

RNS INSTITUTE OF TECHNOLOGY

(AICTE Approved, VTU Affiliated and NAAC 'A' Accredited)

UG Programs - CSE, ECE, ISE, EIE and EEE have been Accredited by NBA up to 30.06.2025

DR. VISHNUvardhan ROAD, CHANNASANDRA, RR NAGAR POST, BENGALURU – 560 098



ESTD : 2001

An Institute with a Difference

FULLSTACK DEVELOPMENT

For Sixth Semester B.E
[VTU/NEP, 2021 syllabus]

Subject Code – 21CS62

NAME : _____

BRANCH : _____

SECTION : _____

USN : _____

VISION AND MISSION OF INSTITUTION

Vision

Building RNSIT into a World Class Institution

Mission

To impart high quality education in Engineering, Technology and Management with a Difference, Enabling Students to Excel in their Career by

1. Attracting quality Students and preparing them with a strong foundation in fundamentals so as to achieve distinctions in various walks of life leading to outstanding contributions
2. Imparting value based, need based, choice based and skill based professional education to the aspiring youth and carving them into disciplined, World class Professionals with social responsibility
3. Promoting excellence in Teaching, Research and Consultancy that galvanizes academic consciousness among Faculty and Students
4. Exposing Students to emerging frontiers of knowledge in various domains and make them suitable for Industry, Entrepreneurship, Higher studies, and Research & Development
5. Providing freedom of action and choice for all the Stake holders with better visibility

VISION AND MISSION OF DEPARTMENT

Vision

Preparing Better Computer Professionals for a Real World

Mission

The Department of CSE will make every effort to promote an intellectual and ethical environment by

1. Imparting solid foundations and applied aspects in both Computer Science Theory and Programming practices
2. Providing training and encouraging R&D and Consultancy Services in frontier areas of Computer Science and Engineering with a Global outlook
3. Fostering the highest ideals of ethics, values and creating awareness of the role of Computing in Global Environment
4. Educating and preparing the graduates, highly sought after, productive, and well-respected for their work culture.
5. Supporting and inducing lifelong learning

FULLSTACK DEVELOPMENT LABORATORY (21CS62)

INTERTANAL EVALUATION SHEET

EVALUATION (MAX MARKS 20)

TEST A	REGULAR EVALUATION B	RECORD	TOTAL MARKS A+B+C
20	10	20	50

R1: REGULAR LAB EVALUATION WRITE UP RUBRIC (MAX MARKS 05)

Sl. No.	Parameters	Good	Average	Needs improvement
a.	Understanding of problem (2M)	Clear understanding of problem statement while designing and implementing (2)	Problem statement is understood clearly but few mistakes while designing and implementing ()	Problem statement is not clearly understood while designing (1)
b.	Write-up (1M)	Write the program neatly, demonstrating the concepts of Data Structure and algorithms using C Language (4)	Moderate understanding of algorithms (3)	Problem statement is not clearly understood while writing a program (1)
c.	Execution and Result (2M)	Program handles all possible conditions (2)	Demonstrates some of the conditions defined (1)	Documentation does not take care all conditions (1)

R2: REGULAR LAB EVALUATION VIVA RUBRIC (MAX MARKS 05)

Sl. No.	Parameter	Excellent	Good	Average	Needs Improvement
a.	Conceptual understanding (05 marks)	Answers 80% of the viva questions asked (5)	Answers 60% of the viva questions asked (4)	Answers 30% of the viva questions asked (3)	Unable to relate the concepts (2)

R3: REGULAR LAB PROGRAM EXECUTION RUBRIC (MAX MARKS 05)

Sl. No.	Parameters	Excellent	Good	Needs Improvement
a.	Design, implementation and demonstration (5M)	Program follows syntax and semantics of C programming language. Demonstrate the complete knowledge of the program written (5)	Program has few logical errors, moderately demonstrates all possible concepts implemented in Program (03)	Documentation on does not take care the given conditions (2)
b.	Result & documentation (5M)	All test cases are successful, all errors are debugged with own practical knowledge and clear documentation according to the guidelines (5)	Moderately debugs the experiment, few test case are unsuccessful and Partial documentation (4)	Test cases are not taken care, unable to debug the errors And no proper documentation (1)

R4: RECORD EVALUATION RUBRIC (MAX MARKS 05)

Sl. No.	Parameter	Excellent	Good	Average	Needs Improvement
a.	Documentation (05 M)	Meticulous record writing required Complete Program with comments and test cases as per the guidelines mentioned (05)	Write up contains Program and test cases with no comments (04)	Write up contains only program (03)	Program written with few mistakes. (02)

A. TEST/LAB INTERNALS MARKS(MAXMARKS10M)

TEST#	Write up 10M	Execution 30M	Viva 10M	Teacher Signature	Total 50M	Average 50M	Final 05
TEST-1							
TEST-2						50M	05M

B. REGULAR LAB EVALUATION (MAX MARKS 5)

Sl. #	Date of Execution	Additional programs	Write up (05)	Execn (05)	Viva (05)	Total 15	Teacher Signature
1							
2							
3							
4							
5							
6							
7							
8							
9							
10							
11							
12							
13							
Total Marks		—	—	15	—	15	—

Final Marks obtained from A. Test (05) + B. Regular Evaluation (10) + C. Record (05)	— 20	Lab in charge:
		HOD:

PREFACE

We have developed this comprehensive laboratory manual on **FULL STACK DEVELOPMENT 21CS62** with the primary objectives: Familiarize the students with computer hardware, concept of naming the program files, storing, compilation, execution and debugging.

This material provides the students an exposure to problem solving approaches and solution to number of problems using C programming Language. The problems discussed in this manual comprises of an algorithm, programming solution, alternative logic and extensive test cases. Viva questions, frequently appeared examination questions and practicing programming problems constitute an indispensable part of this material.

Our profound and sincere efforts will be fruitful only when students acquire the extensive knowledge by reading this manual and apply the concepts learnt apart from the requirements specified in Microcontroller and Embedded Systems Laboratory as prescribed by VTU, Belagavi.

Departments of CSE

ACKNOWLEDGMENT

A material of this scope would not have been possible without the contribution of many people. We express our sincere gratitude to Dr. R N Shetty, Chairman, RNS Group of Companies for his magnanimous support in all our endeavors.

We are grateful to Dr. Ramesh Babu H.S, Principal, RNSIT, Dr. P Kiran, HOD, CSE for extending their constant encouragement and support.

Our heartfelt thanks to Ms. Vidya Y, Ms. Anupama and Ms. Deepa Shetty for their unparalleled contribution throughout the preparation of this comprehensive manual. We also acknowledge our colleagues for their timely suggestions and unconditional support.

Departments of CSE

TABLE OF CONTENTS

SL. NO.	CONTENTS	PAGE NO.	CO'S
	FULLSTACK DEVELOPMENT		
	LABORATORY PROGRAMS		
1	Installation of Python, Django and Visual Studio code editors can be demonstrated.		
2	Creation of virtual environment, Django project and App should be demonstrated.		
3	Develop a Django app that displays current date and time in server.		
4	Develop a Django app that displays date and time four hours ahead and four hours before as an offset of current date and time in server.		
5	Develop a simple Django app that displays an unordered list of fruits and ordered list of selected students for an event		
6	Develop a layout.html with a suitable header (containing navigation menu) and footer with copyright and developer information. Inherit this layout.html and create 3 additional pages: contact us, About Us and Home page of any website.		
7	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.		
8	For student and course models created in Lab experiment for Module2, register admin interfaces, perform migrations and illustrate data entry through admin forms.		
9	Develop a model form for student that contains his topic chosen for project, languages used and duration with a model called project.		
10	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.		
11	Develop example Django app that performs CSV and PDF generation for any models created in previous laboratory component.		
12	Develop a registration page for student enrolment as done in Module 2 but without page refresh using AJAX.		
13	Develop a search application in Django using AJAX that displays courses enrolled by a student being searched.		

SYALLABUS
SEMESTER – VI
FULLSTACK DEVELOPMENT LABORATORY

Course Code – 21CS62	CIE Marks - 50
Number of Contact Hours/Week -3:0:2:0	SEE Marks - 50
Total Number of Lab Contact Hours – 40t+20P	Exam Hours - 03

Course outcome (Course Skill Set)

At the end of the course the student will be able to:

- CO 1. Understand the working of MVT based full stack web development with Django.
- CO 2. Designing of Models and Forms for rapid development of web pages.
- CO 3. Analyze the role of Template Inheritance and Generic views for developing full stack web applications.
- CO 4. Apply the Django framework libraries to render nonHTML contents like CSV and PDF.
- CO 5. Perform jQuery-based AJAX integration to Django Apps to build responsive full stack web applications

Assessment Details (both CIE and SEE)

The weightage of Continuous Internal Evaluation (CIE) is 50% and for Semester End Exam (SEE) is 50%. The minimum passing mark for the CIE is 40% of the maximum marks (20 marks). A student shall be deemed to have satisfied the academic requirements and earned the credits allotted to each subject/ course if the student secures not less than 35% (18 Marks out of 50) in the semester-end examination (SEE), and a minimum of 40% (40 marks out of 100) in the sum total of the CIE (Continuous Internal Evaluation) and SEE (Semester End Examination) taken together.

Continuous Internal Evaluation:

Three Unit Tests each of **20 Marks (duration 01 hour)**

- 1. First test at the end of 5th week of the semester
- 2. Second test at the end of the 10th week of the semester
- 3. Third test at the end of the 15th week of the semester

Two assignments each of **10 Marks**

- 4. First assignment at the end of 4th week of the semester
- 5. Second assignment at the end of 9th week of the semester

Practical Sessions need to be assessed by appropriate rubrics and viva-voce method. This will contribute to **20 marks**.

- Rubrics for each Experiment taken average for all Lab components – 15 Marks.
- Viva-Voce– 5 Marks (more emphasized on demonstration topics)

The sum of three tests, two assignments, and practical sessions will be out of 100 marks and will be scaled down to 50 marks (to

have a less stressed CIE, the portion of the syllabus should not be common /repeated for any of the methods of the CIE. Each method of CIE should have a different syllabus portion of the course)

Laboratory Components

1. Installation of Python, Django and Visual Studio code editors can be demonstrated.
2. Creation of virtual environment, Django project and App should be demonstrated.
3. Develop a Django app that displays current date and time in server.
4. Develop a Django app that displays date and time four hours ahead and four hours before as an offset of current date and time in server.
5. Develop a simple Django app that displays an unordered list of fruits and ordered list of selected students for an event.
6. Develop a layout.html with a suitable header (containing navigation menu) and footer with copyright and developer information. Inherit this layout.html and create 3 additional pages: contact us, About Us and Home page of any website.
7. 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.
8. For student and course models created in Lab experiment for Module2, register admin interfaces, perform migrations and illustrate data entry through admin forms.
9. Develop a model form for student that contains his topic chosen for project, languages used and duration with a model called project.
10. 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.
11. Develop example Django app that performs CSV and PDF generation for any models created in previous laboratory component.
12. Develop a registration page for student enrolment as done in Module 2 but without page refresh using AJAX.
13. Develop a search application in Django using AJAX that displays courses enrolled by a student being searched.

Experiment-01

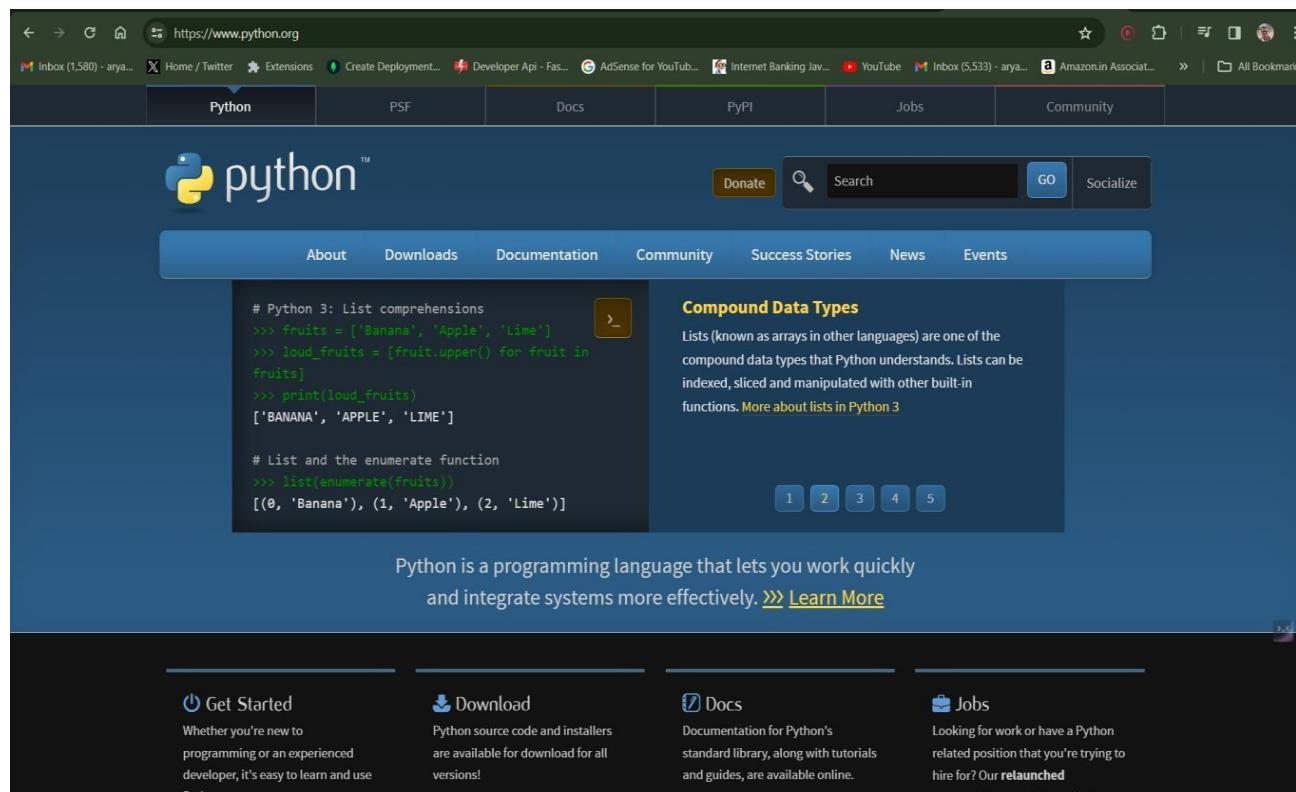
Installation of Python, Django and Visual Studio code editors can be demonstrated.

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 of [Python](#).



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

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

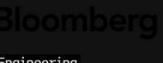
Python version	Maintenance status	First released	End of support	Release schedule
3.13	prerelease	2024-10-01 (planned)	2029-10	PEP 719
3.12	bugfix	2023-10-02	2028-10	PEP 693
3.11	bugfix	2022-10-24	2027-10	PEP 664
3.10	bugfix	2021-10-04	2026-10	PEP 610

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.

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

[View older releases](#)

Sponsors
Visionary sponsors help to host Python downloads.

[Licenses](#) [Sources](#) [Alternative](#) [History](#)

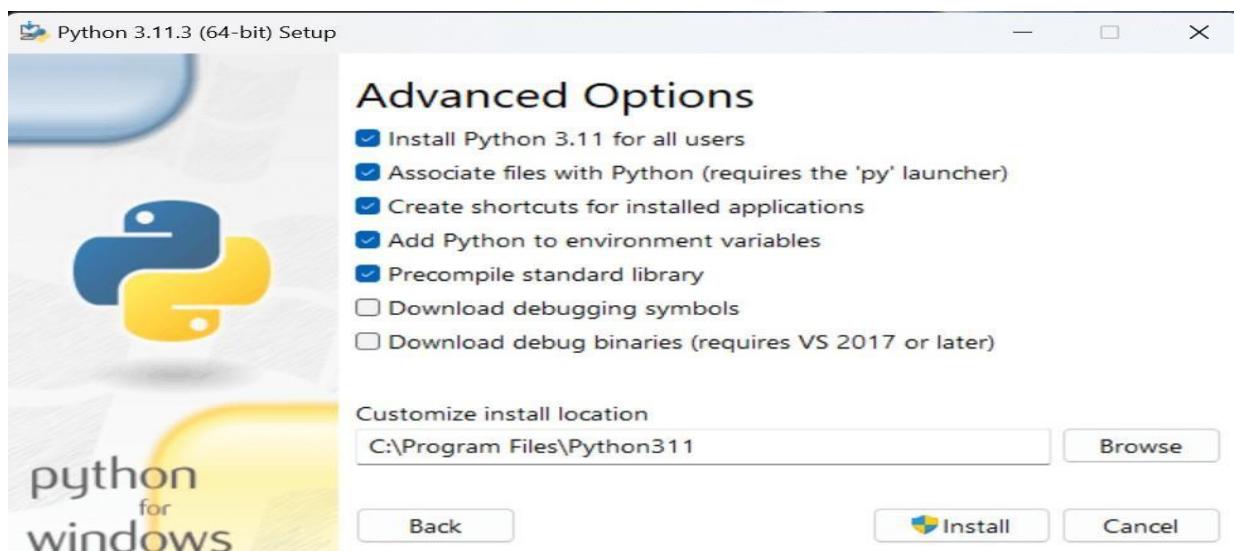
Step - 2: Click on the Install Now

Double-click the executable file, which is downloaded.

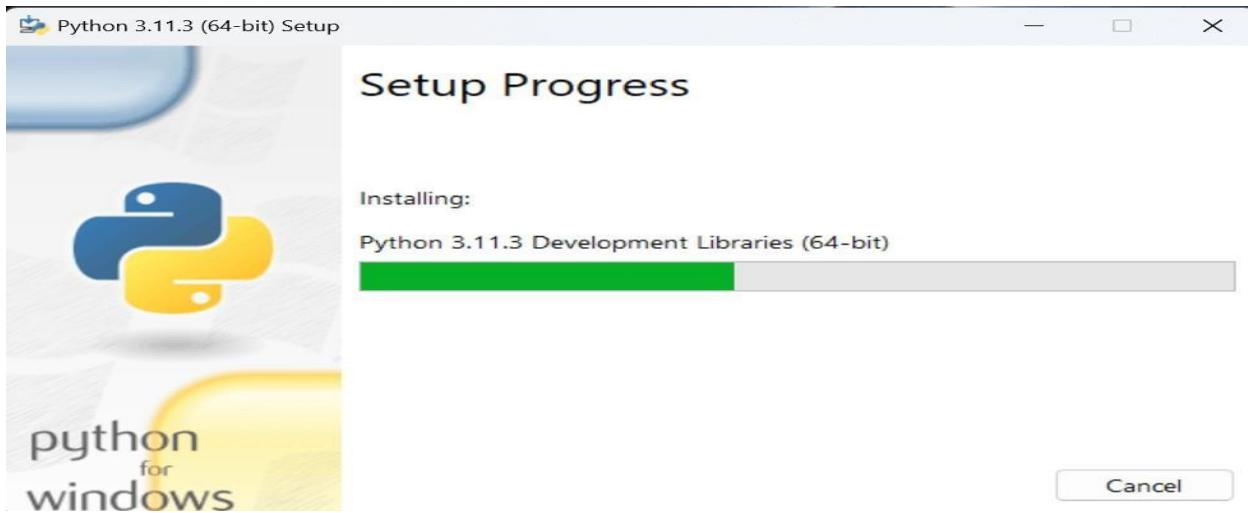


The following window will open. Click on the Add Path check box, it will set the Python path automatically.

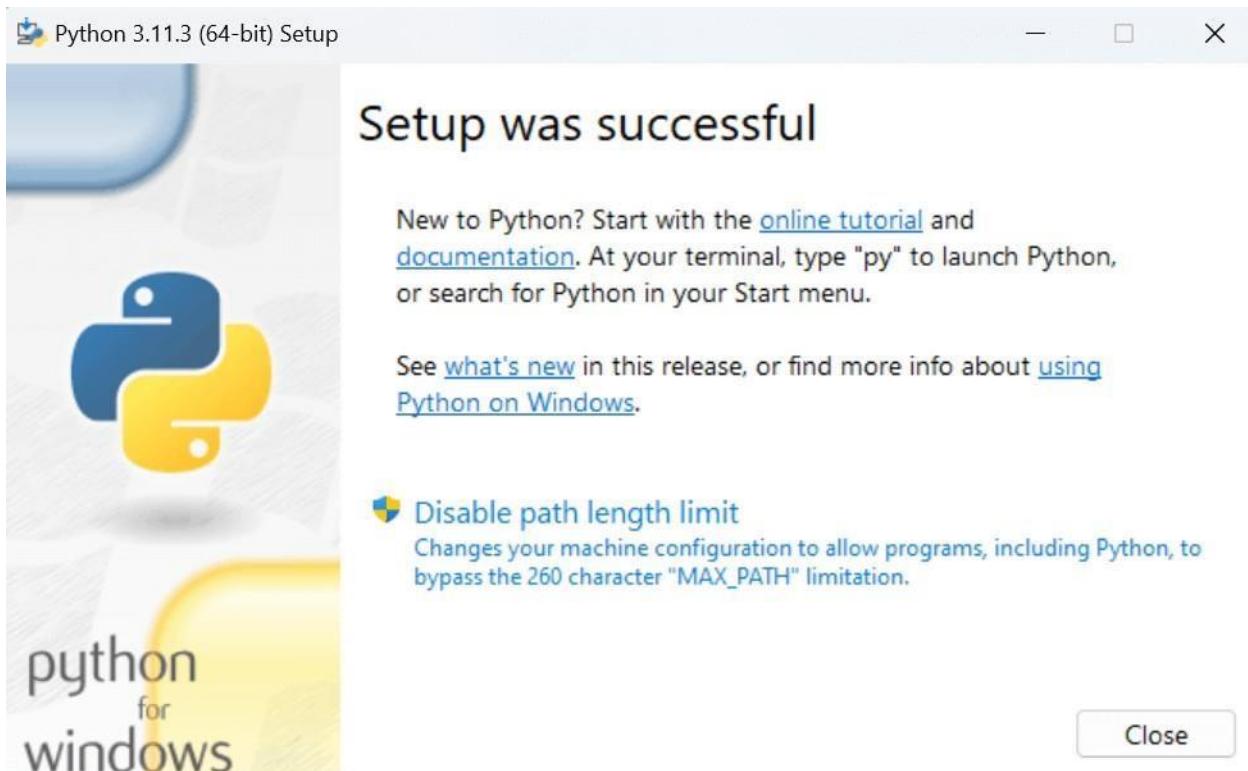
Now, Select Customize installation and proceed. We can also click on the customize installation to choose desired location and features. Other important thing is installing launcher for the all user must be checked.



Step - 3 Installation in Process



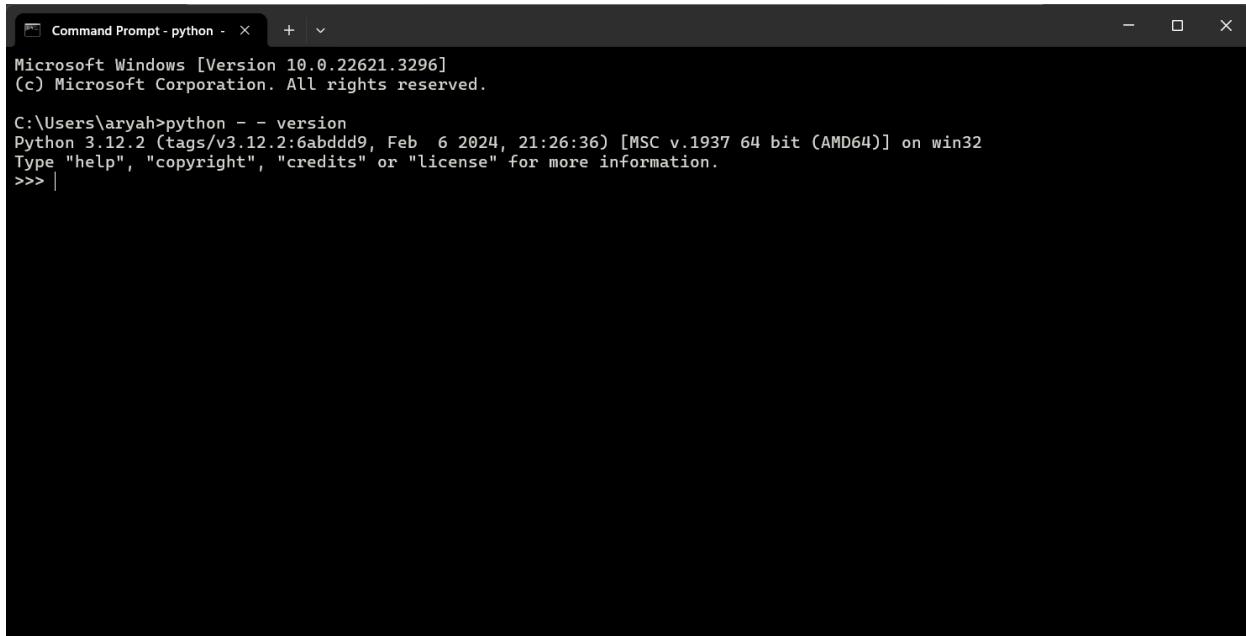
The set up is in progress. All the python libraries, packages, and other python default files will be installed in our system. Once the installation is successful, the following page will appear saying " Setup was successful ".



Step - 4: Verifying the Python Installation

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

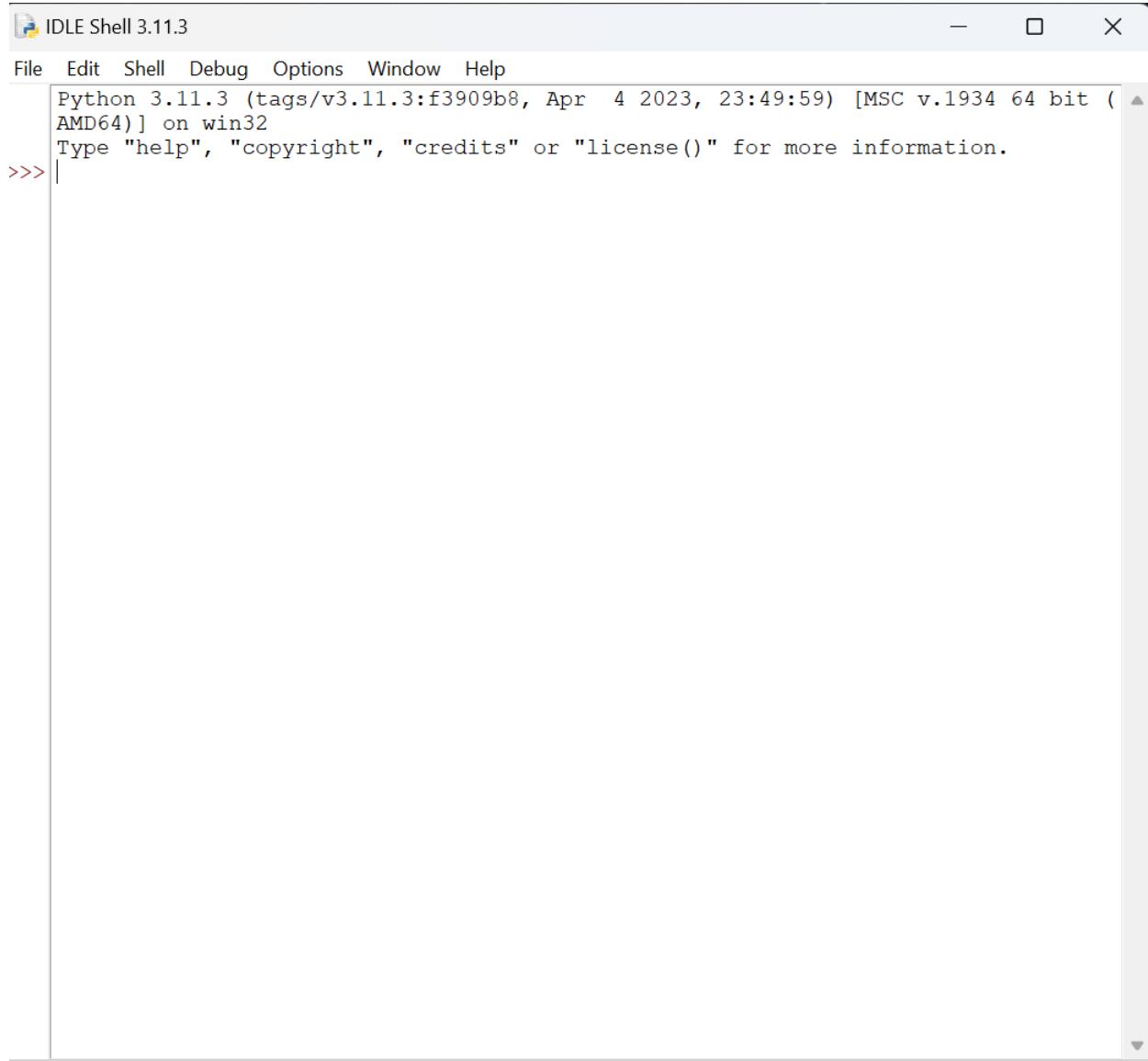
- Go to "Start" button, and search " cmd ".
- Then type, " python -- version ".
- If python is successfully installed, then we can see the version of the python installed.
- If not installed, then it will print the error as " 'python' is not recognized as an internal or external command, operable program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



The screenshot shows a Windows Command Prompt window titled "Command Prompt - python". The window title bar also includes the text "python - x + v". The window content displays the following text:
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 window displays the Python 3.11.3 version information: "Python 3.11.3 (tags/v3.11.3:f3909b8, Apr 4 2023, 23:49:59) [MSC v.1934 64 bit (AMD64)] on win32". It also shows the message "Type "help", "copyright", "credits" or "license()" for more information." followed by a command prompt ">>>".

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

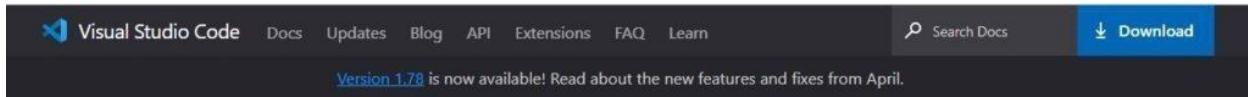
Step – 1: Open Google and type Visual Studio Code download in the search bar.



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

A screenshot of a Google search results page. The search bar at the top contains the query "visual studio code download". Below the search bar, there are several search filters: "All" (selected), "Shopping", "Videos", "Books", "Images", "More", and "Tools". The text "About 33,90,00,000 results (0.31 seconds)" is displayed. The first search result is highlighted with a yellow box. It shows a thumbnail of the Visual Studio logo, the text "Visual Studio", the URL "https://code.visualstudio.com/download", and three vertical dots. The result title is "Download Visual Studio Code - Mac, Linux, Windows". A snippet below the title reads: "Visual Studio Code is free and available on your favorite platform - Linux, macOS, and Windows. Download Visual Studio Code to experience a redefined code ...". The second search result, which is not highlighted, shows a Microsoft logo, the text "Microsoft", the URL "https://visualstudio.microsoft.com/downloads", and three vertical dots. The result title is "Download Visual Studio Tools - Install Free for Windows, Mac ...". A snippet below the title reads: "5 days ago — Download Visual Studio IDE or VS Code for free. Try out Visual Studio Professional or Enterprise editions on Windows, Mac.".

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



The screenshot shows the official Visual Studio Code website. At the top, there's a navigation bar with links for Visual Studio Code, Docs, Updates, Blog, API, Extensions, FAQ, and Learn. To the right of the navigation is a search bar labeled "Search Docs" and a blue "Download" button. Below the navigation, a message says "Version 1.78 is now available! Read about the new features and fixes from April." The main content area is titled "Download Visual Studio Code" and includes the subtext "Free and built on open source. Integrated Git, debugging and extensions." Below this, there are download links for different operating systems: Windows (Windows 8, 10, 11), Linux (Ubuntu, Debian, Fedora, SUSE), and macOS (macOS 10.11+). The Windows link is highlighted with a red box.

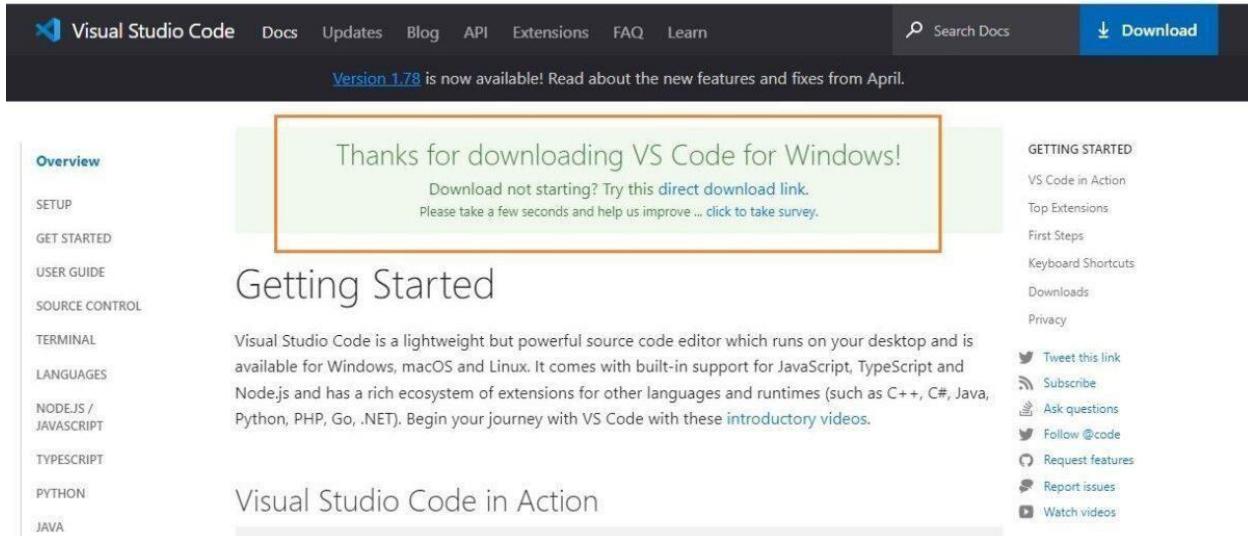
Download Visual Studio Code

Free and built on open source. Integrated Git, debugging and extensions.

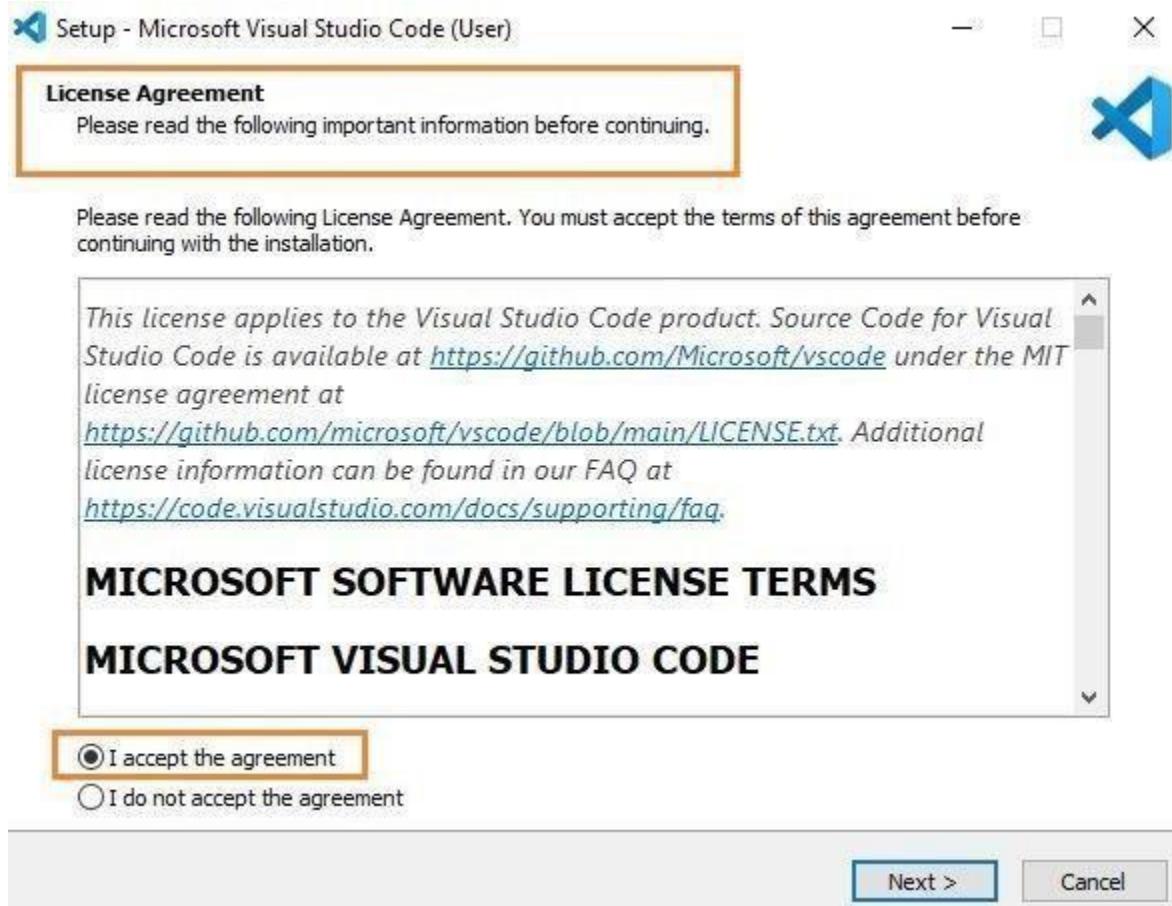


Step – 4: The file will be downloaded onto your system. Open the file and then click on Install.

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

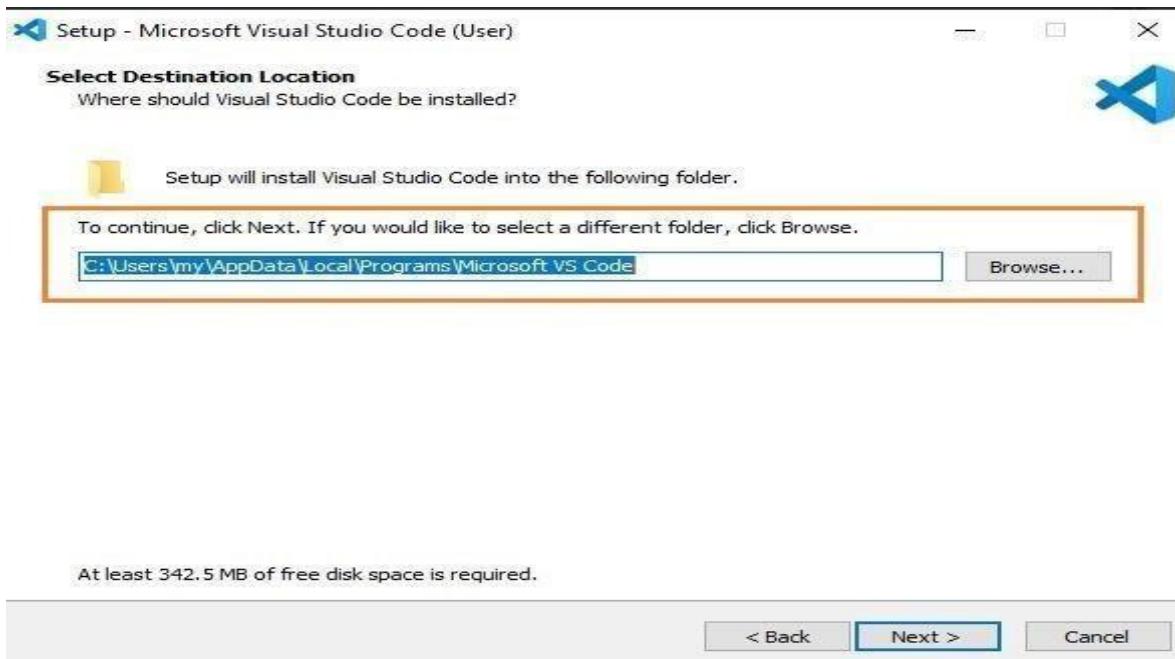


The screenshot shows the Visual Studio Code website after a download has been completed. The top navigation bar and search bar are visible. A central message box says "Thanks for downloading VS Code for Windows!" with a link to a direct download if the download didn't start. Below this, there's a "Getting Started" section with a brief introduction to VS Code and a "Visual Studio Code in Action" section. On the left, there's a sidebar with links for Overview, SETUP, GET STARTED, USER GUIDE, SOURCE CONTROL, TERMINAL, LANGUAGES, NODEJS / JAVASCRIPT, TYPESCRIPT, PYTHON, and JAVA. On the right, there's a "GETTING STARTED" sidebar with links for VS Code in Action, Top Extensions, First Steps, Keyboard Shortcuts, Downloads, Privacy, and social media links for Twitter, YouTube, and GitHub.

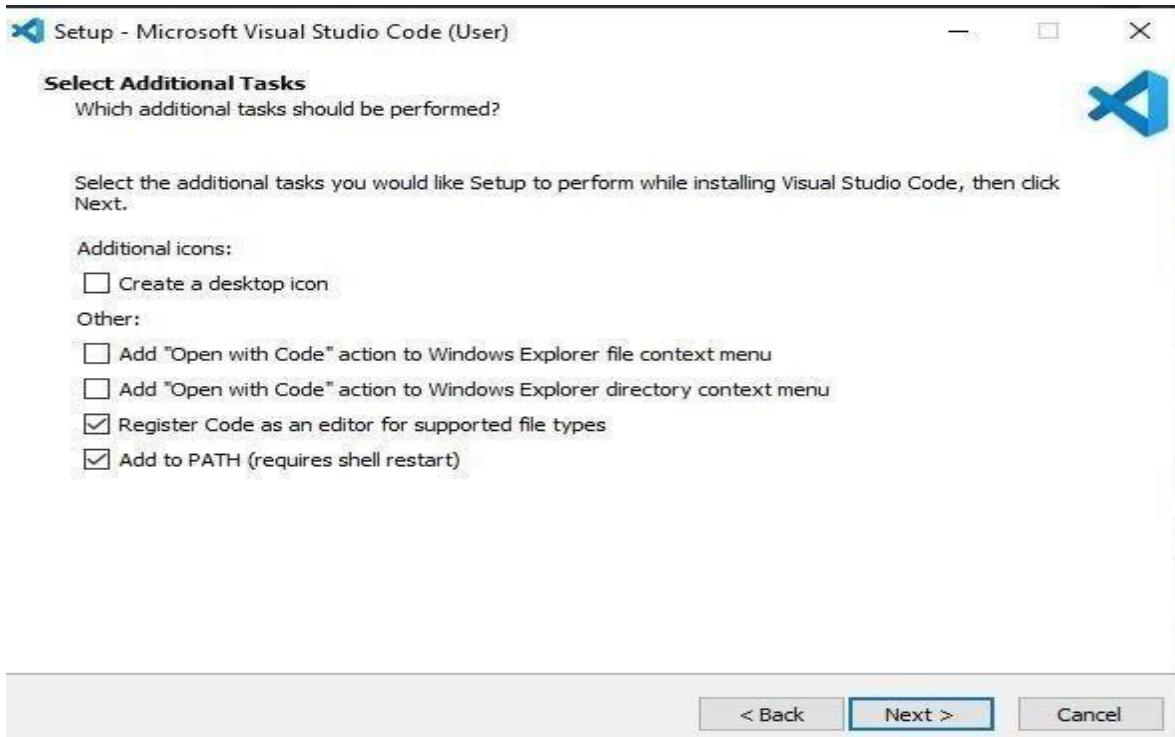
Step – 5: Now accept the license agreement.

Step – 6: Then it prompts for the file location, where you want to save the VS Code file.

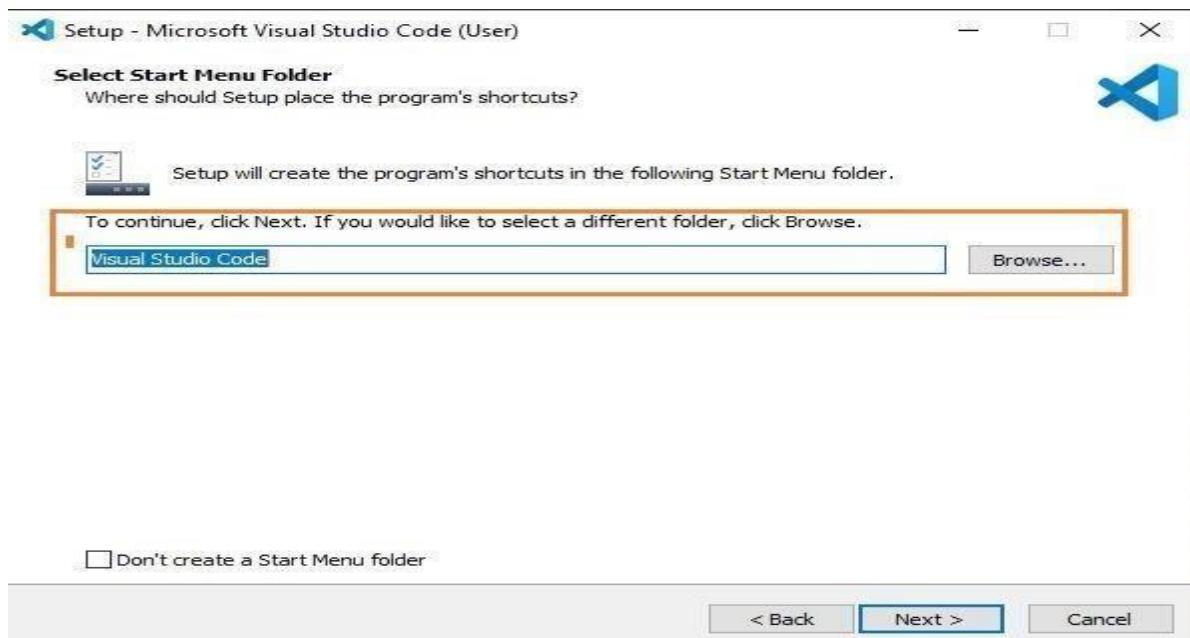
Browse the location and then click on Next.



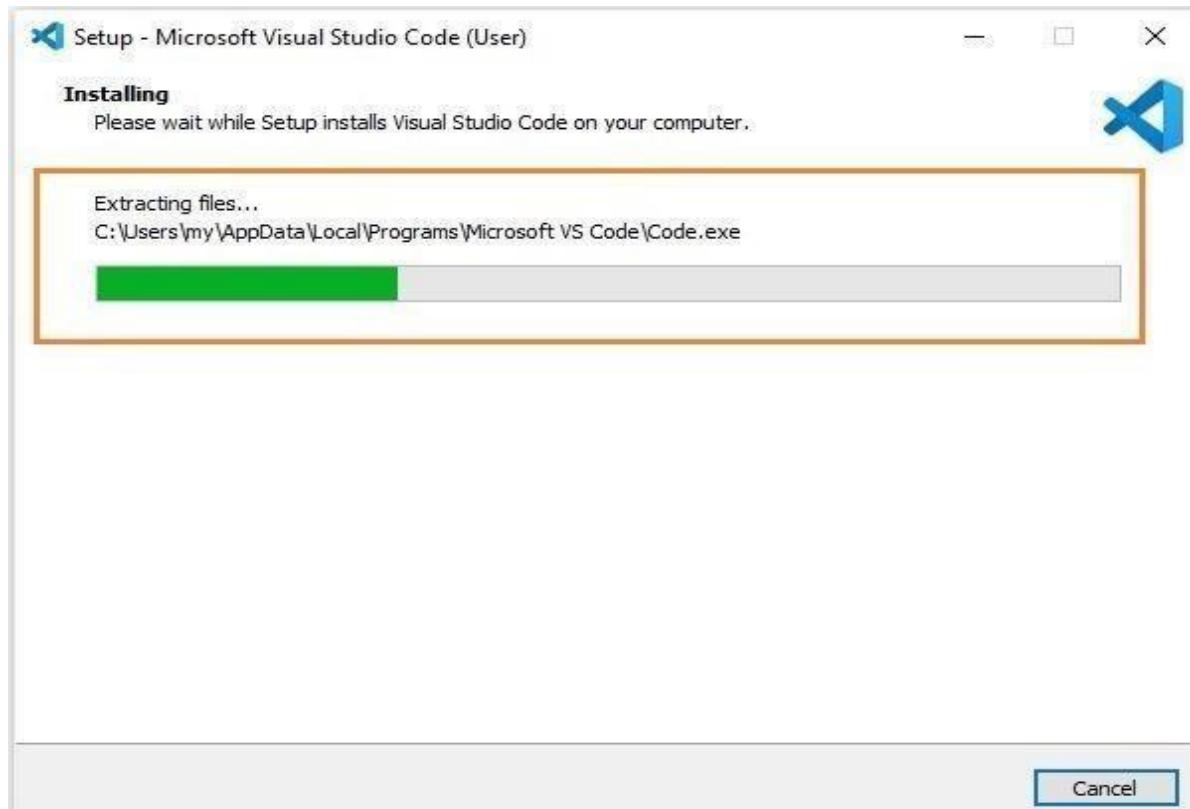
Step – 7: Next, you see the prompt for the additional task which we want the VS Code to perform. At this step, choose the default settings and then click on next.



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



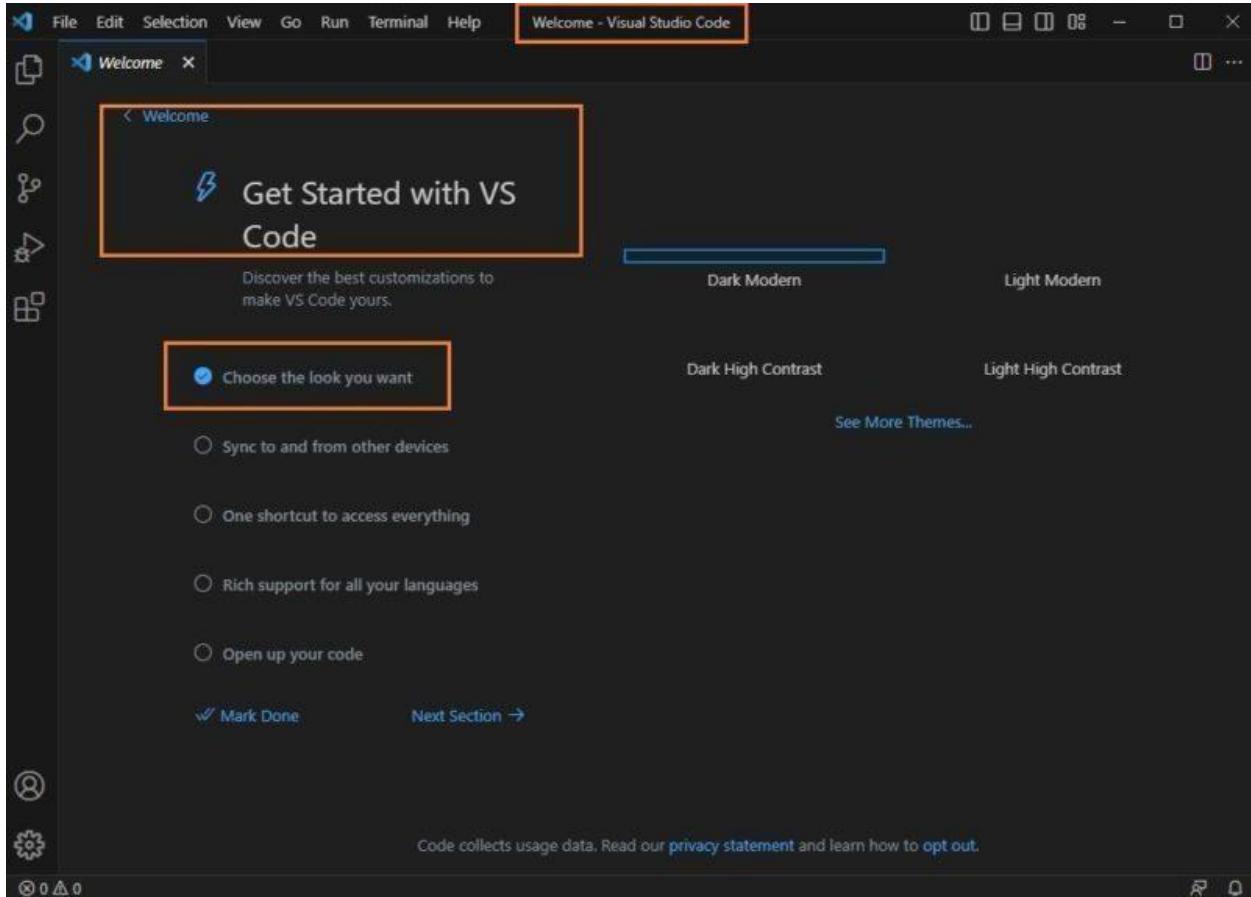
Step – 9: The installation of VS Code will now begin.



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



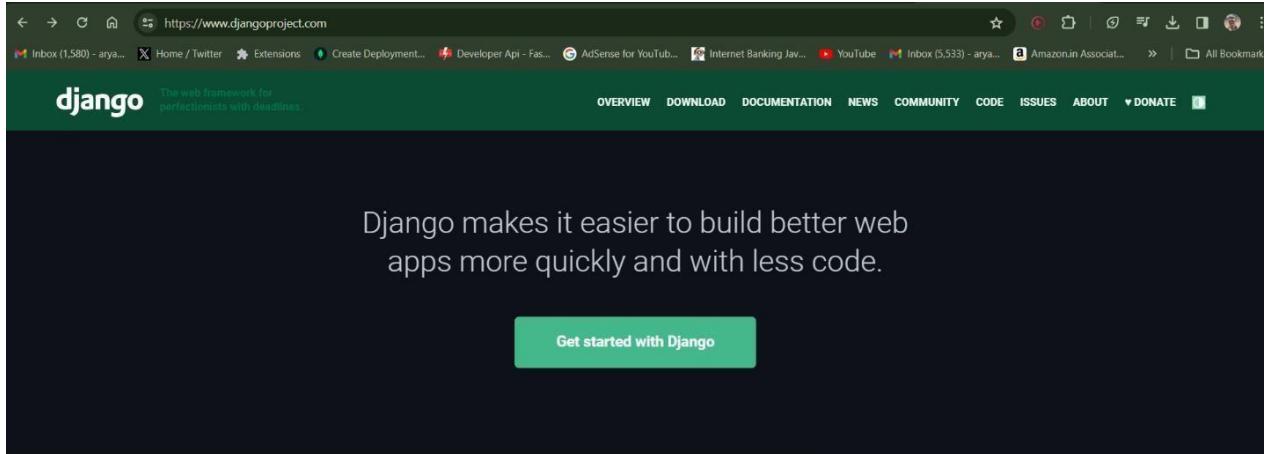
Step – 11: Now that VS Code installation is successful, the page appears as below:



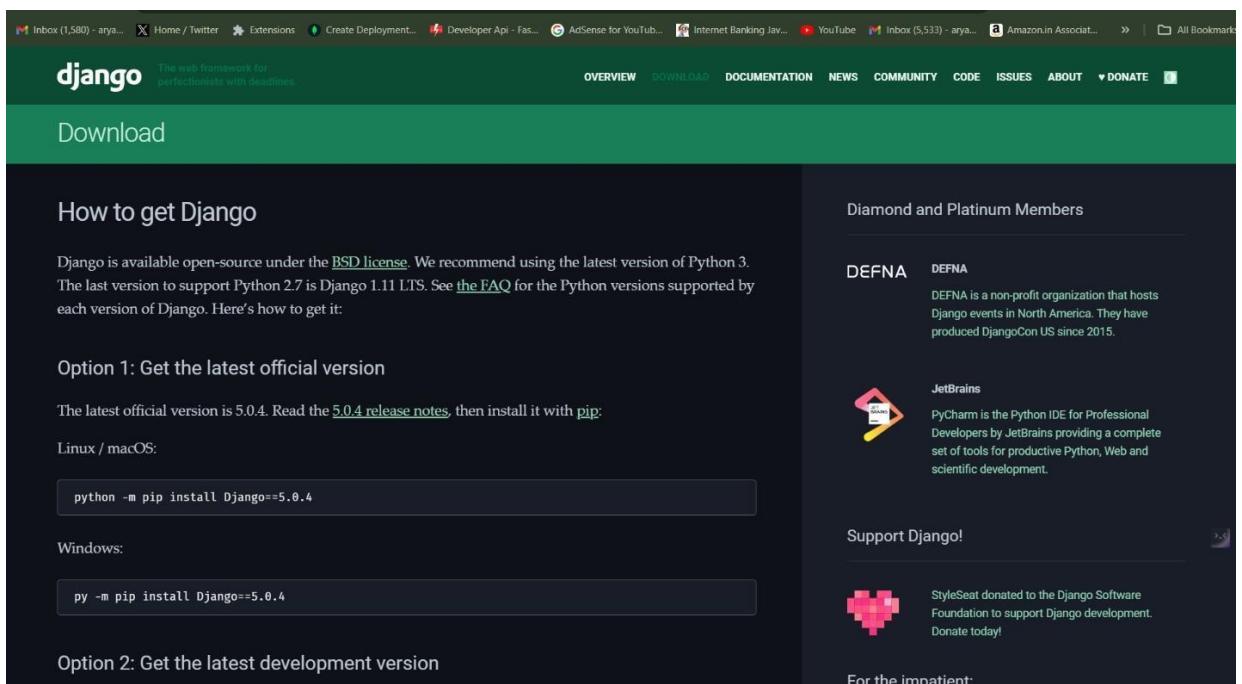
We can change the look as per our choice and continue working on it.

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.



The screenshot shows the main Django project page. At the top, there's a navigation bar with links like Overview, Download, Documentation, News, Community, Code, Issues, About, and a Donate button. Below the navigation, a large green button says "Get started with Django". The main content area features a dark background with white text: "Django makes it easier to build better web apps more quickly and with less code." Below this, there's a section titled "Meet Django" with a sub-section "Ridiculously fast." featuring a lightning bolt icon. To the right, there's a "Download latest release: 5.0.4" button and a "Support Django!" section with a pink heart icon and a donation message from Happy Herbivore Inc.



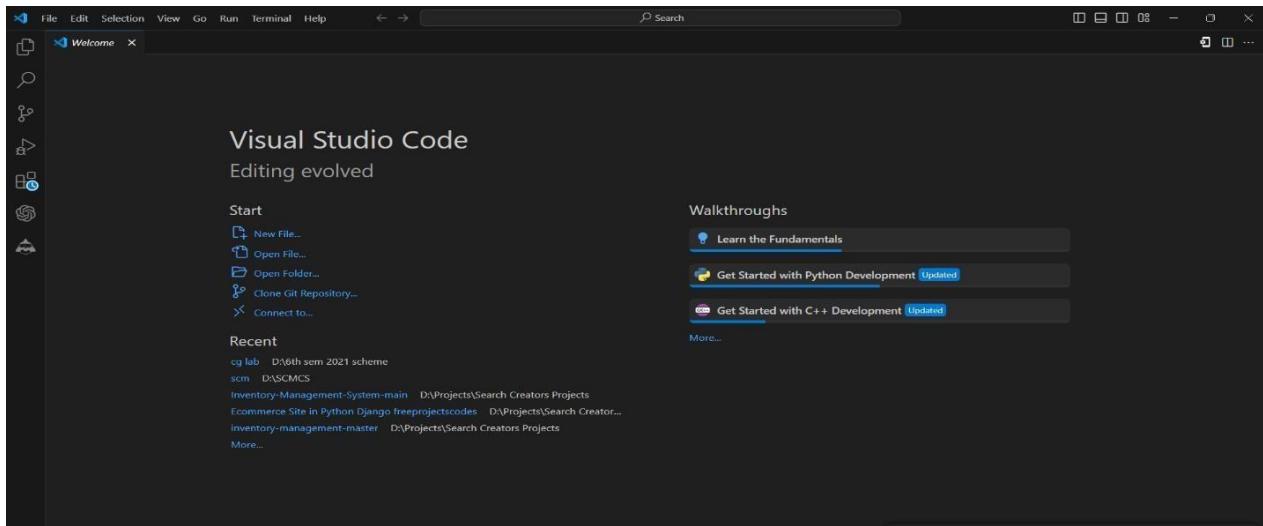
The screenshot shows the same Django project page, but the main content area is now focused on the "Download" section. It includes a "How to get Django" section with information about the BSD license and supported Python versions, and two "Option" sections: "Option 1: Get the latest official version" and "Option 2: Get the latest development version". The "Option 1" section provides instructions for Linux/macOS and Windows, with command-line examples: "python -m pip install Django==5.0.4" for Linux/macOS and "py -m pip install Django==5.0.4" for Windows. To the right, there are sections for "Diamond and Platinum Members" (listing DEFNA and JetBrains), a "Support Django!" section with a pink heart icon and a donation message from StyleSeat, and a "For the impatient:" section.

Experiment-02

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

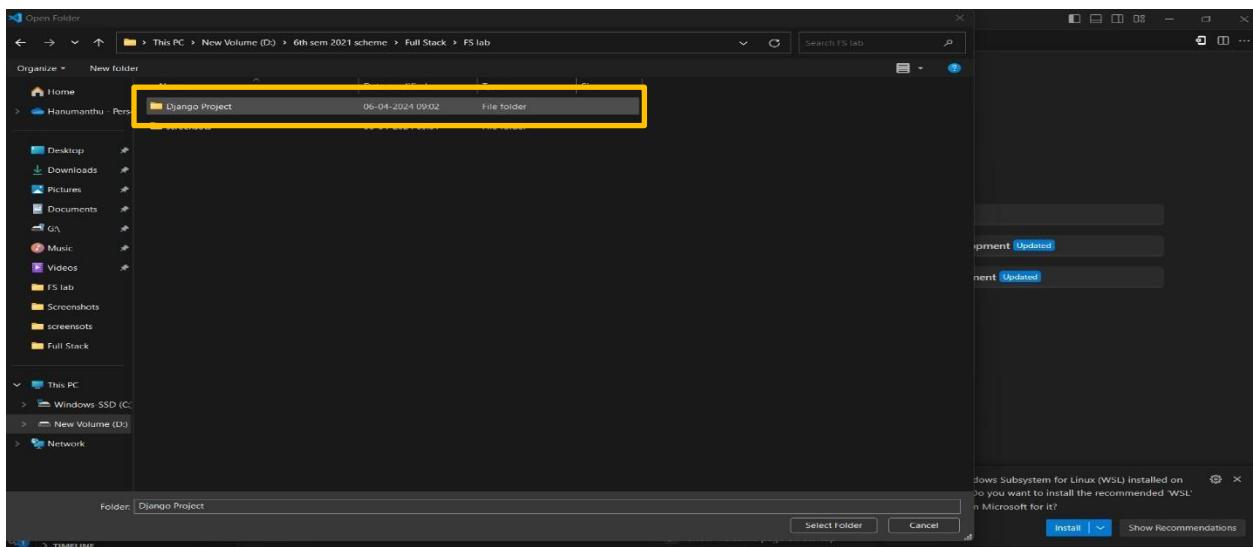
Here are the steps you can follow to create virtual environment: Steps to Create Virtual Environment, Django Project and App

Step-01: Open Visual Studio Code



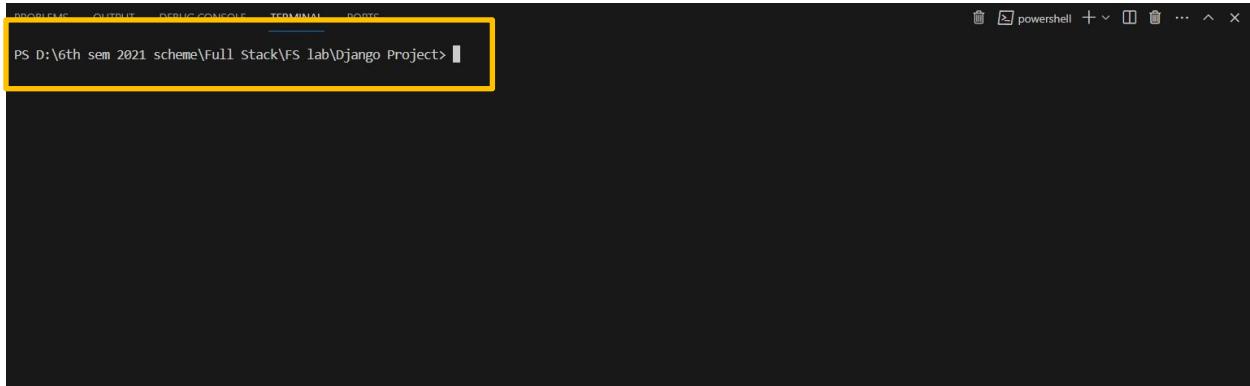
Step-02: Create a new folder for your project

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

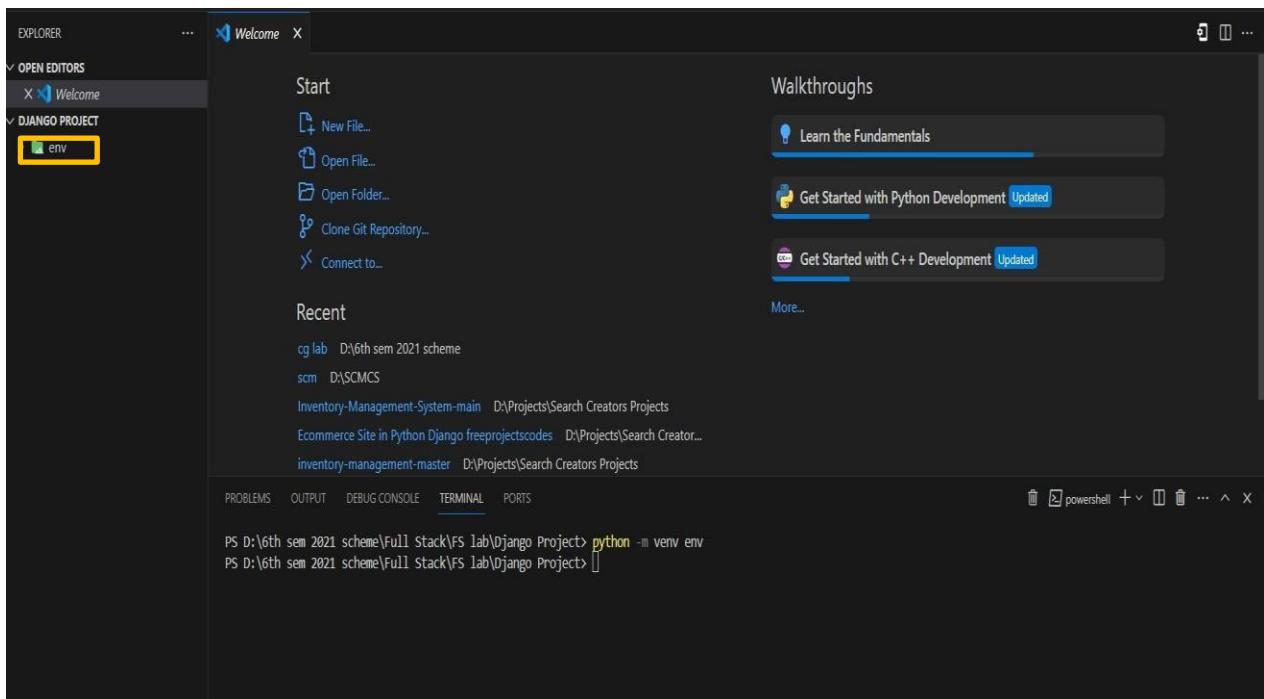
In VS Code, go to **View > Terminal** or use the keyboard shortcut **Ctrl+``** (Windows/Linux) or **Cmd+``** (macOS) to open the integrated terminal.



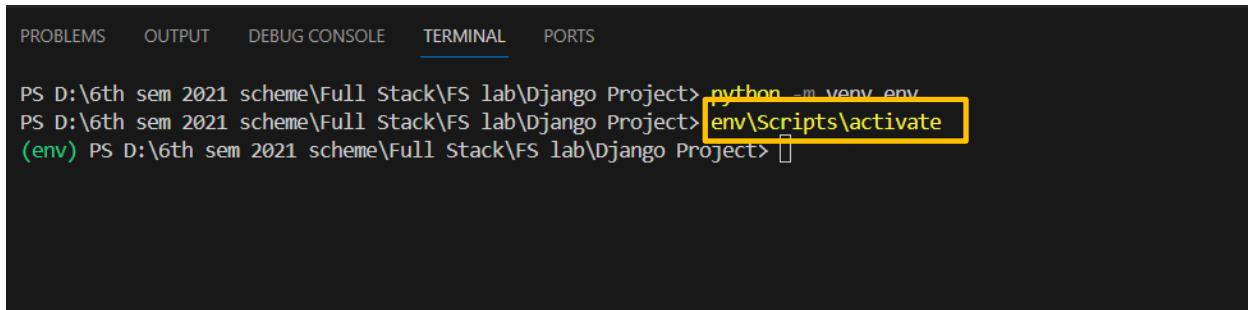
Step-03: Create a virtual environment

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

```
python -m venv env
```



Step-04: Activate the virtual environment

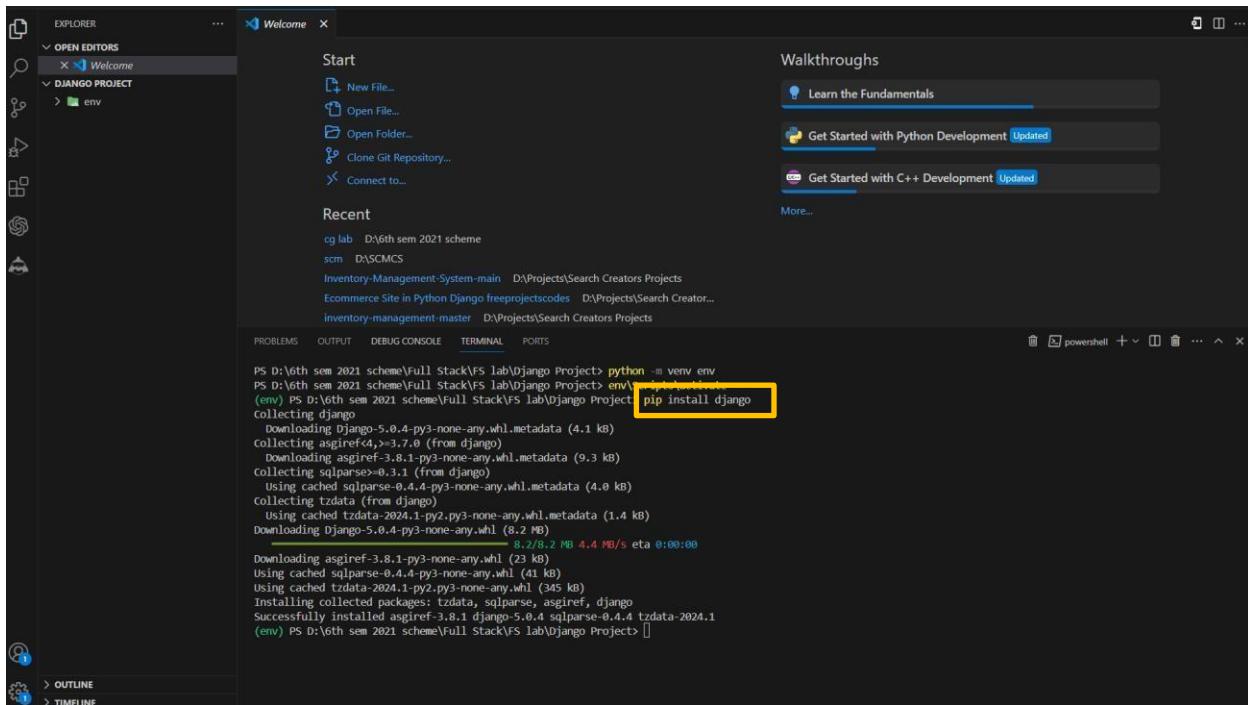


```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

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 OPEN EDITORS Welcome ...
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

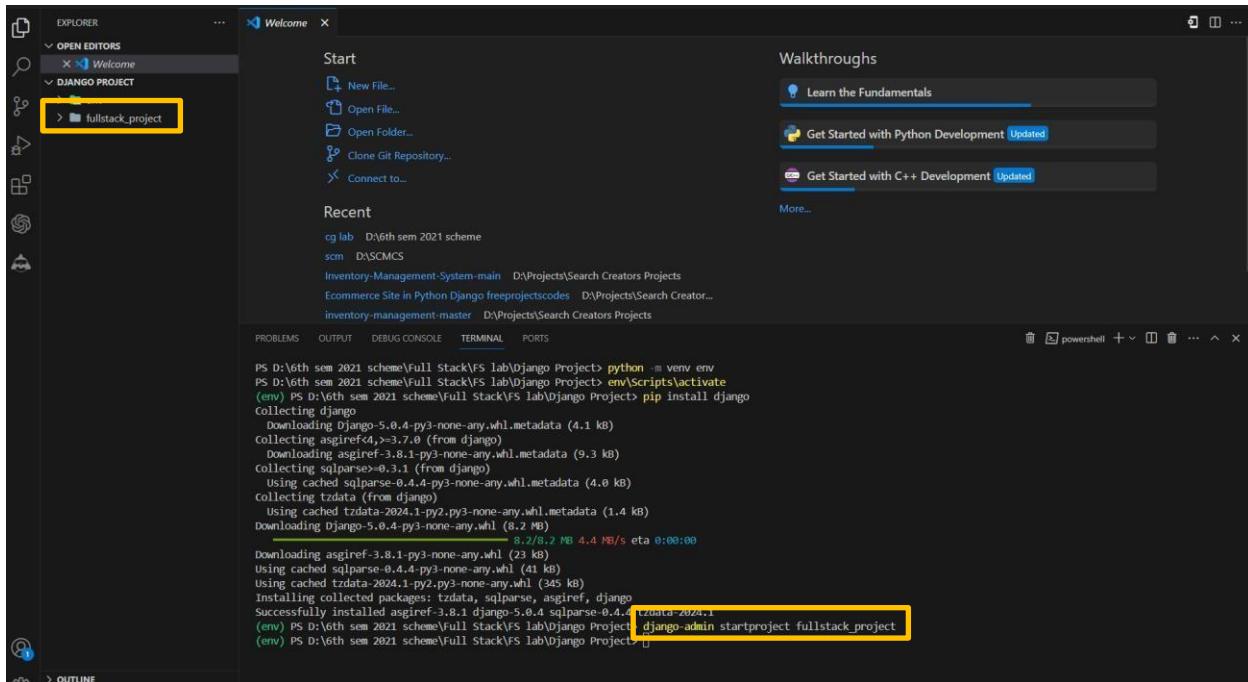
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
powershell + ×

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 aspires<4,>=3.7.0 (from django)
    Downloading aspire-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)
  Downloading aspire-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, aspire, django
  Successfully installed aspire-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]



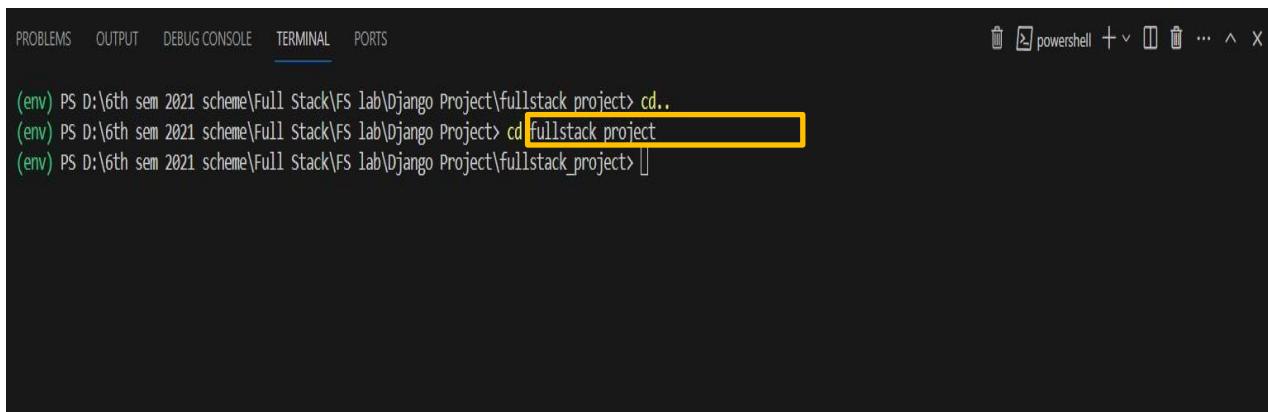
The screenshot shows the VS Code interface with the terminal tab active. The terminal window displays the command `django-admin startproject fullstack_project` being run. The output shows the project structure being created, including the `fullstack_project` directory and its subfolders like `__init__.py`, `settings.py`, `urls.py`, and `wsgi.py`. The terminal also shows the activation of a Python environment and the installation of Django via pip.

```
ps D:\6th sem 2021 scheme\full Stack\Fs lab\ Django Project> python -m venv env
ps D:\6th sem 2021 scheme\full Stack\Fs lab\ Django Project> env\Scripts\activate
(env) PS D:\6th sem 2021 scheme\full Stack\Fs 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
(env) PS D:\6th sem 2021 scheme\full Stack\Fs lab\ Django Project> django-admin startproject fullstack_project
(env) PS D:\6th sem 2021 scheme\full Stack\Fs lab\ Django Project>
```

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]



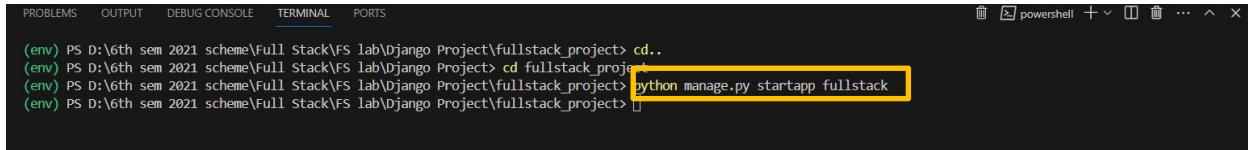
The screenshot shows the VS Code terminal window with the command `cd myproject` being run. The output shows the user navigating to the `myproject` directory within the `full Stack\Fs lab\ Django Project` folder. The terminal then displays the creation of a new Django app named `fullstack`.

```
(env) PS D:\6th sem 2021 scheme\full Stack\Fs lab\ Django Project> cd..
(env) PS D:\6th sem 2021 scheme\full Stack\Fs lab\ Django Project> cd fullstack
(env) PS D:\6th sem 2021 scheme\full Stack\Fs lab\ Django Project\fullstack> []
```

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.]

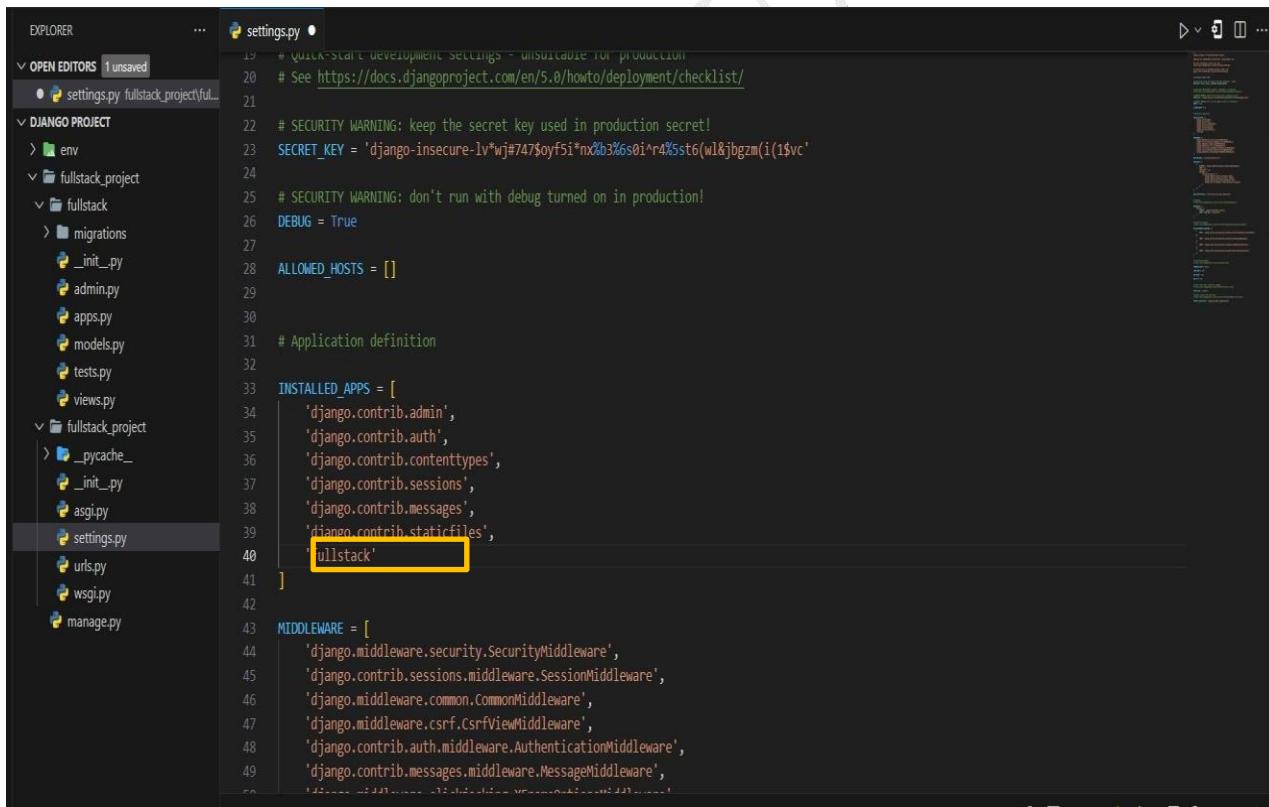


```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
(env) PS D:\6th sem 2021 scheme\Full Stack\FS lab\Django Project\fullstack_project> cd..
(env) PS D:\6th sem 2021 scheme\Full Stack\FS lab\Django Project> cd fullstack project
(env) PS D:\6th sem 2021 scheme\Full Stack\FS lab\Django Project\fullstack_project> 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

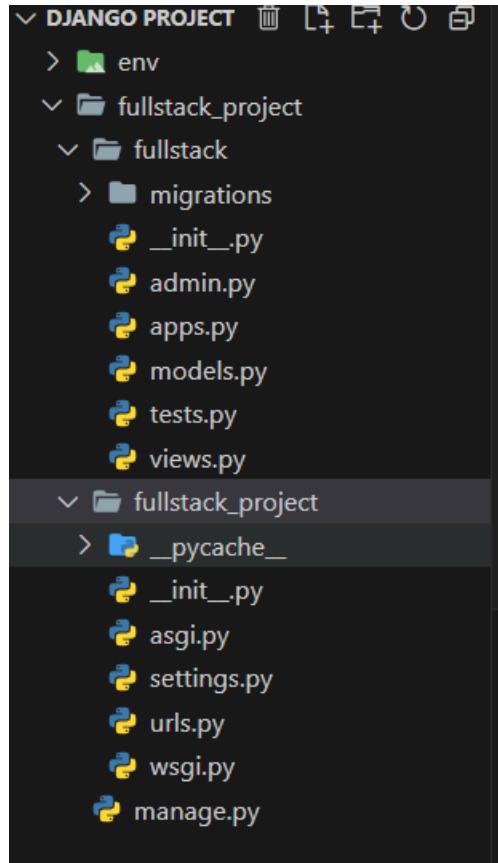
In the VS Code file explorer, locate the settings.py file (**usually located in the myproject directory**) and open it.

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



```
EXPLORER OPEN EDITORS 1 unsaved ... settings.py ●
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$oyf5i*nxb3%6s01^r4%5st6(wl&jbgzm(i1$vc'
24
25 # SECURITY WARNING: don't run with debug turned on in production!
26 DEBUG = True
27
28 ALLOWED_HOSTS = []
29
30
31 # Application definition
32
33 INSTALLED_APPS = [
34     'django.contrib.admin',
35     'django.contrib.auth',
36     'django.contrib.contenttypes',
37     'django.contrib.sessions',
38     'django.contrib.messages',
39     'django.contrib.staticfiles',
40     'fullstack'
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 ]
```

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)

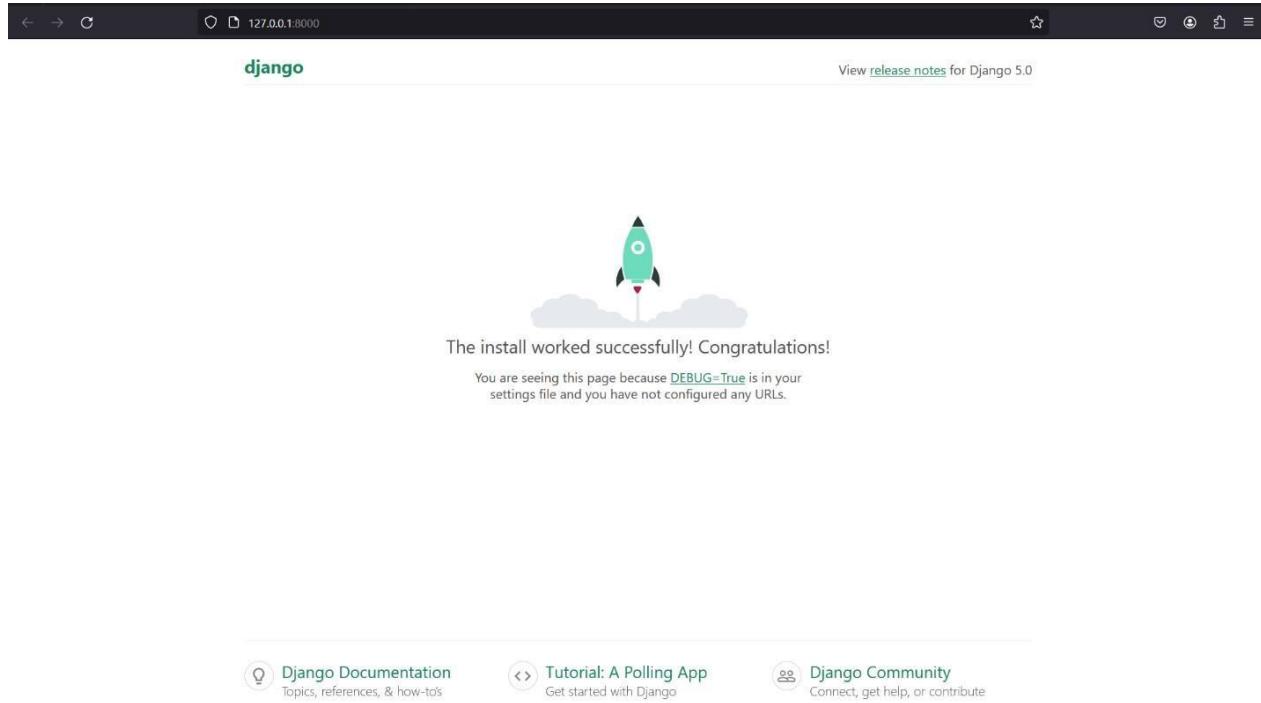
```
(env) PS D:\6th sem 2021 scheme\Full Stack\FS lab\ Django Project\fullstack_project> python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

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



Experiment-03

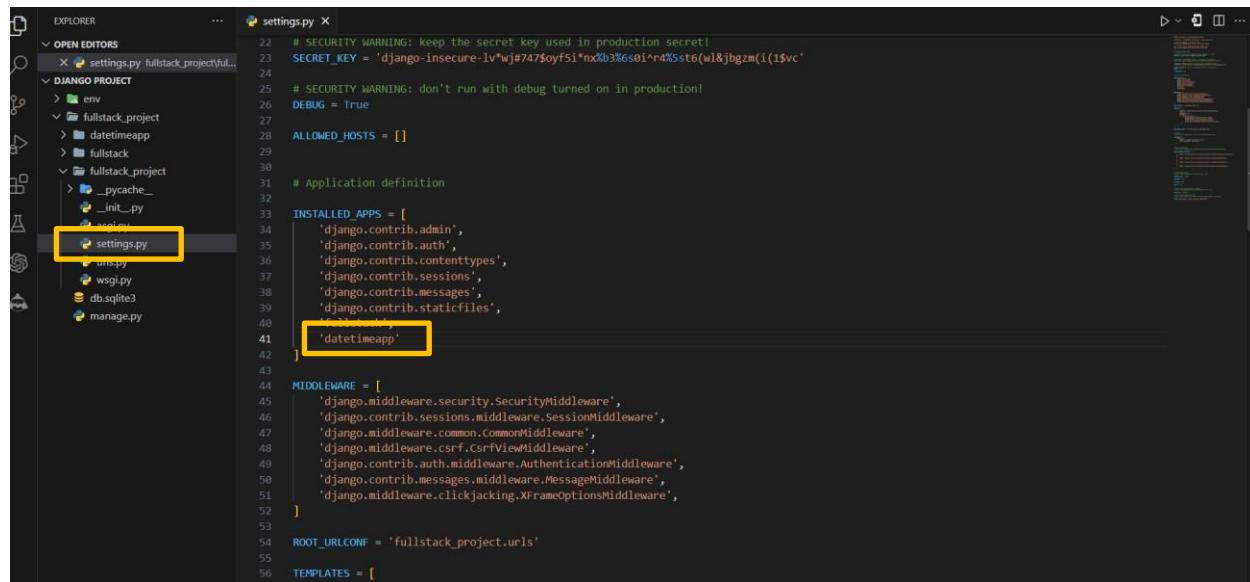
Develop a Django app that displays current date and time in server.

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

Step-02: Add the app to INSTALLED_APPS

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

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



```

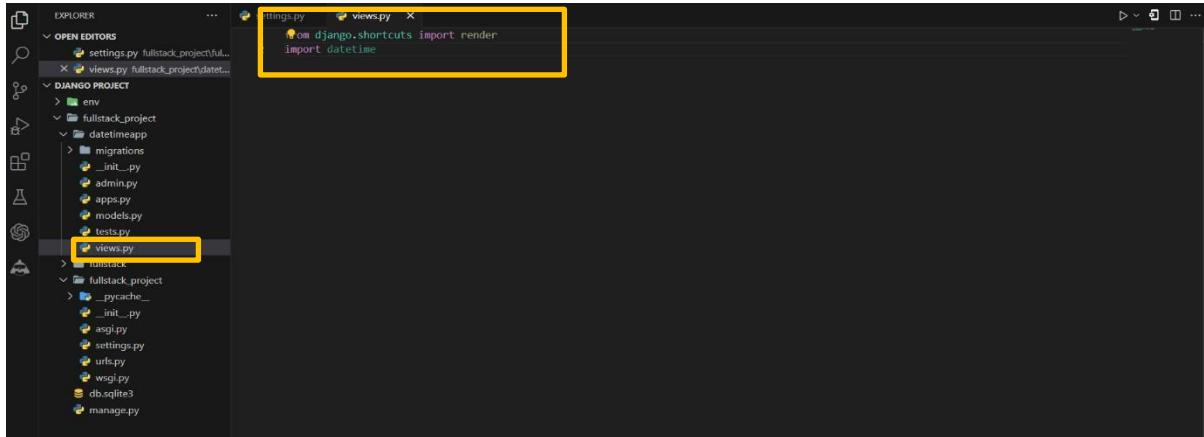
22 # SECURITY WARNING: keep the secret key used in production secret!
23 SECRET_KEY = 'django-insecure-lv*j#747$oyf5i*nx@3gs0i^r4%st6(wl&jbgzm(i1$vc'
24
25 # SECURITY WARNING: don't run with debug turned on in production!
26 DEBUG = True
27
28 ALLOWED_HOSTS = []
29
30
31 # Application definition
32
33 INSTALLED_APPS = [
34     'django.contrib.admin',
35     'django.contrib.auth',
36     'django.contrib.contenttypes',
37     'django.contrib.sessions',
38     'django.contrib.messages',
39     'django.contrib.staticfiles',
40     'datetimeapp' // This line is highlighted with a yellow box
41 ]
42
43
44 MIDDLEWARE = [
45     'django.middleware.security.SecurityMiddleware',
46     'django.contrib.sessions.middleware.SessionMiddleware',
47     'django.middleware.common.CommonMiddleware',
48     'django.middleware.csrf.CsrfViewMiddleware',
49     'django.contrib.auth.middleware.AuthenticationMiddleware',
50     'django.contrib.messages.middleware.MessageMiddleware',
51     'django.middleware.clickjacking.XFrameOptionsMiddleware',
52 ]
53
54 ROOT_URLCONF = 'fullstack_project.urls'
55
56 TEMPLATES = [

```

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

Step-03: Create a view function

- Open the views.py file in your Django app's directory (e.g., **datetimeapp/views.py**).
- Import the necessary modules at the top of the file



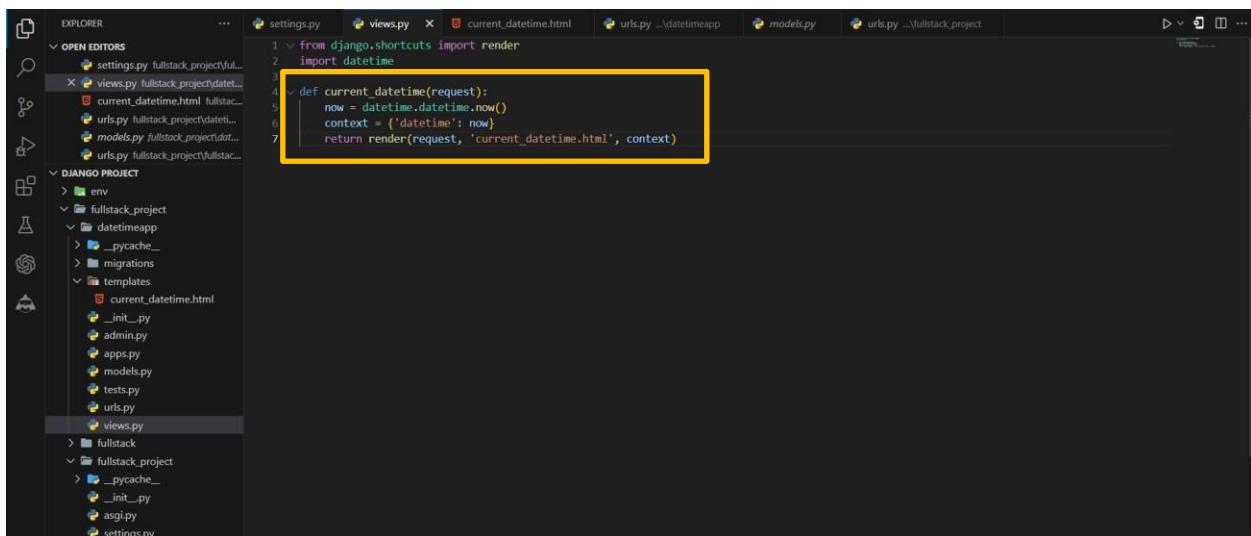
- Create a new **view function** that will handle the request and render the date and time:

```
def current_datetime(request):
```

```
    now = datetime.datetime.now()
```

```
    context = {'datetime': now}
```

```
    return render(request, 'current_datetime.html', context)
```



Step-04: Create a template

- In your app's directory (**e.g., datetimeapp**), create a new folder named templates.
 - Inside the templates folder, create another folder with the same name as your app (e.g., myapp).
-
- Inside the **datetimeapp folder**, create a new file named **current_datetime.html**.
 - Open **current_datetime.html** and add the following code to display the current date and time:

```
<!DOCTYPE html>

<html lang="en">

<head><meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Current Date and Time</title>

<!-- Bootstrap CSS -->

<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/css/bootstrap.min.css" rel="stylesheet">

<style>

/* Center content vertically */

html, body {

height: 100%;

display: flex;

justify-content: center;

align-items: center;

}

</style>

</head>
```

```
<body>

<div class="container text-center">

    <h1>Current Date and Time on the Server</h1>

    <p>{{ datetime }}</p>

</div>

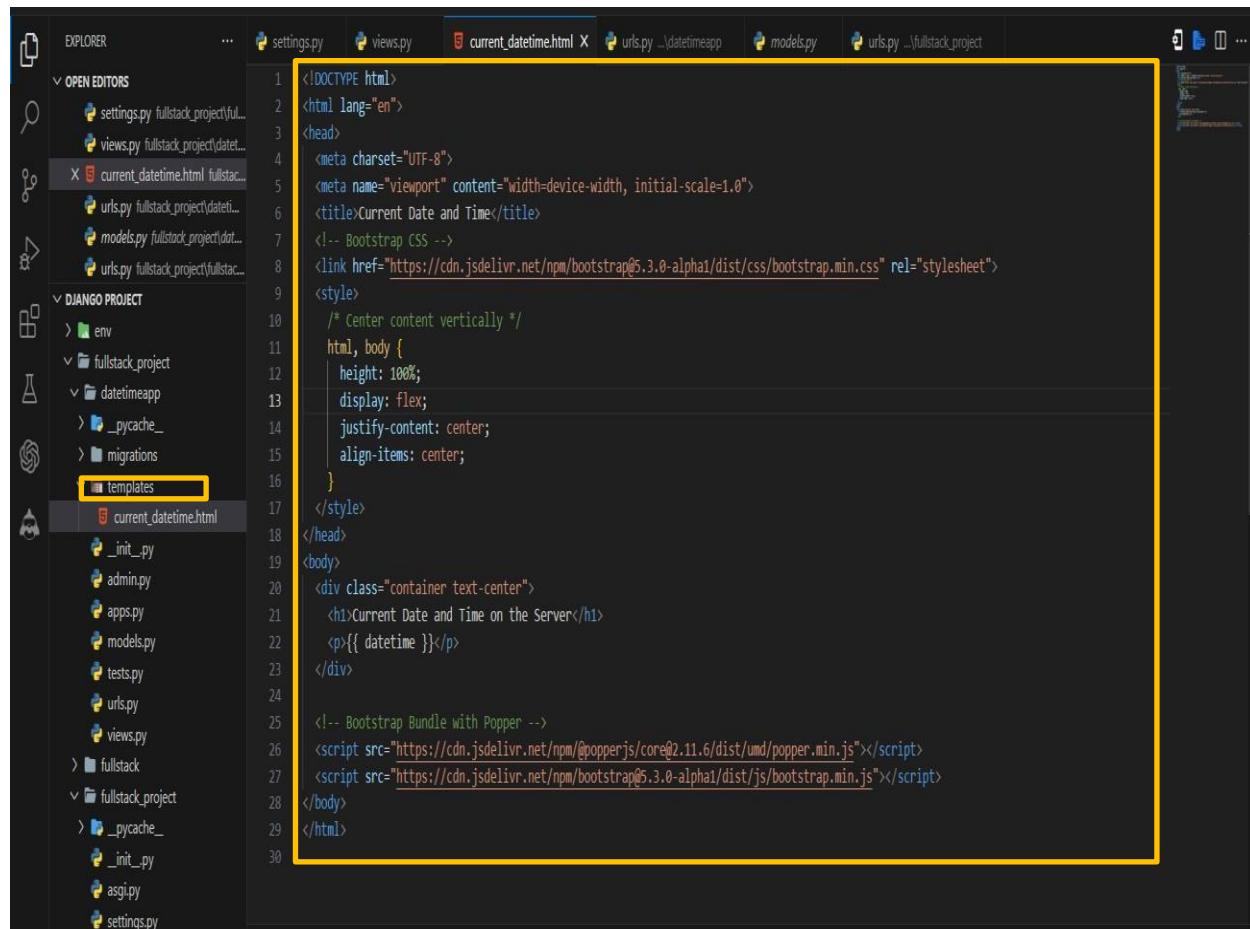
<!-- Bootstrap Bundle with Popper -->

<script
src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.11.6/dist/umd/popper.min.js"></script>

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/js/bootstrap.min.js"></script>

</body>

</html>
```



```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Current Date and Time</title>
    <!-- Bootstrap CSS -->
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/css/bootstrap.min.css" rel="stylesheet">
    <style>
        /* Center content vertically */
        html, body {
            height: 100%;
            display: flex;
            justify-content: center;
            align-items: center;
        }
    </style>
</head>
<body>
    <div class="container text-center">
        <h1>Current Date and Time on the Server</h1>
        <p>{{ datetime }}</p>
    </div>

    <!-- Bootstrap Bundle with Popper -->
    <script src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.11.6/dist/umd/popper.min.js"></script>
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/js/bootstrap.min.js"></script>
</body>
</html>
```

Step-05: Map the view function to a URL

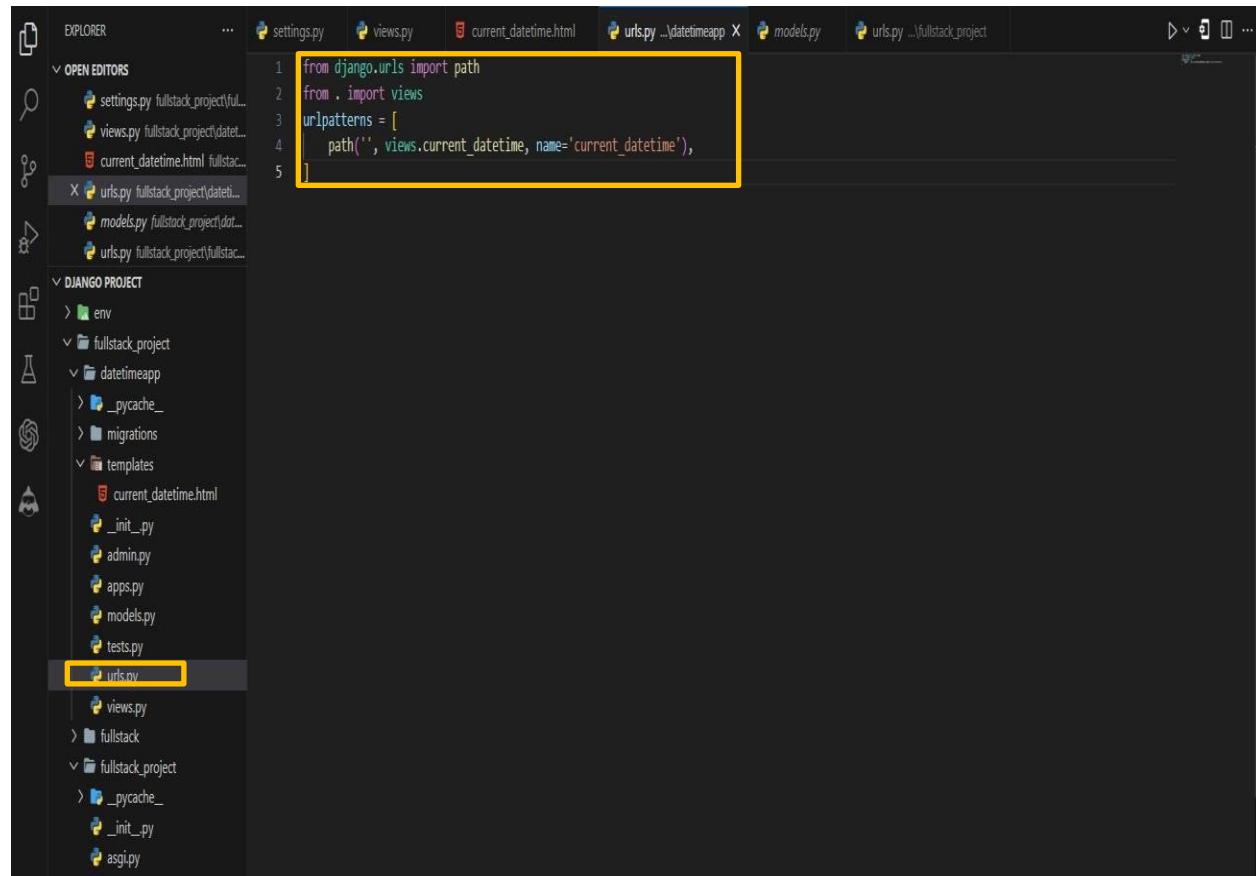
- Open the **urls.py** file in your Django app's directory (e.g., `datetimeapp/urls.py`).
- Import the view function at the top of the file
- Add a new URL pattern to the `urlpatterns` list

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

```
from . import views
```

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

This maps the **current_datetime** view function to the root URL `/`.



```
from django.urls import path  
from . import views  
urlpatterns = [  
    path ('', views.current_datetime, name='current_datetime'),  
]
```

Step-06: Include the app's URLs in the project's URL patterns

- Open the **urls.py** file in your project's directory (e.g., **fullstack_project/urls.py**).
- Import the include function from **django.urls** and the path function from **django.urls**:

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

- Add a new URL pattern to the urlpatterns list
- ```
path ('', include ('datetimeapp.urls')),
```
- This includes the URL patterns from your app's urls.py file.

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

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

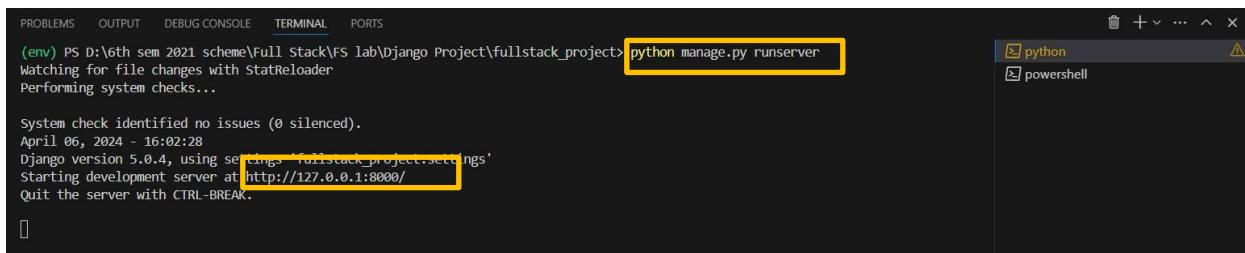
The screenshot shows a code editor interface with several files open in tabs at the top: settings.py, views.py, current\_datetime.html, urls.py (the active tab), models.py, and urls.py (another tab). The left sidebar shows a file tree for a 'Django Project' named 'fullstack\_project'. Inside 'fullstack\_project', there is a 'datetimeapp' folder containing \_\_pycache\_\_, migrations, and templates (with current\_datetime.html). In the main editor area, the 'urls.py' file is being edited. It contains the code above, which includes the 'datetimeapp.urls' file. The 'urls.py' file in the file tree is highlighted with a yellow box.

**Step-07: 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/>.



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
(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.
```

**Final Output of the Date and Time App****Current Date and Time on the Server**

April 6, 2024, 4:03 p.m.

## **Experiment-04**

Develop a Django app that displays date and time four hours ahead and four hours before as an offset of current date and time in server.

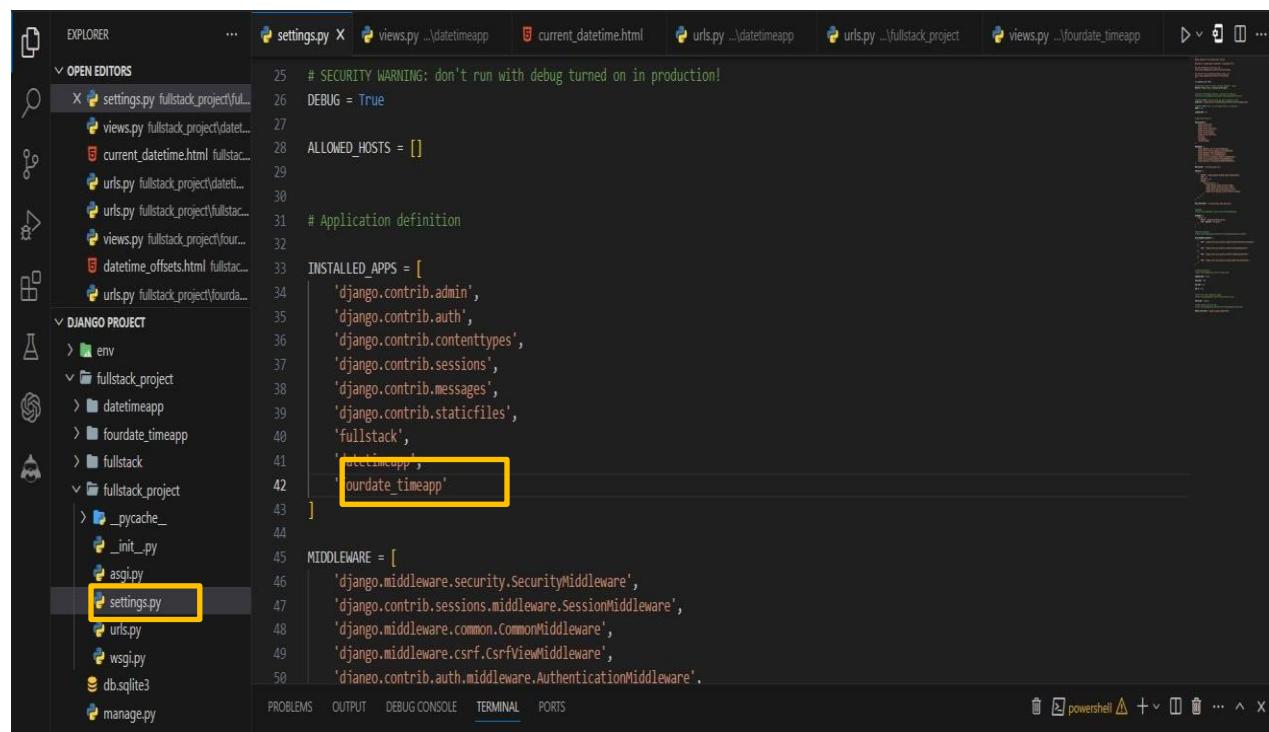
**Step-01:** This app will be created in the Django project we made earlier.

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

**Step-02:** Add the app to INSTALLED\_APPS

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

Locate the **INSTALLED\_APPS** list and add the name of your new app to the list:



```

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 'datetimeapp',
41 'fourdate_timeapp',
42]
43
44 MIDDLEWARE = [
45 'django.middleware.security.SecurityMiddleware',
46 'django.contrib.sessions.middleware.SessionMiddleware',
47 'django.middleware.common.CommonMiddleware',
48 'django.middleware.csrf.CsrfViewMiddleware',
49 'django.contrib.auth.middleware.AuthenticationMiddleware',
50]

```

**Step-03: Create a view function**

- Open the views.py file in your Django app's directory (e.g., **fourdate\_timeapp/views.py**).
- Import the necessary modules at the top of the file
- Create a new **view function** that will handle the request and render the date and time:

```
from django.shortcuts import render

import datetime

from dateutil import tz

def datetime_offsets(request):

 now = datetime.datetime.now()

 context = {

 'current_datetime': now,

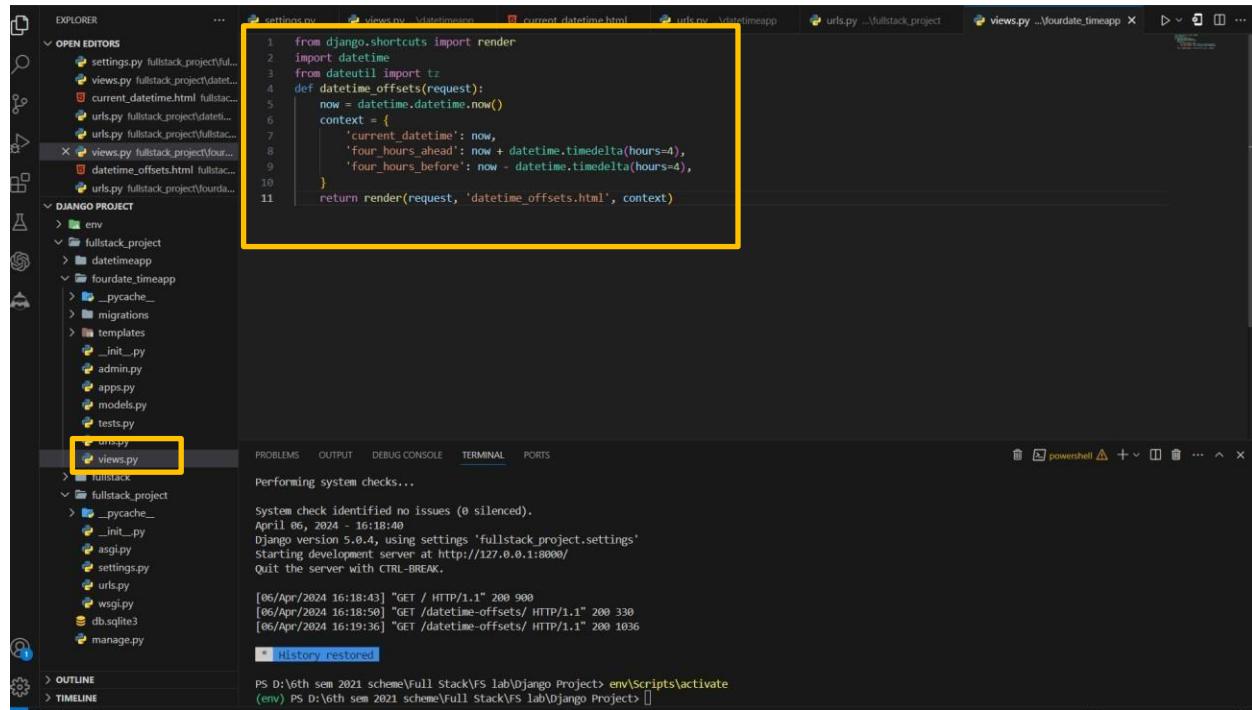
 'four_hours_ahead': now + datetime.timedelta(hours=4),

 'four_hours_before': now - datetime.timedelta(hours=4),

 }

 return render(request, 'datetime_offsets.html', context)
```

- This view function gets the current date and time using `datetime.datetime.now()`, calculates the date and time four hours ahead and four hours before using `datetime.timedelta`, and then passes all three values to the template as context variables.



```

1 from django.shortcuts import render
2 import datetime
3 from dateutil import tz
4 def datetime_offsets(request):
5 now = datetime.datetime.now()
6 context = {
7 'current_datetime': now,
8 'four_hours_ahead': now + datetime.timedelta(hours=4),
9 'four_hours_before': now - datetime.timedelta(hours=4),
10 }
11 return render(request, 'datetime_offsets.html', context)

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

Performing system checks...

System check identified no issues (0 silenced).

April 06, 2024 - 16:18:40

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.

[06/Apr/2024 16:18:43] "GET / HTTP/1.1" 200 900

[06/Apr/2024 16:18:50] "GET /datetime-offsets/ HTTP/1.1" 200 330

[06/Apr/2024 16:19:36] "GET /datetime-offsets/ HTTP/1.1" 200 1036

[History restored]

PS D:\6th sem 2021 scheme\Full Stack\FS lab\ Django Project> env\Scripts\activate  
 (env) PS D:\6th sem 2021 scheme\Full Stack\FS lab\ Django Project> []

#### Step-04: Create a new template

- In your app's directory (e.g., **fourdate\_timeapp**), create a new folder named **templates** (if it doesn't already exist).
- Inside the **templates** folder, create another folder with the same name as your app (e.g., **fourdate\_timeapp**).
- Inside the **fourdate\_timeapp** folder, create a new file named **datetime\_offsets.html**.
- Open **datetime\_offsets.html** and add the following code to display the current date and time, along with the offsets:

```

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

```

```
<title>Date and Time Offsets</title>

<!-- Bootstrap CSS -->

<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/css/bootstrap.min.css"
rel="stylesheet">

<style>

/* Center content vertically */ tml,
body {
 height: 100%;
 display: flex;
 justify-content: center;
 align-items: center;
}

</style>

</head>

<body>

<div class="container text-center">

<h1>Current Date and Time on the Server</h1>

<p>{{ current_datetime }}</p>

<h2>Four Hours Ahead</h2>

<p>{{ four_hours_ahead }}</p>

<h2>Four Hours Before</h2>
```

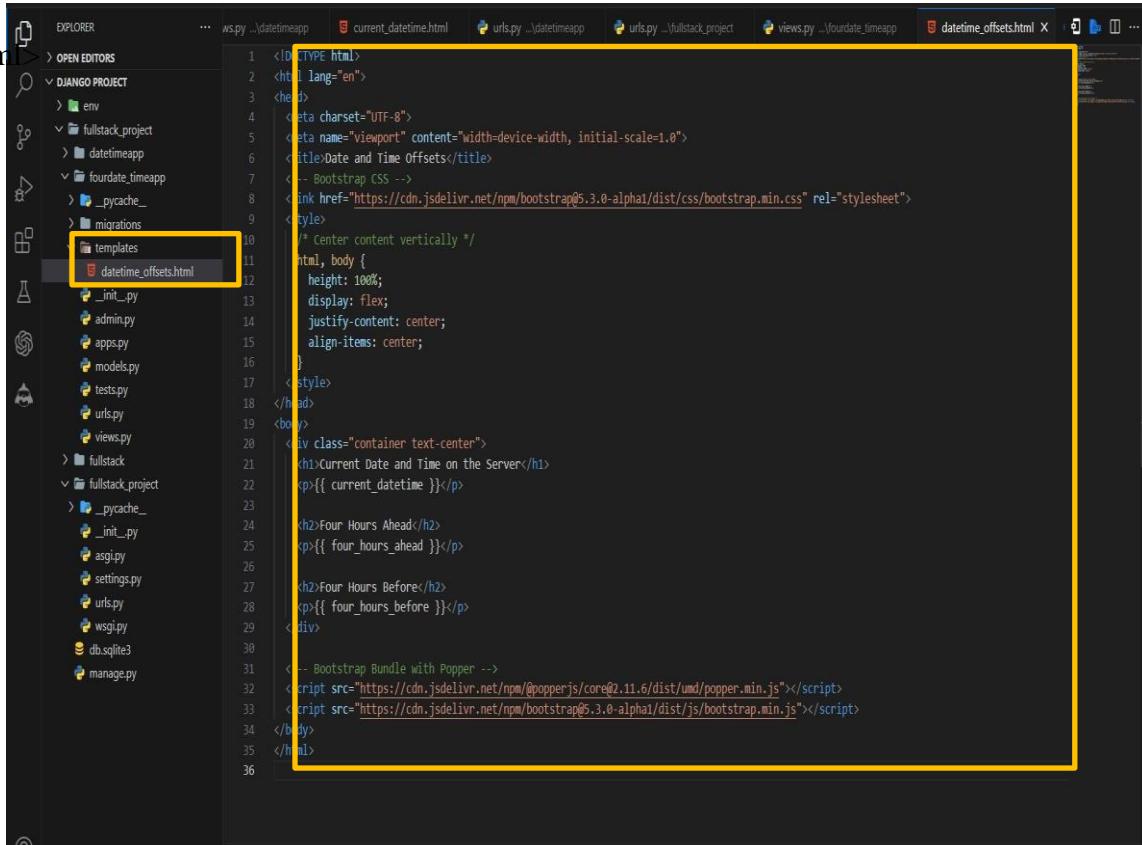
```
<p>{{ four_hours_before }}</p>
</div>
```

```
<!-- Bootstrap Bundle with Popper -->
```

- This template displays the **current\_datetime**, **four\_hours\_ahead**, and **four\_hours\_before** variables passed from the view function.

```
<script
src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.11.6/dist/umd/popper.min.js"></script>

<script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/js/bootstrap.min.js"></script>
```



```
</body></html>
```

```

1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4 <meta charset="UTF-8">
5 <meta name="viewport" content="width=device-width, initial-scale=1.0">
6 <title>Date and Time Offsets</title>
7 <!-- Bootstrap CSS -->
8 <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/css/bootstrap.min.css" rel="stylesheet">
9 <style>
10 /* Center content vertically */
11 html, body {
12 height: 100%;
13 display: flex;
14 justify-content: center;
15 align-items: center;
16 }
17 </style>
18 </head>
19 <body>
20 <div class="container text-center">
21 <h1>Current Date and Time on the Server</h1>
22 <p>{{ current_datetime }}</p>
23
24 <h2>Four Hours Ahead</h2>
25 <p>{{ four_hours_ahead }}</p>
26
27 <h2>Four Hours Before</h2>
28 <p>{{ four_hours_before }}</p>
29 </div>
30
31 <!-- Bootstrap Bundle with Popper -->
32 <script src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.11.6/dist/umd/popper.min.js"></script>
33 <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/js/bootstrap.min.js"></script>
34 </body>
35 </html>

```

## Step-05: Map the view function to a URL

- Open the **urls.py** file in your Django app's directory (e.g., fourdate\_timeapp/urls.py).
- Import the view function at the top of the file
- Add a new URL pattern to the urlpatterns list

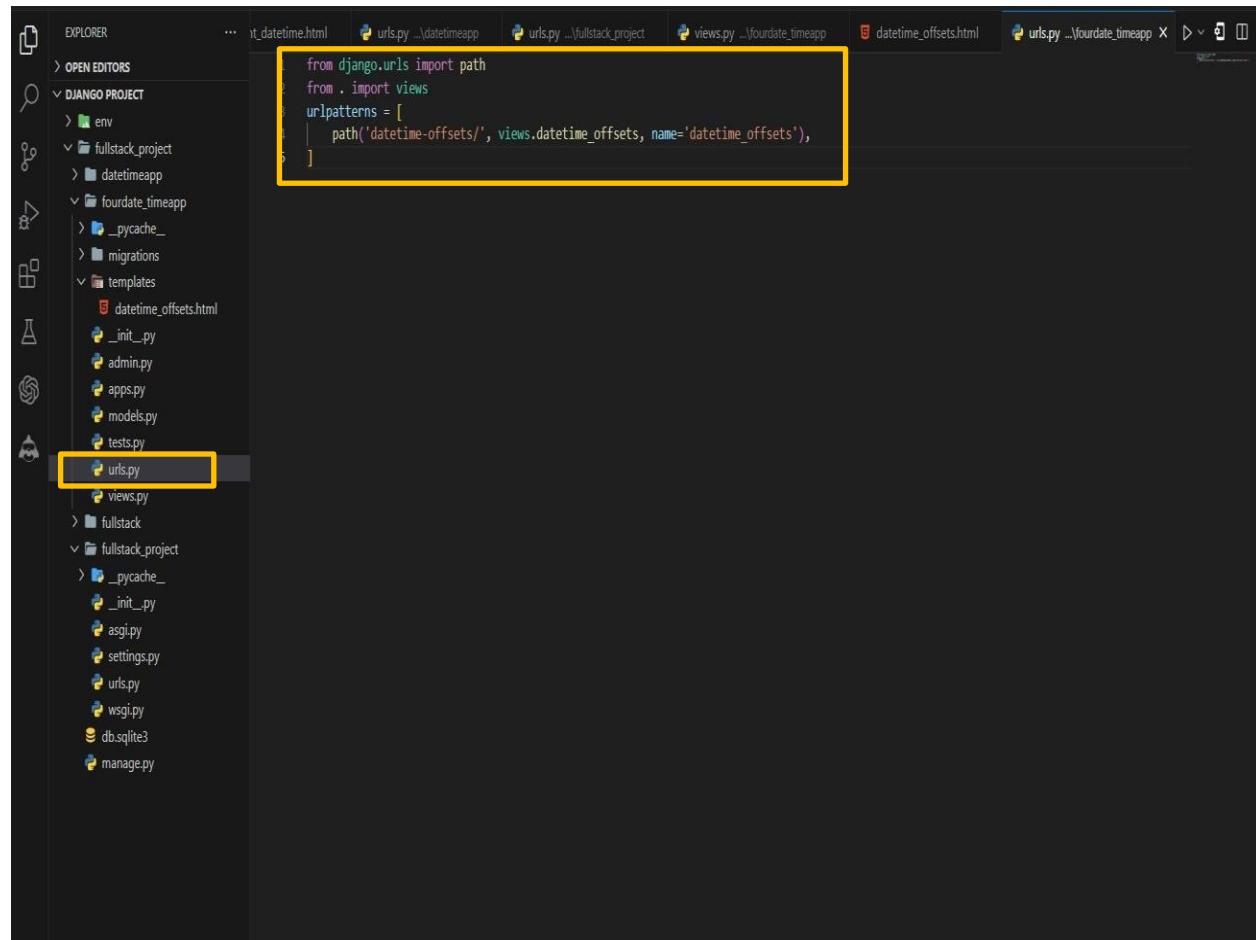
```
from django.urls import path
```

```
from . import views
```

```
urlpatterns = [
```

```
 path('datetime-offsets/', views.datetime_offsets, name='datetime_offsets'),
```

```
]
```



```
from django.urls import path
from . import views
urlpatterns = [
 path('datetime-offsets/', views.datetime_offsets, name='datetime_offsets'),
]
```

## Step-06: Include the app's URLs in the project's URL patterns

- Open the **urls.py** file in your project's directory (e.g., **fullstack\_project/urls.py**).
- Import the include function from **django.urls** and the path function from **django.urls**:

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

- Add a new URL pattern to the urlpatterns list

```
path('', include('fourdate_timeapp.urls')),
```

- This includes the URL patterns from your app's **urls.py** file.

The screenshot shows the Visual Studio Code interface with the following details:

- EXPLORER:** Shows the project structure under "Django Project". It includes the `env`, `fullstack_project` (which contains `datetimeapp` and `fourdate_timeapp`), and `fullstack`. Each app folder has its own `__pycache__`, `migrations`, and `templates` subfolders. The `fourdate_timeapp` folder also contains `admin.py`, `apps.py`, `models.py`, `tests.py`, `urls.py` (which is highlighted with a yellow box), and `views.py`. The `fullstack` folder contains `__init__.py`, `asgi.py`, `settings.py`, `urls.py` (highlighted with a yellow box), and `wsgi.py`.
- EDITOR:** The `urls.py` file for the `fourdate_timeapp` is open. The code is as follows:

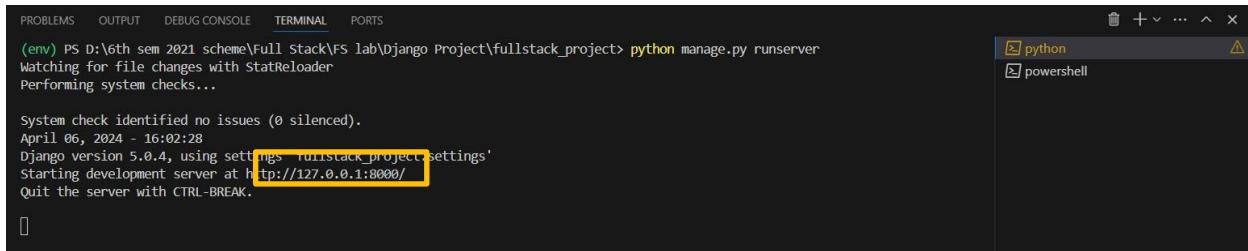
```
1 from django.contrib import admin
2 from django.urls import include, path
3
4 urlpatterns = [
5 path('admin/', admin.site.urls),
6 path('^', include('datetimeapp.urls')),
7 path('^', include('fourdate_timeapp.urls'))
8]
```

## Step-07: 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/>.
- Type or copy this <http://127.0.0.1:8000/datetime-offsets/>



```
(env) PS D:\6th sem 2021 scheme\Full Stack\F5 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.
```

## Final Output of the Date and Time App



### Current Date and Time on the Server

April 6, 2024, 10:15 p.m.

### Four Hours Ahead

April 7, 2024, 2:15 a.m.

### Four Hours Before

April 6, 2024, 6:15 p.m.

## Experiment-05

Develop a simple Django app that displays an unordered list of fruits and ordered list of selected students for an event.

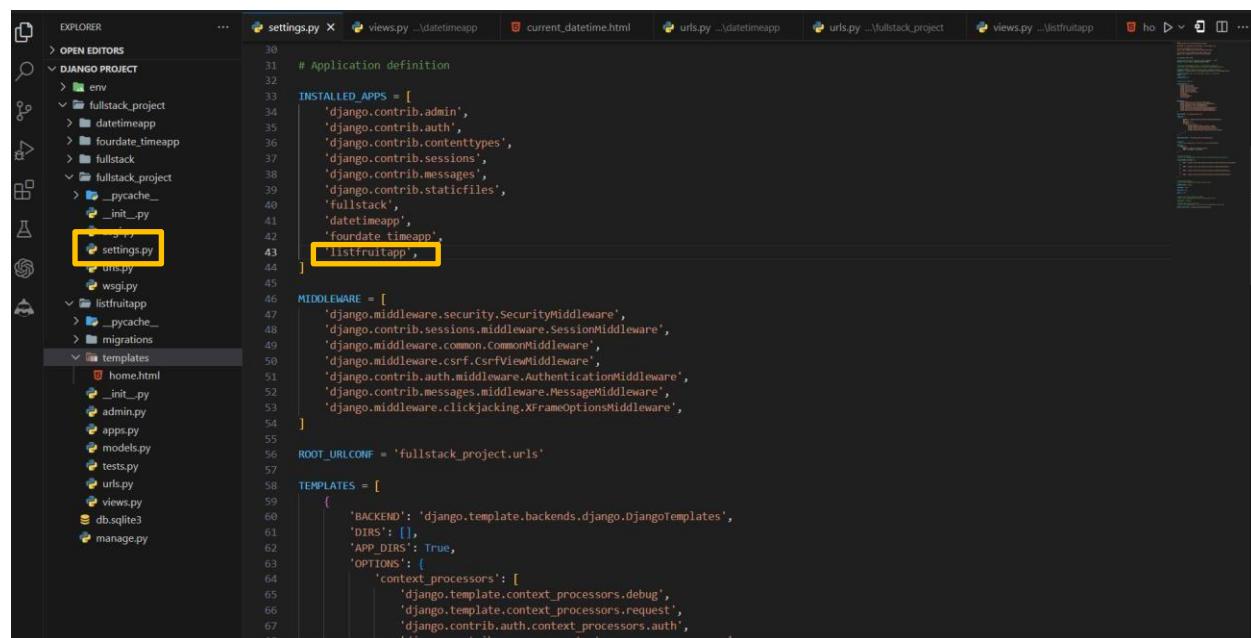
**Step-01:** This app will be created in the Django project we made earlier.

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

**Step-02: Add the app to INSTALLED\_APPS**

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

Locate the **INSTALLED\_APPS** list and add the name of your new app to the list:



```

Application definition

INSTALLED_APPS = [
 'django.contrib.admin',
 'django.contrib.auth',
 'django.contrib.contenttypes',
 'django.contrib.sessions',
 'django.contrib.messages',
 'django.contrib.staticfiles',
 'fullstack',
 'datetimeapp',
 'foundatimeapp',
 'listfruitapp',
]

MIDDLEWARE = [
 'django.middleware.security.SecurityMiddleware',
 'django.contrib.sessions.middleware.SessionMiddleware',
 'django.middleware.common.CommonMiddleware',
 'django.middleware.csrf.CsrfViewMiddleware',
 'django.contrib.auth.middleware.AuthenticationMiddleware',
 'django.contrib.messages.middleware.MessageMiddleware',
 'django.middleware.clickjacking.XFrameOptionsMiddleware',
]

ROOT_URLCONF = 'fullstack_project.urls'

TEMPLATES = [
 {
 'BACKEND': 'django.template.backends.django.DjangoTemplates',
 'DIRS': [],
 'APP_DIRS': True,
 'OPTIONS': {
 'context_processors': [
 'django.template.context_processors.debug',
 'django.template.context_processors.request',
 'django.contrib.auth.context_processors.auth',
],
 },
 },
]

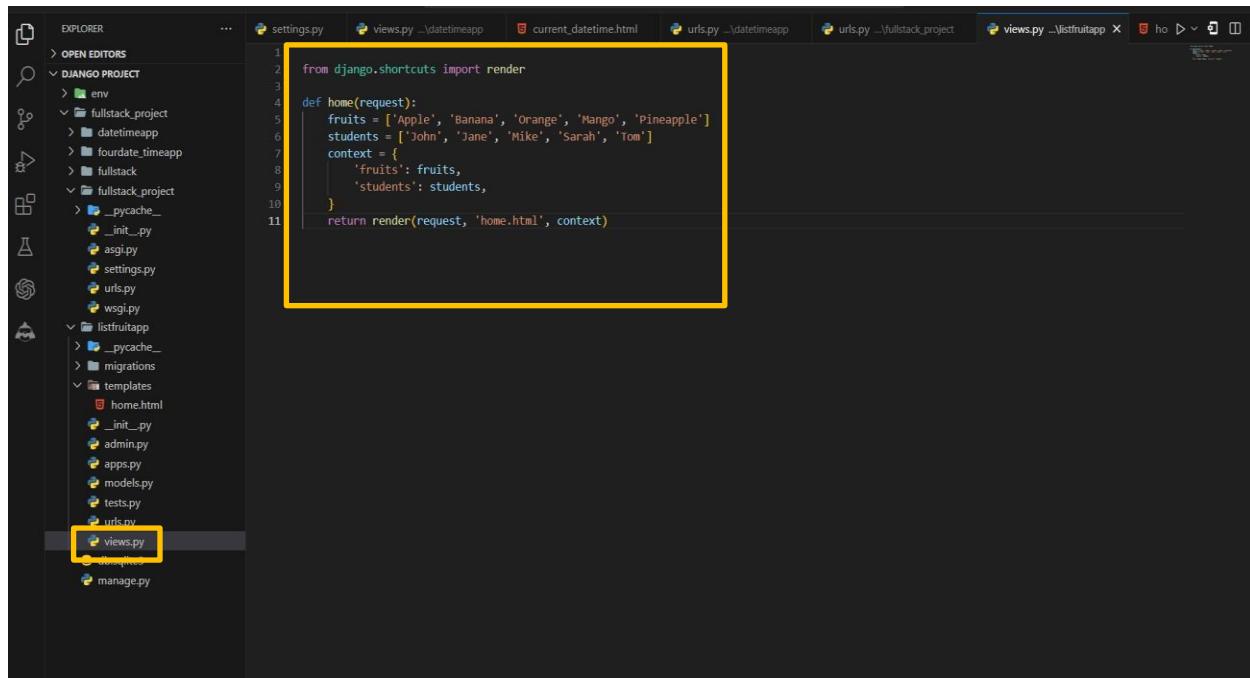
```

### Step-03: Create a view function

- Open the views.py file in your Django app's directory (e.g., listfruitapp/views.py).
- Create a new **view function** that will handle the request and render the date and time:  

```
from django.shortcuts import render
```

```
def home(request):
 fruits = ['Apple', 'Banana', 'Orange', 'Mango', 'Pineapple']
 students = ['John', 'Jane', 'Mike', 'Sarah', 'Tom']
 context = {
 'fruits': fruits,
 'students': students,
 }
 return render(request, 'home.html', context)
```



The screenshot shows a code editor interface with a dark theme. On the left is the 'EXPLORER' sidebar showing the project structure. The main area displays the contents of the 'views.py' file. A yellow box highlights the code for the 'home' view function. The code itself is as follows:

```
from django.shortcuts import render

def home(request):
 fruits = ['Apple', 'Banana', 'Orange', 'Mango', 'Pineapple']
 students = ['John', 'Jane', 'Mike', 'Sarah', 'Tom']
 context = {
 'fruits': fruits,
 'students': students,
 }
 return render(request, 'home.html', context)
```

- Here, we define a view function `home` that creates two lists: `fruits` and `students`. These lists are then passed to the template as a context dictionary.

**Step-04: Create a new template**

- In your app's directory (**e.g., listfruitapp**), create a new folder named templates (if it doesn't already exist).
- Inside the templates folder, create another folder with the same name as your app (e.g., **listfruitapp**).  
display: flex;  
justify-content: center;
- Inside the fourdate\_timeapp folder, create a new file named **home.html**.
- Open **home.html** and add the following code.

```
<!DOCTYPE html>

<html>

<head>
 <title>Fruits and Students</title>

 <!-- Add Bootstrap CSS -->

 <link href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css"
 rel="stylesheet">

 <style>
 /* Center content vertically */
 html, body {
 height: 100%;
 }
 body {
 align-items: center;
 }
 </style>

```

```
 }
```

```
.container {
```

```
 text-align: center;
```

```
 <li class="list-group-item">{ { fruit } }
```

```
 {% endfor %}
```

```

```

```
}
```

```
/* Style for lists */
```

```
.list-container { isplay:
```

```
 inline-block;
```

```
 margin: 0 20px; /* Add space between lists */
```

```
}
```

```
</style>
```

```
</head>
```

```
<body>
```

```
 <div class="container">
```

```
 <div class="row">
```

```
 <div class="col">
```

```
 <h1>Fruits</h1>
```

```
 <ul class="list-group list-container">
```

```
</div>

<div class="col">

 <h1>Selected Students</h1>

 <ol class="list-group list-container">

 {% for student in students %}

 <li class="list-group-item">{{ student }}

 {% endfor %}

</div>

</div>

</div>

<!-- Bootstrap JS (optional) -->

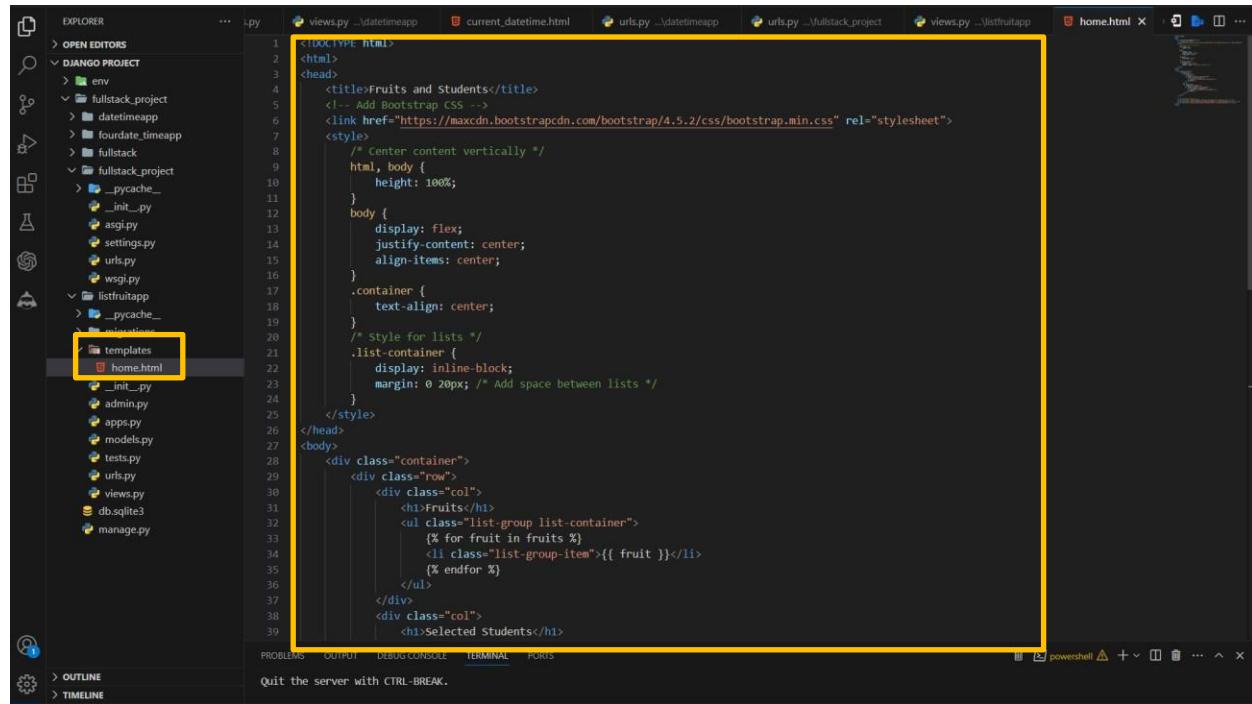
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>

<script
src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.16.0/umd/popper.min.js"></script>

<script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>

</body>

</html>
```



```

<!DOCTYPE html>
<html>
<head>
 <title>Fruits and Students</title>
 <!-- Add Bootstrap CSS -->
 <link href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css" rel="stylesheet">
 <style>
 /* Center content vertically */
 html, body {
 height: 100%;
 }
 body {
 display: flex;
 justify-content: center;
 align-items: center;
 }
 .container {
 text-align: center;
 }
 /* Style for lists */
 .list-container {
 display: inline-block;
 margin: 0 20px; /* Add space between lists */
 }
 </style>
</head>
<body>
 <div class="container">
 <div class="row">
 <div class="col">
 <h1>Fruits</h1>
 <ul class="list-group list-container">
 {% for fruit in fruits %}
 <li class="list-group-item">{{ fruit }}
 {% endfor %}

 </div>
 <div class="col">
 <h1>Selected Students</h1>
 </div>
 </div>
 </div>
</body>

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

- In this template, we use Django's template tags to loop through the fruits and students lists and render them as an unordered list and an ordered list, respectively.

## Step-05: Map the view function to a URL

- Open the **urls.py** file in your Django app's directory (e.g., listfruitapp/urls.py).
- Import the view function at the top of the file
- Add a new URL pattern to the urlpatterns list

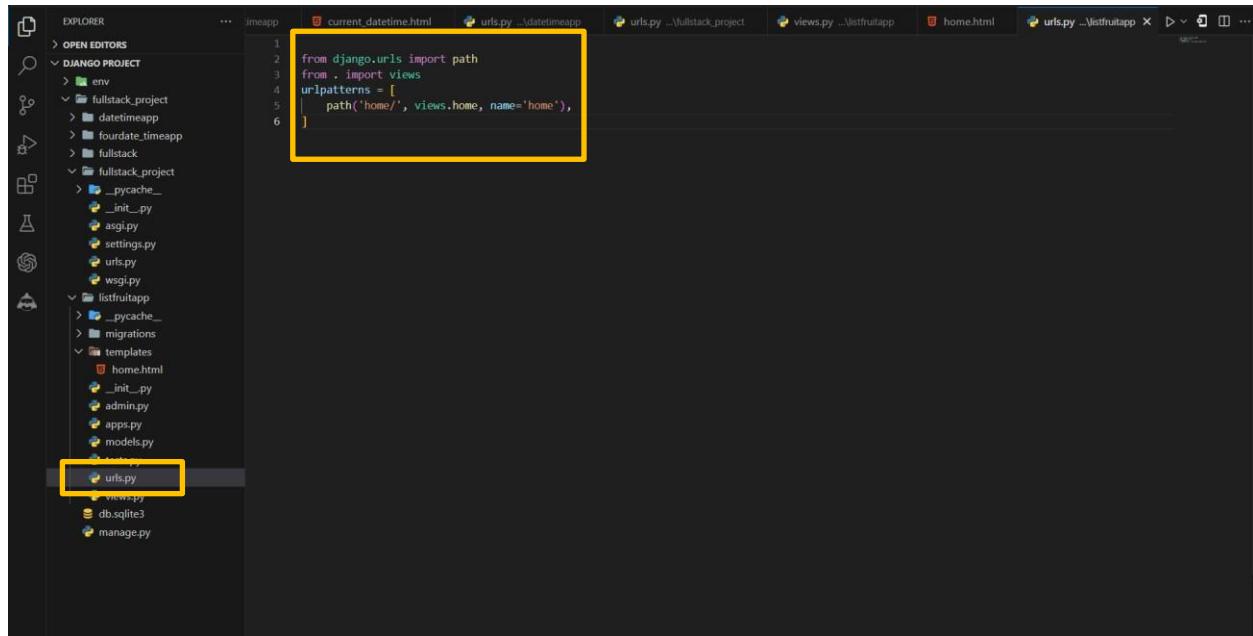
```
from django.urls import path
```

```
from . import views
```

```
urlpatterns = [
```

```
 path('home/', views.home, name='home'),
```

```
]
```



```

from django.urls import path
from . import views
urlpatterns = [
 path('home/', views.home, name='home'),
]

```

### Step-06: Include the app's URLs in the project's URL patterns

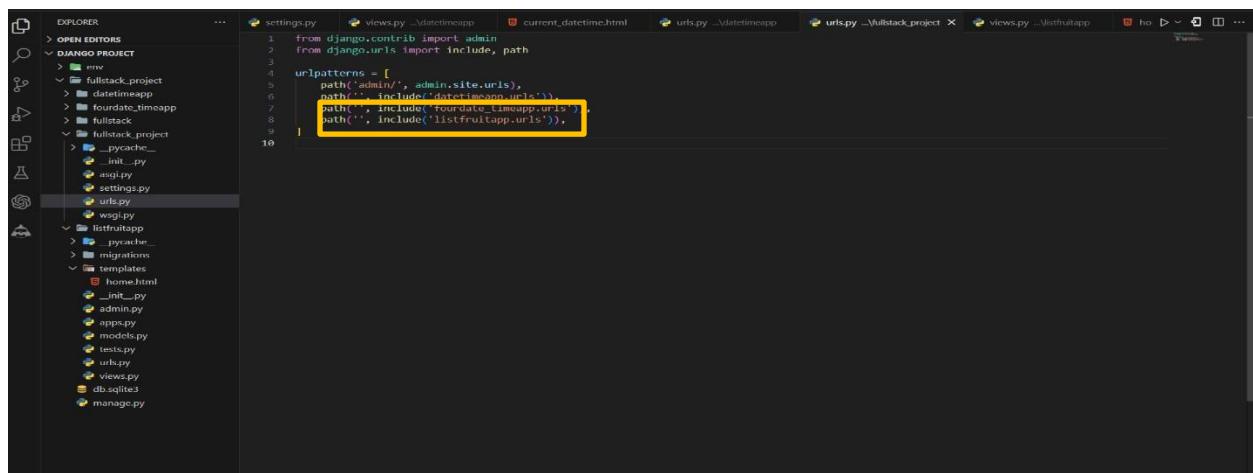
- Open the **urls.py** file in your project's directory (e.g., **fullstack\_project/urls.py**).
- Import the **include** function from **django.urls** and the **path** function from **django.urls**:

**from django.urls import include, path**

- Add a new URL pattern to the **urlpatterns** list

**path('', include(listfruitapp.urls)),**

- This includes the URL patterns from your app's **urls.py** file.



```

from django.contrib import admin
from django.urls import include, path
urlpatterns = [
 path('admin/', admin.site.urls),
 path('', include('datetimeapp.urls')),
 path('', include('fourdate_timeapp.urls')),
 path('', include('listfruitapp.urls')),
]

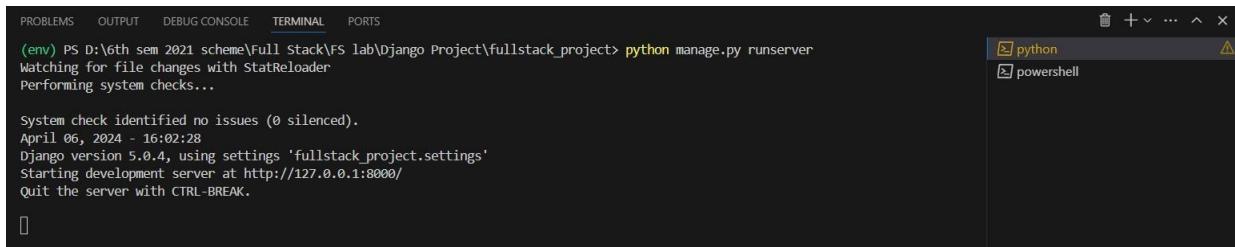
```

## Step-07: 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/>.

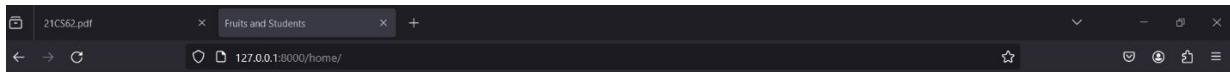


```
(env) PS D:\6th sem 2021 scheme\Full Stack\FS lab\ Django Project\fullstack_project> python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
April 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.
```

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

## Final Output of the Unorder list of Fruits and Students

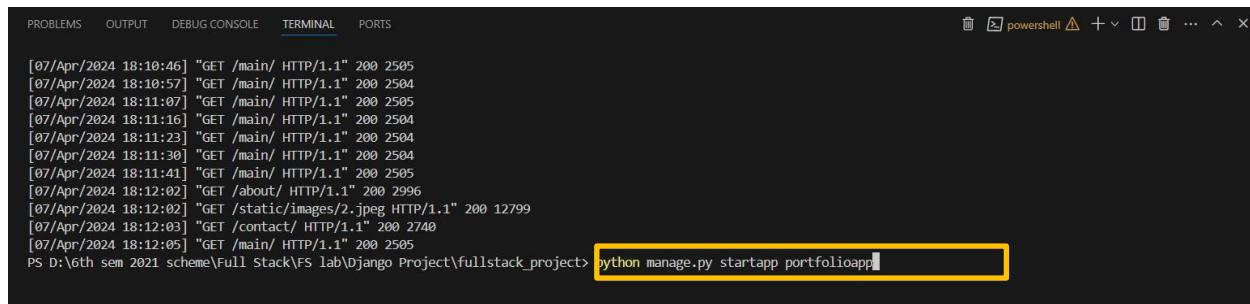


Fruits	Selected Students
Apple	John
Banana	Jane
Orange	Mike
Mango	Sarah
Pineapple	Tom

## **Experiment-06**

Develop a layout.html with a suitable header (containing navigation menu) and footer with copyright and developer information. Inherit this layout.html and create 3 additional pages: contact us, About Us and Home page of any website.

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

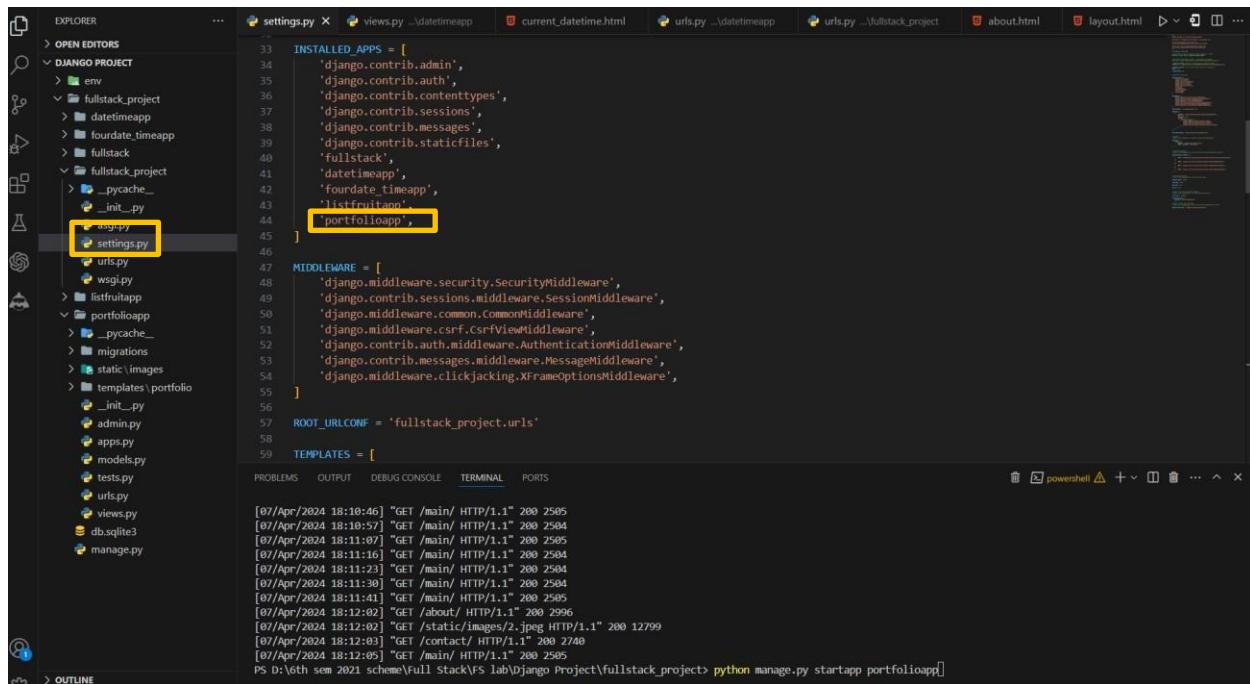


```
[07/Apr/2024 18:10:46] "GET /main/ HTTP/1.1" 200 2505
[07/Apr/2024 18:10:57] "GET /main/ HTTP/1.1" 200 2504
[07/Apr/2024 18:11:07] "GET /main/ HTTP/1.1" 200 2505
[07/Apr/2024 18:11:16] "GET /main/ HTTP/1.1" 200 2504
[07/Apr/2024 18:11:23] "GET /main/ HTTP/1.1" 200 2504
[07/Apr/2024 18:11:30] "GET /main/ HTTP/1.1" 200 2504
[07/Apr/2024 18:11:41] "GET /main/ HTTP/1.1" 200 2505
[07/Apr/2024 18:12:02] "GET /about/ HTTP/1.1" 200 2996
[07/Apr/2024 18:12:02] "GET /static/images/2.jpeg HTTP/1.1" 200 12799
[07/Apr/2024 18:12:03] "GET /contact/ HTTP/1.1" 200 2740
[07/Apr/2024 18:12:05] "GET /main/ HTTP/1.1" 200 2505
PS D:\6th sem 2021 scheme\Full Stack\FS lab\django Project\fullstack_project> python manage.py startapp portfolioapp
```

**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:



```
INSTALLED_APPS = [
 'django.contrib.admin',
 'django.contrib.auth',
 'django.contrib.contenttypes',
 'django.contrib.sessions',
 'django.contrib.messages',
 'django.contrib.staticfiles',
 'fullstack',
 'datetimeapp',
 'foundate_timeapp',
 'listfruitapp',
 'portfolioapp',
]
```

**Step-03: Create a new template**

- Inside the portfolioapp/templates/ portfolio directory, create the following files:

**layout.html**

```
<!DOCTYPE html>
<html lang="en">

<head>

 <meta charset="UTF-8">

 <meta name="viewport" content="width=device-width, initial-scale=1.0">

 <title>{ % block title % }{ % endblock % }</title>

 <!-- Include Bootstrap CSS from CDN -->

 <link rel="stylesheet"
 href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">

 <style>

 /* Add some basic styling */

 body {

 font-family: Arial, sans-serif;

 margin: 0;

 padding: 0;

 }

 header {

 background-color: #333;

 color: #fff;

 }

 </style>
```

```
padding: 10px;

text-align: center; /* Center align header content */

}

nav ul {

list-style-type: none;

margin: 0;

padding: 0;

display: inline-block; /* Display nav items inline-block */

}
nav ul li {

display: inline;

margin-right: 10px;

}

nav ul li a {

color: #fff;

text-decoration: none;

}

footer {

background-color: #333;

color: #fff;

padding: 10px;
```

```
text-align: center;

}

/* Center align content */

About Us

Contact Us

.content-wrapper {

display: flex;

justify-content: center;

align-items: center;

min-height: 80vh; /* Set minimum height for content area */

}

</style>

{% load static %}

</head>

<body>

<header>

<nav>

Home


```

```
</nav>

</header>

<div class="content-wrapper">

 {% block content %}

 {% endblock %}

</div>

<footer>

 <p>© 2024 Search Creators. Developed by Hanumanthu.</p>

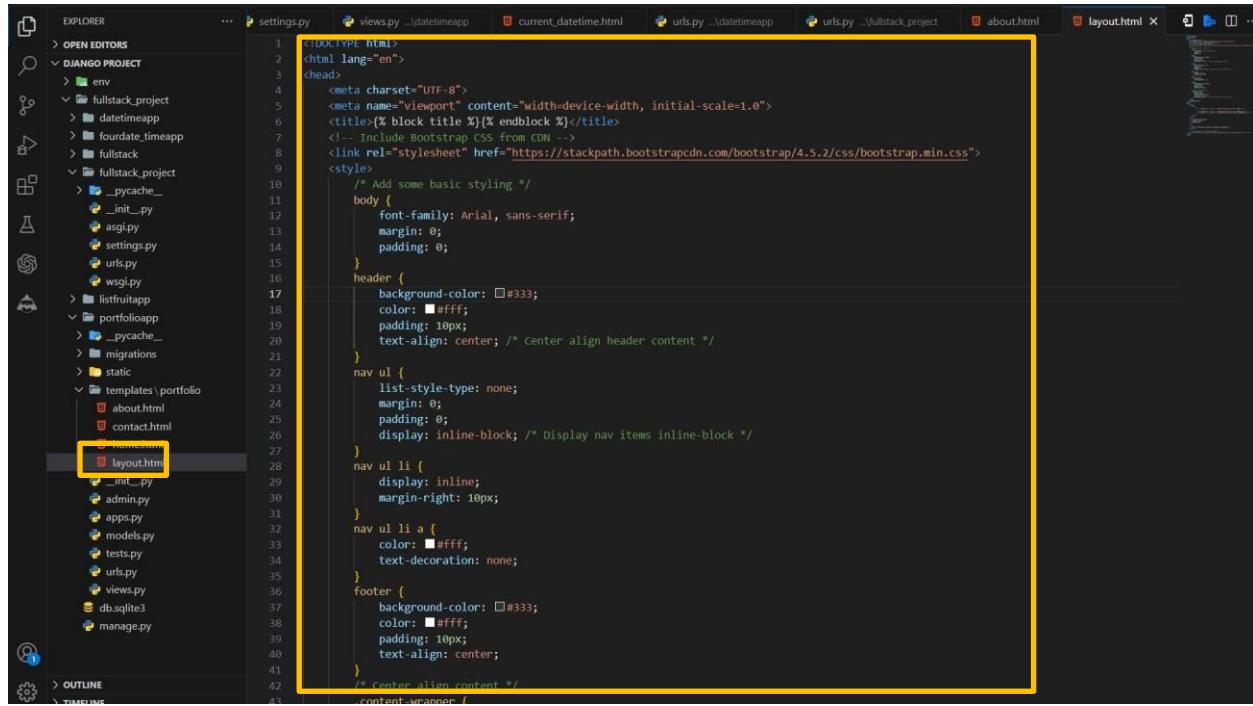
</footer>

<!-- Include Bootstrap JS from CDN (Optional) -->

<script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>

</body>

</html>
```



```

1 <!DOCTYPE HTML>
2 <html lang="en">
3 <head>
4 <meta charset="UTF-8">
5 <meta name="viewport" content="width=device-width, initial-scale=1.0">
6 <title>{% block title %}{% endblock %}</title>
7 <!-- Include Bootstrap CSS from CDN -->
8 <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
9 /* Add some basic styling */
10 body {
11 font-family: Arial, sans-serif;
12 margin: 0;
13 padding: 0;
14 }
15 header {
16 background-color: #333;
17 color: #fff;
18 padding: 10px;
19 text-align: center; /* Center align header content */
20 }
21 nav ul {
22 list-style-type: none;
23 margin: 0;
24 padding: 0;
25 display: inline-block; /* Display nav items inline-block */
26 }
27 nav ul li {
28 display: inline;
29 margin-right: 10px;
30 }
31 nav ul li a {
32 color: #fff;
33 text-decoration: none;
34 }
35 footer {
36 background-color: #333;
37 color: #fff;
38 padding: 10px;
39 text-align: center;
40 }
41 /* Center align content */
42 .center-content {
43 display: flex;
44 justify-content: center;
45 }

```

## home.html

```

{ % extends 'portfolio/layout.html' % }

{ % block title % }Home{ % endblock % }

{ % block extra_css % }

<!-- Include Bootstrap CSS from CDN -->

<link rel="stylesheet"
 href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">

<style>

/* Custom CSS to center align content */

.center-content {

 display: flex;

 justify-content: center;

```

```
align-items: center;

height: 80vh; /* Set height to viewport height for full page centering */

.center-

content>div {

text-align:

center;

}

</style>

{ % endblock % }

{ % block content % }

<div class="center-content">

<div>

<h1 class="text-center my-3">Welcome to My Website</h1>

<p class="text-center">This is the home page of our website.</p>

</div>

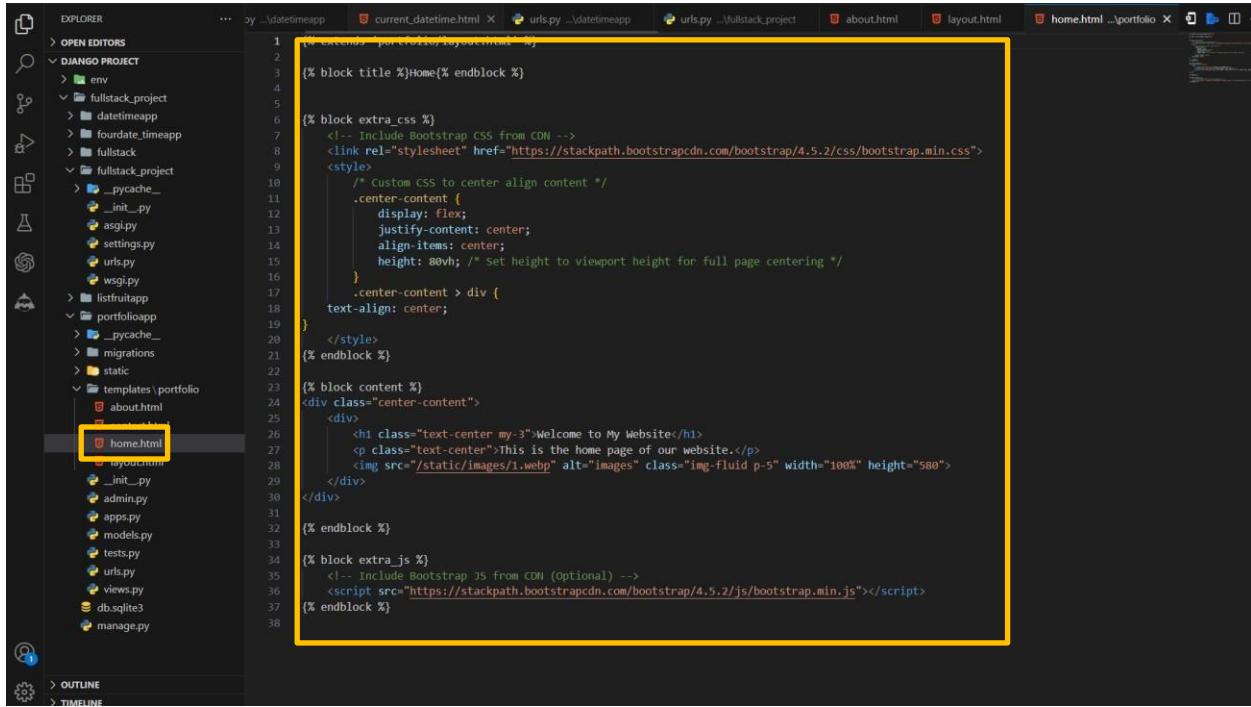
</div>

{

<!-- Include Bootstrap JS from CDN (Optional) -->

<script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>
```

```
{% endblock %}
```



The screenshot shows the VS Code interface with the Django project structure in the Explorer sidebar. The current file is `home.html`, which is highlighted with a yellow border. The code editor displays the following content:

```

1 1<div> extends 'portfolio/layout.html' </div>
2
3 2<% block title %>Home<% endblock %>
4
5 3<% block extra_css %>
6 4 <!-- Include Bootstrap CSS from CDN -->
7 5 <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
8
9 6<% block content %>
10 7 <!-- Custom CSS to center align content -->
11 8 .center-content {
12 9 display: flex;
13 10 justify-content: center;
14 11 align-items: center;
15 12 height: 80vh; /* set height to viewport height for full page centering */
16 13 }
17 14 .center-content > div {
18 15 text-align: center;
19 16 }
20 17 </style>
21 18<% endblock %>
22
23 19<% block extra_js %>
24 20 <!-- Include Bootstrap JS from CDN (Optional) -->
25 21 <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>
26 22<% endblock %>
27
28
29
30
31
32
33
34
35
36
37
38

```

## about.html

```
{% extends 'portfolio/layout.html' %}
```

```
{% block title %}About Us{% endblock %}
```

```
{% block extra_css %}
```

```
<!-- Include Bootstrap CSS from CDN -->
```

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

```
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
```

```
{% endblock %}
```

```
{% block content %}
```

```
<div class="container">
```

```
<div class="row justify-content-center">
```

```
<div class="col-md-8">

 <h1 class="text-center">About Us</h1>

 <p class="text-center">We are a company that provides awesome products and
services.</p>

</div>

</div>

<!-- Service Images -->

<div class="row justify-content-center mt-5">

 <div class="col-md-3 text-center">

 <p>Web Development</p>
 </div>

 <div class="col-md-3 text-center">

 <p>Boost Your Skill With ChatGPT</p>
 </div>

 <!-- Add more service images here -->

</div>

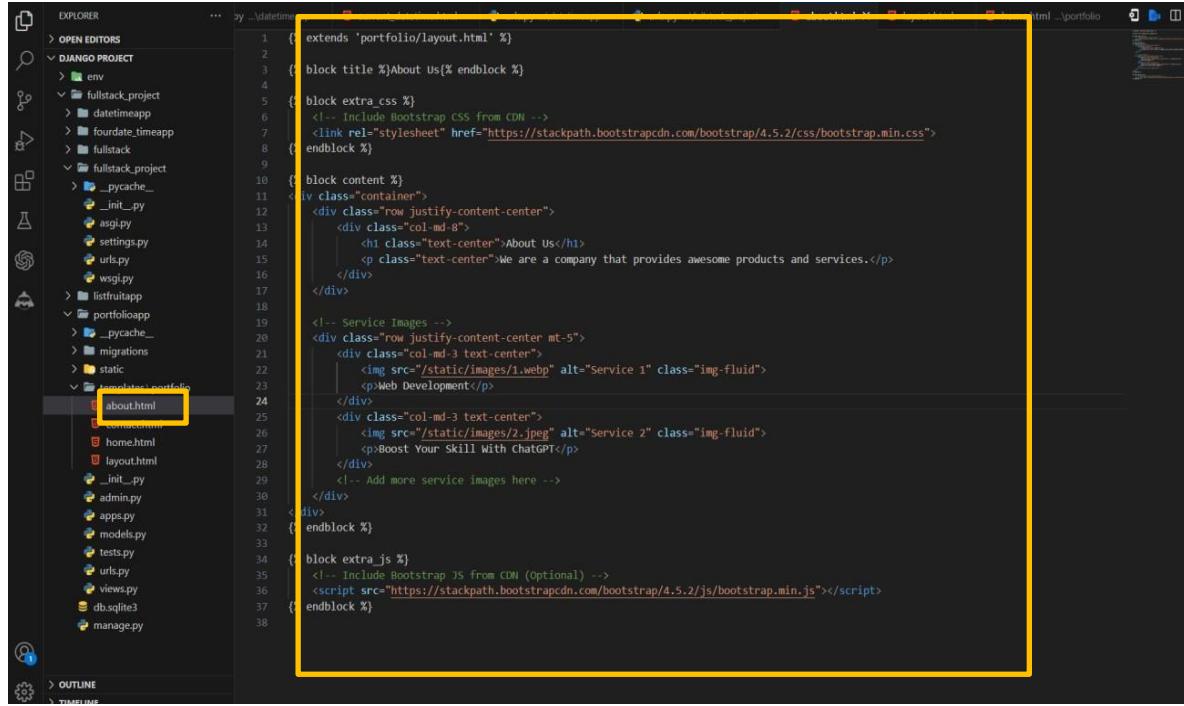
</div>% endblock %}

{ % block extra_js % }
```

<!-- Include Bootstrap JS from CDN (Optional) -->

```
<script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>
```

```
{% endblock %}
```



```

EXPLORER
OPEN EDITORS
DJANGO PROJECT
> env
fullstack_project
> datetimeapp
> foudate_timeapp
> fullstack
> fullstack_project
> _pycache_
> __init__.py
> asgi.py
> settings.py
> urls.py
> wsgi.py
> listruitapp
portfolioapp
> _pycache_
> migrations
> static
> templates/portfolio
 > about.html
 > contact.html
 > home.html
 > layout.html
 > __init__.py
 > admin.py
 > apps.py
 > models.py
 > tests.py
 > urls.py
 > views.py
 db.sqlite3
 manage.py
> OUTLINE
> TIMELINE

about.html

1 {% extends 'portfolio/layout.html' %}
2
3 {% block title %}About Us{% endblock %}
4
5 {% block extra_css %}
6 <!-- Include Bootstrap CSS from CDN -->
7 <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
8 {% endblock %}
9
10 {% block content %}
11 <div class="container">
12 <div class="row justify-content-center">
13 <div class="col-md-8">
14 <h1 class="text-center">About Us</h1>
15 <p class="text-center">We are a company that provides awesome products and services.</p>
16 </div>
17
18 <!-- Service Images -->
19 <div class="row justify-content-center mt-5">
20 <div class="col-md-3 text-center">
21
22 <p>Web Development</p>
23 </div>
24 <div class="col-md-3 text-center">
25
26 <p>Boost Your Skill With ChatGPT</p>
27 </div>
28 <!-- Add more service images here -->
29 </div>
30 </div>
31 {% endblock %}
32
33 {% block extra_js %}
34 <!-- Include Bootstrap JS from CDN (Optional) -->
35 <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>
36 {% endblock %}
37
38

```

## contact.html

```
{% extends 'portfolio/layout.html' %}
```

```
{% block title %}Contact Us{% endblock %}
```

```
{% block extra_css %}
```

<!-- Include Bootstrap CSS from CDN -->

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

```
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
```

```
{% endblock %}
```

```
{% block content %}
```

```
<div class="container">

 <div class="row justify-content-center">

 <div class="col-md-6">

 <h1 class="text-center">Contact Us</h1>

 <p class="text-center">You can reach us at:</p>

 <ul class="list-unstyled">

 <li class="text-center">Email: contact@mywebsite.com

 <li class="text-center">Phone: 123-456-7890

 <li class="text-center">Address: 123 Main Street, City, State

 </div>

 </div>

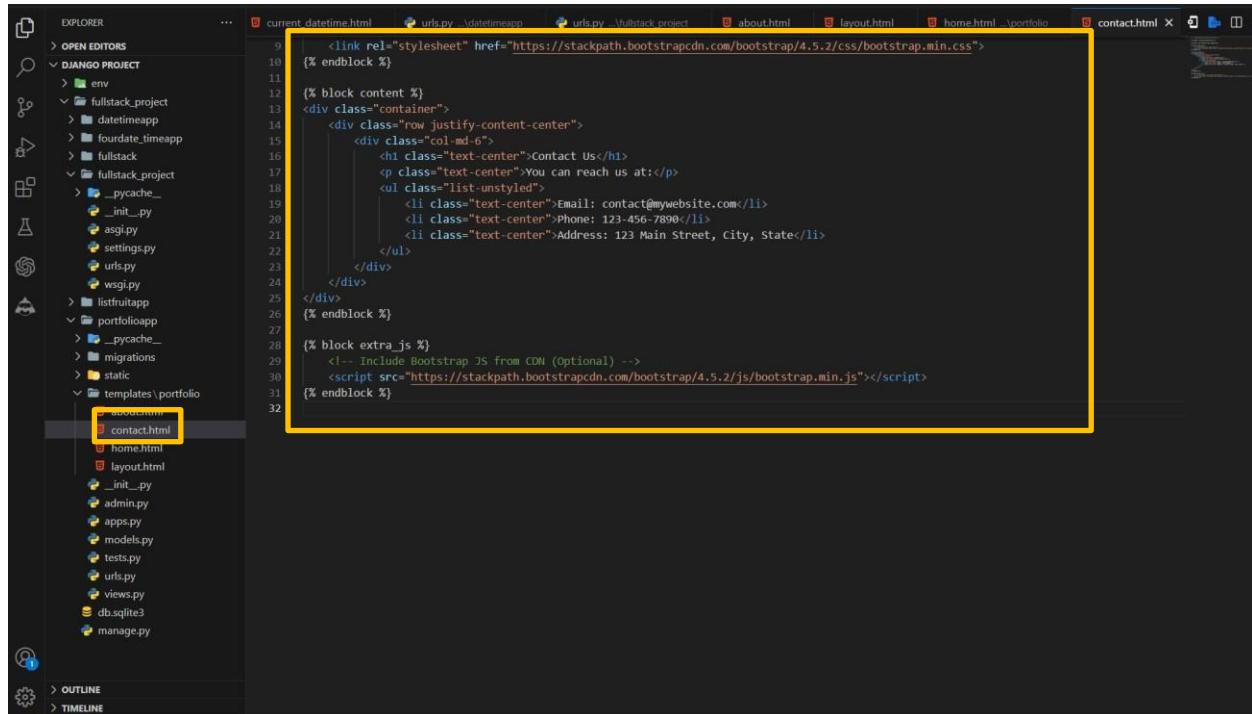
 {% endblock %}

 {% block extra_js %}

 <!-- Include Bootstrap JS from CDN (Optional) -->

 <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>

 {% endblock %}
```



The screenshot shows the VS Code interface with the Django project structure in the Explorer sidebar. The current file open is `contact.html`, which contains the following HTML and Django template code:

```

9 <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
10 {% endblock %}
11
12 {% block content %}
13 <div class="container">
14 <div class="row justify-content-center">
15 <div class="col-md-6">
16 <h1 class="text-center">Contact Us</h1>
17 <p class="text-center">You can reach us at:</p>
18 <ul class="list-unstyled">
19 <li class="text-center">Email: contact@mywebsite.com
20 <li class="text-center">Phone: 123-456-7890
21 <li class="text-center">Address: 123 Main Street, City, State
22
23 </div>
24 </div>
25 {% endblock %}
26
27 {% block extra_js %}
28 <!-- Include Bootstrap JS from CDN (Optional) -->
29 <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>
30
31 {% endblock %}
32

```

## Step-04: Create a view function

- Open the `views.py` file in your Django app's directory (e.g., `portfolioapp/views.py`).

```
from django.shortcuts import render
```

```
def main(request):
```

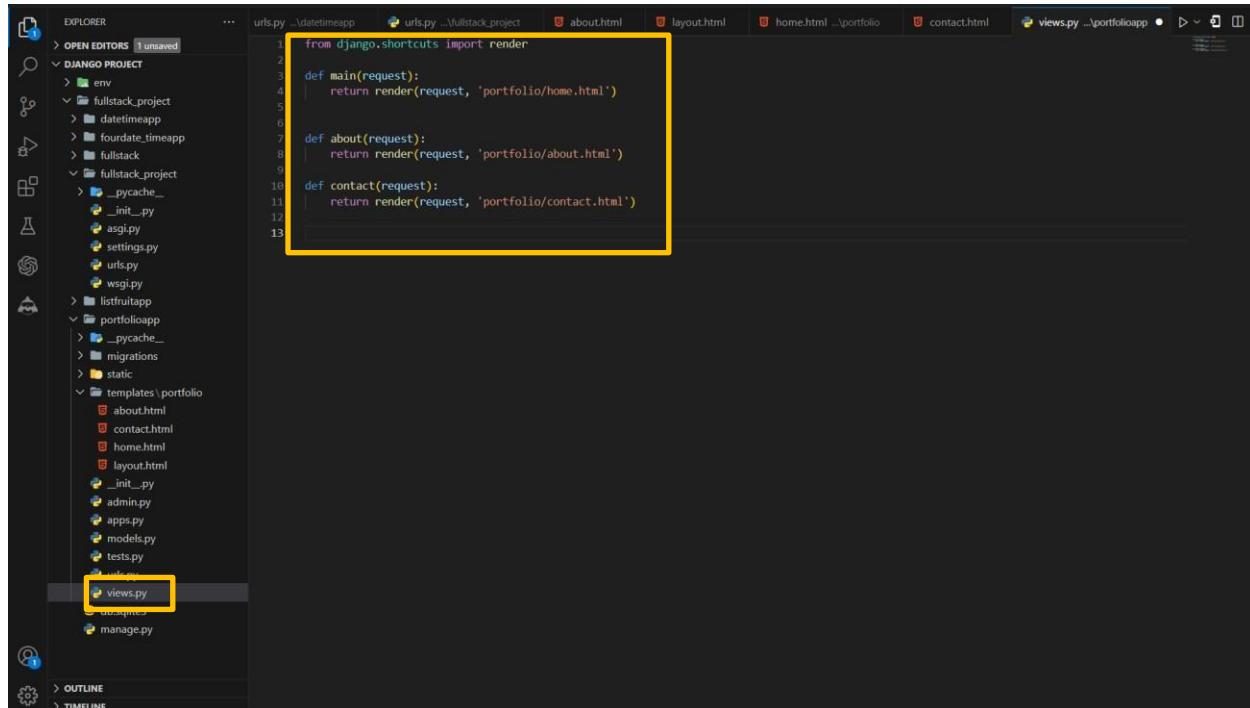
```
 return render(request, 'portfolio/home.html')
```

```
def about(request):
```

```
 return render(request, 'portfolio/about.html')
```

```
def contact(request):
```

```
 return render(request, 'portfolio/contact.html')
```



```

from django.shortcuts import render

def main(request):
 return render(request, 'portfolio/home.html')

def about(request):
 return render(request, 'portfolio/about.html')

def contact(request):
 return render(request, 'portfolio/contact.html')

```

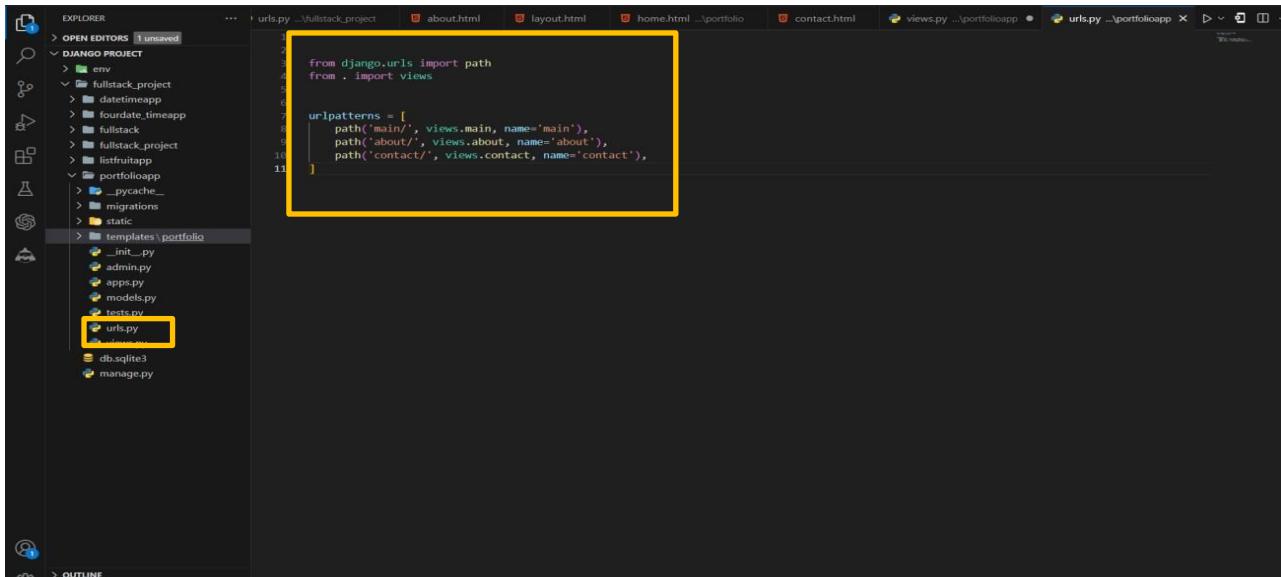
## Step-05: Map the view function to a URL

- Open the **urls.py** file in your Django app's directory (e.g., listfruitapp/urls.py).
- Import the view function at the top of the file
- Add a new URL pattern to the urlpatterns list

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

```
from . import views
```

```
urlpatterns = [
 path('main/', views.main, name='main'),
 path('about/', views.about, name='about'),
 path('contact/', views.contact, name='contact'),
]
```



```

from django.urls import path
from . import views

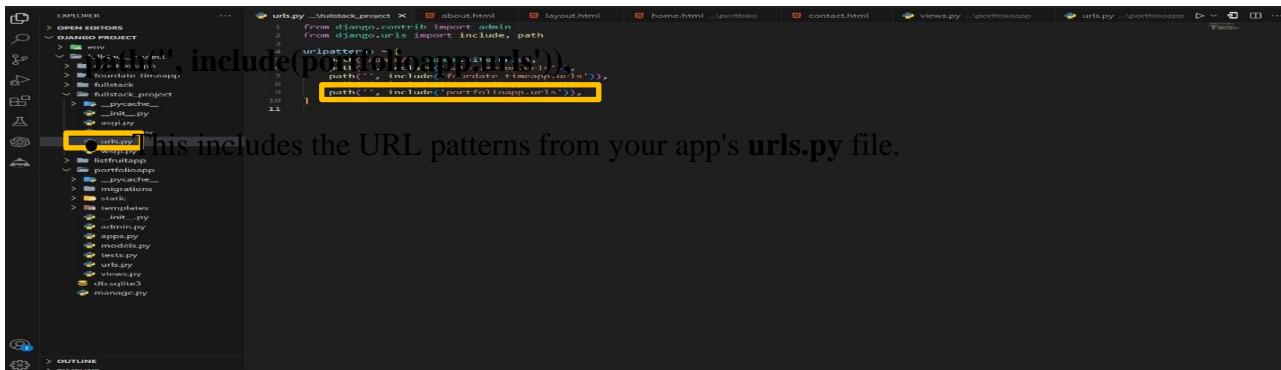
urlpatterns = [
 path('main/', views.main, name='main'),
 path('about/', views.about, name='about'),
 path('contact/', views.contact, name='contact'),
]

```

### Step-06: Include the app's URLs in the project's URL patterns

- Open the **urls.py** file in your project's directory (e.g., **fullstack\_project/urls.py**).
- Import the **include** function from **django.urls** and the **path** function from **django.urls**:
- Add a new URL pattern to the **urlpatterns** list

**from django.urls import include, path [if does not exists]**



```

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

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

```

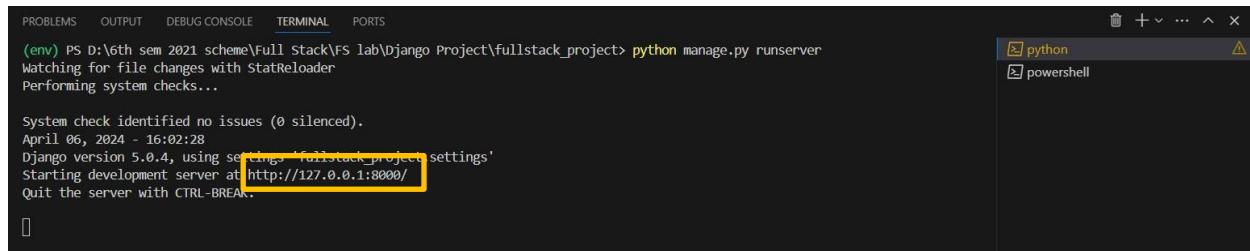
This includes the URL patterns from your app's **urls.py** file.

### **Step-07: 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/>.



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
(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.
```

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

**Final Output By Clicking Above Navbar Home, About Us, Contact Us Buttons**

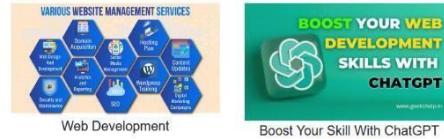


**Fig: Home Page Screen**

Home About Us Contact Us

## About Us

We are a company that provides awesome products and services.



© 2024 Search Creators. Developed by Hanumanthu.

**Fig: About Us Screen**

Home About Us Contact Us

## Contact Us

You can reach us at:

Email: contact@mywebsite.com  
Phone: 123-456-7890  
Address: 123 Main Street, City, State

© 2024 Search Creators. Developed by Hanumanthu.

**Fig: About Us Screen**

## Experiment-07

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.

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

```
[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: 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:

```
35 'django.contrib.auth',
36 'django.contrib.contenttypes',
37 'django.contrib.sessions',
38 'django.contrib.messages',
39 'django.contrib.staticfiles',
40 'fullstack',
41 'datetimeapp',
42 'foundate_timeapp',
43 'listfruitapp',
44 'portfolioapp',
45 student_course_registration_app',
46]
47
48 MIDDLEWARE = [
49 'django.middleware.security.SecurityMiddleware',
50 'django.contrib.sessions.middleware.SessionMiddleware',
51 'django.middleware.common.CommonMiddleware',
52 'django.middleware.csrf.CsrfViewMiddleware',
53 'django.contrib.auth.middleware.AuthenticationMiddleware',
54 'django.contrib.messages.middleware.MessageMiddleware',
55 'django.middleware.clickjacking.XFrameOptionsMiddleware',
56]
57
58 ROOT_URLCONF = 'fullstack_project.urls'
59
60
61 TEMPLATES = [
62 {
63 'BACKEND': 'django.template.backends.django.DjangoTemplates',
64 'DIRS': [],
65 'APP_DIRS': True,
66 'OPTIONS': {
67 'context_processors': [
68 'django.template.context_processors.debug',
69 'django.template.context_processors.request',
70],
71 },
72 },
73]
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
259
260
261
262
263
264
265
266
267
268
269
269
270
271
272
273
274
275
276
277
278
279
279
280
281
282
283
284
285
286
287
287
288
289
289
290
291
292
293
294
295
296
297
297
298
299
299
300
301
302
303
304
305
305
306
307
308
309
309
310
311
312
313
313
314
315
316
316
317
318
318
319
319
320
320
321
321
322
322
323
323
324
324
325
325
326
326
327
327
328
328
329
329
330
330
331
331
332
332
333
333
334
334
335
335
336
336
337
337
338
338
339
339
340
340
341
341
342
342
343
343
344
344
345
345
346
346
347
347
348
348
349
349
350
350
351
351
352
352
353
353
354
354
355
355
356
356
357
357
358
358
359
359
360
360
361
361
362
362
363
363
364
364
365
365
366
366
367
367
368
368
369
369
370
370
371
371
372
372
373
373
374
374
375
375
376
376
377
377
378
378
379
379
380
380
381
381
382
382
383
383
384
384
385
385
386
386
387
387
388
388
389
389
390
390
391
391
392
392
393
393
394
394
395
395
396
396
397
397
398
398
399
399
400
400
401
401
402
402
403
403
404
404
405
405
406
406
407
407
408
408
409
409
410
410
411
411
412
412
413
413
414
414
415
415
416
416
417
417
418
418
419
419
420
420
421
421
422
422
423
423
424
424
425
425
426
426
427
427
428
428
429
429
430
430
431
431
432
432
433
433
434
434
435
435
436
436
437
437
438
438
439
439
440
440
441
441
442
442
443
443
444
444
445
445
446
446
447
447
448
448
449
449
450
450
451
451
452
452
453
453
454
454
455
455
456
456
457
457
458
458
459
459
460
460
461
461
462
462
463
463
464
464
465
465
466
466
467
467
468
468
469
469
470
470
471
471
472
472
473
473
474
474
475
475
476
476
477
477
478
478
479
479
480
480
481
481
482
482
483
483
484
484
485
485
486
486
487
487
488
488
489
489
490
490
491
491
492
492
493
493
494
494
495
495
496
496
497
497
498
498
499
499
500
500
501
501
502
502
503
503
504
504
505
505
506
506
507
507
508
508
509
509
510
510
511
511
512
512
513
513
514
514
515
515
516
516
517
517
518
518
519
519
520
520
521
521
522
522
523
523
524
524
525
525
526
526
527
527
528
528
529
529
530
530
531
531
532
532
533
533
534
534
535
535
536
536
537
537
538
538
539
539
540
540
541
541
542
542
543
543
544
544
545
545
546
546
547
547
548
548
549
549
550
550
551
551
552
552
553
553
554
554
555
555
556
556
557
557
558
558
559
559
560
560
561
561
562
562
563
563
564
564
565
565
566
566
567
567
568
568
569
569
570
570
571
571
572
572
573
573
574
574
575
575
576
576
577
577
578
578
579
579
580
580
581
581
582
582
583
583
584
584
585
585
586
586
587
587
588
588
589
589
590
590
591
591
592
592
593
593
594
594
595
595
596
596
597
597
598
598
599
599
600
600
601
601
602
602
603
603
604
604
605
605
606
606
607
607
608
608
609
609
610
610
611
611
612
612
613
613
614
614
615
615
616
616
617
617
618
618
619
619
620
620
621
621
622
622
623
623
624
624
625
625
626
626
627
627
628
628
629
629
630
630
631
631
632
632
633
633
634
634
635
635
636
636
637
637
638
638
639
639
640
640
641
641
642
642
643
643
644
644
645
645
646
646
647
647
648
648
649
649
650
650
651
651
652
652
653
653
654
654
655
655
656
656
657
657
658
658
659
659
660
660
661
661
662
662
663
663
664
664
665
665
666
666
667
667
668
668
669
669
670
670
671
671
672
672
673
673
674
674
675
675
676
676
677
677
678
678
679
679
680
680
681
681
682
682
683
683
684
684
685
685
686
686
687
687
688
688
689
689
690
690
691
691
692
692
693
693
694
694
695
695
696
696
697
697
698
698
699
699
700
700
701
701
702
702
703
703
704
704
705
705
706
706
707
707
708
708
709
709
710
710
711
711
712
712
713
713
714
714
715
715
716
716
717
717
718
718
719
719
720
720
721
721
722
722
723
723
724
724
725
725
726
726
727
727
728
728
729
729
730
730
731
731
732
732
733
733
734
734
735
735
736
736
737
737
738
738
739
739
740
740
741
741
742
742
743
743
744
744
745
745
746
746
747
747
748
748
749
749
750
750
751
751
752
752
753
753
754
754
755
755
756
756
757
757
758
758
759
759
760
760
761
761
762
762
763
763
764
764
765
765
766
766
767
767
768
768
769
769
770
770
771
771
772
772
773
773
774
774
775
775
776
776
777
777
778
778
779
779
780
780
781
781
782
782
783
783
784
784
785
785
786
786
787
787
788
788
789
789
790
790
791
791
792
792
793
793
794
794
795
795
796
796
797
797
798
798
799
799
800
800
801
801
802
802
803
803
804
804
805
805
806
806
807
807
808
808
809
809
810
810
811
811
812
812
813
813
814
814
815
815
816
816
817
817
818
818
819
819
820
820
821
821
822
822
823
823
824
824
825
825
826
826
827
827
828
828
829
829
830
830
831
831
832
832
833
833
834
834
835
835
836
836
837
837
838
838
839
839
840
840
841
841
842
842
843
843
844
844
845
845
846
846
847
847
848
848
849
849
850
850
851
851
852
852
853
853
854
854
855
855
856
856
857
857
858
858
859
859
860
860
861
861
862
862
863
863
864
864
865
865
866
866
867
867
868
868
869
869
870
870
871
871
872
872
873
873
874
874
875
875
876
876
877
877
878
878
879
879
880
880
881
881
882
882
883
883
884
884
885
885
886
886
887
887
888
888
889
889
890
890
891
891
892
892
893
893
894
894
895
895
896
896
897
897
898
898
899
899
900
900
901
901
902
902
903
903
904
904
905
905
906
906
907
907
908
908
909
909
910
910
911
911
912
912
913
913
914
914
915
915
916
916
917
917
918
918
919
919
920
920
921
921
922
922
923
923
924
924
925
925
926
926
927
927
928
928
929
929
930
930
931
931
932
932
933
933
934
934
935
935
936
936
937
937
938
938
939
939
940
940
941
941
942
942
943
943
944
944
945
945
946
946
947
947
948
948
949
949
950
950
951
951
952
952
953
953
954
954
955
955
956
956
957
957
958
958
959
959
960
960
961
961
962
962
963
963
964
964
965
965
966
966
967
967
968
968
969
969
970
970
971
971
972
972
973
973
974
974
975
975
976
976
977
977
978
978
979
979
980
980
981
981
982
982
983
983
984
984
985
985
986
986
987
987
988
988
989
989
990
990
991
991
992
992
993
993
994
994
995
995
996
996
997
997
998
998
999
999
1000
1000
1001
1001
1002
1002
1003
1003
1004
1004
1005
1005
1006
1006
1007
1007
1008
1008
1009
1009
1010
1010
1011
1011
1012
1012
1013
1013
1014
1014
1015
1015
1016
1016
1017
1017
1018
1018
1019
1019
1020
1020
1021
1021
1022
1022
1023
1023
1024
1024
1025
1025
1026
1026
1027
1027
1028
1028
1029
1029
1030
1030
1031
1031
1032
1032
1033
1033
1034
1034
1035
1035
1036
1036
1037
1037
1038
1038
1039
1039
1040
1040
1041
1041
1042
1042
1043
1043
1044
1044
1045
1045
1046
1046
1047
1047
1048
1048
1049
1049
1050
1050
1051
1051
1052
1052
1053
1053
1054
1054
1055
1055
1056
1056
1057
1057
1058
1058
1059
1059
1060
1060
1061
1061
1062
1062
1063
1063
1064
1064
1065
1065
1066
1066
1067
1067
1068
1068
1069
1069
1070
1070
1071
1071
1072
1072
1073
1073
1074
1074
1075
1075
1076
1076
1077
1077
1078
1078
1079
1079
1080
1080
1081
1081
1082
1082
1083
1083
1084
1084
1085
1085
1086
1086
1087
1087
1088
1088
1089
1089
1090
1090
1091
1091
1092
1092
1093
1093
1094
1094
1095
1095
1096
1096
1097
1097
1098
1098
1099
1099
1100
1100
1101
1101
1102
1102
1103
1103
1104
1104
1105
1105
1106
1106
1107
1107
1108
1108
1109
1109
1110
1110
1111
1111
1112
1112
1113
1113
1114
1114
1115
1115
1116
1116
1117
1117
1118
1118
1119
1119
1120
1120
1121
1121
1122
1122
1123
1123
1124
1124
1125
1125
1126
1126
1127
1127
1128
1128
1129
1129
1130
1130
1131
1131
1132
1132
1133
1133
1134
1134
1135
1135
1136
1136
1137
1137
1138
1138
1139
1139
1140
1140
1141
1141
1142
1142
1143
1143
1144
1144
1145
1145
1146
1146
1147
1147
1148
1148
1149
1149
1150
1150
1151
1151
1152
1152
1153
1153
1154
1154
1155
1155
1156
1156
1157
1157
1158
1158
1159
1159
1160
1160
1161
1161
1162
1162
1163
1163
1164
1164
1165
1165
1166
1166
1167
1167
1168
1168
1169
1169
1170
1170
1171
1171
1172
1172
1173
1173
1174
1174
1175
1175
1176
1176
1177
1177
1178
1178
1179
1179
1180
1180
1181
1181
1182
1182
1183
1183
1184
1184
1185
1185
1186
1186
1187
1187
1188
1188
1189
1189
1190
1190
1191
1191
1192
1192
1193
1193
1194
1194
1195
1195
1196
1196
1197
1197
1198
1198
1199
1199
1200
1200
1201
1201
1202
1202
1203
1203
1204
1204
1205
1205
1206
1206
1207
1207
1208
1208
1209
1209
1210
1210
1211
1211
1212
1212
1213
1213
1214
1214
1215
1215
1216
1216
1217
1217
1218
1218
1219
1219
1220
1220
1221
1221
1222
1222
1223
1223
1224
1224
1225
1225
1226
1226
1227
1227
1228
1228
1229
1229
1230
1230
1231
1231
1232
1232
1233
1233
1234
1234
1235
1235
1236
1236
1237
1237
1238
1238
1239
1239
1240
1240
1241
1241
1242
1242
1243
1243
1244
1244
1245
1245
1246
1246
1247
1247
1248
1248
1249
1249
1250
1250
1251
1251
1252
1252
1253
1253
1254
1254
1255
1255
1256
1256
1257
1257
1258
1258
1259
1259
1260
1260
1261
1261
1262
1262
1263
1263
1264
1264
1265
1265
1266
1266
1267
1267
1268
1268
1269
1269
1270
1270
1271
1271
1272
1272
1273
1273
1274
1274
1275
1275
1276
1276
1277
1277
1278
1278
1279
1279
1280
1280
1281
1281
1282
1282
1283
1283
1284
1284
1285
1285
1286
1286
1287
1287
1288
1288
1289
1289
1290
1290
1291
1291
1292
1292
1293
1293
1294
1294
1295
1295
1296
1296
1297
1297
1298
1298
1299
1299
1300
1300
1301
1301
1302
1302
1303
1303
1304
1304
1305
1305
1306
1306
1307
1307
1308
1308
1309
1309
1310
1310
1311
1311
1312
1312
1313
1313
1314
1314
1315
1315
1316
1316
1317
1317
1318
1318
1319
1319
1320
1320
1321
1321
1322
1322
1323
1323
1324
1324
1325
1325
1326
1326
1327
1327
1328
1328
1329
1329
1330
1330
1331
1331
1332
1332
1333
1333
1334
1334
1335
1335
1336
1336
1337
1337
1338
1338
1339
1339
1340
1340
1341
1341
1342
1342
1343
1343
1344
1344
1345
1345
1346
1346
1347
1347
1348
1348
1349
1349
1350
1350
1351
1351
1352
1352
1353
1353
1354
1354
1355
1355
1356
1356
1357
1357
1358
1358
1359
1359
1360
1360
1361
1361
1362
1362
1363
1363
1364
1364
1365
1365
1366
1366
1367
1367
1368
1368
1369
1369
1370
1370
1371
1371
1372
1372
1373
1373
1374
1374
1375
1375
1376
1376
1377
1377
1378
1378
1379
1379
1380
1380
1381
1381
1382
1382
1383
1383
1384
1384
1385
1385
1386
1386
1387
1387
1388
1388
1389
1389
1390
1390
1391
1391
1392
1392
1393
1393
1394
1394
1395
1395
1396
1396
1397
1397
1398
1398
1399
1399
1400
1400
1401
1401
1402
1402
1403
1403
1404
1404
1405
1405
1406
1406
1407
1407
1408
1408
1409
1409
1410
1410
1411
1411
1412
1412
1413
1413
1414
1414
1415
1415
1416
1416
1417
1417
1418
1418
1419
1419
1420
1420
1421
1421
1422
1422
1423
1423
1424
1424
1425
1425
1426
1426
1427
1427
1428
1428
1429
1429
1430
1430
1431
1431
1432
1432
1433
1433
1434
1434
1435
1435
1436
1436
1437
1437
1438
1438
1439
1439
1440
1440
1441
1441
1442
1442
1443
1443
1444
1444
1445
1445
1446
1446
1447
1447
1448
1448
1449
1449
1450
1450
1451
1451
1452
1452
1453
1453
1454
1454
1455
1455
1456
1456
1457
1457
1458
1458
1459
1459
1460
1460
1461
1461
1462
1462
1463
1463
1464
1464
1465
1465
1466
1466
1467
1467
1468
1468
1469
1469
1470
1470
1471
1471
1472
1472
1473
1473
1474
1474
1475
1475
1476
1476
1477
1477
1478
1478
1479
1479
1480
1480
1481
1481
1482
1482
1483
1483
1484
1484
1485
1485
1486
1486
1487
1487
1488
1488
1489
1489
1490
1490
1491
1491
1492
1492
1493
1493
1494
1494
1495
1495
1496
1496
1497
1497
1498
1498
1499
1499
1500
1500
1501
1501
1502
1502
1503
1503
1504
1504
1505
1505
1506
1506
1507
1507
1508
1508
1509
1509
1510
1510
1511
1511
1512
1512
1513
1513
1514
1514
1515
1515
1516
1516
1517
1517
1518
1518
1519
1519
1520
1520
1521
1521
1522
1522
1523
1523
1524
1524
1525
1525
1526
1526
1527
1527
1528
1528
1529
1529
1530
1530
1531
1531
1532
1532
1533
1533
1534
1534
1535
1535
1536
1536
1537
1537
1538
1538
1539
1539
1540
1540
1541
1541
1542
1542
1543
1543
1544
1544
1545
1545
1546
1546
1547
1547
1548
1548
1549
1549
1550
1550
1551
1551
1552
1552
1553
1553
1554
1554
1555
1555
1556
1556
1557

```

**Step-03: Create models**

- Open the **models.py** file inside the student\_course\_registration\_app and define your models:

```
from django.db import models
```

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

```
 name = models.CharField(max_length=255)
```

```
 description = models.TextField(blank=True, null=True)
```

```
 def __str__(self):
```

```
 return self.name
```

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

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

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

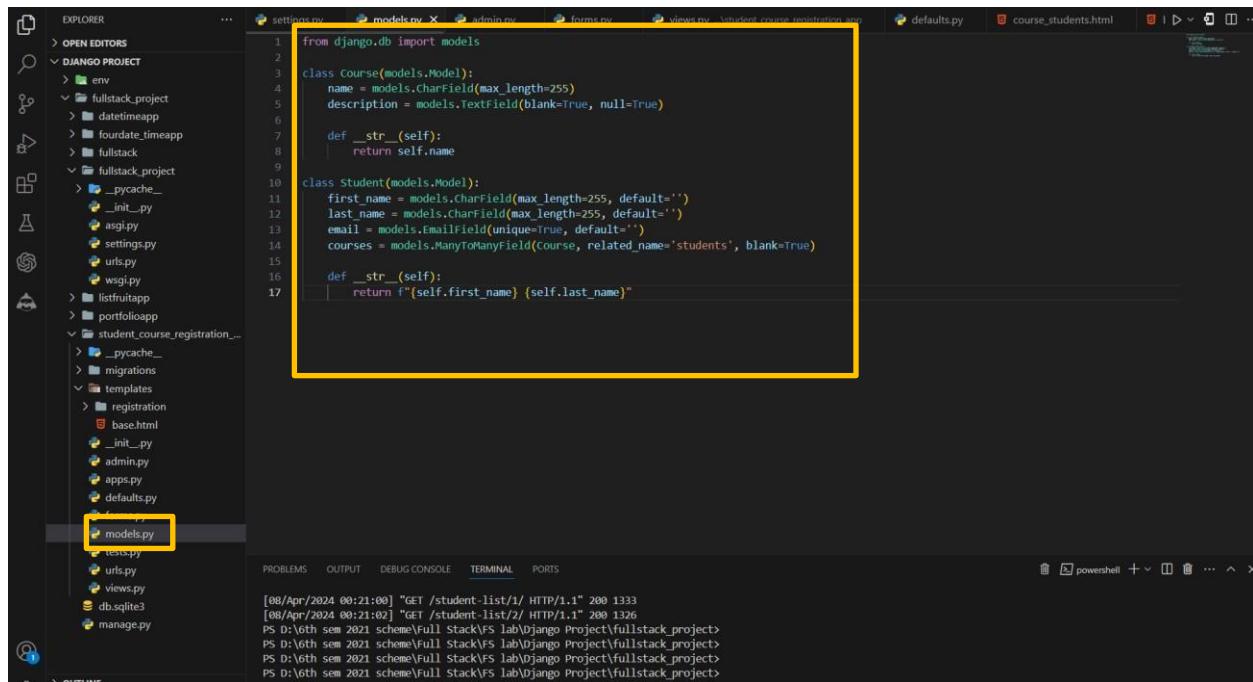
```
 email = models.EmailField(unique=True, default='')
```

```
 courses = models.ManyToManyField(Course, related_name='students', blank=True)
```

```
 - - -
```

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

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



```

1 from django.db import models
2
3 class Course(models.Model):
4 name = models.CharField(max_length=255)
5 description = models.TextField(blank=True, null=True)
6
7 def __str__(self):
8 return self.name
9
10 class Student(models.Model):
11 first_name = models.CharField(max_length=255, default='')
12 last_name = models.CharField(max_length=255, default='')
13 email = models.EmailField(unique=True, default='')
14 courses = models.ManyToManyField('Course', related_name='students', blank=True)
15
16 def __str__(self):
17 return f'{self.first_name} {self.last_name}'

```

#### Step-04: Register models in the admin site

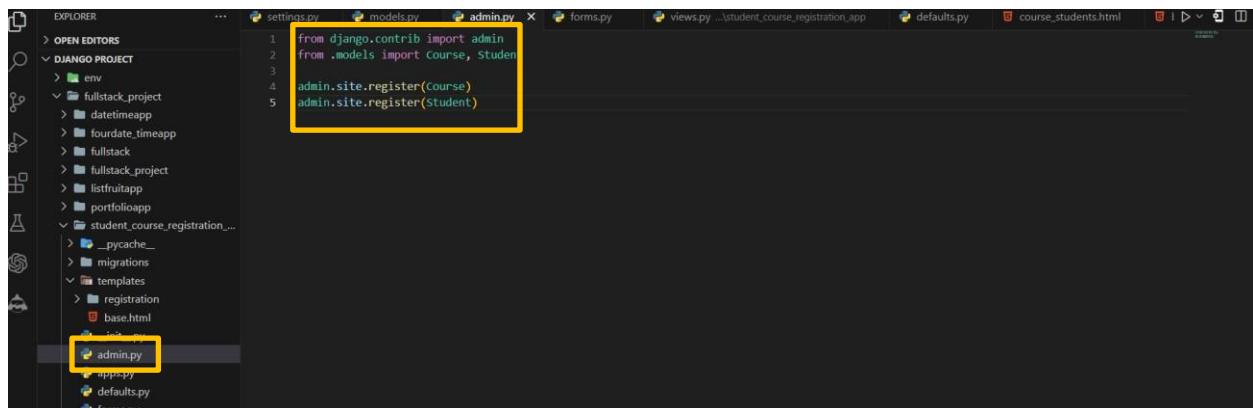
- Open the **admin.py** file inside the `student_course_registration_app` and register your models:

```
from django.contrib import admin
```

```
from .models import Course, Student
```

```
admin.site.register(Course)
```

```
admin.site.register(Student)
```



```

1 from django.contrib import admin
2 from .models import course, Student
3
4 admin.site.register(course)
5 admin.site.register(Student)

```

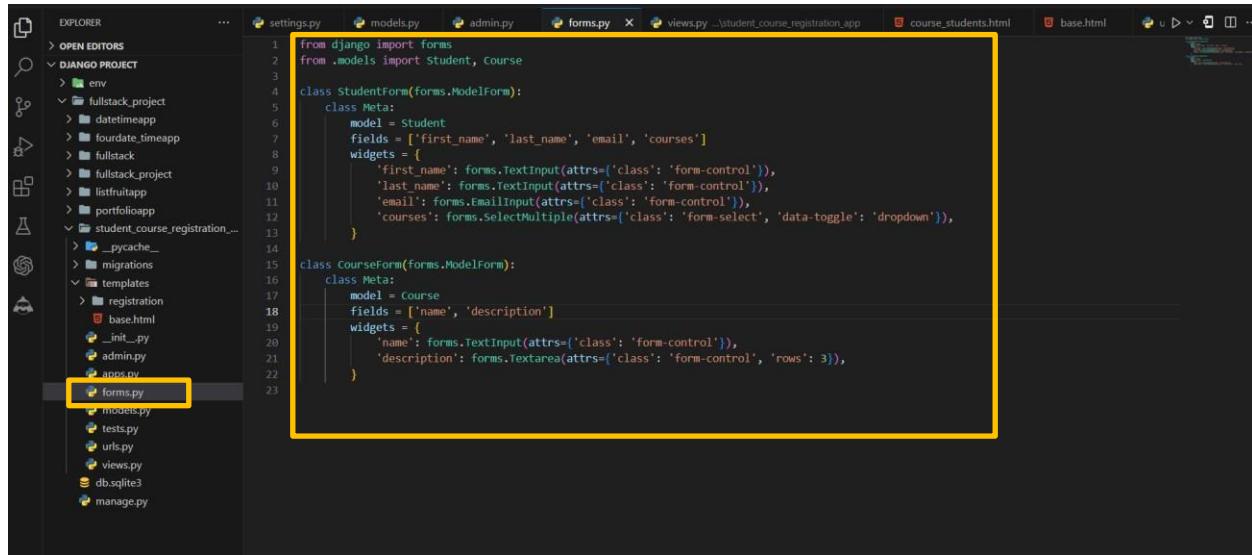
**Step-05: Create forms**

- Create a new file called **forms.py** inside the student\_course\_registration\_app and define your forms:

```
from django import forms
from .models import Student, Course

class StudentForm(forms.ModelForm):
 class Meta:
 model = Student
 fields = ['first_name', 'last_name', 'email', 'courses']
 widgets = {
 'first_name': forms.TextInput(attrs={'class': 'form-control'}),
 'last_name': forms.TextInput(attrs={'class': 'form-control'}),
 'email': forms.EmailInput(attrs={'class': 'form-control'}),
 'courses': forms.SelectMultiple(attrs={'class': 'form-select', 'data-toggle': 'dropdown'}),
 }

class CourseForm(forms.ModelForm):
 class Meta:
 model = Course
 fields = ['name', 'description']
 widgets = {
 'name': forms.TextInput(attrs={'class': 'form-control'}),
 'description': forms.Textarea(attrs={'class': 'form-control', 'rows': 3}),
 }
```



```

1 from django import forms
2 from .models import Student, Course
3
4 class StudentForm(forms.ModelForm):
5 class Meta:
6 model = student
7 fields = ['first_name', 'last_name', 'email', 'courses']
8 widgets = {
9 'first_name': forms.TextInput(attrs={'class': 'form-control'}),
10 'last_name': forms.TextInput(attrs={'class': 'form-control'}),
11 'email': forms.EmailInput(attrs={'class': 'form-control'}),
12 'courses': forms.SelectMultiple(attrs={'class': 'form-select', 'data-toggle': 'dropdown'}),
13 }
14
15 class CourseForm(forms.ModelForm):
16 class Meta:
17 model = course
18 fields = ['name', 'description']
19 widgets = {
20 'name': forms.TextInput(attrs={'class': 'form-control'}),
21 'description': forms.Textarea(attrs={'class': 'form-control', 'rows': 3}),
22 }
23

```

## Step-06: Create views

Open the **views.py** file inside the **student\_course\_registration\_app** and define your views:

```

from django.shortcuts import render, redirect
from .models import Course, Student
from .forms import StudentForm, CourseForm

def index(request):
 courses = Course.objects.all()
 return render(request, 'registration/index.html', {'courses': courses})

def register_student(request):
 if request.method == 'POST':
 form = StudentForm(request.POST)
 if form.is_valid():
 form.save()
 return redirect('index')
 else:

```

```

form = StudentForm()

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

def register_course(request):

 if request.method == 'POST':

 form = CourseForm(request.POST)

 if form.is_valid():

 form.save()

 return redirect('index')

 else:

 form = CourseForm()

 return render(request, 'registration/register_course.html', {'form': form})

def student_list(request, course_id):

 course = Course.objects.get(id=course_id)

 students = course.students.all()

 return render(request, 'registration/student_list.html', {'students': students, 'course': course})

```

```

from django.shortcuts import render, redirect
from .models import Course, Student
from .forms import StudentForm, CourseForm

def index(request):
 courses = Course.objects.all()
 return render(request, 'registration/index.html', {'courses': courses})

def register_student(request):
 if request.method == 'POST':
 form = StudentForm(request.POST)
 if form.is_valid():
 form.save()
 return redirect('index')
 else:
 form = StudentForm()
 return render(request, 'registration/register_student.html', {'form': form})

def register_course(request):
 if request.method == 'POST':
 form = CourseForm(request.POST)
 if form.is_valid():
 form.save()
 return redirect('index')
 else:
 form = CourseForm()
 return render(request, 'registration/register_course.html', {'form': form})

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

```

## Step-07: Create templates

- Create a new directory called templates inside your student\_course\_registration\_app, and create the following template files:

### index.html

```
{% extends 'base.html' %}

{% block content %}

<div class="container">

<h1 class="mt-5">Courses</h1>

<ul class="list-group mt-3">

 {% for course in courses %}

 <li class="list-group-item">{{ course.name }}

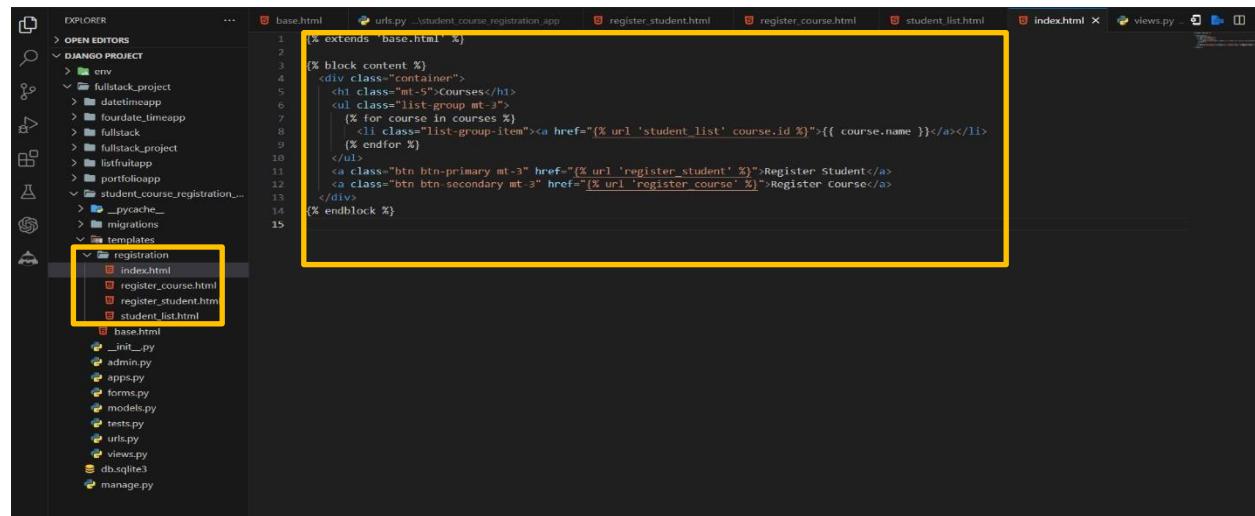
 {% endfor %}

Register Student

Register Course

</div>

{% endblock %}
```



**register\_student.html**

```
<!DOCTYPE html>

<html lang="en">

<head>
 <meta charset="UTF-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <title>Student Registration</title>
 <!-- Bootstrap CSS -->
 <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/css/bootstrap.min.css" rel="stylesheet">
</head>

<body>
 <div class="container">
 <h1>Register Student</h1>
 <form method="post">
 {% csrf_token %}
 {{ form.as_p }}
 <button type="submit" class="btn btn-primary">Register</button>
 </form>
 </div>
 <!-- Bootstrap JavaScript (Optional) -->
 <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/js/bootstrap.bundle.min.js"></script>
 <!-- Your custom JavaScript to initialize the dropdown (Optional) -->
 <script>
 // Example JavaScript to initialize the dropdown
```

```

var dropdownElementList = [].slice.call(document.querySelectorAll('.dropdown-toggle'))

var dropdownList = dropdownElementList.map(function (dropdownToggleEl) {

 return new bootstrap.Dropdown(dropdownToggleEl)

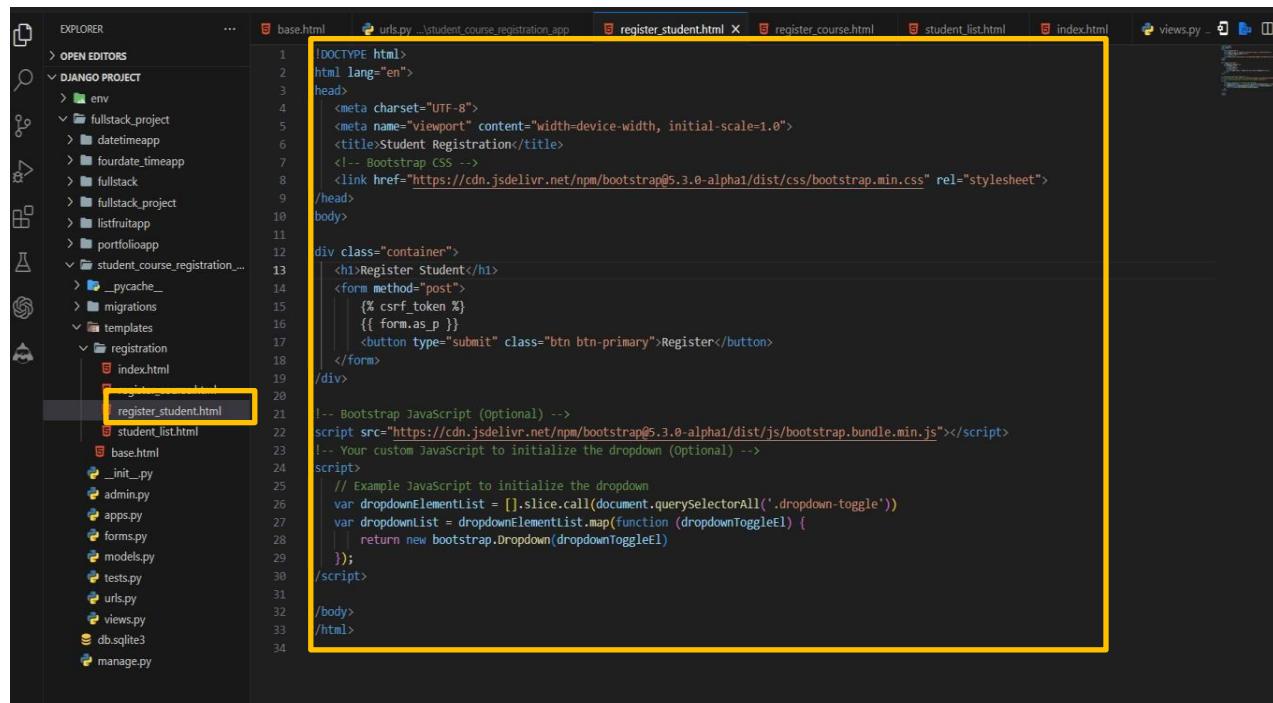
});

</script>

</body>

</html>

```



```

1 !DOCTYPE html>
2 html lang="en">
3 head>
4 <meta charset="UTF-8">
5 <meta name="viewport" content="width=device-width, initial-scale=1.0">
6 <title>Student Registration</title>
7 <!-- Bootstrap CSS -->
8 <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/css/bootstrap.min.css" rel="stylesheet">
9 /head>
10 body>
11
12 div class="container">
13 <h1>Register Student</h1>
14 <form method="post">
15 {% csrf_token %}
16 {{ form.as_p }}
17 <button type="submit" class="btn btn-primary">Register</button>
18 </form>
19 /div>
20
21 // Bootstrap JavaScript (Optional) -->
22 script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/js/bootstrap.bundle.min.js"></script>
23 // Your custom JavaScript to initialize the dropdown (Optional) -->
24 script>
25 // Example JavaScript to initialize the dropdown
26 var dropdownElementList = [].slice.call(document.querySelectorAll('.dropdown-toggle'))
27 var dropdownList = dropdownElementList.map(function (dropdownToggleEl) {
28 return new bootstrap.Dropdown(dropdownToggleEl)
29 });
30 /script>
31
32 /body>
33 /html>

```

## register\_course.html

```

{ % extends 'base.html' % }

{ % load static % }

{ % block content % }

<h1>Register Course</h1>

<div class="container">

<div class="row justify-content-center">

```

```
<div class="col-md-6">

 <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 text-center">Register</button>

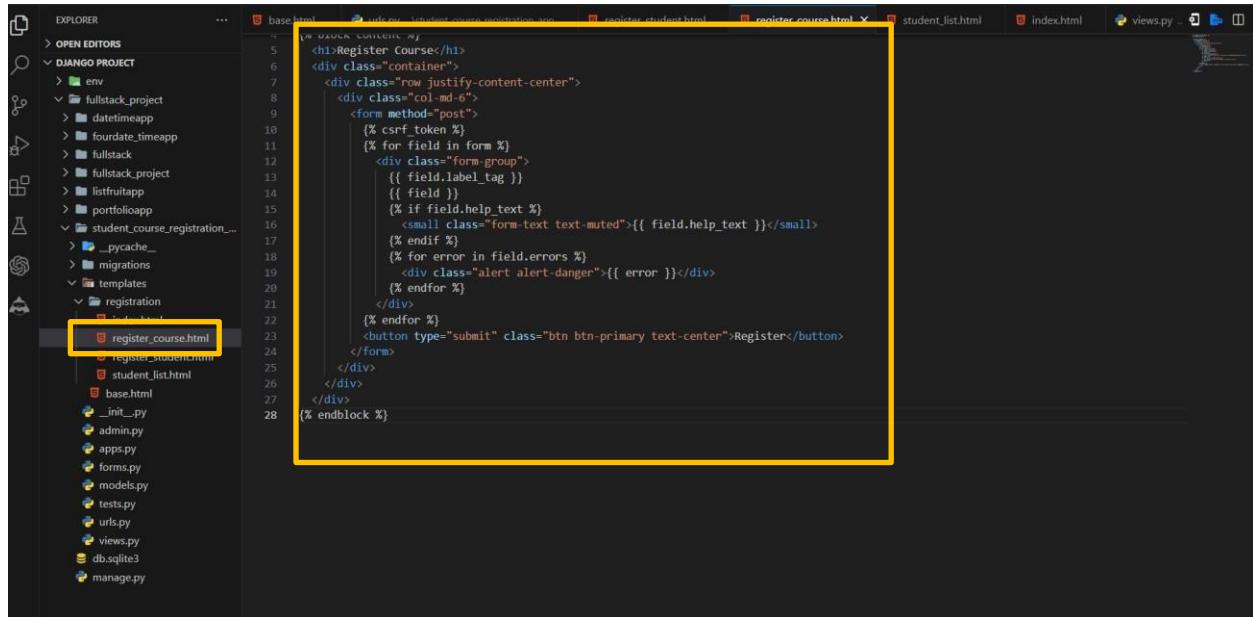
 </form>

 </div>

 </div>

 </div>

 {% endblock %}
```



```

5 <h1>Register Course</h1>
6 <div class="container">
7 <div class="row justify-content-center">
8 <div class="col-md-6">
9 <form method="post">
10 {% csrf_token %}
11 {% for field in form %}
12 <div class="form-group">
13 {{ field.label_tag }}
14 {{ field }}
15 {% if field.help_text %}
16 <small class="form-text text-muted">{{ field.help_text }}</small>
17 {% endif %}
18 {% for error in field.errors %}
19 <div class="alert alert-danger">{{ error }}</div>
20 {% endfor %}
21 </div>
22 {% endfor %}
23 <button type="submit" class="btn btn-primary text-center">Register</button>
24 </form>
25 </div>
26 </div>
27 {% endblock %}

```

## student\_list.html

```

{% extends 'base.html' %}

{% block content %}

<h1>Students Registered for {{ course.name }}</h1>

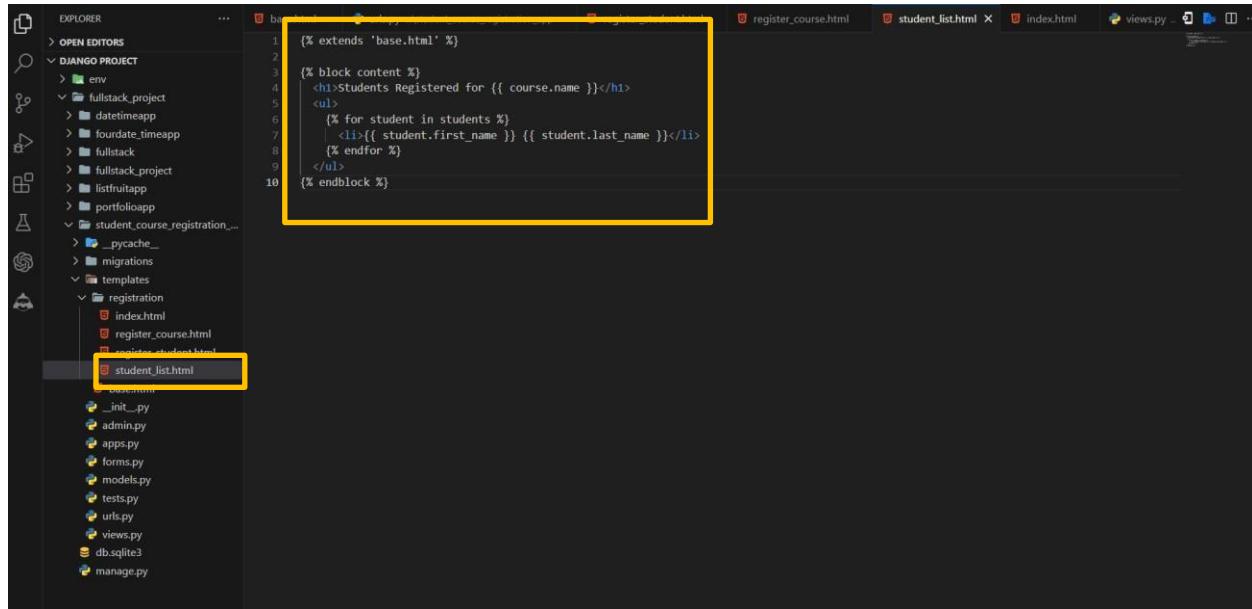
 {% for student in students %}

 {{ student.first_name }} {{ student.last_name }}

 {% endfor %}

{% endblock %}

```



```

1 {% extends 'base.html' %}
2
3 {% block content %}
4 <h1>Students Registered for {{ course.name }}</h1>
5
6 {% for student in students %}
7 {{ student.first_name }} {{ student.last_name }}
8 {% endfor %}
9
10 {% endblock %}

```

### Step-08: Create a base template

- Create a **base.html** file inside the templates directory with the following content:

#### **base.html**

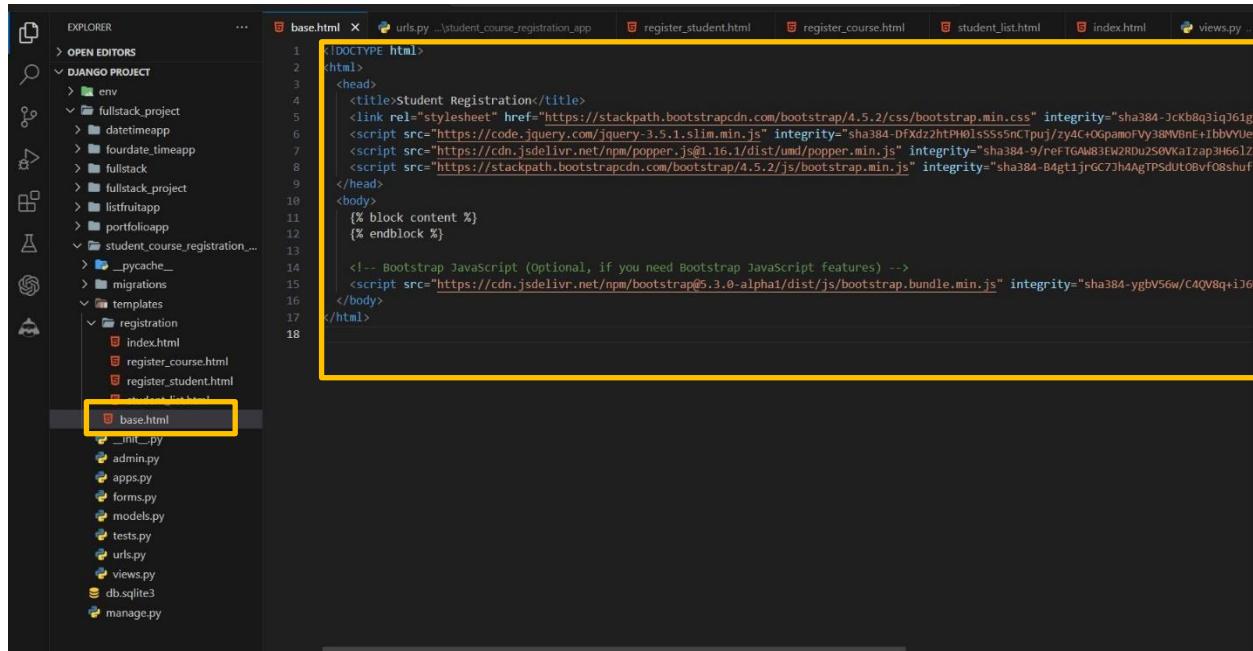
```

<!DOCTYPE html>

<html>
 <head>
 <title>Student Registration</title>
 </head>
 <body>
 {% block content %}

 {% endblock %}
 </body>
</html>

```



```

1 <!DOCTYPE html>
2 <html>
3 <head>
4 <title>Student Registration</title>
5 <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css" integrity="sha384-Jckb8q3iqJ61g
6 <script src="https://code.jquery.com/jquery-3.5.1.slim.min.js" integrity="sha384-Dfxz2htPH0lSS5nCtpuj/zY4C+OgpamoFVy3BMVnE+1bbVVUe
7 <script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.1/dist/umd/popper.min.js" integrity="sha384-9/rEFtGAW83EW2RDU2S0VKaIzapH6612
8 <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js" integrity="sha384-B4gt1jrGC7jh4AgTPSdUtOBvf08shuf
9 </head>
10 <body>
11 {% block content %}
12 {% endblock %}
13
14 <!-- Bootstrap JavaScript (Optional, if you need Bootstrap JavaScript features) -->
15 <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-alpha1/dist/js/bootstrap.bundle.min.js" integrity="sha384-ygbV56w/C4QV8q+1J
16
17 </body>
18 </html>

```

## Step-09: Configure URLs

- Open the urls.py file inside your student\_course\_registration\_app and define your URLs:

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

```
from . import views
```

```
urlpatterns = [
```

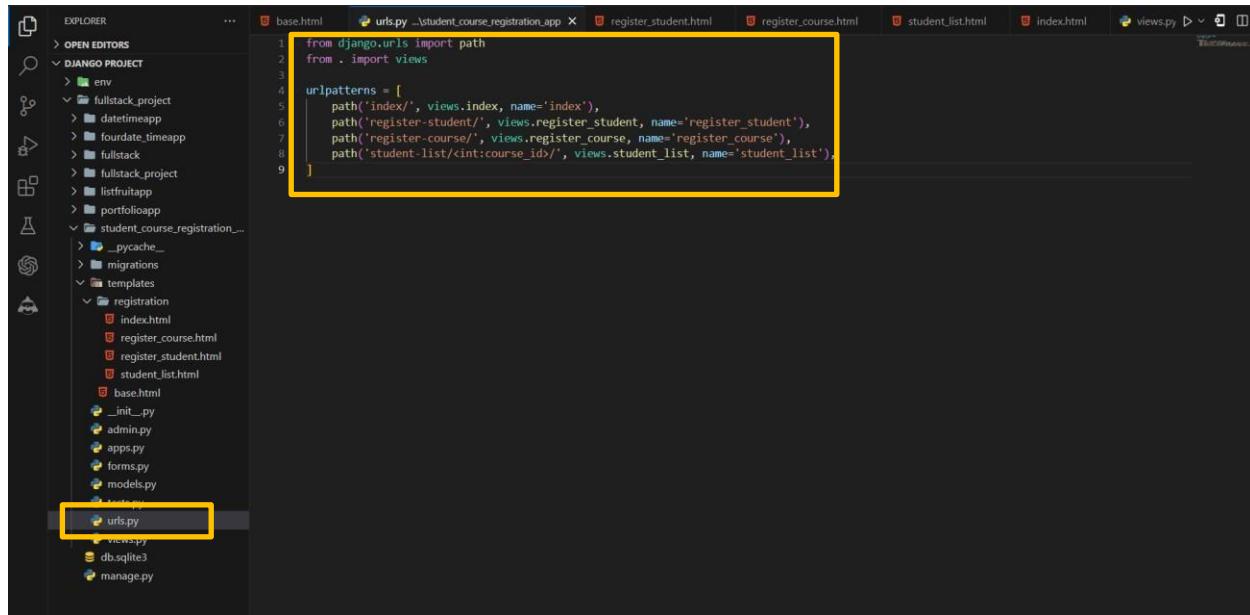
```
 path('index/', views.index, name='index'),
```

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

```
 path('register-course/', views.register_course, name='register_course'),
```

```
 path('student-list/<int:course_id>', views.student_list, name='student_list'),
```

```
]
```



```

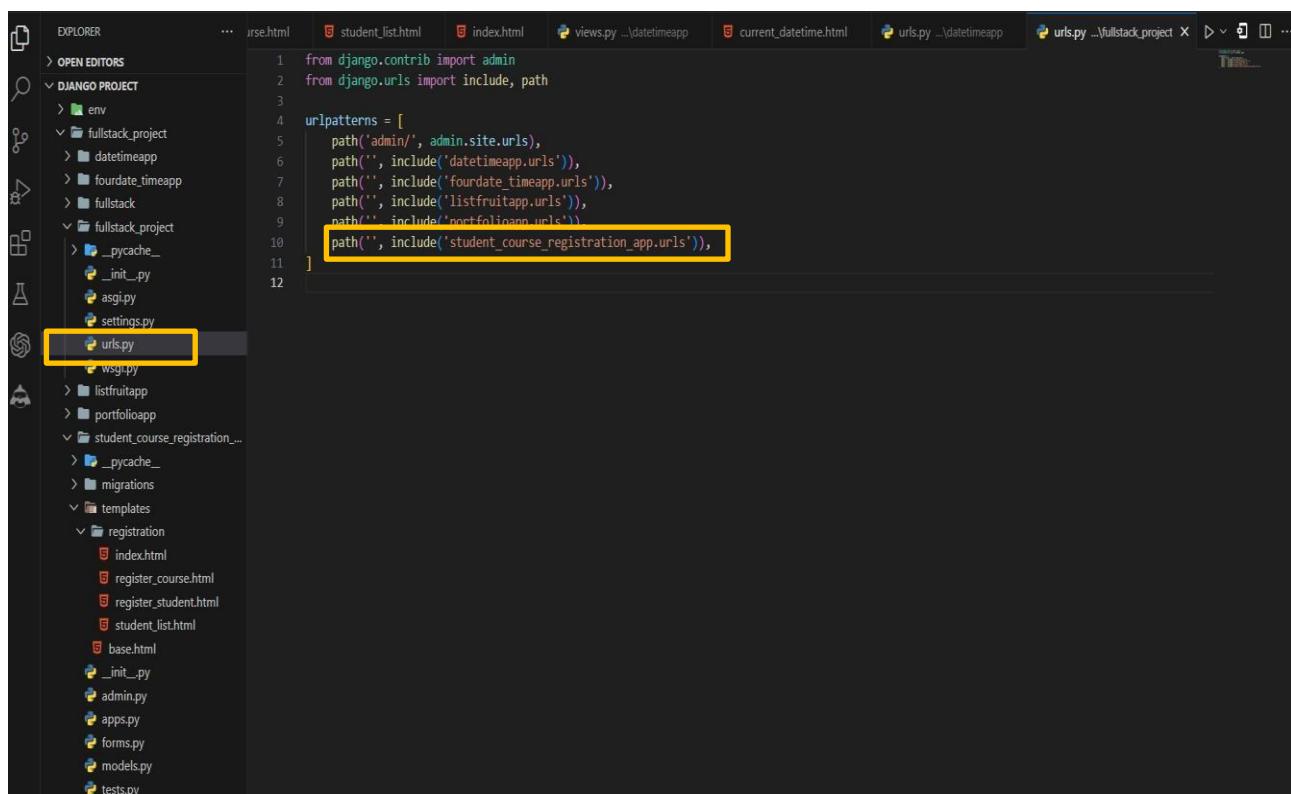
from django.urls import path
from . import views

urlpatterns = [
 path('index/', views.index, name='index'),
 path('register-student/', views.register_student, name='register_student'),
 path('register-course/', views.register_course, name='register_course'),
 path('student-list/int:course_id/', views.student_list, name='student_list')
]

```

## Step-09: Update project URLs

- Open the urls.py file inside your project and include the URLs from the student\_course\_registration\_app:



```

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

urlpatterns = [
 path('admin/', admin.site.urls),
 path('', include('datetimeapp.urls')),
 path('', include('foudate_timeapp.urls')),
 path('', include('listfruitapp.urls')),
 path('', include('portfolioapp.urls')),
 path('', include('student_course_registration_app.urls'))
]

```

**Step-10: Make Migration for check Models saved or not into the database**

```
PS D:\6th sem 2021 scheme\Full Stack\FS lab\ Django Project\fullstack_project> python manage.py makemigrations
Migrations for 'student_course_registration_app':
 student_course_registration_app\migrations\0002_remove_student_name_student_courses_and_more.py
 - Remove field name from student
 - Add field courses to student
 - Add field first_name to student
 - Add field last_name to student
 - Alter field description on course
 - Alter field name on course
 - Alter field email on student
 - Delete model Enrollment
```

**Step-10: Migrate To Save the Models into database**

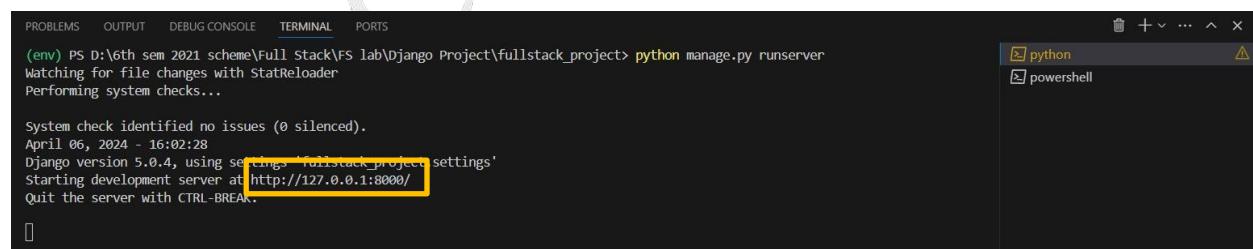
```
PS D:\6th sem 2021 scheme\Full Stack\FS lab\ Django Project\fullstack_project> python manage.py migrate
Operations to perform:
 Apply all migrations: admin, auth, contenttypes, sessions, student_course_registration_app
Running migrations:
 Applying student_course_registration_app.0002_remove_student_name_student_courses_and_more... OK
```

**Step-07: 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**

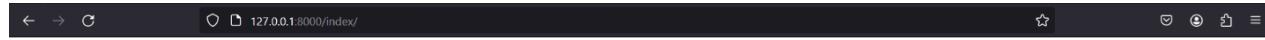
- Open your web browser and visit <http://127.0.0.1:8000/>.



```
(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 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.
```

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

**Final Output By Clicking Register Student and Register Course**

**Fig: Index Screen**

Courses

**Fig: Student Register Screen**A screenshot of a web browser window showing a form titled "Courses". The form has two input fields: "DBMS" and "CN". At the bottom, there are two buttons: "Register Student" (highlighted in blue) and "Register Course". The browser's navigation bar and status bar are also visible.

A screenshot of a web browser showing a form titled "Register Course". The URL in the address bar is 127.0.0.1:8000/register-course/. The form has two fields: "Name:" containing "Software Engineering" and "Description:" containing "Software Engineering it is a every Engineering Student Know About the this Subject". A blue "Register" button is at the bottom.

**Fig: Course Register Screen**

A screenshot of a web browser showing a list titled "Students Registered for DBMS". The URL in the address bar is 127.0.0.1:8000/student-list/1/. The list contains two items: "Hanumanthu hanu" and "Hanumanthu Hanu".

**Fig: Students Register Particular Courses Screen [You can Check as You Register Courses]**

## **Experiment-08**

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

- In this code, we've registered the Course and Student models with the Django admin site.

### **Step-01: Register models in the admin site**

```
from django.contrib import admin

from .models import Course, Student

@admin.register(Course)

class CourseAdmin(admin.ModelAdmin):

 list_display = ('name', 'description')

@admin.register(Student)

class StudentAdmin(admin.ModelAdmin):

 list_display = ('first_name', 'last_name', 'email')

 filter_horizontal = ('courses',)
```

```
from django.contrib import admin
from .models import Course, Student

@admin.register(Course)
class CourseAdmin(admin.ModelAdmin):
 list_display = ('name', 'description')

@admin.register(Student)
class StudentAdmin(admin.ModelAdmin):
 list_display = ('first_name', 'last_name', 'email')
 filter_horizontal = ('courses',)
```

- We've also specified the fields to be displayed in the list view for each model using the `list_display` attribute.
- For the `StudentAdmin` class, we've added the `filter_horizontal` attribute to display the `courses` field (which is a many-to-many relationship) as a horizontal filter in the admin interface.

### **Step-02: Create and apply migrations**

**python manage.py makemigrations**

```
No such file or directory
(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 makemigrations
No changes detected
```

**python manage.py migrate**

```
(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, student_course_registration_app
Running migrations:
 No migrations to apply.
```

### **Step-03: Create a superuser**

- If you haven't already created a superuser for your project, you'll need to do so to access the admin interface.
- In your terminal or command prompt, run the following command

**python manage.py createsuperuser**

```
(env) PS D:\6th sem 2021 scheme\Full Stack\FS lab\Django Project\fullstack_project> python manage.py createsuperuser
Username (leave blank to use 'aryah'): admin
Email address:
Password:
Password (again):
This password is too short. It must contain at least 8 characters.
This password is too common.
This password is entirely numeric.
Bypass password validation and create user anyway? [y/N]: y
Superuser created successfully.
```

**Type Username: admin**

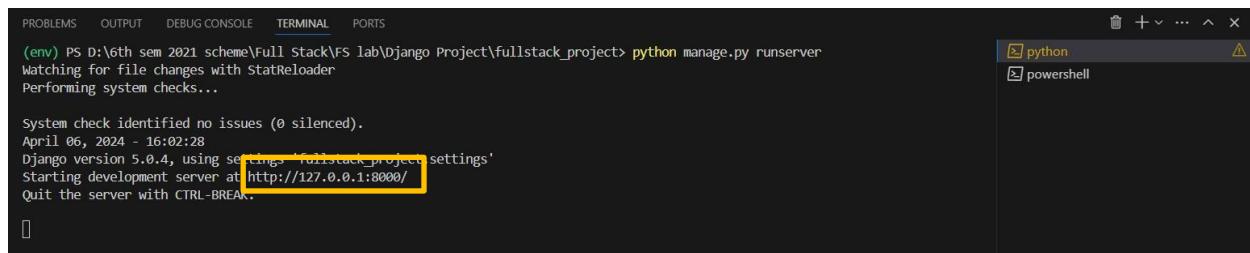
**Password:1234 [For Simply but your wish to create your own username and password]**

## Step-03: Access the admin interface

- Start the Django development server by running the following command:
- 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/>.

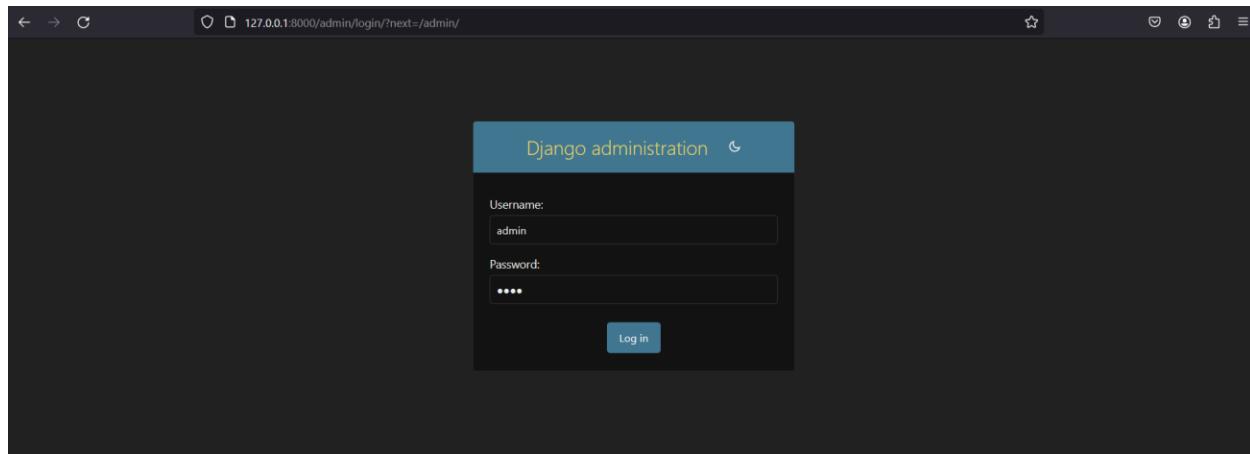


```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
(env) PS D:\6th sem 2021 scheme\Full Stack\FS lab\Django Project\fullstack_project> python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...
System check identified no issues (0 silenced).
April 06, 2024 - 16:02:28
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.
```

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

## Step-04: Log in to the admin interface

- Back in your web browser, enter the superuser credentials you just created and log in to the admin interface.



### Step-05: Add data through admin forms

Once you're logged in to the admin interface, you'll see the "Courses" and "Students" sections in the left sidebar. Click on "Courses" to add a new course.

- Click on the "Add course" button in the top-right corner.

**Fig: Admin Main Screen**

- Fill in the "Name" and "Description" fields for the new course.
- Click the "Save" button to create the new course.

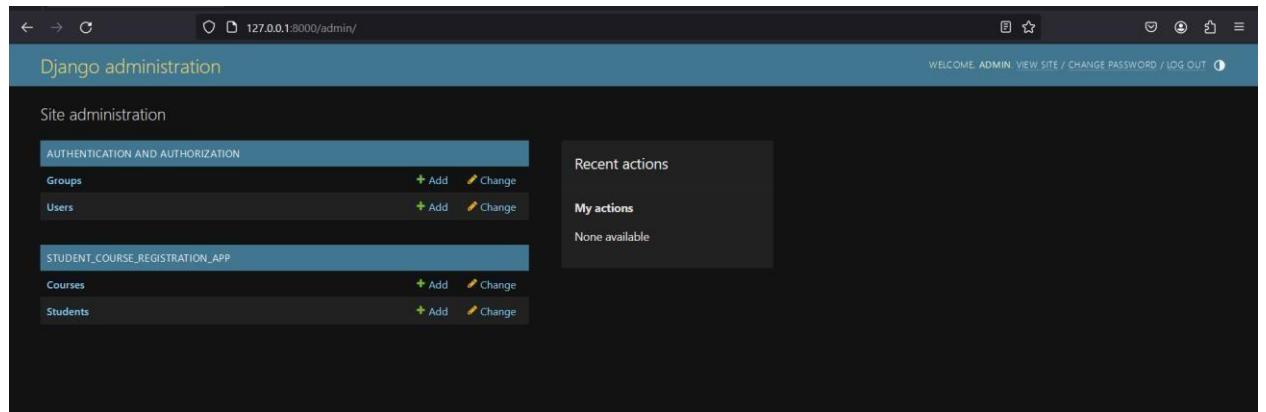
Next, click on "Students" in the left sidebar to add a new student.

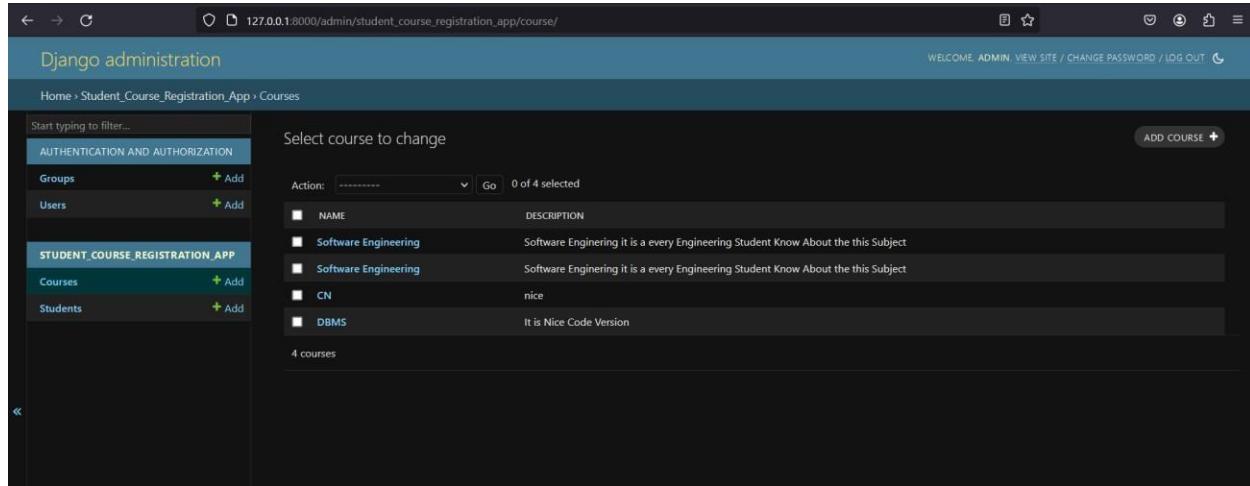
- Click on the "Add student" button in the top-right corner.
- Fill in the "First name", "Last name", and "Email" fields for the new student.
- In the "Courses" section, select the courses you want to enroll the student in by checking the appropriate boxes.
- Click the "Save" button to create the new student.

You can repeat these steps to add more courses and students through the admin interface.

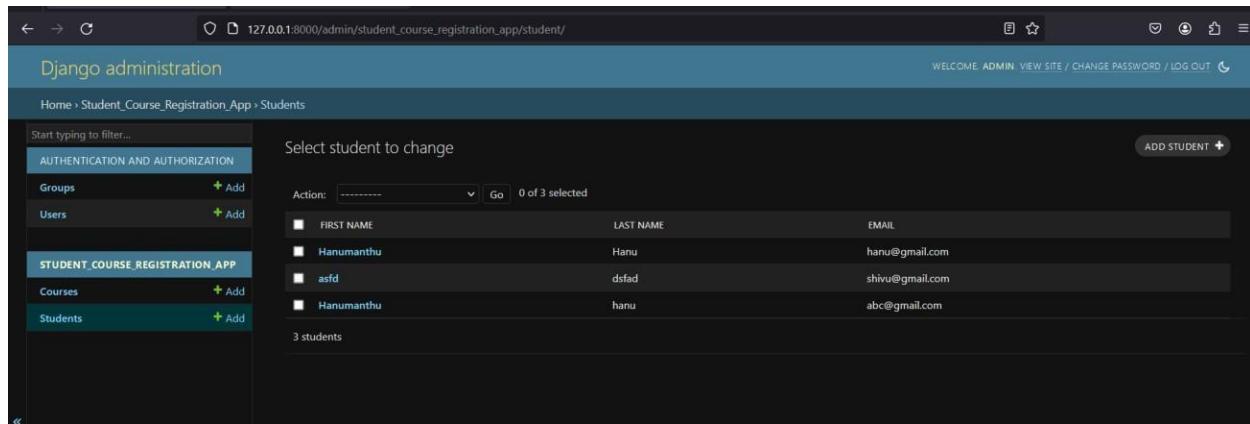
Additionally, you can view, edit, and delete existing courses and students from the admin interface.

### Final Output





**Fig: Admin Course Screen**



**Fig: Admin Student Screen**

## Experiment-09

Develop a Model form for student that contains his topic chosen for project, languages used and duration with a model called project.

### **Step 1: 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 2: Open the `models.py` file in the projects app and define the Project model.**

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

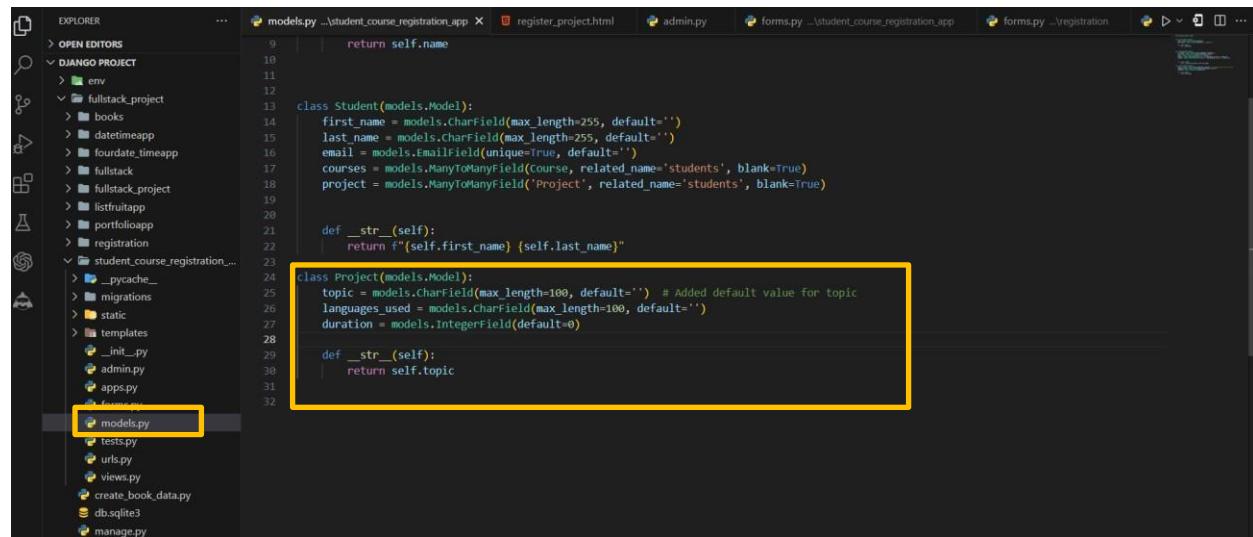
```
topic = models.CharField(max_length=100, default="") # Added default value for topic
```

```
languages_used = models.CharField(max_length=100, default="")
```

```
duration = models.IntegerField(default=0)
```

```
def __str__(self):
```

```
 return self.topic
```



```

class Project(models.Model):
 topic = models.CharField(max_length=100, default="") # Added default value for topic
 languages_used = models.CharField(max_length=100, default="")
 duration = models.IntegerField(default=0)

 def __str__(self):
 return self.topic

```

**Step 3: Create and apply migrations to create the necessary database tables.**

**python manage.py makemigrations**

```
No such file or directory
(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 makemigrations
No changes detected
```

**python manage.py migrate**

```
(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, student_course_registration_app
Running migrations:
 No migrations to apply.
```

**Step 4: Create a `forms.py` file in the projects app and define a `ProjectForm` based on the Project model.**

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

```
 class Meta:
```

```
 model = Project
```

```
 fields = ('topic', 'languages_used', 'duration')
```

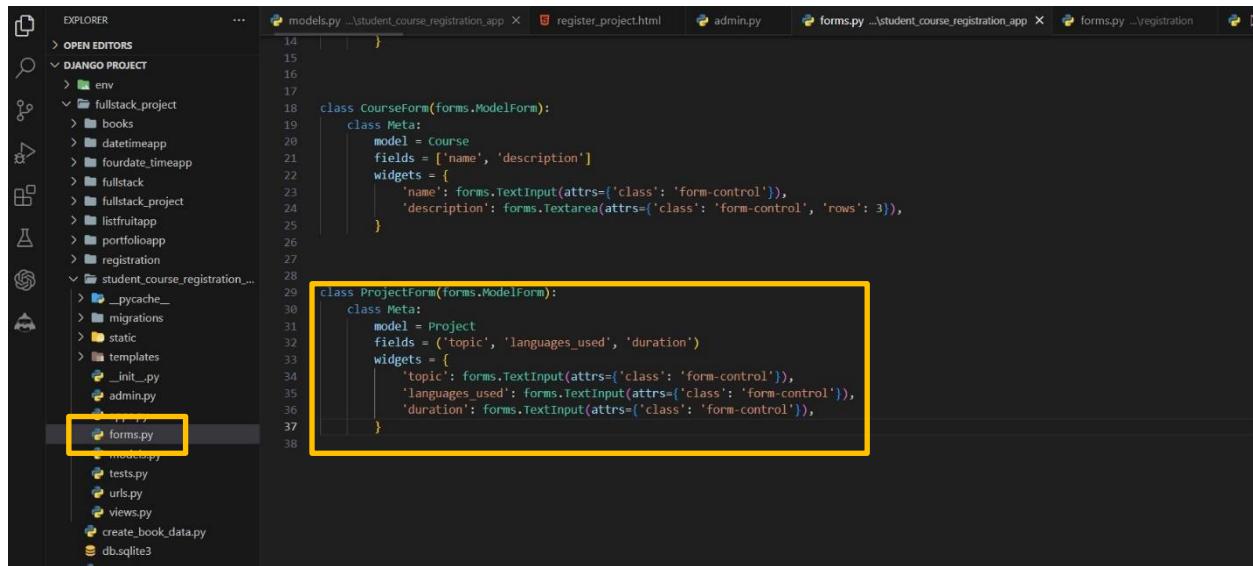
```
 widgets = {
```

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

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

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

```
 }
```



```

models.py ...\\student_course_registration_app X register_project.html admin.py forms.py ...\\student_course_registration_app X forms.py ...\\registration
14
15
16
17
18 class courseForm(forms.ModelForm):
19 class Meta:
20 model = Course
21 fields = ['name', 'description']
22 widgets = {
23 'name': forms.TextInput(attrs={'class': 'form-control'}),
24 'description': forms.Textarea(attrs={'class': 'form-control', 'rows': 3}),
25 }
26
27
28
29 class ProjectForm(forms.ModelForm):
30 class Meta:
31 model = Project
32 fields = ('topic', 'languages_used', 'duration')
33 widgets = {
34 'topic': forms.TextInput(attrs={'class': 'form-control'}),
35 'languages_used': forms.TextInput(attrs={'class': 'form-control'}),
36 'duration': forms.TextInput(attrs={'class': 'form-control'})
37 }
38

```

**Step 5:** In **views.py** view function, import the **ProjectForm** and handle the form submission.

```
from .forms import ProjectForm
```

```
def register_project(request):
```

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

```
 form = ProjectForm(request.POST)
```

```
 if form.is_valid():
```

```
 project = form.save()
```

```
Redirect to a success page or another view
```

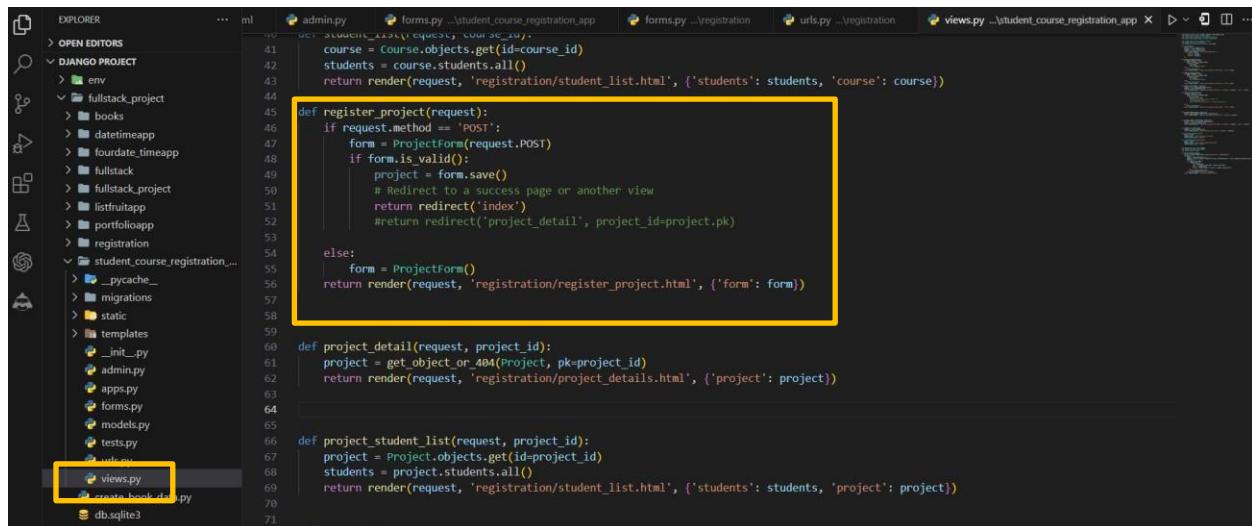
```
 return redirect('index')
```

```
#return redirect('project_detail', project_id=project.pk)
```

```
 else:
```

```
 form = ProjectForm()
```

```
 return render(request, 'registration/register_project.html', {'form': form})
```



```

EXPLORER
OPEN EDITORS
DJANGO PROJECT
> env
fullstack_project
> books
> datetimeapp
> fourdate_timeapp
> fullstack
> listfruitapp
> portfolioapp
> registration
> student_course_registration...
> _pycache_-
migrations
static
templates
__init__.py
admin.py
apps.py
forms.py
models.py
tests.py
urls.py
views.py
books/models.py
db.sqlite3

admin.py forms.py ...student_course_registration_app forms.py ...registration urls.py ...registration views.py ...student_course_registration_app

def register_project(request):
 if request.method == 'POST':
 form = ProjectForm(request.POST)
 if form.is_valid():
 project = form.save()
 # Redirect to a success page or another view
 return redirect('index')
 #return redirect('project_detail', project_id=project.pk)

 else:
 form = ProjectForm()
 return render(request, 'registration/register_project.html', {'form': form})

def project_detail(request, project_id):
 project = get_object_or_404(Project, pk=project_id)
 return render(request, 'registration/project_details.html', {'project': project})

def project_student_list(request, project_id):
 project = Project.objects.get(id=project_id)
 students = project.students.all()
 return render(request, 'registration/student_list.html', {'students': students, 'project': project})

```

**Step 6: In `views.py` index function, add the projects Objects for Displaying the Register Projects .**

`def index(request):`

`courses = Course.objects.all()`

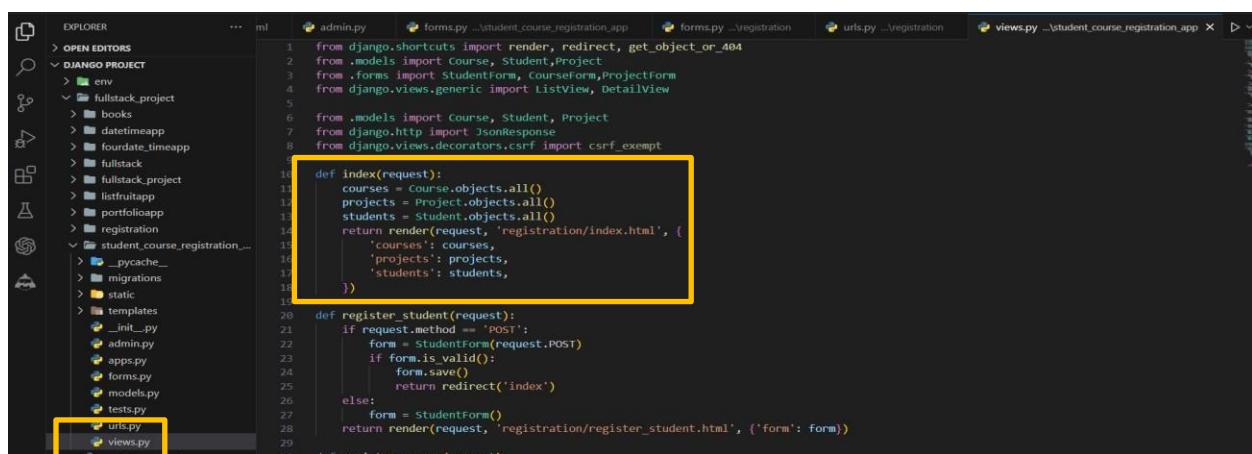
`projects = Project.objects.all()`

`return render(request, 'registration/index.html', {`

`'courses': courses,`

`'projects': projects,`

`)`



```

EXPLORER
OPEN EDITORS
DJANGO PROJECT
> env
fullstack_project
> books
> datetimeapp
> fourdate_timeapp
> fullstack
> listfruitapp
> portfolioapp
> registration
> student_course_registration...
> _pycache_-
migrations
static
templates
__init__.py
admin.py
apps.py
forms.py
models.py
tests.py
urls.py
views.py
books/models.py
db.sqlite3

admin.py forms.py ...student_course_registration_app forms.py ...registration urls.py ...registration views.py ...student_course_registration_app

from django.shortcuts import render, get_object_or_404
from .models import Course, Student, Project
from .forms import StudentForm, CourseForm, ProjectForm
from django.views.generic import ListView, DetailView
from .models import Course, Student, Project
from django.http import JsonResponse
from django.views.decorators.csrf import csrf_exempt

def index(request):
 courses = Course.objects.all()
 projects = Project.objects.all()
 students = Student.objects.all()
 return render(request, 'registration/index.html', {
 'courses': courses,
 'projects': projects,
 'students': students,
 })

def register_student(request):
 if request.method == 'POST':
 form = StudentForm(request.POST)
 if form.is_valid():
 form.save()
 return redirect('index')

 else:
 form = StudentForm()
 return render(request, 'registration/register_student.html', {'form': form})

def register_project(request):
 if request.method == 'POST':
 form = ProjectForm(request.POST)
 if form.is_valid():
 project = form.save()
 # Redirect to a success page or another view
 return redirect('index')
 #return redirect('project_detail', project_id=project.pk)

 else:
 form = ProjectForm()
 return render(request, 'registration/register_project.html', {'form': form})

```

**Step 7: Create an HTML template file register\_project.html in the templates/registration/ directory and render the form.**

```
<!-- create_project.html -->

{ % extends 'base.html' % }

{ { form.duration } }

{ % block content % }

<div class="container my-3">

<h1>Create Project</h1>

<form method="post">

{ % csrf_token %}

<div class="form-group">

<label for="id_topic">Topic:</label>

{ { form.topic } }

</div>

<div class="form-group">

<label for="id_languages_used">Languages Used:</label>

{ { form.languages_used } }

</div>

<div class="form-group">

<label for="id_duration">Duration:</label>

</div>
```

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

</form>

</div>

{ % endblock % }
```

```
<!-- create_project.html -->
{% extends 'base.html' %}

{% block content %}

Create Project

<form method="post">
 {% csrf_token %}
 <div class="form-group">
 <label for="id_topic">Topic:</label>
 {{ form.topic }}
 </div>
 <div class="form-group">
 <label for="id_languages_used">Languages Used:</label>
 {{ form.languages_used }}
 </div>
 <div class="form-group">
 <label for="id_duration">Duration:</label>
 {{ form.duration }}
 </div>
 <button type="submit" class="btn btn-primary">Submit</button>
</form>
</div>

{% endblock %}


```

#### Step 8: add the index.html to show the registered projects

```
<h1 class="mt-5 my-5">Projects</h1>

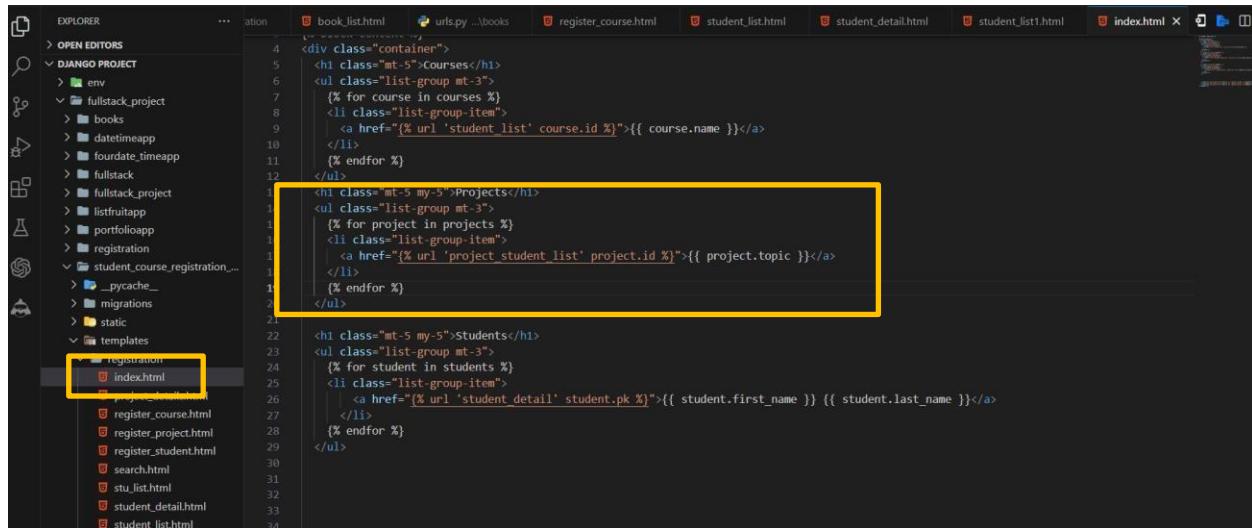
<ul class="list-group mt-3">

 {% for project in projects %}

 <li class="list-group-item">
 {{ project.topic }}

 {% endfor %}


```



```

<div class="container">
 <h1 class="mt-5">Courses</h1>
 <ul class="list-group mt-3">
 {% for course in courses %}
 <li class="list-group-item">
 {{ course.name }}

 {% endfor %}

 <h1 class="mt-5 my-5">Projects</h1>
 <ul class="list-group mt-3">
 {% for project in projects %}
 <li class="list-group-item">
 {{ project.topic }}

 {% endfor %}

 <h1 class="mt-5 my-5">Students</h1>
 <ul class="list-group mt-3">
 {% for student in students %}
 <li class="list-group-item">
 {{ student.first_name }} {{ student.last_name }}

 {% endfor %}


```

- Add the `register_student.html` to this is Student can Register the Created Project

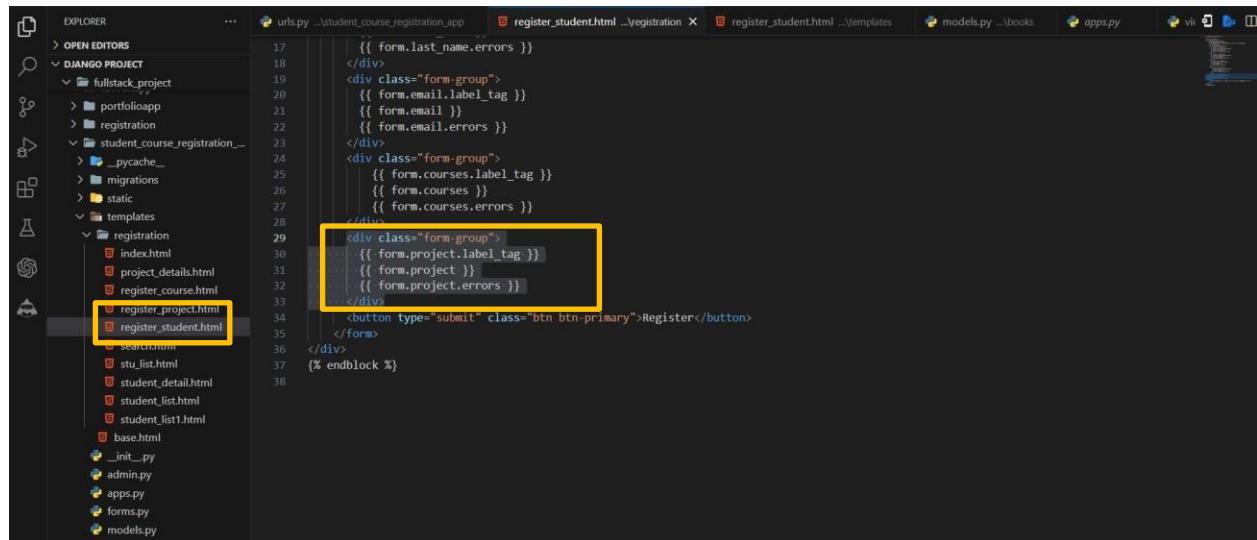
```
<div class="form-group">
```

```
 {{ form.project.label_tag }}
```

```
 {{ form.project }}
```

```
 {{ form.project.errors }}
```

```
</div>
```



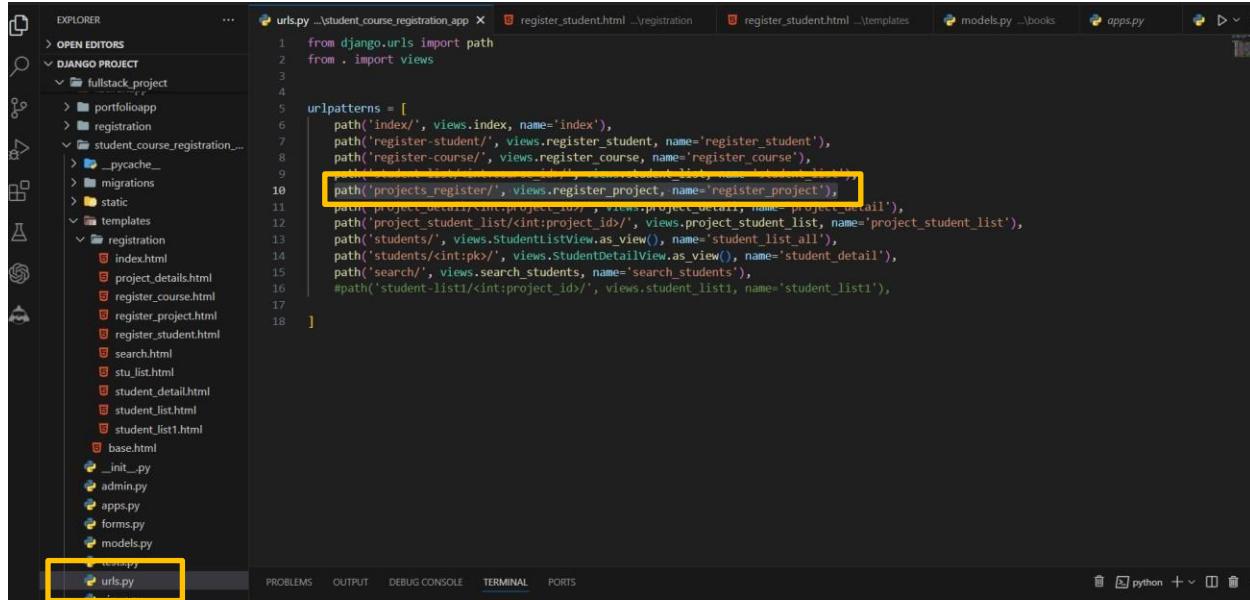
```

<div class="form-group">
 {{ form.last_name.errors }}
</div>
<div class="form-group">
 {{ form.email.label_tag }}
 {{ form.email }}
 {{ form.email.errors }}
</div>
<div class="form-group">
 {{ form.courses.label_tag }}
 {{ form.courses }}
 {{ form.courses.errors }}
</div>
<div class="form-group">
 {{ form.project.label_tag }}
 {{ form.project }}
 {{ form.project.errors }}
</div>
<button type="submit" class="btn btn-primary">Register</button>
</form>
<% endblock %>

```

## Step 9: Add a URL pattern for the register\_project in your urls.py file

```
path('projects_register/', views.register_project, name='register_project'),
```



```

EXPLORER OPEN EDITORS ...
Django Project fullstack_project
 portfolioapp
 registration
 student_course_registration...
 __pycache__
 migrations
 static
 templates
 registration
 index.html
 project_details.html
 register_course.html
 register_project.html
 register_student.html
 search.html
 stu_list.html
 student_detail.html
 student_list.html
 student_list1.html
 base.html
 __init__.py
 admin.py
 apps.py
 forms.py
 models.py
 urls.py
 wsgi.py
urls.py ... register_student.html ... registration register_student.html ... templates models.py ... books apps.py ...
1 from django.urls import path
2 from . import views
3
4
5 urlpatterns = [
6 path('index/', views.index, name='index'),
7 path('register-student/', views.register_student, name='register_student'),
8 path('register-course/', views.register_course, name='register_course'),
9 path('project-list//', views.project_list, name='project_list'),
10 path('projects_register/', views.register_project, name='register_project'),
11 path('project_detail//', views.project_detail, name='project_detail'),
12 path('project_student_list/int:project_id/', views.project_student_list, name='project_student_list'),
13 path('students/', views.studentListview.as_view(), name='student_list_all'),
14 path('students/int:pk>/', views.StudentDetailView.as_view(), name='student_detail'),
15 path('search/', views.search_students, name='search_students'),
16 #path('student-list1/int:project_id>/', views.student_list1, name='student_list1'),
17]
18]

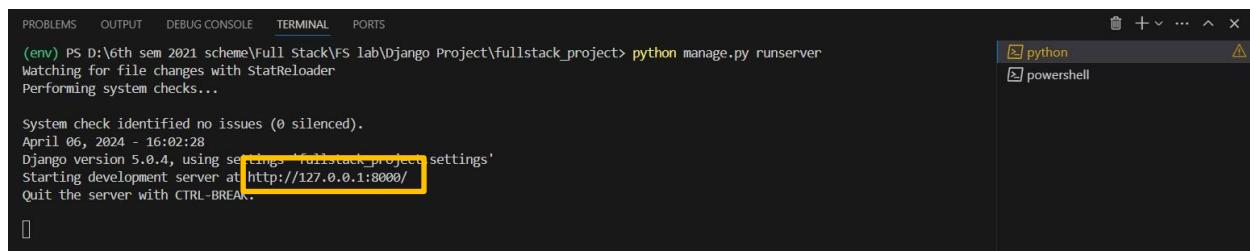
```

## Step-10: 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/>.



```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
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.

```

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

**Final Output By Clicking Register Project and Register Student**

---

**Projects**

Covide vaccine
child Vaccine System

**Students**

Hanumanthu hanu
asfd dsfad
Hanumanthu Hanu
hh df
hanumanthu hanu
hello how
gangu gangu
nagesh nagu

[Register Student](#) [Register Course](#) [Register Project](#)

**Fig: Home Page Screen**

### Create Project

Topic:

Languages Used:

Duration:

**Submit**

**Fig: Project Creation Screen**

Software Engineering
Software Engineering

### Projects

Covide vaccine
child Vaccine System
Login App

**Fig: Project Successfully Creation Screen**

## Student Registration

First name:

Last name:

Email:

Courses:

DBMS

CN

Software Engineering

Software Engineering

Project:

Covid19 vaccine

child Vaccine System

Login App

**Fig: Student Register the Project**

---

## Students Registered for

- Demo demo

**Fig: Student Registered Particular Projects**

## **Experiment-10**

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.

**Step-1: First, let's create models for Student and Course if you haven't already:**

### **Models.py**

```
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="")
 courses = models.ManyToManyField(Course, related_name='students',
 blank=True)
 project = models.ManyToManyField('Project', related_name='students',
 blank=True)
```

```
models.py -\student_course_registration_app x register_project.html admin.py forms.py -\student_course_registration_app forms.py -\registration D
OPEN EDITORS
D JANGO PROJECT
fullstack_project
 \fullstack_project
 > migrations
 > static
 > templates
 > registration
 index.html
 project_details.html
 register_course.html
 register_project.html
 register_student.html
 search.html
 stu_list.html
 student_detail.html
 student_list.html
 student_list1.html
 base.html
 __init__.py
 admin.py
 apps.py
 forms.py
models.py
reass.py
urls.py
views.py
create_book_data.py
db.sqlite3
```

```
1 from django.db import models
2
3
4 class Course(models.Model):
5 name = models.CharField(max_length=255)
6 description = models.TextField(blank=True, null=True)
7
8 def __str__(self):
9 return self.name
10
11
12
13 class Student(models.Model):
14 first_name = models.CharField(max_length=255, default='')
15 last_name = models.CharField(max_length=255, default='')
16 email = models.EmailField(unique=True, default='')
17 courses = models.ManyToManyField(Course, related_name='students', blank=True)
18 project = models.ManyToManyField('Project', related_name='students', blank=True)
19
20
21 def __str__(self):
22 return f'{self.first_name} {self.last_name}'
23
24
25 class Project(models.Model):
26 topic = models.CharField(max_length=100, default='') # Added default value for topic
27 languages_used = models.CharField(max_length=100, default='')
28 duration = models.IntegerField(default=0)
29
30 def __str__(self):
31 return self.topic
32
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
[14/Apr/2024 01:41:24] "GET /project_student_list/32/ HTTP/1.1" 200 1361
[14/Apr/2024 01:41:50] "GET /register-student/ HTTP/1.1" 200 2958
[14/Apr/2024 01:42:30] "POST /register-student/ HTTP/1.1" 302 0
[14/Apr/2024 01:42:30] "GET /index/ HTTP/1.1" 200 3347
[14/Apr/2024 01:43:03] "GET /project_student_list/34/ HTTP/1.1" 200 1323
```

**Step-2: Next, create views for the list and detail views and import necessary packages:**

```
from django.views.generic import ListView, DetailView
```

**class StudentListView(ListView):**

**model = Student**

```
template_name = 'registration/stu_list.html'
```

**context\_object\_name = 'students'**

```
class StudentDetailView(DetailView):
```

**model = Student**

```
template_name = 'registration/student_detail.html'
```

**context\_object\_name = 'student'**

The screenshot shows the PyCharm IDE interface with the code editor open to a Python file named `views.py`. The code defines several views for a Django project. A yellow box highlights the `student_list_new` method. The code uses Django's `render` function to return an HTML template with context variables. It also includes imports for `render`, `JsonResponse`, and `Student` from the `models` module.

```
def project_student_list(request, project_id):
 project = Project.objects.get(id=project_id)
 students = project.students.all()
 return render(request, 'registration/student_list.html', {'students': students, 'project': project})

def student_list_new(request):
 students = Student.objects.all()
 return render(request, 'registration/stu_list.html', {'students': students})

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

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

from django.shortcuts import render
from django.http import JsonResponse
from .models import Student

def search_students(request):
 if request.headers.get('X-Requested-With') == 'XMLHttpRequest':
 if request.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:
 results.append({'id': student.id, 'name': student.name})
 return JsonResponse(results)
```

- In the StudentListView, we've overridden the `get_queryset` method to filter the students based on the `course_id` parameter in the URL. If the `course_id` is

provided, it will return the students associated with that course; otherwise, it will return all students.

**Step-3: Now, let's create the templates:****stu\_list.html**

```
{% extends 'base.html' %}

{% block content %}

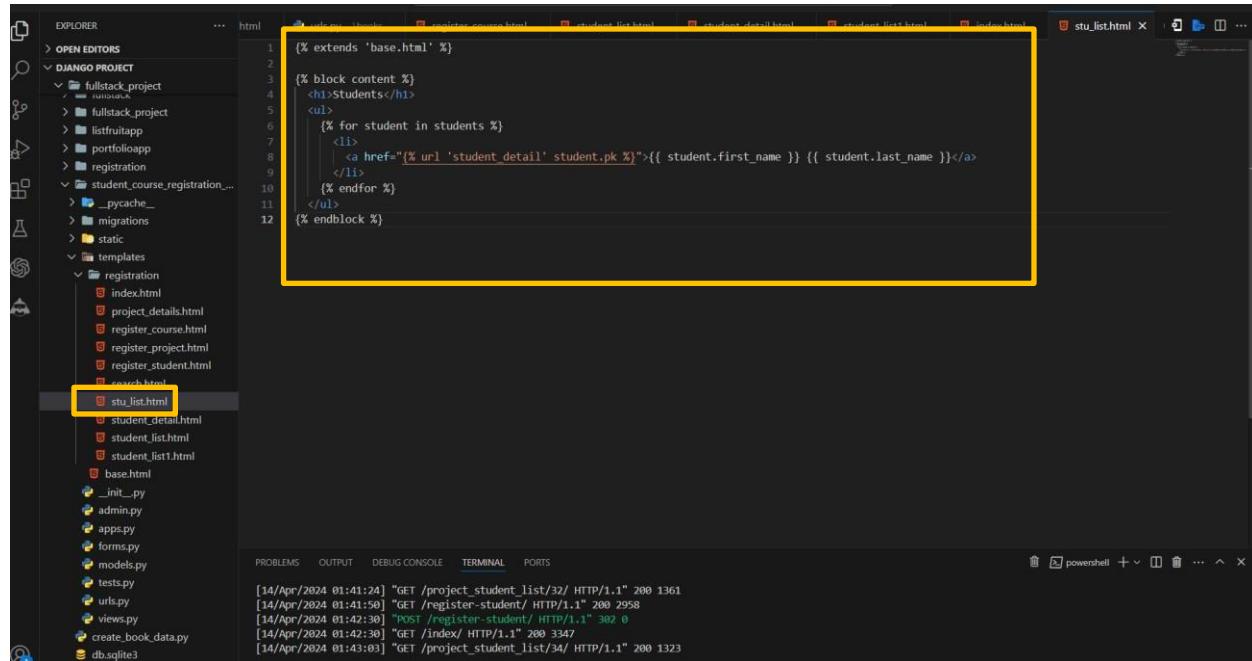
<h1>Students</h1>

 {% for student in students %}

 {{ student.first_name }} {{ student.last_name }}

 {% endfor %}

{% endblock %}
```



```

1 {% extends 'base.html' %}
2
3 {% block content %}
4 <h1>Students</h1>
5
6 {% for student in students %}
7
8 {{ student.first_name }} {{ student.last_name }}
9
10 {% endfor %}
11
12 {% endblock %}

```

## student\_detail.html

```

{ % extends 'base.html' % }

{ % block content % }

<div class="container my-5">

<div class="row">

<div class="col-md-6 offset-md-3">

<div class="card">

<div class="card-body">

<h1 class="card-title"> Name: {{ student.first_name }} {{ student.last_name }}</h1>

<h1 class="card-title">Email: {{ student.email }}</h1>

</div>

</div>

```

```
</div>
</div>
</div>
{% endblock %}
```

The screenshot shows a code editor interface with a dark theme. On the left is the Explorer sidebar showing the project structure:

- OPEN EDITORS
- DIANGO PROJECT
- fullstack\_project
- fullstackapp
- listfrutapp
- portfolioapp
- registration
- student\_course\_registration...
- \_pycache\_
- migrations
- static
- templates
- registration
- index.html
- project\_details.html
- register\_course.html
- register\_project.html
- register\_student.html
- search.html
- stu\_list.html
- student\_detail.html
- student\_list1.html
- base.html
- \_\_init\_\_.py
- admin.py
- apps.py
- forms.py
- models.py
- tests.py
- urls.py
- views.py
- create\_book\_data.py
- db.sqlite3

The main editor area displays a template file named `student_detail.html`:

```
1 {% extends "base.html" %}
2
3 {% block content %}
4 <div class="container my-5">
5 <div class="row">
6 <div class="col-md-6 offset-md-3">
7 <div class="card">
8 <div class="card-body">
9 <h1 class="card-title"> Name: {{ student.first_name }} {{ student.last_name }}</h1>
10 <h1 class="card-title">Email: {{ student.email }}</h1>
11 </div>
12 </div>
13 </div>
14 </div>
15 </div>
16 {% endblock %}
```

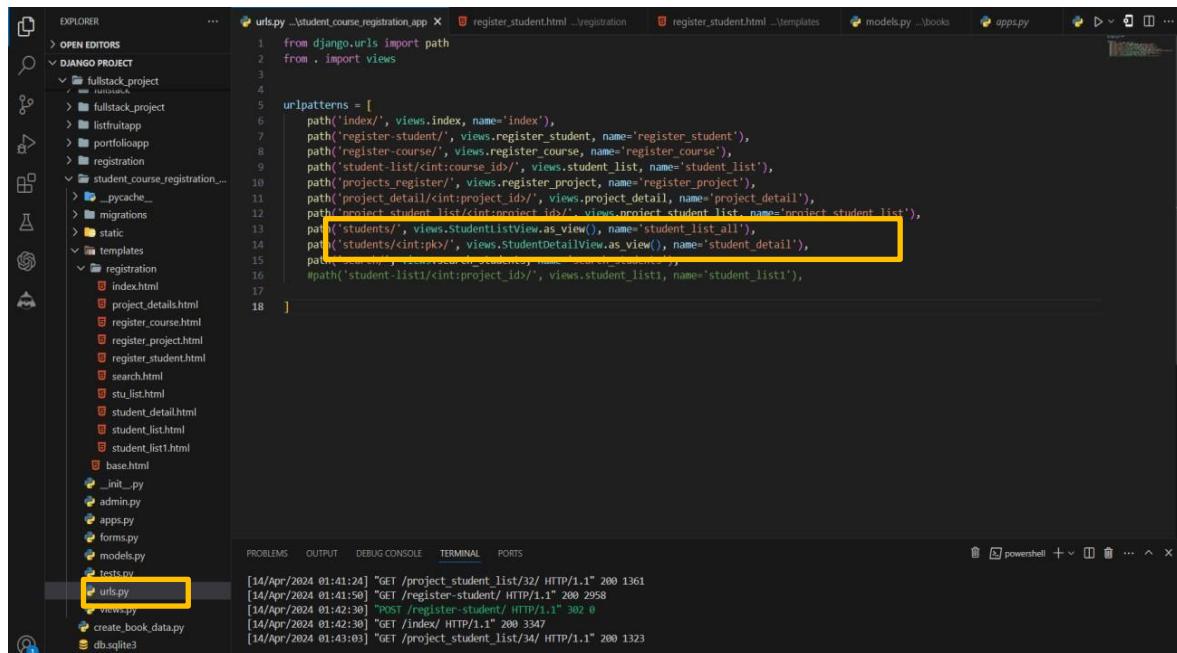
A yellow rectangular box highlights the entire content of the `student_detail.html` file. Below the editor, the terminal shows the following log output:

```
[14/Apr/2024 01:41:24] "GET /project_student_list/32/ HTTP/1.1" 200 1361
[14/Apr/2024 01:41:50] "GET /register-student/ HTTP/1.1" 200 2998
[14/Apr/2024 01:42:30] "POST /register-student/ HTTP/1.1" 302 0
[14/Apr/2024 01:42:30] "GET /index/ HTTP/1.1" 200 3347
[14/Apr/2024 01:43:03] "GET /project_student_list/34/ HTTP/1.1" 200 1323
```

**Step-4: add the URL patterns:**

```
path('students/', views.StudentListView.as_view(), name='student_list_all'),
```

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



```

EXPLORER
OPEN EDITORS
Django Project
fullstack_project
 fullstack_project
 listruiapp
 portfolioapp
 registration
 student_course_registration...
 urls.py
 migrations
 static
 templates
 registration
 index.html
 project_details.html
 register_course.html
 register_project.html
 register_student.html
 search.html
 stu_list.html
 student_detail.html
 student_list.html
 student_list1.html
 base.html
 __init__.py
 admin.py
 apps.py
 forms.py
 models.py
 tests.py
 urls.py
 views.py
 create_book_data.py
 db.sqlite3

urls.py -\student_course_registration_app x register_student.html ...registration register_student.html ...templates models.py ...books apps.py

from django.urls import path
from . import views

urlpatterns = [
 path('index/', views.index, name='index'),
 path('register-student/', views.register_student, name='register_student'),
 path('register-course/', views.register_course, name='register_course'),
 path('student-list/<int:course_id>', views.student_list, name='student_list'),
 path('projects-register/', views.register_project, name='register_project'),
 path('project_detail/<int:project_id>', views.project_detail, name='project_detail'),
 path('project-student-list/<int:project_id>', views.project_student_list, name='project_student_list'),
 path('students/', views.StudentListView.as_view(), name='student_list_all'),
 path('students/<int:pk>', views.StudentDetailView.as_view(), name='student_detail'),
 path('student-list1/<int:project_ids>', views.student_list1, name='student_list1'),
]
#path('student-list1/<int:project_ids>', views.student_list1, name='student_list1'),
]

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

[14/Apr/2024 01:41:24] "GET /project\_student\_list/32/ HTTP/1.1" 200 1361  
[14/Apr/2024 01:41:56] "GET /register-student/ HTTP/1.1" 200 2958  
[14/Apr/2024 01:42:38] "POST /register-student/ HTTP/1.1" 302 0  
[14/Apr/2024 01:42:38] "GET /index/ HTTP/1.1" 200 3347  
[14/Apr/2024 01:43:03] "GET /project\_student\_list/34/ HTTP/1.1" 200 1323

**Step-5: add Index.html For showing the Students In Main Page: Index.html**

```

<h1 class="mt-5 my-5">Students</h1>

<ul class="list-group mt-3">

 { % for student in students % }

 <li class="list-group-item">

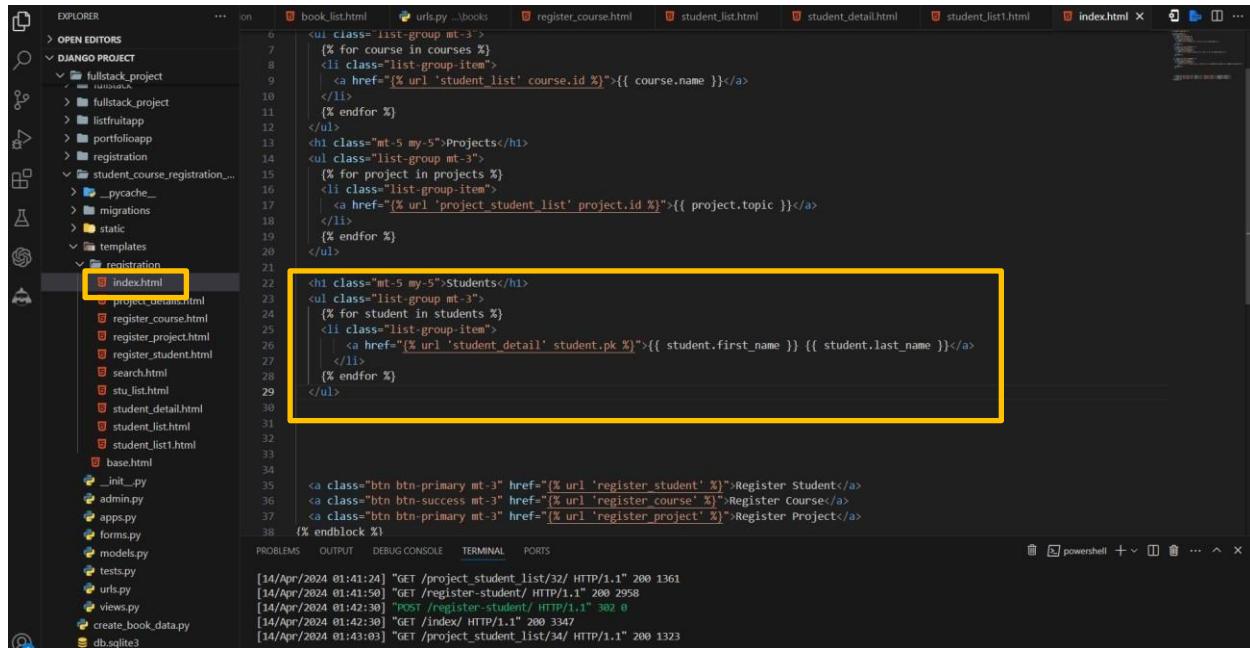
 {{ student.first_name }} {{

 student.last_name }}

 { % endfor % }

```

</ul>



```

<ul class="list-group mt-3">
 {% for course in courses %}
 <li class="list-group-item">
 {{ course.name }}

 {% endfor %}

<h1 class="mt-5 my-5">Projects</h1>
<ul class="list-group mt-3">
 {% for project in projects %}
 <li class="list-group-item">
 {{ project.topic }}

 {% endfor %}

<h1 class="mt-5 my-5">Students</h1>
<ul class="list-group mt-3">
 {% for student in students %}
 <li class="list-group-item">
 {{ student.first_name }} {{ student.last_name }}

 {% endfor %}

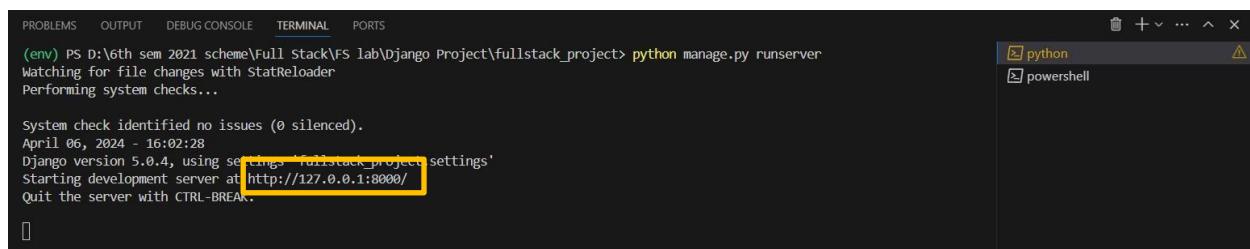

```

## Step-6: 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/>.

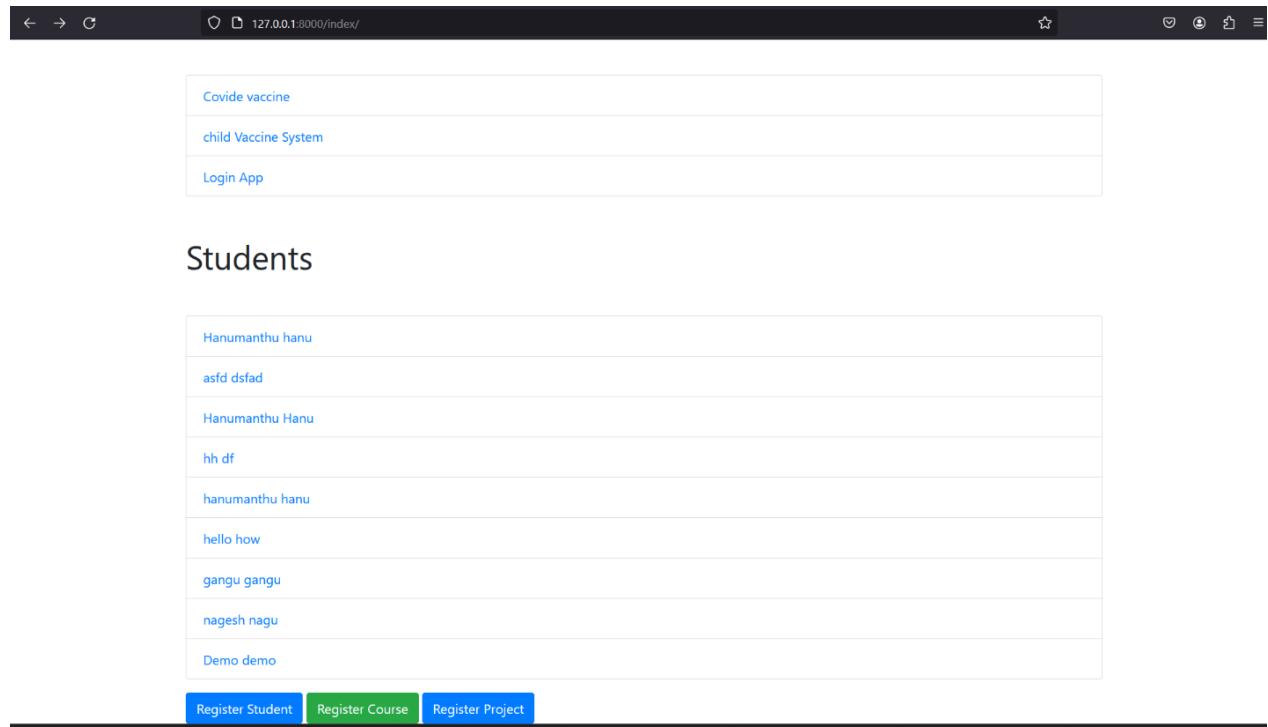
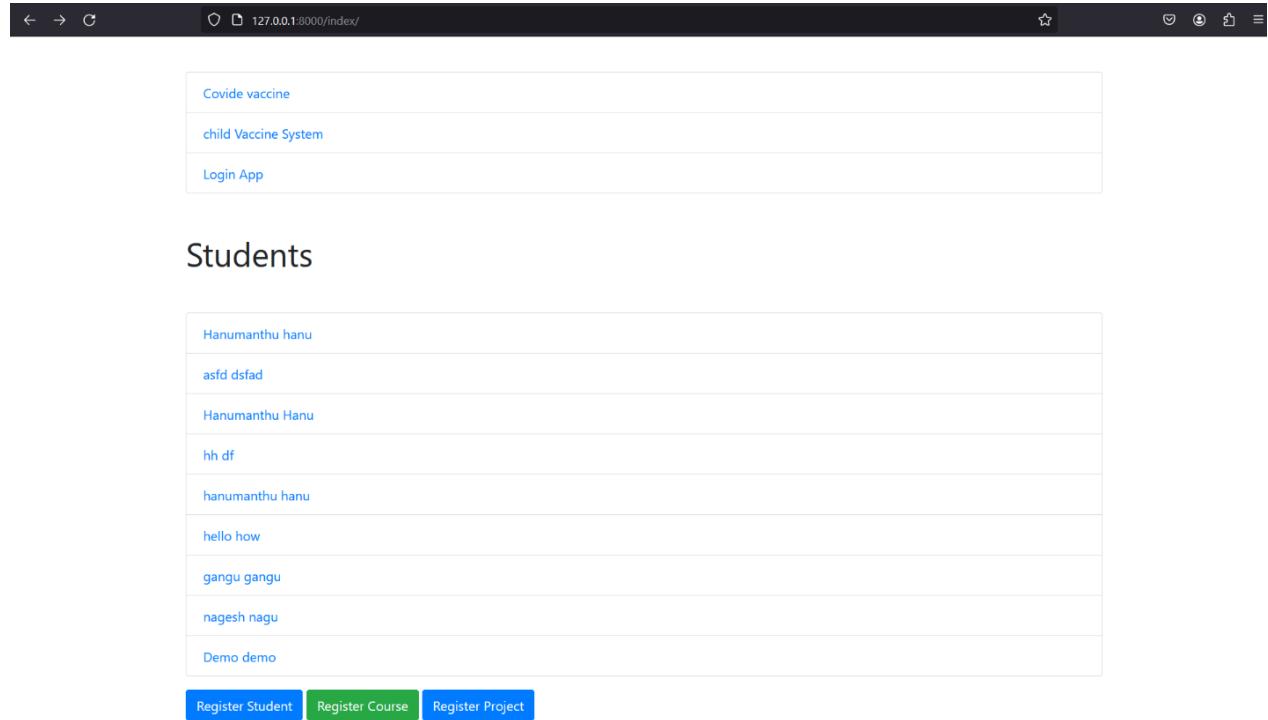


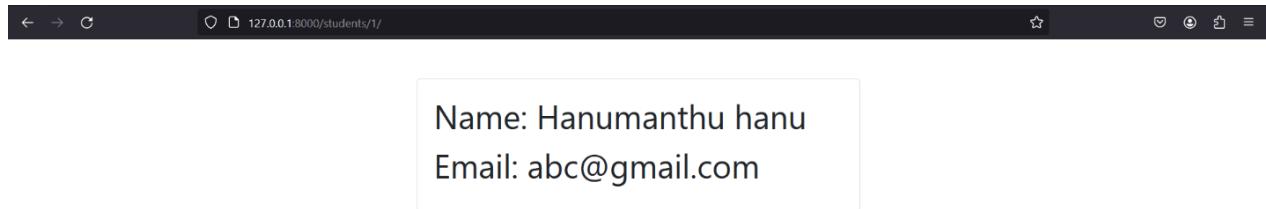
```

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.

```

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

**Final Output By Clicking Register Project and Register Student****Fig: Main Screen****Fig: List View of Students Screen**



**Fig: Detailed View of Student Screen**

## **Experiment-11**

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

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

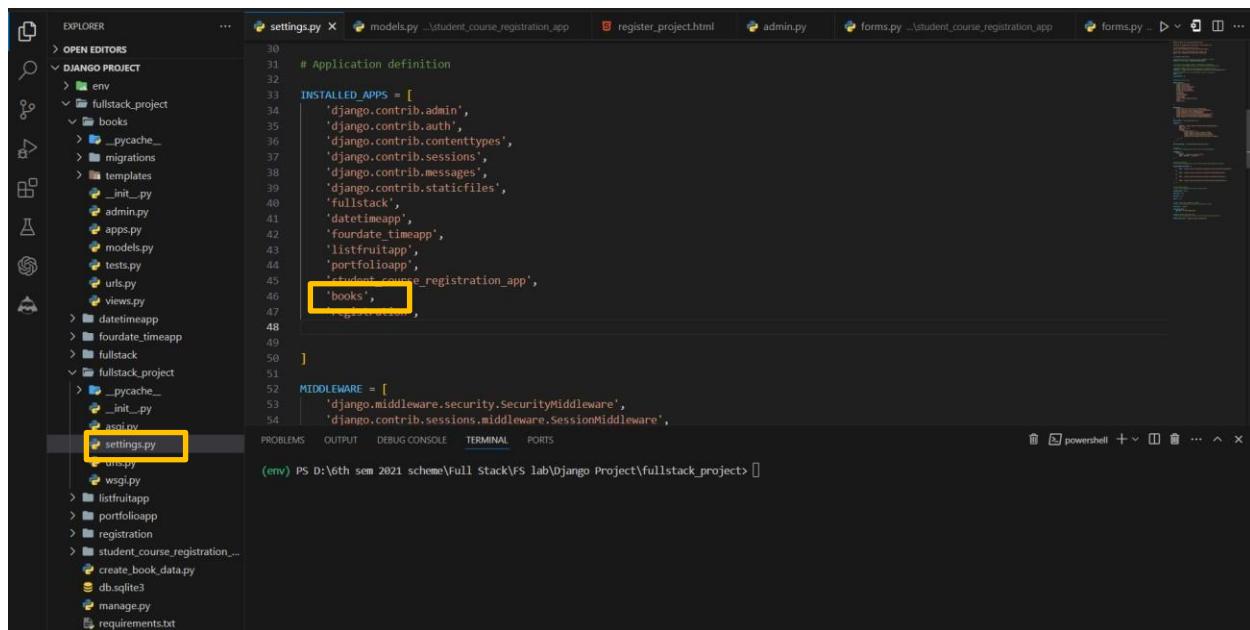


```
(env) PS D:\6th sem 2021 scheme\Full Stack FS lab\Django Project\fullstack_project> django-admin startapp books
```

**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:



```
settings.py
...
INSTALLED_APPS = [
 'django.contrib.admin',
 'django.contrib.auth',
 'django.contrib.contenttypes',
 'django.contrib.sessions',
 'django.contrib.messages',
 'django.contrib.staticfiles',
 'fullstack',
 'datetimeapp',
 'datetime_timeapp',
 'foundate_timeapp',
 'fullstack',
 'listfruitapp',
 'portfolioapp',
 'student_course_registration_app',
 'books',
]
MIDDLEWARE = [
 'django.middleware.security.SecurityMiddleware',
 'django.contrib.sessions.middleware.SessionMiddleware',
 ...
]
```

## Step-03: Create models

- Open the **models.py** file inside the student\_course\_registration\_app and define your models:

```
from django.db import models
```

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

```
 title = models.CharField(max_length=200)
```

```
 author = models.CharField(max_length=200)
```

```
 publication_date = models.DateField()
```

```
 def __str__(self):
```

```
books/models.py
from django.db import models

class Book(models.Model):
 title = models.CharField(max_length=200)
 author = models.CharField(max_length=200)
 publication_date = models.DateField()

 def __str__(self):
 return self.title
```

**Step-04:** Create a **views.py** file in your books app and define the views for generating CSV and PDF files

```
import csv

from django.http import HttpResponse

from django.template.loader import get_template

from xhtml2pdf import pisa

from .models import Book

def export_books_csv(request):

 response = HttpResponse(content_type='text/csv')

 response['Content-Disposition'] = 'attachment; filename="books.csv"'

 writer = csv.writer(response)

 writer.writerow(['Title', 'Author', 'Publication Date'])

 books = Book.objects.all().values_list('title', 'author', 'publication_date')

 for book in books:

 writer.writerow(book)

 return response

def export_books_pdf(request):

 books = Book.objects.all()
 template_path = 'book_list.html'

 context = {'books': books}
```

```

response = HttpResponse(content_type='application/pdf')

response['Content-Disposition'] = 'attachment; filename="books.pdf"'"

template = get_template(template_path)

html = template.render(context)

pisa_status = pisa.CreatePDF(
 html, dest=response)

if pisa_status.err:

 return HttpResponse('We had some errors <pre>' + html + '</pre>')

return response

```

```

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

 writer = csv.writer(response)
 writer.writerow(['Title', 'Author', 'Publication Date'])

 books = Book.objects.all().values_list('title', 'author', 'publication_date')
 for book in books:
 writer.writerow(book)

 return response

def export_books_pdf(request):
 books = Book.objects.all()
 template_path = 'book_list.html'
 context = {'books': books}
 response = HttpResponse(content_type='application/pdf')
 response['Content-Disposition'] = 'attachment; filename="books.pdf"'

 template = get_template(template_path)
 html = template.render(context)

 pisa_status = pisa.CreatePDF(
 html, dest=response)
 if pisa_status.err:
 return HttpResponse('We had some errors <pre>' + html + '</pre>')
 return response

```

Net: Found: /favicon.ico/  
[16/Apr/2024 11:12:16,272] - Broken pipe: from ('127.0.0.1', 56147)  
[16/Apr/2024 11:13:14] "GET /register-student/ HTTP/1.1" 200 2958  
[16/Apr/2024 11:13:24] "GET /students/1/ HTTP/1.1" 200 1581  
[16/Apr/2024 11:15:05] "GET /students/1/ HTTP/1.1" 200 1581

**Step-04:** In your books app, create a templates directory and an html file for rendering the PDF:

**books/**

**templates/**

**books/**

**book\_list.html**

**book\_list.html**

<!DOCTYPE html>

<html>

<head>

<title>Book List</title>

<style>

body {

font-family: Arial, sans-serif;

}

table {

border-collapse: collapse;

width: 100%;

}

th, td {

padding: 8px;

```
text-align: left;

border-bottom: 1px solid #ddd;

}

</style>

</head>

<body>

<h1>Book List</h1>

<table>

<thead>

<tr>

<th>Title</th>

<th>Author</th>

<th>Publication Date</th>

</tr>

</thead>

<tbody>

{% for book in books %}

<tr>

<td>{{ book.title }}</td>

<td>{{ book.author }}</td>

<td>{{ book.publication_date }}</td>
```

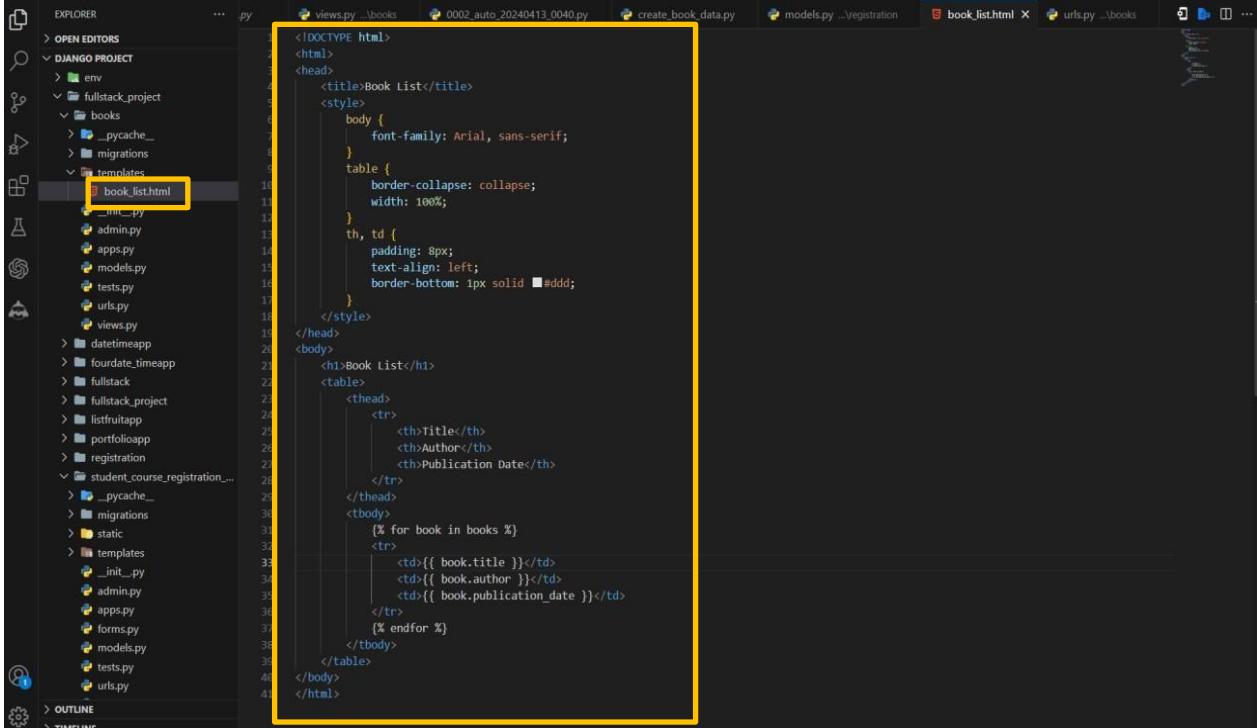
```
</tr>

{ % endfor % }

</table>

</body>

</html>
```



The screenshot shows a code editor interface with a dark theme. On the left is the Explorer sidebar showing a project structure for a Django application named 'fullstack\_project'. The 'templates' folder contains a file named 'book\_list.html', which is highlighted with a yellow box. The main editor area displays the contents of 'book\_list.html'.

```
<!DOCTYPE html>
<html>
<head>
 <title>Book List</title>
 <style>
 body {
 font-family: Arial, sans-serif;
 }
 table {
 border-collapse: collapse;
 width: 100%;
 }
 th, td {
 padding: 8px;
 text-align: left;
 border-bottom: 1px solid black;
 }
 </style>
</head>
<body>
 <h1>Book List</h1>
 <table>
 <thead>
 <tr>
 <th>Title</th>
 <th>Author</th>
 <th>Publication Date</th>
 </tr>
 </thead>
 <tbody>
 {% for book in books %}
 <tr>
 <td>{{ book.title }}</td>
 <td>{{ book.author }}</td>
 <td>{{ book.publication_date }}</td>
 </tr>
 {% endfor %}
 </tbody>
 </table>
</body>
</html>
```

&lt;/tbody&gt;

## Step-05: add the URL patterns and import Necessary Packages:

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

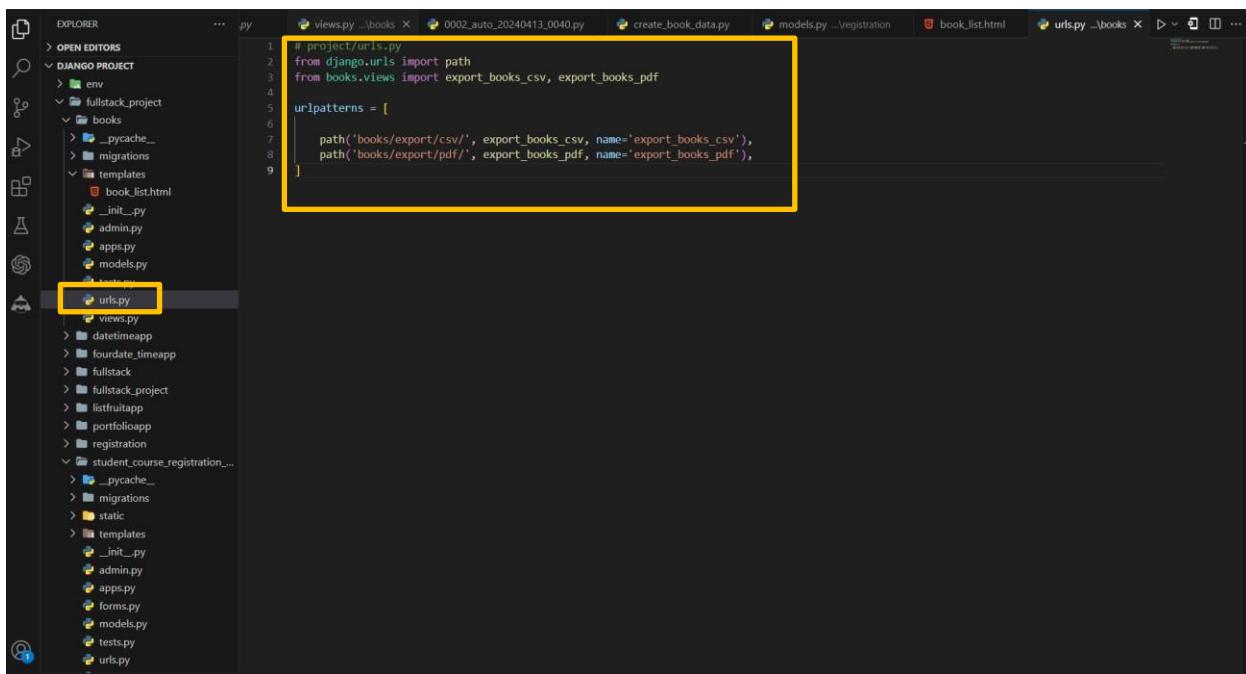
```
from books.views import export_books_csv, export_books_pdf
```

```
urlpatterns = [
```

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

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

```
]
```

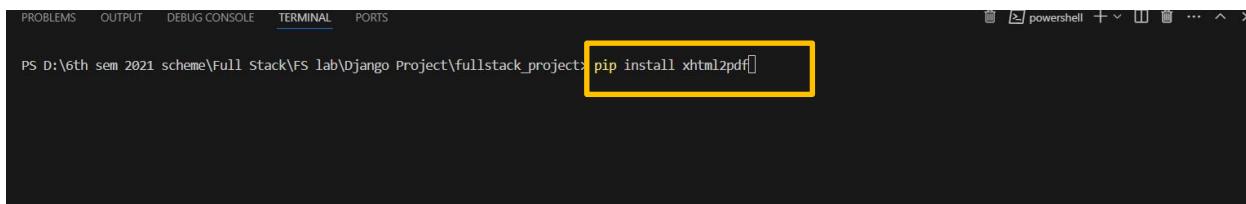


```
project/urls.py
from django.urls import path
from books.views import export_books_csv, export_books_pdf

urlpatterns = [
 path('books/export/csv/', export_books_csv, name='export_books_csv'),
 path('books/export/pdf/', export_books_pdf, name='export_books_pdf'),
]
```

## Step-06: Install the xhtml2pdf library for generating PDF files:

```
pip install xhtml2pdf
```



```
PS D:\6th sem 2021 scheme\Full Stack\FS lab\ Django Project\fullstack_project> pip install xhtml2pdf
```

- If you want to include the book data as part of your project's initial data setup, you can
- Step-07: Create the book titles and author and publication date**  
create a data migration. Here's how you can do it

### Django Shell

Run the following command to create a new data migration

- If you just want to create some sample data for testing or development purposes, you can
- python manage.py makemigrations books --empty**  
run the code in the Django shell as shown in the previous example. This is a quick and

The screenshot shows a dark-themed interface of VS Code. On the left is the Explorer sidebar with a tree view of a 'fullstack\_project' directory containing 'books', 'migrations', and other Django-related files like 'admin.py', 'models.py', and 'views.py'. A file named '0002\_auto\_20240413\_0040.py' is selected and highlighted with a yellow box. The main editor area contains Python code for a migration:

```

1 from django.db import migrations
2
3 def create_book_data(apps, schema_editor):
4 Book = apps.get_model('books', 'Book')
5 Book.objects.bulk_create([
6 Book(title='To Kill a Mockingbird', author='Harper Lee', publication_date='1960-07-11'),
7 Book(title='1984', author='George Orwell', publication_date='1949-06-08'),
8 Book(title='Pride and Prejudice', author='Jane Austen', publication_date='1813-01-28'),
9 Book(title='The Great Gatsby', author='F. Scott Fitzgerald', publication_date='1925-04-10'),
10 Book(title='The Catcher in the Rye', author='J.D. Salinger', publication_date='1951-07-16'),
11])
12
13 class Migration(migrations.Migration):
14
15 dependencies = [
16 ('books', '0001_initial'),
17]
18
19 operations = [
20 migrations.RunPython(create_book_data),
21]

```

Below the editor is a terminal window showing a PowerShell prompt with the command: `PS D:\6th sem 2021 scheme\Full Stack\5 lab\django Project\fullstack_project python manage.py makemigrations books --empty`. The terminal output is currently empty.

- This will create a new empty migration file in the book's/migrations/

- Open the newly created migration file (e.g., books/migrations/0002\_auto\_<...>.py) and add the code to create the book objects in the operations list of the Migration class.

```
from django.db import migrations

def create_book_data(apps, schema_editor):

 Book = apps.get_model('books', 'Book')

python manage.py migrate
```

- This will create the book objects in your database.

```
Book.objects.bulk_create([
 Book(title='To Kill a Mockingbird', author='Harper Lee', publication_date='1960-07-11'),
 Book(title='1984', author='George Orwell', publication_date='1949-06-08'),
 Book(title='Pride and Prejudice', author='Jane Austen', publication_date='1813-01-28'),
 Book(title='The Great Gatsby', author='F. Scott Fitzgerald', publication_date='1925-04-10'),
 Book(title='The Catcher in the Rye', author='J.D. Salinger', publication_date='1951-07-16'),
])
```

```
class Migration(migrations.Migration):

 dependencies = [
 ('books', '0001_initial'),
]

 operations = [
 migrations.RunPython(create_book_data),
]
```

**Run the following command to apply the data migration**

**Step-08: Custom Script**

- If you need to create the book data programmatically (e.g., during deployment or as part of a data import process), you can create a custom Python script and run it as needed.
- Create a new Python file ([e.g., create\\_book\\_data.py](#)) in your project's root directory, and add the code to create the book objects:

```
import os

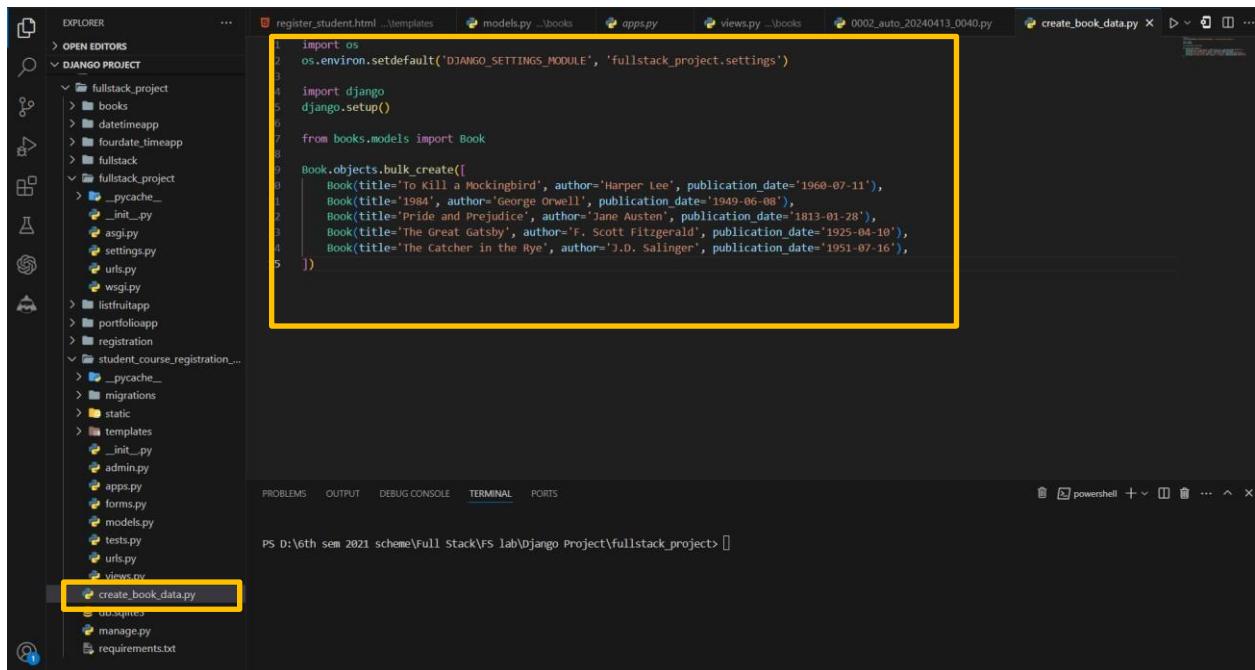
os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'fullstack_project.settings')

import django

django.setup()

from books.models import Book

Book.objects.bulk_create([
 Book(title='To Kill a Mockingbird', author='Harper Lee', publication_date='1960-07-11'),
 Book(title='1984', author='George Orwell', publication_date='1949-06-08'),
 Book(title='Pride and Prejudice', author='Jane Austen', publication_date='1813-01-28'),
 Book(title='The Great Gatsby', author='F. Scott Fitzgerald', publication_date='1925-04-10'),
 Book(title='The Catcher in the Rye', author='J.D. Salinger', publication_date='1951-07-16'),
])
```



A screenshot of the Visual Studio Code interface. On the left is the Explorer sidebar showing a project structure under 'DJANGO PROJECT'. In the center is the code editor with a Python script named 'create\_book\_data.py'. The script contains code to bulk create books from a list of titles and authors. A yellow box highlights the entire code block. At the bottom of the code editor, the status bar shows the path 'D:\6th sem 2021 scheme\Full Stack\FS lab\ Django Project\fullstack\_project> []'. Below the code editor are tabs for 'PROBLEMS', 'OUTPUT', 'DEBUG CONSOLE', 'TERMINAL', and 'PORTS'. The terminal tab is selected, showing the command 'PS D:\6th sem 2021 scheme\Full Stack\FS lab\ Django Project\fullstack\_project> []'.

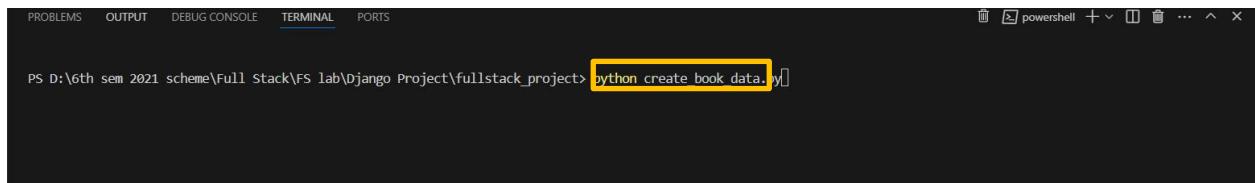
```

1 import os
2 os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'fullstack_project.settings')
3
4 import django
5 django.setup()
6
7 from books.models import Book
8
9 Book.objects.bulk_create([
10 Book(title='To Kill a Mockingbird', author='Harper Lee', publication_date='1960-07-11'),
11 Book(title='1984', author='George Orwell', publication_date='1949-06-08'),
12 Book(title='Pride and Prejudice', author='Jane Austen', publication_date='1813-01-28'),
13 Book(title='The Great Gatsby', author='F. Scott Fitzgerald', publication_date='1925-04-10'),
14 Book(title='The Catcher in the Rye', author='J.D. Salinger', publication_date='1951-07-16'),
15])

```

Run the script using the following command:

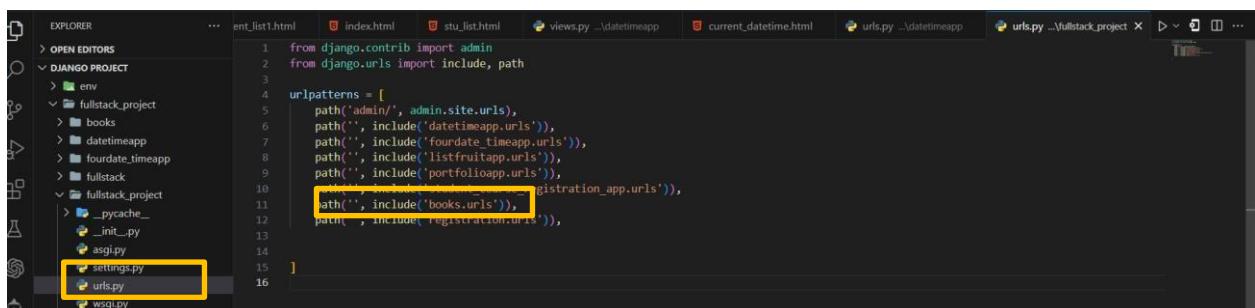
`python create_book_data.py`



A screenshot of the terminal window in VS Code. The command 'python create\_book\_data.py' is typed into the terminal. The status bar at the bottom shows the path 'D:\6th sem 2021 scheme\Full Stack\FS lab\ Django Project\fullstack\_project> [ ]'.

### Step-09: Update project URLs

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



A screenshot of the VS Code interface. The Explorer sidebar shows a project structure with a 'fullstack\_project' folder containing 'books', 'datetimeapp', 'foundate\_timeapp', and 'fullstack' subfolders. The 'fullstack' folder contains files like '\_pycache\_'. In the code editor, a file named 'urls.py' is open under 'fullstack'. The code defines a URL pattern that includes the 'books.urls' module. A yellow box highlights the line 'path('', include('books.urls')),'. The status bar at the bottom shows the path 'D:\6th sem 2021 scheme\Full Stack\FS lab\ Django Project\fullstack\_project> [ ]'.

```

1 from django.contrib import admin
2 from django.urls import include, path
3
4 urlpatterns = [
5 path('admin/', admin.site.urls),
6 path('', include('datetimeapp.urls')),
7 path('', include('foundate_timeapp.urls')),
8 path('', include('listfruitapp.urls')),
9 path('', include('portfolioapp.urls')),
10 path('', include('student_course_registration_app.urls')),
11 path('', include('books.urls')),
12 path('', include('registration.urls')),
13]

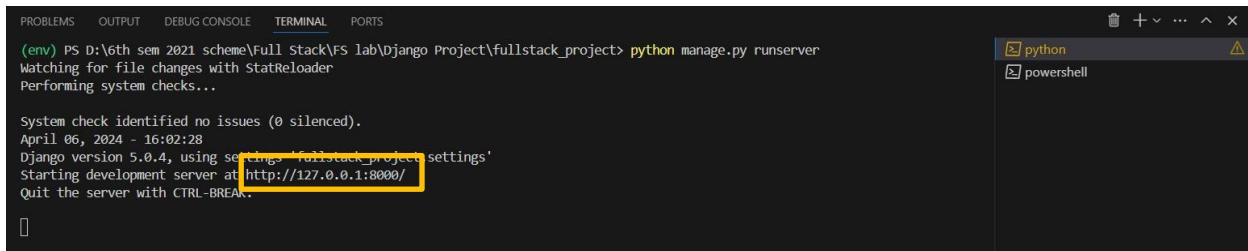
```

## 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/>.



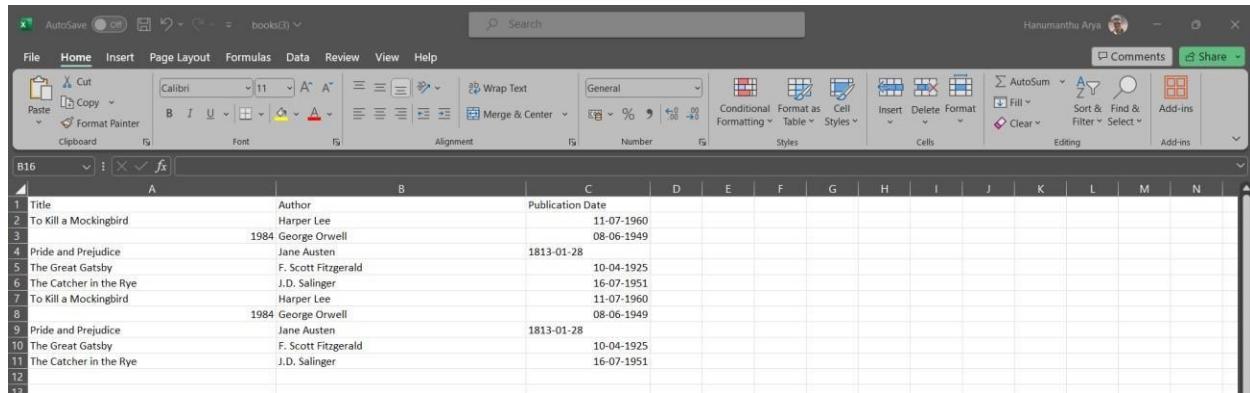
```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
(env) PS D:\6th sem 2021 scheme\Full StackFS lab\ Django Project\fullstack_project> python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...
System check identified no issues (0 silenced).
April 06, 2024 - 16:02:28
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.
```

- Type or copy this <http://127.0.0.1:8000/books/export/csv/>

## Final Output By Clicking Register Project and Register Student



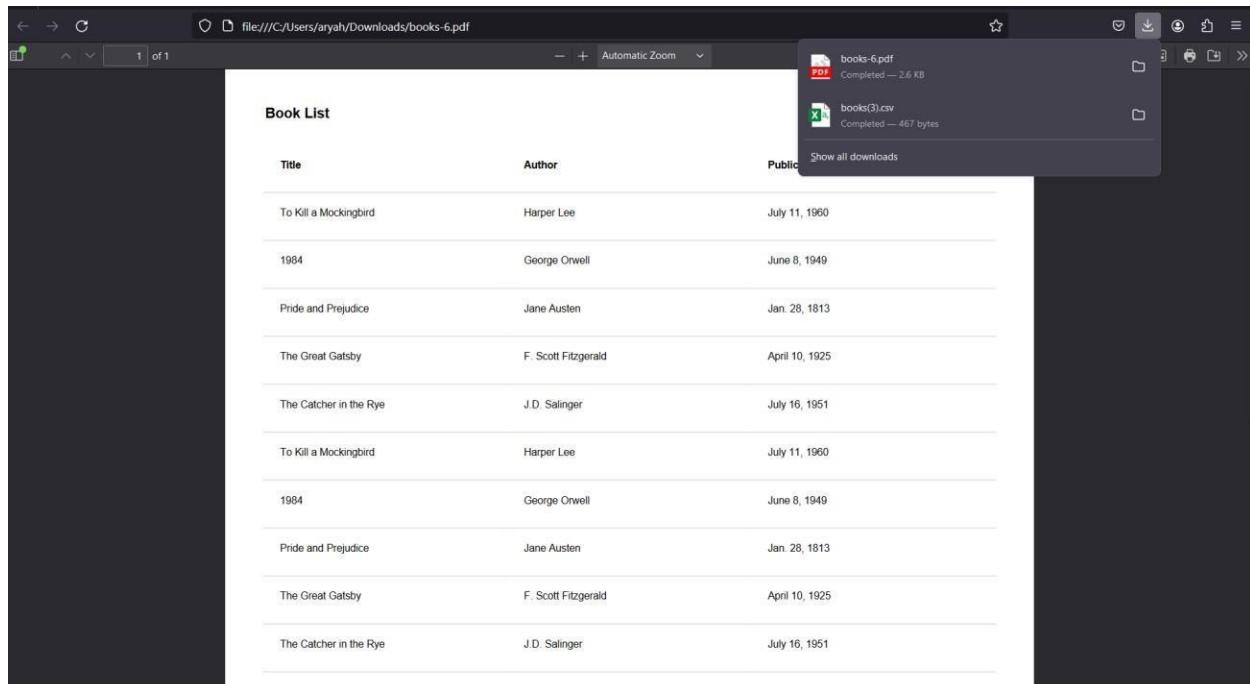
**Fig: CSV File Download Screen**



A screenshot of Microsoft Excel showing a table of book data. The table has columns for Title, Author, Publication Date, and other details. The data includes books like "To Kill a Mockingbird" by Harper Lee, "Pride and Prejudice" by Jane Austen, and "The Great Gatsby" by F. Scott Fitzgerald.

	Title	Author	Publication Date
1	To Kill a Mockingbird	Harper Lee	11-07-1960
2			08-06-1949
3	1984	George Orwell	
4	Pride and Prejudice	Jane Austen	1813-01-28
5	The Great Gatsby	F. Scott Fitzgerald	10-04-1925
6	The Catcher in the Rye	J.D. Salinger	16-07-1951
7	To Kill a Mockingbird	Harper Lee	11-07-1960
8			08-06-1949
9	1984	George Orwell	1813-01-28
10	Pride and Prejudice	Jane Austen	10-04-1925
11	The Great Gatsby	F. Scott Fitzgerald	16-07-1951
12	The Catcher in the Rye	J.D. Salinger	

- Type or copy this <http://127.0.0.1:8000/books/export/pdf/>



A screenshot of a browser window showing a PDF file download screen. The page title is "Book List". A download dialog box is open, showing two files: "books-6.pdf" (Completed — 2.6 KB) and "books(3).csv" (Completed — 467 bytes). The "Show all downloads" button is visible.

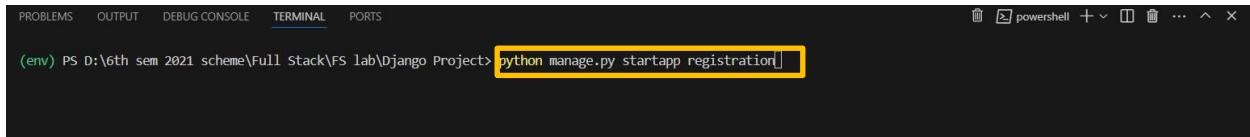
Title	Author	Publication Date
To Kill a Mockingbird	Harper Lee	July 11, 1960
1984	George Orwell	June 8, 1949
Pride and Prejudice	Jane Austen	Jan. 28, 1813
The Great Gatsby	F. Scott Fitzgerald	April 10, 1925
The Catcher in the Rye	J.D. Salinger	July 16, 1951
To Kill a Mockingbird	Harper Lee	July 11, 1960
1984	George Orwell	June 8, 1949
Pride and Prejudice	Jane Austen	Jan. 28, 1813
The Great Gatsby	F. Scott Fitzgerald	April 10, 1925
The Catcher in the Rye	J.D. Salinger	July 16, 1951

**Fig: PDF File Download Screen**

## **Experiment-12**

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.

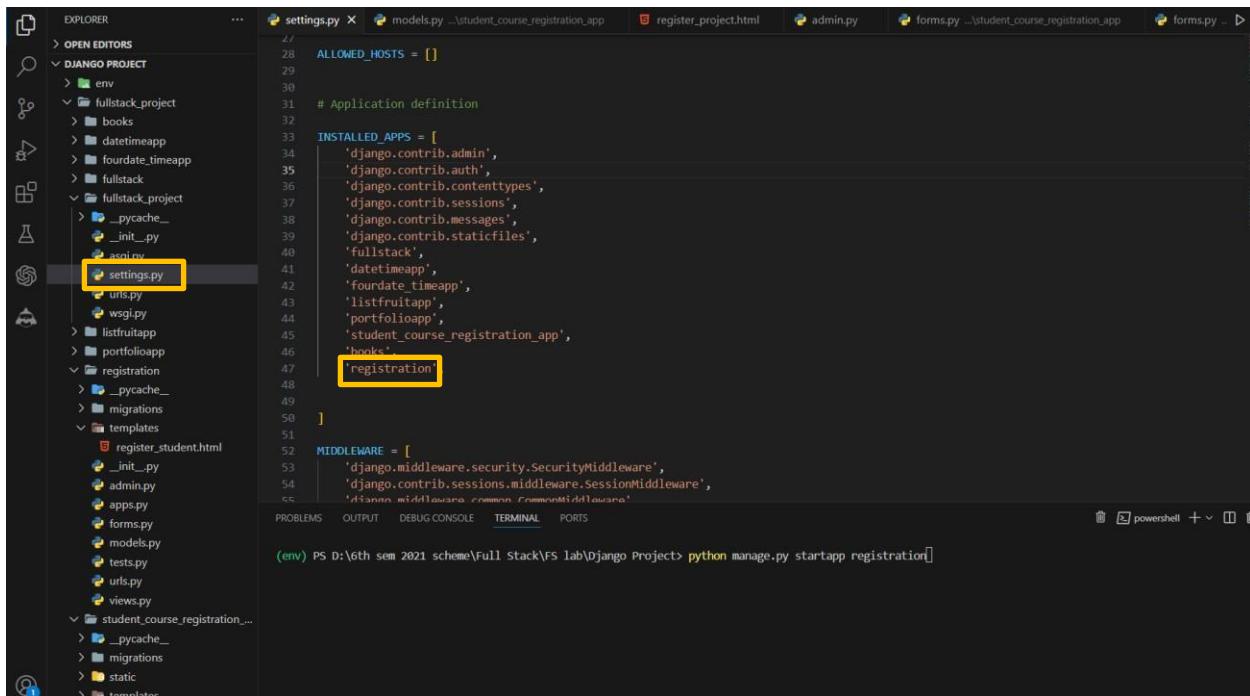


```
(env) PS D:\6th sem 2021 scheme\Full Stack\FS lab\ Django Project> python manage.py startapp registration
```

**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:



```
settings.py
...
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 'datetimeapp',
41 'foudate_timeapp',
42 'listfruitapp',
43 'portfolioapp',
44 'student_course_registration_app',
45 'books'
46 'registration'
47]
48
49 MIDDLEWARE = [
50 'django.middleware.security.SecurityMiddleware',
51 'django.contrib.sessions.middleware.SessionMiddleware',
52 'django.middleware.common.CommonMiddleware'
```

## Step-03: Create models

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

```
from django.db import models
```

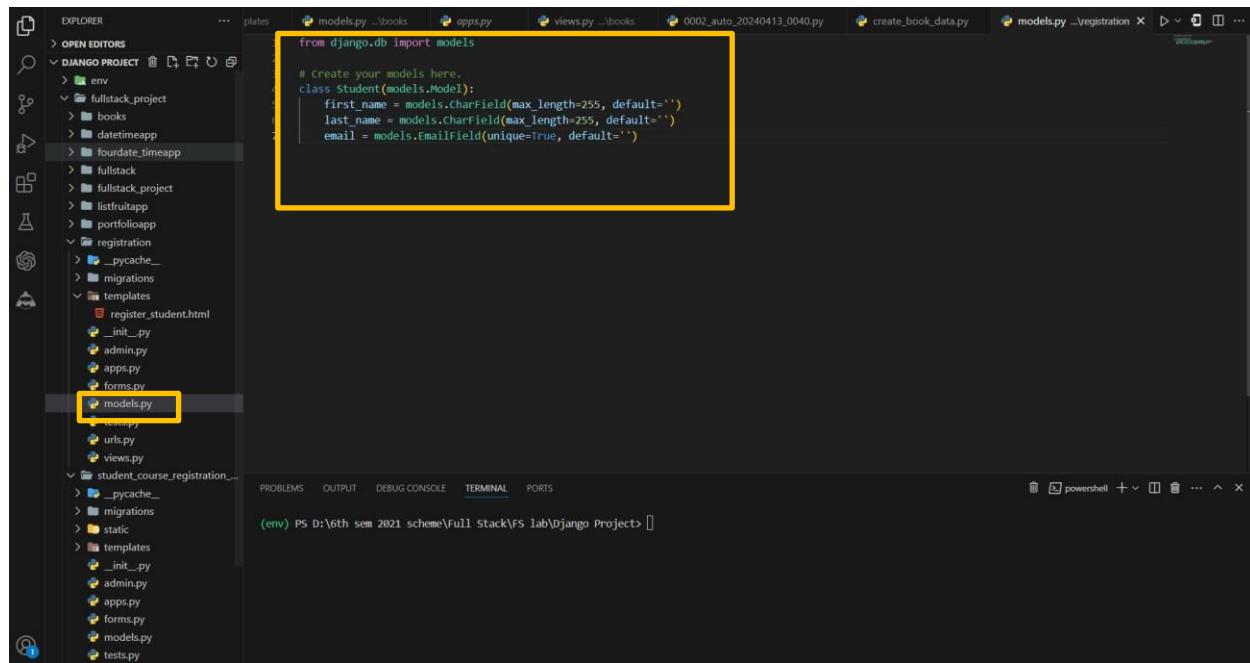
```
Create your models here.
```

```
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='')
```



```
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})
```

```
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)**

**rm): class Meta:**

**model = Student**

**fields** = ['first\_name',

```
'last_name', 'email']) widgets = {
```

```
'first_name': forms.TextInput(attrs={'class': 'form-
```

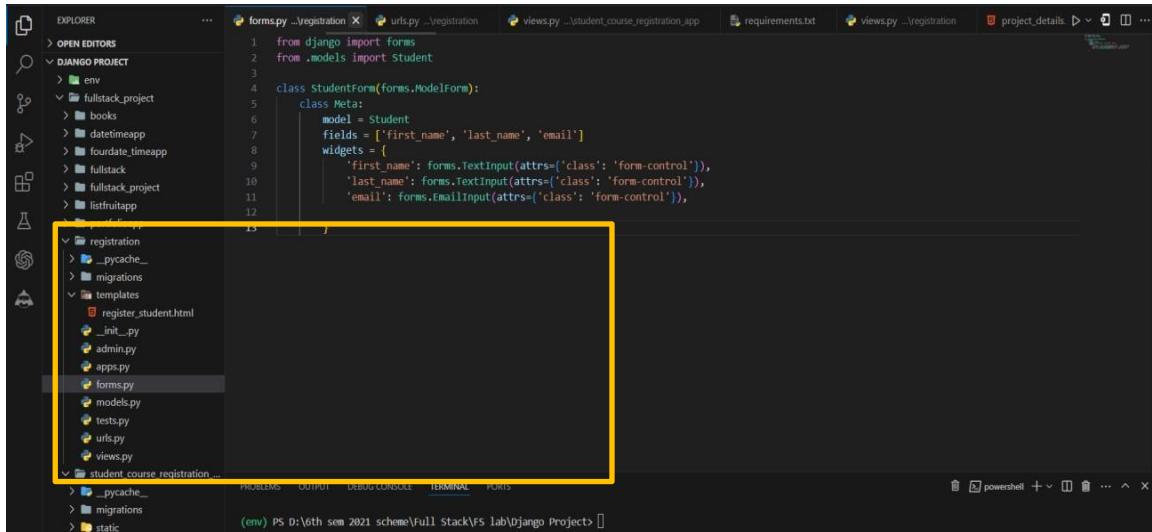
**'control'{}),** **'last\_name':**

```
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>

```

```
{% 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

 ${response.message}</div>');

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

 try again.</div>');

 }

 },

 });

 });

});
```

```

 error: function(xhr, errmsg, err) {

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

 }

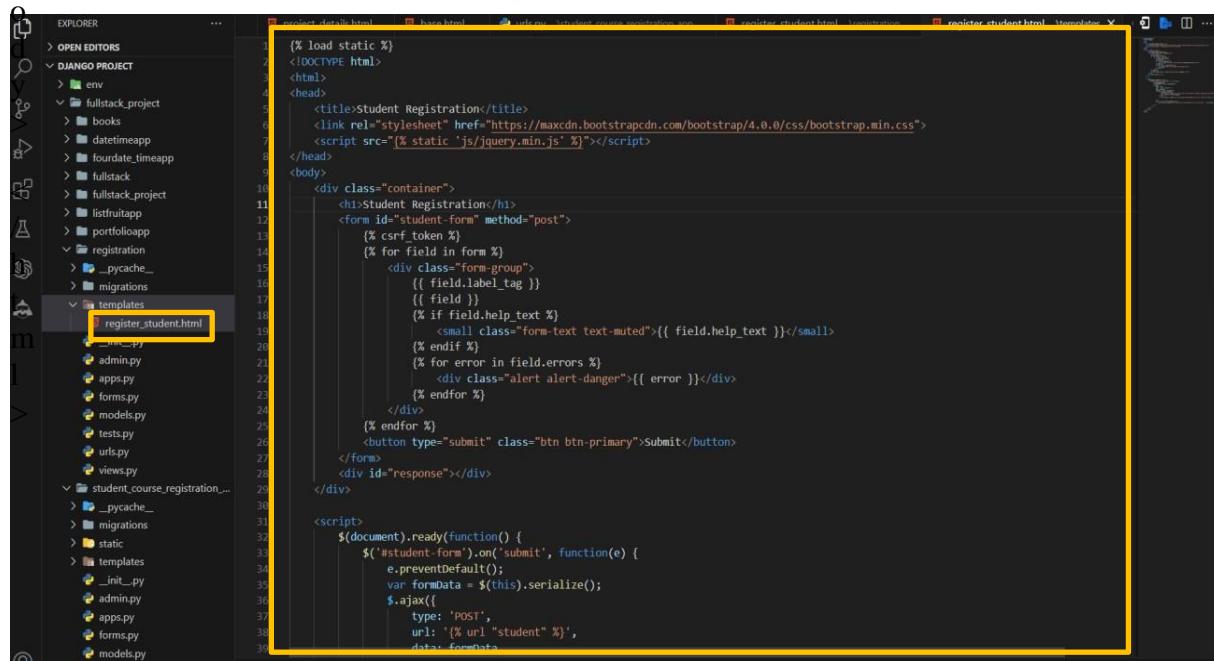
 });

}

);
;

</script>
<
/
b

```



```

1 {% load static %}
2 <!DOCTYPE html>
3 <html>
4 <head>
5 <title>Student Registration</title>
6 <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css">
7 <script src="{% static 'js/jquery.min.js' %}"></script>
8 </head>
9 <body>
10 <div class="container">
11 <h1>Student Registration</h1>
12 <form id="student-form" method="post">
13 {% csrf_token %}
14 {% for field in form %}
15 <div class="form-group">
16 {{ field.label_tag }}
17 {{ field }}
18 {% if field.help_text %}
19 <small class="form-text text-muted">{{ field.help_text }}</small>
20 {% endif %}
21 {% for error in field.errors %}
22 <div class="alert alert-danger">{{ error }}</div>
23 {% endfor %}
24 </div>
25 {% endfor %}
26 <button type="submit" class="btn btn-primary">Submit</button>
27 </form>
28 <div id="response"></div>
29 </div>
30
31 <script>
32 $(document).ready(function() {
33 $('#student-form').on('submit', function(e) {
34 e.preventDefault();
35 var formData = $(this).serialize();
36 $.ajax({
37 type: 'POST',
38 url: '{% url "student" %}',
39 data: formData
40 })
41 })
42 })
43 </script>

```

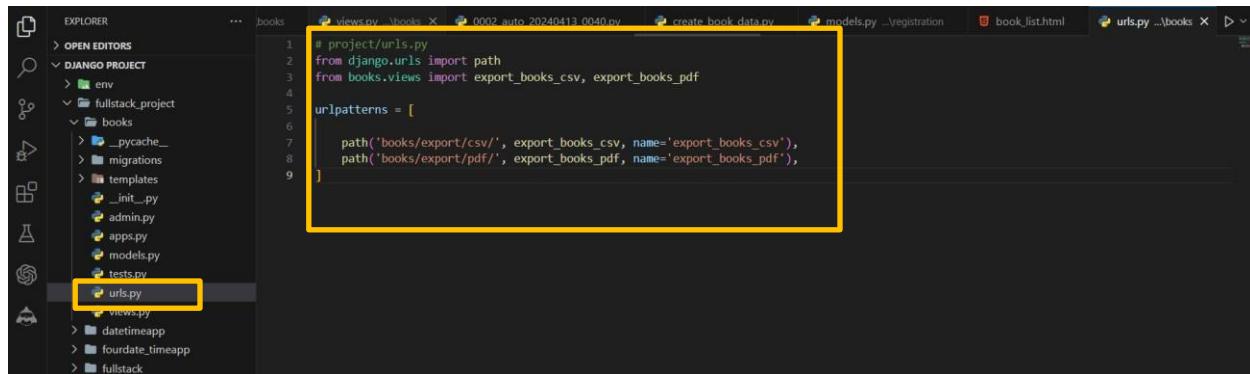
## Step-07: Update app URLs

```
project/urls.py

from django.urls import path

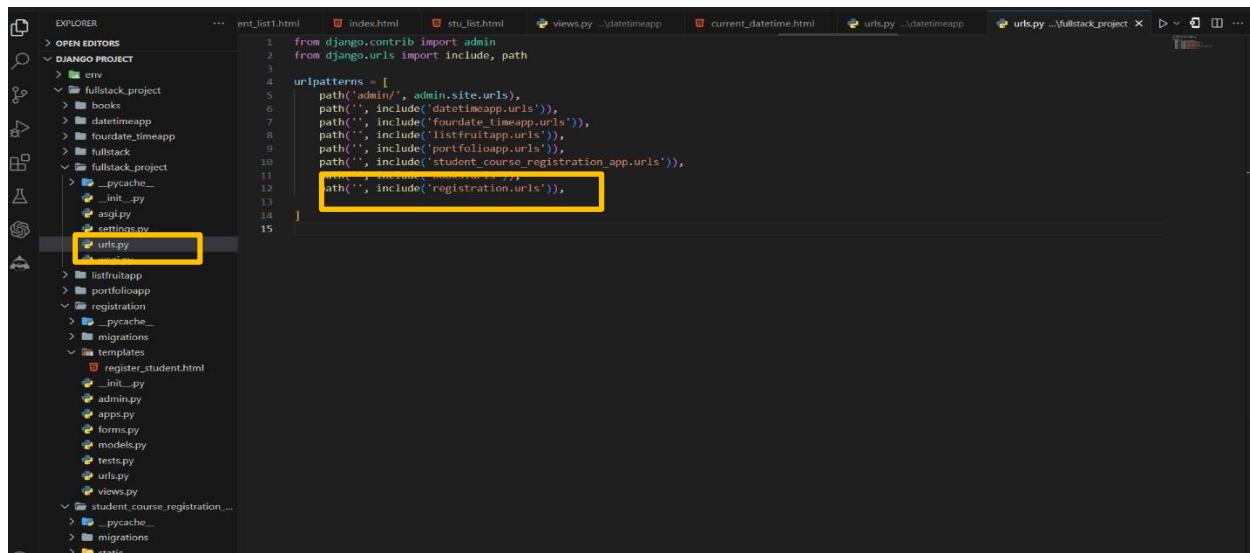
from books.views import export_books_csv, export_books_pdf

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:

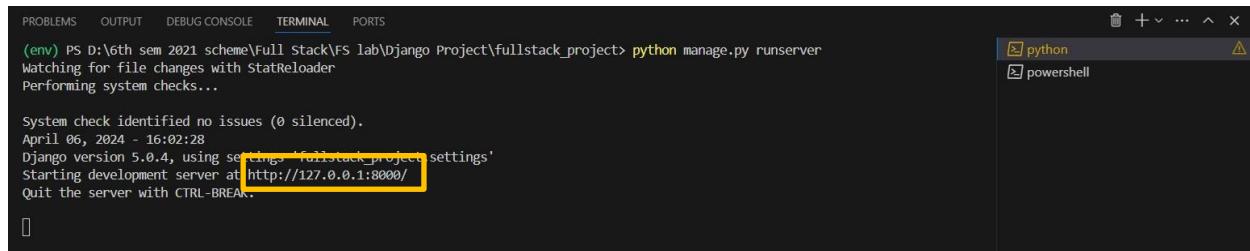


## Step-09: Run the development server

- 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/>.

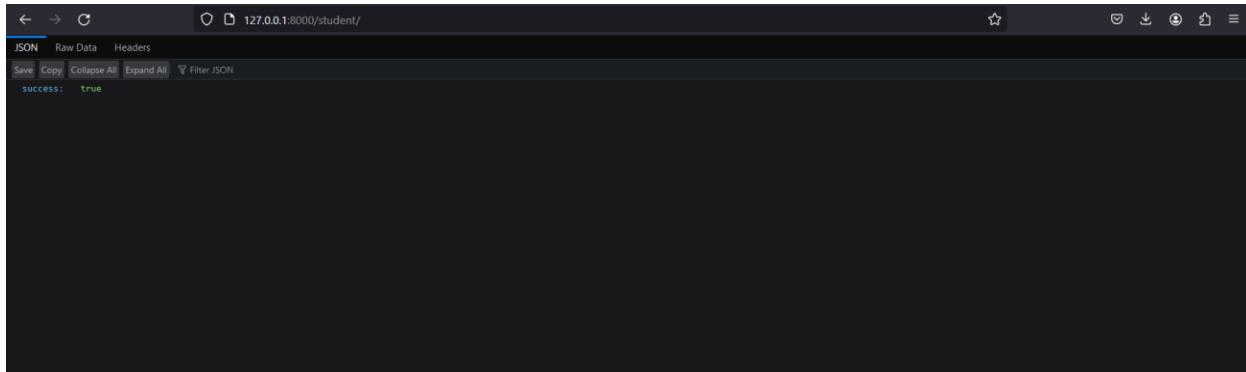


```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
(env) PS D:\6th sem 2021 scheme\Full Stack\FS lab\ Django Project\fullstack_project> python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...
System check identified no issues (0 silenced).
April 06, 2024 - 16:02:28
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.
```

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



**Fig: Student Registration Screen**



**Fig: Student Registration Success Message Screen**

## **Experiment-13**

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')
```

```

OPEN EDITORS
DJANGO PROJECT
fullstack_project
books
datetimeapp
foudrate_timeapp
fullstack
listruitapp
portfolioapp
registration
student_course_registration...
__pycache__
migrations
static
templates
init.py
admin.py
apps.py
forms.py
models.py
tests.py
urls.py
views.py
db.sqlite3
manage.py
requirements.txt

from django.shortcuts import render
from django.http import JsonResponse
from .models import Student

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 += '<p>Email: ' + student.email + '</p>';
 html += '<p>Courses: ' + student.courses.join(',') + '</p>';
 html += '';
 }
 }
 });
 }
);
});
```

```
 }

 html += '';

 } else {

 html += '<p>No students found.</p>';

 }

 $('#search-results').html(html);

}

});

});

});

</script>

</head>

<body>

{% csrf_token %}

<div class="container">

<h1>Search Students</h1>

<input type="text" id="search-input" class="form-control" placeholder="Search for a student...">

<div id="search-results"></div>

</div>

</body>

</html>
```

```

1 {% load static %}
2 <!DOCTYPE html>
3 <html>
4 <head>
5 <title>Search Students</title>
6 <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css">
7 <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
8
9
10 <script>
11 $.ajaxSetup({
12 headers: {
13 'X-CSRFToken': '{{ csrf_token }}'
14 }
15 });
16
17 $(document).ready(function() {
18 $('#search-input').on('input', function() {
19 var query = $(this).val();
20
21 $.ajax({
22 url: '{% url "search_students" %}',
23 data: {
24 'query': query
25 },
26 success: function(data) {
27 var results = data.results;
28 var html = '';
29 if (results.length > 0) {
30 html += '<ul class="list-group">';
31 for (var i = 0; i < results.length; i++) {
32 var student = results[i];
33 html += '<li class="list-group-item">';
34 html += '<h5>' + student.name + '</h5>';
35 html += '<p>Email: ' + student.email + '</p>';
36 html += '<p>Courses: ' + student.courses.join(', ') + '</p>';
37 html += '';
38 }
39 }
40 }
41 });
42 })
43 });
44 </script>
45
46 <body>
47 <div>
48 <input type="text" id="search-input" placeholder="Search Students" />
49 <ul class="list-group" id="search-results">
50 </div>
51 </body>
52 </html>

```

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

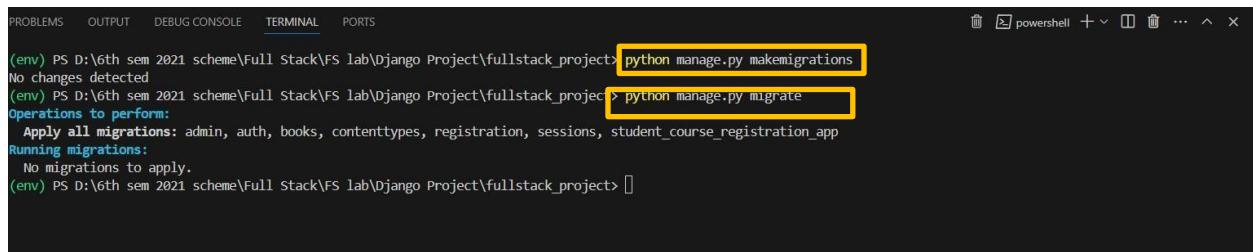
### Step-03: Update app URLs

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

```

1 from django.urls import path
2 from . import views
3
4
5 urlpatterns = [
6 path('index/', views.index, name='index'),
7 path('register-student/', views.register_student, name='register_student'),
8 path('register-course/', views.register_course, name='register_course'),
9 path('student-list/<int:course_id>', views.student_list, name='student_list'),
10 path('projects_register/', views.register_project, name='register_project'),
11 path('project_detail/<int:project_id>', views.project_detail, name='project_detail'),
12 path('project_student_list/<int:project_id>', views.project_student_list, name='project_student_list'),
13 path('students/', views.StudentListView.as_view(), name='student_list_all'),
14 path('students/<int:pk>', views.StudentDetailView.as_view(), name='student_detail'),
15 path('search/', views.search_students, name='search_students'),
16]

```

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


The screenshot shows a terminal window in VS Code with the following command history:

```
(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/>.

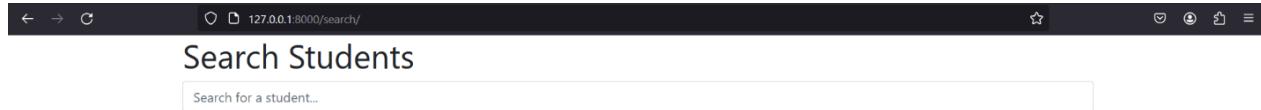


The screenshot shows a terminal window in VS Code with the following command history:

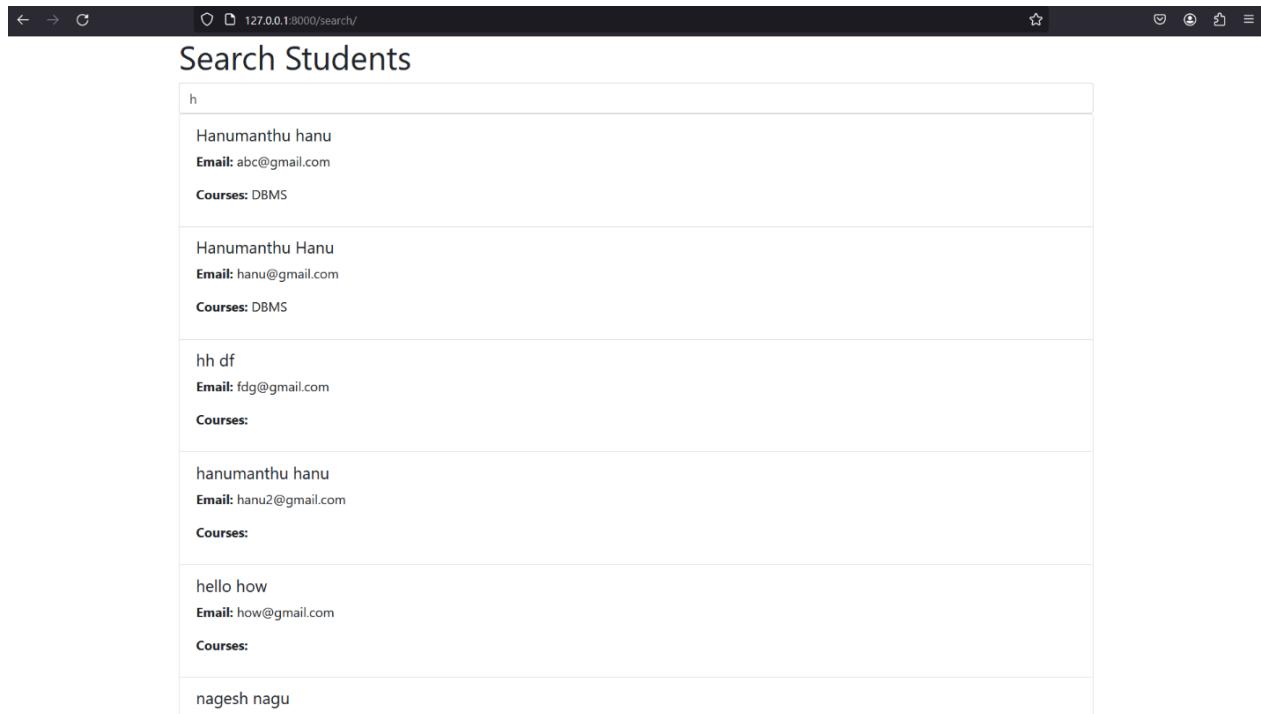
```
(env) PS D:\6th sem 2021 scheme\Full Stack\FS lab\Django Project\fullstack_project> python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

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.
```

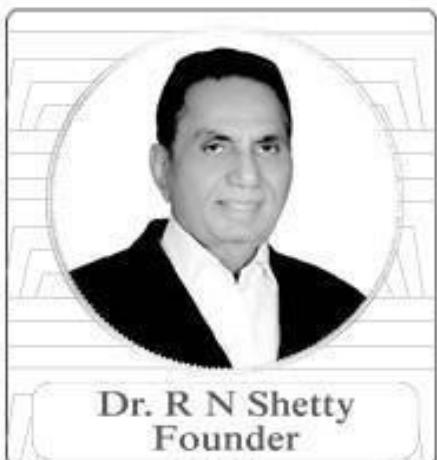
- Type or copy this <http://127.0.0.1:8000/search/>



**Fig: Before Search Result contains only Search Bar Screen**



**Fig: Search Screen**



**Dr. R N Shetty**  
Founder

## Under Graduates / Post Graduates Programs Offered

B.E. in Computer Science & Engineering  
B.E. in Information Science & Engineering  
B.E. in Artificial Intelligence & Machine Learning  
B.E. in CSE (Data Science)  
B.E. in CSE (Cyber Security)  
B.E. in Artificial Intelligence & Data Science  
B.E. in Electronics & Communication Engineering  
B.E. in Electrical & Electronics Engineering  
B.E. in Mechanical Engineering  
B.E. in Civil Engineering  
Master of Business Administration - MBA  
Master of Computer Applications - MCA

## OUR MAJOR RECRUITERS

