NAME: OLAIYA BISOYE OLORUNFEMI

TRACK: SOFTWARE DEVELOPMENT (PYTHON WEB DEVELOPMENT)

TASK: WEEK 4

DATE OF SUBMISSION: 28th November, 2019

GITLAB USERNAME: bisoyefemi

Project Link: https://gitlab.com/bisoyefemi/week-four-task.git

Week Four: Web App

i. Research and read on restful API using Django
ii. Setup any of the frameworks on your PC
iii. Write a restful API to return hello world
iv. Push your project to Git and add link to your report.
v. Research on PostgreSQL database
vi. create a Database using PostgreSQL locally
vii. Add tables to your database
viii. Add, read, update and delete your first record on the table
ix. Connect to the database using your API to create, read, update and delete entries from the database
x. export the database into SQL and CSV format for submission.
xi. Good luck guys

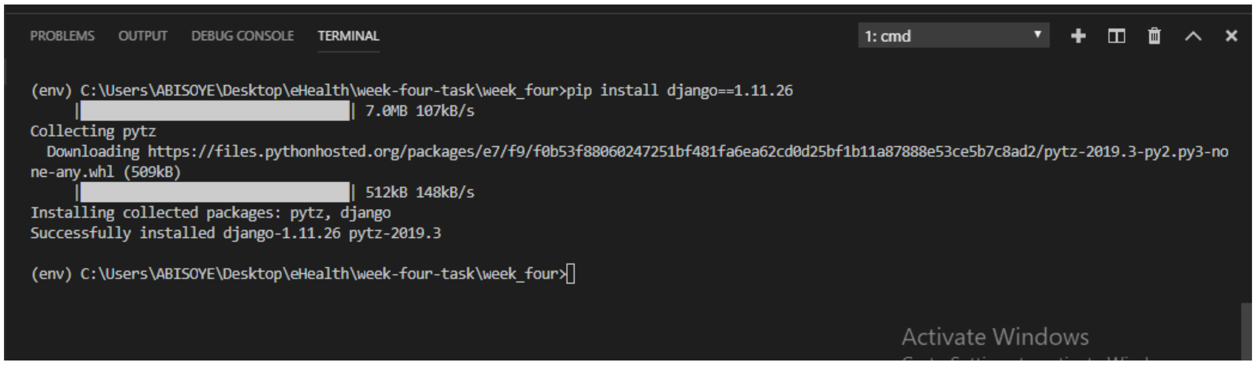
Solution:

1. **REST Services with Django**
   Representational State Transfer (REST) services or simply RESTful services has become of the most popular techniques in web development. A REST service provides access to data through an endpoint or Internet URL making no assumption about who uses it (e.g. an IoT device, a mobile phone or a desktop browser); this means that there is no device or environment requirements to access REST services. On top of this, because REST services operate on Internet URLS, they provide a very intuitive access scheme. REST services work as a reusable data back bone which spreads across a wide area. For example, practically, an entire web API (Application Programming Interface) world operates around REST web services. Because by definition, an API must provide a device/environment neutral way for customers to access their functionality. In fact, there is a high chance that most of the websites you visit makes use of REST services one way or another, a practice that allows website operators reuse the same data and then format it for users on desktop browsers, IoT device, mobile application, RSS feeds, or any other target like Accelerated Mobile Pages (AMP).

2. **Django and REST Framework Set up**
   To set up Django web framework on your PC, you must have Python installed first. Django is a Python web framework. I have python 3.6 version installed on my pc. Next thing is to create a project folder which I named week_four_task which was cloned from my gitlab repository. Afterwards, I created a virtual environment by entering the command “python -m venv env” in my terminal then afterwards activated my virtual environment with the command “env\Scripts\activated”.

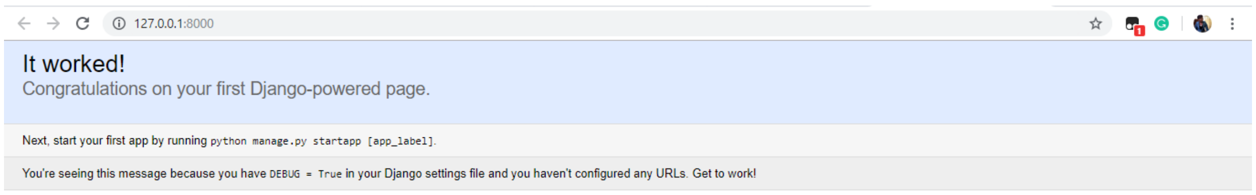
Once my environment is activated, I installed Django version 1.11.26 by entering the command “pip install Django==1.11.26”. Once Django is successfully installed, I installed Django rest framework by entering the command
“pip install djangorestframework” ; I created two project folders task_one_src and task_two_src to contain the first task about API and the second task about Postgres Database.

For task_one_src folder:
I created a project called myapp by entering the command “django-admin startproject myapp . ” and then created appone by entering the command “python manage.py startapp appone” Once appone is created successful, I installed rest_framework and appone in my myapp settings file.



```
(env) C:\Users\ABISOYE\Desktop\eHealth\week-four-task\week_four>pip install django==1.11.26
     |████████████████████████████████| 7.0MB 107kB/s
Collecting pytz
  Downloading https://files.pythonhosted.org/packages/e7/f9/f0b53f88060247251bf481fa6ea62cd0d25bf1b11a87888e53ce5b7c8ad2/pytz-2019.3-py2.py3-no
ne-any.whl (509kB)
     |████████████████████████████████| 512kB 148kB/s
Installing collected packages: pytz, django
Successfully installed django-1.11.26 pytz-2019.3

(env) C:\Users\ABISOYE\Desktop\eHealth\week-four-task\week_four>
```

Figure 1.1: Successful Install of Django Version 1.11.26

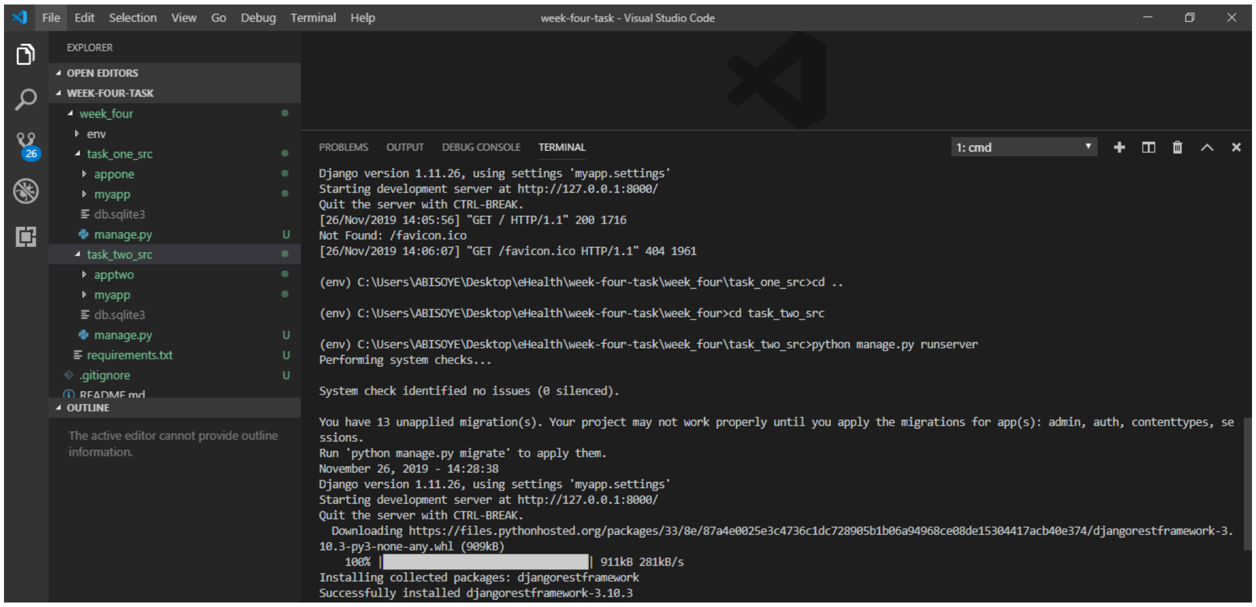


It worked!
Congratulations on your first Django-powered page.

Next, start your first app by running `python manage.py startapp [app_label]`.

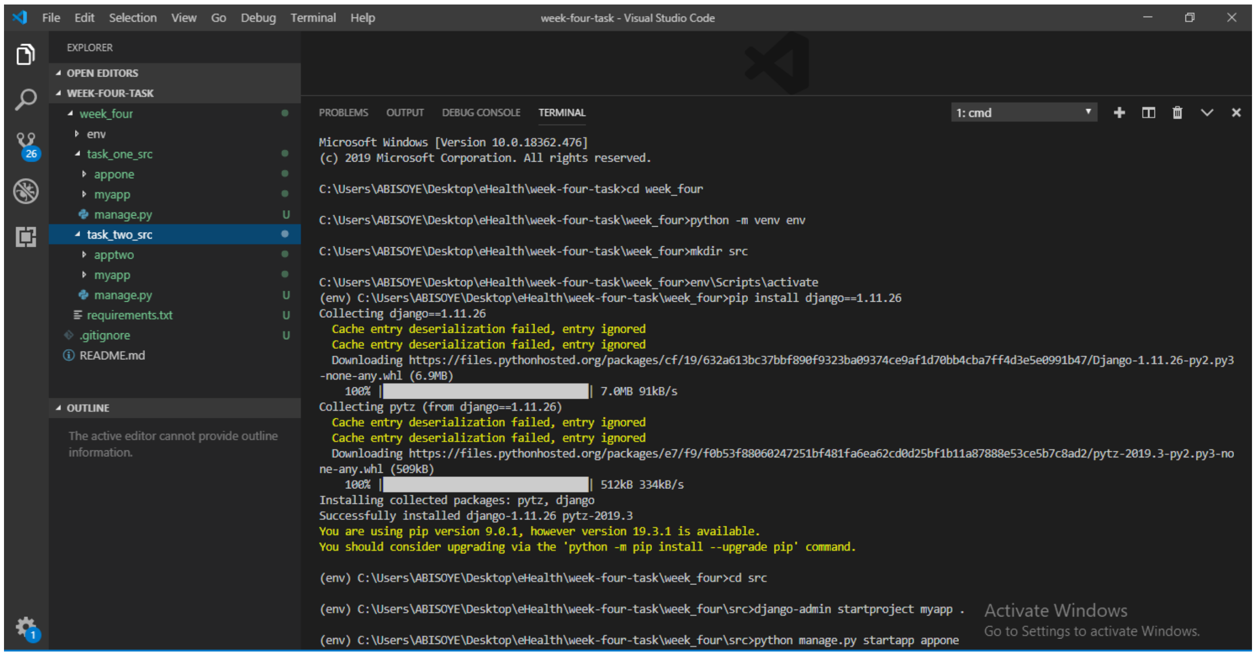
You're seeing this message because you have `DEBUG = True` in your Django settings file and you haven't configured any URLs. Get to work!

Figure 1.2: Successful Django Rendered Page



```
Django version 1.11.26, using settings 'myapp.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
[26/Nov/2019 14:05:56] "GET / HTTP/1.1" 200 1716
Not Found: /favicon.ico
[26/Nov/2019 14:06:07] "GET /favicon.ico HTTP/1.1" 404 1961

(env) C:\Users\ABISOYE\Desktop\eHealth\week-four-task\week_four\task_one_src>cd ..

(env) C:\Users\ABISOYE\Desktop\eHealth\week-four-task\week_four>cd task_two_src

(env) C:\Users\ABISOYE\Desktop\eHealth\week-four-task\week_four\task_two_src>python manage.py runserver
Performing system checks...

System check identified no issues (0 silenced).

You have 13 unapplied migration(s). Your project may not work properly until you apply the migrations for app(s): admin, auth, contenttypes, se
ssions.
Run 'python manage.py migrate' to apply them.
November 26, 2019 - 14:28:38
Django version 1.11.26, using settings 'myapp.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
  Downloading https://files.pythonhosted.org/packages/33/8e/87a4e0025e3c4736c1dc728905b1b06a94968ce08de15304417acb40e374/djangorestframework-3.
10.3-py3-none-any.whl (909kB)
    100% |████████████████████████████████| 911kB 281kB/s
Installing collected packages: djangorestframework
Successfully installed djangorestframework-3.10.3
```

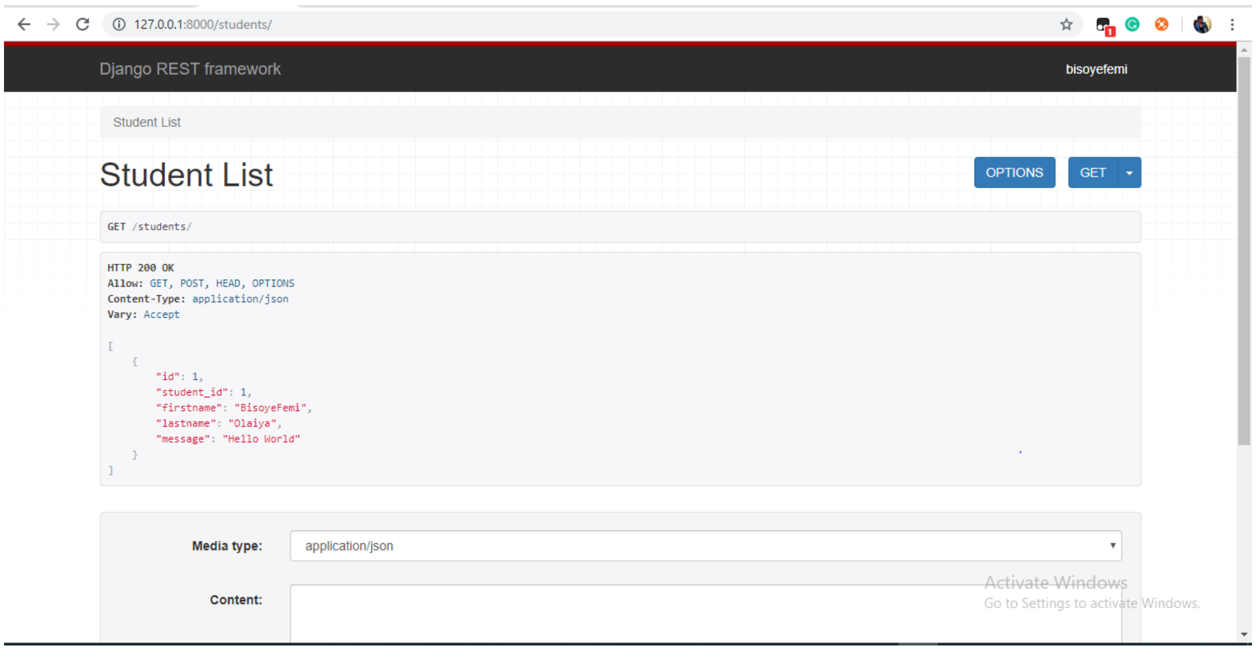
Figure 1.3: Successful Install of Django Rest Framework


File Edit Selection View Go Debug Terminal Help
week-four-task - Visual Studio Code
EXPLORER
OPEN EDITORS
WEEK-FOUR-TASK
week_four
env
task_one_src
appone
myapp
manage.py
task_two_src
apptwo
myapp
manage.py
requirements.txt
.gitignore
README.md
OUTLINE
The active editor cannot provide outline information.
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
1: cmd
Activate Windows
Go to Settings to activate Windows.

```
Microsoft Windows [Version 10.0.18362.476]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\ABISOYE\Desktop\eHealth\week-four-task>cd week_four

C:\Users\ABISOYE\Desktop\eHealth\week-four-task\week_four>python -m venv env

C:\Users\ABISOYE\Desktop\eHealth\week-four-task\week_four>mkdir src

C:\Users\ABISOYE\Desktop\eHealth\week-four-task\week_four>env\Scripts\activate
(env) C:\Users\ABISOYE\Desktop\eHealth\week-four-task\week_four>pip install django==1.11.26
Collecting django==1.11.26
  Cache entry deserialization failed, entry ignored
  Cache entry deserialization failed, entry ignored
  Downloading https://files.pythonhosted.org/packages/cf/19/632a613bc37bbf890f9323ba09374ce9af1d70bb4cba7ff4d3e5e0991b47/Django-1.11.26-py2.py3
-none-any.whl (6.9MB)
    100% |████████████████████████████████| 7.0MB 91kB/s
Collecting pytz (from django==1.11.26)
  Cache entry deserialization failed, entry ignored
  Cache entry deserialization failed, entry ignored
  Downloading https://files.pythonhosted.org/packages/e7/f9/f0b53f88060247251bf481fa6ea62cd0d25bf1b11a87888e53ce5b7c8ad2/pytz-2019.3-py2.py3-no
ne-any.whl (509kB)
    100% |████████████████████████████████| 512kB 334kB/s
Installing collected packages: pytz, django
Successfully installed django-1.11.26 pytz-2019.3
You are using pip version 9.0.1, however version 19.3.1 is available.
You should consider upgrading via the 'python -m pip install --upgrade pip' command.

(env) C:\Users\ABISOYE\Desktop\eHealth\week-four-task\week_four>cd src

(env) C:\Users\ABISOYE\Desktop\eHealth\week-four-task\week_four\src>django-admin startproject myapp .

(env) C:\Users\ABISOYE\Desktop\eHealth\week-four-task\week_four\src>python manage.py startapp appone
```

Fig 1.4: Successful Project Setup of myapp project and appone

3. **Write a RESTful API to return Hello World.**
   I created a Student class in my models.py file and registered the model in my admin.py file. Afterwards I did migrations "python manage.py makemigrations" then "python manage.py migrate" Once the migrations were successful, I created a superuser account "python manage.py createsuperuser" to access the admin panel. I verified the student model created was registered on my admin panel; I created a serializers.py file to create a student serializer.
   Thereafter I modified appone views.py and myapp project urls.py. The screenshot below is the result api interface the returns a student message "Hello World"



```
GET /students/
```

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

[
    {
        "id": 1,
        "student_id": 1,
        "firstname": "BisoyeFemi",
        "lastname": "Olaiya",
        "message": "Hello World"
    }
]
```

Figure1.5: Screenshot showing result of student message displaying "Hello World"

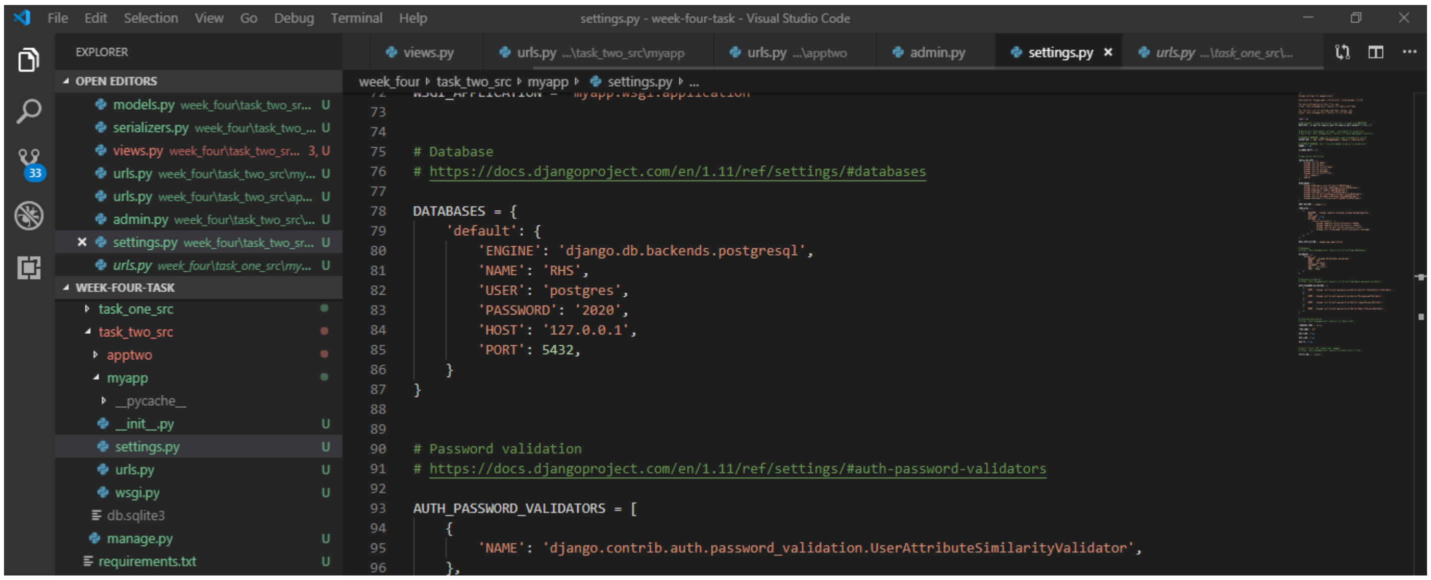
4. **Research on PostgreSQL Database**

   PostgreSQL can be far dated to the 1970's as a research project called Ingres Database Project was developed but it was not perfect so in the 1980's, a follow up project was started to fix some problems with Ingres. Because it came after Ingres, this project was called Postgres and then the 1990's even more improvements were made instead of using the POSTQUEL query language, the database was updated to support SQL this led to the name PostgreSQL till date.

   PostgreSQL a powerful object-relational database management system is modern, popular and an open source software that is free to use by anyone. It has over 30 years of active development that has earned it a strong reputation for reliability, feature robustness, and performance. Amongst its competitors such as MySQL, Oracle, Microsoft SQL Server and others, Postgres remains one of the top options of many individuals and organizations in our present time.

5. **Create a Database using Postgres Locally.**
To create a Postgres database, first you must set it up locally by visiting the official website download page www.postgresql.org/download to download the installer. I have PostgreSQL 11.6 installed on my PC. Afterward you will need to download pgAdmin 4 Browser User Interface and have it set up as well. Thereafter, on your project folder working directory, enter the command “pip install psycopg2” connector for PostgreSQL database setup.

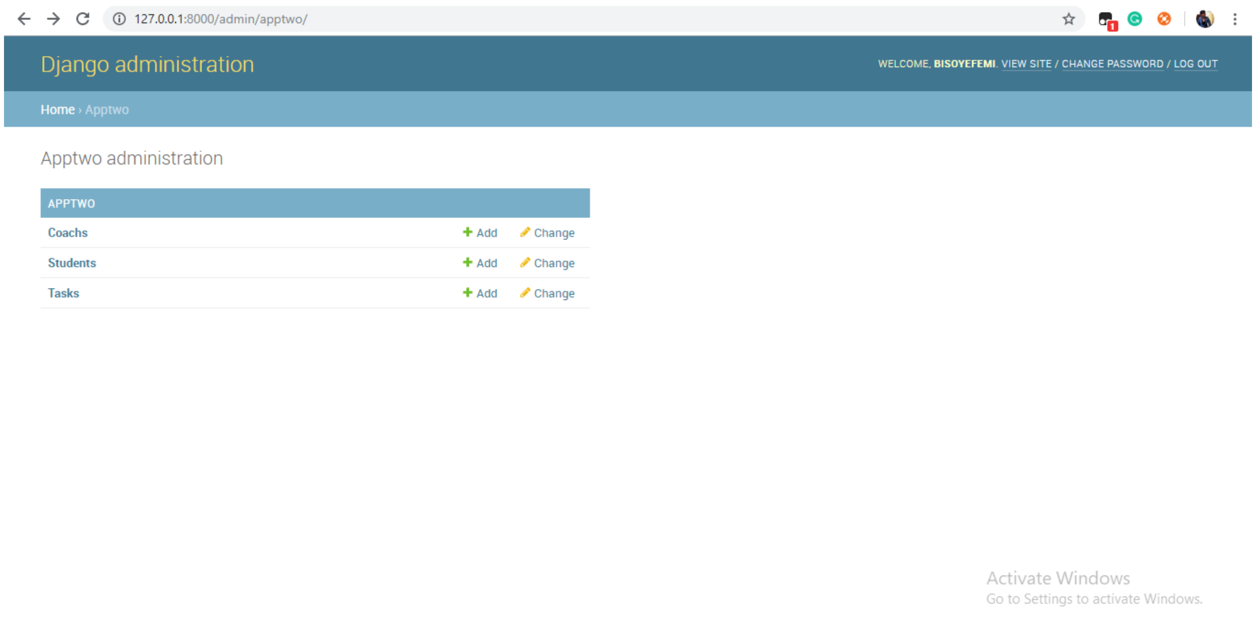
After the above, I changed into task_two_src project folder directory and created an app called apptwo, installed it and rest_framwork on my settings.py file. I modified apptwo models.py file and created three classes Student, Coach and Task as tables. Then I registered the model in the admin.py file and modified my settings file changing the database configuration to a PostgreSQL backend engine.


settings.py - week-four-task - Visual Studio Code
week_four › task_two_src › myapp › settings.py › ...

```
# Database
# https://docs.djangoproject.com/en/1.11/ref/settings/#databases

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',
        'NAME': 'RHS',
        'USER': 'postgres',
        'PASSWORD': '2020',
        'HOST': '127.0.0.1',
        'PORT': 5432,
    }
}


# Password validation
# https://docs.djangoproject.com/en/1.11/ref/settings/#auth-password-validators

AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME': 'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
    },
```

Figure 1.6 Screenshot displaying the database configurations settings for PostgreSQL engine.

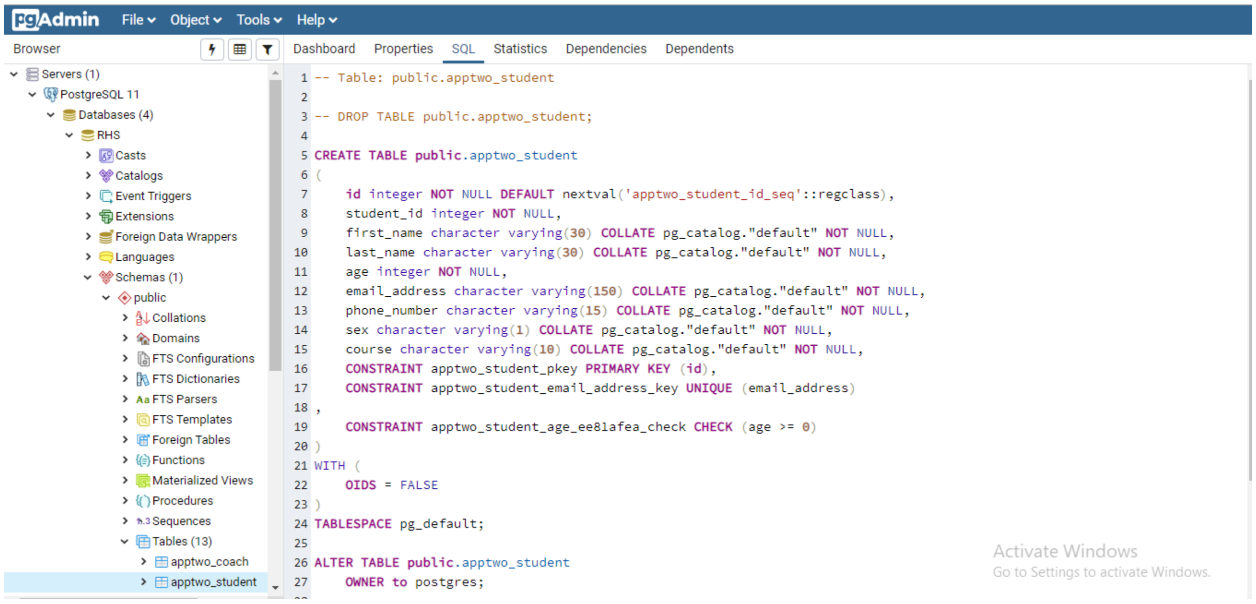
After this setup, I did migrations, successfully created my tables and created superuser.


127.0.0.1:8000/admin/apptwo/
Django administration
WELCOME, BISOYEFEMI. VIEW SITE / CHANGE PASSWORD / LOG OUT
Home › Apptwo
Activate Windows
Go to Settings to activate Windows.

Apptwo administration

APPTWO		
Coachs	+ Add	Change
Students	+ Add	Change
Tasks	+ Add	Change

Figure 1.7: Screenshot showing Admin Panel after successful Migration



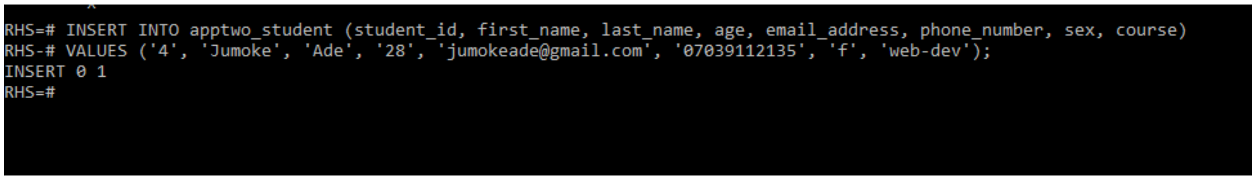
```sql
-- Table: public.apptwo_student

-- DROP TABLE public.apptwo_student;

CREATE TABLE public.apptwo_student
(
    id integer NOT NULL DEFAULT nextval('apptwo_student_id_seq'::regclass),
    student_id integer NOT NULL,
    first_name character varying(30) COLLATE pg_catalog."default" NOT NULL,
    last_name character varying(30) COLLATE pg_catalog."default" NOT NULL,
    age integer NOT NULL,
    email_address character varying(150) COLLATE pg_catalog."default" NOT NULL,
    phone_number character varying(15) COLLATE pg_catalog."default" NOT NULL,
    sex character varying(1) COLLATE pg_catalog."default" NOT NULL,
    course character varying(10) COLLATE pg_catalog."default" NOT NULL,
    CONSTRAINT apptwo_student_pkey PRIMARY KEY (id),
    CONSTRAINT apptwo_student_email_address_key UNIQUE (email_address)
,
    CONSTRAINT apptwo_student_age_ee81afea_check CHECK (age >= 0)
)
WITH (
    OIDS = FALSE
)
TABLESPACE pg_default;

ALTER TABLE public.apptwo_student
    OWNER to postgres;
```

Figure 1.8: Screenshot of pgAdmin User Interface showing RHS Database with the tables

6. Add, read, update and delete your first record on the table

To execute the above, I utilized the psql SQL Shell to perform manipulate the tables.

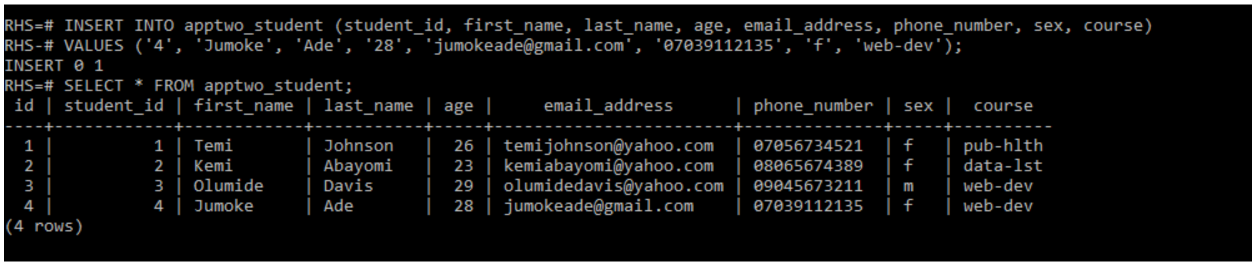
To add a new record:



```
RHS=# INSERT INTO apptwo_student (student_id, first_name, last_name, age, email_address, phone_number, sex, course)
RHS-# VALUES ('4', 'Jumoke', 'Ade', '28', 'jumokeade@gmail.com', '07039112135', 'f', 'web-dev');
INSERT 0 1
RHS=#
```

Figure 1.9: Screenshot showing a successful insert entry

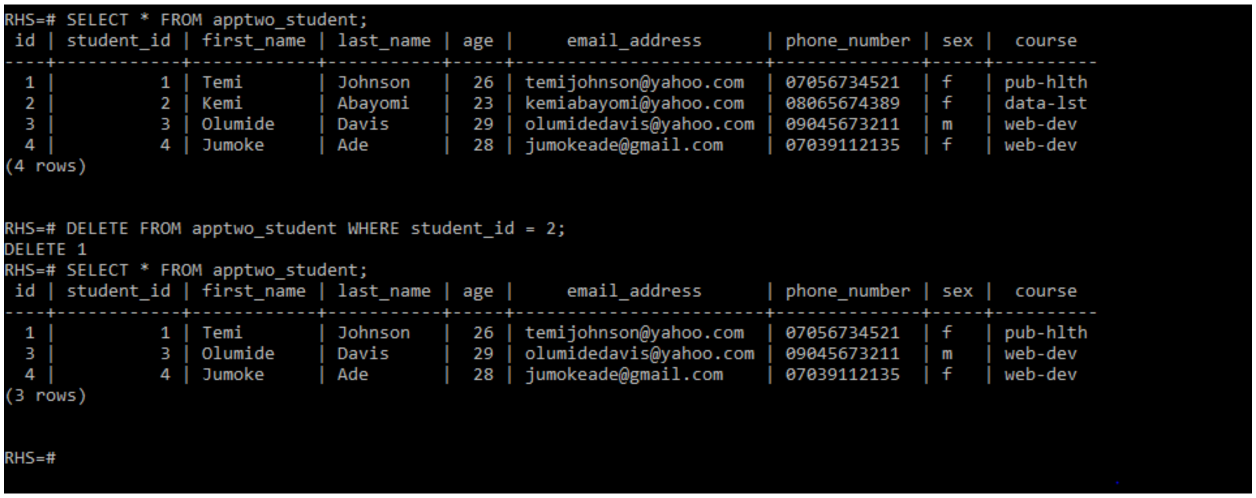
To read all records:



```
RHS=# INSERT INTO apptwo_student (student_id, first_name, last_name, age, email_address, phone_number, sex, course)
RHS-# VALUES ('4', 'Jumoke', 'Ade', '28', 'jumokeade@gmail.com', '07039112135', 'f', 'web-dev');
INSERT 0 1
RHS=# SELECT * FROM apptwo_student;
 id | student_id | first_name | last_name | age |      email_address      | phone_number | sex |  course
----+------------+------------+-----------+-----+-------------------------+--------------+-----+----------
  1 |          1 | Temi       | Johnson   |  26 | temijohnson@yahoo.com   | 07056734521  | f   | pub-hlth
  2 |          2 | Kemi       | Abayomi   |  23 | kemiabayomi@yahoo.com   | 08065674389  | f   | data-lst
  3 |          3 | Olumide    | Davis     |  29 | olumidedavis@yahoo.com  | 09045673211  | m   | web-dev
  4 |          4 | Jumoke     | Ade       |  28 | jumokeade@gmail.com     | 07039112135  | f   | web-dev
(4 rows)
```

Figure 2.0 Screenshot showing all records in apptwo_student table

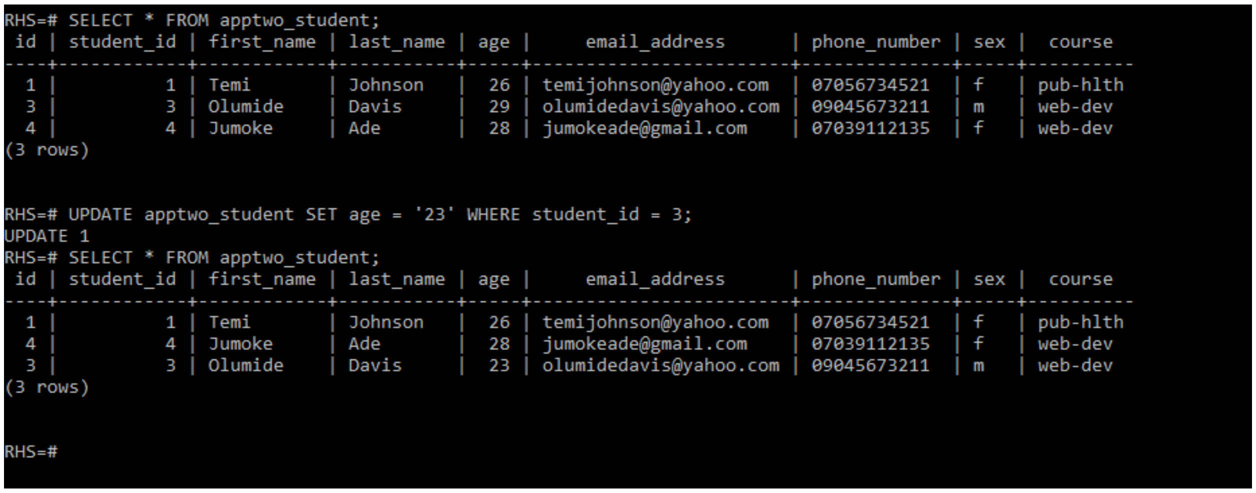
To delete a record:



```
RHS=# SELECT * FROM apptwo_student;
 id | student_id | first_name | last_name | age |      email_address      | phone_number | sex |  course
----+------------+------------+-----------+-----+-------------------------+--------------+-----+----------
  1 |          1 | Temi       | Johnson   |  26 | temijohnson@yahoo.com   | 07056734521  | f   | pub-hlth
  2 |          2 | Kemi       | Abayomi   |  23 | kemiabayomi@yahoo.com   | 08065674389  | f   | data-lst
  3 |          3 | Olumide    | Davis     |  29 | olumidedavis@yahoo.com  | 09045673211  | m   | web-dev
  4 |          4 | Jumoke     | Ade       |  28 | jumokeade@gmail.com     | 07039112135  | f   | web-dev
(4 rows)


RHS=# DELETE FROM apptwo_student WHERE student_id = 2;
DELETE 1
RHS=# SELECT * FROM apptwo_student;
 id | student_id | first_name | last_name | age |      email_address      | phone_number | sex |  course
----+------------+------------+-----------+-----+-------------------------+--------------+-----+----------
  1 |          1 | Temi       | Johnson   |  26 | temijohnson@yahoo.com   | 07056734521  | f   | pub-hlth
  3 |          3 | Olumide    | Davis     |  29 | olumidedavis@yahoo.com  | 09045673211  | m   | web-dev
  4 |          4 | Jumoke     | Ade       |  28 | jumokeade@gmail.com     | 07039112135  | f   | web-dev
(3 rows)


RHS=#
```

Figure 2.1 Screenshot showing a successful deleted record in apptwo_student table

To update a record:

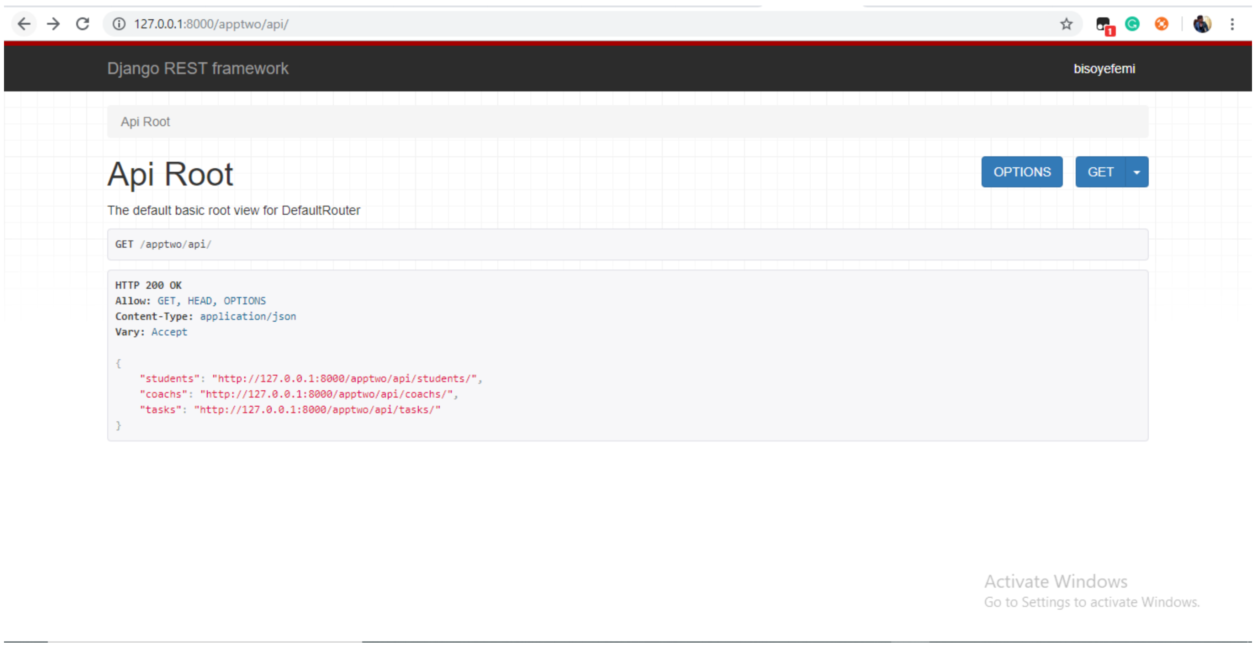


```
RHS=# SELECT * FROM apptwo_student;
 id | student_id | first_name | last_name | age |     email_address      | phone_number | sex |  course
----+------------+------------+-----------+-----+------------------------+--------------+-----+----------
  1 |          1 | Temi       | Johnson   |  26 | temijohnson@yahoo.com  | 07056734521  | f   | pub-hlth
  3 |          3 | Olumide    | Davis     |  29 | olumidedavis@yahoo.com | 09045673211  | m   | web-dev
  4 |          4 | Jumoke     | Ade       |  28 | jumokeade@gmail.com    | 07039112135  | f   | web-dev
(3 rows)


RHS=# UPDATE apptwo_student SET age = '23' WHERE student_id = 3;
UPDATE 1
RHS=# SELECT * FROM apptwo_student;
 id | student_id | first_name | last_name | age |     email_address      | phone_number | sex |  course
----+------------+------------+-----------+-----+------------------------+--------------+-----+----------
  1 |          1 | Temi       | Johnson   |  26 | temijohnson@yahoo.com  | 07056734521  | f   | pub-hlth
  4 |          4 | Jumoke     | Ade       |  28 | jumokeade@gmail.com    | 07039112135  | f   | web-dev
  3 |          3 | Olumide    | Davis     |  23 | olumidedavis@yahoo.com | 09045673211  | m   | web-dev
(3 rows)


RHS=#
```

Figure 2.2: Screenshot showing a successful update record in apptwo_student table

7. **Connect to the database using your API to create, read, update and delete entries from the database**

To execute the above, I created a serializers.py file and urls.py file to handle my model serializers and register my routers. Also, I had my views.py file modified with my ModelViewSet. Once this was all set up, I entered the command "python manage.py runserver" on my terminal to view my api result.



127.0.0.1:8000/apptwo/api/

Django REST framework

bisoyefemi

Api Root

Api Root

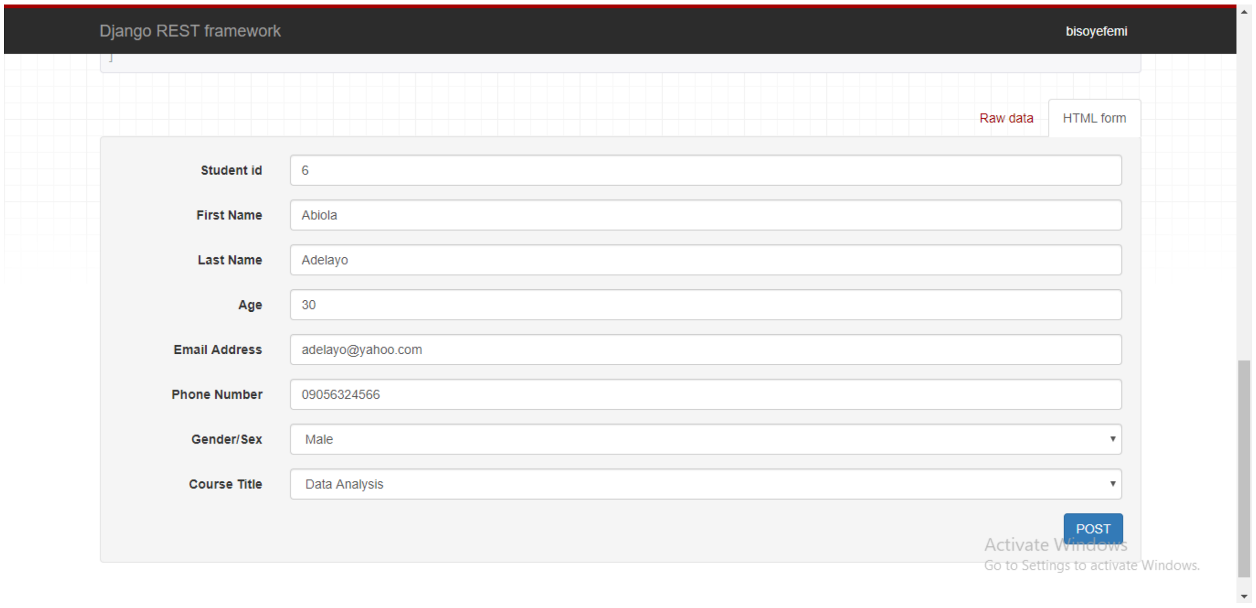
The default basic root view for DefaultRouter

```
GET /apptwo/api/
```

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

{
    "students": "http://127.0.0.1:8000/apptwo/api/students/",
    "coachs": "http://127.0.0.1:8000/apptwo/api/coachs/",
    "tasks": "http://127.0.0.1:8000/apptwo/api/tasks/"
}
```

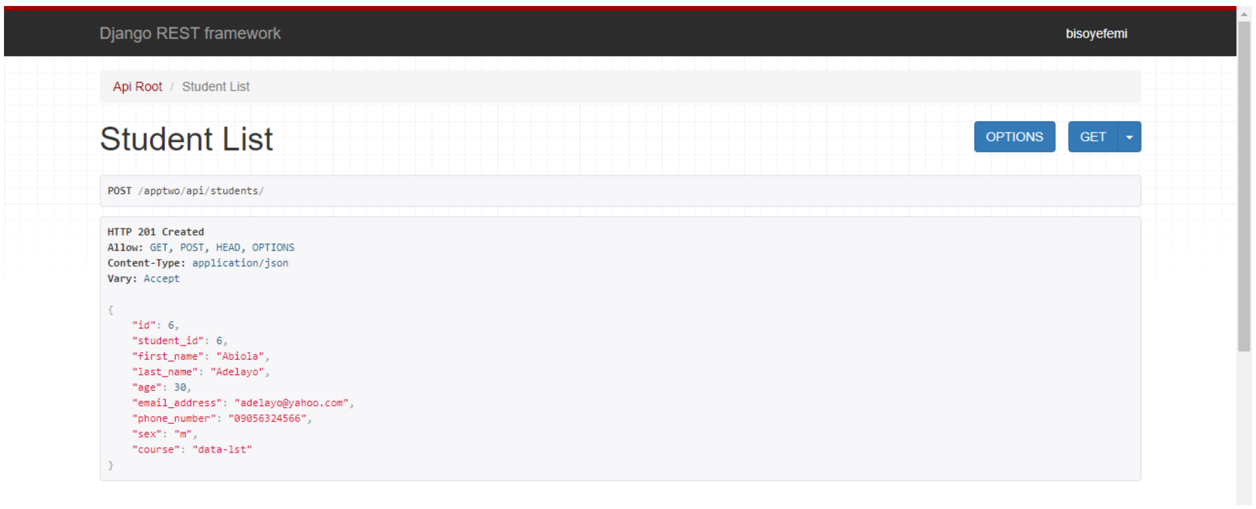
Figure 2.3: Screenshot showing API Routers for Student, Coach and Task Model

To Post/Create a new record:



Student id	6
First Name	Abiola
Last Name	Adelayo
Age	30
Email Address	adelayo@yahoo.com
Phone Number	09056324566
Gender/Sex	Male
Course Title	Data Analysis

Figure 2.4: Screenshot showing POST request field for Student API



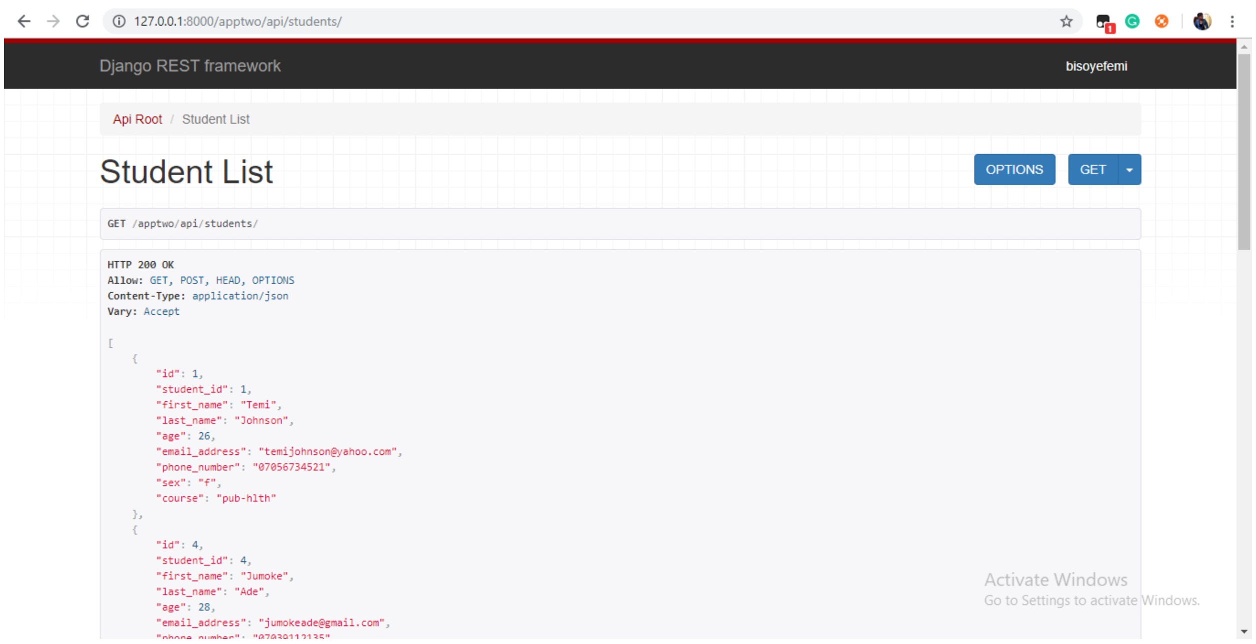
Student List

```
POST /apptwo/api/students/
```

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

{
    "id": 6,
    "student_id": 6,
    "first_name": "Abiola",
    "last_name": "Adelayo",
    "age": 30,
    "email_address": "adelayo@yahoo.com",
    "phone_number": "09056324566",
    "sex": "m",
    "course": "data-1st"
}
```

Figure 2.5: Screenshot showing successful POST record

To Read a Record:



127.0.0.1:8000/apptwo/api/students/

Django REST framework bisoyefemi

Api Root / Student List

Student List

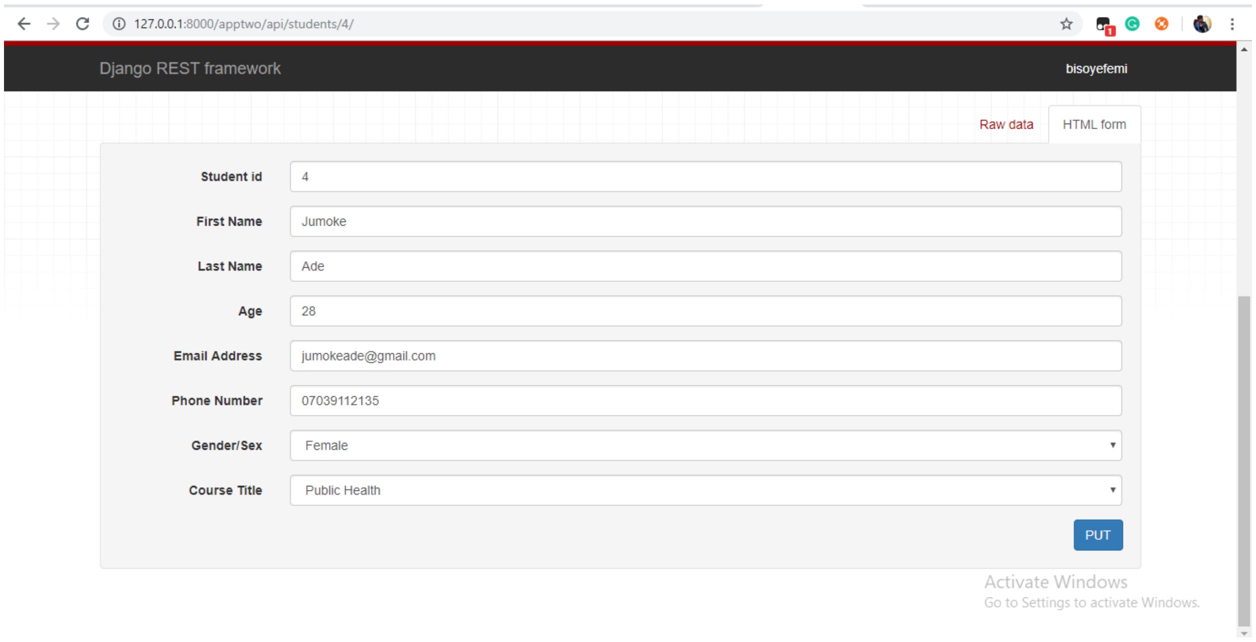
OPTIONS GET

```
GET /apptwo/api/students/
```

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

[
    {
        "id": 1,
        "student_id": 1,
        "first_name": "Temi",
        "last_name": "Johnson",
        "age": 26,
        "email_address": "temijohnson@yahoo.com",
        "phone_number": "07056734521",
        "sex": "f",
        "course": "pub-hlth"
    },
    {
        "id": 4,
        "student_id": 4,
        "first_name": "Jumoke",
        "last_name": "Ade",
        "age": 28,
        "email_address": "jumokeade@gmail.com",
        "phone_number": "07039112135"
```

Figure 2.6: Screenshot showing the all records in the Student List

To Update a Record:



127.0.0.1:8000/apptwo/api/students/4/

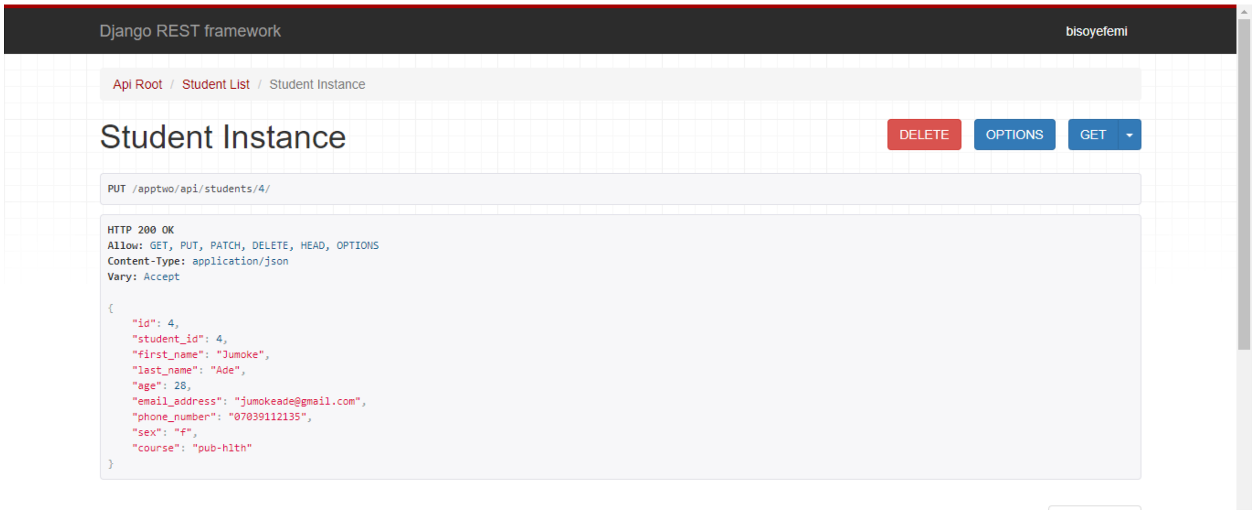
Django REST framework bisoyefemi

Raw data HTML form

Student id	4
First Name	Jumoke
Last Name	Ade
Age	28
Email Address	jumokeade@gmail.com
Phone Number	07039112135
Gender/Sex	Female
Course Title	Public Health

PUT

Figure 2.7: Screenshot showing a PUT request to update a record

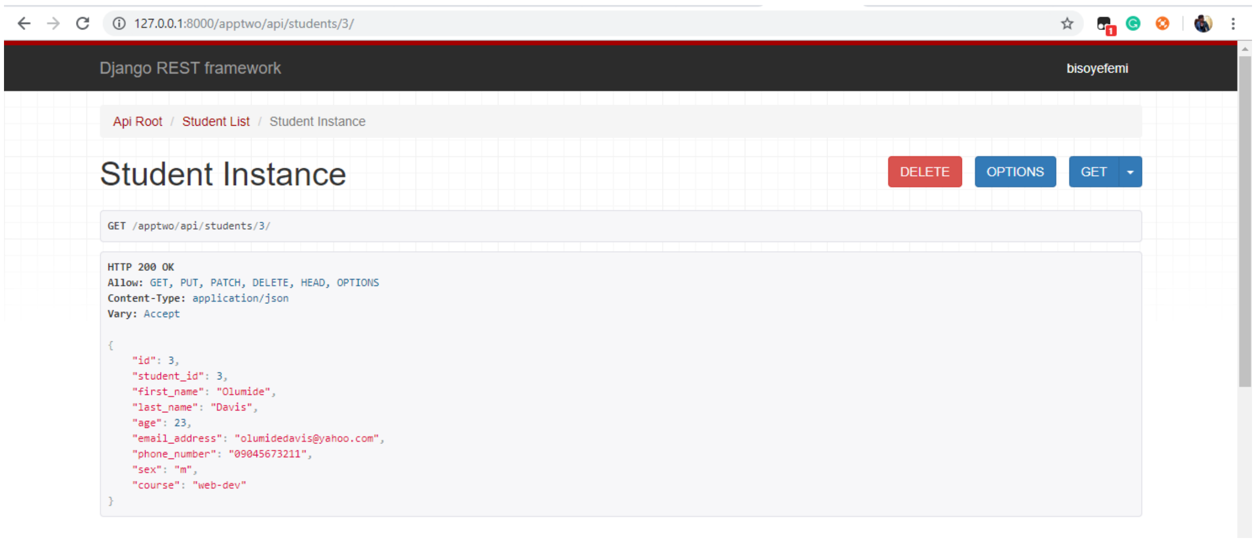


```
PUT /apptwo/api/students/4/
```

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

{
    "id": 4,
    "student_id": 4,
    "first_name": "Jumoke",
    "last_name": "Ade",
    "age": 28,
    "email_address": "jumokeade@gmail.com",
    "phone_number": "07039112135",
    "sex": "f",
    "course": "pub-hlth"
}
```

Figure 2.8: Screenshot of successful update of record for student_id (4)

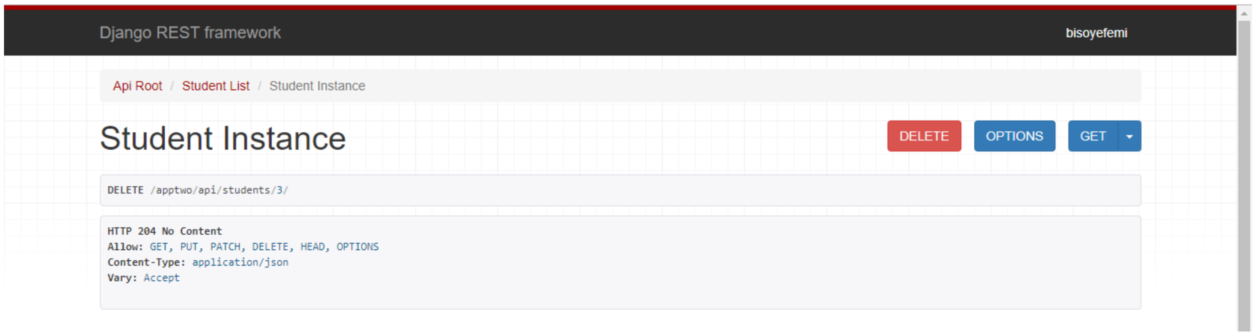
To Delete a Record:


127.0.0.1:8000/apptwo/api/students/3/
Django REST framework
bisoyefemi
Api Root / Student List / Student Instance
Student Instance
DELETE
OPTIONS
GET

```
GET /apptwo/api/students/3/
```

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

{
    "id": 3,
    "student_id": 3,
    "first_name": "Olumide",
    "last_name": "Davis",
    "age": 23,
    "email_address": "olumidedavis@yahoo.com",
    "phone_number": "09045673211",
    "sex": "m",
    "course": "web-dev"
}
```

Figure 2.9: Screenshot for student instance 3 to initiate a delete

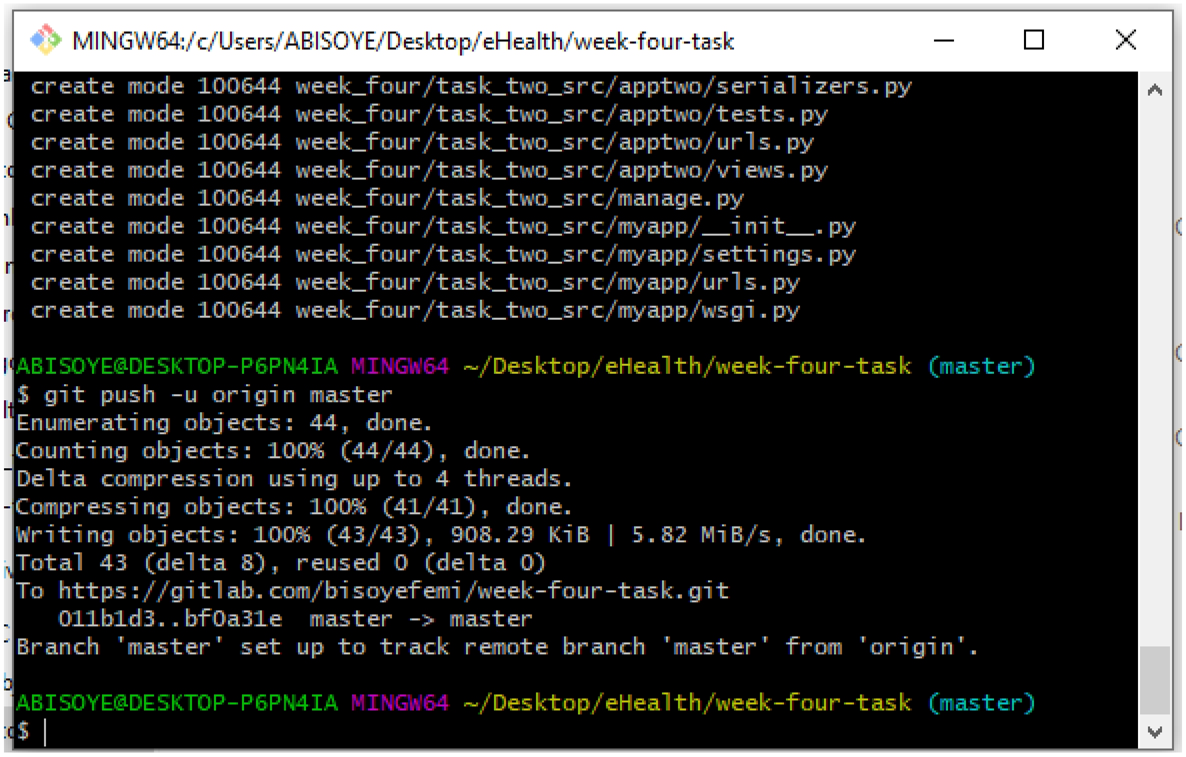


Student Instance

```
DELETE /apptwo/api/students/3/
```

```
HTTP 204 No Content
Allow: GET, PUT, PATCH, DELETE, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept
```

Figure 3.0: Screenshot of deleted student instance 3



```
 create mode 100644 week_four/task_two_src/apptwo/serializers.py
 create mode 100644 week_four/task_two_src/apptwo/tests.py
 create mode 100644 week_four/task_two_src/apptwo/urls.py
 create mode 100644 week_four/task_two_src/apptwo/views.py
 create mode 100644 week_four/task_two_src/manage.py
 create mode 100644 week_four/task_two_src/myapp/__init__.py
 create mode 100644 week_four/task_two_src/myapp/settings.py
 create mode 100644 week_four/task_two_src/myapp/urls.py
 create mode 100644 week_four/task_two_src/myapp/wsgi.py

ABISOYE@DESKTOP-P6PN4IA MINGW64 ~/Desktop/eHealth/week-four-task (master)
$ git push -u origin master
Enumerating objects: 44, done.
Counting objects: 100% (44/44), done.
Delta compression using up to 4 threads.
Compressing objects: 100% (41/41), done.
Writing objects: 100% (43/43), 908.29 KiB | 5.82 MiB/s, done.
Total 43 (delta 8), reused 0 (delta 0)
To https://gitlab.com/bisoyefemi/week-four-task.git
   011b1d3..bf0a31e  master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.

ABISOYE@DESKTOP-P6PN4IA MINGW64 ~/Desktop/eHealth/week-four-task (master)
$ 
```

Figure 3.1: Screenshot of project pushed successfully