**Simple CRUD operation Using Django**

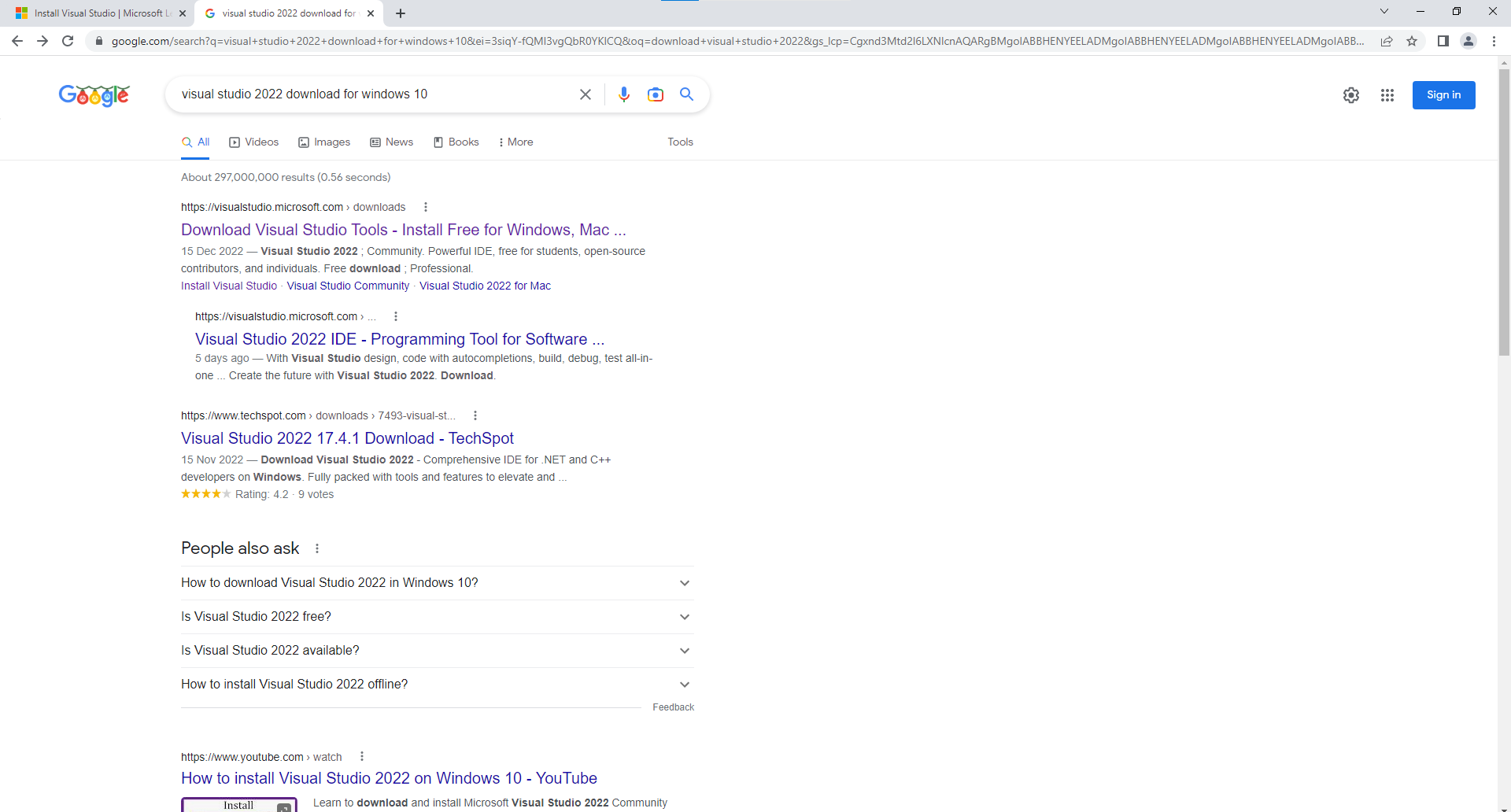
We should require the following tools:

* Install Visual Studio
* Install python
* Install Django

**How to install Visual Studio**

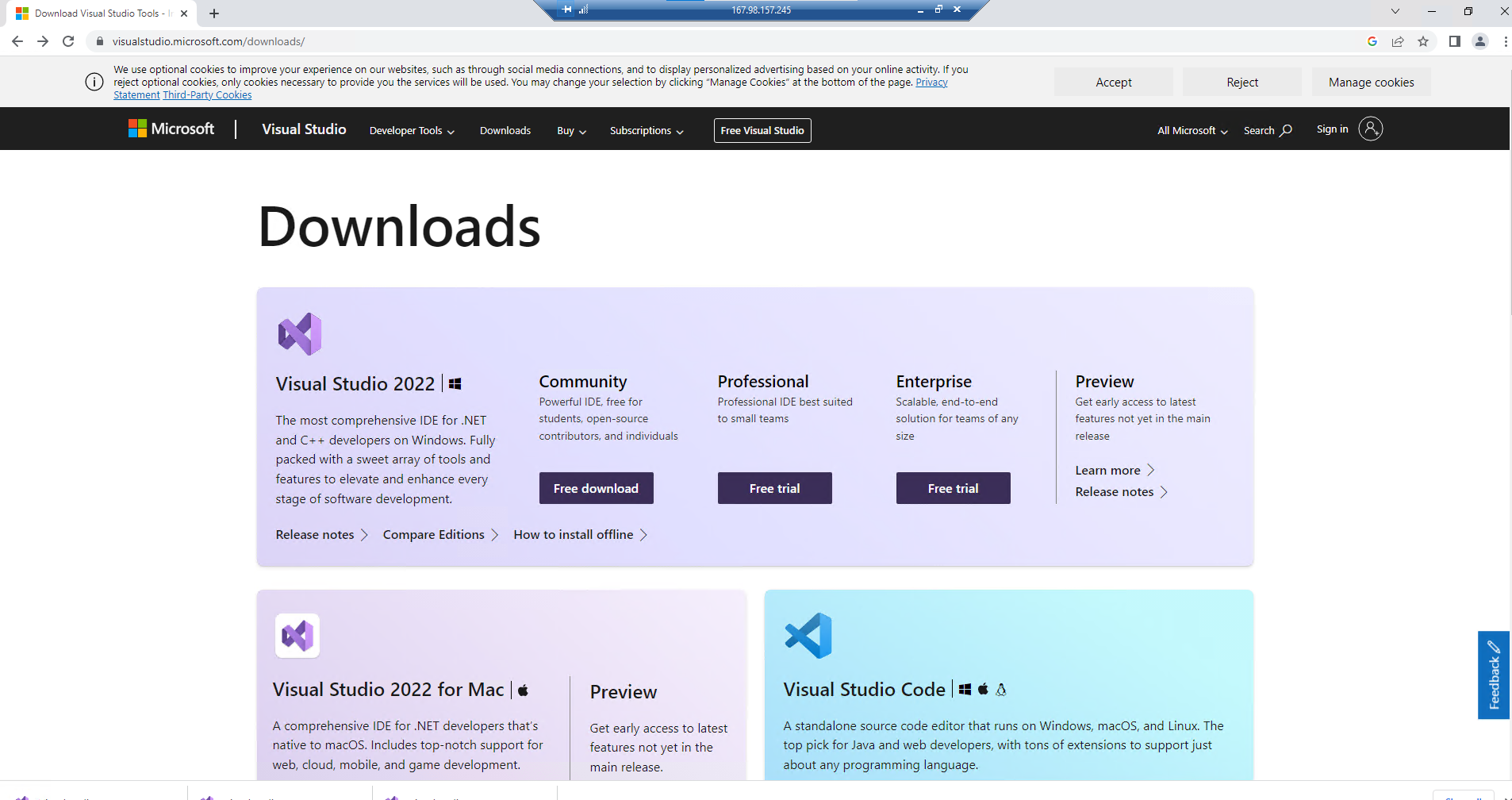
Step1: Download Visual Studio

Step2: Double click the downloaded file

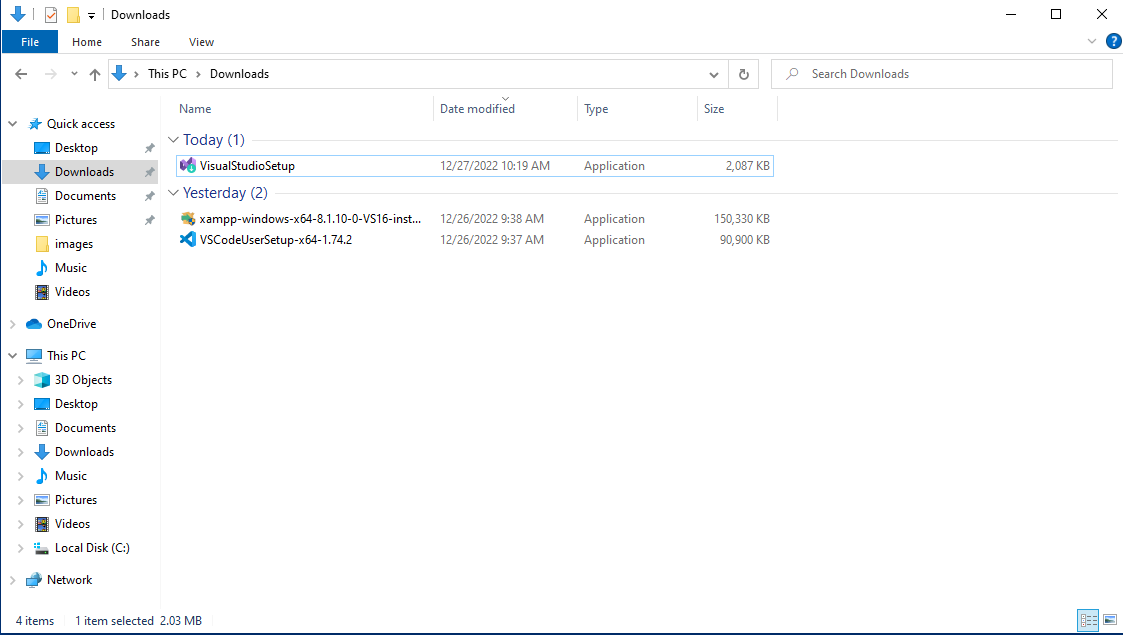


Click the first link

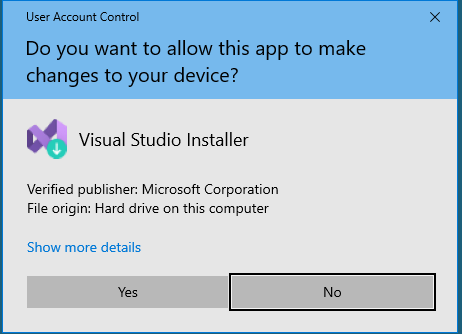
Step3: Click **free download** from **Community**.



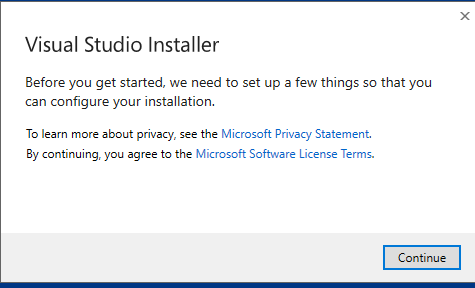
Step4: Double click the downloaded file from downloads.



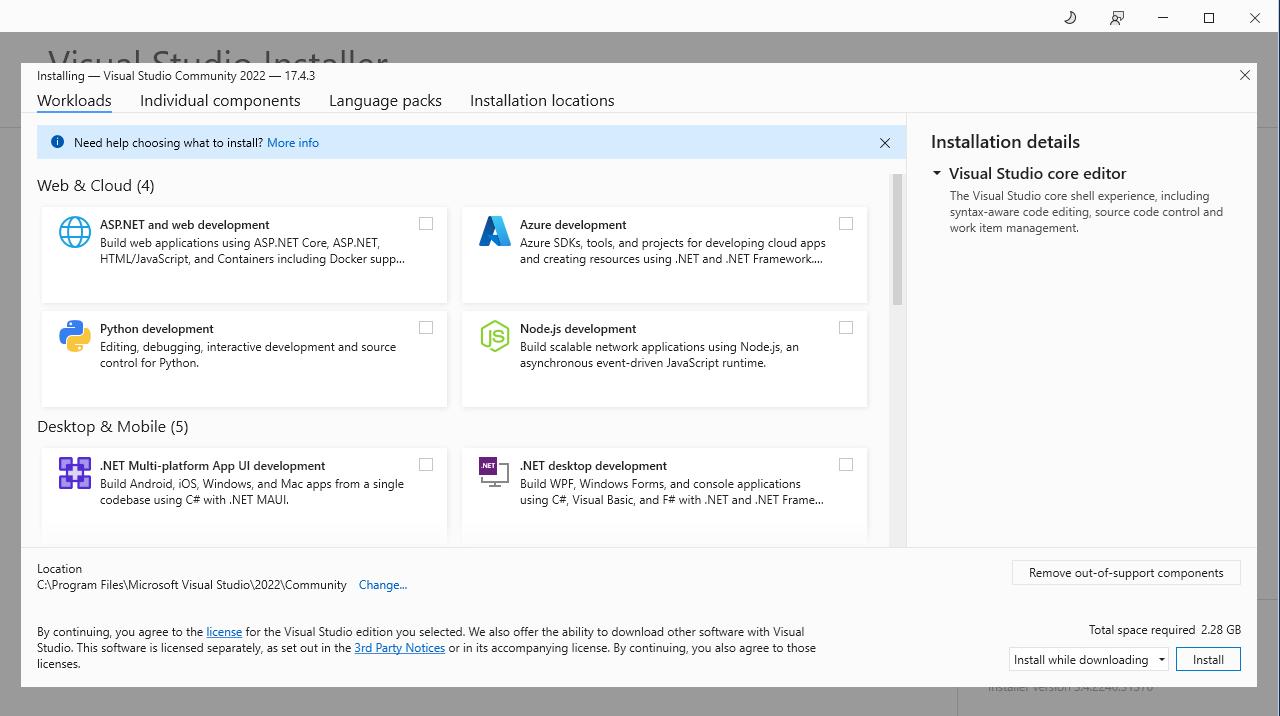
Step5: Then click ’yes’



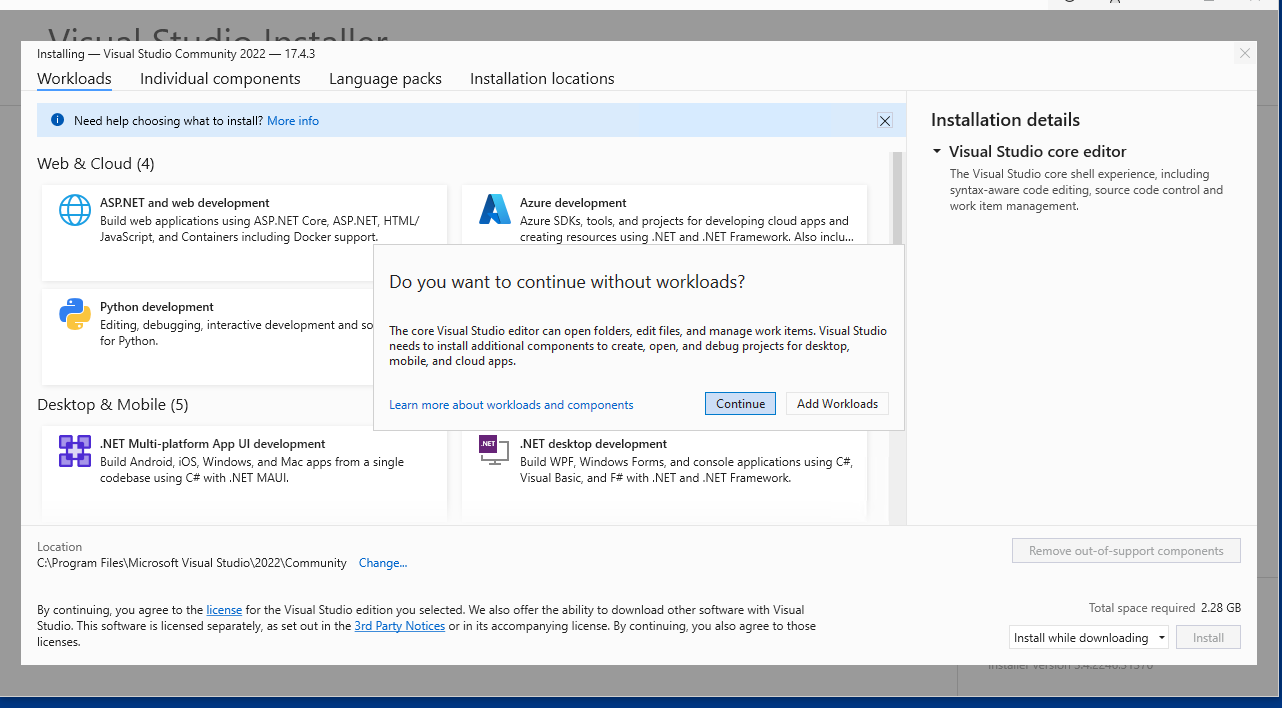
Step6: Click continue.



Step7: Click the install button



Step8: Click the continue button



Step9: Then click **start visual studio**

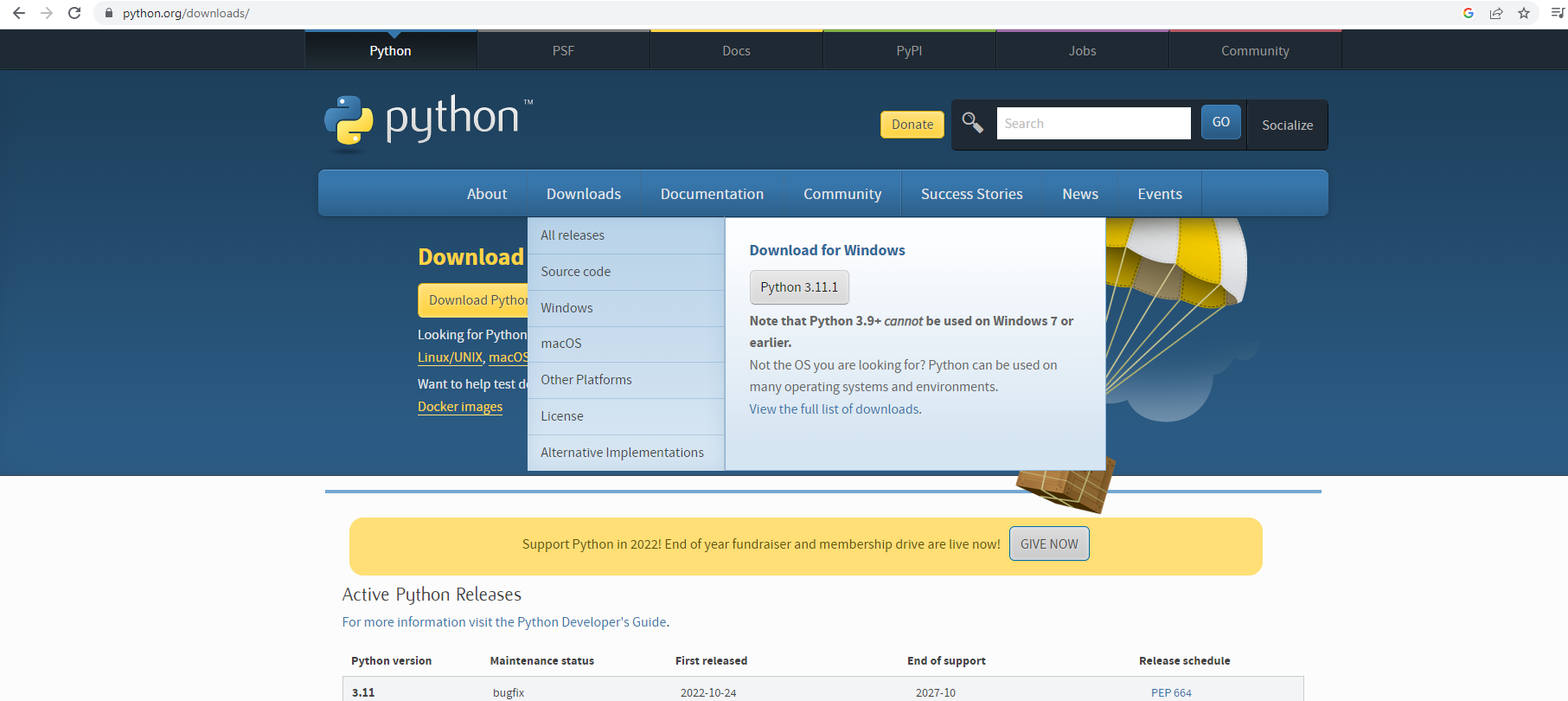


Now visual studio will appear in your screen.

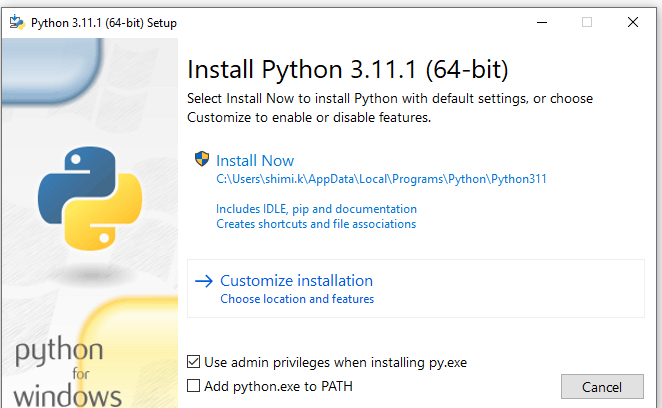
**How to install Python**

Use this link to download the new version of python

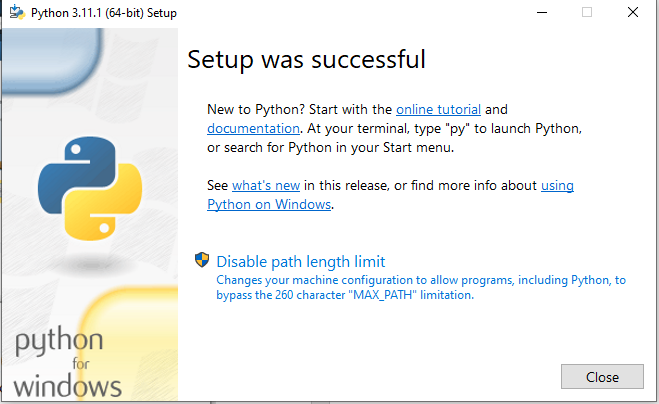
<https://www.python.org/downloads/>



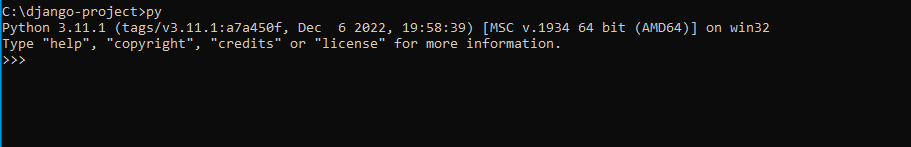
Double click the downloaded exe file



Make sure the "Add Python to PATH" checkbox is checked and click the’ install now’ option.

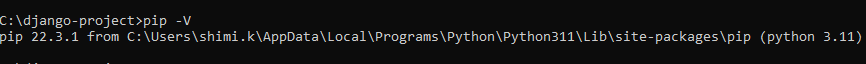


Now python installed in our system. Then open terminal type py to check the version of python.



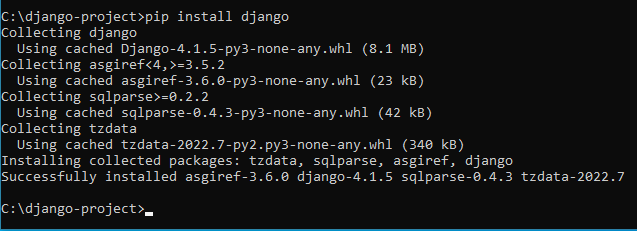
Next step is to check ‘pip’ is installed or not. Typically, pip is installed automatically together with Python. We can check that by using following command.

**pip -V**

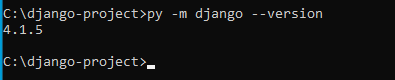


Next step is to install Django.

**pip install django** command is used to install django.



To check the version of Django using the following command.



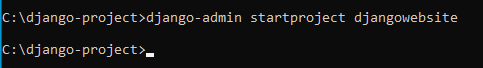
Then create a folder ‘djangoproject’ and go to inside of that folder in command line prompt.

**cd djangoproject**

The django-admin tool can then be used to create a new skeleton site called "djangoproject."

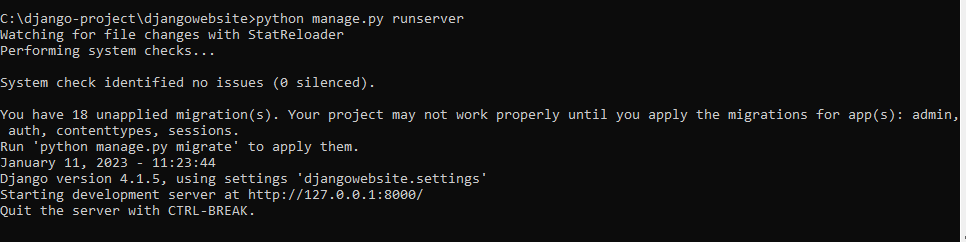
**django-admin startproject mytestsite**

**cd mytestsite**

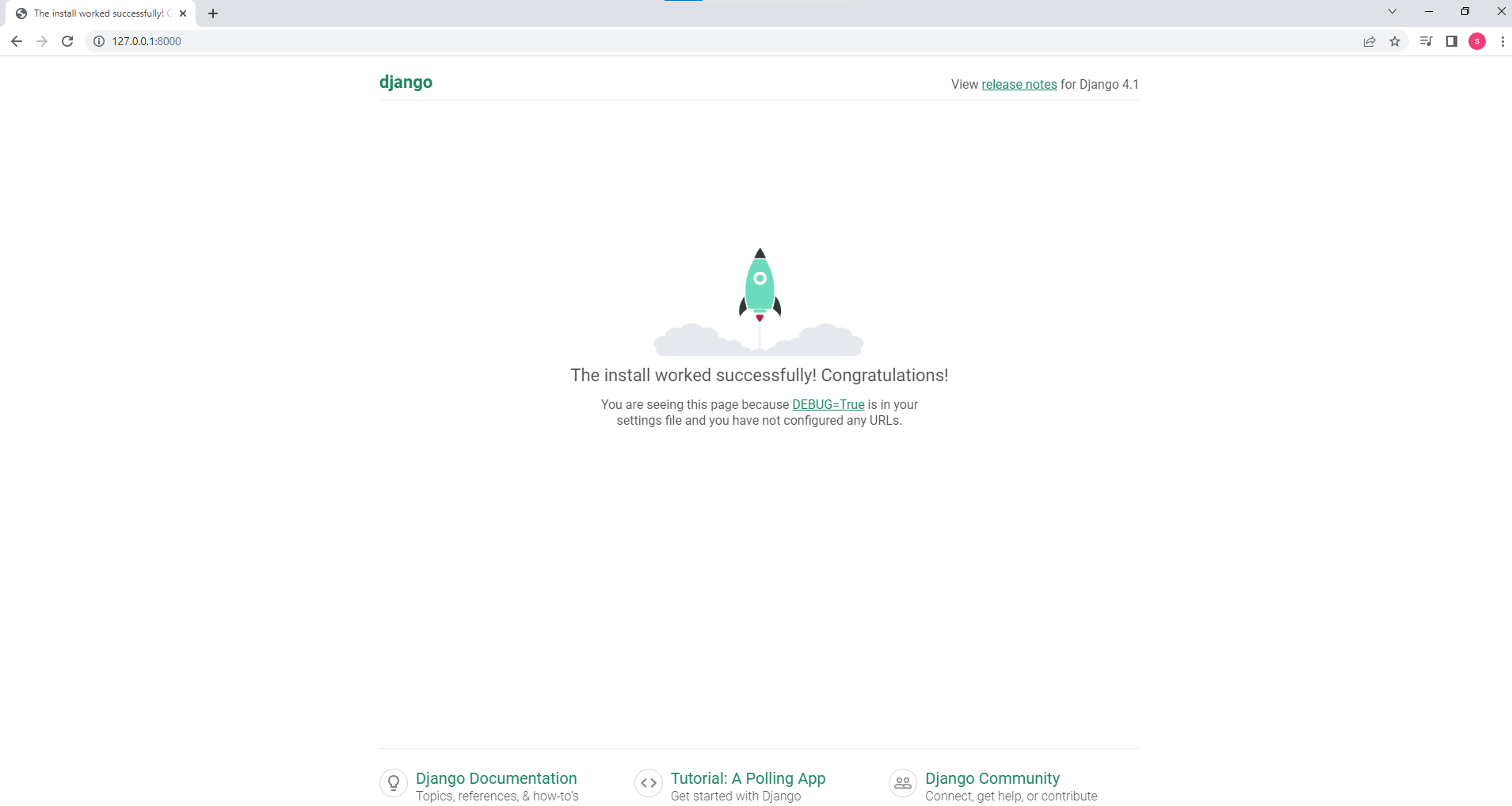


Using manage.py and the runserver command, we can run the development web server from within this folder, as demonstrated.

**python manage.py runserver**



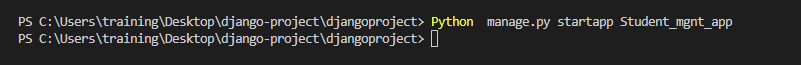
When the server is up and running, you can access the website by going to http://127.0.0.1:8000/ on your local web browser. You should visit a website with the following design:



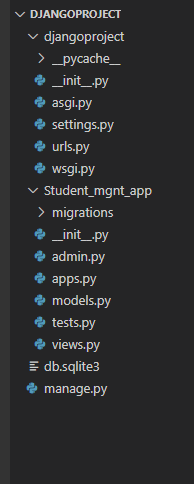
Next step is to create an app using the following commands

Python manage.py startapp <app\_name>

Here we are creating ‘student\_mgnt\_app’



Then open our project in visual studio. Our project structure should be like as follows:



Python treats directories containing it as modules by virtue of the \_\_init .py file.

both wsgi.py and asgi.py.

The acronyms WSGI and ASGI stand for Web Server Gateway Interface and Asynchronous Server Gateway Interface, respectively. Between the web server and a Python web application or framework, they both specify the interface.

A crucial file in Django projects is settings.py. It contains all configuration setting required for your web app to function.

What is urls py in Django?

The "urlpatterns" tuple is the most crucial element in the url.py file. Here, you specify how URLs and views are mapped to one another.

What is manage py in Django?

Django's Manage.py command-line tool functions similarly to the django-admin command. The distinction is that it makes a reference to the project's settings. "py file"

Next step is to mention our app name in setting.py file.

**Settings.py**

INSTALLED\_APPS = [

.

..

Student\_mgnt\_app’

]

Then we should create a file named urls.py inside of student\_mgnt\_app and update urls.py inside of our main project djangoproject.

**urls.py**

from django.contrib import admin

from django.urls import path,include

urlpatterns = [

    path('admin/', admin.site.urls),

    path('',include('Student\_mgnt\_app.urls')),

]

Add the following in new created file urls.py.

**Student\_mgnt\_app/urls.py**

from django.contrib import admin

from django.urls import path,include

from . import views

urlpatterns = [

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

]

Then add the following function inside of student\_mgnt\_app/views.py.

**views.py**

from django.shortcuts import render

# Create your views here.

def index(request):

    return render(request,'index.html')

Create a folder named **templates** inside of student\_mgnt\_app and inside of **templates** folder create a file **index.html**.

Next step is to create a new model named **Student** and add the following code in models.py.

Models.py

from django.db import models

# Create your models here.

class Student(models.Model):

    StudentId=models.AutoField(primary\_key=True)

    StudentName=models.CharField(max\_length=100,null=False)

    Dob=models.DateField()

    City=models.CharField(max\_length=100,null=False)

    Qualification=models.CharField(max\_length=100,null=False)

    Email=models.CharField(max\_length=100,null=False)

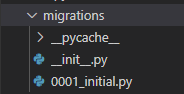
    Phone=models.IntegerField(default=0)

To really build the table in the database after describing a Model in the models.py file, we must run a command.

**python manage.py makemigrations**



This will create the 0001\_initial.py in migrations folder.



**0001\_initial.py**

 migrations.CreateModel(

            name='Student',

            fields=[

                ('StudentId', models.AutoField(primary\_key=True, serialize=False)),

                ('StudentName', models.CharField(max\_length=100)),

                ('Dob', models.DateField()),

                ('City', models.CharField(max\_length=100)),

                ('Qualification', models.CharField(max\_length=100)),

                ('Email', models.CharField(max\_length=100)),

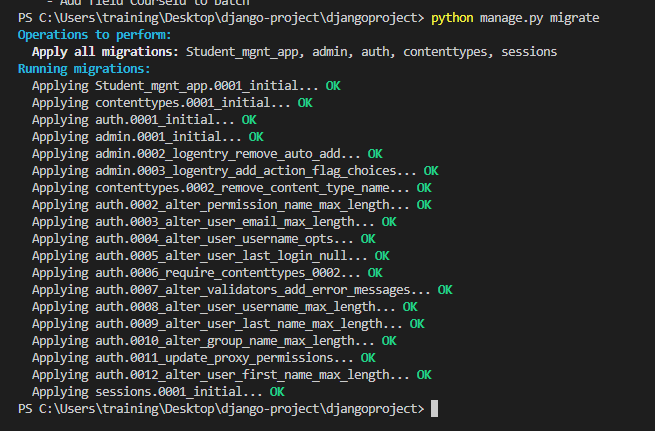
                ('Phone', models.IntegerField(default=0)),

            ],

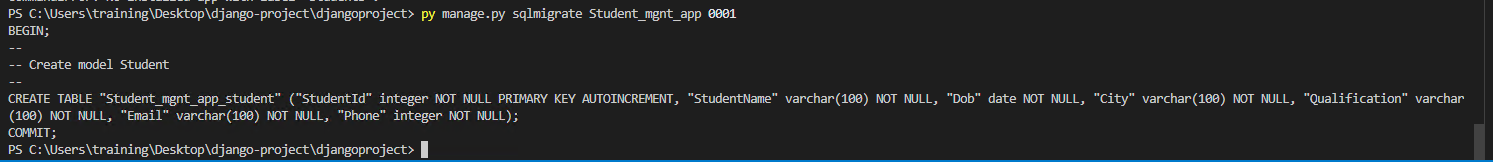
        ),

There is still one more command you must run before Django creates and executes a SQL statement based on the contents of the newly produced file in the /migrations/ folder.

py manage.py migrate



You can see the SQL query that was executed as a result of the aforementioned migration. You only need to execute the following command while using the migration number.



Next we should register the model Student inside of admin.py

**admin.py**

# Register your models here.

admin.site.register(Student)

We can use bootstrap in our project.

A free and open source front-end development framework called Bootstrap is used to build websites and web applications. Before using bootstrap, we should create a template folder inside of our app.

**templates/index.html**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta http-equiv="X-UA-Compatible" content="IE=edge">

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

<link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/css/bootstrap.min.css">

<script src="https://code.jquery.com/jquery-3.5.1.min.js"></script>

<script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.min.js"></script>

<script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/js/bootstrap.min.js"></script>

    <title>Document</title>

</head>

<body>

  <nav class="navbar navbar-expand-lg " style="background-color:rgba(68,49,141,.89) ;">

    <div class="container-fluid">

     <a class="navbar-brand" href="/" style="color:white;font-size: 22px;">The Knowledge Academy </a>

      <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarSupportedContent" aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle navigation">

        <span class="navbar-toggler-icon"></span>

      </button>

      <div class="collapse navbar-collapse" id="navbarSupportedContent" style="padding-left:100px;">

        <ul class="navbar-nav me-auto mb-2 mb-lg-0"   >

          <li class="nav-item">

            <a class="nav-link" aria-current="page" href="/" style="color:white; font-size: 18px;">Home</a>

          </li>

          <li class="nav-item">

            <a class="nav-link active" href="/student\_details" style="color:white ;font-size: 18px;">Student Details</a>

          </li>

    </div>

  </nav>

      <div class="container">

       <h1 style="color:rgba(68,49,141,.89) ;text-align: center;" >Student Management system</h1>

      <div id="carouselExampleIndicators" class="carousel slide" data-ride="carousel">

        <ol class="carousel-indicators">

          <li data-target="#carouselExampleIndicators" data-slide-to="0" class="active"></li>

          <li data-target="#carouselExampleIndicators" data-slide-to="1"></li>

          <li data-target="#carouselExampleIndicators" data-slide-to="2"></li>

        </ol>

        <div class="carousel-inner">

          <div class="carousel-item active">

            {% load static %}

            <img class="d-block w-100" src="{% static 'itil.jpg' %}" alt="First slide" style="width: 100%;height: 800px;">

          </div>

          <div class="carousel-item">

            <img class="d-block w-100" src="{%  static 'crm\_training.jpg' %}" alt="Second slide" style="width: 100%;height: 800px;">

          </div>

          <div class="carousel-item">

            <img class="d-block w-100" src="{%  static 'img1.png' %}" alt="Third slide" style="width: 100%;height: 800px;">

          </div>

        </div>

        <a class="carousel-control-prev" href="#carouselExampleIndicators" role="button" data-slide="prev">

          <span class="carousel-control-prev-icon" aria-hidden="true"></span>

          <span class="sr-only">Previous</span>

        </a>

        <a class="carousel-control-next" href="#carouselExampleIndicators" role="button" data-slide="next">

          <span class="carousel-control-next-icon" aria-hidden="true"></span>

          <span class="sr-only">Next</span>

        </a>

      </div>

    </div>

</body>

</html>

* We used bootstrap navbar and carousel in our index page.
* We can add image in django using the following steps:

1. Go to the web app folder Student\_mgnt\_app and open the urls.py folder.
2. If it isn't already there, add the phrase to staticfiles urlpatterns.

**from django. contrib. staticfiles. Urls import staticfiles\_urlpatterns**

1. If it isn't already there, add this line to the end of the same urls.py file.

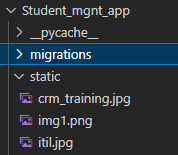
**urlpatterns += staticfiles\_urlpatterns()**

1. urls.py

from django.contrib.staticfiles.urls import staticfiles\_urlpatterns

urlpatterns += staticfiles\_urlpatterns()

1. Make a folder for images in the static folder static of the web application.
2. Add images inside of static folder (for example Student\_mgnt\_app/static/img.jpg)



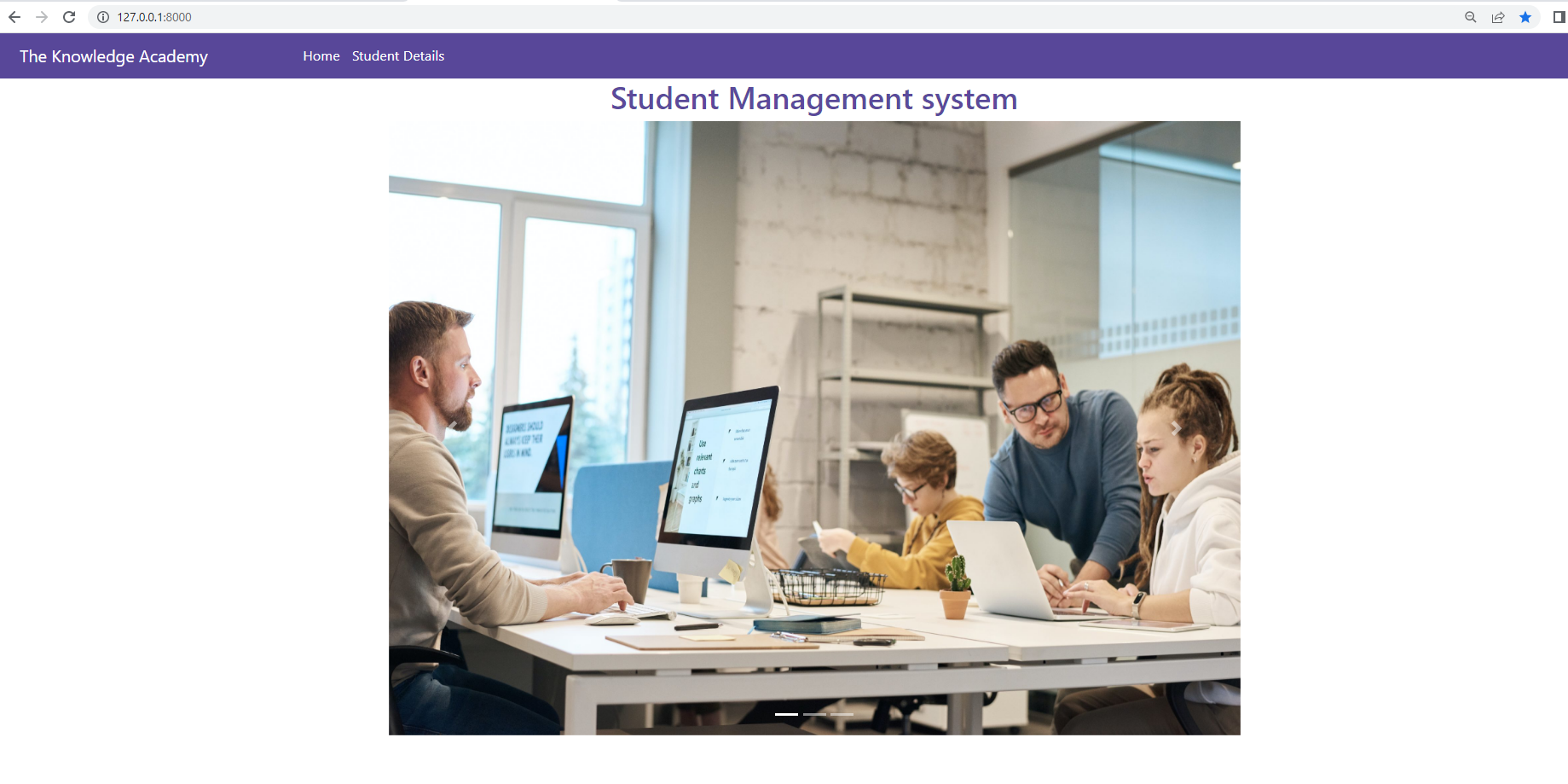
1. To include the image file in an HTML template, open the template and add the HTML img tag along with placeholders for "% load static%" and "% static "image file url>>"%", as shown below.

**Student\_mgnt\_app/templates/index.html**

 {% load static %}

            <img class="d-block w-100" src="{% static 'itil.jpg' %}" alt="First slide" style="width: 100%;height: 800px;">

1. The output will be like as follows:



**student\_details.html**

<div class="container-xl">

    <div class="table-responsive">

        <div class="table-wrapper">

            <div class="table-title" style="background-color:rgba(68,49,141,.89) ;">

                <div class="row">

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

                        <h2 ><b>Manage Student</b></h2>

                    </div>

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

                        <a href="#addEmployeeModal" class="btn btn-success" data-toggle="modal" style="background-color:darkgoldenrod ;font-size: 15px;border: none;"><i class="material-icons">&#xE147;</i> <span >Add New Student</span></a>

                    </div>

                </div>

            </div>

     <table class="table table-striped table-hover">

                <thead>

                    <tr>

                        <th>

                        </th>

                        <th>Student Name</th>

                        <th>Date of Birth</th>

                        <th>City</th>

                        <th>Qualification</th>

                        <th>Email</th>

                        <th>Phone</th>

                        <th>Actions</th>

                    </tr>

                </thead>

                <tbody>

                    {% for stud in std %}

                    <tr>

                        <td>

                        </td>

                        <td>{{stud.StudentName}}</td>

                        <td>{{stud.Dob}}</td>

                        <td>{{stud.City}}</td>

                        <td>{{stud.Qualification}}</td>

                        <td>{{stud.Email}}</td>

                        <td>{{stud.Phone}}</td>

                        <td>

                            <a href="edit\_student/{{stud.StudentId}}"  class="edit" ><i class="material-icons" data-toggle="tooltip" title="Edit">&#xE254;</i></a>

                            <a href="delete\_student/{{stud.StudentId}}" class="delete" ><i class="material-icons" data-toggle="tooltip" title="Delete">&#xE872;</i></a>

                                </td>

                    </tr>

                    {% endfor %}

                </tbody>

            </table>

        </div>

    </div>

</div>

<!-- Add Modal HTML -->

<div id="addEmployeeModal" class="modal fade"  >

    <div class="modal-dialog">

        <div class="modal-content" style="color:rgba(68,49,141,.89);font-size: 12px; ">

            <form action="/add\_student" method="post">

                {% csrf\_token %}

                <div class="modal-header">

                    <h4 class="modal-title">Add A Student</h4>

                    <button type="button" class="close" data-dismiss="modal" aria-hidden="true">&times;</button>

                </div>

                <div class="modal-body" >

                    <div class="form-group">

                        <label>Student Name</label>

                        <input type="text" class="form-control" required name="StudentName">

                    </div>

                    <div class="form-group">

                        <label>Date of Birth</label>

                        <input type="date" class="form-control" required name=" Dob">

                    </div>

                    <div class="form-group">

                        <label>City</label>

                        <input type="text" class="form-control" required name="City">

                    </div>

                    <div class="form-group">

                        <label>Qualification</label>

                        <input type="text" class="form-control" required name="Qualification" id="Qualification">

                    </div>

                    <div class="form-group">

                        <label>Email id</label>

                        <input type="text" class="form-control" required name="Email">

                    </div>

                    <div class="form-group">

                        <label>Phone Number</label>

                        <input type="text" class="form-control" required name="Phone">

                    </div>

                </div>

                <div class="modal-footer">

                    <input type="button" class="btn btn-default" data-dismiss="modal" value="Cancel">

                    <input type="submit"  style="font-size: 15px; background-color:rgba(68,49,141,.89) ;width: 100px; height: 40px; color: white;border: none;" value="Add">

                </div>

            </form>

        </div>

    </div>

</div>

**edit\_student.html**

<form method="POST" action="updates\_student/{{studs.StudentId}}">

        {% csrf\_token %}

       <div class="container">

        <div class="modal-header">

            <h4 class="modal-title">Edit Student</h4>

        </div>

        <div class="modal-body">

            <div class="form-group">

                <label>Student Name</label>

                <input type="text" class="form-control" required name="StudentName" value="{{studs.StudentName}}" >

            </div>

            <div class="form-group">

                <label>City</label>

                <input type="text" class="form-control" required name="City" value="{{studs.City}}" >

            </div>

            <div class="form-group">

                <label>Qualification</label>

                <input type="text" class="form-control" required name="Qualification" id="Qualification" value="{{studs.Qualification}}">

            </div>

            <div class="form-group">

                <label>Email id</label>

                <input type="text" class="form-control" required name="Email" value="{{studs.Email}}" >

            </div>

            <div class="form-group">

                <label>Phone Number</label>

                <input type="text" class="form-control" required name="Phone" value="{{studs.Phone}}">

            </div>

        </div>

        <div class="modal-footer">

            <input type="submit" style="background-color: rgba(68,49,141,.89); width: 130px;height: 60px; color: white; border:none" value="UPDATE">

        </div> </div>

    </form>

Our urls.py will be look like as follows:

**urls.py**

from django.contrib import admin

from django.urls import path,include

from . import views

from django.contrib.staticfiles.urls import staticfiles\_urlpatterns

urlpatterns = [

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

    path('student\_details',views.student\_details,name = 'student\_details'),

    path('add\_student',views.add\_student,name = 'add\_student'),

    path('edit\_student/<int:id>',views.edit\_student,name = 'edit\_student'),

    path('edit\_student/updates\_student/<int:id>',views.updates\_student,name = 'updates\_student'),

    path('delete\_student/<int:id>',views.delete\_student,name = 'delete\_student'),

]

urlpatterns += staticfiles\_urlpatterns()

from django.shortcuts import render,HttpResponse, HttpResponseRedirect

from .models import Student

from django.urls import reverse

from django.template import loader

# Create your views here.

def index(request):

    return render(request,'index.html')

def student\_details(request):

    return render(request,'student\_details.html')

def add\_student(request):

    if request.method == 'POST':

        StudentName=request.POST['StudentName']

        Dob=request.POST[' Dob']

        City=request.POST['City']

        Qualification=request.POST['Qualification']

        Phone=int(request.POST['Phone'])

        Email=(request.POST['Email'])

        new\_stud=Student(StudentName=StudentName,Dob=Dob,City=City,Qualification=Qualification,Phone= Phone,Email= Email)

        new\_stud.save()

        return HttpResponseRedirect(reverse('student\_details'))

def student\_details(request):

    std=Student.objects.all()

    context={

        'std':std

    }

    print(context)

    return render(request,'student\_details.html',context)

def edit\_student(request,id):

    studs= Student.objects.get(StudentId=id)

    return render(request,'edit\_student.html', {'studs':studs})

def updates\_student(request,id):

    stud = Student.objects.get(StudentId=id)

    stud.StudentName=request.POST['StudentName']

    stud.City=request.POST['City']

    stud.Qualification=request.POST['Qualification']

    stud.Phone=int(request.POST['Phone'])

    stud.Email=request.POST['Email']

    stud.save()

    return HttpResponseRedirect(reverse('student\_details'))

def delete\_student(request, id):

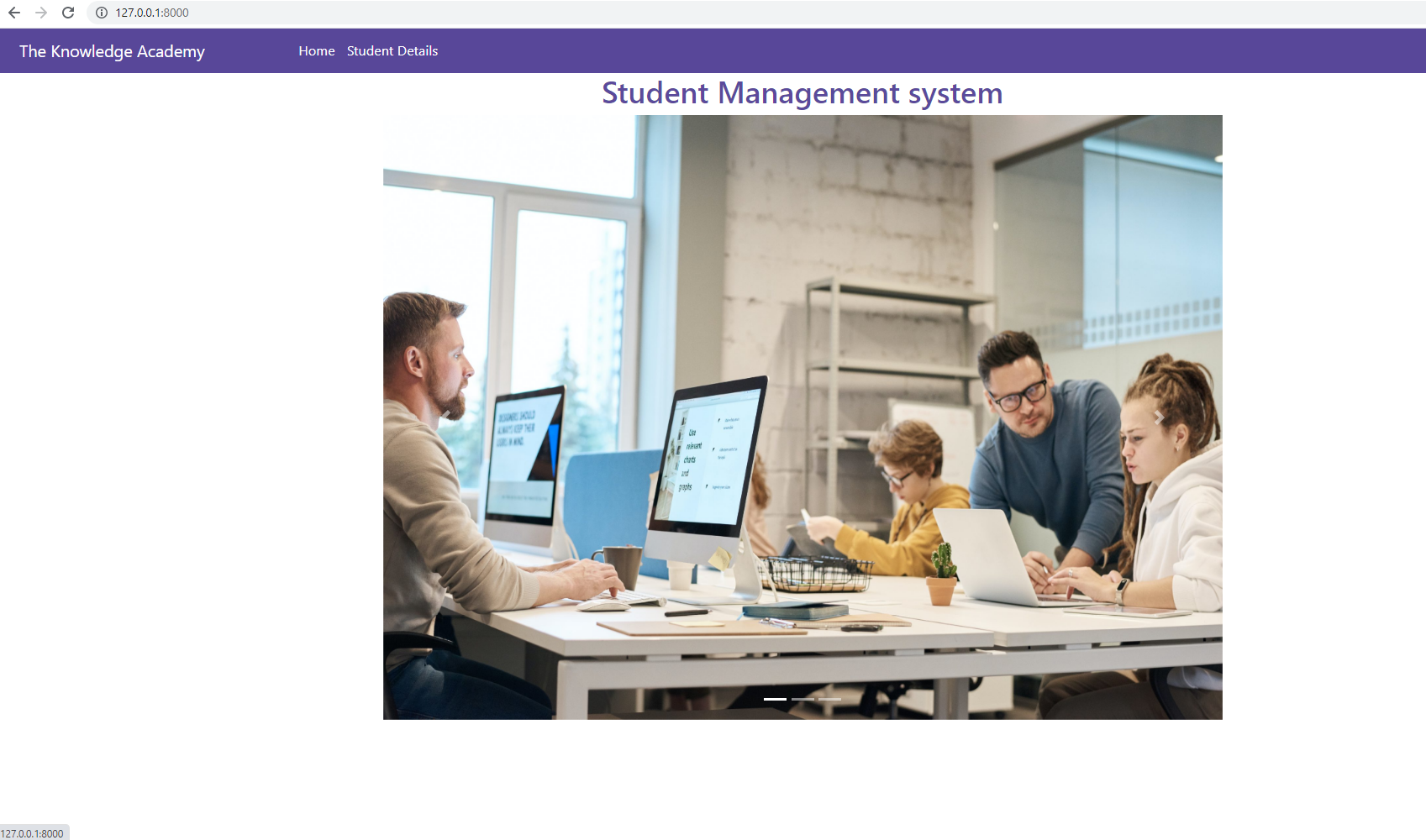
    member = Student.objects.get(StudentId=id)

    member.delete()

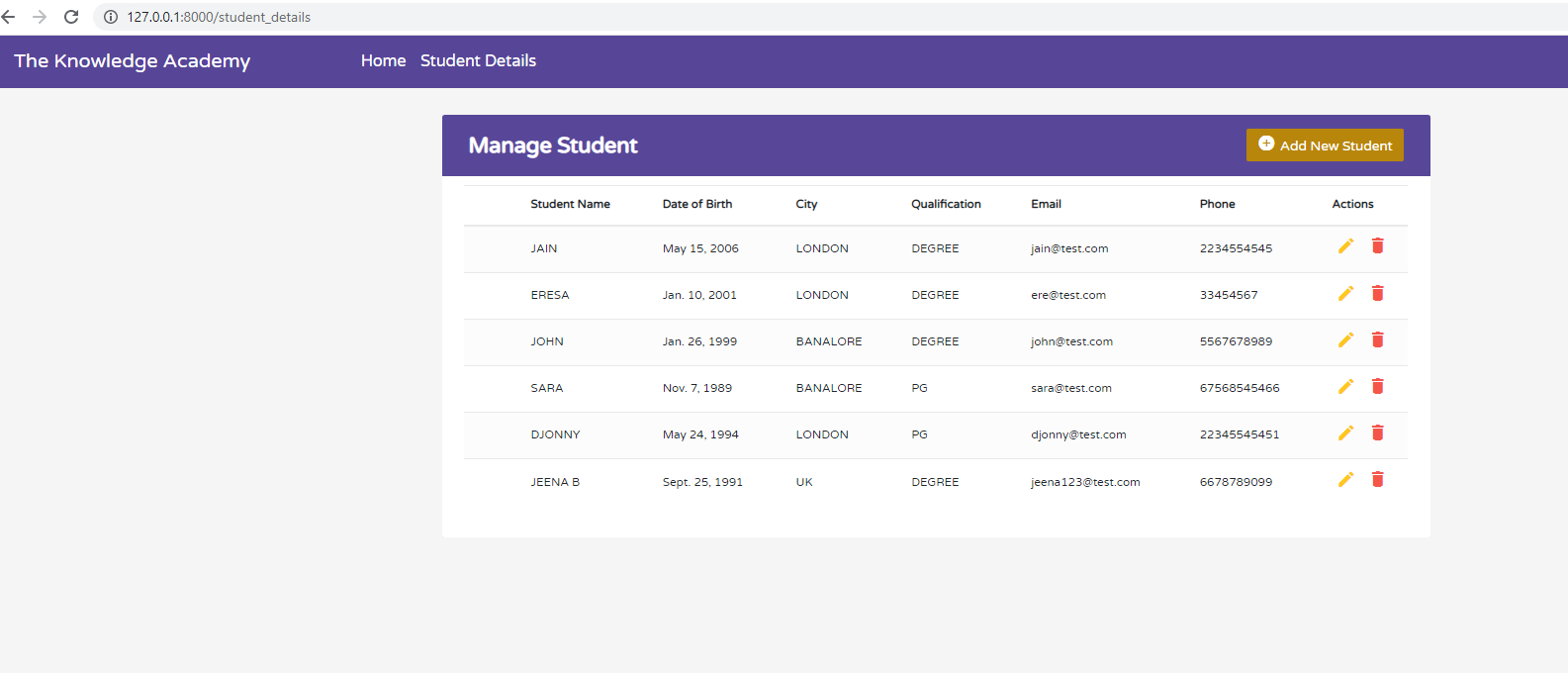
    return HttpResponseRedirect(reverse('student\_details'))

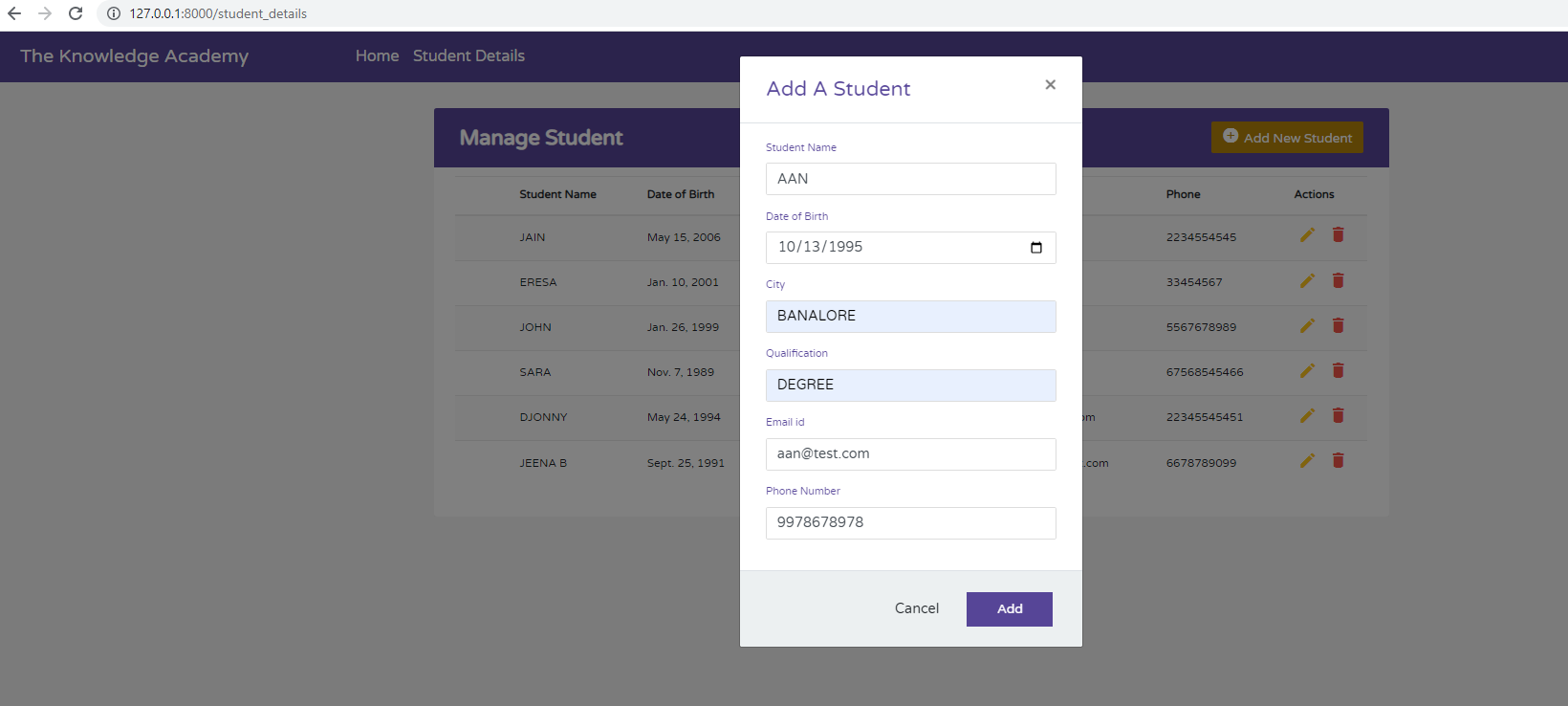
The final output will be like as follows:

**Home page**

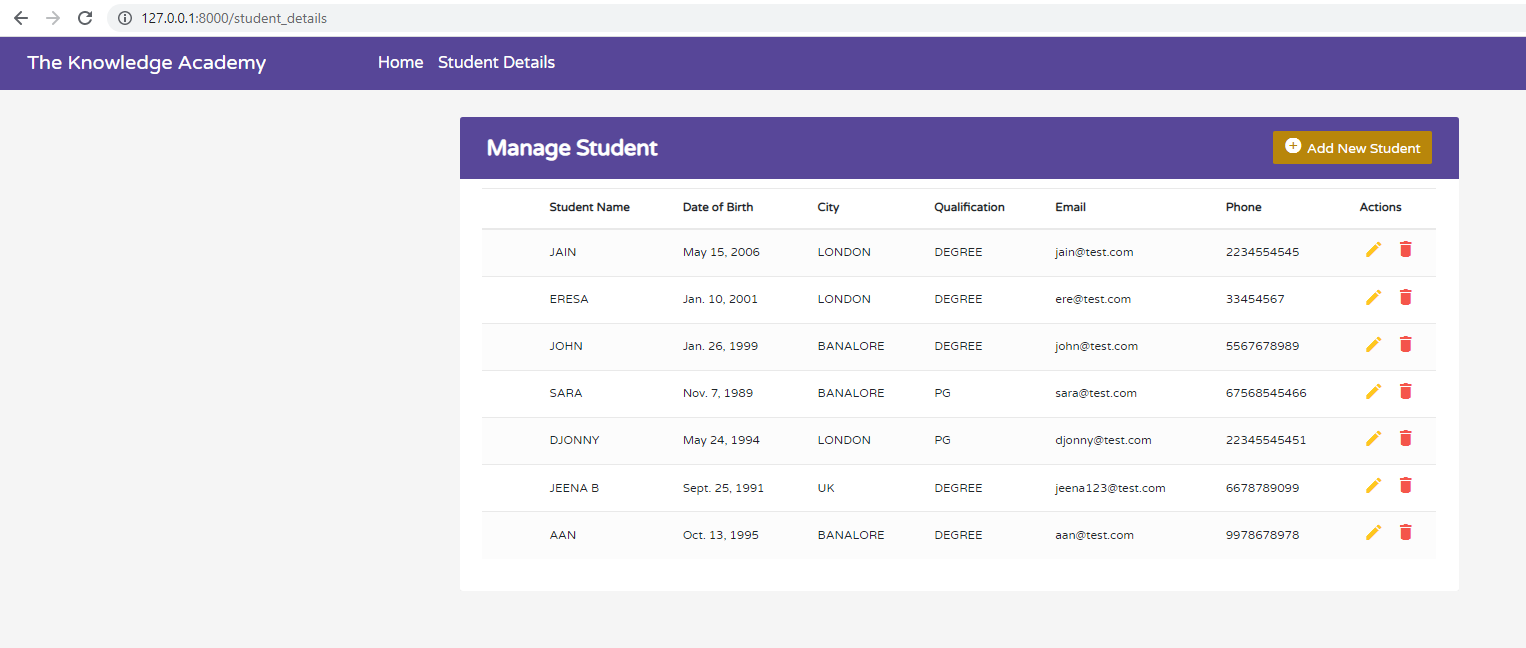


**Student details page**





After clicking add button we can see the new student details in our **student\_details page**

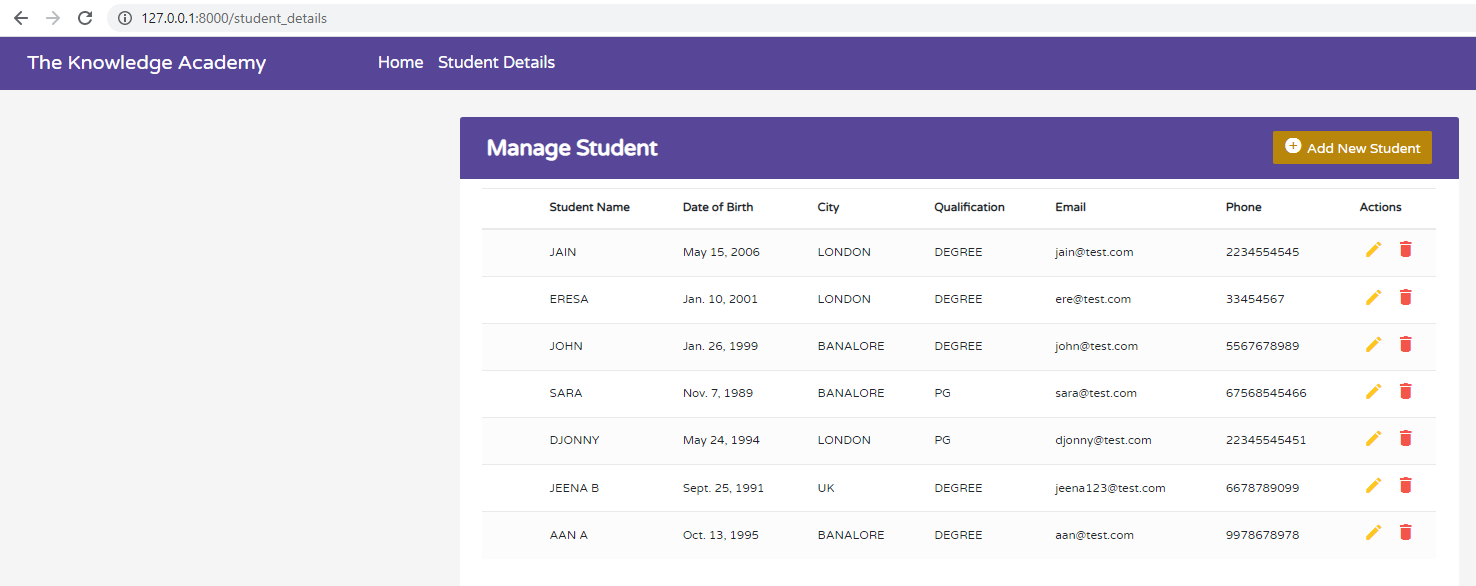
****

If we want to update the student name ANN to ANN A. We can click the edit option button.

**Edit\_student page**

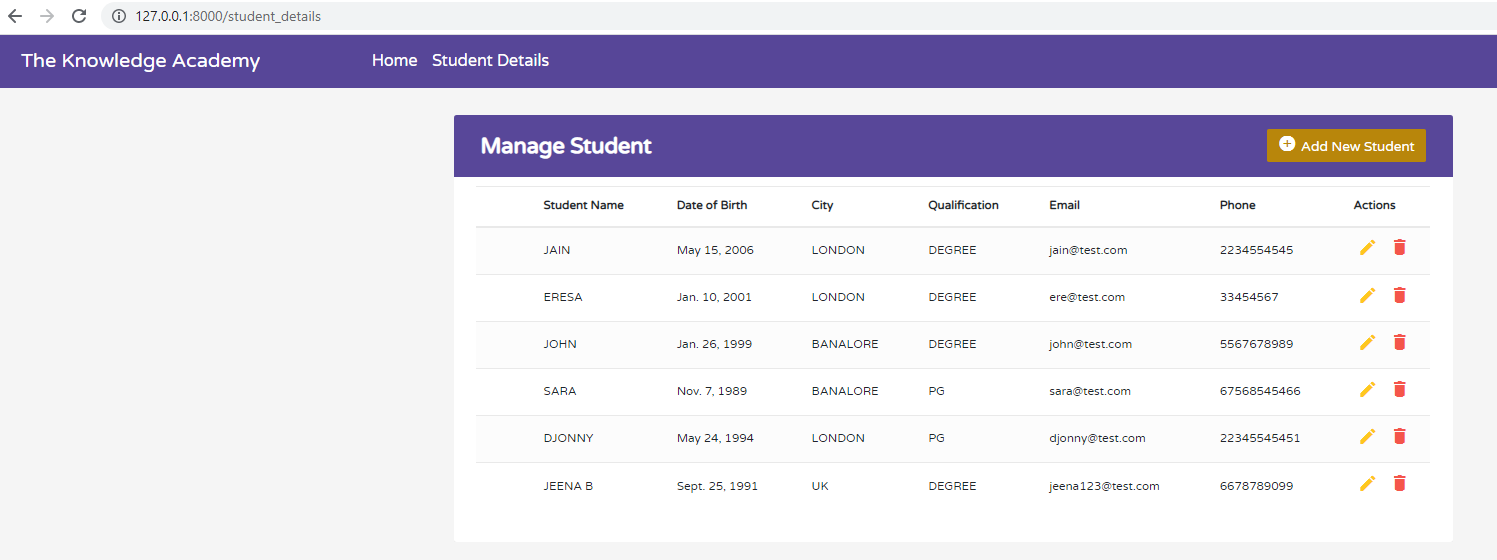
****

The output view page will be like as follows:



If you want to delete ANN details, click the delete button.

The output is:



We cannot see the ANN details in the view page. Because it is deleted from the database.

The GitHub link is <https://github.com/knowledgeacademy532/Django-project>