1)Set Environment

venv/Scripts/activate

2)Install Django

ensure you have Django installed.

pip install django

3)Create a new Django project using the following command:

django-admin startproject StudentsData .

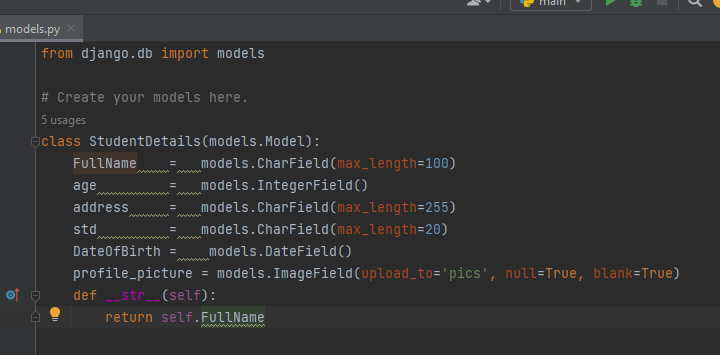
4)Create a new Django app using the following command:

python manage.py startapp DetailsApp

5)Add app name into settings.py INSTALLED\_APP

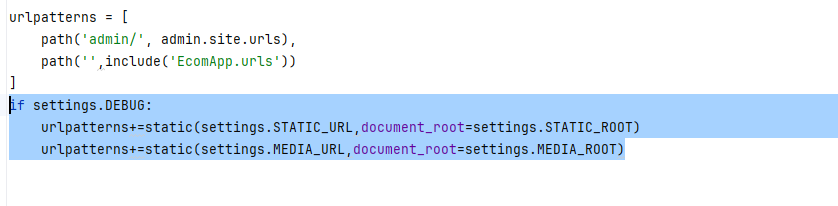
6)Create new table within DerailsApp

models.py



* Defining Model Fields:
  + FullName = models.CharField(max\_length=100): This line defines a field named FullName as a character field (CharField) with a maximum length of 100 characters.
  + age = models.IntegerField(): This line defines a field named age as an integer field (IntegerField).
  + address = models.CharField(max\_length=255): This line defines a field named address as a character field (CharField) with a maximum length of 255 characters.
  + std = models.CharField(max\_length=20): This line defines a field named std as a character field (CharField) with a maximum length of 20 characters.
  + DateOfBirth = models.DateField(): This line defines a field named DateOfBirth as a date field (DateField).
  + profile\_picture = models.ImageField(upload\_to='pics', null=True, blank=True): This line defines a field named profile\_picture as an image field (ImageField). It specifies that uploaded images should be stored in the 'pics' directory within the media directory of the Django project. It allows null values (null=True) and indicates that the field can be left blank (blank=True).
* Defining the \_\_str\_\_ Method:
  + def \_\_str\_\_(self):: This line defines a special method called \_\_str\_\_.
  + This method returns a string representation of the object.
  + In this case, it's overridden to return the FullName of the StudentDetails instance. This is useful for displaying meaningful information about the object, especially in contexts like Django's admin interface.

7)Create static two folder in main project folder Give name *‘static’* and *‘media’*

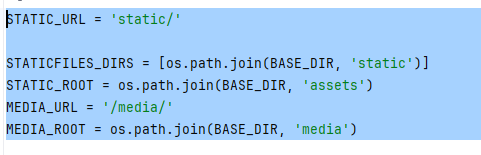
8) 

Include above code into urls.py in main project to support media

1. Ensure DEBUG is set to True in your project's settings file (settings.py).
2. Add the following code snippet to your urls.py file:
3. Explanation:
   1. This code dynamically adds URL patterns for serving static and media files when Django is running in debug mode.
   2. STATIC\_URL and MEDIA\_URL are the URLs used to access static and media files respectively.
   3. STATIC\_ROOT and MEDIA\_ROOT are the directories where static and media files are stored.

By adding this code, Django will serve static files directly during development, making it easier to work on your project's frontend.

9)



Configuring Static and Media Files in Django Settings

To configure static and media file handling in your Django project:

1. Set STATIC\_URL to '/static/' and MEDIA\_URL to '/media/'.
2. Define STATICFILES\_DIRS to include the directory where your static files reside.
3. Set STATIC\_ROOT to the directory where collected static files will be stored.
4. Define MEDIA\_ROOT to specify the directory where uploaded media files will be stored.

By configuring these settings in your settings.py file, Django will handle static and media files appropriately.

10)To create database migrations for the StudentDetails model in Django, follow these steps:

* Make sure your app is registered: Ensure that your app containing the StudentDetails model is included in the INSTALLED\_APPS setting of your Django project's settings.py file.
* Generate Migrations: Django provides a command to automatically generate migration files based on the changes in your models. In your terminal, navigate to the directory containing your manage.py file and run the following command:
* 

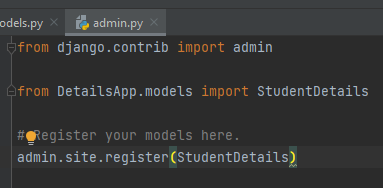
11)Review the Migration Files: After running makemigrations, you should see a new migration file created in the migrations directory of your app. Open the migration file to review the changes that will be applied to the database schema.



12)Create superuser

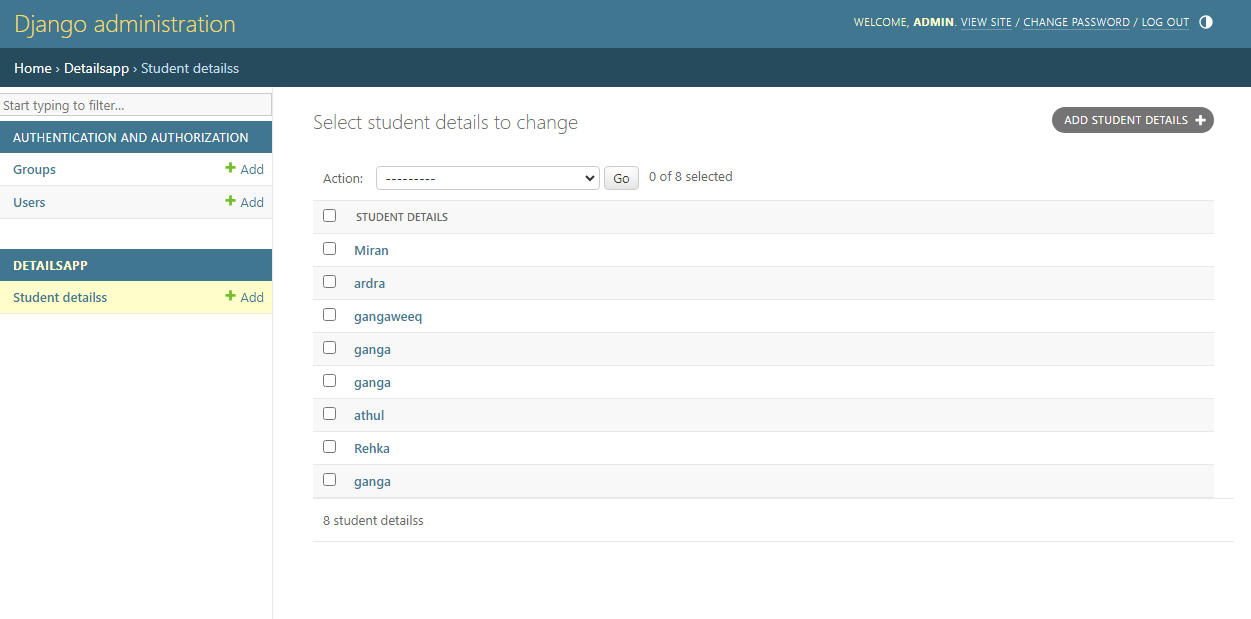


13)



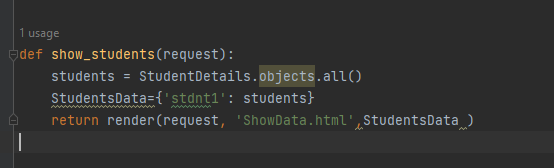
* This line registers the StudentDetails model with the Django admin interface.
* By registering the model, Django will automatically generate an interface in the admin site where you can view, add, edit, and delete instances of the StudentDetails model.

14)Add 3 to 4 new students details in admin page

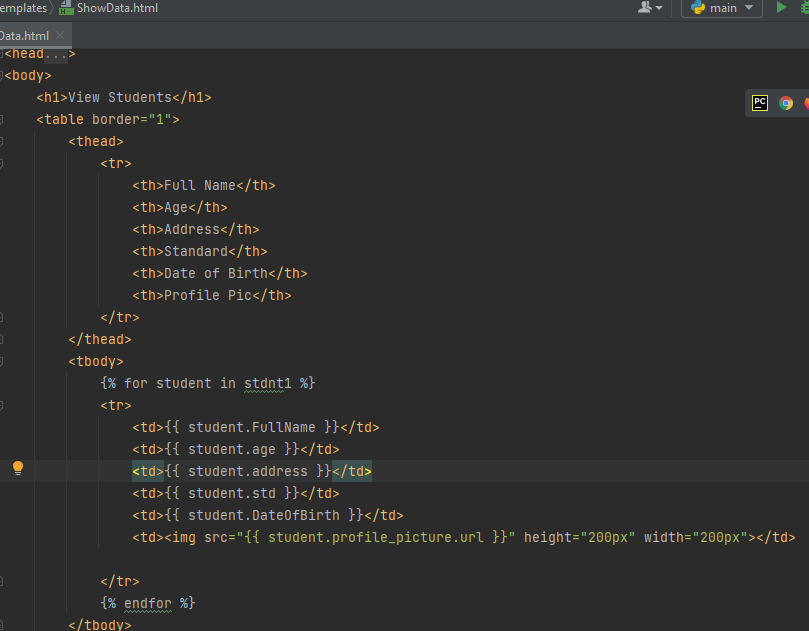


15)function show\_students is a view in a Django application that renders a template named 'ShowData.html' with data about all students.

view function retrieves all student details from the database, organizes them into a dictionary, and renders a template named 'ShowData.html' with the student data passed as context.



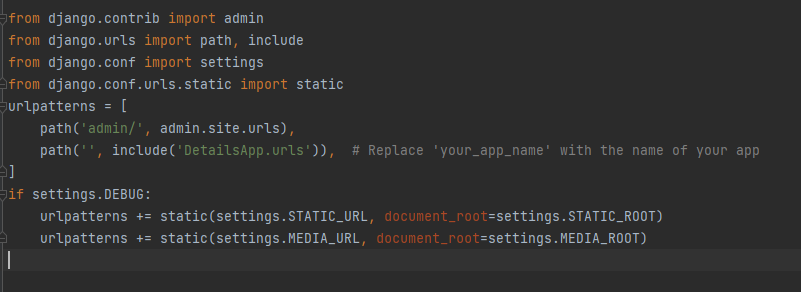
16)



This HTML template is designed to display a list of students with their details, including their full name, age, address, standard, date of birth, and profile picture. Let's break down the code:

* + nside the loop, each student's details are displayed within a table row (<tr>).
  + <td> tags represent table cells for each piece of student information.
  + Django template variables ({{ student.FullName }}, {{ student.age }}, etc.) are used to dynamically insert the corresponding values for each student.
  + The profile picture is displayed using an <img> tag, with the src attribute set to the URL of the profile picture obtained from the profile\_picture field of each student. The height and width attributes are set to 200 pixels to control the size of the image.

Overall, this HTML template works together with the Django view to dynamically display student details in a structured format on a web page.

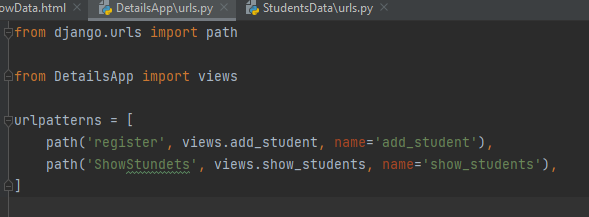
17)

* This block of code is conditional on DEBUG mode being enabled in the project settings (settings.DEBUG).
* It adds URL patterns to serve static files (CSS, JavaScript, images, etc.) and media files (user-uploaded files) during development.
* static() is used to serve static files, with the URL prefix defined by STATIC\_URL setting and the root directory defined by STATIC\_ROOT setting.
* Similarly, static() is used to serve media files, with the URL prefix defined by MEDIA\_URL setting and the root directory defined by MEDIA\_ROOT setting.

urlpatterns is a list of URL patterns for the Django project.

* The first pattern maps the URL /admin/ to the Django admin interface.
* The second pattern includes URL patterns defined in the DetailsApp.urls module. It matches the root URL ('') and delegates further routing to the URLs defined in the DetailsApp.urls module.

18)



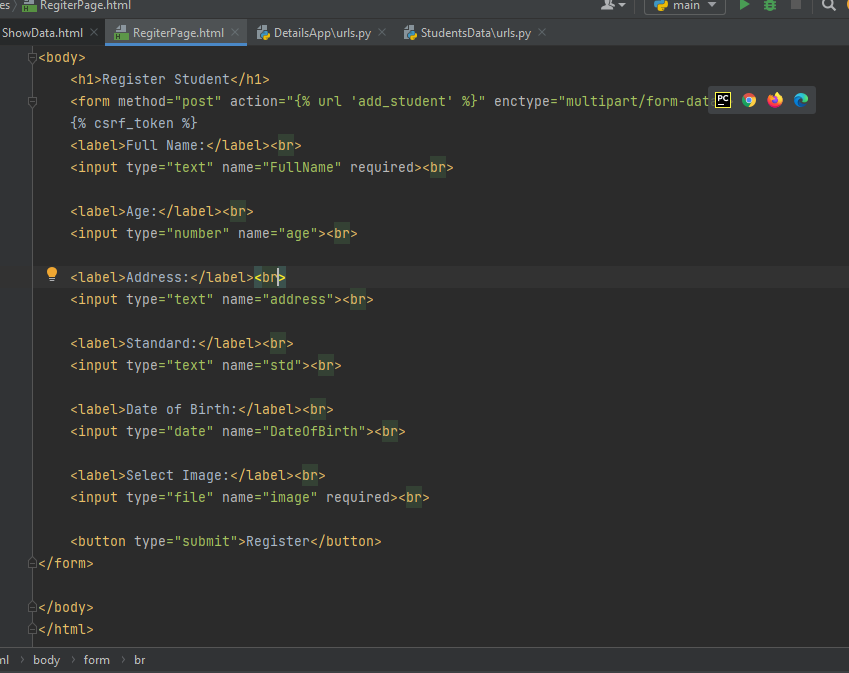
* + urlpatterns is a list of URL patterns for the DetailsApp application.
  + The first pattern maps the URL /register to the add\_student view function from views.py in the DetailsApp app. It also assigns the name 'add\_student' to this URL pattern, which can be used to refer to it in Django templates or code.
  + The second pattern maps the URL /ShowStundets to the show\_students view function. Similarly, it assigns the name 'show\_students' to this URL pattern.

In summary, this code configures two URL patterns for the DetailsApp application: one for registering new students (/register) and another for displaying existing students (/ShowStundets). Requests to these URLs will be routed to the respective view functions (add\_student and show\_students) for processing.

**To save data into the database**

**1)**Create an HTML Form:

Let's create a simple HTML form (register.html**)** where users can input data. We'll use this form to collect student details.



This HTML code creates a registration form where users can input their details, including their full name, age, address, standard, date of birth, and a profile picture. When the form is submitted, the data is sent to the server for processing

Opens a <form> element with the following attributes:

* method="post": Specifies that the form data will be submitted using the HTTP POST method.
* action="{% url 'add\_student' %}": Specifies the URL where the form data will be submitted. {% url 'add\_student' %} is a Django template tag that generates the URL for the add\_student view.
* enctype="multipart/form-data": Specifies how form data should be encoded when submitting the form. In this case, it's set to multipart/form-data, which is necessary when the form includes file uploads.

This is a Django template tag that adds a CSRF (Cross-Site Request Forgery) token to the form. It helps protect against CSRF attacks.

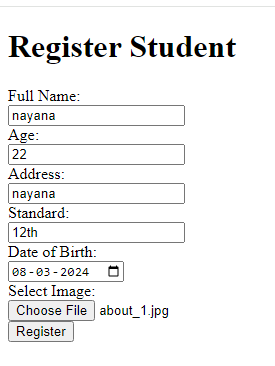
2)



* These lines retrieve data from the submitted form. request.POST.get() is used to get the values of form fields submitted via POST method, and request.FILES.get() is used to get the uploaded file (profile picture in this case).
* Each line extracts the value of a specific field from the form data based on its name attribute.
* This line creates a new instance of the StudentDetails model using the create() method, which simultaneously saves the object to the database.
* The field values for the new instance are set based on the data obtained from the form submission.

After creating the StudentDetails object, this line explicitly saves it to the database. Though create() method also saves the object, it's a good practice to call save() explicitly.

This view function encapsulates the process of handling form submissions, creating new StudentDetails objects, saving them to the database, and redirecting the user to another page upon successful submission.



This page will direct into students list page

