Lab 5 – Student CRUD Application

In this exercise you will create a simple Student Data CRUD application that allows users to create, edit, and delete student entries. The homepage will list all student names and there will be a dedicated student details page for each individual student. On the student details page there will be links to edit & delete the student.

Initial Set Up

1. Go to www.github.com and log in to your GitHub account.
2. Go to Moodle and click on the link for Lab 5 Upload Student CRUD Application.
3. Once you have accepted the assignment you are asked to refresh the page, and you are then presented with a link to your repository for lab 5 as shown below:
4. When you click on the repository link, you are taken to the repository where you see a readme file has already been added:
5. In Windows Command line make sure that you are in the **djangoprojects** folder and use the following command to activate the virtual environment:

```
env\scripts\activate.bat
```

6. Use the following command to clone this repo to your local computer:

```
git clone <repo name>
```

Replace this part with your repo name.

You can get the repo details by clicking on the Code button in the repo. After the clone command is finished you have a new folder called **lab-5-student-crud-application-<username>** inside the **djangoprojects** folder.

7. Move into this new folder using the **cd** command.

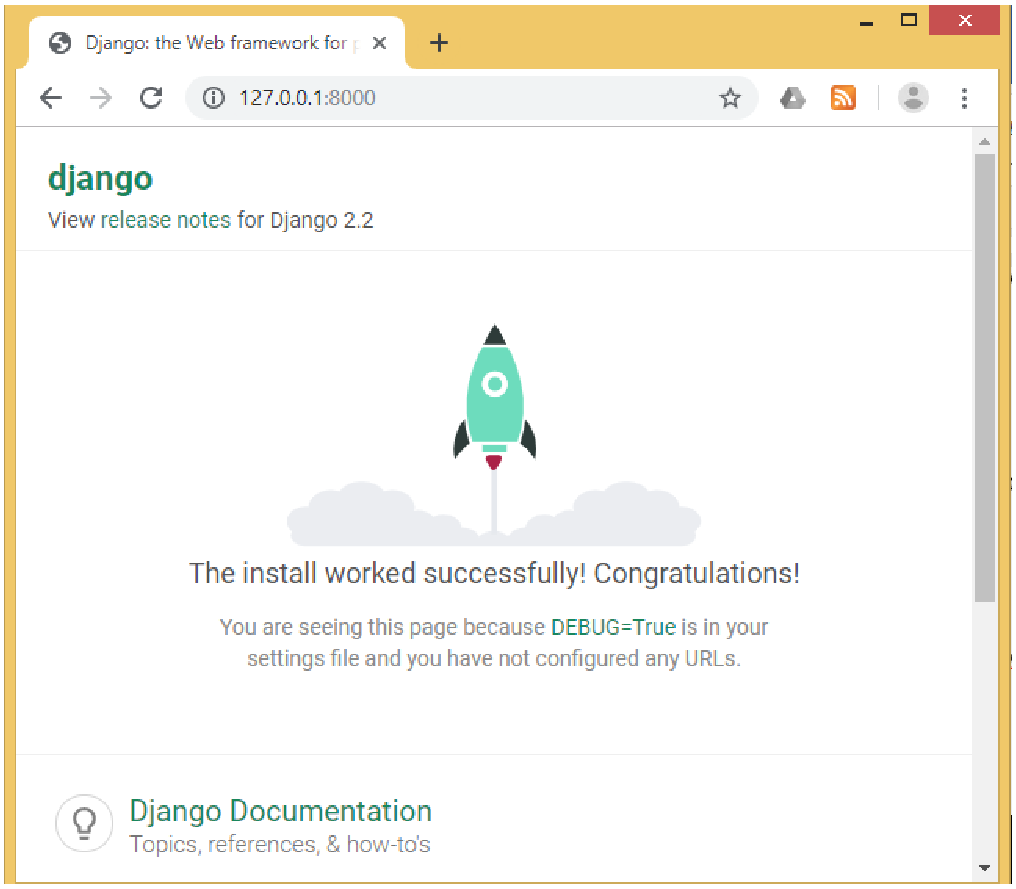
8. Create a new Django project called **studentproject** with the following command. Don't forget the period (fullstop) at the end:

```
django-admin startproject studentproject .
```

9. Verify that the Django project works by typing in the following command:

```
python manage.py runserver 8080
```

10. Open http://127.0.0.1:8080/ and you should see the familiar Django welcome page:



django

View release notes for Django 2.2

The install worked successfully! Congratulations!

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

Django Documentation

Topics, references, & how-to's

The output in the command line shows a warning about "18 unapplied migrations" although this warning has no effect on the project at this point. Django is letting us know that we have not yet “migrated” or configured our initial database. Since we don’t use a database in this exercise, the warning won’t affect the result.

11. You can remove the warning by running the migrate command as shown here. You will need to stop the server first using **Control+c**:

---

python manage.py migrate

When you run this command, you will see in the output that 18 migrations are applied. We will look at the meaning of these migrations later. If you execute the **python manage.py runserver 8080** again, the warning message is gone.

12. Create an app called **student**. From the command line, quit the server with **Control+c**. Then use the **startapp** command as shown below:

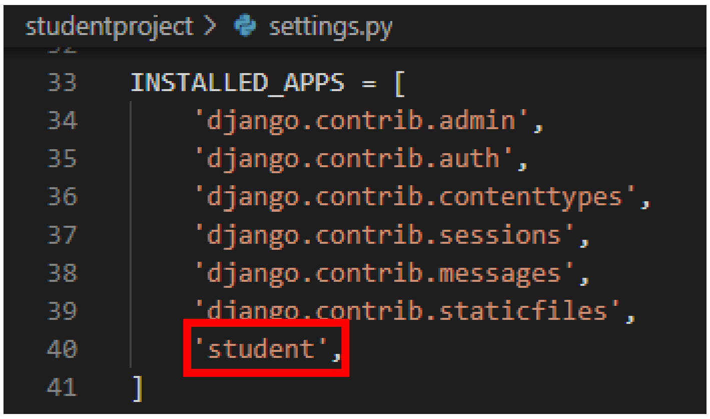
---

python manage.py startapp student

Settings.py

1. To ensure Django knows about our new app, open the project using VS Code and add the new app to **INSTALLED_APPS** in the **settings.py** file:

Code



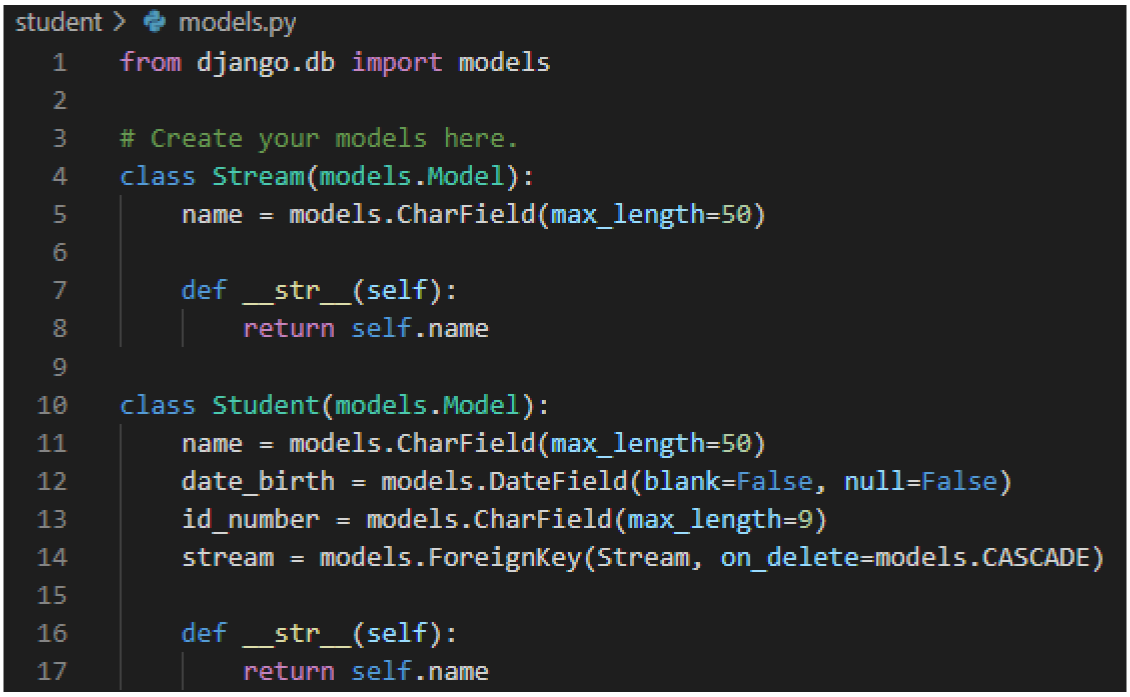
```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'student',
]
```

Database Models

Next, we will create a **Stream** class and a **Student** class.

1. Open the file **student/models.py** and enter the code below:

Code



student > models.py

```python
from django.db import models

# Create your models here.
class Stream(models.Model):
    name = models.CharField(max_length=50)

    def __str__(self):
        return self.name

class Student(models.Model):
    name = models.CharField(max_length=50)
    date_birth = models.DateField(blank=False, null=False)
    id_number = models.CharField(max_length=9)
    stream = models.ForeignKey(Stream, on_delete=models.CASCADE)

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

In the **Student** class, note the use of the **DateField** to represent the date of birth. For the stream field we use a ForeignKey which allows for a many-to-one relationship. This means that a given stream can have many students. Remember that for all many-to-one relationships such as a ForeignKey we must also specify an on_delete option.

Now that our new database models are created, we need to create a new migration record and migrate the change into our database.

2. Stop the server with Control+c. This two-step process can be completed with the commands below:

Command Line

Type the two commands shown below:

---

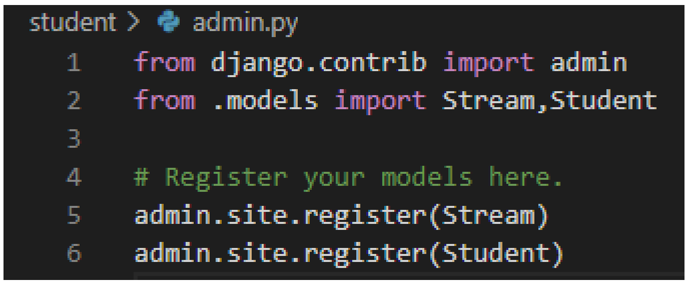
python manage.py makemigrations student

---

python manage.py migrate

Admin

1. To access our data using Django admin we need to register our models. Open **student/admin.py** & add the following code:



```
student > admin.py
from django.contrib import admin
from .models import Stream,Student

# Register your models here.
admin.site.register(Stream)
admin.site.register(Student)
```

2. Create a superuser account in **Windows Command Line** using the following command & follow the instructions:

---

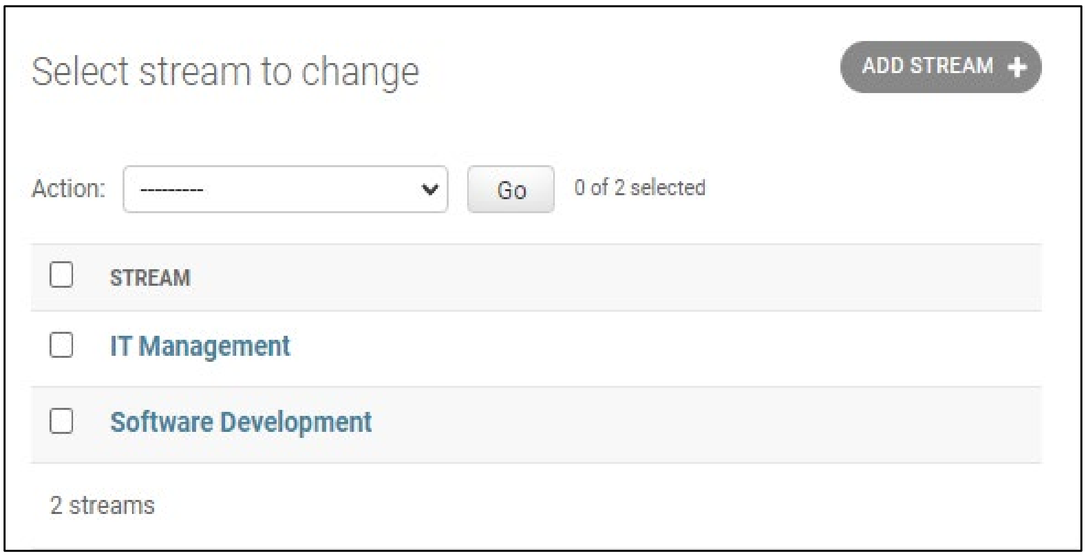
python manage.py createsuperuser

---

3. Start running the Django server again with the command **python manage.py runserver 8080** and open the Django admin page at http://127.0.0.1:8080/admin/.

4. Log in with your new superuser account.
5. Add two **Stream** objects and two **Student** objects to the database as shown below:

Here is the admin homepage with the objects added:

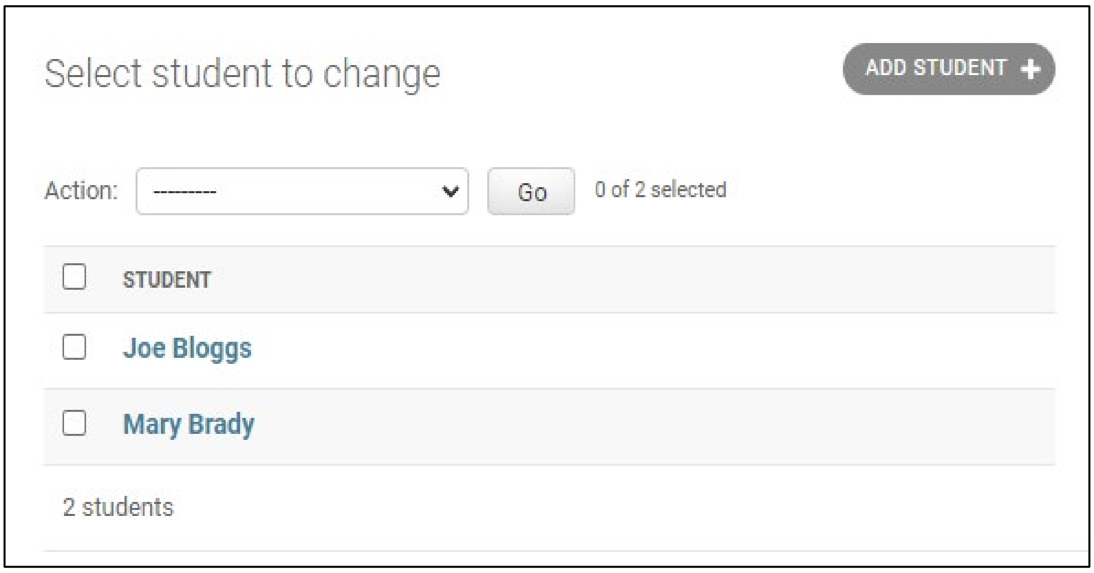


Select stream to change

0 of 2 selected

STREAM
IT Management
Software Development

2 streams



Select student to change

0 of 2 selected

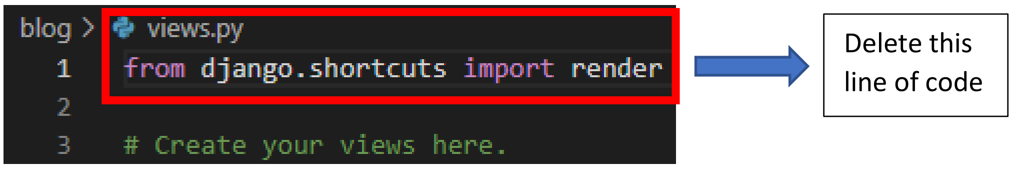
STUDENT
Joe Bloggs
Mary Brady

2 students

To display the information on our web application, we need to create the necessary views, URLs, and templates.

Views

1. Open the file **student/views.py** & delete the line of code at the top of the file.

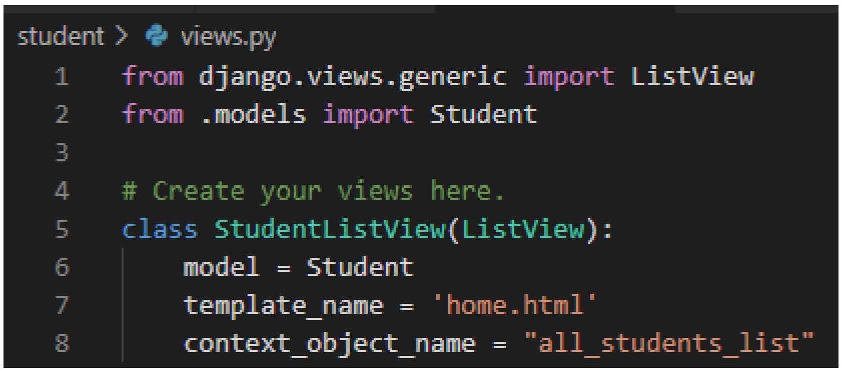


```
blog > views.py
from django.shortcuts import render

# Create your views here.
```

Delete this line of code

2. Add the code below to display the list of **Student** objects using **ListView**.

Code



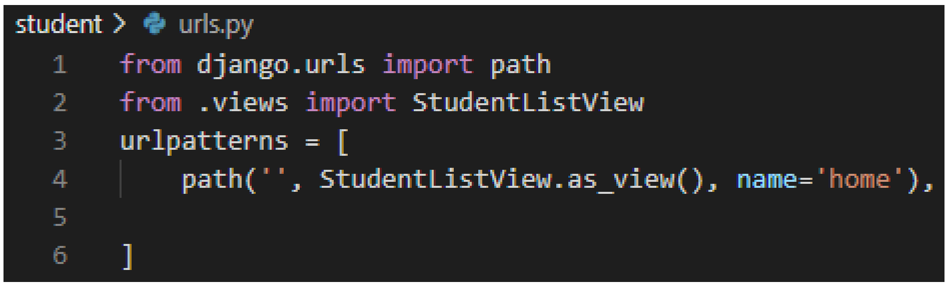
```python
from django.views.generic import ListView
from .models import Student

# Create your views here.
class StudentListView(ListView):
    model = Student
    template_name = 'home.html'
    context_object_name = "all_students_list"
```

On the top two lines we import **ListView** and our database model **Student**. Then we subclass **ListView** and set values for our model and template.

URLs

1. In VS Code create a new **urls.py** inside the **student** app & update it with the code below

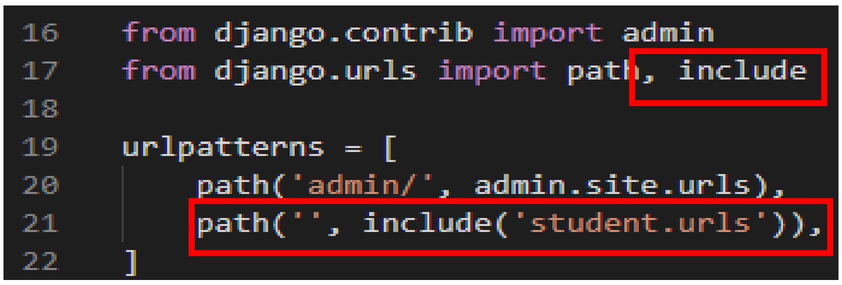
Code



```python
from django.urls import path
from .views import StudentListView
urlpatterns = [
    path('', StudentListView.as_view(), name='home'),

]
```

2. Add the following code to the **studentproject/urls.py** file so that Django knows to forward all requests directly to the **student** app.

Code

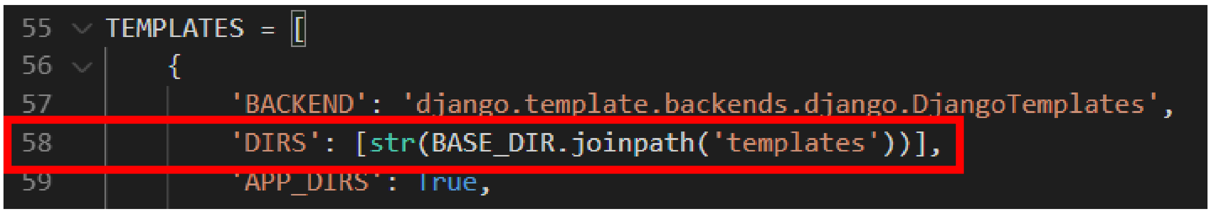


```python
from django.contrib import admin
from django.urls import path, include

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

Templates

1. In VS Code create a new folder called **templates.**
2. Inside the **templates** folder, create two new files: **base.html** and **home.html**:
3. Update the **DIRS** field in our **settings.py** file so that Django knows to look in this **templates** directory for our html files.

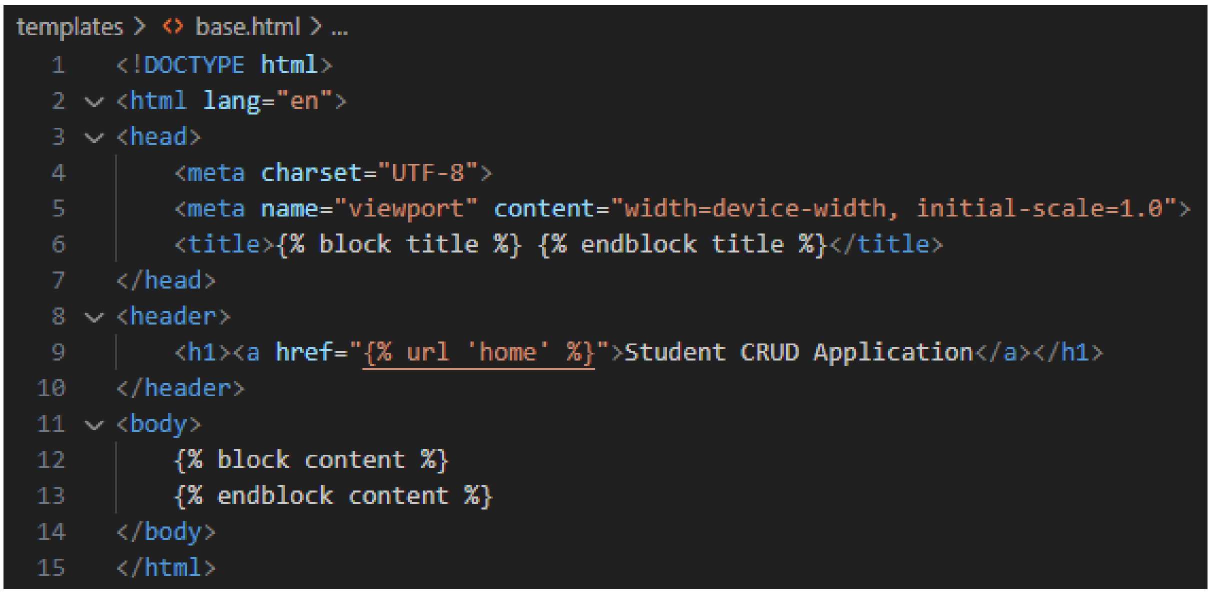
Code



```
TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [str(BASE_DIR.joinpath('templates'))],
        'APP_DIRS': True,
```

Open **base.html** and use the keyboard keys **shift, 1 and enter** to auto-populate HTML Doctype in HTML

Code



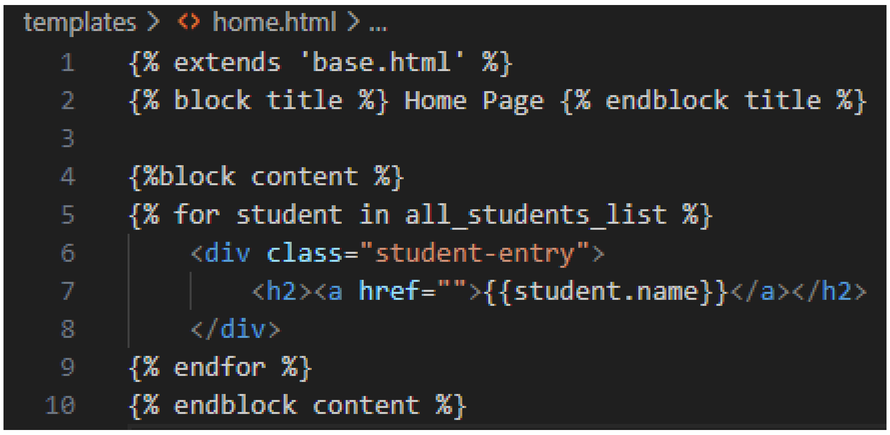
templates > <> base.html > ...

```
<!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 %}</title>
</head>
<header>
    <h1><a href="{% url 'home' %}">Student CRUD Application</a></h1>
</header>
<body>
    {% block content %}
    {% endblock content %}
</body>
</html>
```

Note that the code between {% block %} and {% endblock %} can be filled by other child templates.

1. Enter the following code into **home.html**

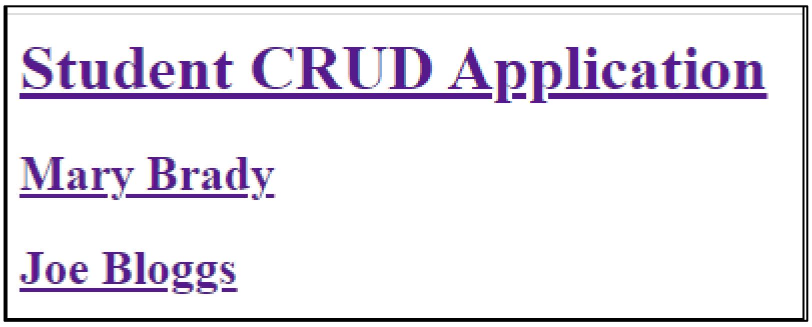
Code



```
templates > <> home.html > ...
{% extends 'base.html' %}
{% block title %} Home Page {% endblock title %}

{%block content %}
{% for student in all_students_list %}
    <div class="student-entry">
        <h2><a href="">{{student.name}}</a></h2>
    </div>
{% endfor %}
{% endblock content %}
```

Notice that in the **home.html** file we have removed all of the boilerplate skeleton HTML from before. It is no longer needed since when this page loads, it will inherit the HTML found in the **base** template. We also see those block template tags in this template file as well. We have a **title block** and a **content block**. Any data inside of these blocks will be dynamically injected into the block definitions we created in the **base.html** template.

2. Start the Django server again using the command: **python manage.py runserver 8080**
3. Refresh the web page & we can see that the application is working:



Student CRUD Application

Mary Brady

Joe Bloggs

The next step is to add some CSS styling to improve the appearance of the web site.

Static files

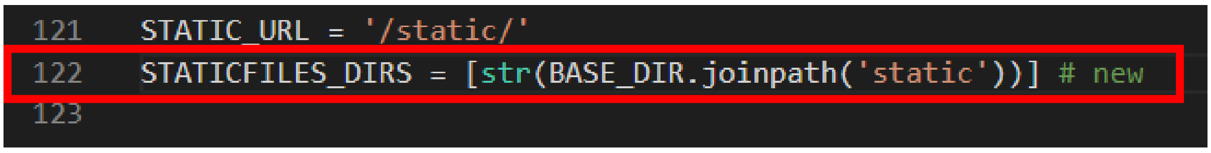
We need to add some CSS which is referred to as a static file because, unlike our dynamic database content, it doesn't change.

1. In **VS Code** create a new folder called **static** inside the **lab5-student-crud-application-<username>** folder.

Just as we did with our **templates** directory, we need to update **settings.py** to tell Django where to look for these static files. We can update **settings.py** with a one-line change for **STATICFILES_DIRS**.

2. Scroll to the end of the file and add this line at the bottom of the file below the entry for **STATIC_URL**

Code



```
STATIC_URL = '/static/'
STATICFILES_DIRS = [str(BASE_DIR.joinpath('static'))] # new
```

3. In VS Code create a **css** folder within the **static** folder.
4. In VS Code right click on the **css** folder and create a new **base.css** file within it
5. Copy-paste the following **css** code to the file

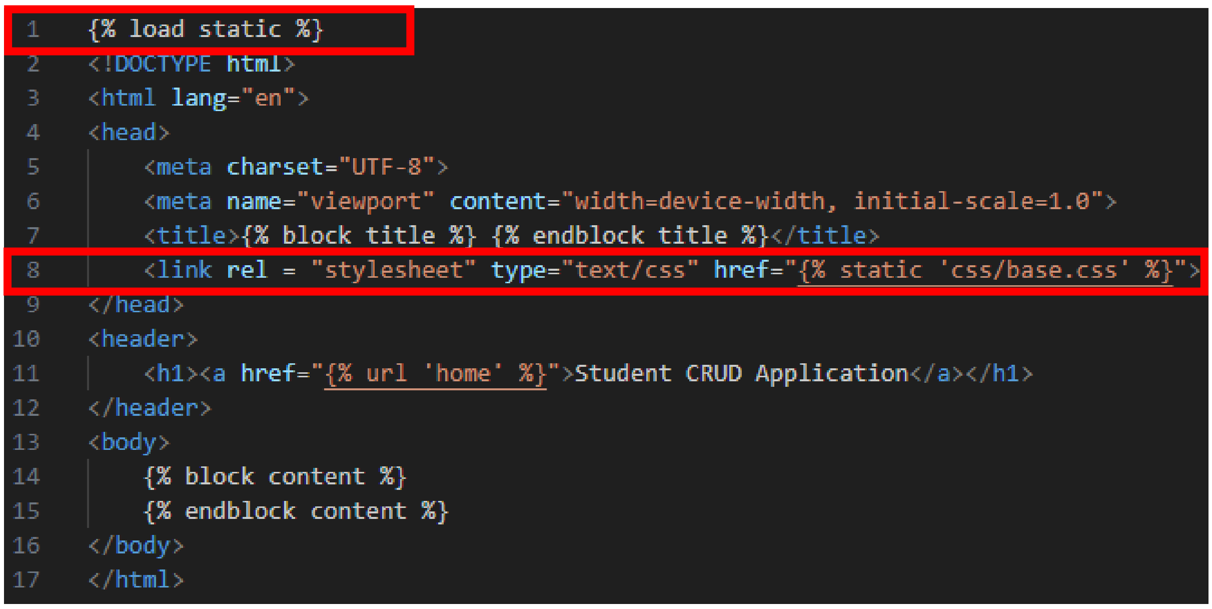
Code

```
body {
  font-family: 'Source Sans Pro', sans-serif;
  font-size: 18px;
}

header {
  border-bottom: 1px solid #999;
  margin-bottom: 2rem;
  display: flex;
}

header h1 a {
  color: red;
  text-decoration: none;
}

.student-entry {
  margin-bottom: 2rem;
}

.student-entry h2 {
  margin: 0.5rem 0;
}

.student-entry h2 a,
.student-entry h2 a:visited {
  color: blue;
  text-decoration: none;
}

.student-entry p {
  margin: 0;
  font-weight: 400;
}

.student-entry h2 a:hover {
  color: red;
}
```

We need to add the static files to our templates.

6. To do this, add **{% load static %}** to the top of **base.html** as shown in line 1 below
7. Include a new line as shown in line 8 below at the bottom of the <head></head> code that explicitly references our new **base.css** file.

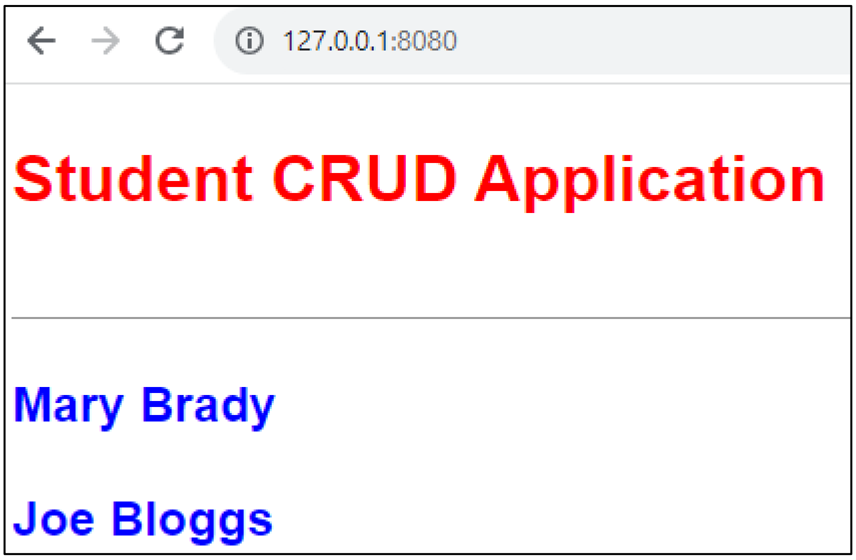
Code



```
{% load static %}
<!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 %}</title>
    <link rel = "stylesheet" type="text/css" href="{% static 'css/base.css' %}">
</head>
<header>
    <h1><a href="{% url 'home' %}">Student CRUD Application</a></h1>
</header>
<body>
    {% block content %}
    {% endblock content %}
</body>
</html>
```

Now we can add static files to our static directory, and they will automatically appear in all our templates.

8. In Windows Command Line, check that the server is running and reload the updated homepage at http://127.0.0.1:8080/



127.0.0.1:8080

Student CRUD Application

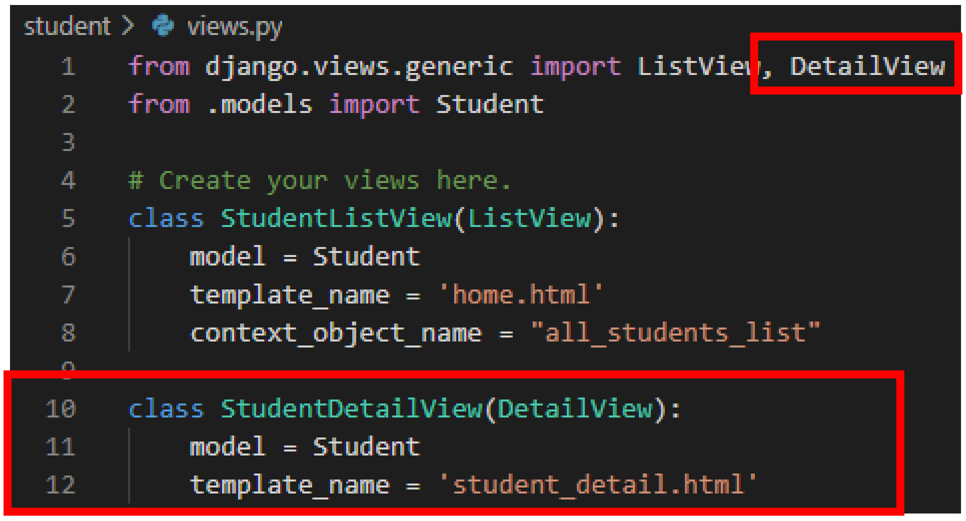
Mary Brady

Joe Bloggs

Individual student pages

1. At the top of the **views.py** file add **DetailView** to the list of imports and then add the code from lines 10-12 to create a new view called **StudentDetailView:**

Code

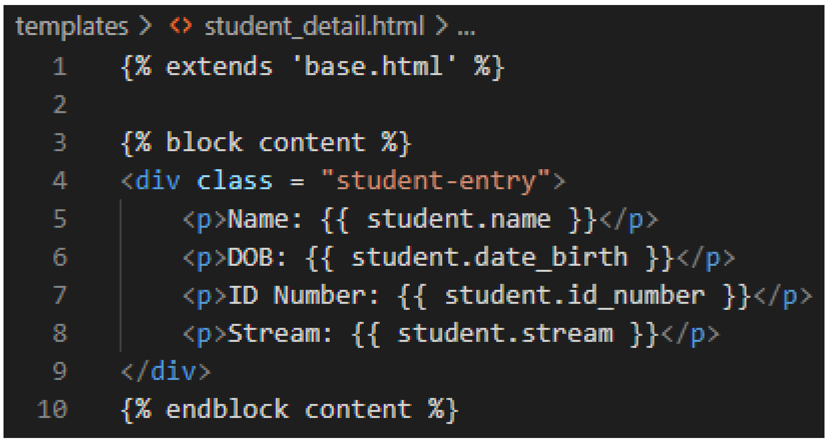


```python
1   from django.views.generic import ListView, DetailView
2   from .models import Student
3
4   # Create your views here.
5   class StudentListView(ListView):
6       model = Student
7       template_name = 'home.html'
8       context_object_name = "all_students_list"
9
10  class StudentDetailView(DetailView):
11      model = Student
12      template_name = 'student_detail.html'
```

In this new view, we define the model we are using called **Student** and the template we want it associated with, **student_detail.html**.

2. In VS Code create a new template called **student_detail.html** and add in the code below.

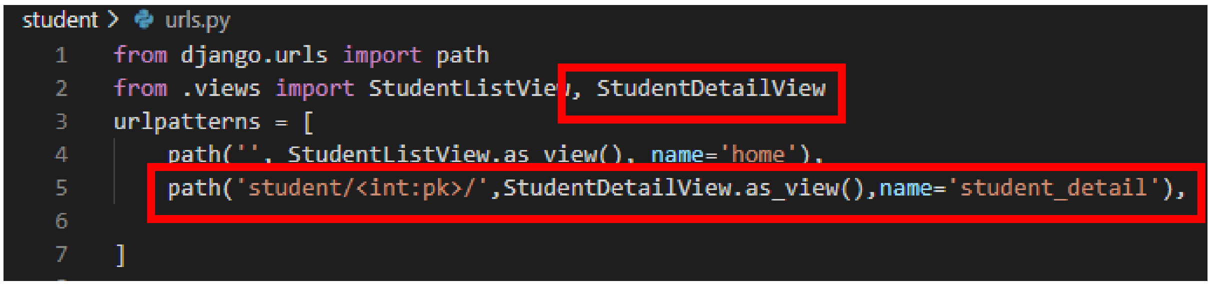
Code



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

{% block content %}
<div class = "student-entry">
    <p>Name: {{ student.name }}</p>
    <p>DOB: {{ student.date_birth }}</p>
    <p>ID Number: {{ student.id_number }}</p>
    <p>Stream: {{ student.stream }}</p>
</div>
{% endblock content %}
```

3. In **student/urls.py** add a new URLConf for our view as follows:

Code

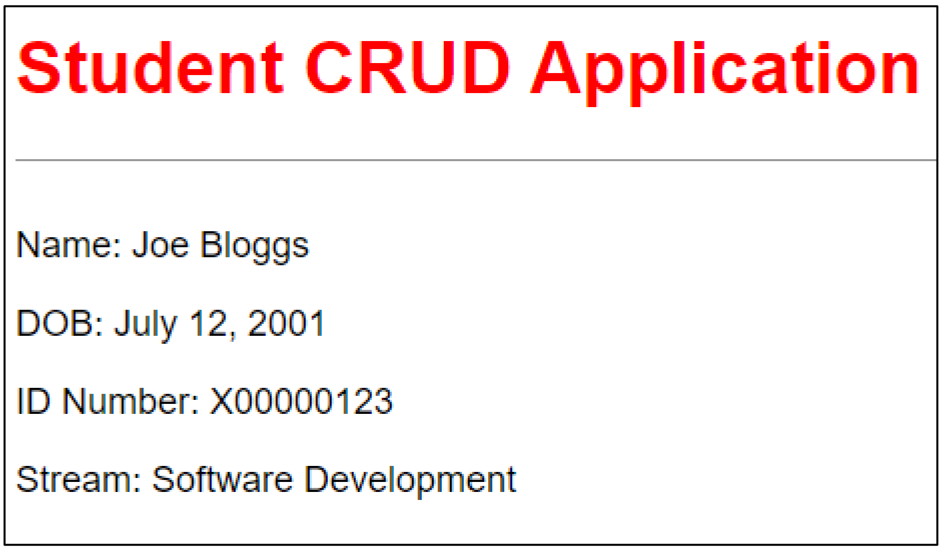


```
student > urls.py
from django.urls import path
from .views import StudentListView, StudentDetailView
urlpatterns = [
    path('', StudentListView.as_view(), name='home'),
    path('student/<int:pk>/',StudentDetailView.as_view(),name='student_detail'),

]
```

All student entries will start with student/. The next part is the primary key for our student entry which is represented as an integer <int:pk>. Django automatically adds an auto-incrementing primary key to our database models.

4. Make sure the server is running and and go directly to http://127.0.0.1:8080/student/1/

You will see a dedicated page for the first student as shown below:



Student CRUD Application

Name: Joe Bloggs

DOB: July 12, 2001

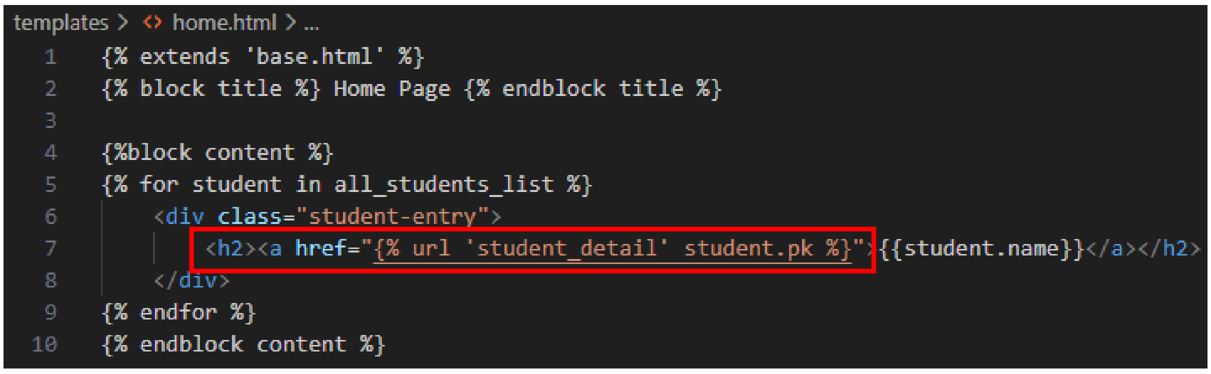
ID Number: X00000123

Stream: Software Development

5. Go to http://127.0.0.1:8080/Student/2/ to see the second entry.

Update the link on the homepage so we can directly access individual students from there. Currently in **home.html** our link is empty: **<a href="">**. Update it as shown in the code below:

Code



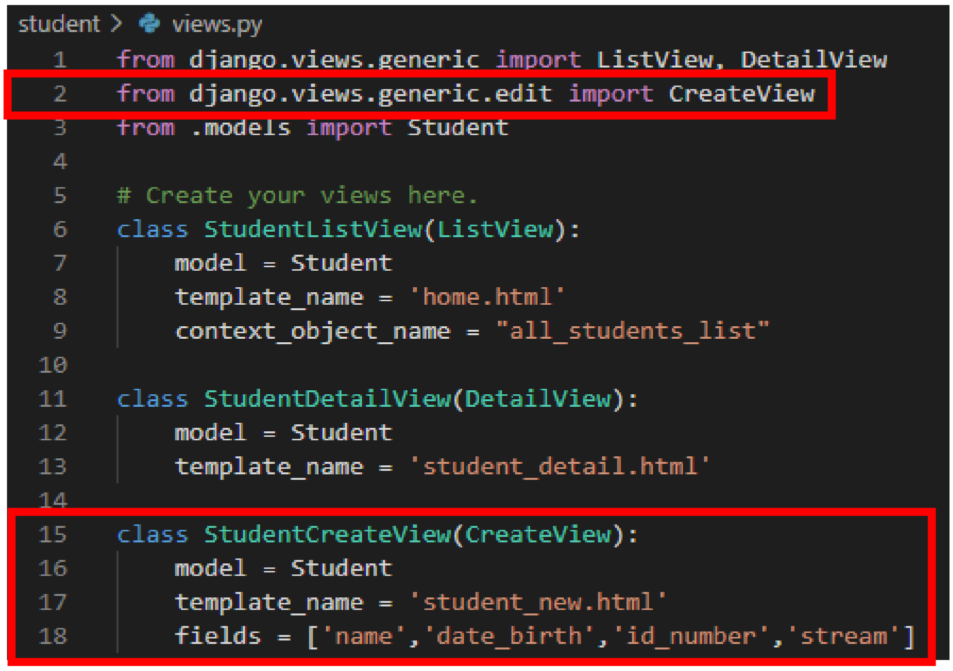
```
templates > <> home.html > ...
{% extends 'base.html' %}
{% block title %} Home Page {% endblock title %}

{%block content %}
{% for student in all_students_list %}
    <div class="student-entry">
        <h2><a href="{% url 'student_detail' student.pk %}">{{student.name}}</a></h2>
    </div>
{% endfor %}
{% endblock content %}
```

6. To confirm everything works, refresh the main page at http://127.0.0.1:8080/ and click on the title of each student to confirm the new link works.

Create New Student

Open the **views.py** file and create a new view using the code provided below. Here we are importing a new generic class called **CreateView** at the top and then subclassing it to create a new view called **StudentCreateView**.

Code

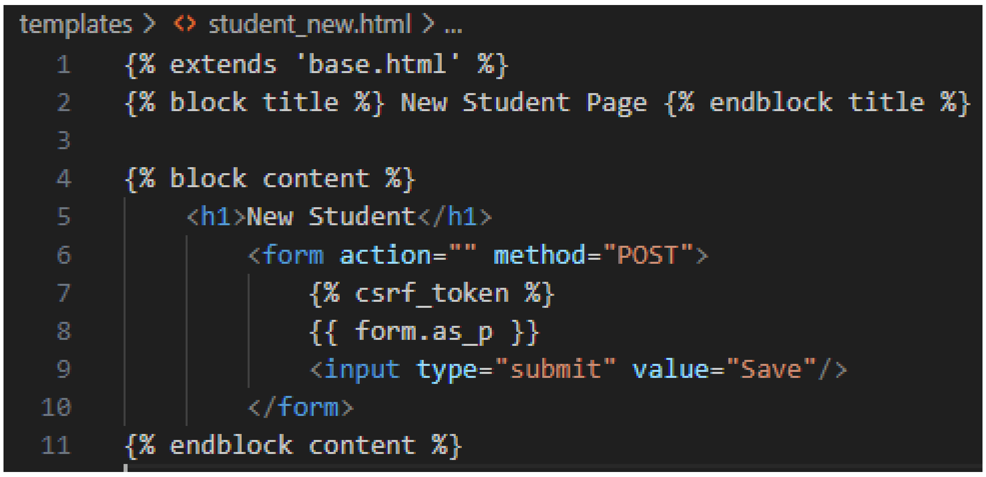


```python
from django.views.generic import ListView, DetailView
from django.views.generic.edit import CreateView
from .models import Student

# Create your views here.
class StudentListView(ListView):
    model = Student
    template_name = 'home.html'
    context_object_name = "all_students_list"

class StudentDetailView(DetailView):
    model = Student
    template_name = 'student_detail.html'

class StudentCreateView(CreateView):
    model = Student
    template_name = 'student_new.html'
    fields = ['name','date_birth','id_number','stream']
```

Template

Create a new template in VS Code called **student_new.html** and type in the following code:

Code

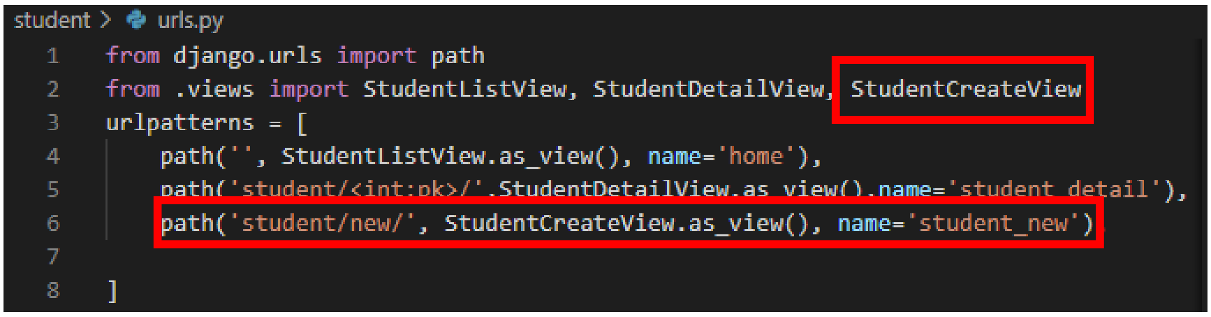


```html
{% extends 'base.html' %}
{% block title %} New Student Page {% endblock title %}

{% block content %}
    <h1>New Student</h1>
        <form action="" method="POST">
            {% csrf_token %}
            {{ form.as_p }}
            <input type="submit" value="Save"/>
        </form>
{% endblock content %}
```

URLConf

In **student/urls.py** add a new URLConf for **post_new** using the code shown below.

Code



```
student > urls.py
1    from django.urls import path
2    from .views import StudentListView, StudentDetailView, StudentCreateView
3    urlpatterns = [
4        path('', StudentListView.as_view(), name='home'),
5        path('student/<int:pk>/',StudentDetailView.as_view(),name='student_detail'),
6        path('student/new/', StudentCreateView.as_view(), name='student_new')
7
8    ]
```

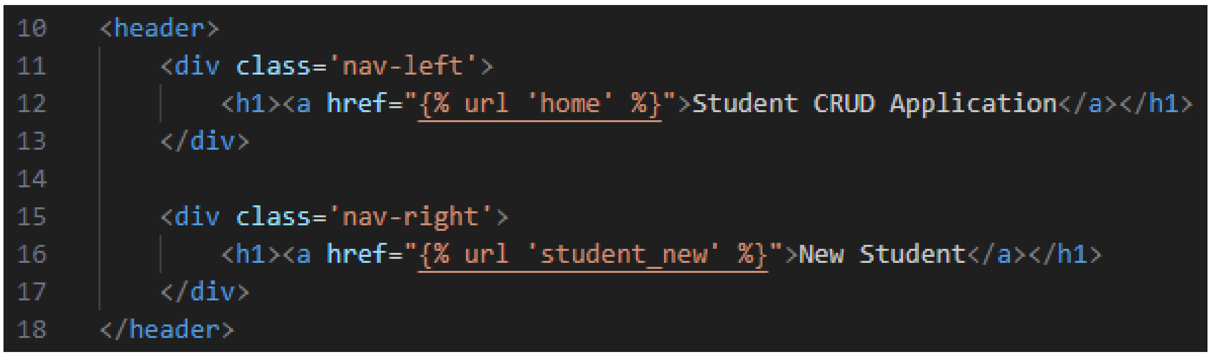
We just imported our view called **StudentCreateView** at line 2 and then we added the path to the new URL which will start with **student/new/** and is called **student_new**.

Update Template

We will need to update our base template to display a link to a page for entering new student details. It will take the form **<a href="{% url 'student_new' %}"></a>** where **student_new** is the name for our URL.

Open the **base.html** file. Update the code inside the header to look like the file shown below. Make sure to add the code to define the div classes **nav-left** 1 and **nav-right** 2 for both URLs in the header to enable CSS styling to be applied from **`base.css`**.

Code



```
<header>
    <div class='nav-left'>
        <h1><a href="{% url 'home' %}">Student CRUD Application</a></h1>
    </div>

    <div class='nav-right'>
        <h1><a href="{% url 'student_new' %}">New Student</a></h1>
    </div>
</header>
```

Update CSS

Open **base.css** and add the following code to the style nav-left and nav-right classes:

```
  .nav-left {
    margin-right: auto;
  }


  .nav-right {
    display: flex;
    padding-top: 2rem;
  }
```

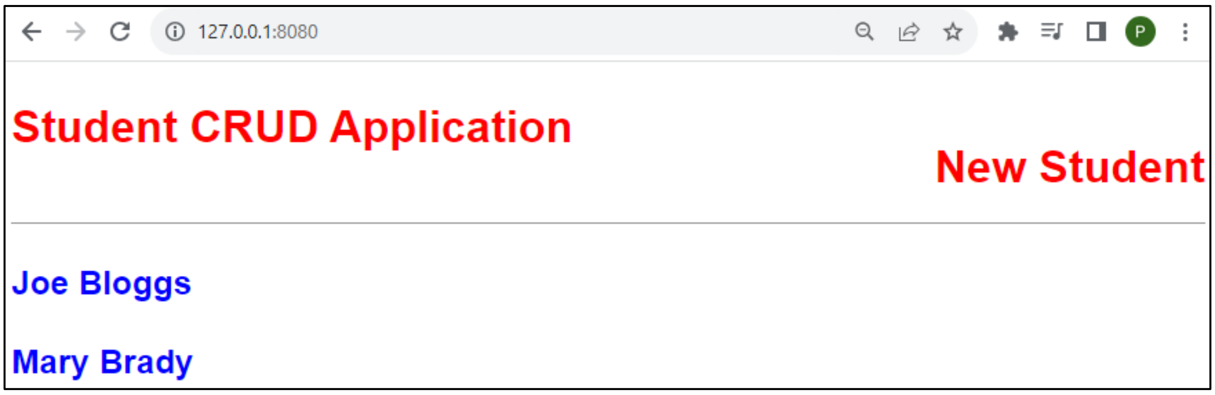
Before you run the server, add the following import and function to the **Student** class in **models.py** to direct the user back to the student details page after the new student has been added.

```
from django.urls import reverse
```

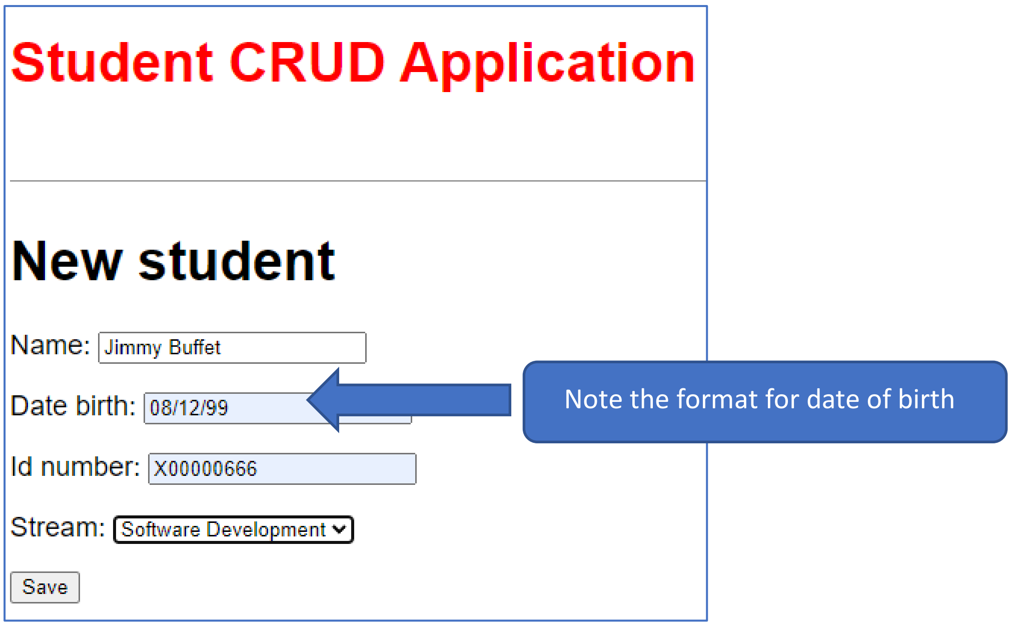
```
    def get_absolute_url(self):
        return reverse('student_detail', args=[str(self.id)])
```

Run Server

Start the server with the command **python manage.py runserver** and go to the homepage at http://127.0.0.1:8080/.



Click on our link for "New Student" which will redirect you to: http://127.0.0.1:8080/student/new/ where you can add a new student. **Note the format below for the date of birth**.


Student CRUD Application

New student

Name: Jimmy Buffet

Date birth: 08/12/99

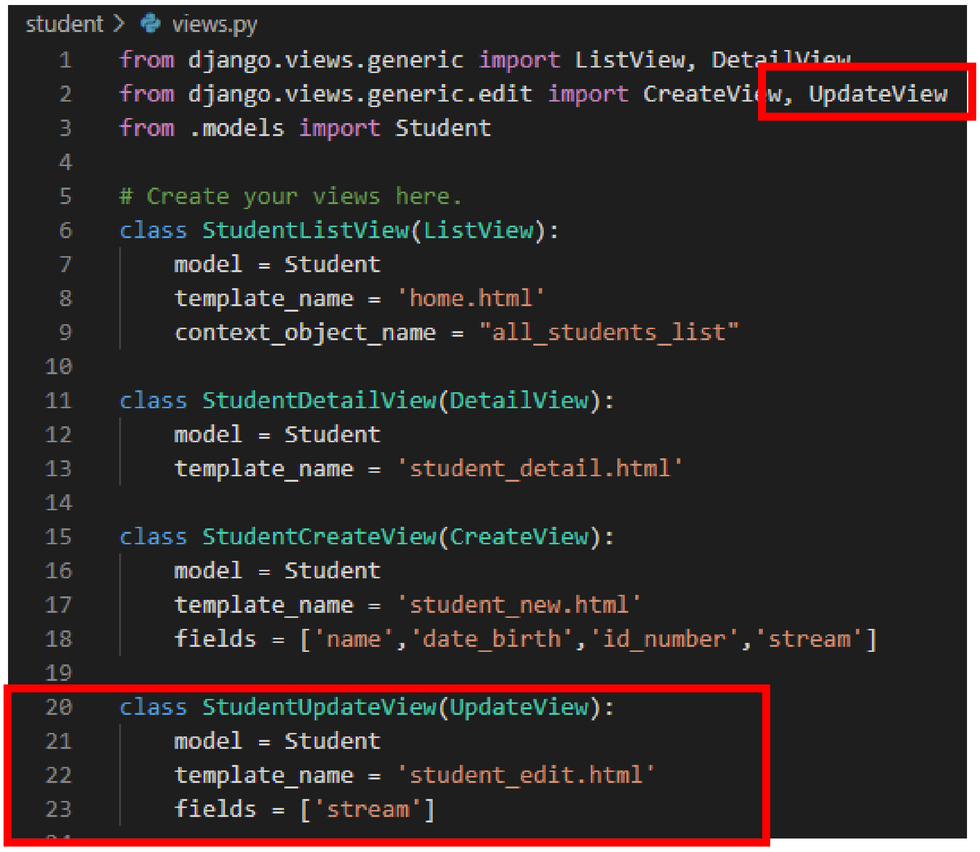
Id number: X00000666

Stream: Software Development

Update Student details

We need to import **UpdateView** on the second-from-the-top line and then subclass it in our new view **StudentUpdateView**. Add the code highlighted below to the **views.py** file.

Code



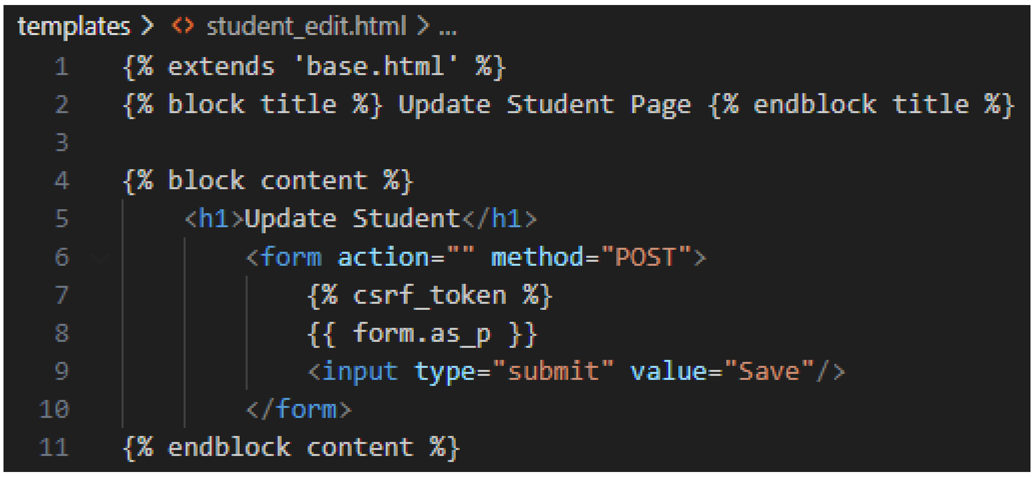
```python
from django.views.generic import ListView, DetailView
from django.views.generic.edit import CreateView, UpdateView
from .models import Student

# Create your views here.
class StudentListView(ListView):
    model = Student
    template_name = 'home.html'
    context_object_name = "all_students_list"

class StudentDetailView(DetailView):
    model = Student
    template_name = 'student_detail.html'

class StudentCreateView(CreateView):
    model = Student
    template_name = 'student_new.html'
    fields = ['name','date_birth','id_number','stream']

class StudentUpdateView(UpdateView):
    model = Student
    template_name = 'student_edit.html'
    fields = ['stream']
```

Notice that in **StudentUpdateView** we are explicitly listing the field we want to use ['stream']. The only field than a student can change is the stream that they are in.

Template

Create the template in VS Code for the edit page called **student_edit.html** & add in the following code.

Code



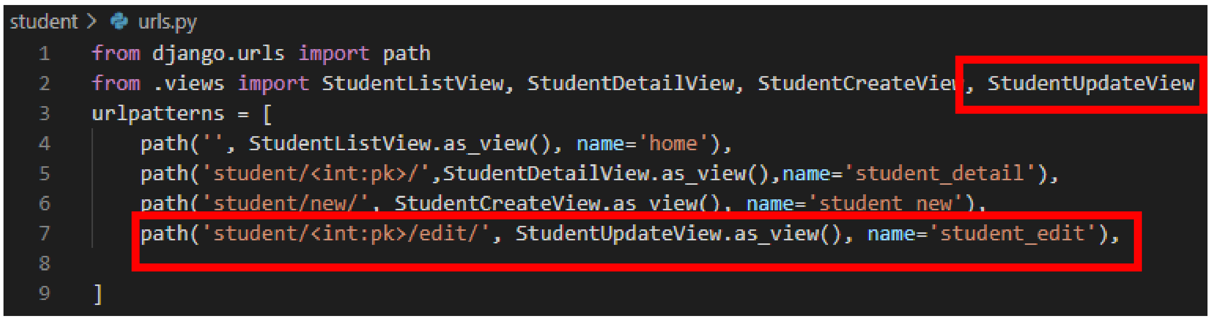
templates > student_edit.html > ...

```
{% extends 'base.html' %}
{% block title %} Update Student Page {% endblock title %}

{% block content %}
    <h1>Update Student</h1>
        <form action="" method="POST">
            {% csrf_token %}
            {{ form.as_p }}
            <input type="submit" value="Save"/>
        </form>
{% endblock content %}
```

URLConf

Update the **student/urls.py** file as follows. Add the **StudentUpdateView** at the end of line 2 and then add the new route at line 7.

Code

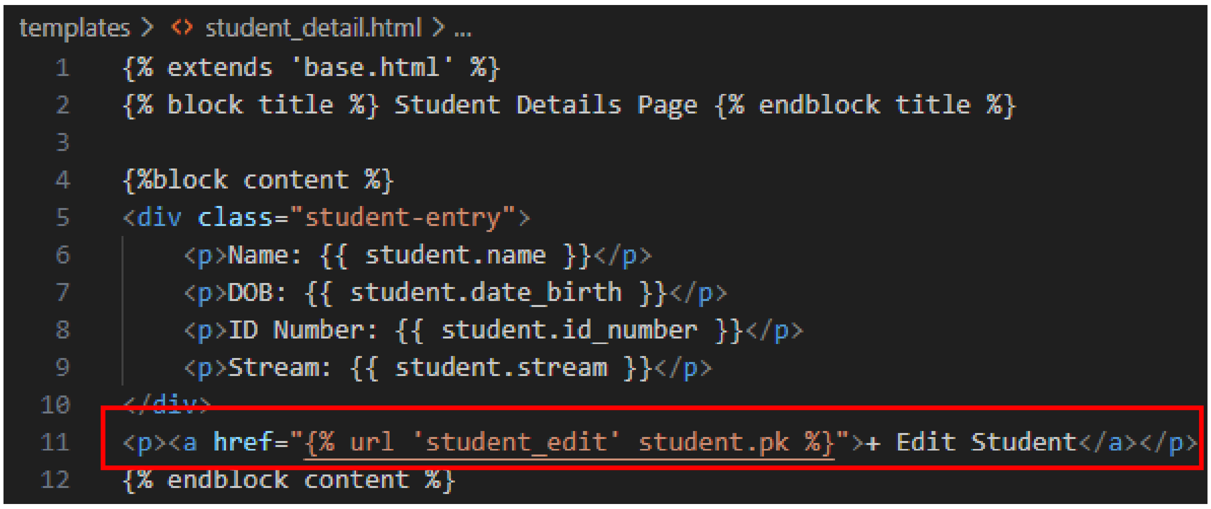


student > urls.py

```
1   from django.urls import path
2   from .views import StudentListView, StudentDetailView, StudentCreateView, StudentUpdateView
3   urlpatterns = [
4       path('', StudentListView.as_view(), name='home'),
5       path('student/<int:pk>/',StudentDetailView.as_view(),name='student_detail'),
6       path('student/new/', StudentCreateView.as_view(), name='student_new'),
7       path('student/<int:pk>/edit/', StudentUpdateView.as_view(), name='student_edit'),
8   
9   ]
```

Add a new link shown at line 11 below to **student_detail.html** so that the option to edit a student appears on an individual student page.

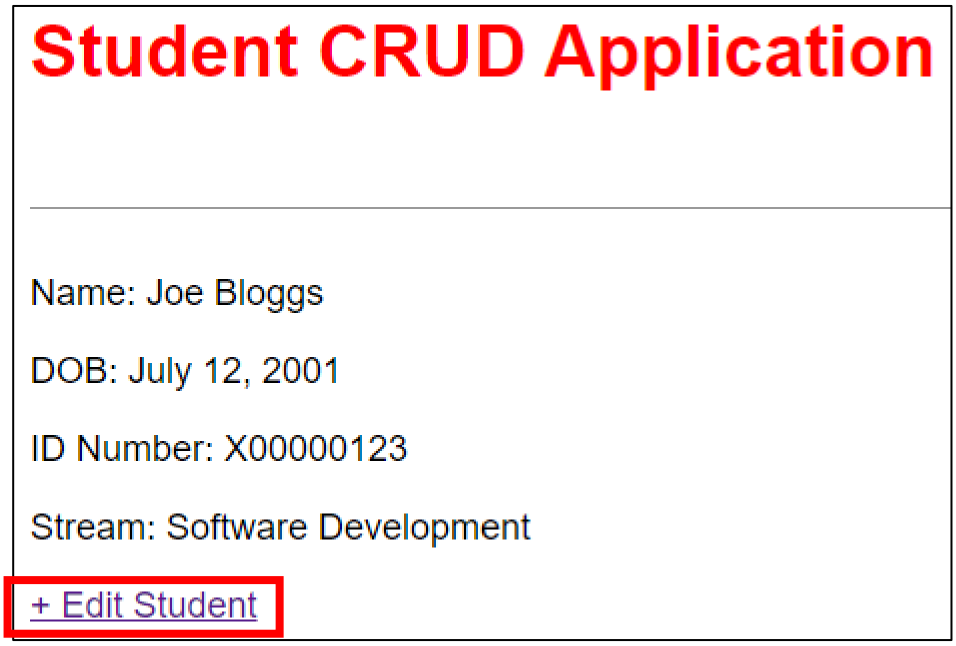
Code



templates > <> student_detail.html > ...

```
1   {% extends 'base.html' %}
2   {% block title %} Student Details Page {% endblock title %}
3
4   {%block content %}
5   <div class="student-entry">
6       <p>Name: {{ student.name }}</p>
7       <p>DOB: {{ student.date_birth }}</p>
8       <p>ID Number: {{ student.id_number }}</p>
9       <p>Stream: {{ student.stream }}</p>
10  </div>
11  <p><a href="{% url 'student_edit' student.pk %}">+ Edit Student</a></p>
12  {% endblock content %}
```

1. If you click on a student entry you will see the new Edit button.



Student CRUD Application

Name: Joe Bloggs

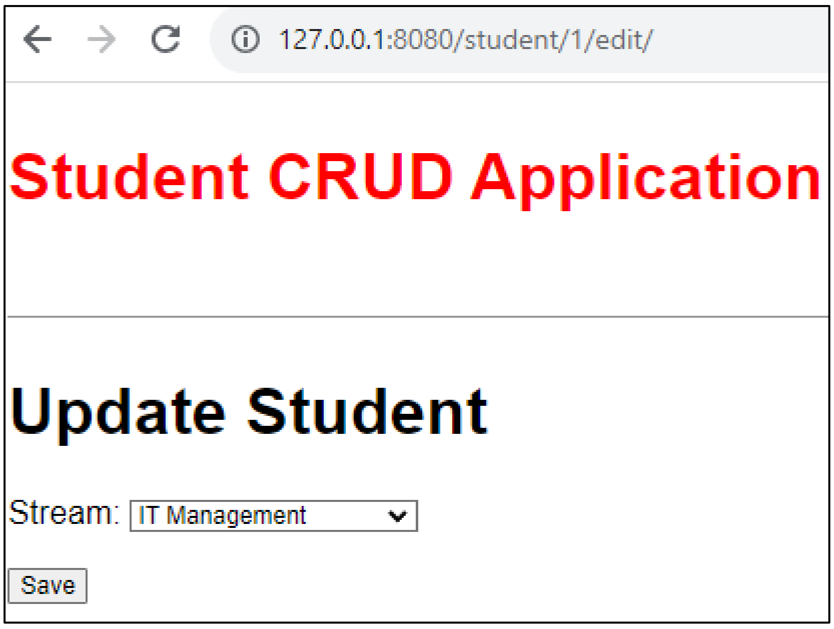
DOB: July 12, 2001

ID Number: X00000123

Stream: Software Development

+ Edit Student

2. Click on “+ Edit Student” and you will be redirected to the following URL http://127.0.0.1:8080/student/1/edit/ if you selected your first student post.

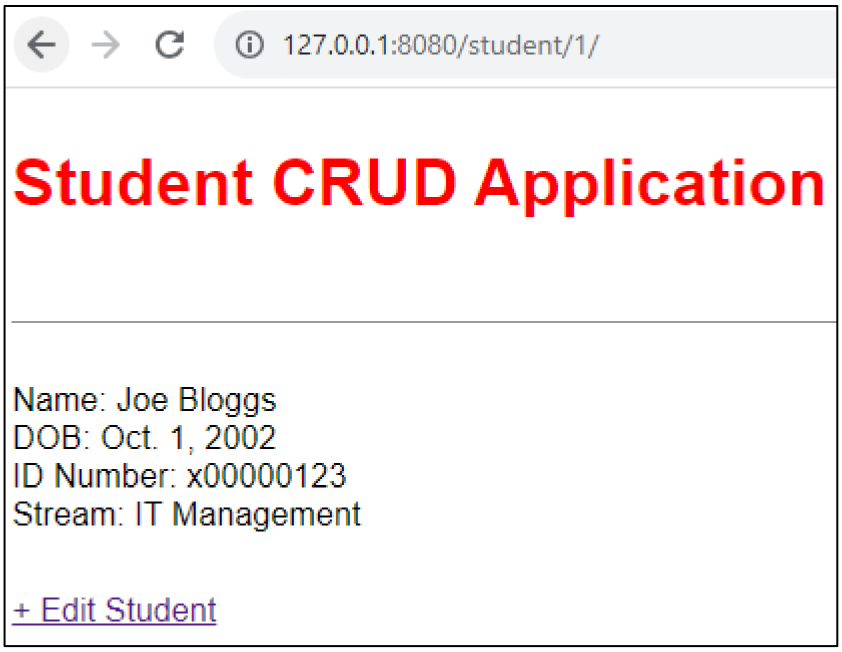

127.0.0.1:8080/student/1/edit/

Student CRUD Application

Update Student

Stream: IT Management

3. Change the choice of stream and click **Update.**
4. Navigate to the homepage and you can see that Joe Bloggs has changed to the IT Management stream.


127.0.0.1:8080/student/1/

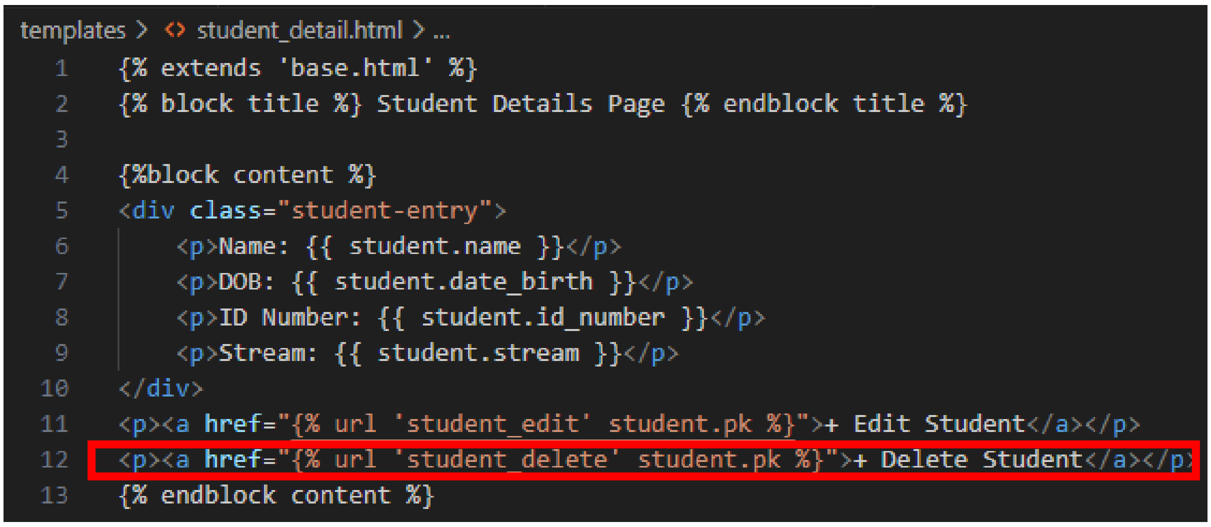
Student CRUD Application

Name: Joe Bloggs
DOB: Oct. 1, 2002
ID Number: x00000123
Stream: IT Management

Delete a Student

1. Add a new link (line 12) to **student_detail.html** to delete a student.

Code

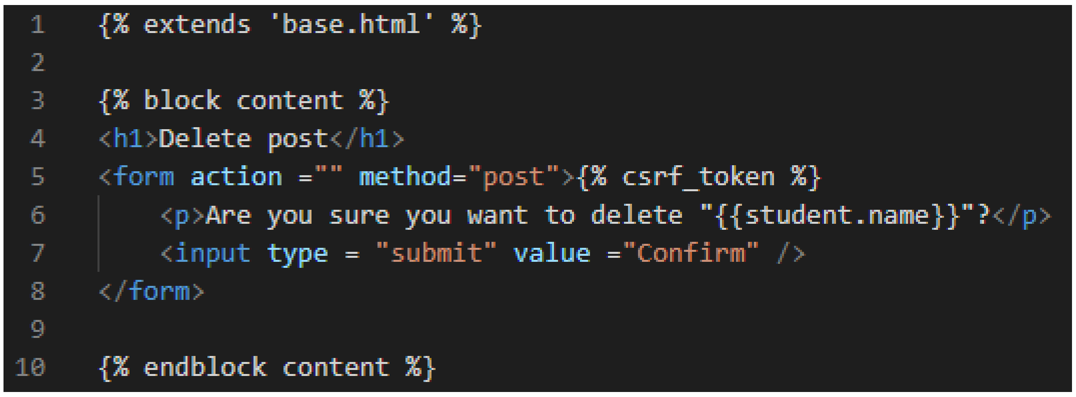


```
templates > student_detail.html > ...
 1  {% extends 'base.html' %}
 2  {% block title %} Student Details Page {% endblock title %}
 3
 4  {%block content %}
 5  <div class="student-entry">
 6      <p>Name: {{ student.name }}</p>
 7      <p>DOB: {{ student.date_birth }}</p>
 8      <p>ID Number: {{ student.id_number }}</p>
 9      <p>Stream: {{ student.stream }}</p>
10  </div>
11  <p><a href="{% url 'student_edit' student.pk %}">+ Edit Student</a></p>
12  <p><a href="{% url 'student_delete' student.pk %}">+ Delete Student</a></p
13  {% endblock content %}
```

Template

In VS Code, create a new template file called **student_delete.html** and enter the following code:

Code



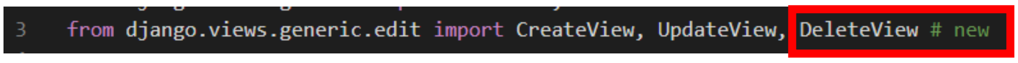
```
{% extends 'base.html' %}

{% block content %}
<h1>Delete post</h1>
<form action ="" method="post">{% csrf_token %}
    <p>Are you sure you want to delete "{{student.name}}"?</p>
    <input type = "submit" value ="Confirm" />
</form>

{% endblock content %}
```

View

Update the **views.py** file using the code provided below, by importing **DeleteView** and **reverse_lazy** at the top, then create a new view that subclasses **DeleteView**.

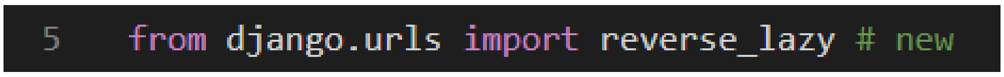
Code

Import **DeleteView**



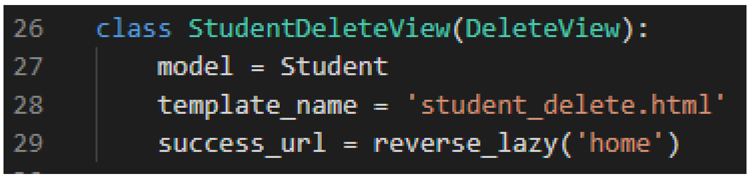
```python
from django.views.generic.edit import CreateView, UpdateView, DeleteView # new
```

Add the import **reverse_lazy**



```python
from django.urls import reverse_lazy # new
```

Add the new class



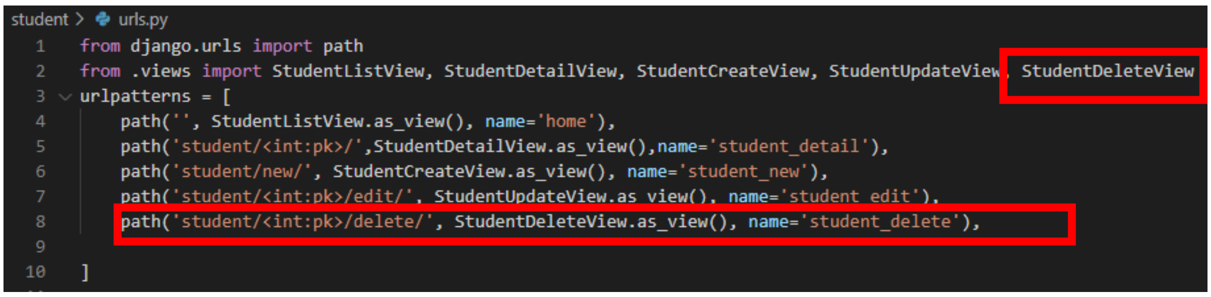
```python
class StudentDeleteView(DeleteView):
    model = Student
    template_name = 'student_delete.html'
    success_url = reverse_lazy('home')
```

We use **reverse_lazy** as opposed to just reverse so that Django won't execute the URL redirect until our view has finished deleting the student.

URLs

Open **student/urls.py** and add the following code to create a URL.

Code

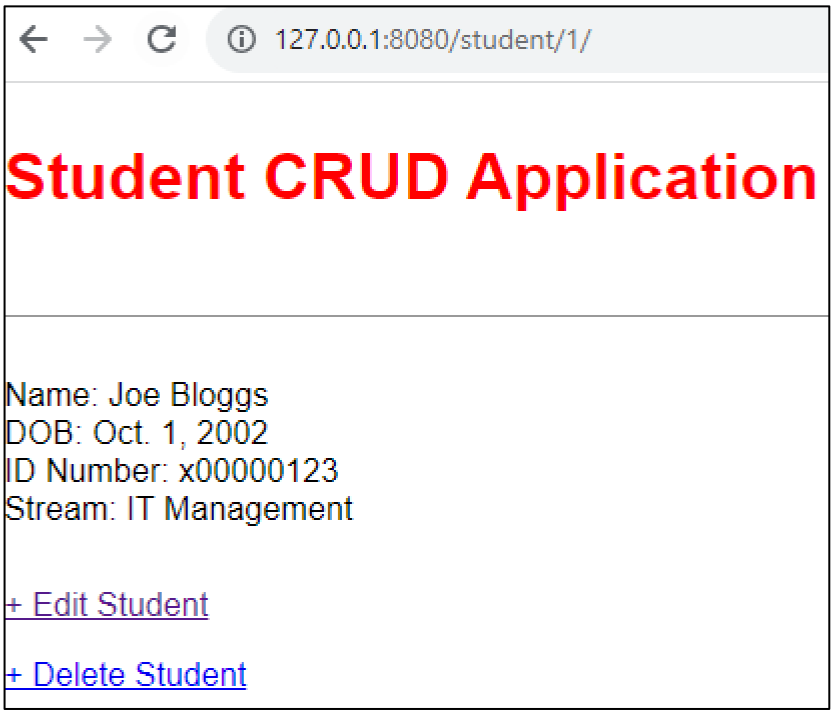


student > urls.py

```python
from django.urls import path
from .views import StudentListView, StudentDetailView, StudentCreateView, StudentUpdateView, StudentDeleteView
urlpatterns = [
    path('', StudentListView.as_view(), name='home'),
    path('student/<int:pk>/',StudentDetailView.as_view(),name='student_detail'),
    path('student/new/', StudentCreateView.as_view(), name='student_new'),
    path('student/<int:pk>/edit/', StudentUpdateView.as_view(), name='student_edit'),
    path('student/<int:pk>/delete/', StudentDeleteView.as_view(), name='student_delete'),

]
```

Run server

1. Start the server again using **python manage.py runserver 8080** and refresh the individual student page to see the "Delete Student" link.



127.0.0.1:8080/student/1/

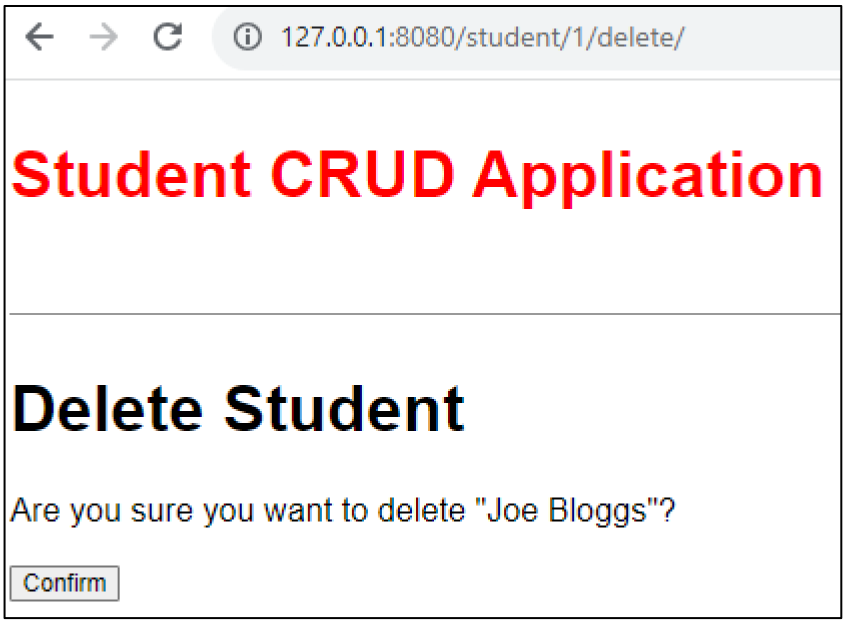
Student CRUD Application

Name: Joe Bloggs
DOB: Oct. 1, 2002
ID Number: x00000123
Stream: IT Management

+ Edit Student

+ Delete Student

2. Click the link and it takes you to the delete page for the student, which displays the name of the student.



127.0.0.1:8080/student/1/delete/

Student CRUD Application

Delete Student

Are you sure you want to delete "Joe Bloggs"?

3. Click on the “Confirm” button and you are redirected you to the homepage where the student has been deleted!

Upload

1. Open the **README.md** file in Vs Code and replace the contents with your name and id number.
2. Save a snapshot of the current project state with the following command:
   git add -A
3. Commit the changes along with a suitable message:
   git commit -m “lab 5 part 1 commit”
4. Update the remote repository with the local commits:
   git push -u origin main
5. Go to your GitHub page and refresh the page to see your local code now hosted online.
6. To deactivate the virtual environment type, deactivate as shown below:
   deactivate
7. To exit out of Windows Command Line type exit:
   exit