

BACKEND WEB DEVELOPMENT

DJANGO – A PYTHON FRAMEWORK

SOURABH PAL

LIST OF CLASSWORK

- Day 1: Display on client side (HttpResponse)
- Day 2: Display a html page from templates
- Day 3: Admin panel (adding members)
- Day 4: Pull data from database (admin panel added members information display)
- Day 5: Creating a form (Get and post methods) – applying CSS
- Day 6: function evaluation – output display
- Day 7: local library models

LIST OF CLASSWORK

- Day 8:
 - TASK 1: Decorators
 - TASK 2: JSON File
 - TASK 3: API using Django rest Framework – POSTMAN TOOL
- Day 9: restful API – GET and POST method
- DAY 10: Project Web Scraper

DOWNLOAD VISUAL STUDIO CODE AND PYTHON

DOWNLOAD LINKS:-

- <https://code.visualstudio.com/download>
- <https://www.python.org/downloads/>

DAY 1

Creating first Django project

DISPLAY CONTENT ON THE CLIENT SIDE
(BROWSER)

Creating project (use vs code terminal OR cmd)

EXECUTE THE FOLLOWING CODES SEQUENTIALLY

- `mkdir <FolderName>`
- `cd <FolderName>`
- `python -m venv env`
- `cd env`
- `Scripts\activate.bat`
- `Python -m pip install Django`
- `Django-admin startproject <ProjectName>`
- `Cd <ProjectName>`
- `Python manage.py runserver`
- `Crtl + C`
- `Code .`

NOTE THAT

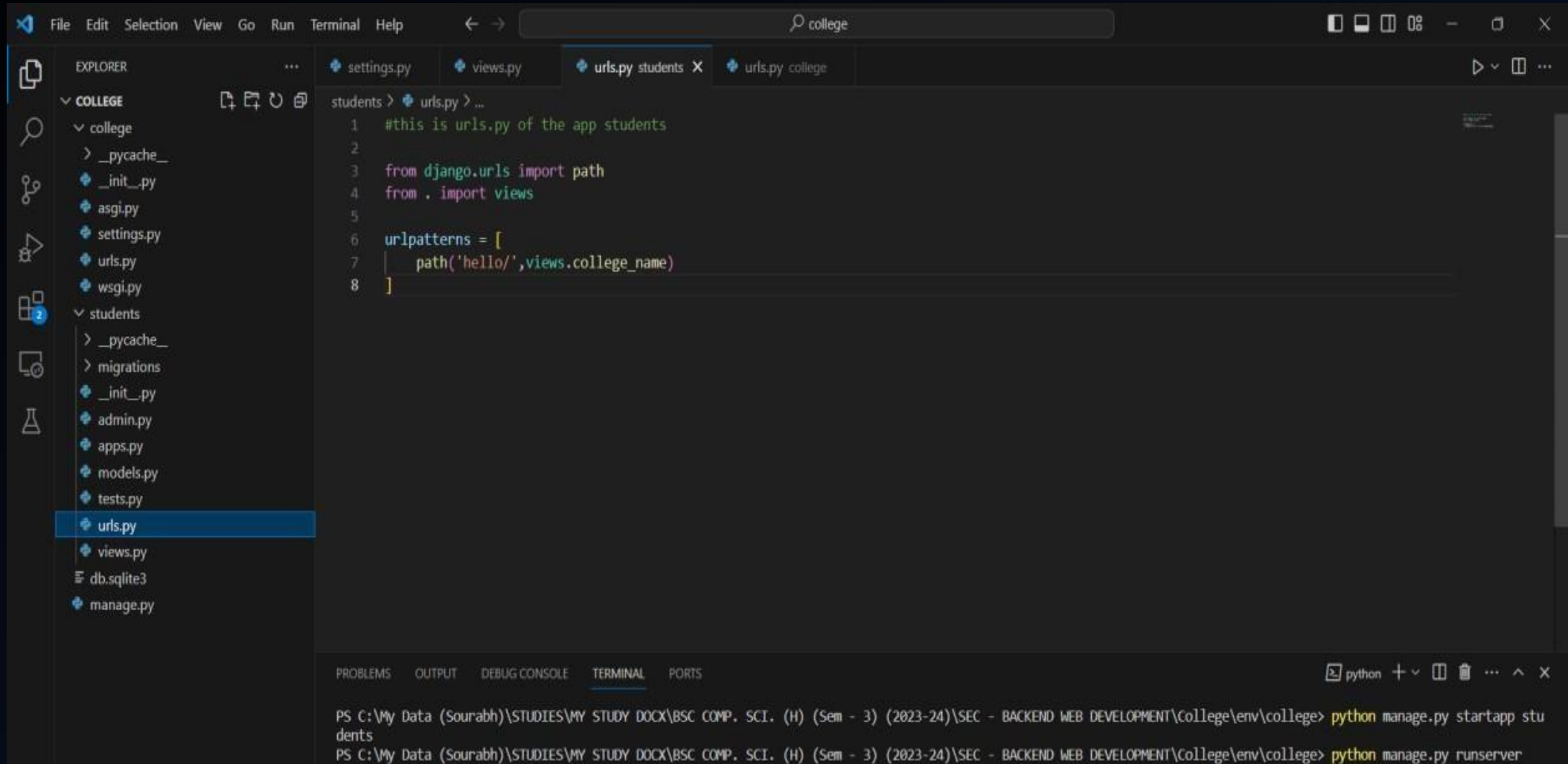
- Runserver command will generate a url which shows if the project creation is successful or not.
- To break the connection from server, press ctrl + C.
- Code . Command will open the present directories (project) in vs code (ide)

FOLLOW THESE STEPS

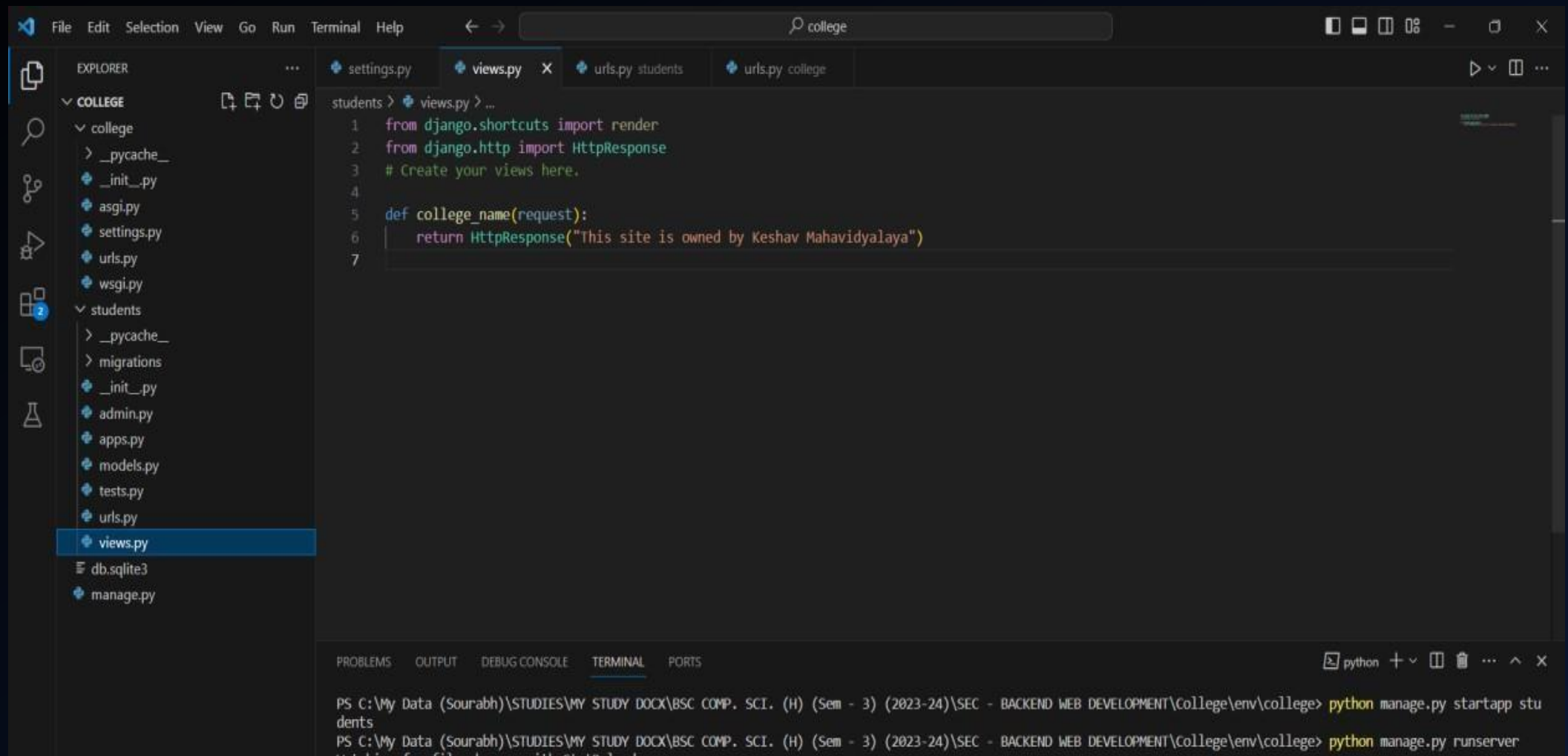
- Create app using the following command
- `Python manage.py startapp <appName>`
- Create `urls.py` in app directory
- Add the `appName` in quotes in the installed apps section of `settings.py` of the project
- Then make the following changes in `views.py`, `urls.py` (project and app).
- In the given example
- Directory name = College
- Project name = college
- App name = students

PROJECT – URLS.PY

APP – URLS.PY



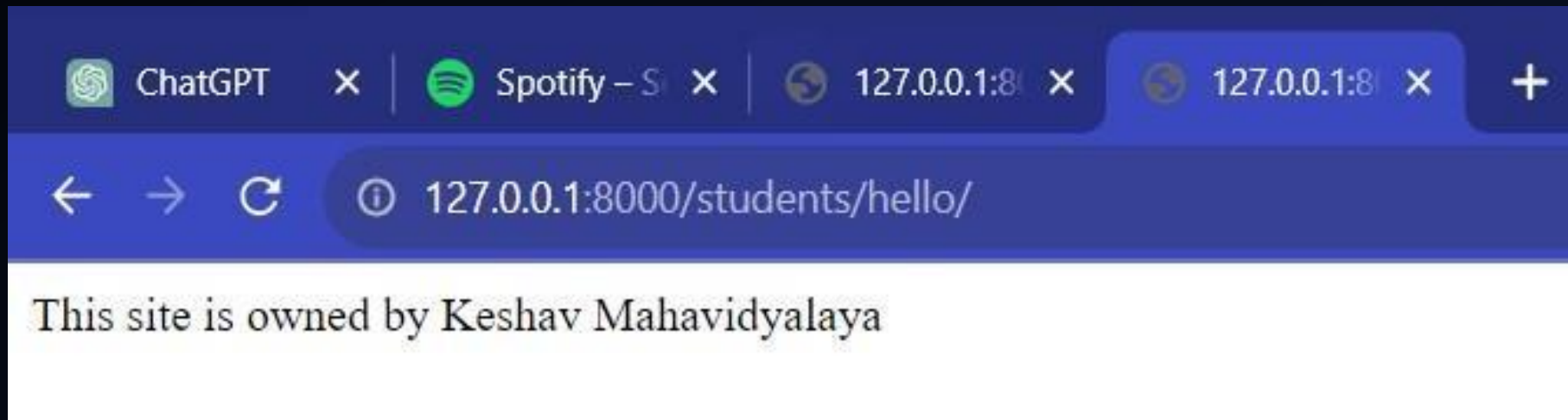
VIEWS.PY



BROWSER

- Run the following command
- `Python manage.py runserver`
- Then go to the url generated
- Then append the url with `/students/hello`

RESULT



DAY 2

Creating templates

DISPLAY HTML PAGES ON THE CLIENT SIDE
(BROWSER)

TEMPLATE – HTML PAGE

- To the same project, make the following changes
- Create templates folder inside the app directory and create html page inside it.
- Then Change views.py

VIEWS.PY

settings.py

views.py X

structure.html

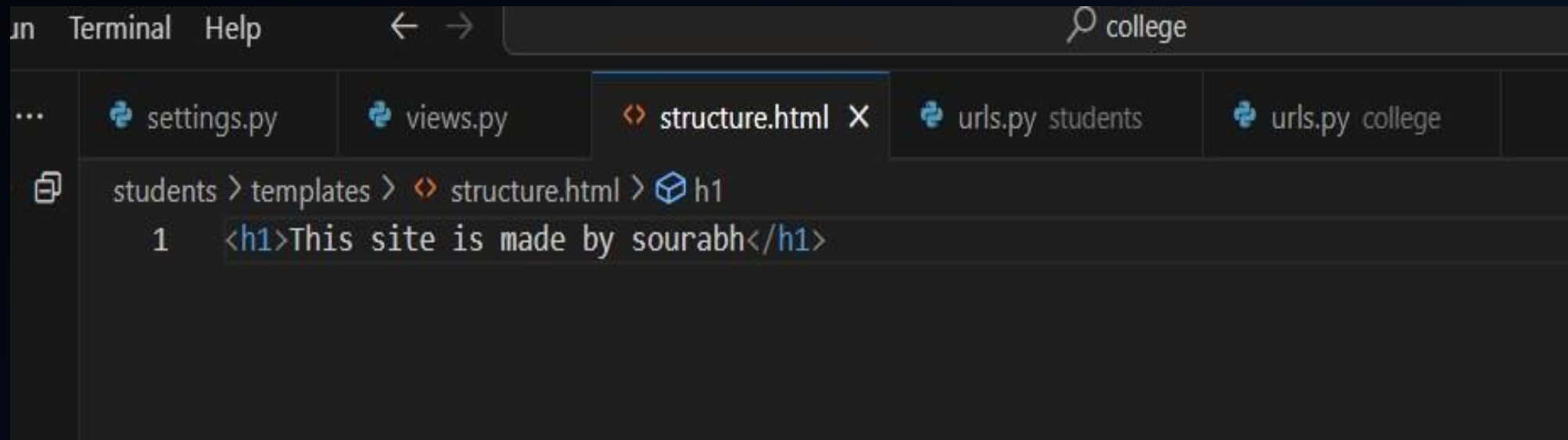
urls.py students

urls.py college

students > views.py > college_name

```
1 from django.shortcuts import render
2 from django.http import HttpResponse
3 # Create your views here.
4
5 def college_name(request):
6     #return HttpResponse("This site is owned by Keshav Mahavidyalaya")
7     return render(request, 'structure.html')
```


HTML PAGE INSIDE TEMPLATES



The screenshot shows a web browser window with a dark theme. The address bar at the top contains the text "college". Below the address bar, there are several tabs: "settings.py", "views.py", "structure.html" (which is the active tab and has a close button 'X'), "urls.py students", and "urls.py college". The main content area of the browser shows the file path "students > templates > structure.html" followed by an "h1" tag icon. Below this, the first line of code is displayed: "1 <h1>This site is made by sourabh</h1>". The code is syntax-highlighted, with the opening and closing tags in blue and the text in white.

```
students > templates > structure.html > h1
1 <h1>This site is made by sourabh</h1>
```

BROWSER



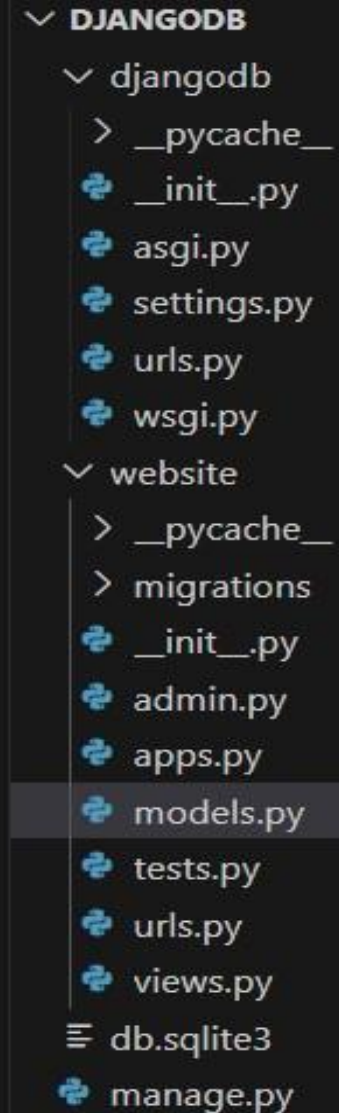
DAY 3

The Admin panel

USER ADDING DATA TO THE DATABASE

INFORMATION

- Create another project
- Exampe given:
- Directory name = DjangoDb
- Project name =.djangodb
- Appname = website



A screenshot of a file explorer window showing the directory structure of a Django project. The root directory is 'DJANGODB', which contains a subdirectory 'djangodb' and another subdirectory 'website'. The 'djangodb' directory contains files: '__pycache__', '__init__.py', 'asgi.py', 'settings.py', 'urls.py', and 'wsgi.py'. The 'website' directory contains files: '__pycache__', 'migrations', '__init__.py', 'admin.py', 'apps.py', 'models.py' (which is highlighted), 'tests.py', 'urls.py', 'views.py', a subdirectory 'db.sqlite3', and 'manage.py'.

```
▼ DJANGODB
  ▼ djangodb
    > __pycache__
    • __init__.py
    • asgi.py
    • settings.py
    • urls.py
    • wsgi.py
  ▼ website
    > __pycache__
    > migrations
    • __init__.py
    • admin.py
    • apps.py
    • models.py
    • tests.py
    • urls.py
    • views.py
    db.sqlite3
    • manage.py
```

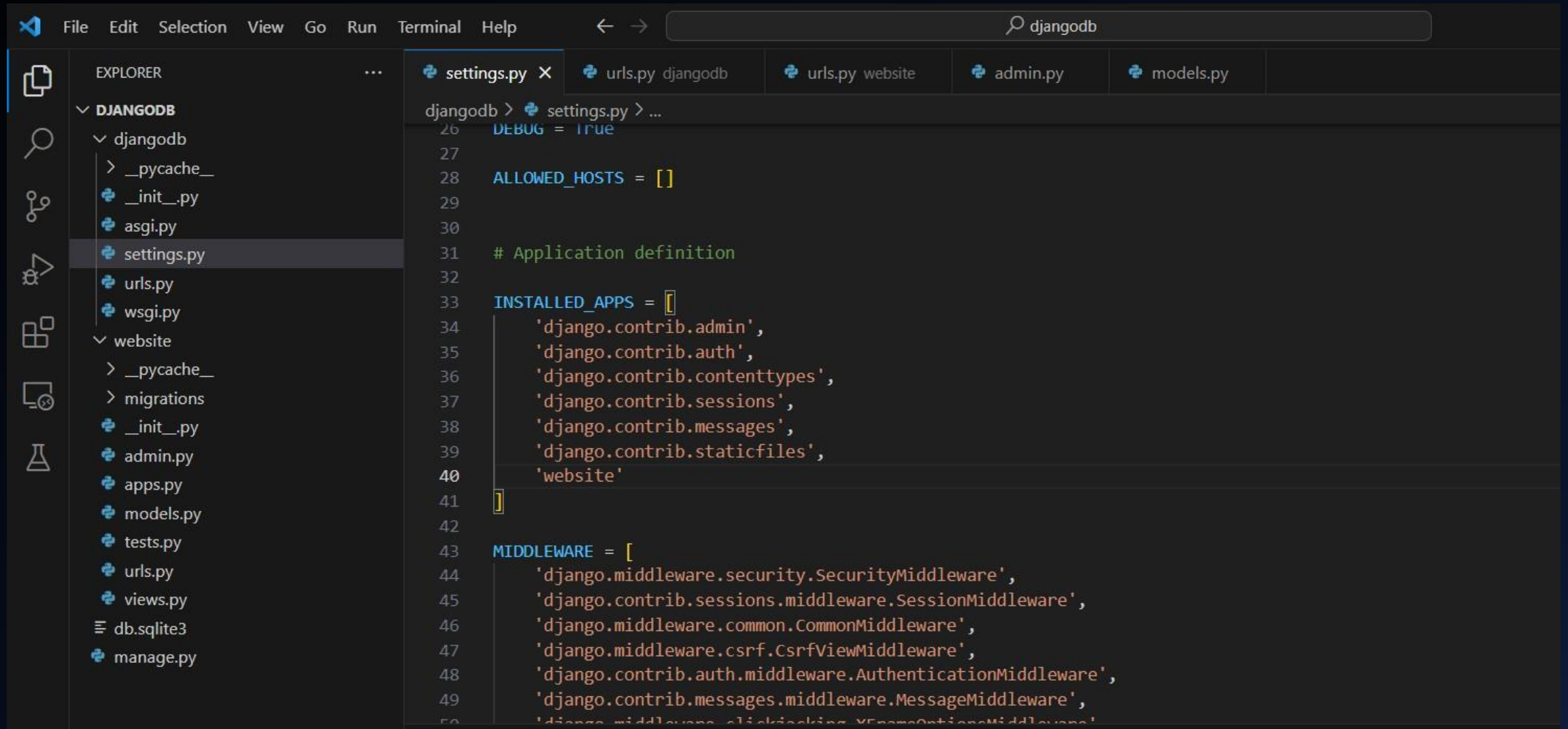
INSTRUCTIONS

- Run -> `python manage.py migrate`
- Then make changes in the `models.py` and other files as shown in next slides
- Then create an admin user by running the command -> `python manage.py createsuperuser`
- Then create a user – remember the username and password
- Then run the command -> `python manage.py makemigrations`
- Then again -> `python manage.py migrate`
- `Python manage.py runserver`



CHANGES BEFORE RUNNING THE INSTRUCTED COMMANDS

SETTINGS.PY



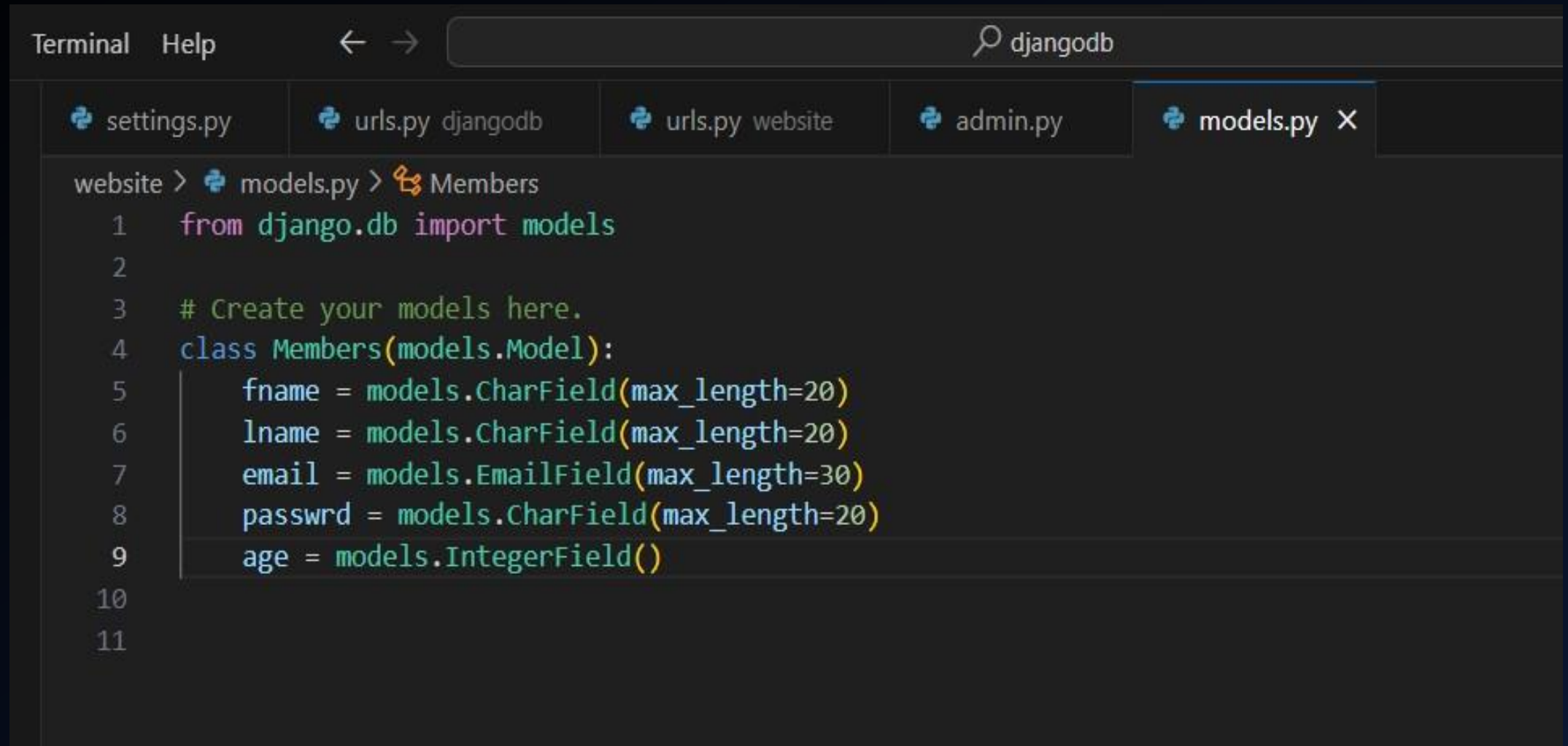
The screenshot shows a code editor with a dark theme. On the left is the 'EXPLORER' sidebar showing a file tree for a project named 'djangodb'. The tree includes a 'djangodb' folder with subfolders '__pycache__' and files like '__init__.py', 'asgi.py', 'settings.py' (which is selected), 'urls.py', and 'wsgi.py'. There is also a 'website' folder with its own '__pycache__' and 'migrations' subfolders, and files like '__init__.py', 'admin.py', 'apps.py', 'models.py', 'tests.py', 'urls.py', 'views.py', 'db.sqlite3', and 'manage.py'. The main editor area displays the 'settings.py' file for the 'djangodb' project. The code includes a 'DEBUG' setting set to 'True', an empty 'ALLOWED_HOSTS' list, and an 'INSTALLED_APPS' list containing 'django.contrib.admin', 'django.contrib.auth', 'django.contrib.contenttypes', 'django.contrib.sessions', 'django.contrib.messages', 'django.contrib.staticfiles', and 'website'. A 'MIDDLEWARE' list is also shown, containing various Django middleware classes like 'SecurityMiddleware', 'SessionMiddleware', 'CommonMiddleware', 'CsrfViewMiddleware', 'AuthenticationMiddleware', 'MessageMiddleware', and 'XFrameOptionsMiddleware'.

```
File Edit Selection View Go Run Terminal Help
djangodb

EXPLORER
  DJANGODB
    djangodb
      __pycache__
      __init__.py
      asgi.py
      settings.py
      urls.py
      wsgi.py
    website
      __pycache__
      migrations
      __init__.py
      admin.py
      apps.py
      models.py
      tests.py
      urls.py
      views.py
      db.sqlite3
      manage.py

djangodb > settings.py > ...
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     'website'
41 ]
42
43 MIDDLEWARE = [
44     'django.middleware.security.SecurityMiddleware',
45     'django.contrib.sessions.middleware.SessionMiddleware',
46     'django.middleware.common.CommonMiddleware',
47     'django.middleware.csrf.CsrfViewMiddleware',
48     'django.contrib.auth.middleware.AuthenticationMiddleware',
49     'django.contrib.messages.middleware.MessageMiddleware',
50     'django.middleware.clickjacking.XFrameOptionsMiddleware',
```

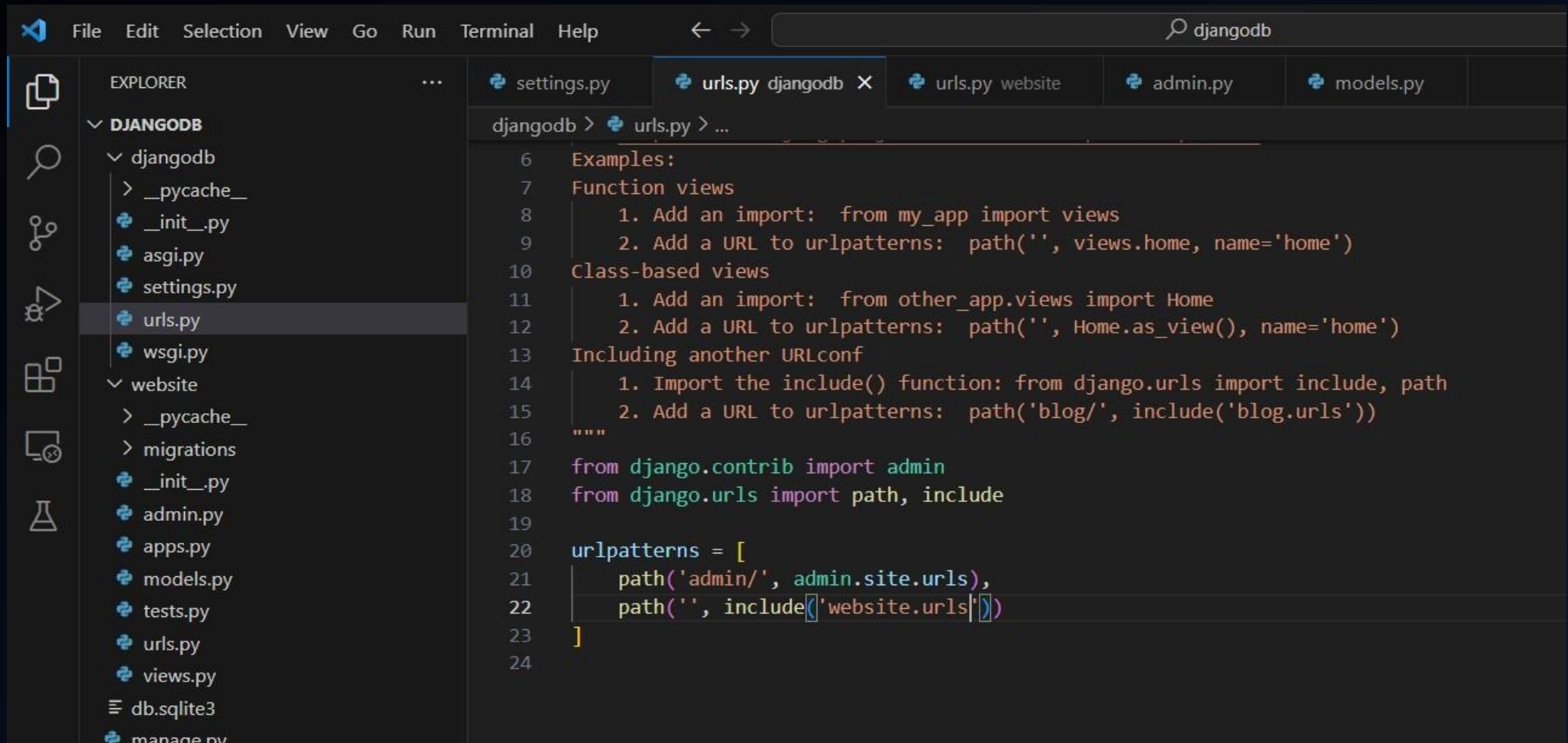
MODELS.PY



The image shows a code editor window with a dark theme. At the top, there's a menu bar with 'Terminal' and 'Help'. Below the menu bar is a toolbar with navigation arrows and a search bar containing 'djangodb'. The editor has several tabs open: 'settings.py', 'urls.py djangodb', 'urls.py website', 'admin.py', and 'models.py' (which is the active tab). The code in the 'models.py' tab is as follows:

```
website > models.py > Members
1  from django.db import models
2
3  # Create your models here.
4  class Members(models.Model):
5      fname = models.CharField(max_length=20)
6      lname = models.CharField(max_length=20)
7      email = models.EmailField(max_length=30)
8      passwd = models.CharField(max_length=20)
9      age = models.IntegerField()
10
11
```

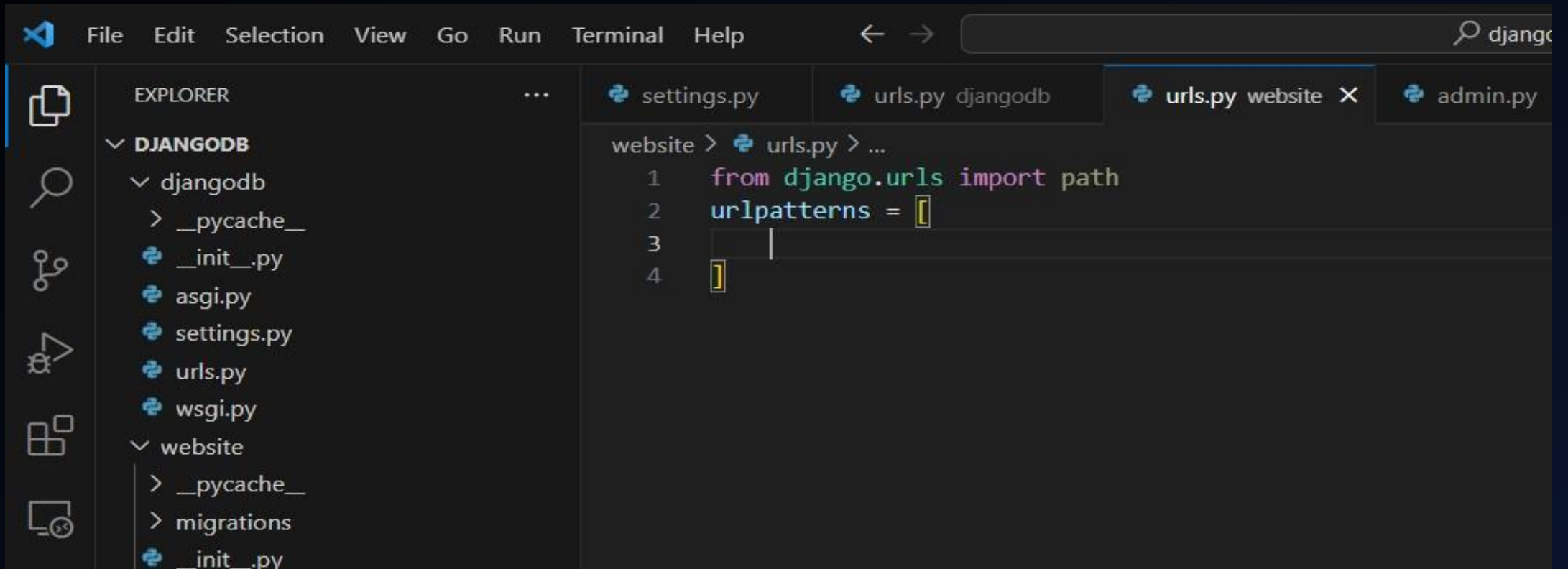

PROJECT – URLS.PY



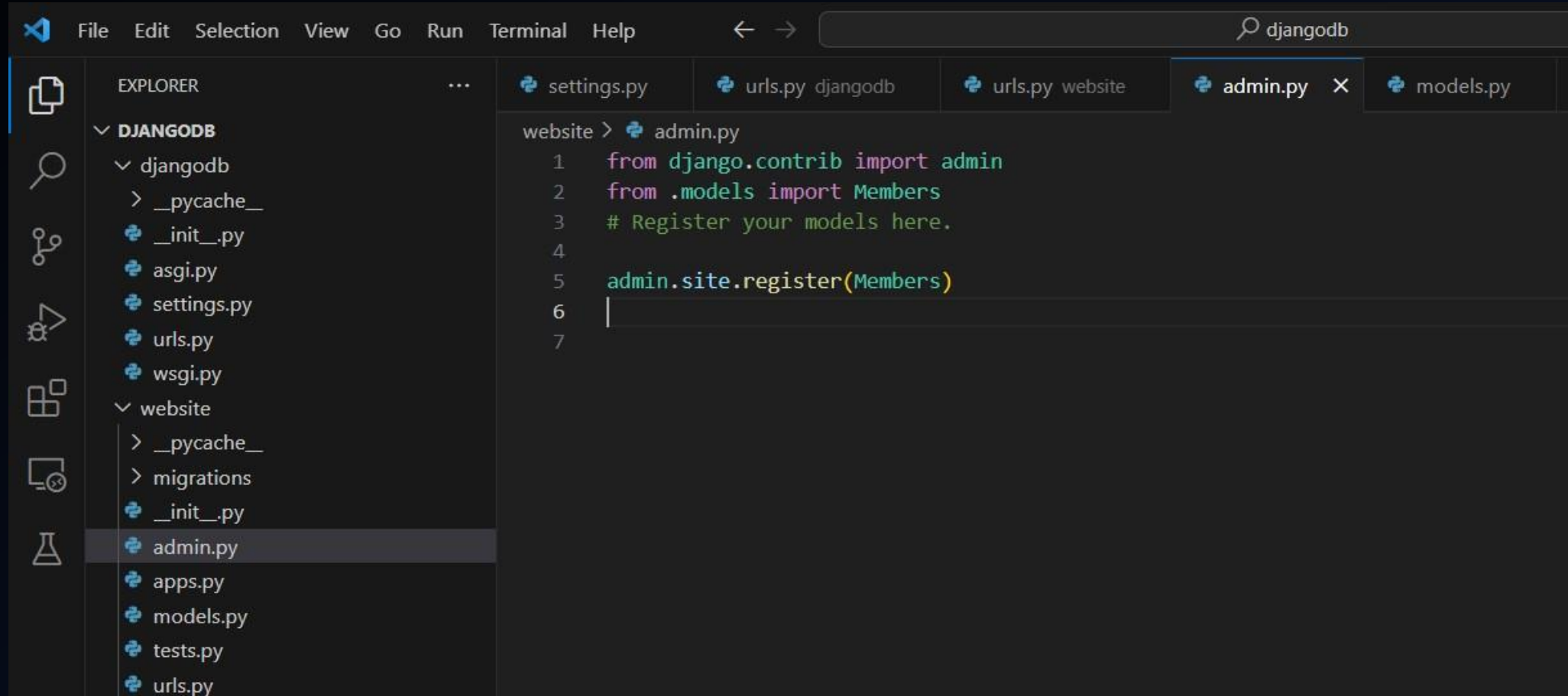
The screenshot shows a code editor with a dark theme. On the left is the Explorer sidebar showing a project named 'DJANGOODB'. Under 'django' are files like '__pycache__', '__init__.py', 'asgi.py', 'settings.py', 'urls.py' (selected), and 'wsgi.py'. Under 'website' are '__pycache__', 'migrations', '__init__.py', 'admin.py', 'apps.py', 'models.py', 'tests.py', 'urls.py', 'views.py', 'db.sqlite3', and 'manage.py'. The top menu bar includes File, Edit, Selection, View, Go, Run, Terminal, and Help. The top right shows a search bar with 'django' and several open tabs: 'settings.py', 'urls.py django' (active), 'urls.py website', 'admin.py', and 'models.py'. The main editor area shows the content of 'urls.py' with line numbers 6 to 24. The code includes comments for examples of function and class-based views, and a list of URL patterns for the 'admin' and 'website' apps.

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

APP – URLS.PY



ADMIN.PY



The screenshot shows a code editor with a dark theme. The Explorer panel on the left shows a project structure with a folder named 'DJANGOODB' containing a subfolder 'djangoedb' and a folder 'website'. The 'website' folder contains files like '__pycache__', 'migrations', '__init__.py', 'admin.py', 'apps.py', 'models.py', 'tests.py', and 'urls.py'. The 'admin.py' file is selected and its content is displayed in the main editor area. The code in 'admin.py' imports 'admin' from 'django.contrib' and 'Members' from '.models', and registers 'Members' with 'admin.site.register(Members)'. The top of the editor shows a menu bar with 'File', 'Edit', 'Selection', 'View', 'Go', 'Run', 'Terminal', and 'Help'. The top right shows a search bar with 'djangoedb' and a tab bar with 'settings.py', 'urls.py djangoedb', 'urls.py website', 'admin.py', and 'models.py'.

```
website > admin.py
1  from django.contrib import admin
2  from .models import Members
3  # Register your models here.
4
5  admin.site.register(Members)
6
7
```

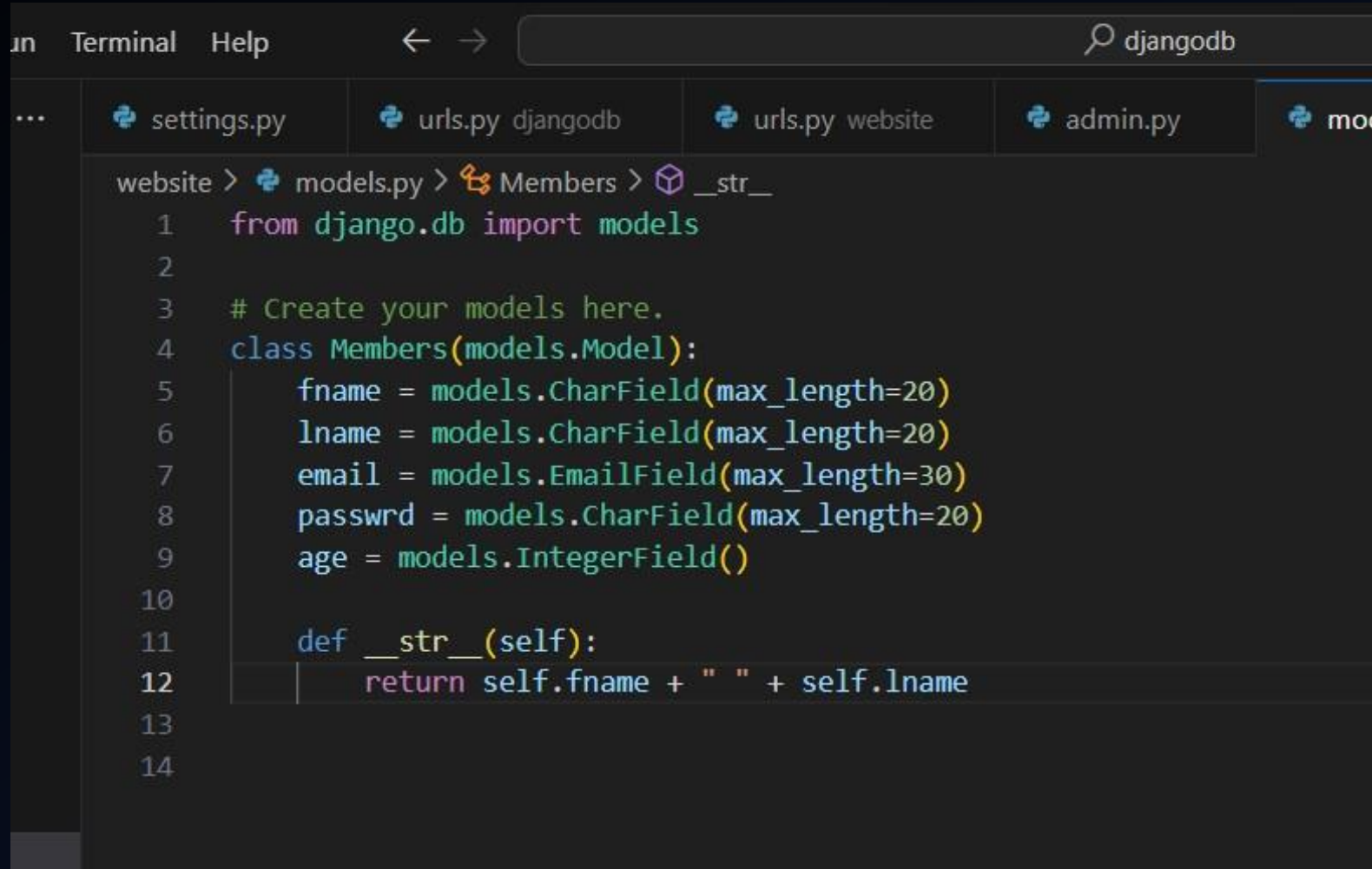
ADDING MEMBERS ON THE CLIENT SIDE

The screenshot displays the Django administration interface in a web browser. The browser's address bar shows the URL `127.0.0.1:8000/admin/website/members/add/`. The page title is "Django administration". The left sidebar contains a navigation menu with "AUTHENTICATION AND AUTHORIZATION" (Groups, Users) and "WEBSITE" (Members). The main content area is titled "Add members" and contains a form with fields for "Fname:", "Lname:", "Email:", "Password:", and "Age:". Below the form are three buttons: "SAVE", "Save and add another", and "Save and continue editing".

Below the form, a green message bar states: "The members 'Members object (1)' was added successfully."

Below the message bar, there is a section titled "Select members to change" with an "ADD MEMBERS +" button. It includes an "Action:" dropdown menu, a "Go" button, and a list of members. The list shows "MEMBERS" and "Members object (1)". Below the list, it says "1 members".

UPDATED MODELS.PY – TO DISPLAY NAME LIST



```
django.db Terminal Help  ← →  django.db  
... settings.py urls.py django.db urls.py website admin.py models.py  
website > models.py > Members > __str__  
1  from django.db import models  
2  
3  # Create your models here.  
4  class Members(models.Model):  
5      fname = models.CharField(max_length=20)  
6      lname = models.CharField(max_length=20)  
7      email = models.EmailField(max_length=30)  
8      passwd = models.CharField(max_length=20)  
9      age = models.IntegerField()  
10  
11     def __str__(self):  
12         return self.fname + " " + self.lname  
13  
14
```

DAY 4

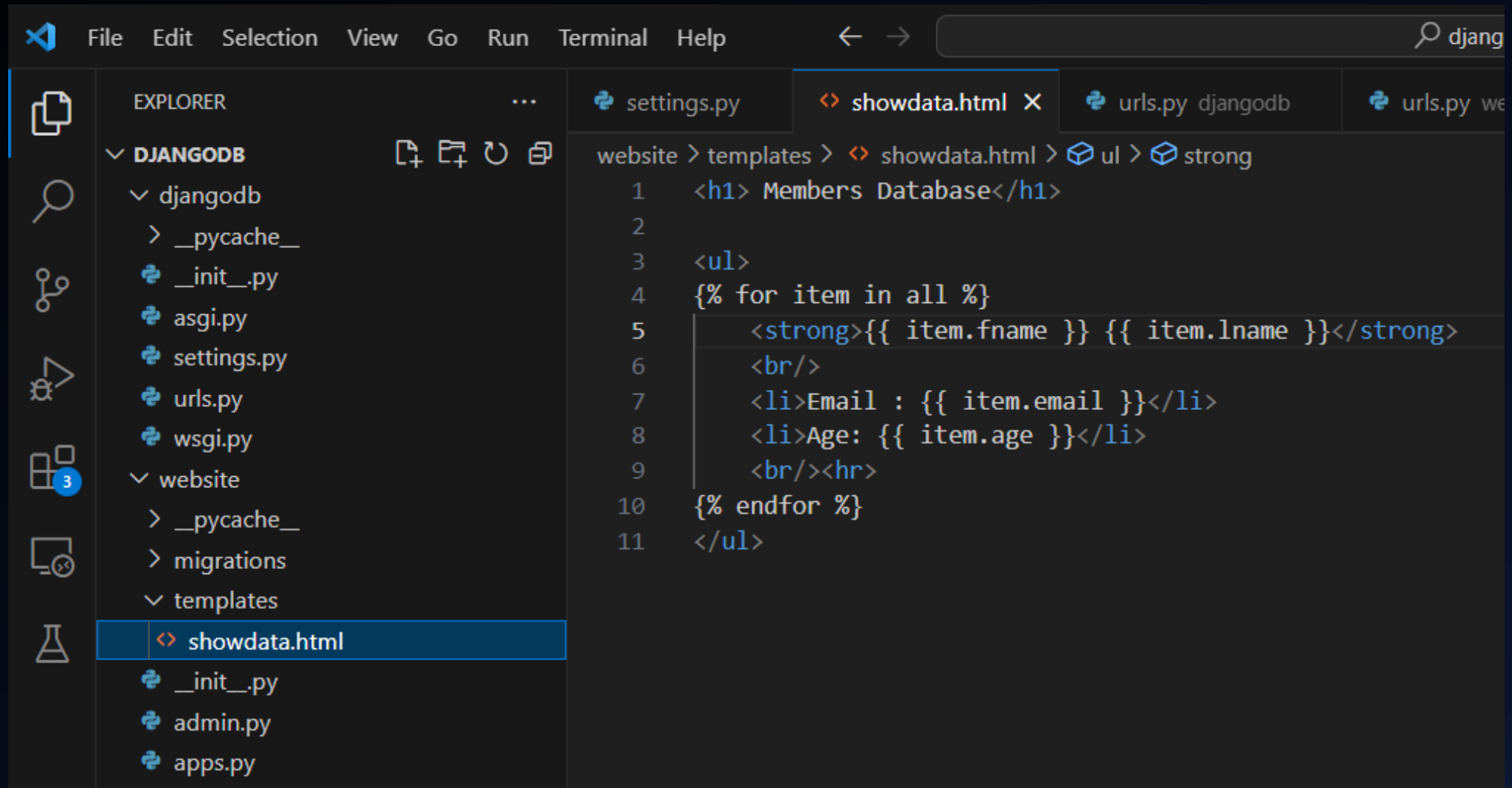
Pull data from database

EXTRACT DATA FROM DATABASE AND
DISPLAY ON THE CLIENT SIDE

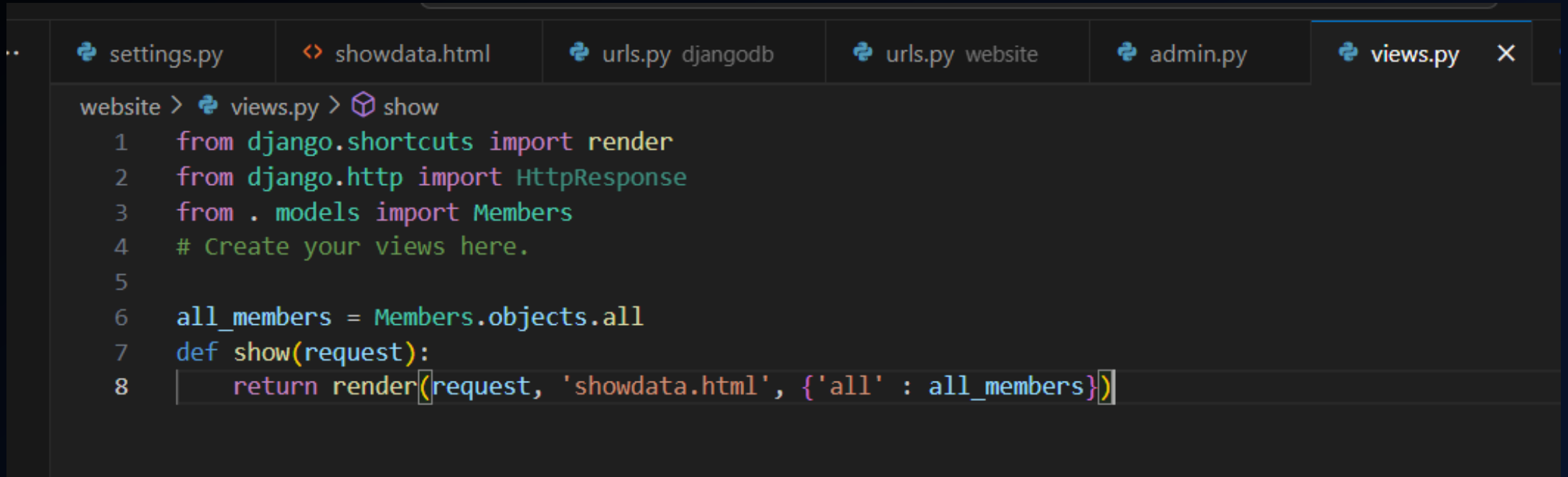
INSTRUCTIONS

- This is continuation of the previous project in which we display the content of the database on to the client screen.
- These data are entered by the user in the admin panel earlier.
- We need to create an html file for this
- Let's name it showdata.html
- Make changes to the html page in the templates

SHOWDATA.HTML



VIEWS.PY



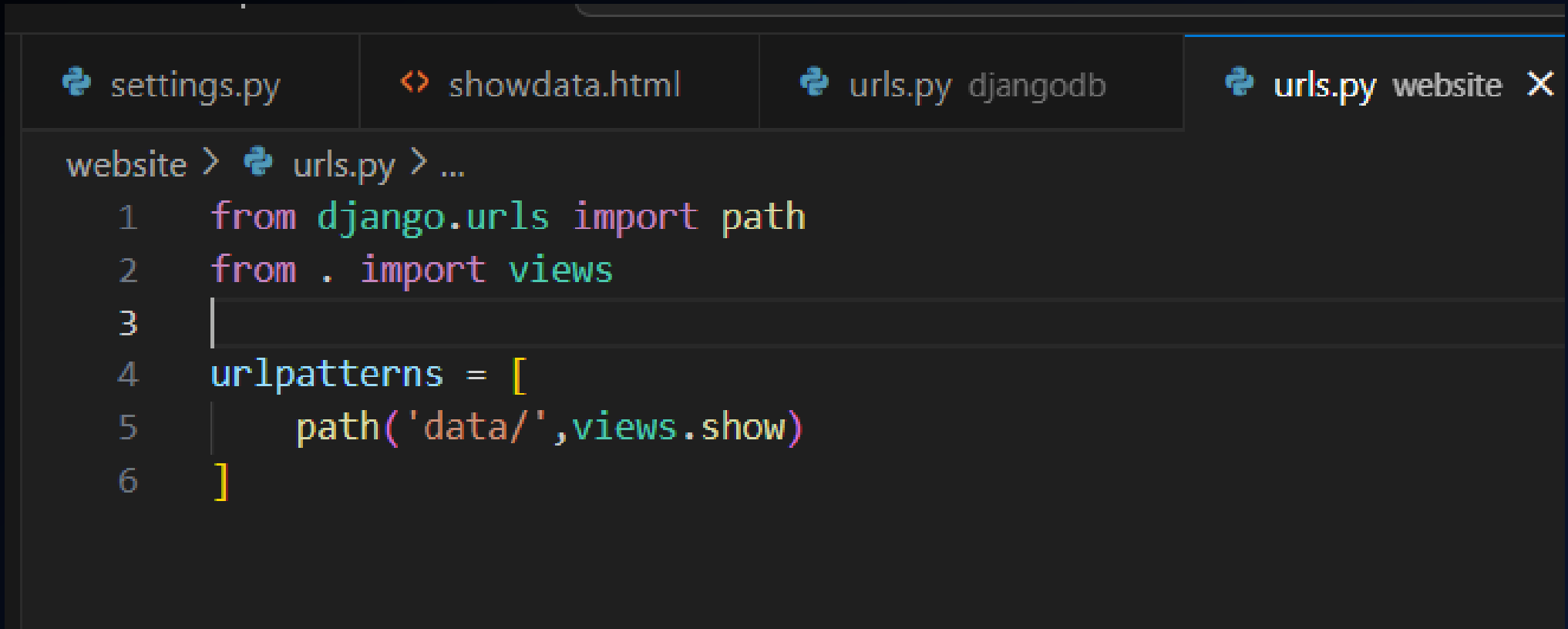
The image shows a code editor window with several tabs at the top: settings.py, showdata.html, urls.py.djangodb, urls.py.website, admin.py, and views.py (which is the active tab). The code in the views.py file is as follows:

```
website > views.py > show
1  from django.shortcuts import render
2  from django.http import HttpResponse
3  from .models import Members
4  # Create your views here.
5
6  all_members = Members.objects.all
7  def show(request):
8      return render(request, 'showdata.html', {'all' : all_members})
```

PROJECT – URLS.PY

```
15 | 2. Add a URL to urlpatterns: path('blog/', in
16 | """
17 | from django.contrib import admin
18 | from django.urls import path, include
19 |
20 | urlpatterns = [
21 |     path('admin/', admin.site.urls),
22 |     path('website/', include('website.urls'))
23 | ]
24 |
```

APP – URLS.PY

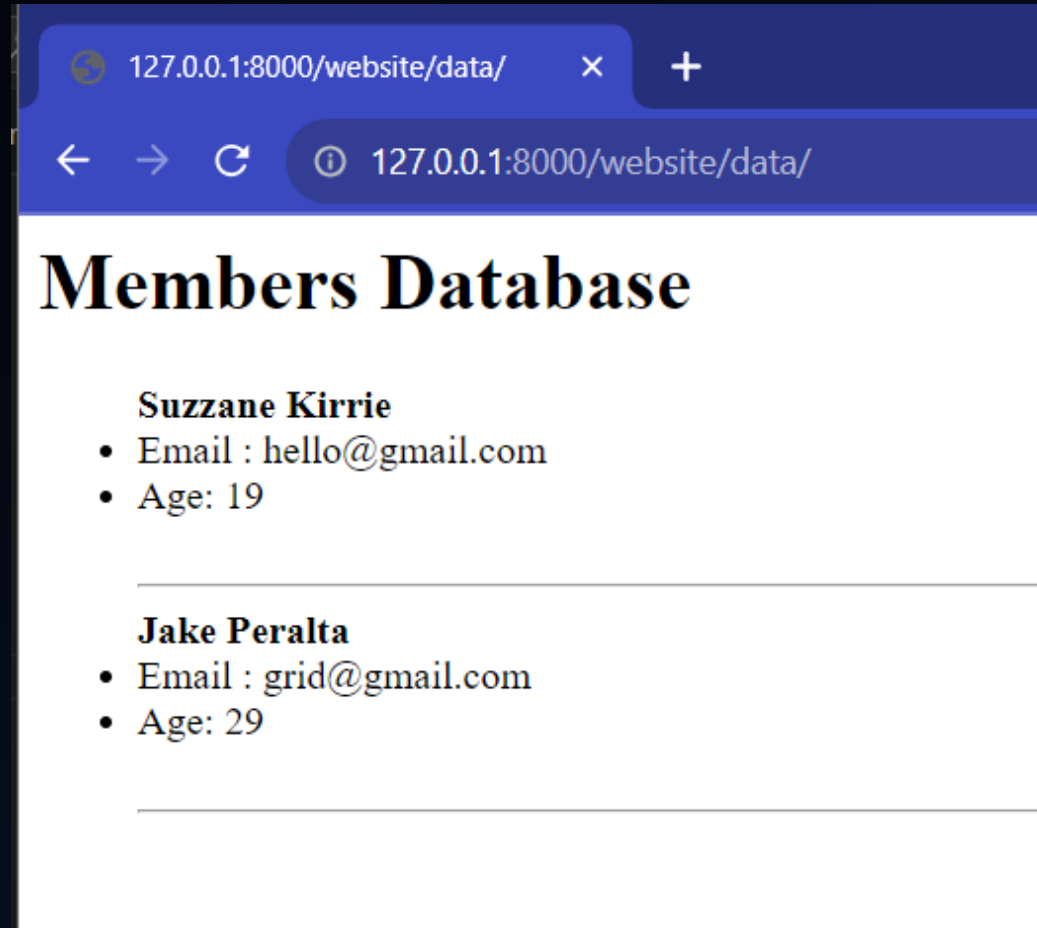


The screenshot shows a code editor with four tabs at the top: 'settings.py', 'showdata.html', 'urls.py.djangodb', and 'urls.py.website'. The 'urls.py.website' tab is active. The code in the editor is as follows:

```
website > urls.py > ...
1  from django.urls import path
2  from . import views
3
4  urlpatterns = [
5      path('data/', views.show)
6  ]
```

No change in admin.py and models.py

CLIENT SIDE



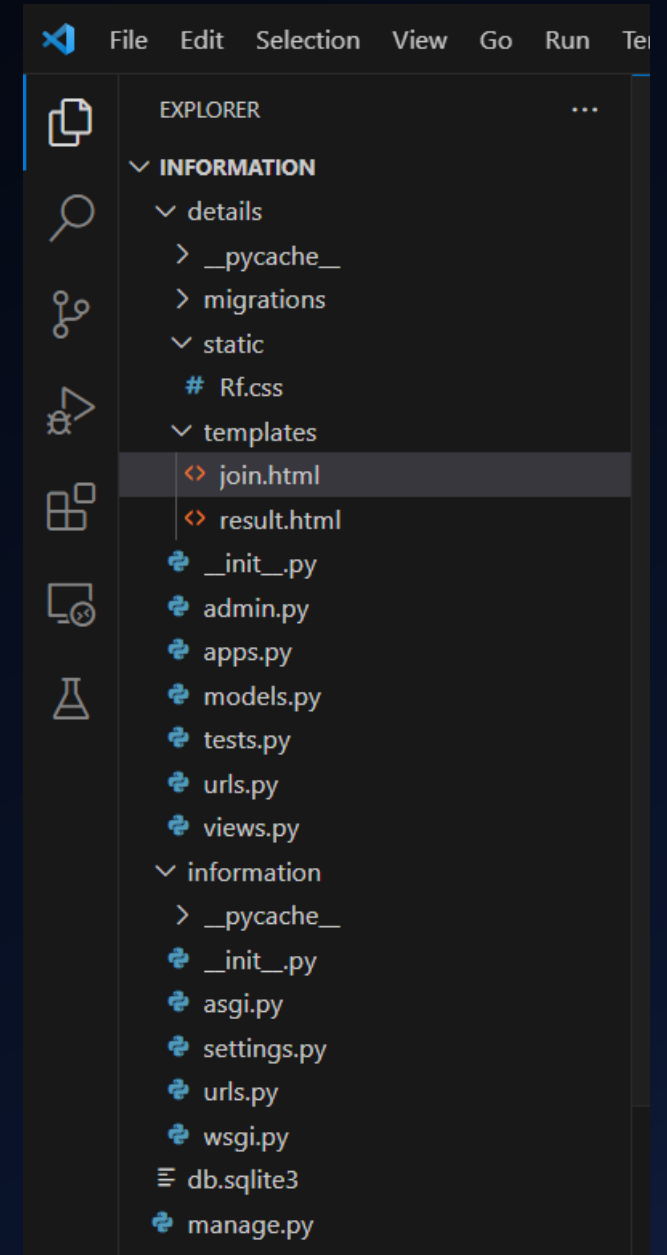
DAY 5

Creating Form and applying CSS

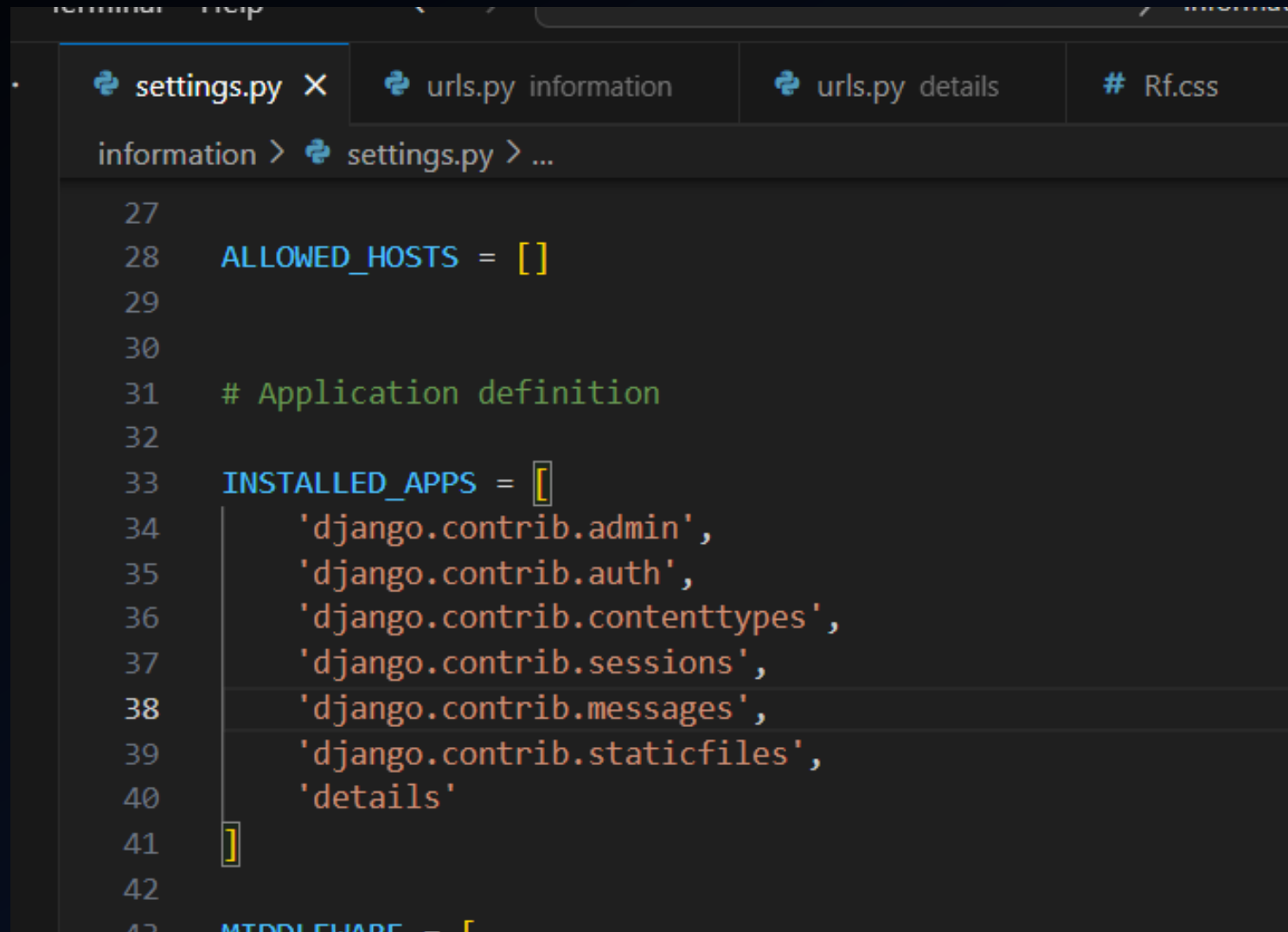
TAKING INPUT FROM USER AND
SHOWING DESIRED OUTPUT

INSTRUCTIONS

- Directory name = INFO
- Project name = information
- App name = details
- Html page 1 = join.html
- Html page 2 = result.html
- Views.py functions = join(), result()
- Make the following changes



SETTINGS.PY

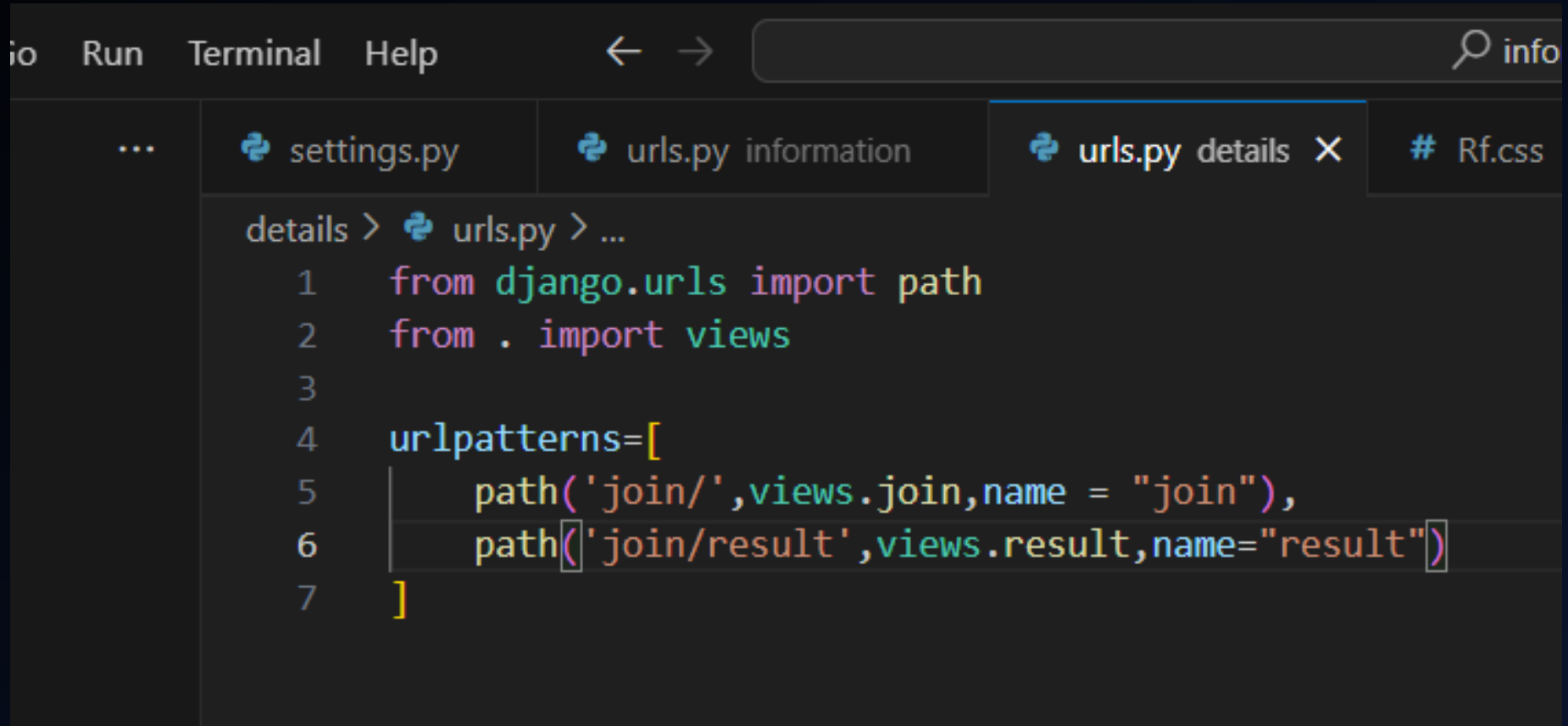


```
27
28 ALLOWED_HOSTS = []
29
30
31 # Application definition
32
33 INSTALLED_APPS = [
34     'django.contrib.admin',
35     'django.contrib.auth',
36     'django.contrib.contenttypes',
37     'django.contrib.sessions',
38     'django.contrib.messages',
39     'django.contrib.staticfiles',
40     'details'
41 ]
42
43 MIDDLEWARE = [
```

PROJECT – URLS.PY

```
... settings.py urls.py information X urls.py details # Rf.css views.py
information > urls.py > ...
1  """
2  URL configuration for information project.
3
4  The `urlpatterns` list routes URLs to views. For more information please see:
5  | https://docs.djangoproject.com/en/4.2/topics/http/urls/
6  | Examples:
7  | Function views
8  |     1. Add an import: from my_app import views
9  |     2. Add a URL to urlpatterns: path('', views.home, name='home')
10 | Class-based views
11 |     1. Add an import: from other_app.views import Home
12 |     2. Add a URL to urlpatterns: path('', Home.as_view(), name='home')
13 | Including another URLconf
14 |     1. Import the include() function: from django.urls import include, path
15 |     2. Add a URL to urlpatterns: path('blog/', include('blog.urls'))
16 | """
17 from django.contrib import admin
18 from django.urls import path,include
19
20 urlpatterns = [
21     path('admin/', admin.site.urls),
22     path('details/', include('details.urls'))
23 ]
24
```

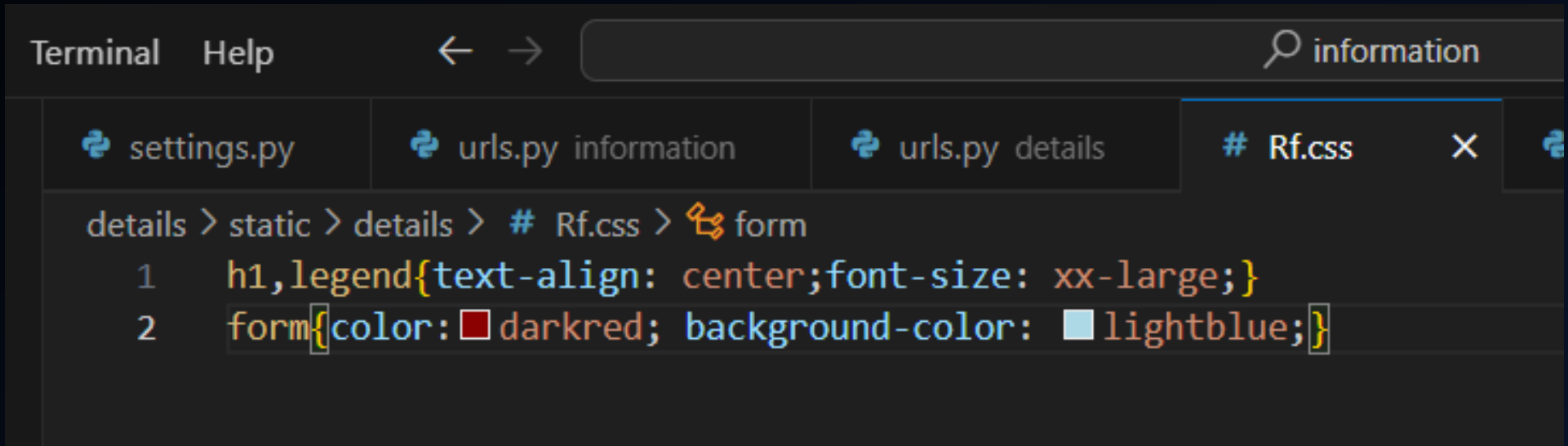

APP – URLS.PY



The image shows a code editor window with a dark theme. The top menu bar includes 'io', 'Run', 'Terminal', and 'Help'. To the right of the menu is a search bar with a magnifying glass icon and the text 'info'. Below the menu bar, there are several tabs: '...', 'settings.py', 'urls.py information', 'urls.py details' (which is active and has a close button 'X'), and '# Rf.css'. The main area of the editor displays the code for 'urls.py'. The code is as follows:

```
details > urls.py > ...
1  from django.urls import path
2  from . import views
3
4  urlpatterns=[
5      path('join/',views.join,name = "join"),
6      path('join/result',views.result,name="result")
7  ]
```

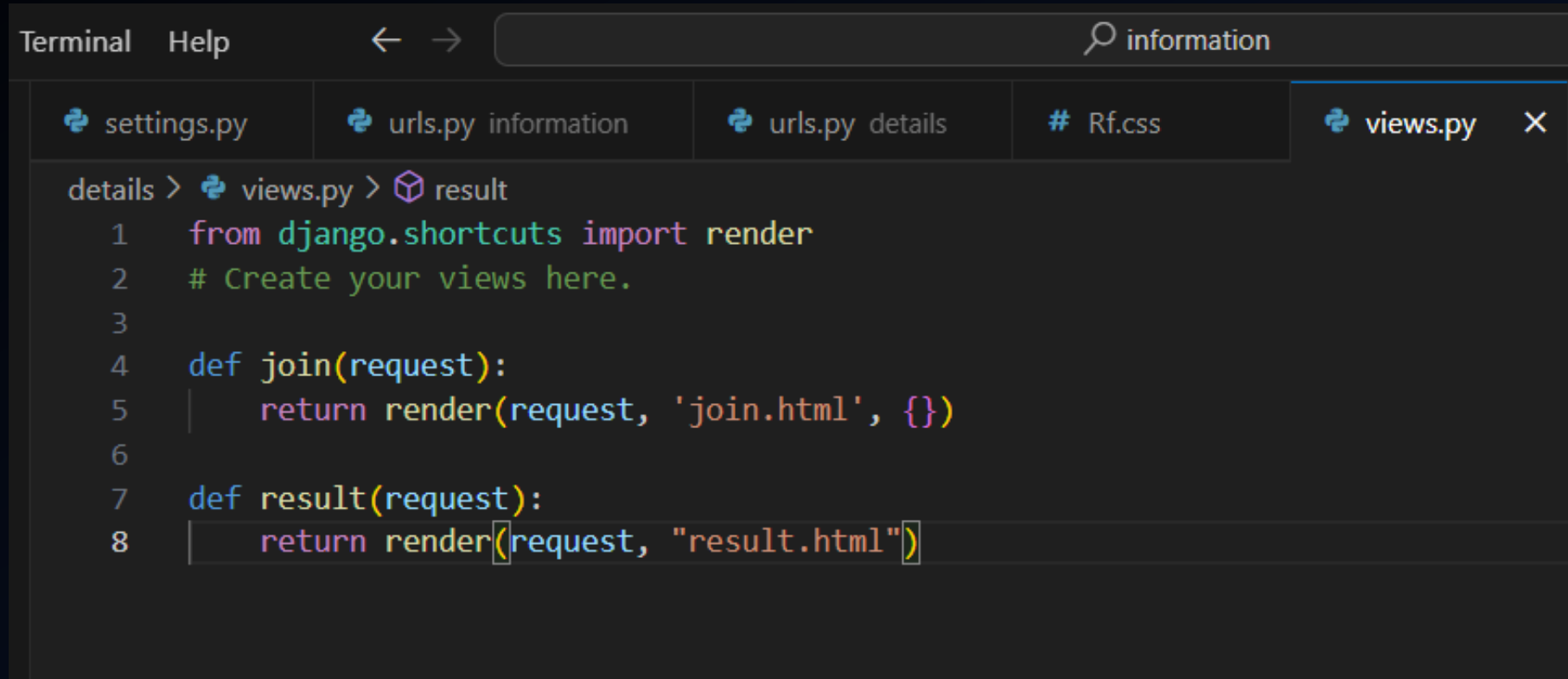
STYLESHEET – Rf.CSS



The image shows a web browser's developer tools interface. The top bar includes 'Terminal' and 'Help' buttons, navigation arrows, and a search bar labeled 'information'. Below this, a tab bar shows several open files: 'settings.py', 'urls.py information', 'urls.py details', and the active tab, '# Rf.css'. The main content area displays the CSS file's path as 'details > static > details > # Rf.css > form'. The CSS code is listed as follows:

```
1 h1,legend{text-align: center;font-size: xx-large;}
2 form{color: darkred; background-color: lightblue;}
```

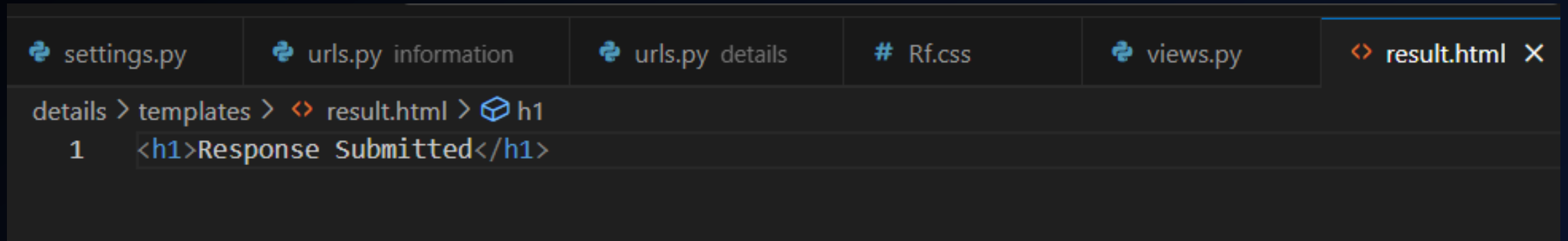
VIEWS.PY



The image shows a code editor window with a dark theme. The title bar at the top contains 'Terminal', 'Help', navigation arrows, and a search bar with the text 'information'. Below the title bar, there are five tabs: 'settings.py', 'urls.py information', 'urls.py details', '# Rf.css', and 'views.py'. The 'views.py' tab is currently selected. The code editor displays the following Python code:

```
details > views.py > result
1  from django.shortcuts import render
2  # Create your views here.
3
4  def join(request):
5      return render(request, 'join.html', {})
6
7  def result(request):
8      return render(request, "result.html")
```

RESULT.HTML



The screenshot shows a web application interface with a dark theme. At the top, there is a navigation bar with several tabs: 'settings.py', 'urls.py information', 'urls.py details', '# Rf.css', 'views.py', and 'result.html'. The 'result.html' tab is currently selected and highlighted. Below the navigation bar, there is a breadcrumb trail: 'details > templates > result.html > h1'. The main content area displays a single line of HTML code: '<h1>Response Submitted</h1>'. The line number '1' is visible on the left side of the code editor.

```
settings.py  urls.py information  urls.py details  # Rf.css  views.py  result.html X
```

details > templates > result.html > h1

```
1  <h1>Response Submitted</h1>
```

JOIN.HTML - 1

details > templates > join.html > html > body > script > validate

```
1  {% load static %}
2
3  <!DOCTYPE html>
4  <html>
5      <head>
6          <title>Registration Form</title>
7          <link rel="stylesheet" href="{% static 'Rf.css' %}">
8
9      </head>
10     <body>
11         <script>
12             function validate(){
13                 a = document.getElementById("naam").value;
14                 b = document.getElementById("phone").value;
15                 if (a === ""){
16                     alert("Name is compulsory");
17                 } else if (b === ""){
18                     alert("phone is compulsory");
19                 };
20             };
21         </script>
22         <h1>Examination Registration Form</h1>
23         <form action="result">
```

JOIN.HTML - 2

```
20      </script>
21      <h1>Examination Registration Form</h1>
22      <form action="result">
23          <fieldset>
24              <legend>Personal Details</legend>
25
26              <label for="naam">Name:</label>
27              <input type="text" id="naam" >
28
29              <br>
30              <br>
31
32              <label for="phone">Phone Number:</label>
33              <input type="number" id="phone" >
34
35              <br>
36              <br>
37
38              <label for="email">Email::</label>
39              <input type="email" id ="email">
40
41              <br>
42              <br>
43
44              <label for="dob">Date Of Birth:</label>
45              <input type="date" id="dob">
46
47              <br>
48              <br>
49
50              <label for="add">Address:</label>
51              <textarea id="add"></textarea>
52
53              <br>
54              <br>
```

JOIN.HTML - 3

```
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
```

```
<br>
<br>

<label for="file">Upload your ID:</label>
<input type="file" id="file" >

</fieldset>

<hr>
<br>

<fieldset>
  <legend>Course Details</legend>

  <p>SELECT COURSE</p>
  <input type="radio" id = "op1" name="course">
  <label for="op1">B.Sc. Computer Science (Hons)</label>

  <br>

  <input type="radio" id = "op2" name="course">
  <label for="op2">B.Sc. Mathematics (Hons)</label>

  <br>

  <input type="radio" id = "op3" name="course">
  <label for="op3">BMS</label>

  <hr>
  <br>

  <p>SELECT SUBJECTS</p>

  <input type="checkbox" id = "sb1" checked>
  <label for="sb1">Mathematics</label>
```

ppt_by_sourabh_pai

JOIN.HTML - 4

```
82
83 <p>SELECT SUBJECTS</p>
84
85 <input type="checkbox" id = "sb1" checked>
86 <label for="sb1">Mathematics</label>
87
88 <br>
89 <input type="checkbox" id = "sb2" checked>
90 <label for="sb2">Python</label>
91
92 <br>
93 <input type="checkbox" id = "sb3" checked>
94 <label for="sb3">Computer System Architechture</label>
95
96 <br>
97 <p>AEC:</p>
98 <input type="radio" id = "sb4a" name="hindi">
99 <label for="sb4a">Hindi - A</label>
100 <input type="radio" id = "sb4b" name="hindi">
101 <label for="sb4b">Hindi - B</label>
102 <input type="radio" id = "sb4c" name="hindi">
103 <label for="sb4c">Hindi - C</label>
104
105 <br>
106 <p>GE:</p>
107 <input type="radio" id = "sb5a" name="GE">
108 <label for="sb5a">COMMERCE</label>
109 <input type="radio" id = "sb5b" name="GE">
110 <label for="sb5b">CALCULUS</label>
111 <input type="radio" id = "sb5c" name="GE">
112 <label for="sb5c">ELECTRONICS</label>
113 ppt_by_sourabh_pal
```


JOIN.HTML - 5

```
109 <input type="radio" id = "sb5b" name="GE" >
110 <label for="sb5b">CALCULUS</label>
111 <input type="radio" id = "sb5c" name="GE">
112 <label for="sb5c">ELECTRONICS</label>
113
114 <br>
115 <p>VAC:</p>
116 <input type="radio" id = "sb6a" name="VAC">
117 <label for="sb6a">VEDIC MATHEMATICS</label>
118 <input type="radio" id = "sb6b" name="VAC">
119 <label for="sb6b">DIGITAL EMPOWERMENT</label>
120 <input type="radio" id = "sb6c" name="VAC">
121 <label for="sb6c">PSYCHOLOGY</label>
122
123 <br>
124 <p>SEC:</p>
125 <input type="radio" id = "sb7a" name="SEC">
126 <label for="sb7a">FRONT-END</label>
127 <input type="radio" id = "sb7b" name="SEC">
128 <label for="sb7b">STATISTICS</label>
129 <input type="radio" id = "sb7c" name="SEC">
130 <label for="sb7c">PERSONAL FINANCE</label>
131
132
133 </fieldset>
134 <BR>
135 <BR>
136
137 <select>
138 <option value="">Select ratings</option>
139 <p>rate your experience</p>
140 <optgroup label="High ratings">
141
142 <option>10 stars</option>
143 <option>8 stars</option>
144
145 </optgroup>ppt_by_sourabh_pal
```

JOIN.HTML - 6

```
133     </fieldset>
134     <BR>
135     <BR>
136
137     <select>
138         <option value="">Select ratings</option>
139         <p>rate your experience</p>
140         <optgroup label="High ratings">
141
142             <option>10 stars</option>
143             <option>8 stars</option>
144
145         </optgroup>
146         <optgroup label="Low ratings">
147
148             <option>5 stars</option>
149             <option>3 stars</option>
150         </optgroup>
151
152
153     </select>
154     <br>
155     <br>
156
157     <button type="submit" onclick="validate()">SUBMIT</button>
158     <button type="reset">RESET</button>
159 </form>
160 </body>
161 </html>
```

BROWSER

Spotify - Web Player: Music for € x

YouTube x

New Tab x

Registration Form x

+

← → ↺

127.0.0.1:8000/details/join/

🔍 📄 ☆ ⚙️ 🖨️ 👤 ⋮

Examination Registration Form

Personal Details

Name:

Phone Number:

Email:

Date Of Birth:

Address:

Upload your ID: No file chosen

Course Details

SELECT COURSE

- ☐ B.Sc. Computer Science (Hons)
- ☐ B.Sc. Mathematics (Hons)
- ☐ BMS

SELECT SUBJECTS

- ☒ Mathematics
- ☒ Python
- ☒ Computer System Architecture

AEC:

- ☐ Hindi - A ☐ Hindi - B ☐ Hindi - C

GE:

- ☐ COMMERCE ☐ CALCULUS ☐ ELECTRONICS

VAC:

- ☐ VEDIC MATHEMATICS ☐ DIGITAL EMPOWERMENT ☐ PSYCHOLOGY

SEC:

- ☐ FRONT-END ☐ STATISTICS ☐ PERSONAL FINANCE

Select ratings ▼

33°C
High winds soon

🖥️ 📁 📞 🌐 📄

ppt_by_sourabh_pal

^

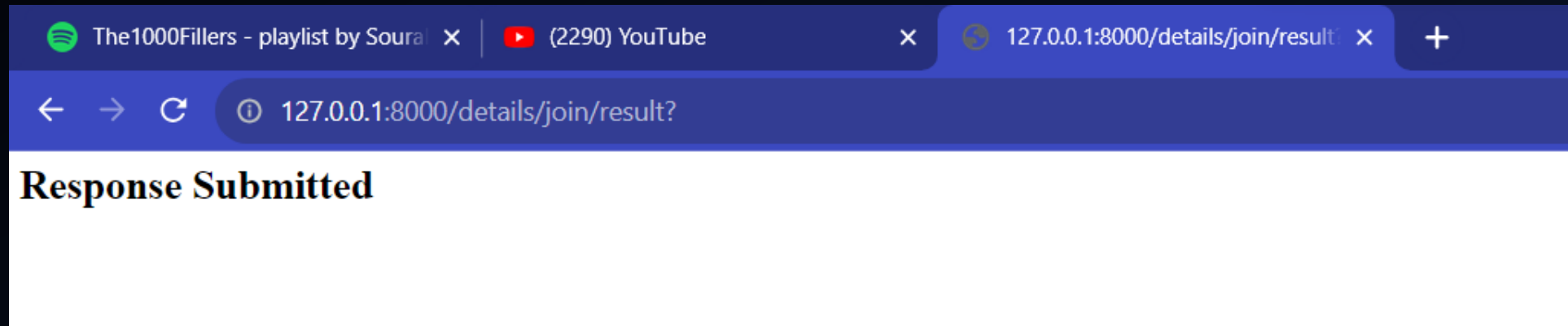
ENG
IN

📶 🔊 🔋

13:29
01-10-2023

1

BROWSER – AFTER SUBMIT



DAY 6

EVALUATION FUNCTIONS

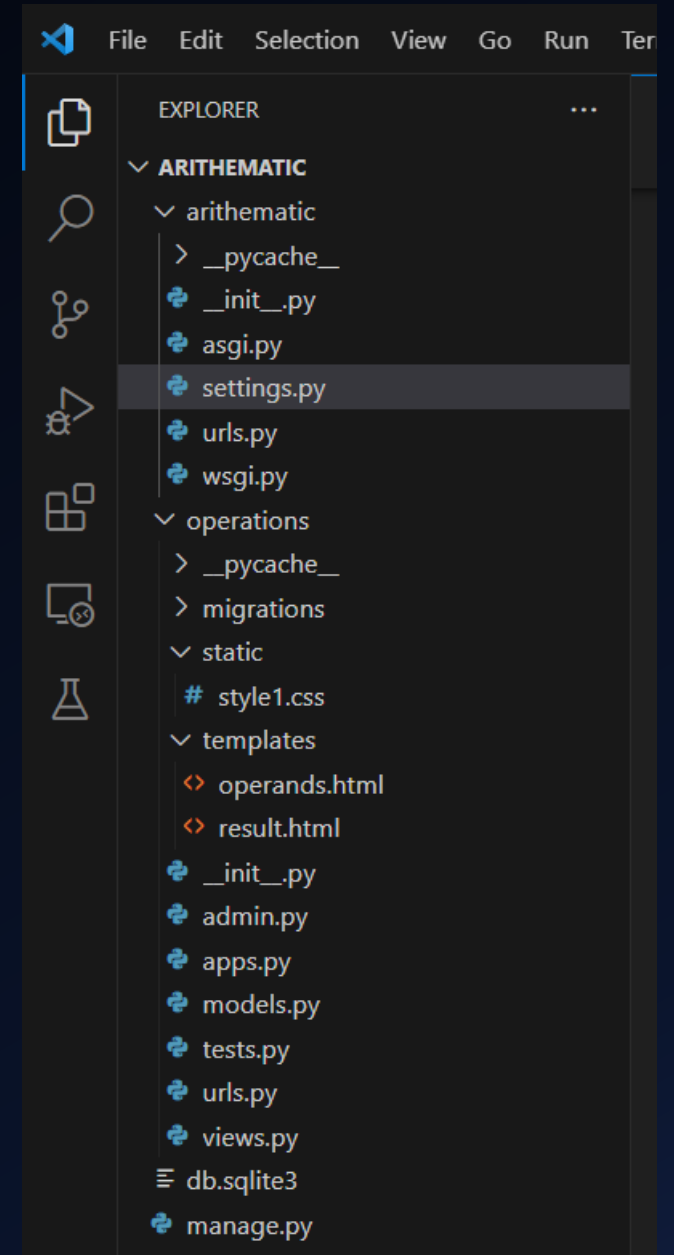
ARITHMETIC OPERATIONS AND RESULT
DISPLAY ON THE CLIENT PAGE

INSTRUCTIONS

- Directory name = Arithmetic
- Project name = arithmetic
- App name = operations
- Html page 1 =operands.html
- Html page 2 = result.html
- Views.py functions = enter(), add()
- Make the following changes

SETTINGS.PY

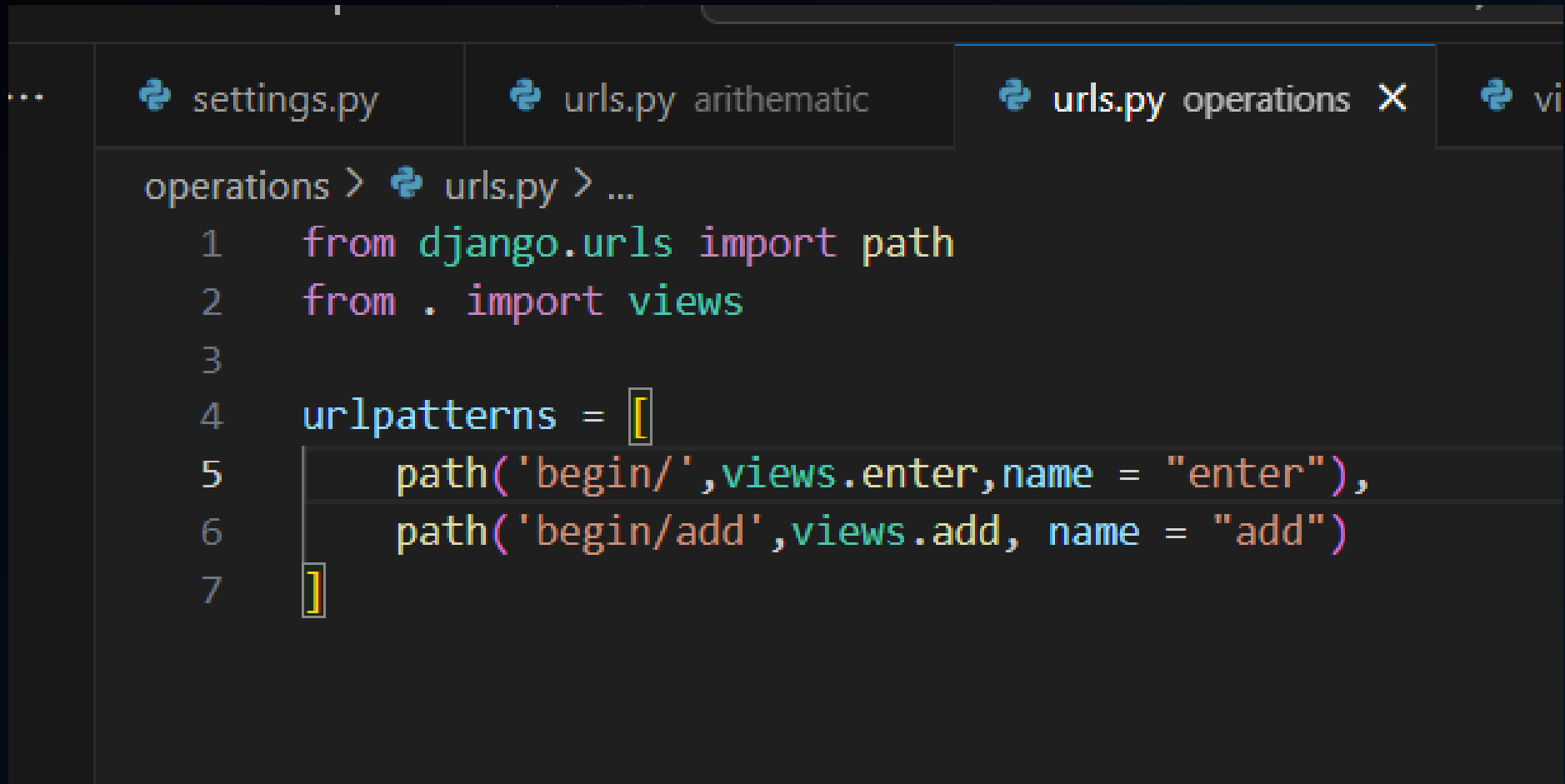
```
25 # SECURITY WARNING: don't run with debug
26 DEBUG = True
27
28 ALLOWED_HOSTS = []
29
30
31 # Application definition
32
33 INSTALLED_APPS = [
34     'django.contrib.admin',
35     'django.contrib.auth',
36     'django.contrib.contenttypes',
37     'django.contrib.sessions',
38     'django.contrib.messages',
39     'django.contrib.staticfiles',
40     'operations'
41 ]
42
43 MIDDLEWARE = [
```



PROJECT – URLS.PY

```
16      """
17      from django.contrib import admin
18      from django.urls import path, include
19
20      urlpatterns = [
21          path('admin/', admin.site.urls),
22          path('operations/', include('operations.urls'))
23      ]
24
```


APP – URLS.PY



The image shows a code editor window with three tabs: 'settings.py', 'urls.py arithmetic', and 'urls.py operations'. The 'urls.py operations' tab is active. The code in the editor is as follows:

```
operations > urls.py > ...  
1  from django.urls import path  
2  from . import views  
3  
4  urlpatterns = [  
5      path('begin/', views.enter, name = "enter"),  
6      path('begin/add', views.add, name = "add")  
7  ]
```

VIEWS.PY

```
..  settings.py  urls.py arithmetic  urls.py operations  views.py X

operations > views.py > add
1  from django.shortcuts import render
2
3  # Create your views here.
4  def enter(request):
5      return render(request, 'Operands.html',{})
6
7  def add(request):
8      val1 = int(request.GET['num1'])
9      val2 = int(request.GET['num2'])
10     res = val1+val2
11     return render(request, "result.html", {"result": res})
```

OPERANDS.HTML

operations > templates > <> operands.html >  html

```
7     </head>
8     <body>
9     <form action="add" method="get">
10         {% csrf_token %}
11         Enter First operand: <input type="number", name="num1"><br/><br/>
12         Enter Second operand: <input type="number", name="num2"><br/>
13         <br/>
14         <input type="submit" name="new">
15         <input type="reset" name="new">
16     </form>
17 </body>
18 </html>
```

STYLE1.CSS

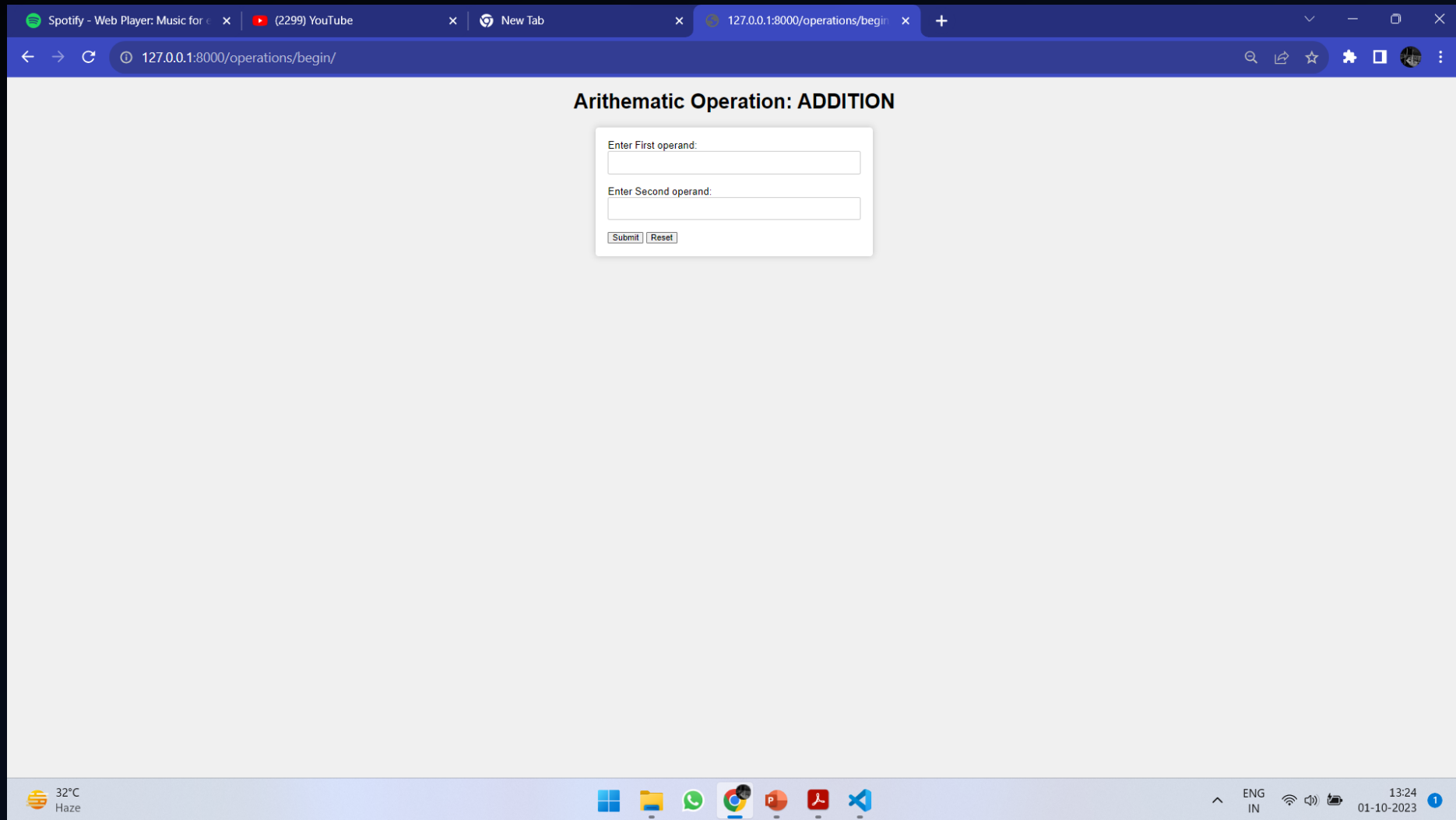
```
settings.py  urls.py arithmetic  urls.py operations  +
operations > static > # style1.css > h1
1  body {
2      font-family: Arial, sans-serif;
3      background-color: #f0f0f0;
4      margin: 0;
5      padding: 0;
6  }
7
8  h1 {
9      text-align: center;
10     margin-top: 20px;
11 }
12
13 form {
14     max-width: 400px;
15     margin: 0 auto;
16     padding: 20px;
17     background-color: #fff;
18     box-shadow: 0 0 10px rgba(0, 0, 0, 0.2);
19     border-radius: 5px;
20 }input[type="submit"]{
21     text-align: center;
22 }
23
24 .input-container {
25     margin-bottom: 100px;
26     text-align: center;
27 }
28
29 input[type="number"] {
30     width: 100%;
31     padding: 10px;
32     border: 1px solid #ccc;
33     border-radius: 3px;
34 }
35
36 ppt_by_sourabh_pal
```

RESULT.HTML

```
operations > templates > <> result.html
```

```
1  Solution: {{result}}
```

BROWSER



BROWSER

New Tab x 127.0.0.1:8000/operations/begin x +

Arithmetic Operation: ADDITION

Enter First operand:

Enter Second operand:

127.0.0.1:8000/operations/begin x Spotify - Web Player: Music for

← → ↻ ⓘ 127.0.0.1:8000/operations/begin/add?csrfmiddle

Solution: 100

DAY 7

Creating a Local Library Model

AUTHORS AND THEIR BOOKS

INSTRUCTIONS

- DIRECTORY NAME: django_project1
- PROJECT NAME: locallibrary
- APP NAME: catalog
- Create environment, project, app, superuser, directories [templates, static (image and stylesheet), urls.py(apps)], HTML page (index.html)
- After creating the models, migrate the contents using the following commands
- Python manage.py makemigrations
- Python manage.py migrate

SETTINGS.PY

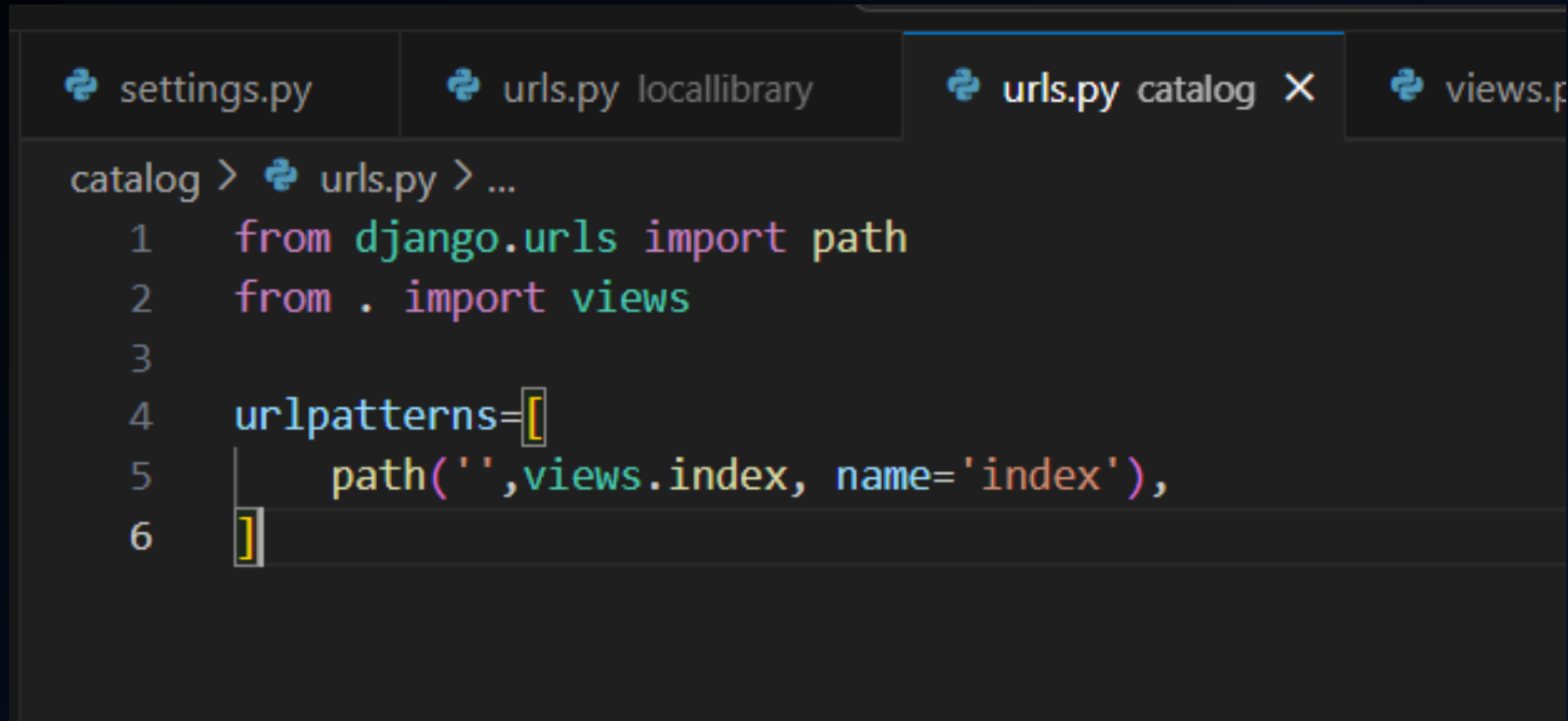
```
settings.py ×  urls.py locallibrary  urls.py catalog  views.py
locallibrary > settings.py > ...
22  # SECURITY WARNING: keep the secret key used in production secret
23  SECRET_KEY = 'django-insecure-kbxy%pqiz#hjr^!f1ad=!kp%^7qnp7wi=
24
25  # SECURITY WARNING: don't run with debug turned on in production
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      'catalog'
41  ]
```

URLS.PY - PROJECT

```
settings.py  urls.py locallibrary X  urls.py catalog  views.py  admin.py

locallibrary > urls.py > ...
11 1. Add an import:  from other_app.views import home
12 2. Add a URL to urlpatterns:  path('', Home.as_view(), name='home')
13 Including another URLconf
14 1. Import the include() function: from django.urls import include
15 2. Add a URL to urlpatterns:  path('blog/', include('blog.urls'))
16 """
17 from django.contrib import admin
18 from django.urls import path, include
19 from django.views.generic import RedirectView
20 urlpatterns = [
21     path('admin/', admin.site.urls),
22     path('catalog/', include('catalog.urls')),
23     path('', RedirectView.as_view(url='catalog/', permanent=True))
24 ]
25
```

URLS.PY - APP



The image shows a code editor with four tabs at the top: 'settings.py', 'urls.py locallibrary', 'urls.py catalog' (which is the active tab), and 'views.p...'. The active tab displays the following Python code:

```
catalog > urls.py > ...
1  from django.urls import path
2  from . import views
3
4  urlpatterns=[
5      path('', views.index, name='index'),
6  ]
```

VIEWS.PY

settings.py

urls.py locallibrary

urls.py catalog

views.py

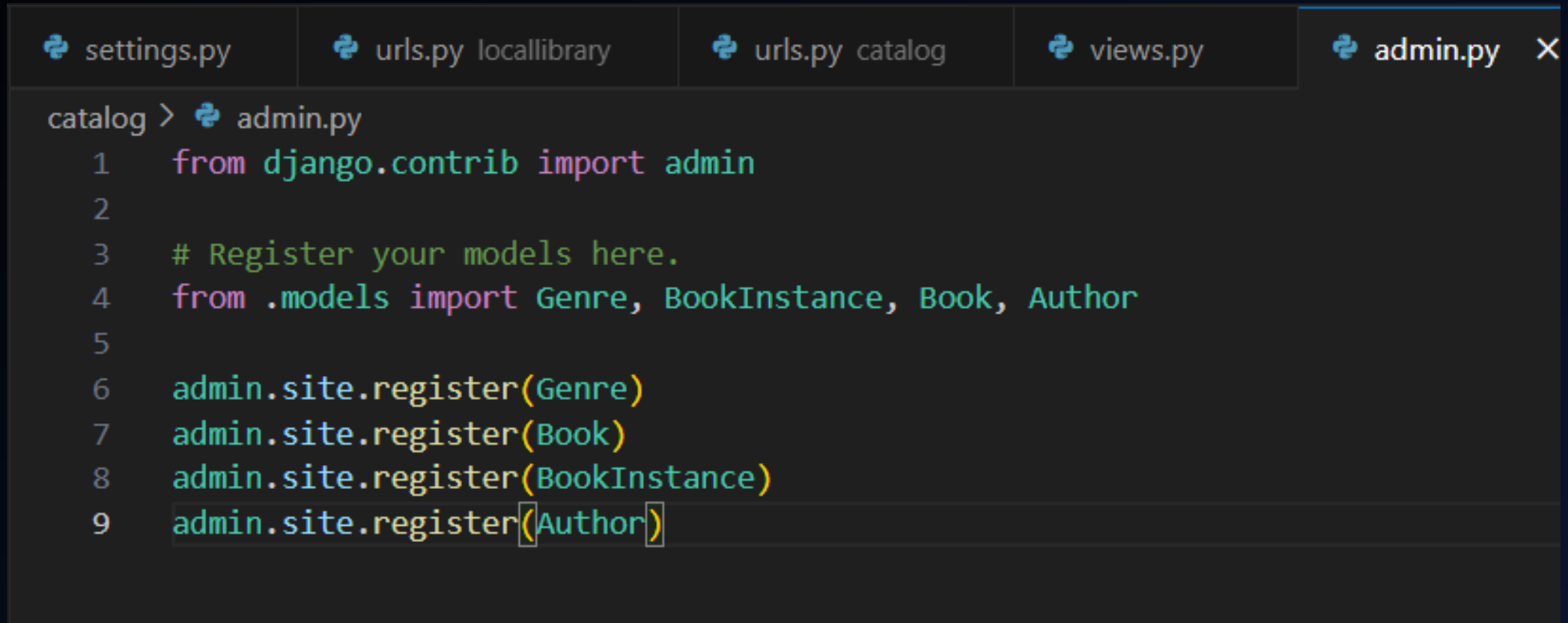
admin.py

model

catalog > views.py > ...

```
1  from django.shortcuts import render
2  from .models import Book, Author, BookInstance, Genre
3
4  def index(request):
5      num_books = Book.objects.all().count()
6      num_instances = BookInstance.objects.all().count()
7      num_instances_available = BookInstance.objects.filter(status__exact='a').count()
8
9      num_authors = Author.objects.count()
10     context = {
11         'num_books': num_books,
12         'num_instances': num_instances,
13         'num_instances_available': num_instances_available,
14         'num_authors': num_authors,
15     }
16
17     return render(request, 'index.html', context=context)
18
```

ADMIN.PY



The image shows a code editor window with five tabs at the top: settings.py, urls.py locallibrary, urls.py catalog, views.py, and admin.py. The admin.py tab is active, showing the following Python code:

```
catalog > admin.py
1  from django.contrib import admin
2
3  # Register your models here.
4  from .models import Genre, BookInstance, Book, Author
5
6  admin.site.register(Genre)
7  admin.site.register(Book)
8  admin.site.register(BookInstance)
9  admin.site.register(Author)
```

MODELS.PY

```
catalog > models.py > BookInstance
1  from django.db import models
2
3  class Genre(models.Model):
4      name = models.CharField(max_length=200, help_text='Enter a book genre (e.g. Science Fiction)')
5
6      def __str__(self):
7          return self.name
8
9  from django.urls import reverse
10 class Book(models.Model):
11     title = models.CharField(max_length=200)
12     author = models.ForeignKey('Author', on_delete=models.SET_NULL, null=True)
13     summary = models.TextField(max_length=1000, help_text='Enter a brief description of the book')
14     isbn = models.CharField('ISBN', max_length=13, unique=True, help_text='13 Character <a href="https://www.isbn-international.org/content/what-isbn">ISBN')
15     genre = models.ManyToManyField(Genre, help_text='Select a genre for this book')
16
17
18     def __str__(self):
19         return self.title
20
21     def get_absolute_url(self):
22         return reverse('book-detail', args=[str(self.id)])
23
```

MODELS.PY

```
23
24 import uuid
25 class BookInstance(models.Model):
26     id = models.UUIDField(primary_key=True, default=uuid.uuid4, help_text='Unique ID for this particular book across whole library')
27     book = models.ForeignKey('Book', on_delete=models.RESTRICT, null=True)
28     imprint = models.CharField(max_length=200)
29     due_back = models.DateField(null=True, blank=True)
30     LOAN_STATUS = (
31         ('m', 'Maintenance'),
32         ('o', 'On loan'),
33         ('a', 'Available'),
34         ('r', 'Reserved'),
35     )
36     status = models.CharField(
37         max_length=1,
38         choices=LOAN_STATUS,
39         blank=True,
40         default='m',
41         help_text='Book availability',
42     )
43
44     class Meta:
45         ordering = ['due_back']
46         def __str__(self):
47             return f'{self.id} ({self.book.title})'
48
```


MODELS.PY

```
48
49 class Author(models.Model):
50     first_name = models.CharField(max_length=100)
51     last_name = models.CharField(max_length=100)
52     date_of_birth = models.DateField(null=True, blank=True)
53     date_of_death = models.DateField('Died', null=True, blank=True)
54
55     class Meta:
56         ordering = ['last_name', 'first_name']
57         def get_absolute_url(self):
58             return reverse('author-detail', args=[str(self.id)])
59         def __str__(self):
60             return f'{self.last_name}, {self.first_name}'
```

INDEX.HTML

```
settings.py  urls.py locallibrary  urls.py catalog  views.py  admin.py  models.py  index.html X
```

catalog > templates > index.html > img

```
1  <!DOCTYPE html>
2  <html lang="en">
3      <head>
4          <h1>Local Library Home</h1>
5          <p>Welcome to Local Library</p>
6          <h2>Dynamic content</h2>
7          <p>The library has the following record counts:</p>
8          <ul>
9              <li><strong>Books:</strong> {{ num_books }}</li>
10             <li><strong>Copies:</strong> {{ num_instances }}</li>
11             <li><strong>Copies available:</strong> {{ num_instances_available }}</li>
12             <li><strong>Authors:</strong> {{ num_authors }}</li>
13         </ul>
14     </head>
15     <body>
16         <div class="container-fluid">
17             <div class="row">
18                 <div class="col-sm-2">
19                     {% block sidebar %}
20                     <ul class="sidebar-nav">
21                         <li><a href="{% url 'index' %}">Home</a></li>
22                         <li><a href="">All books</a></li>
23                         <li><a href="">All authors</a></li>
24                     </ul>
25                     {% endblock %}
26                 </div>
27                 <div class="col-sm-10">{% block content %}{% endblock %}</div>
28             </div>
29         </div>
30     </body>
31 </html>
32
33 {% load static %}
34 <link rel="stylesheet" href="{% static 'styles.css' %}" />
35 
```

STYLES.CSS

settings.py

urls.py locallibrary

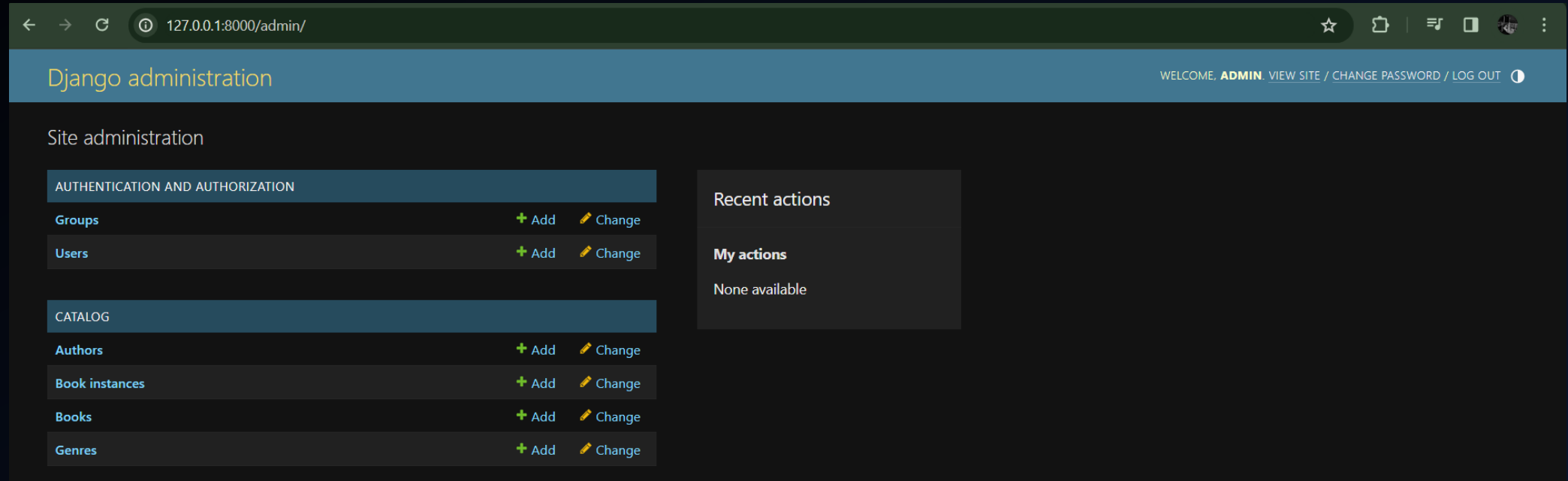
urls.py catalog

views.p

catalog > static > # styles.css > h1

```
1
2  .sidebar-nav {
3      margin-top: 20px;
4      padding: 0;
5      list-style: none;
6  }
7
8  h1,h2 {
9      color: green;
10 }
```

ADMIN PAGE (CLIENT)



ADMIN PAGE (CLIENT)

Action: 0 of 3 selected

<input type="checkbox"/>	AUTHOR
<input type="checkbox"/>	Duffer (Brothers)
<input type="checkbox"/>	J.K. (Rowling)
<input type="checkbox"/>	George R.R. (Martin)

3 authors

Select genre to change

Action: 0 of 4 selected

<input type="checkbox"/>	GENRE
<input type="checkbox"/>	Thriller
<input type="checkbox"/>	Horror
<input type="checkbox"/>	Sci-Fi
<input type="checkbox"/>	Fantasy

4 genres

ADMIN PAGE (CLIENT)





Add book

Title:

The Song of Ice and Fire

Author:

George R.R. (Martin)

Summary:

A Song of Ice and Fire is a series of epic fantasy novels by the American novelist and screenwriter George R. R. Martin. He began writing the first volume, A Game of Thrones, in 1991, publishing it in 1996.

Enter a brief description of the book

ISBN:

589635986235

13 Character ISBN number


Genre:

Fantasy

Sci-Fi

Horror

Thriller



Select a genre for this book Hold down "Control", or "Command" on a Mac, to select more than one.

SAVE

Save and add another

Save and continue editing

Select book to change

Action:

Go

 0 of 2 selected

☐

BOOK

☐

Harry Potter Deathly Hallows

☐


The Song of Ice and Fire

2 books

ADMIN PAGE (CLIENT)

Select book instance to change

Action:

Go

0 of 1 selected



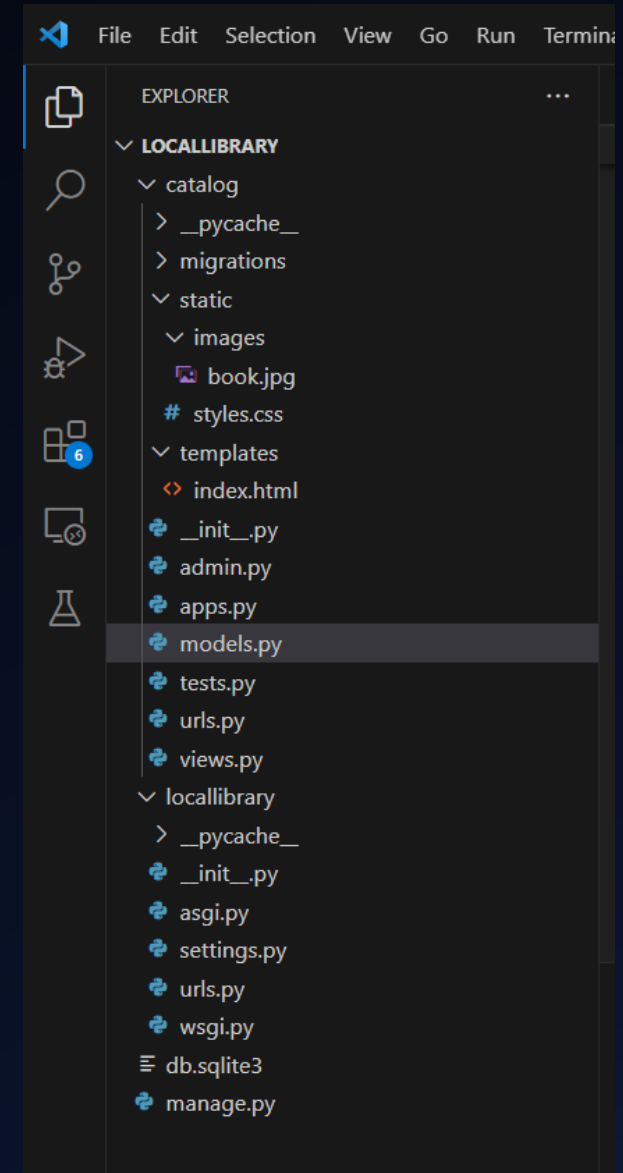
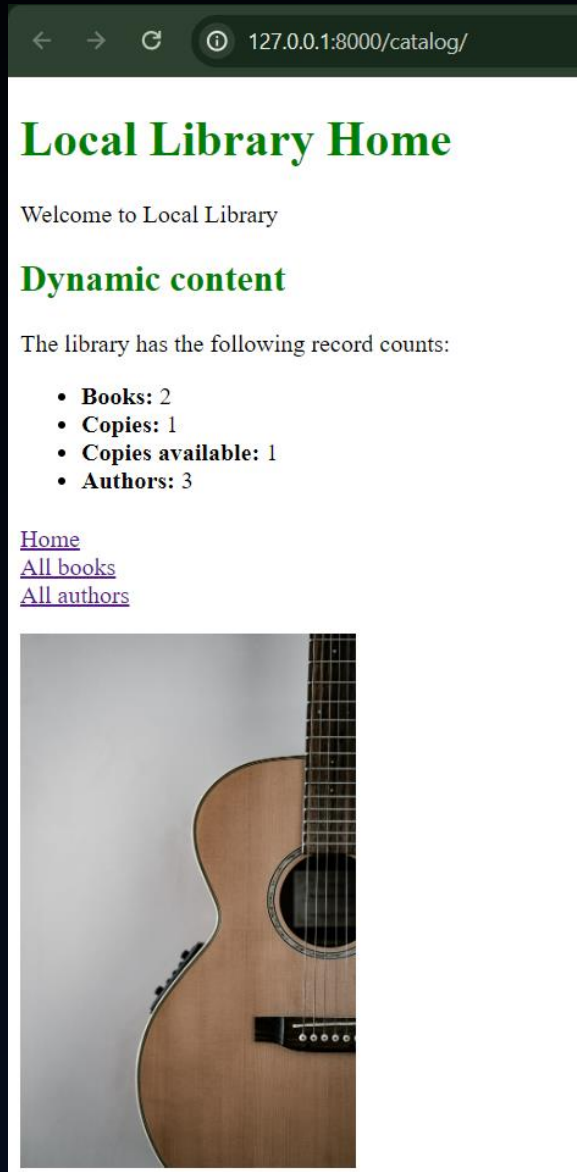
BOOK INSTANCE



BookInstance object (d9c41708-e8b9-43a4-8052-4833e5a578a6)

1 book instance

/CATALOG/



DAY 8 – TASK 1

DECORATORS

USING FUNCTIONS - TIME

DECORATORS

understanding_decorators.py X

C: > My Data (Sourabh) > STUDIES > MY STUDY DOCX > BSC COMP. SCI. (H) (Sem - 3) (C

```
1  import time
2  def tictoc(func):
3      def wrapper():
4          t1 = time.time()
5          func()
6          t2 = time.time()-1
7          print(f'{func.__name__} ran in \f {t2} seconds')
8      return wrapper
9
10 @tictoc
11 def do_this():
12     time.sleep(1.3)
13
14 @tictoc
15 def do_that():
16     time.sleep(.4)
17
18 do_this()
19 do_that()
20 print('Done')
21
```

DAY 8 – TASK 2

INTRODUCING TO JSON

JAVA SCRIPT OBJECT NOTATION

DECORATORS

{} user.json X

C: > My Data (Sourabh) > STUDIES > MY STUDY DOCX > BSC COMP. SCI. (H) (Sem - 3) (2023-24) > SEC - BACKEND WEB DEVELOPMENT > PROJECTS > jsc

```
1  {
2      "name" : "Sourabh",
3      "number" : 435,
4      "passed" : true,
5      "strength" : ["c++", "photoshop"]
6  }
```

🔗 json_test.py X

C: > My Data (Sourabh) > STUDIES > MY STUDY DOCX > BSC COMP. SCI. (H) (Sem - 3) (2023-24) > SEC - BACKEND WEB DEVELOPMENT > PROJECTS > 🔗 jsc

```
1  import json
2  with open("user.json","r") as f:
3      data = json.load(f)
4
5  print("The Data is: ",data)
6  print(["The different items are:\n",data.items()])
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

Pyt

TUDIES\MY STUDY DOCX\BSC COMP. SCI. (H) (Sem - 3) (2023-24)\SEC - BACKEND WEB DEVELOPMENT\PROJECTS\json_test.py'

The Data is: {'name': 'Sourabh', 'number': 435, 'passed': True, 'strength': ['c++', 'photoshop']}

The different items are:

dict_items([('name', 'Sourabh'), ('number', 435), ('passed', True), ('strength', ['c++', 'photoshop'])])

PS C:\My Data (Sourabh)\STUDIES\MY STUDY DOCX\BSC COMP. SCI. (H) (Sem - 3) (2023-24)\SEC - BACKEND WEB DEVELOPMENT\PROJECTS>

DAY 8 – TASK 3

API using Django rest Framework

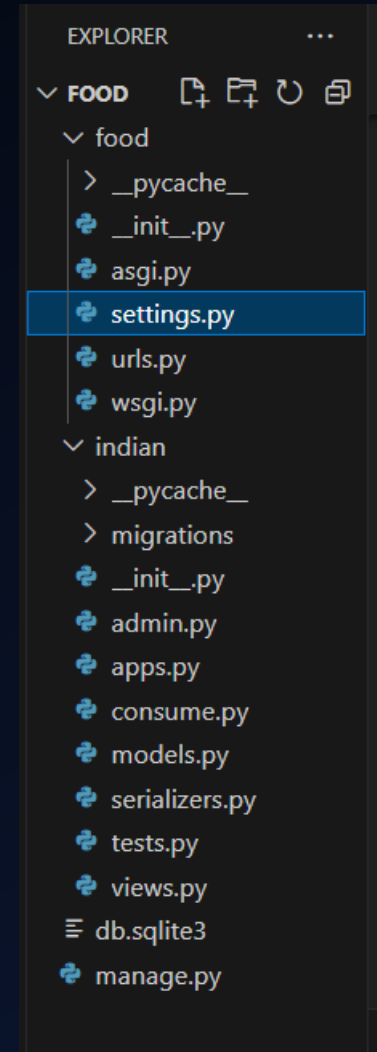
POSTMAN TOOL

INSTRUCTIONS

- DIRECTORY NAME: foods
- PROJECT NAME: food
- APP NAME: indian

SETTINGS.PY

```
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     'indian',
41     'rest_framework'
42 ]
43
```



URLS.PY - PROJECT

```
13 Including another URLconf
14     1. Import the include() function: from djan
15     2. Add a URL to urlpatterns: path('blog/',
16     """
17 from django.contrib import admin
18 from django.urls import path
19 from indian import views
20
21 urlpatterns = [
22     path('admin/', admin.site.urls),
23     path('drinks/', views.drink_list)
24 ]
25
```

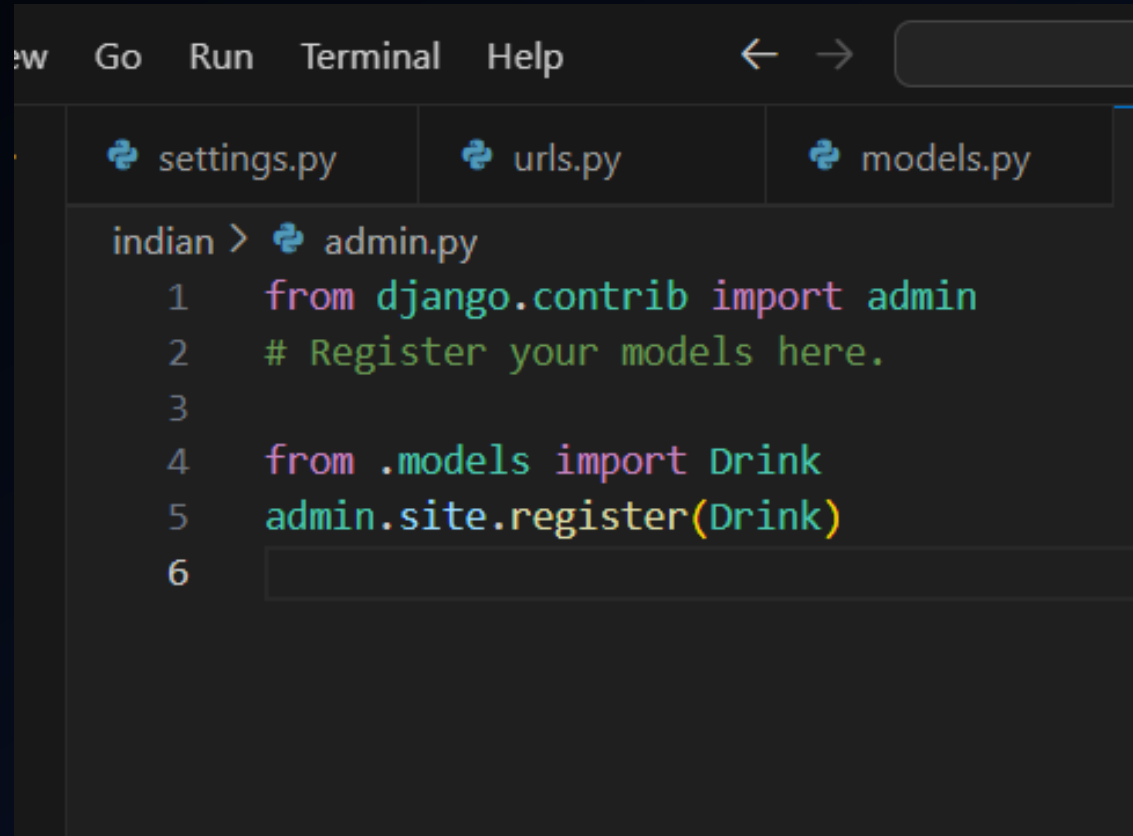

MODELS.PY

```
ew  Go  Run  Terminal  Help  ←  →  
```

```
•  settings.py  urls.py  models.py ×  admin.py
```

```
indian > models.py > ...
1  from django.db import models
2  # Create your models here.
3
4  class Drink(models.Model):
5      name=models.CharField(max_length=200)
6      description=models.CharField(max_length=500)
7
8      def __str__(self):
9          return self.name+" "+self.description
10
11
```

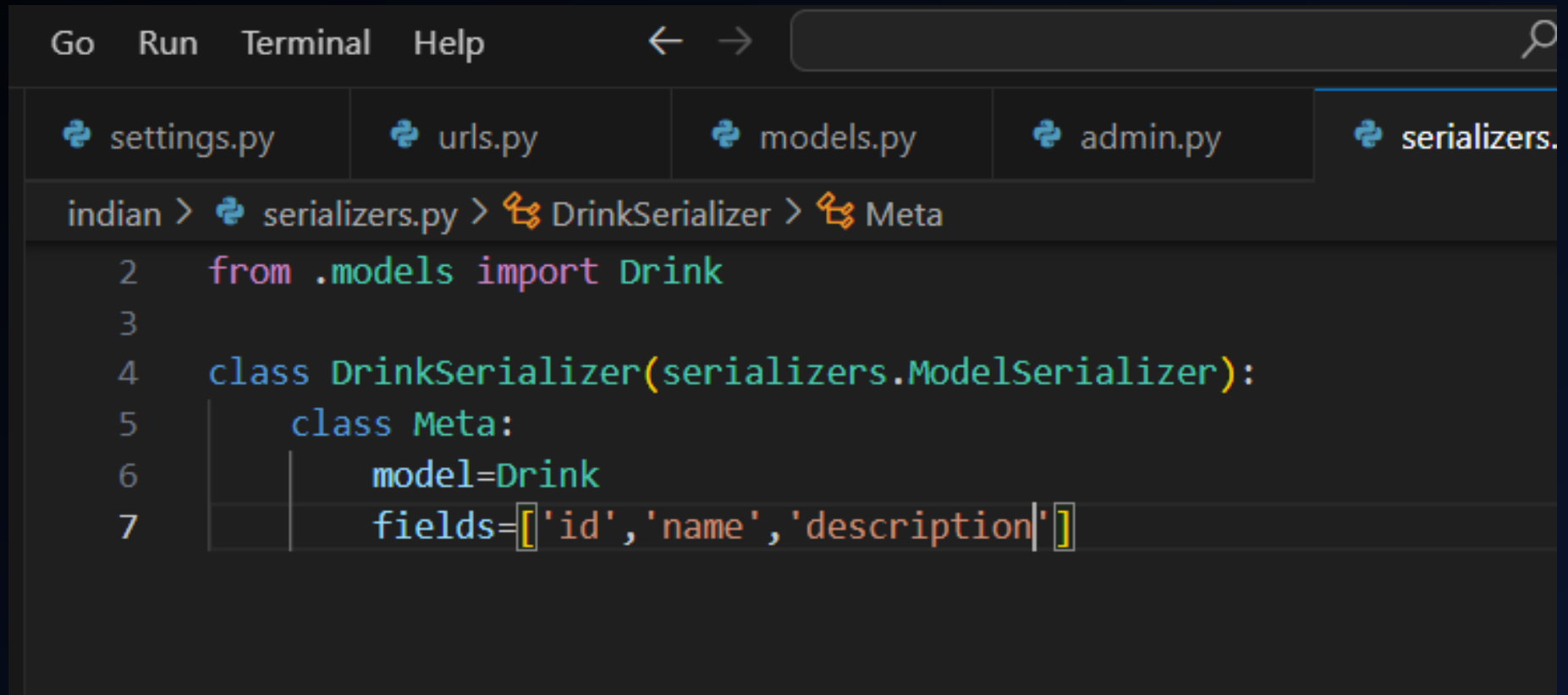
ADMIN.PY



The image shows a code editor window with a dark theme. The menu bar at the top includes 'File', 'Edit', 'View', 'Go', 'Run', 'Terminal', and 'Help'. Below the menu bar, there are three tabs: 'settings.py', 'urls.py', and 'models.py'. The 'admin.py' file is open, and the code is as follows:

```
indian > admin.py
1  from django.contrib import admin
2  # Register your models here.
3
4  from .models import Drink
5  admin.site.register(Drink)
6
```

SERIALIZERS.PY



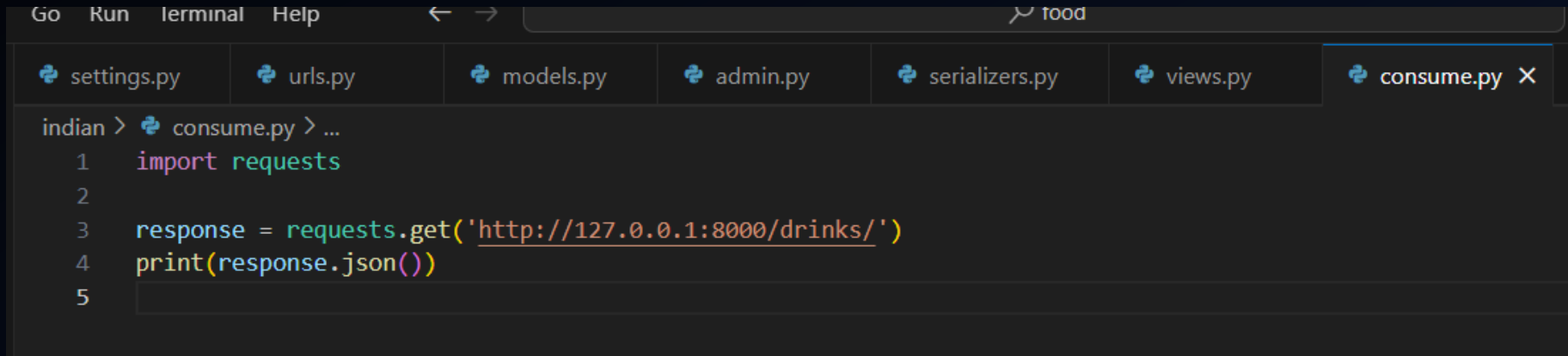
```
Go Run Terminal Help  
settings.py urls.py models.py admin.py serializers.py  
indian > serializers.py > DrinkSerializer > Meta  
2 from .models import Drink  
3  
4 class DrinkSerializer(serializers.ModelSerializer):  
5     class Meta:  
6         model=Drink  
7         fields=['id', 'name', 'description']
```

VIEWS.PY

```
settings.py  urls.py  models.py  admin.py  serializers.py  views.py

indian > views.py > drink_list
1  from django.shortcuts import render
2  from django.http import JsonResponse
3  from .models import Drink
4  from .serializers import DrinkSerializer
5
6  from rest_framework.decorators import api_view
7  from rest_framework.response import Response
8  from rest_framework import status
9
10 # Create your views here.
11
12 @api_view(['GET', 'POST'])
13 |
14 def drink_list(request):
15     if request.method == 'GET':
16         drinks=Drink.objects.all()
17         serializer=DrinkSerializer(drinks, many=True)
18         return JsonResponse({'drinks':serializer.data})
19     if request.method == 'POST':
20         serializer=DrinkSerializer(data=request.data)
21         if serializer.is_valid():
22             serializer.save()
23             return Response(serializer.data, status=status.HTTP_201_CREATED)
24
25
```

CONSUME.PY



The screenshot shows a PyCharm IDE window with a dark theme. The top menu bar includes 'Go', 'Run', 'terminal', and 'Help'. A search bar on the right contains the text 'food'. Below the menu bar, there is a tab bar with several open files: 'settings.py', 'urls.py', 'models.py', 'admin.py', 'serializers.py', 'views.py', and 'consume.py' (which is the active file, indicated by a blue underline). The main editor area displays the code for 'consume.py' with line numbers 1 through 5. The code is as follows:

```
indian > consume.py > ...  
1  import requests  
2  
3  response = requests.get('http://127.0.0.1:8000/drinks/')  
4  print(response.json())  
5
```

ADMIN PANEL

Django administration

Home > Indian > Drinks

Start typing to filter...

AUTHENTICATION AND AUTHORIZATION

Groups + Add

Users + Add

INDIAN

Drinks + Add

«

✓ The drink "Lemon Ice Tea Cold beverage" was added successfully.

Select drink to change

Action:

 Go 0 of 3 selected

☐ DRINK

☐ Lemon Ice Tea Cold beverage

☐ Virgin Mohito pudina

☐ Margarita non alcoholic drink

3 drinks

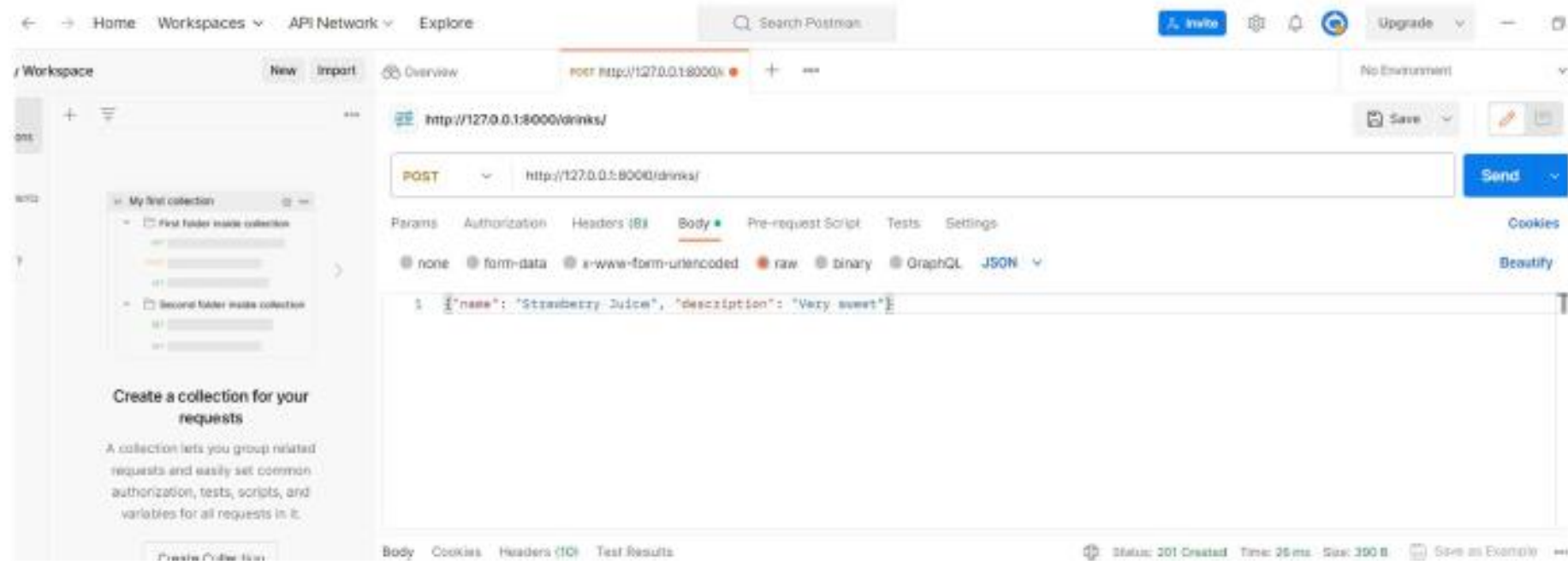
CLIENT PAGE



```
1 {  
2   "drinks": [  
3     {  
4       "id": 1,  
5       "name": "Margarita",  
6       "description": "non alcoholic drink"  
7     },  
8     {  
9       "id": 2,  
10      "name": "Virgin Mohito",  
11      "description": "pudina"  
12    },  
13    {  
14      "id": 3,  
15      "name": "Lemon Ice Tea",  
16      "description": "Cold beverage"  
17    }  
18  ]  
19 }
```

POSTMAN

Adding data using postman tool



POSTMAN (DATA ADDED TO THE DB)

refresh

The screenshot displays a web browser window with two tabs. The top tab shows a REST client response for a GET request to `127.0.0.1:8000/drinks/`. The response is a JSON array of three drink objects.

```
{
  "drinks": [
    {
      "id": 1,
      "name": "Roshufzaa",
      "description": "Very refreshing"
    },
    {
      "id": 2,
      "name": "Orange Juice",
      "description": "Very Orangy"
    },
    {
      "id": 3,
      "name": "Strawberry Juice",
      "description": "Very sweet"
    }
  ]
}
```

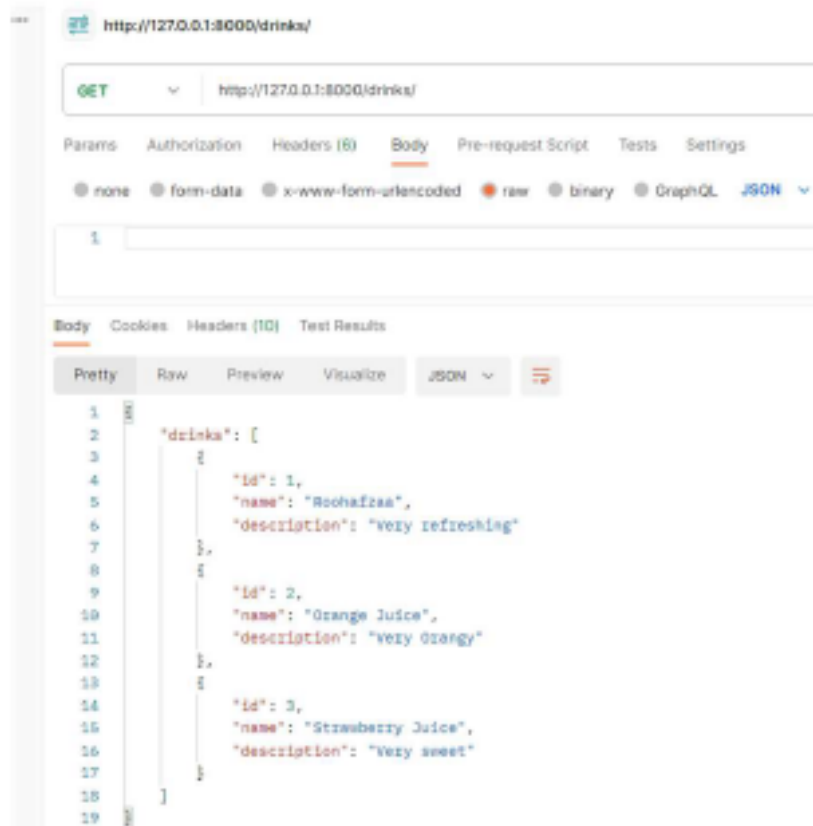
The bottom tab shows the Django administration interface at `127.0.0.1:8000/admin/indian/drink/`. The interface includes a sidebar with a search bar and a list of models under the 'INDIAN' app: 'Groups', 'Users', and 'Drinks'. The 'Drinks' model is selected, and the main content area shows a table of drinks with checkboxes for selection.

Select drink to change	
<input type="checkbox"/>	DRINK
<input type="checkbox"/>	Strawberry Juice Very sweet
<input type="checkbox"/>	Orange Juice Very Orangy
<input type="checkbox"/>	Roshufzaa Very refreshing

At the bottom of the table, it indicates '3 drinks'.

POSTMAN

To check what changes has been done can also use GET in postman



DAY 9

Restful API

GET AND POST METHODS (POSTMAN)

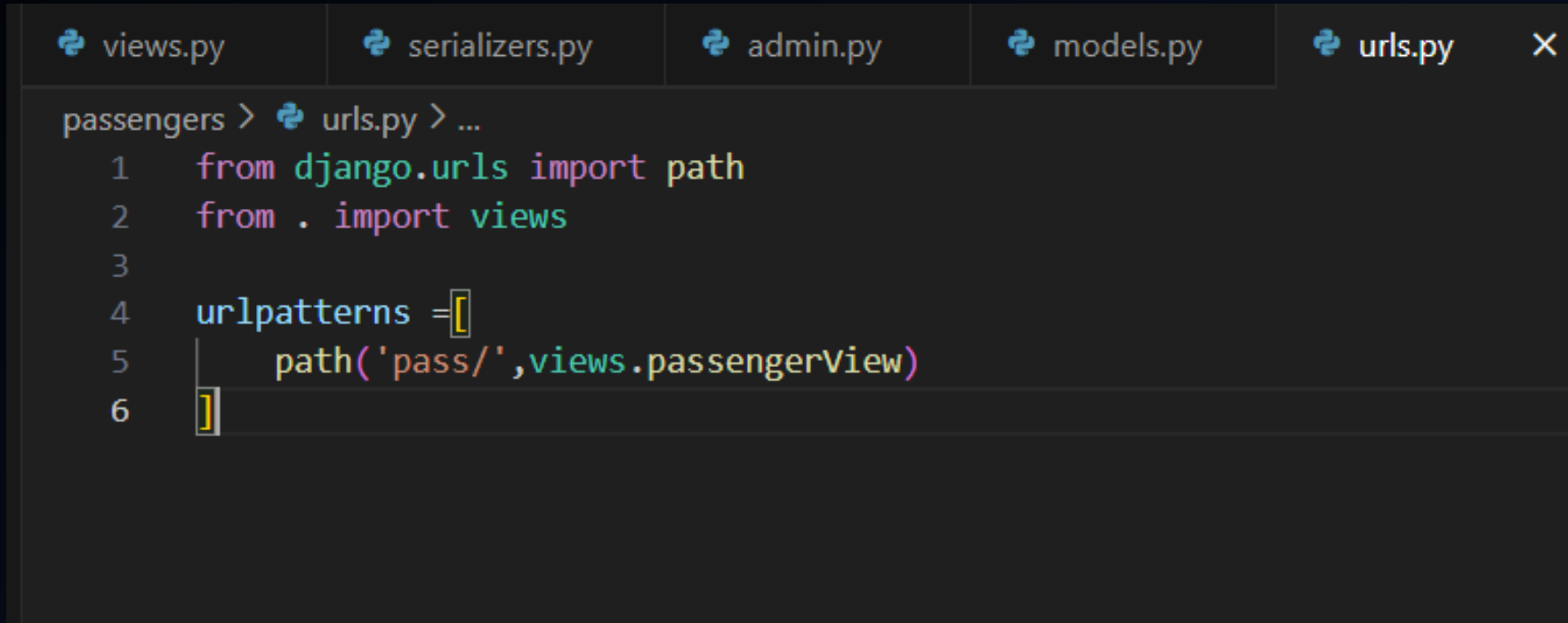
INSTRUCTIONS

- DIRECTORY NAME: project
- PROJECT NAME: project
- APP NAME: passengers
- Pip install requests
- Pip install djangorestframework
- Python manage.py migrate, makemigrations, createsuperuser
- Create files – urls.py (app), serializers.py

URLS.PY - PROJECT

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

URLS.PY - APP



The image shows a code editor window with five tabs at the top: views.py, serializers.py, admin.py, models.py, and urls.py. The urls.py tab is active, showing the following Python code:

```
passengers > urls.py > ...  
1  from django.urls import path  
2  from . import views  
3  
4  urlpatterns = [  
5      path('pass/', views.passengerView)  
6  ]
```

VIEWS.PY

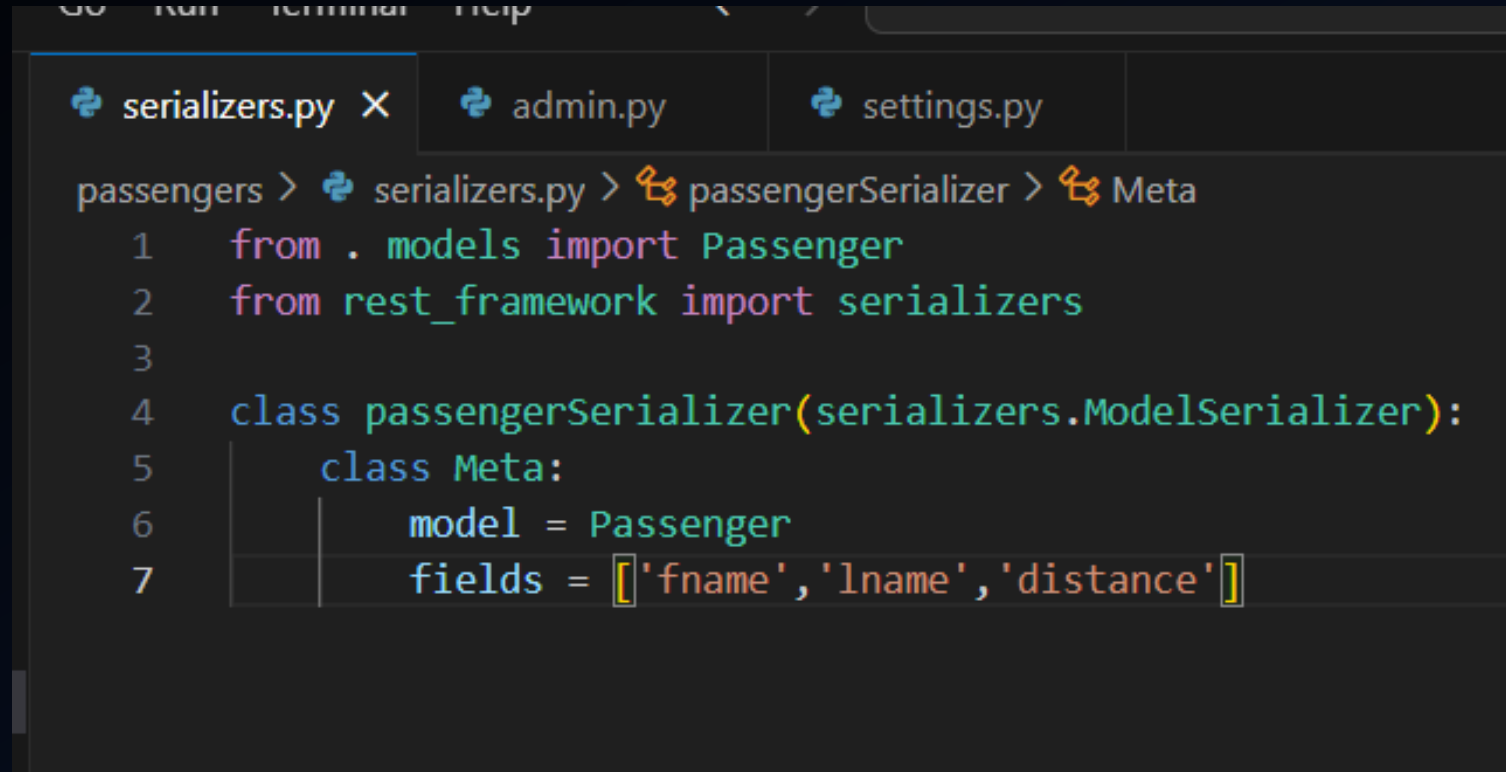
```
views.py X serializers.py admin.py models.py settings.py
passengers > views.py > passengerView
1  from django.shortcuts import render
2  from .serializers import passengerSerializer
3  from rest_framework import status
4  from rest_framework.response import Response
5  from .models import Passenger
6  from rest_framework.decorators import api_view
7
8  # Create your views here.
9  @api_view(['GET', 'POST'])
10 def passengerView(request):
11     if request.method == 'GET':
12         passengers = Passenger.objects.all()
13         serializer = passengerSerializer(passengers, many=True)
14         return Response(serializer.data)
15
16     elif request.method == 'POST':
17         serializer = passengerSerializer(data=request.data)
18         if serializer.is_valid():
19             serializer.save()
20             return Response(serializer.data, status=status.HTTP_201_CREATED)
21     return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)
```

MODELS.PY

```
serializers.py  admin.py  models.py X  settings.py

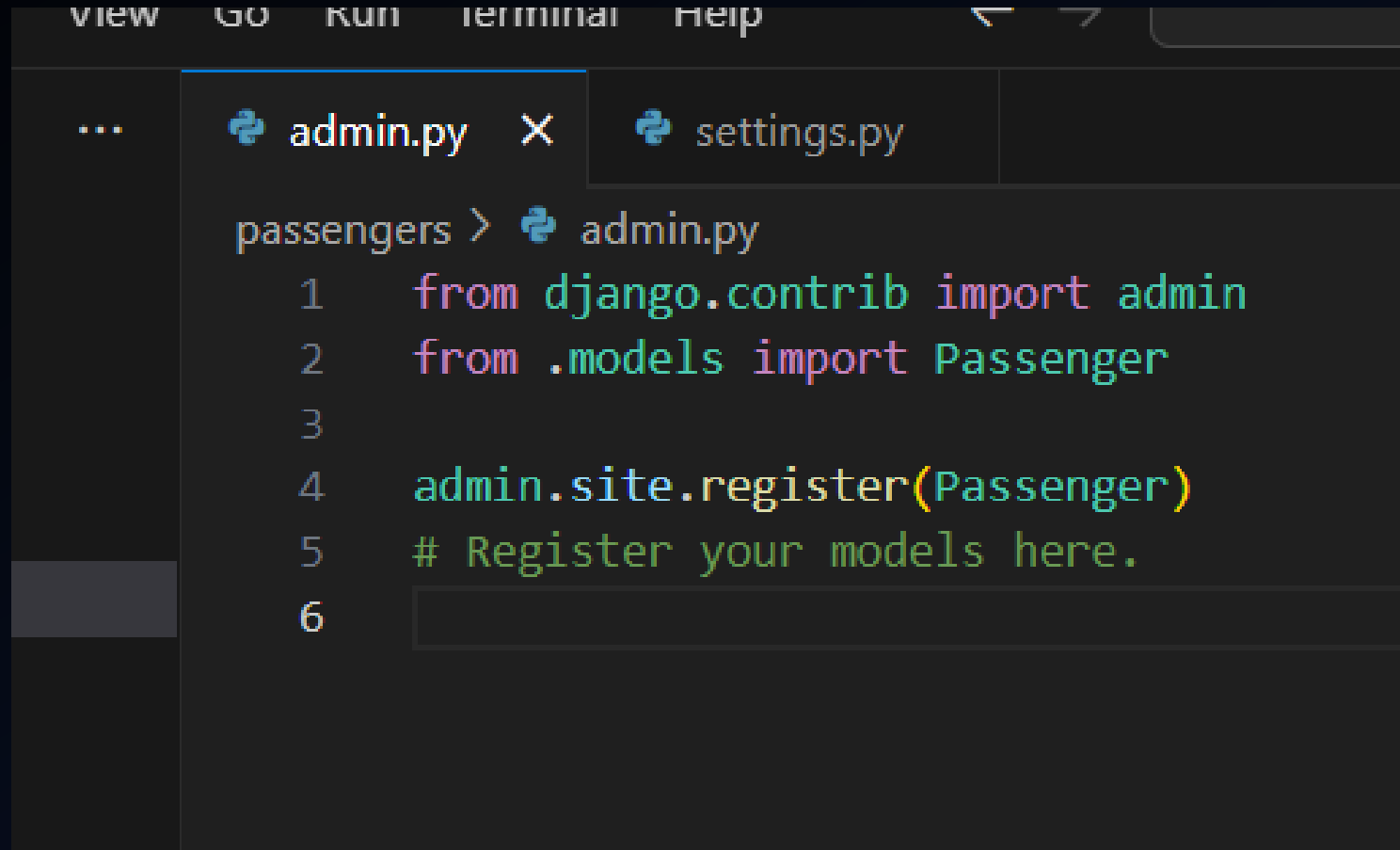
passengers > models.py > Passenger > __str__
1  from django.db import models
2
3  # Create your models here.
4  class Passenger(models.Model):
5      fname = models.CharField(max_length=20,primary_key=True)
6      lname = models.CharField(max_length=20)
7      distance = models.FloatField()
8
9      def __str__(self):
10         return self.fname + " " + self.lname
```


SERIALIZERS.PY



```
passengers > serializers.py > passengerSerializer > Meta
1  from . models import Passenger
2  from rest_framework import serializers
3
4  class passengerSerializer(serializers.ModelSerializer):
5      class Meta:
6          model = Passenger
7          fields = ['fname', 'lname', 'distance']
```

ADMIN.PY



The image shows a code editor window with a dark theme. The menu bar at the top includes 'View', 'Go', 'Run', 'Terminal', and 'Help'. Below the menu bar, there are two tabs: 'admin.py' (active) and 'settings.py'. The 'admin.py' tab contains the following Python code:

```
passengers > admin.py
1  from django.contrib import admin
2  from .models import Passenger
3
4  admin.site.register(Passenger)
5  # Register your models here.
6
```

SETTINGS.PY

```
settings.py X
project > settings.py > ...
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     'passengers',
41     'rest_framework'
42 ]
43
```

ADMIN PAGE

Django administration

Home > Passengers > Passengers

Start typing to filter...

AUTHENTICATION AND AUTHORIZATION

Groups + Add

Users + Add

PASSENGERS

Passengers + Add

✓ The passenger "Sushmita D'Suza" was added successfully.

Select passenger to change

Action: ----- Go 0 of 3 selected

☐ PASSENGER

☐ Vishal Kushwa

☐ Sushmita D'Suza

☐ Sourabh Pal

3 passengers

CLIENT PAGE (API) - GET

Django REST framework

admin

Passenger

Passenger

OPTIONS GET

GET /passengers/pass/?format=api

HTTP 200 OK
Allow: GET, OPTIONS, POST
Content-Type: application/json
Vary: Accept

```
[
  {
    "fname": "Sourabh",
    "lname": "Pal",
    "distance": 27.0
  },
  {
    "fname": "Vishal",
    "lname": "Kushwa",
    "distance": 35.0
  },
  {
    "fname": "Sushmita",
    "lname": "D'Suza",
    "distance": 55.0
  }
]
```

Media type:

application/json

Content:

POST

CLIENT PAGE (API) - POST

```
"lname": "D'Suza",  
"distance": 55.0  
}
```

Media type:

Content:

```
{  
  "fname": "Radha",  
  "lname": "Shyam",  
  "distance": 100.0  
}
```

POST

Passenger

Passenger

OPTIONS GET

POST /passengers/pass/?format=api

HTTP 201 Created
Allow: GET, POST, OPTIONS
Content-Type: application/json
Vary: Accept

```
{  
  "fname": "Radha",  
  "lname": "Shyam",  
  "distance": 100.0  
}
```

CLIENT PAGE (API) - POST

Passenger

Passenger

OPTIONS

GET



GET /passengers/pass/?format=api

HTTP 200 OK

Allow: GET, POST, OPTIONS

Content-Type: application/json

Vary: Accept

```
[
  {
    "fname": "Sourabh",
    "lname": "Pal",
    "distance": 27.0
  },
  {
    "fname": "Vishal",
    "lname": "Kushwa",
    "distance": 35.0
  },
  {
    "fname": "Sushmita",
    "lname": "D'Suza",
    "distance": 55.0
  },
  {
    "fname": "Radha",
    "lname": "Shyam",
    "distance": 100.0
  }
]
```

DAY 10

Project Web Scraper

BEAUTIFULSOUP

INSTRUCTIONS

- DIRECTORY NAME: scraper
- PROJECT NAME: mysite
- APP NAME: myapp
- Pip install requests
- Pip install django rest framework
- Pip install beautifulsoup4
- Python manage.py migrate, makemigrations
- Create files – urls.py (app), results.html (in templates)

URLS.PY - PROJECT

```
14     1. Import the include() function: from django.urls import include
15     2. Add a URL to urlpatterns: path('blog/', include('blog.urls'))
16     """
17     from django.contrib import admin
18     from django.urls import path, include
19     from myapp import views
20
21     urlpatterns = [
22         path('admin/', admin.site.urls),
23         path('', views.scrape, name="scrape"),
24         path('delete/', views.clear, name="delete")
25     ]
26
```

VIEWS.PY

```
Go Run Terminal Help ← → mysite
settings.py results.html urls.py views.py × admin.py
myapp > views.py > clear
1 from django.shortcuts import render
2 import requests
3 from bs4 import BeautifulSoup
4 from django.http import HttpResponseRedirect
5 from .models import Link
6 # Create your views here.
7
8 def scrape(request):
9     if request.method == "POST":
10         site = request.POST.get('site','')
11
12         page = requests.get(site)
13         soup = BeautifulSoup(page.text, 'html.parser')
14
15
16
17         for link in soup.find_all('a'):
18             link_address = link.get('href')
19             link_text = link.string
20             Link.objects.create(address=link_address,name=link_text)
21         return HttpResponseRedirect('/')
22     else:
23         data = Link.objects.all()
24
25         return render(request, 'results.html', {'data':data})
26
27 def clear(request):
28     Link.objects.all().delete()
29     return render(request, 'results.html')
```

ADMIN.PY

```
• settings.py  • results.html  • admin.py  ↗  
myapp > 🐘 admin.py  
1  from django.contrib import admin  
2  
3  # Register your models here.  
4  from .models import Link  
5  admin.site.register(Link)  
6  
7  |
```

MODELS.PY

```
settings.py  results.html  models.py X

myapp > models.py > Link
1  from django.db import models
2
3  # Create your models here.
4  class Link(models.Model):
5
6      def __str__(self):
7          return self.name
8
9      address = models.CharField(max_length=1000,null=True,blank=True)
10     name = models.CharField(max_length=1000,null=True,blank=True)
```

RESULTS.HTML

```
myapp > templates > results.html > html
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4
5     <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css" integrity="sha384-ggOyR0iXCbMQv3Xipri
6     <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.min.js" integrity="sha384-JjSmVgyd0p3pXB1rRibZUAYoIIy60rQ6Vrj:
7 <meta charset="UTF-8">
8 <meta name="viewport" content="width=device-width, initial-scale=1.0">
9 <meta http-equiv="X-UA-Compatible" content="ie=edge">
10 <title>Document</title>
11 </head>
12 <body>
13
14     <div class="container">
15         <div class="row">
16             <div class="col-md-12 m-5">
17                 <h1>Link Collector</h1>
18             </div>
19         </div>
20         <div class="row m-5">
21             <div class="col-md-4">
22                 <form method="POST" action="/">
23                     {% csrf_token %}
24                     <input class="form-control" name="site" type="text" id="site" placeholder="enter site address">
25                 </div>
26                 <div class="col-md-2">
27                     <button class="btn btn-primary" type="submit">Scrape</button>
28                 </div>
29             </form>
30             <div class="col-md-6">
31                 <a class="btn btn-warning" href="/delete">Delete</a>
32             </div>
33         </div>
34     </div>
35
```

RESULTS.HTML

```
35
36
37     <div class="row m-5">
38         <div class="col-md-8">
39             <table class="table">
40                 <thead class="thead-dark">
41                     <tr>
42                         <th scope="col">ID</th>
43                         <th scope="col">Name</th>
44                         <th scope="col">Link</th>
45                     </tr>
46                 </thead>
47                 <tbody>
48                     {% for link in data %}
49                     <tr>
50                         <td>{{link.id}}</td>
51                         <td>{{link.name}}</td>
52                         <td>{{link.address}}</td>
53                     </tr>
54                     {% endfor %}
55                 </tbody>
56             </table>
57         </div>
58     </div>
59
60 </div>
61
62 </div>
63
64
65
66
67
68 </body>
69 </html>
```

SETTINGS.PY

```
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     'myapp'
41 ]
42
```


ADMIN PAGE

Link Collector

ScrapeDelete

ID	Name	Link
----	------	------

ADMIN PAGE - SCRAPPING

Link Collector

ScrapeDelete

ID	Name	Link
1	None	/
2	None	/
3	About	https://www.youtube.com/about/
4	Press	https://www.youtube.com/about/press/
5	Copyright	https://www.youtube.com/about/copyright/
6	Contact us	/t/contact_us/
7	Creator	https://www.youtube.com/creators/
8	Advertise	https://www.youtube.com/ads/
9	Developers	https://developers.google.com/youtube
10	Terms	/t/terms
11	Privacy	/t/privacy
12	Policy & Safety	https://www.youtube.com/about/policies/
13	How YouTube works	https://www.youtube.com/howyoutubeworks?utm_campaign=ytgen&utm_source=ythp&utm_medium=LeftNav&utm_content=txt&u=https%3A%2F%2Fwww.youtube.com%2Fhowyoutubeworks%3Futm_source%3Dythp%26utm_medium%3DLeftNav%26utm_campaign%3Dytgen
14	Test new features	/new

ADMIN PAGE - DELETING

127.0.0.1:8000/delete/

Link Collector

Scrape

Delete

ID	Name	Link
----	------	------

