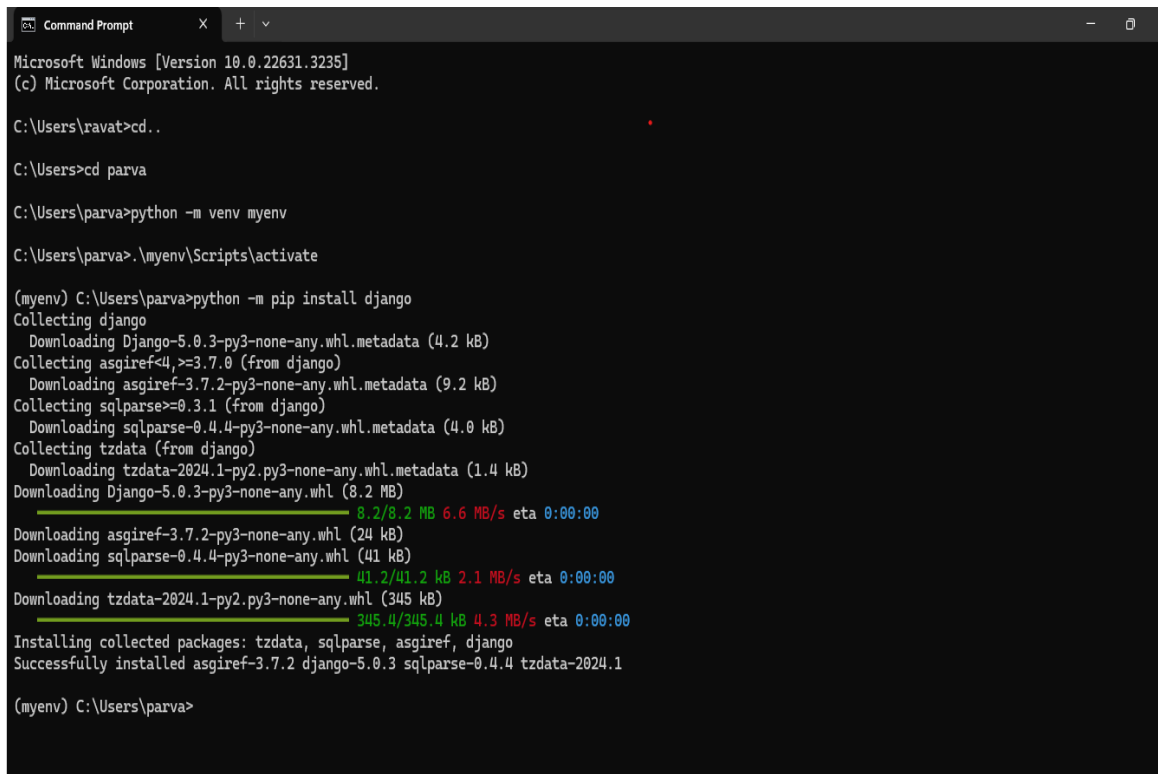


DJANGO TASK DAY - 2

1) Create a virtual environment and install django in it - attach a screen shot.

Steps:

- Python -m venv env_name
- .\env_name\Scripts\activate
- Python -m pip install Django



```
Microsoft Windows [Version 10.0.22631.3235]
(c) Microsoft Corporation. All rights reserved.

C:\Users\ravat>cd..

C:\Users>cd parva

C:\Users\parva>python -m venv myenv

C:\Users\parva>.\myenv\Scripts\activate

(myenv) C:\Users\parva>python -m pip install django
Collecting django
  Downloading Django-5.0.3-py3-none-any.whl.metadata (4.2 kB)
Collecting asgiref<4,>=3.7.0 (from django)
  Downloading asgiref-3.7.2-py3-none-any.whl.metadata (9.2 kB)
Collecting sqlparse>=0.3.1 (from django)
  Downloading sqlparse-0.4.4-py3-none-any.whl.metadata (4.0 kB)
Collecting tzdata (from django)
  Downloading tzdata-2024.1-py2.py3-none-any.whl.metadata (1.4 kB)
Downloading Django-5.0.3-py3-none-any.whl (8.2 MB)
   8.2/8.2 MB 6.6 MB/s eta 0:00:00
Downloading asgiref-3.7.2-py3-none-any.whl (24 kB)
Downloading sqlparse-0.4.4-py3-none-any.whl (41 kB)
   41.2/41.2 kB 2.1 MB/s eta 0:00:00
Downloading tzdata-2024.1-py2.py3-none-any.whl (345 kB)
   345.4/345.4 kB 4.3 MB/s eta 0:00:00
Installing collected packages: tzdata, sqlparse, asgiref, django
Successfully installed asgiref-3.7.2 django-5.0.3 sqlparse-0.4.4 tzdata-2024.1

(myenv) C:\Users\parva>
```

2) Create a django project and app, explain the files of the project and app folders.

Steps:

- django-admin startproject projectname
- django-admin startapp appname
- python manage.py migrate

Project Folder

1. Manage.py

- This file is used basically as a command-line utility and for deploying, debugging, or running our web application. This file contains code for runserver, or makemigrations or migrations, etc. that we use in the shell. Anyway, we do not need to make any changes to the file.

- **runserver:** This command is used to run the server for our web application.
- **Migration:** This is used for applying the changes done to our models into the database. That is if we make any changes to our database then we use migrate command. This is used the first time we create a database.
- **Makemigration:** this is done to apply new migrations that have been carried out due to the changes in the database.

2. `_init_.py`

This file remains empty and is present there only to tell that this particular directory (in this case `django_project`) is a package.

3. `setting.py`

This file is present for adding all the applications and the middleware application present. Also, it has information about templates and databases. Overall, this is the main file of our Django web application.

4. `urls.py`

This file handles all the URLs of our web application. This file has the lists of all the endpoints that we will have for our website.

URL: Universal Resource Locator is used to provide the addresses of the resources (like image, website, etc) that are present there on the internet.

5. `wsgi.py`

This file mainly concerns with the WSGI server and is used for deploying our applications on to servers like Apache etc.

WSGI, short for Web Server Gateway Interface can be thought of as a specification that describes how the servers interact with web applications.

6. `asgi.py`

In the newer versions of Django, you will also find a file named as `asgi.py` apart from `wsgi.py`. ASGI can be considered as a successor interface to the WSGI.

ASGI, short for Asynchronous Server Gateway interface also has the work like WSGI but this is better than the previous one as it gives better freedom in Django development. That's why WSGI is now being increasingly replaced by ASGI.

APP Folder

Apart from the above file, our project contains all the app directories. Now we will look into the Django app structure in detail.

1. `_init_.py`

This file has the same functionality just as in the `_init_.py` file in the Django project structure. It remains empty and is present just to indicate that the specific app directory is a package.

2. admin.py

As the name suggests, this file is used for registering the models into the Django administration. The models that are present have a superuser/admin who can control the information that is being stored.

3. apps.py

This file deals with the application configuration of the apps. The default configuration is sufficient in most of the cases and hence we won't be doing anything here in the beginning.

4. models.py

This file contains the models of our web applications (usually as classes). Models are basically the blueprints of the database we are using and hence contain the information regarding attributes and the fields etc of the database.

5. views.py

This file is a crucial one, it contains all the Views (usually as classes). Views.py can be considered as a file that interacts with the client. Views are a user interface for what we see when we render a Django Web application.

6. urls.py

Just like the project `urls.py` file, this file handles all the URLs of our web application. This file is just to link the Views in the app with the host web URL. The settings `urls.py` has the endpoints corresponding to the Views.

7. tests.py

This file contains the code that contains different test cases for the application. It is used to test the working of the application.

We won't be working on this file in the beginning and hence it is going to be empty as of now.

3) Run the django project - attach screen shots of the Django admin panel and SITE.

Steps:

- `python manage.py createsuperuser`
- `python manage.py runserver`

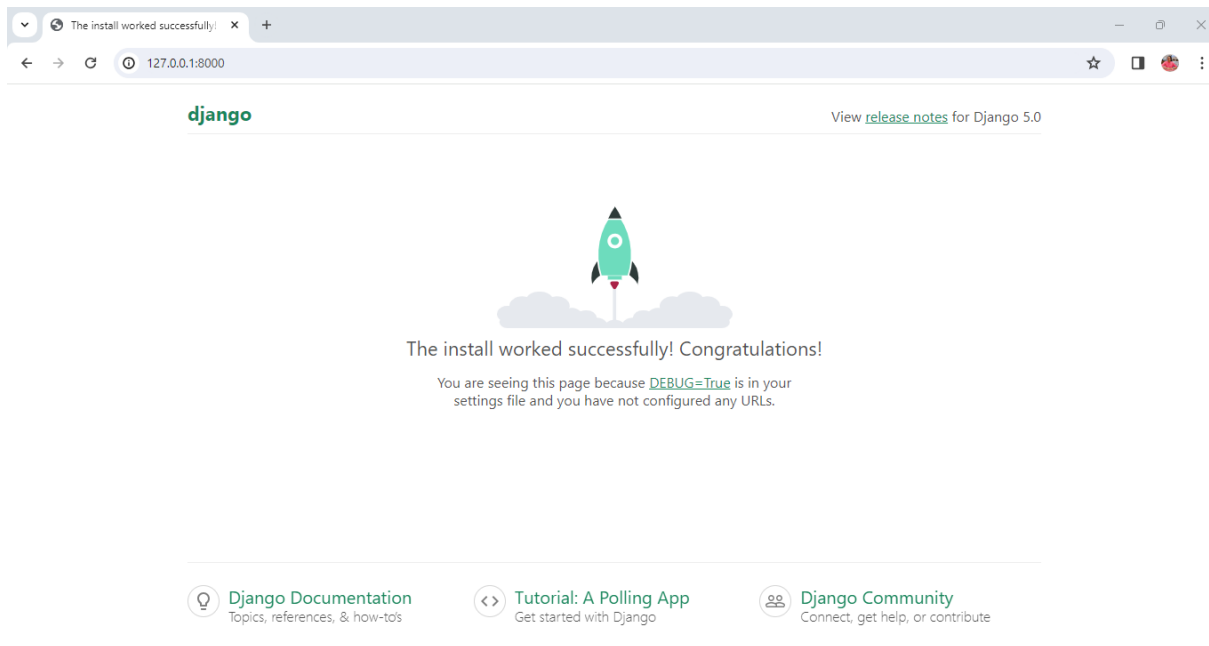


Fig: 1 Site

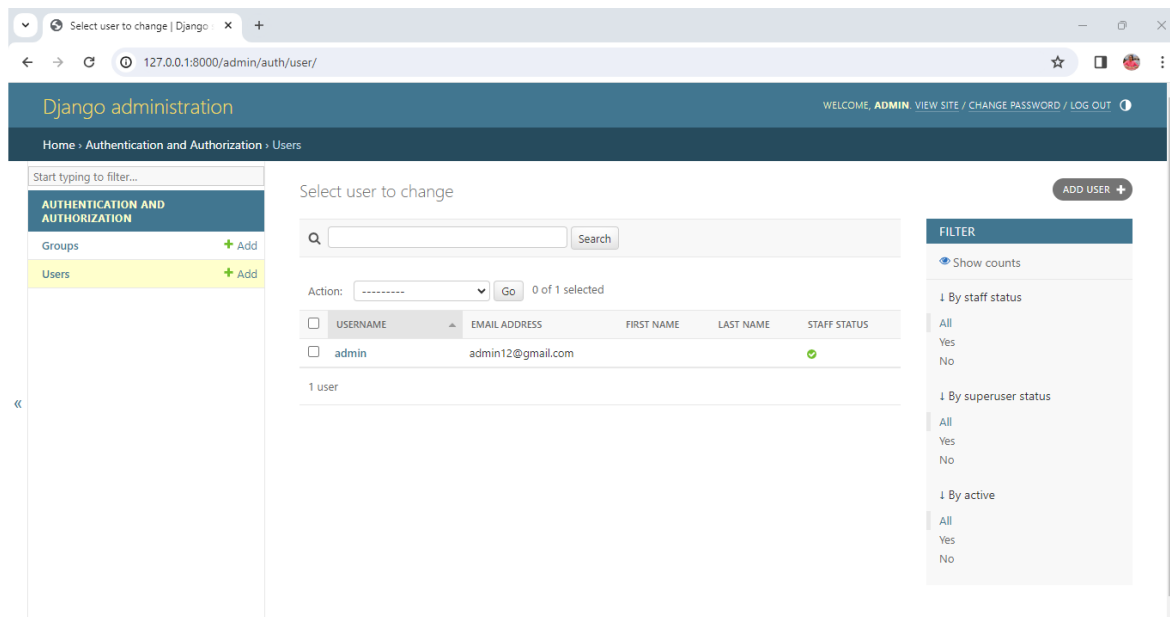


Fig: 2 Django admin panel