

Topic: 6. Single Record Fetching/Updating using PK (PUT/DELETE)

Speaker: / Notebook: API Development using Django Framework



1. To fetch a single record using a primary, we update the URLS.PY and add a new path:

```
api > urls.py > ...
1  from django.urls import path
2  from . import views
3
4  urlpatterns = [
5      path('students/', views.studentsView),
6      path('students/<int:pk>', views.studentDetailView),
7  ]
```

2. Update the VIEWS.PY and add the path plus the primary to search:

127.0.0.1:8000/api/v1/students/1/

Students / Student Detail

Student Detail

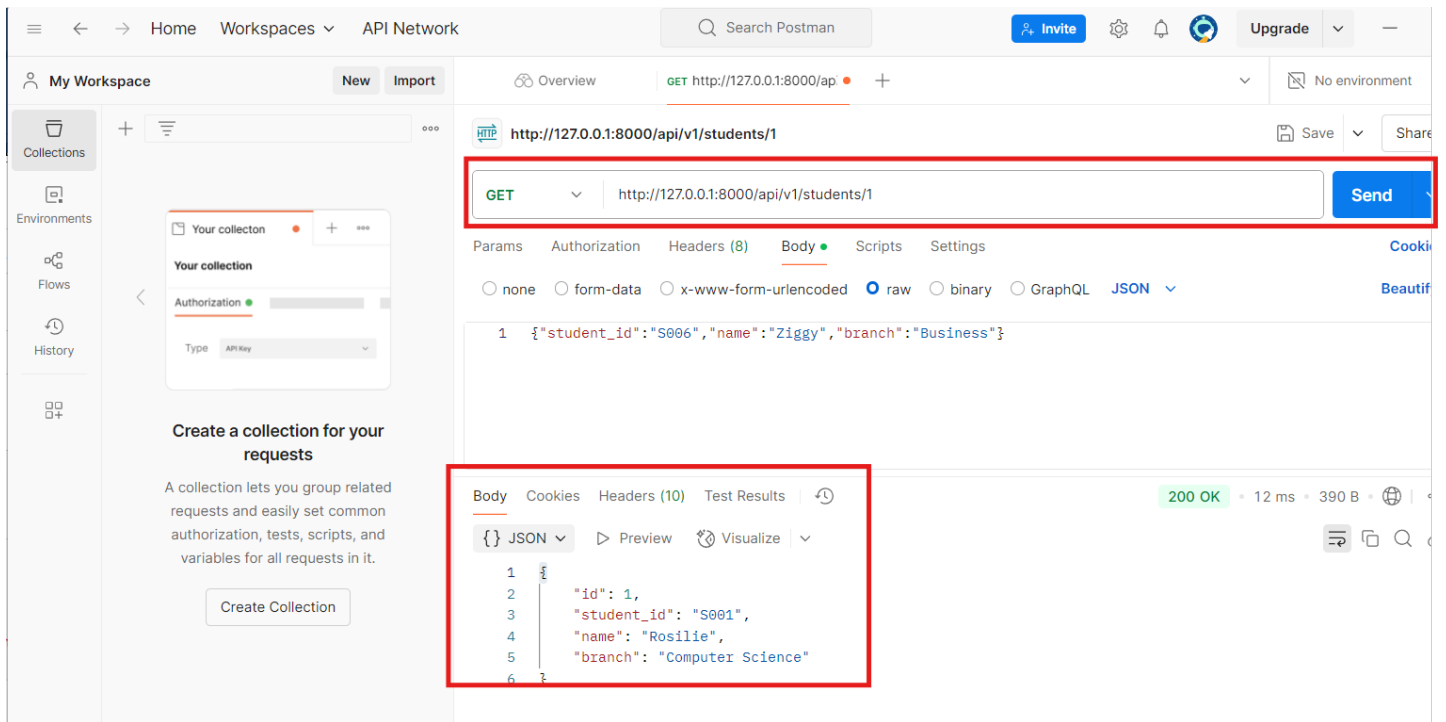
GET /api/v1/students/1/

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

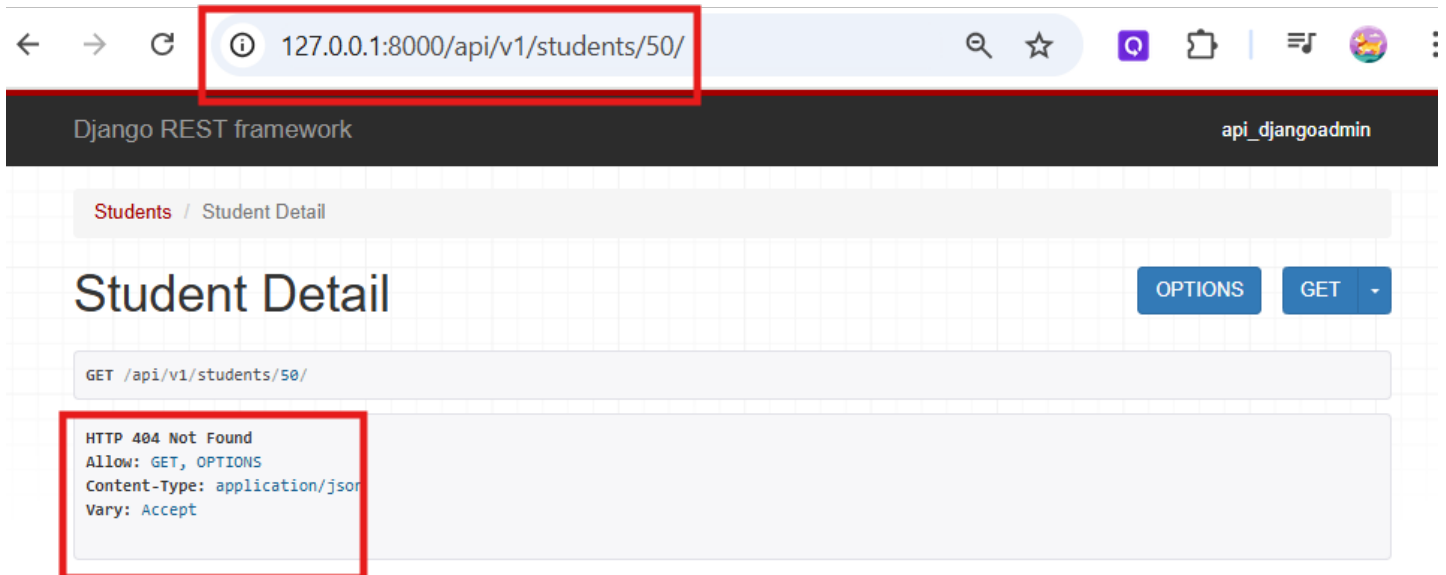
```
{
  "id": 1,
  "student_id": "5001",
  "name": "Rosilie",
  "branch": "Computer Science"
}
```

```
api > views.py > ...
26
27 @api_view(['GET'])
28 def studentDetailView(request, pk):
29     try:
30         student = Student.objects.get(pk=pk)
31     except Student.DoesNotExist:
32         return Response(status=status.HTTP_404_NOT_FOUND)
33
34     if request.method == 'GET':
35         serializer = StudentSerializer(student)
36         return Response(serializer.data, status=status.HTTP_200_OK)
37
38
```

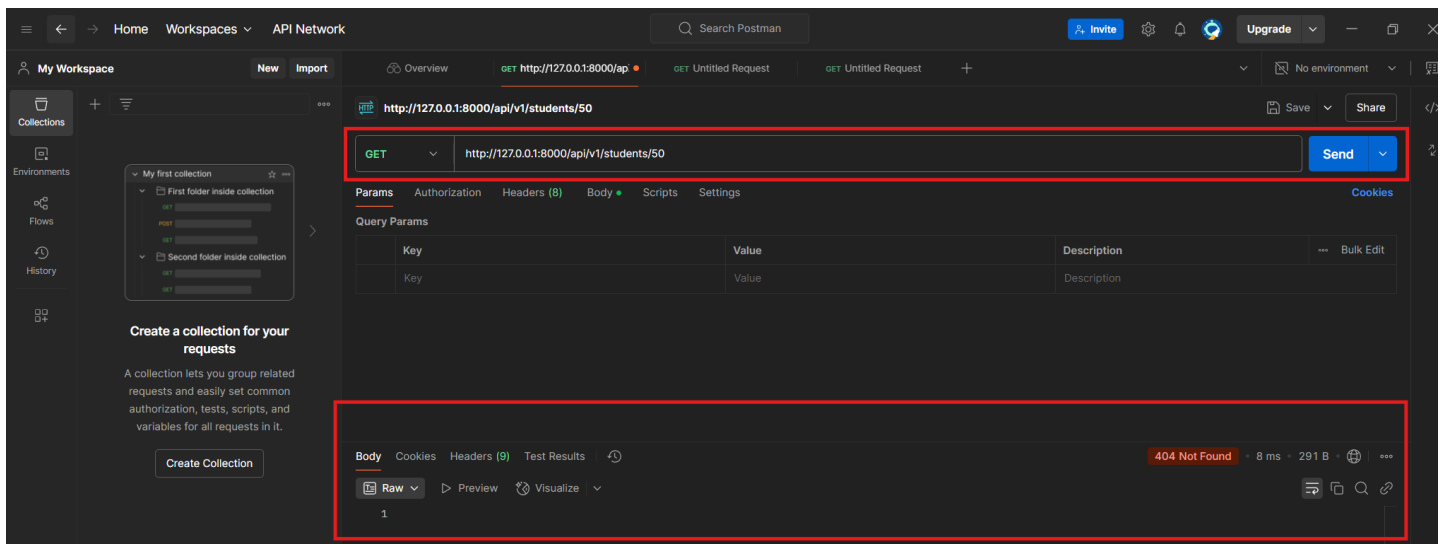
3. Using the POSTMAN, add the part and the primary key to retrieve that specific record. Click SEND:



In case, you search for the record that doesn't exist, the 404 error message should display:



4. In POSTMAN (we changed the theme), if you have a non-existing record:



5. To update a specific record, we update the VIEWS.PY. We use the PUT method.

```
File Edit Selection View ... ← → DjangoREST_APIProject
EXPLORER
  DJANGOREST_APIPROJECT
    api
      > __pycache__
      > migrations
      > __init__.py
      > admin.py
      > apps.py
      > models.py
      > serializers.py
      > tests.py
      > urls.py
      > views.py
    > django_rest_main
      > __pycache__
      > __init__.py
      > asgi.py
      > settings.py
      > urls.py
      > wsgi.py
    > env
    > students
    > __pycache__

api > views.py > studentDetailView
26
27 @api_view(['GET', 'PUT'])
28 def studentDetailView(request, pk):
29     try:
30         student = Student.objects.get(pk=pk)
31     except Student.DoesNotExist:
32         return Response(status=status.HTTP_404_NOT_FOUND)
33
34     # fetch a specific record using the PK
35     if request.method == 'GET':
36         serializer = StudentSerializer(student)
37         return Response(serializer.data, status=status.HTTP_200_OK)
38
39     # update a specific record using the PK
40     elif request.method == 'PUT':
41         serializer = StudentSerializer(student, data=request.data)
42         if serializer.is_valid():
43             serializer.save()
44             return Response(serializer.data, status=status.HTTP_200_OK)
45         else:
46             return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)
47
48
```

Updating the record:

127.0.0.1:8000/api/v1/students/2/

Django REST framework

api_djangoadmin

Students / Student Detail

Student Detail

OPTIONS GET

GET /api/v1/students/2/

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

```
{
  "id": 2,
  "student_id": "S002",
  "name": "Yuri",
  "branch": "Engineering"
}
```

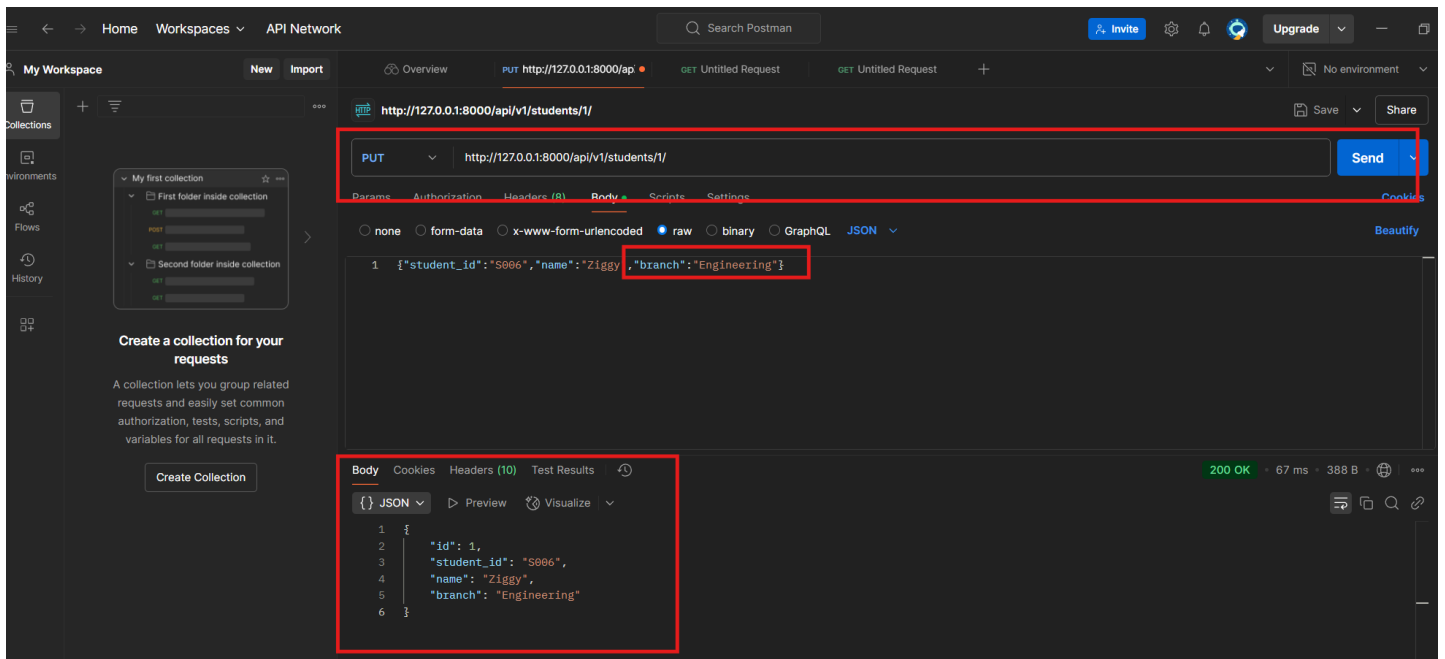
Media type: application/json

Content:

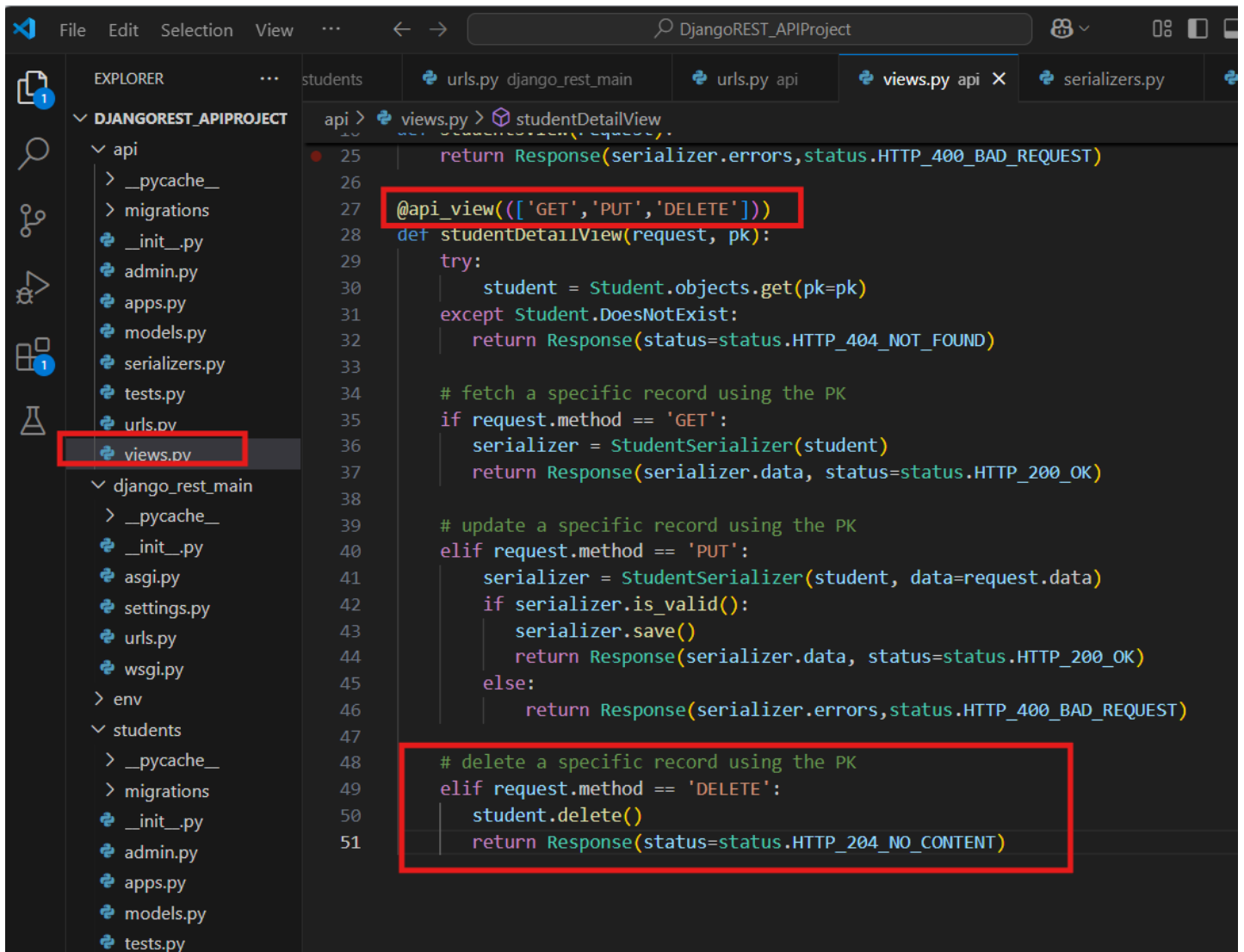
```
{
  "student_id": "S002",
  "name": "Yuri",
  "branch": "Aviation"
}
```

PUT

6. Using the POSTMAN , use the PUT method with the primary record number:



7. To delete a record, use the DELETE method. Update the VIEWS.PY:



← → ↻ ⓘ 127.0.0.1:8000/api/v1/students/ 🔍 ☆ 📄 📁 🎵 🌐

Django REST framework api_djangoadmin

```
{
  "id": 1,
  "student_id": "S001",
  "name": "Rosilie",
  "branch": "Computer Science"
},
{
  "id": 2,
  "student_id": "S002",
  "name": "Yuri",
  "branch": "Aviation"
},
{
  "id": 3,
  "student_id": "S003",
  "name": "Xeria",
  "branch": "Hotel Mgt"
},
{
  "id": 4,
  "student_id": "S004",
  "name": "Russell",
  "branch": "Veterinary"
},
{
  "id": 5,
  "student_id": "S005",
  "name": "Mary Ann",
  "branch": "Engineering"
},
{
  "id": 6,
  "student_id": "S006",
  "name": "Ziggy",
  "branch": "Business"
}
}
```

Media type: application/json

Content: {
 "student_id": "S008",
 "name": "Some Strangers",
 "branch": "N/A"
}

POST

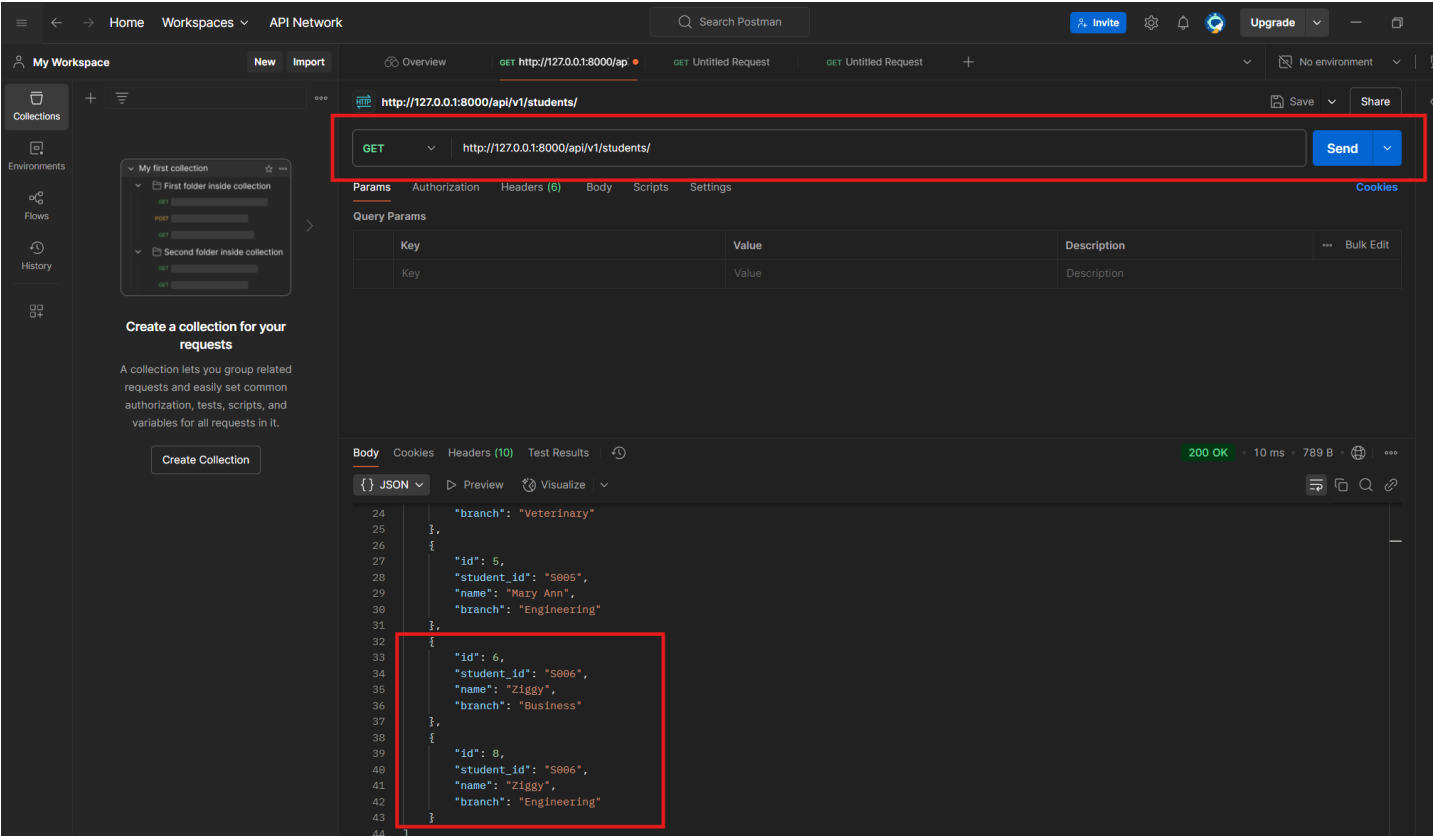
When you reload your page, you will see the DELETE button:

The screenshot shows a web browser at the URL `127.0.0.1:8000/api/v1/students/7/`. The page title is "Student Detail". In the top right corner, there are three buttons: "DELETE" (red), "OPTIONS" (blue), and "GET" (blue). Below the title, the HTTP method "GET" is shown for the endpoint `/api/v1/students/7/`. The response is a JSON object: `{ "id": 7, "student_id": "s008", "name": "Some Strangers", "branch": "N/A" }`. Below the response, there is a "Media type" dropdown set to "application/json" and a "Content" text area. A "PUT" button is located at the bottom right of the content area.

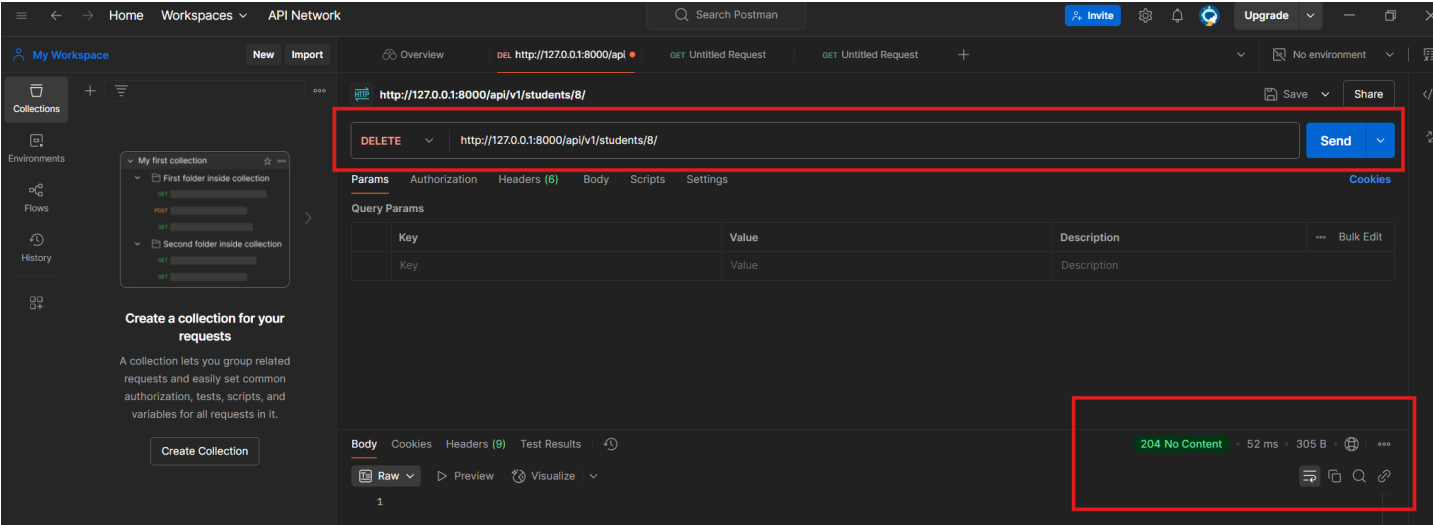
8. Now, reload your page and locate that deleted record, it should return HTTP error message:

The screenshot shows the same web browser at the same URL. The page title is "Student Detail". The "DELETE" button is now highlighted. Below the title, the HTTP method "DELETE" is shown for the endpoint `/api/v1/students/7/`. The response is "HTTP 204 No Content". Below the response, there is a "Media type" dropdown set to "application/json" and a "Content" text area.

9. Using the POSTMAN, use the delete the method. Assume you have added several records. Delete these unnecessary records.



When you use the DELETE method and click SEND, the output should be.



10.