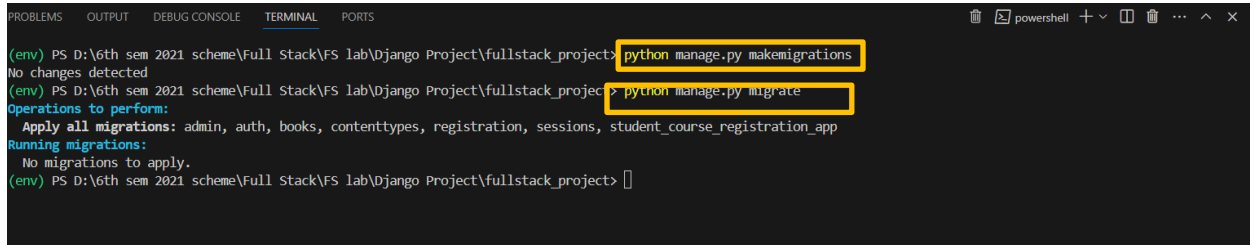
- Very important to add the all The necessary AJAX scripts we provided

Step-03: Update app URLs

- Open the `urls.py` file inside your app and include the URLs



Step-04: Makemigrations and migrate if any changes are made in your application eg. Models etc..



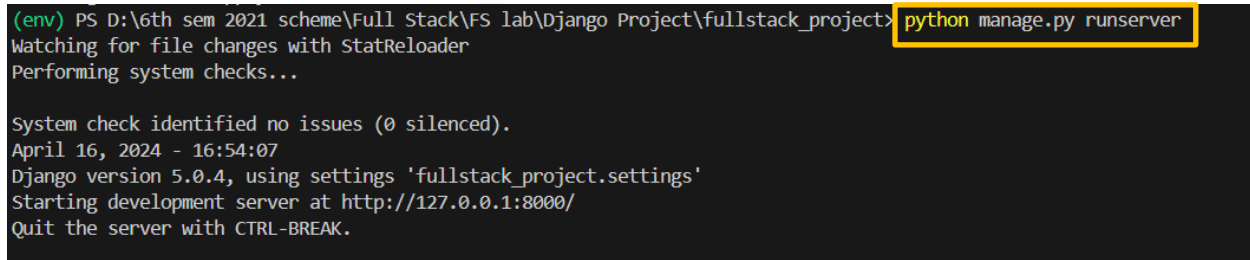
```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
(env) PS D:\6th sem 2021 scheme\Full Stack\FS lab\Django Project\fullstack_project> python manage.py makemigrations
No changes detected
(env) PS D:\6th sem 2021 scheme\Full Stack\FS lab\Django Project\fullstack_project> python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, books, contenttypes, registration, sessions, student_course_registration_app
Running migrations:
  No migrations to apply.
(env) PS D:\6th sem 2021 scheme\Full Stack\FS lab\Django Project\fullstack_project> 
```

Step-05: Run the development server

- In the VS Code terminal, navigate to your Django project's directory (the one containing manage.py).
- Run the development server

python manage.py runserver

- Open your web browser and visit <http://127.0.0.1:8000/>.



```
(env) PS D:\6th sem 2021 scheme\Full Stack\FS lab\Django Project\fullstack_project> python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
April 16, 2024 - 16:54:07
Django version 5.0.4, using settings 'fullstack_project.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

- Type or copy this <http://127.0.0.1:8000/search/>
-

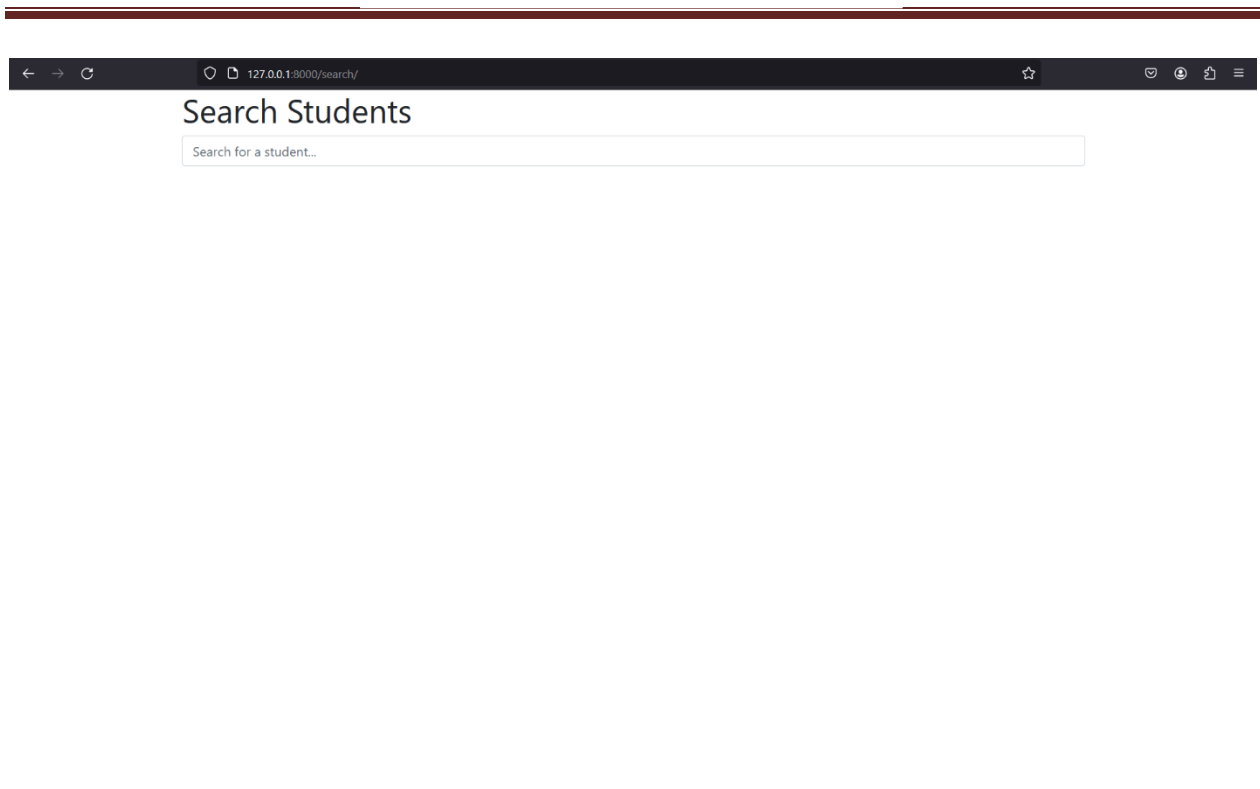


Fig: Before Search Result contains only Search Bar Screen

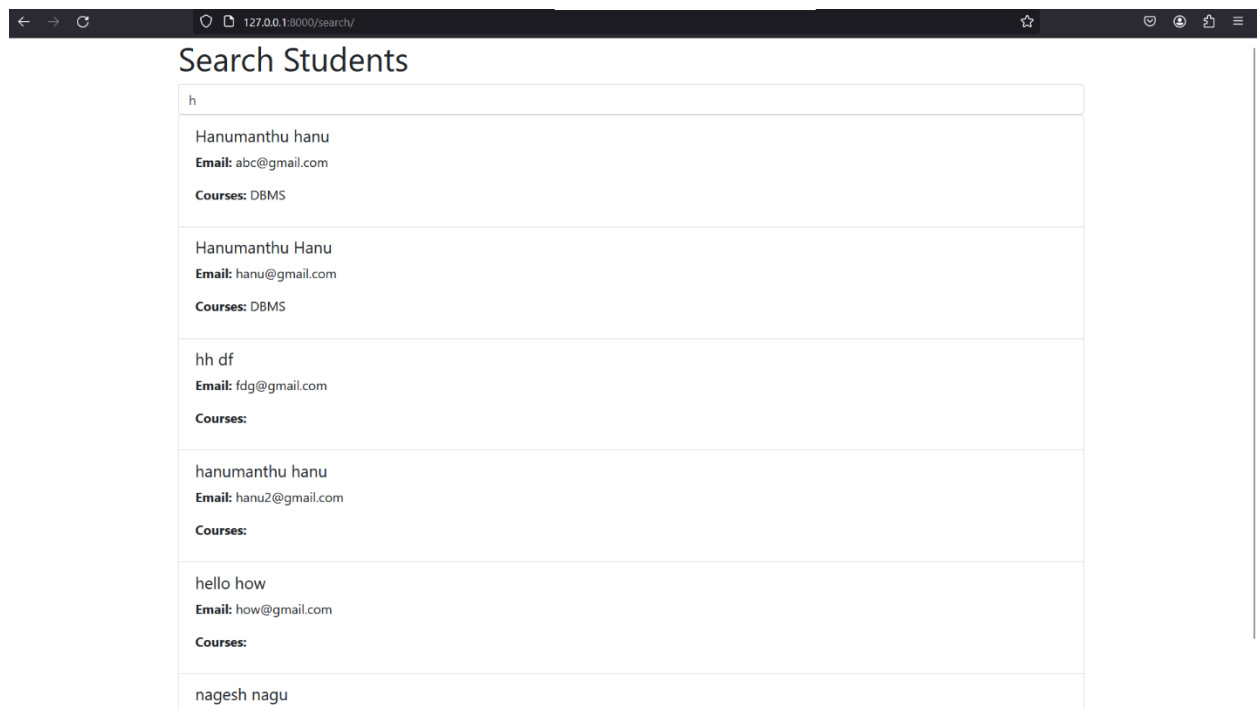


Fig: Search Screen