UNIT-I

1. Introduction to Django
Ans: Python is a versatile programming language known for its simplicity, enabling problem-solving over implementation details.

- Python's open-source nature grants access to a wide range of free libraries, frameworks, and modules.
- Django, a powerful web application development framework built on Python, offers a comprehensive solution for creating web applications.
- Django is freely available and open source, allowing developers to utilize its functionalities without cost.
- Django provides extensive features and capabilities to enhance web application development efficiently.

2. Django Overview
Ans: Django is renowned for its ability to rapidly create web applications.

- Django includes built-in middleware and essential components for running web applications.
- Developers can focus on application development using Django's elegant settings and configurable connections.
- Django was created by web developers at Lawrence Journal-World and released under a BSD license in July 2005.
- Django is well-documented, and staying up-to-date with the latest version is recommended for optimal performance and new features.

3. What is a framework?
Ans: A software framework is like a frame that provides a setup for developers to easily configure and get their job done quickly and with ease.

- Web application development has common requirements that a framework efficiently handles, including authentication, authorization, database connectivity, frontend HTML pages, CSS styling, URL mapping, and request handling.
- Frameworks provide a foundation to easily fulfil web application requirements and save time compared to building from scratch.
- Frameworks offer a structured approach, promoting consistency and organization in web application development.
- Developers can focus on specific functionality without worrying about infrastructure details.
- Frameworks streamline development, promote code reusability, and facilitate maintenance and scalability of applications.

4. Why do we need a framework?
Ans: Frameworks provide a standardized software design structure, facilitating maintenance even when developers change.

- Common functionalities are built-in, reducing code rewriting and enabling easy customization.
- Frameworks undergo extensive testing and continuous bug fixes, ensuring stable and bug-free software.
- Frameworks offer regular updates with new features, keeping software up-to-date without manual intervention.
- Web frameworks provide options for web application development, allowing developers to choose based on specific requirements, programming language preferences, community support, scalability needs, and available resources.

5. What are some famous web frameworks?

Ans: Apart from Django, other web application frameworks available in the industry include Flask, Ruby on Rails, Angular, ASP.NET, Spring, and Play.

- **Flask** is a Python-based web application framework that uses the WSGI toolkit and Jinja2 template engine.
- **Ruby on Rails** is a productive web application framework based on Ruby, known for its faster development compared to Java frameworks.
- **Angular,** developed by Google, is a JavaScript framework for building powerful and maintainable web applications.
- **ASP.NET** is a framework developed by Microsoft for building robust web applications for PCs and mobile devices, using .NET.
- **Spring** is a popular enterprise Java development framework known for its high performance and robustness.
- **Play** is a modern web application framework written in Java and Scala, following the MVC architecture and emphasizing developer productivity.
- Other notable frameworks include Laravel, Symfony, CodeIgniter, Express, React.js, Node.js, and Hibernate.

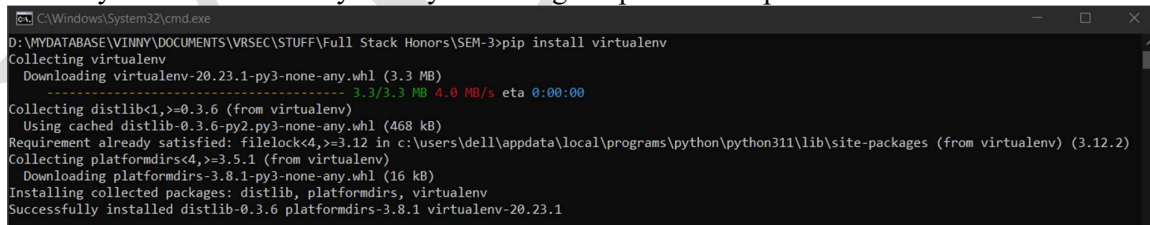### 6. What is a virtual environment?

Ans: Pre-written code in software or web development is called libraries, packages, addons, plugins, etc.

- Frameworks offer pre-written code to implement various features in applications.
- Installation of required libraries and plugins is necessary to utilize pre-written code.
- Different applications require specific frameworks, libraries, and plugins for different features.
- Virtual environments ensure separate setups for each project, avoiding conflicts and enabling simultaneous running of multiple projects with separate interpreters.

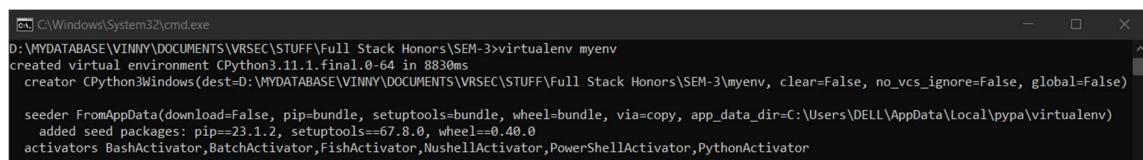### 7. How to create and use a virtual environment?

Ans: First install Python by visiting the official Python website, going to the Downloads tab, selecting your operating system, downloading the Python installable while ensuring to checkmark the "Add Python to the PATH" option, and then running the installable file, choosing the installation location and selecting the checkbox to configure your operating system to use this Python installation as the default if no other installations are specified.

- Open Windows Power Shell or CMD, run the install command (without quotes) to install the virtual environment on your machine, and if you encounter errors such as "pip not found," you may need to reinstall Python by following the previous steps or uninstall and reinstall it.

- After the installation completes, create a virtual environment by entering the following command

- Activate the virtual environment by running the following command

```
C:\Windows\System32\cmd.exe                                                    —    □    ×

D:\MYDATABASE\VINNY\DOCUMENTS\VRSEC\STUFF\Full Stack Honors\SEM-3>myenv\Scripts\activate

(myenv) D:\MYDATABASE\VINNY\DOCUMENTS\VRSEC\STUFF\Full Stack Honors\SEM-3>
```

- Once the virtual environment is activated, use the following command to install NumPy using pip and once the installation completes, go to your lib/site-packages folder and check for numpy installations

```
C:\Windows\System32\cmd.exe                                                    —    □    ×

(myenv) D:\MYDATABASE\VINNY\DOCUMENTS\VRSEC\STUFF\Full Stack Honors\SEM-3>pip install numpy
Collecting numpy
  Downloading numpy-1.25.1-cp311-cp311-win_amd64.whl (15.0 MB)
     ---------------------------------------- 15.0/15.0 MB 4.5 MB/s eta 0:00:00
Installing collected packages: numpy
Successfully installed numpy-1.25.1
```

- To exit the virtual environment, simply enter the following command

```
C:\Windows\System32\cmd.exe                                                    —    □    ×

(myenv) D:\MYDATABASE\VINNY\DOCUMENTS\VRSEC\STUFF\Full Stack Honors\SEM-3>deactivate
D:\MYDATABASE\VINNY\DOCUMENTS\VRSEC\STUFF\Full Stack Honors\SEM-3>
```

8. How to install Django?

Ans: First activate your virtual environment and then type the below command to install Django

```
C:\Windows\System32\cmd.exe

(myenv) D:\MYDATABASE\VINNY\DOCUMENTS\VRSEC\STUFF\Full Stack Honors\SEM-3>pip install django
Collecting django
  Downloading Django-4.2.3-py3-none-any.whl (8.0 MB)
     ---------------------------------------- 8.0/8.0 MB 4.6 MB/s eta 0:00:00
Collecting asgiref<4,>=3.6.0 (from django)
  Downloading asgiref-3.7.2-py3-none-any.whl (24 kB)
Collecting sqlparse>=0.3.1 (from django)
  Downloading sqlparse-0.4.4-py3-none-any.whl (41 kB)
     ---------------------------------------- 41.2/41.2 kB 1.9 MB/s eta 0:00:00
Collecting tzdata (from django)
  Downloading tzdata-2023.3-py2.py3-none-any.whl (341 kB)
     ---------------------------------------- 341.8/341.8 kB 4.2 MB/s eta 0:00:00
Installing collected packages: tzdata, sqlparse, asgiref, django
Successfully installed asgiref-3.7.2 django-4.2.3 sqlparse-0.4.4 tzdata-2023.3
```

- If you wish to install any other version of Django, then use the **pip install Django==x.x.x** command where x.x.x is the version number.

9. How to create a Django project?

Ans: In general, an application is a component that performs a specific task, like a music player web application that plays music. However, when using Django, each of these smaller features, such as playlists and popular songs, is considered a separate application. These applications are then integrated and combined within a Django project, such as the "Music Player" project. In Django, a project can consist of one or more applications that work together to accomplish the desired functionality.

- Check whether Django is installed or not by entering the following command

```
C:\Windows\System32\cmd.exe

(myenv) D:\MYDATABASE\VINNY\DOCUMENTS\VRSEC\STUFF\Full Stack Honors\SEM-3>python -m django --version
4.2.3
```

- Run the following command to create a project named mysite in your Project_root directory

```
C:\Windows\System32\cmd.exe

(myenv) D:\MYDATABASE\VINNY\DOCUMENTS\VRSEC\STUFF\Full Stack Honors\SEM-3>django-admin startproject mysite
```
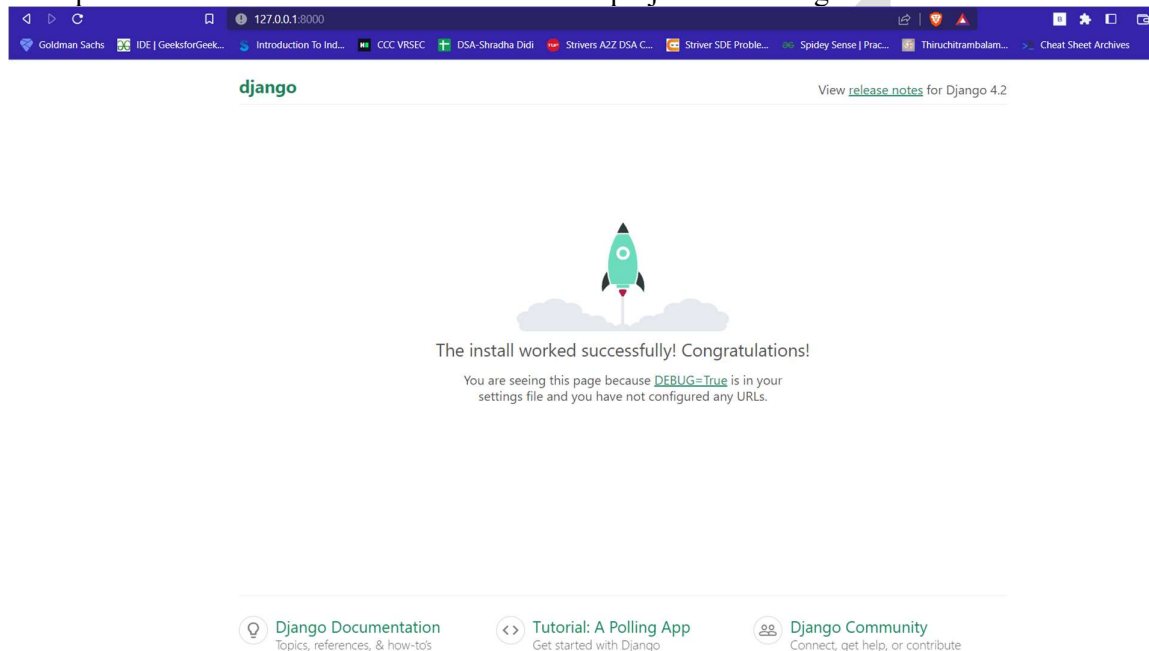
10. Overview of your Django project files

Ans: Move to the mysite folder and execute the manager.py server command to initiate a server at the default host and port

- Open the server in the browser to check if the project is working fine or not

- Stop the server by pressing Ctrl + C together and run the following command to see the list of Django files and directories in the current directory.

- The **db.sqlite3** file is the default SQLite database file used by Django, containing all the database entries, while the **manage.py** file is responsible for running Django-admin scripts, and the **mysite** project folder inside the root folder is the main folder containing internal project files, with a separate internal Django project folder that should not be confused or renamed.
- In the internal Django project folder, you will find files such as **settings.py** for project configurations, **urls.py** for URL mappings and corresponding actions, **wsgi.py** for creating a web

server gateway interface, and **init.py** files indicating that the folders can be used as packages, along with the **pycache** folder.

```
C:\Windows\System32\cmd.exe

(myenv) D:\MYDATABASE\VINNY\DOCUMENTS\VRSEC\STUFF\Full Stack Honors\SEM-3\mysite>cd mysite

(myenv) D:\MYDATABASE\VINNY\DOCUMENTS\VRSEC\STUFF\Full Stack Honors\SEM-3\mysite\mysite>dir
 Volume in drive D is PERSONAL
 Volume Serial Number is 0458-F44C

 Directory of D:\MYDATABASE\VINNY\DOCUMENTS\VRSEC\STUFF\Full Stack Honors\SEM-3\mysite\mysite

14-07-2023  22:58    <DIR>          .
14-07-2023  22:58    <DIR>          ..
14-07-2023  22:52               405 asgi.py
14-07-2023  22:52             3,344 settings.py
14-07-2023  22:52               784 urls.py
14-07-2023  22:52               405 wsgi.py
14-07-2023  22:52                 0 __init__.py
14-07-2023  22:58    <DIR>          __pycache__
               5 File(s)          4,938 bytes
               3 Dir(s)  134,111,428,608 bytes free
```

11. What is architecture?
Ans: A proper setup or design is needed to connect the components of a project and ensure optimal performance.
- Web applications consist of components such as a database, backend, frontend, server, and network connections.
- Django's MVT (Model-View-Template) architecture fulfils the requirement of connecting the database to the backend, rendering data on the frontend, and maintaining communication flow.
- MVT is a variation of the commonly used MVC (Model-View-Controller) architecture.
- Prior knowledge of MVC or any other architecture is not required to understand and learn MVT with Django.
- A web application works by the browser sending a request to fetch data for a given URL, the server receiving and evaluating the request, retrieving data from the database, generating an HTML response, and the browser decoding and displaying the data.

12. What is MVT?
Ans: Django's MVT architecture handles requests from the client (browser) to the server (where code is present) for a Django-built website.
- The server receives the client's request, looks for the requested URL in the urls.py file, and maps it to a corresponding view function in views.py.
- The view function performs operations and returns an HTML template from the Templates folder as a response to the client's request.
- The 'M' in MVT represents the Models.py file, used to create tables in the database and handle database queries within the Django project.
- Before rendering the HTML response, the view function connects to the database, fetches the necessary information, and creates the HTML response to be rendered in the browser.
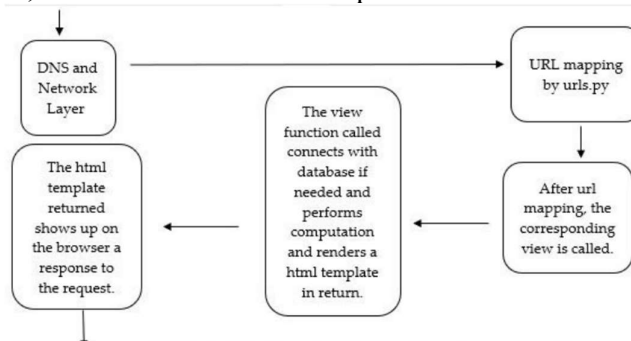
13. What are models in Django's MVT?

Ans: Models in an application describe the type of data to be stored and serve as representations of database tables.

- Each model class corresponds to a database table, and the attributes in the class represent the columns of the table.
- Models provide an easy way to interact with database tables and rows using classes and objects, simplifying database operations.
- Creating an object of a model class is similar to making an entry into a database table, with the object representing a row in the table.
- Models facilitate handling database operations in code, regardless of the programming language used, making it convenient to store and retrieve data.
- ORM (Object-Relational Mapping) is a programming paradigm that allows interaction with databases using object-oriented concepts
- Django provides an inbuilt ORM library, eliminating the need to learn a separate ORM library and offering the advantage of streamlined database operations within the framework.

  Ex:

  SQL Query:

```
CREATE TABLE Users (id INT PRIMARY KEY, name VARCHAR(50), age INT,
email VARCHAR(100));
```

  Model Class:

```
from django.db import models
class User(models.Model):
    id = models.IntegerField(primary_key=True)
    name = models.CharField(max_length=50)
    age = models.IntegerField()
    email = models.EmailField(max_length=100)
```

14. What are Views in Django's MVT?

Ans: In Django's MVT architecture, views serve as the controllers where logic and computations are implemented for rendering the response, and the urls.py file maps requested URLs to view functions defined in the views.py file.

- In the urls.py file, the urlpatterns list contains path function calls that map expected URLs to view functions in the views.py file.
- Regular expressions (regex) are used to define the expected URLs in the first argument of the path function calls.
- When a URL matches the expected URL, the corresponding view function is called to handle the request.
- The view functions, located in the views.py file, execute logic and return a template as a response to the request.
- If no path function calls match the URL in the request, Django renders the built-in 404 error template (page not found).

  Ex:

  urls.py:

```
from django.urls import path
from . import views
urlpatterns = [
    path('', views.home, name='home'),
    path('about/', views.about, name='about'),]
```

  views.py:

```
from django.shortcuts import render
```

```
from django.http import HttpResponse
def home(request):
    return HttpResponse("Welcome to the home page!")
def about(request):
    return render(request, 'about.html')
```

15. What are Templates in Django's MVT?

Ans: Templates in Django allow you to write frontend HTML code and manipulate the content using Python code.

- Templates are a combination of Python and HTML code, written in a special way using tags.
- Django provides a default template engine, but you can also use other template engines like Jinja2 or Mako.
- Templates empower you to dynamically generate HTML content, implement loops, conditionals, and reuse common HTML sections.
- Templating in Django enables you to handle static frontend dynamically through Python code, improving flexibility and reducing repetition.

  Ex:
  ```
  <!DOCTYPE html>
  <html>
  <head>
      <title>My First Django Template</title>
  </head>
  <body>
      <h1>Welcome to {{ website_name }}</h1>
      <ul>
          {% for item in items %}
              <li>{{ item }}</li>
          {% endfor %}
      </ul>
      {% if user_logged_in %}
          <p>Welcome, {{ username }}!</p>
      {% else %}
          <p>Please log in to continue.</p>
      {% endif %}
  </body>
  </html>
  ```
- The **{{ website_name }}** is a placeholder that will be replaced with the actual website name.
- The **{% for item in items %}** and **{% endfor %}** tags create a loop to dynamically generate list items based on the **items** list.
- The **{% if user_logged_in %}** and **{% else %}** tags conditionally display different content based on whether the user is logged in or not.

16. Introduction to Django Admin Utility

Ans: Django provides an admin portal that allows administrators to control the application without the need for additional development.

- The admin portal is included with the Django framework and can be used immediately.
- The admin portal grows along with the application and can be configured and customized as needed.
- The admin page provides a working ORM and allows you to interact with the database and manage data through a user-friendly interface.

17. Admin page of your Django project

Ans: Python provides a built-in admin app in Django for developers to use, which can be found in the list of installed apps in the settings.py file of the project.

- Enter the following command to check for any changes in the models.py file.

```
C:\Windows\System32\cmd.exe                                          —    □    ×
(myenv) D:\MYDATABASE\VINNY\DOCUMENTS\VRSEC\STUFF\Full Stack Honors\SEM-3\mysite>python manage.py makemigrations
No changes detected
```

- The following command is used to implement model classes and update the database accordingly.

```
C:\Windows\System32\cmd.exe                                          —    □    ×
(myenv) D:\MYDATABASE\VINNY\DOCUMENTS\VRSEC\STUFF\Full Stack Honors\SEM-3\mysite>python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, sessions
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying admin.0002_logentry_remove_auto_add... OK
  Applying admin.0003_logentry_add_action_flag_choices... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0002_alter_permission_name_max_length... OK
  Applying auth.0003_alter_user_email_max_length... OK
  Applying auth.0004_alter_user_username_opts... OK
  Applying auth.0005_alter_user_last_login_null... OK
  Applying auth.0006_require_contenttypes_0002... OK
  Applying auth.0007_alter_validators_add_error_messages... OK
  Applying auth.0008_alter_user_username_max_length... OK
  Applying auth.0009_alter_user_last_name_max_length... OK
  Applying auth.0010_alter_group_name_max_length... OK
  Applying auth.0011_update_proxy_permissions... OK
  Applying auth.0012_alter_user_first_name_max_length... OK
  Applying sessions.0001_initial... OK
```

- The following command creates a super user with ultimate admin access for the project, prompting for a user name, email id and password, which can be used to log in to the admin portal.

```
C:\Windows\System32\cmd.exe                                          —    □    ×
(myenv) D:\MYDATABASE\VINNY\DOCUMENTS\VRSEC\STUFF\Full Stack Honors\SEM-3\mysite>python manage.py createsuperuser
Username (leave blank to use 'dell'): user
Email address: swaguser@gmail.com
Password:
Password (again):
Superuser created successfully.
```

- Now start the server and add **/admin** to the end of the URL in your browser to see the login page
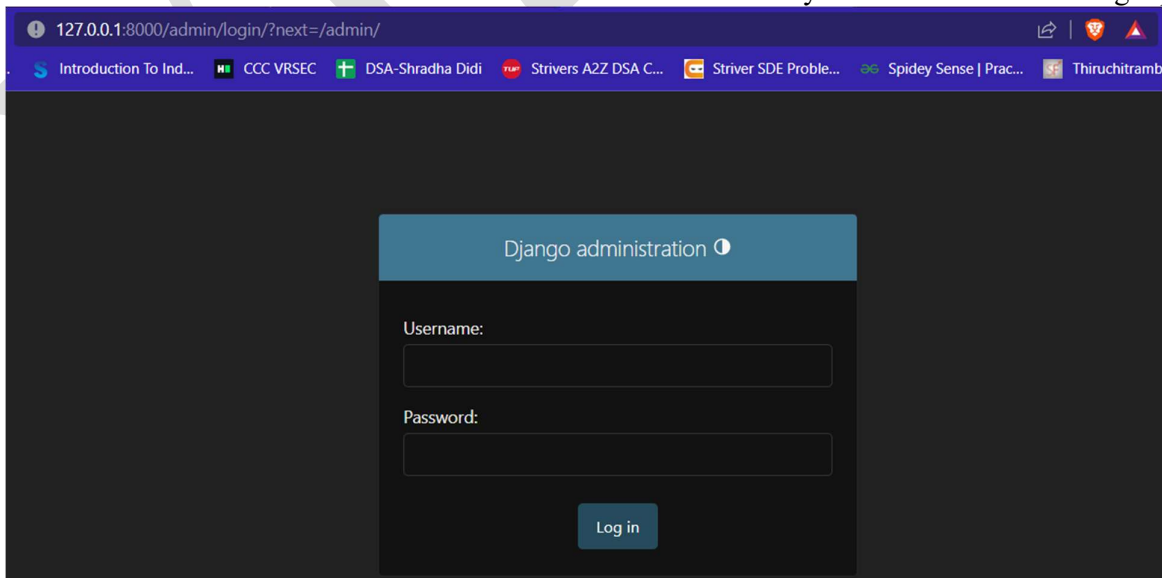

- Enter your credentials and click Log in to enter into the site administration page
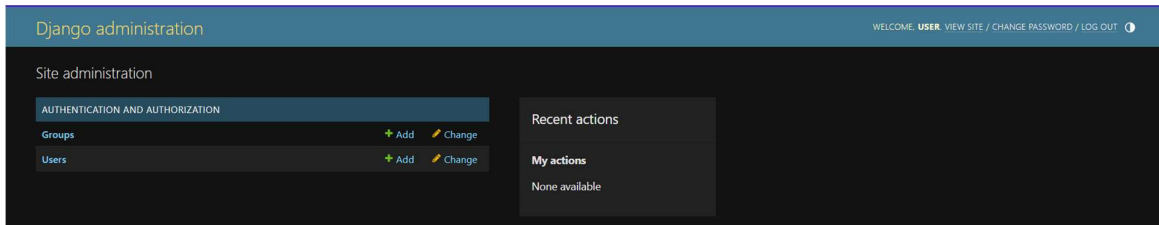
- Groups and Users showing up here are the two model classes of the admin app.
- You can create a user in the Users model, and it will create an object of the user model and then that object will be inserted in the database as a new row.
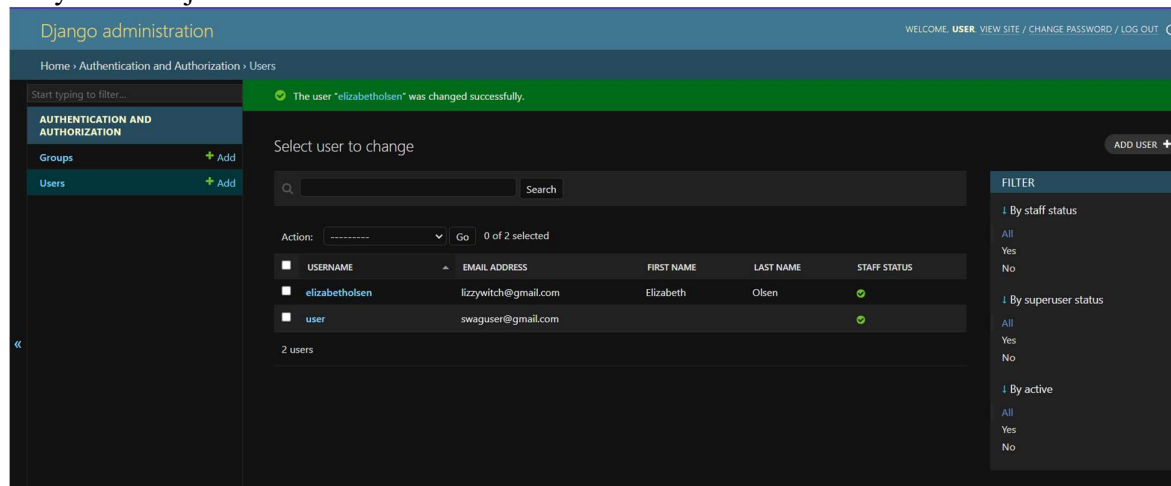- Click on the Add link corresponding to the user row.
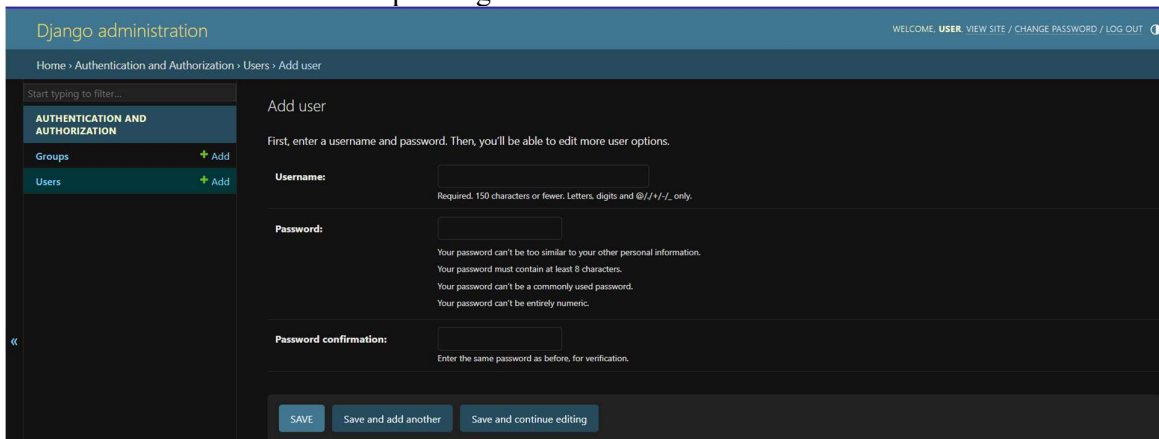
- Fill in the details and click on SAVE and then, you will see a screen where you can fill all the details for the user you just created.
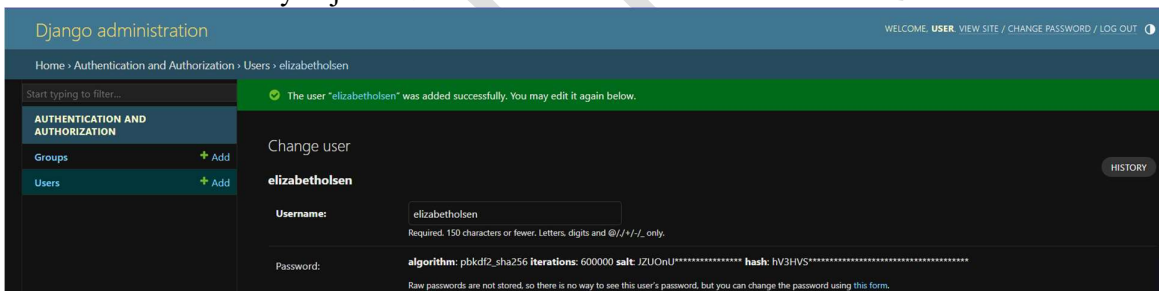
- You can set the permissions, access limits, etc.
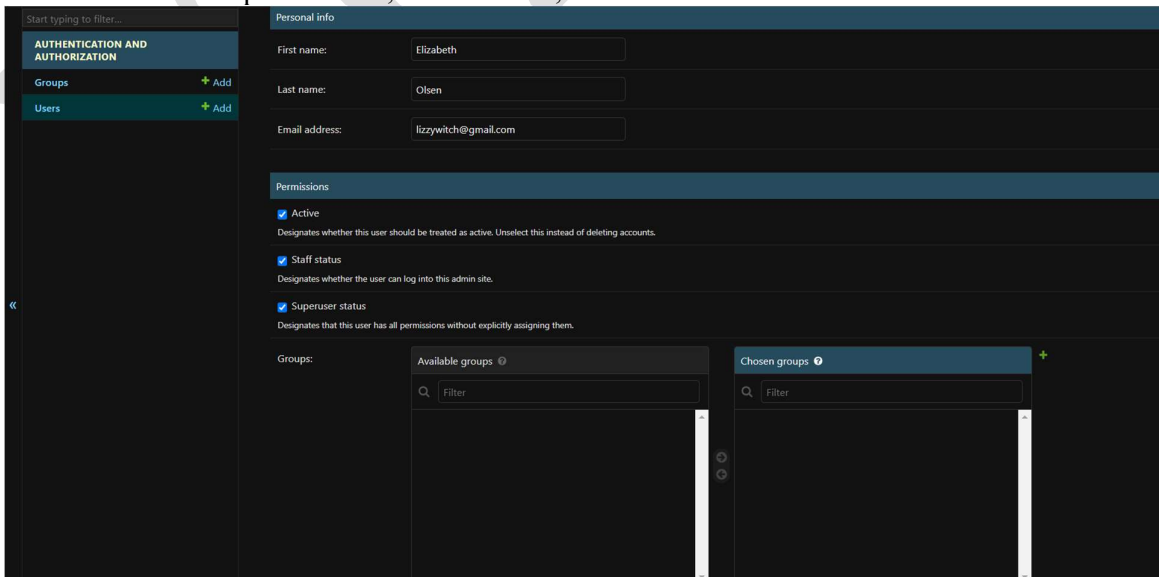
- Once done, go back to the site administration page to see the two users, superuser and the one you added just now.

18. Creating an app

Ans: In a Django project, multiple apps are used to solve smaller dedicated problems, contributing to the overall functionality of the project as a whole.

- There are two ways of creating an app: using the Django-admin tool, using the manage.py script.
- Go to your terminal, shut down the server and execute the following command

- This will create an app named student inside your Django project.
- The folder structure is as follows
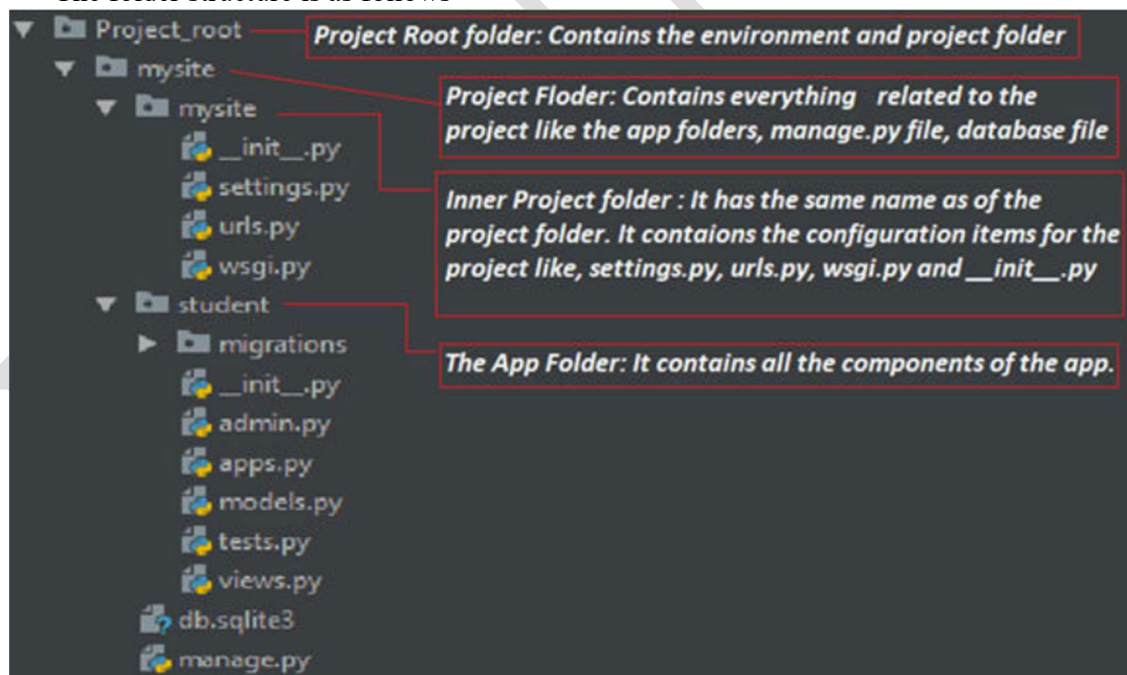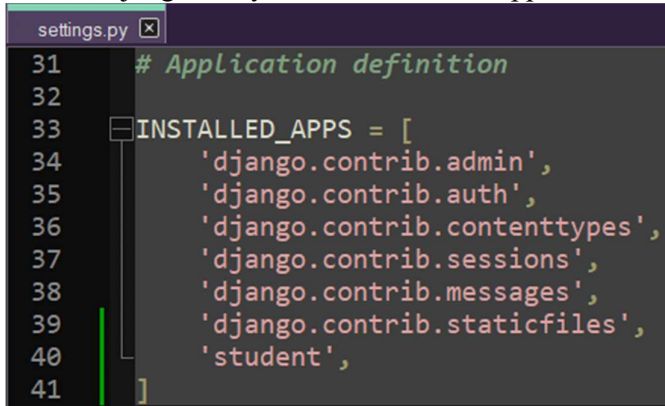
- **admin.py:** This is the file where you register the model classes of your app so that they can appear in the admin app.
- **apps.py:** Django has default configurations for apps, and while it's possible to make changes in the settings, it is recommended to proceed with caution and have a full understanding of the potential impact.
- **models.py:** This is the file where you define your model classes.

- **test.py:** In this file, you define your test cases for testing.
- **views.py:** This is where your logic is implemented and a response is generated.
- **migrations:** This file is used to capture changes made to model classes and store them in migration files, allowing for easy tracking of modifications and potential rollbacks in the future.
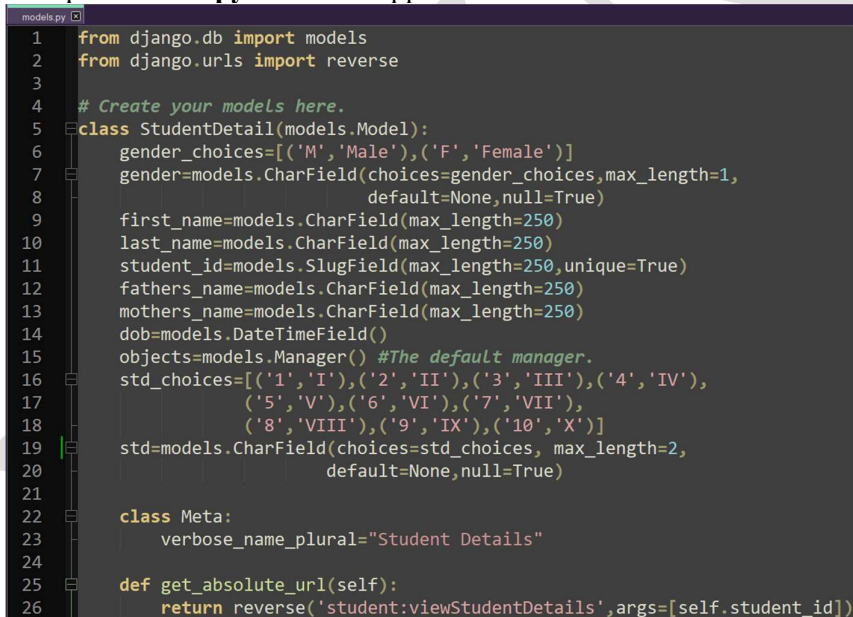
19. Editing models.py, settings.py, and admin.py.

Ans: Open **settings.py** in your project and add your app name in the list of INSTALLED_APPS as you need to tell Django that you have created an app

```
settings.py  ☒
31        # Application definition
32
33      ⊟INSTALLED_APPS = [
34            'django.contrib.admin',
35            'django.contrib.auth',
36            'django.contrib.contenttypes',
37            'django.contrib.sessions',
38            'django.contrib.messages',
39            'django.contrib.staticfiles',
40            'student',
41        ]
```
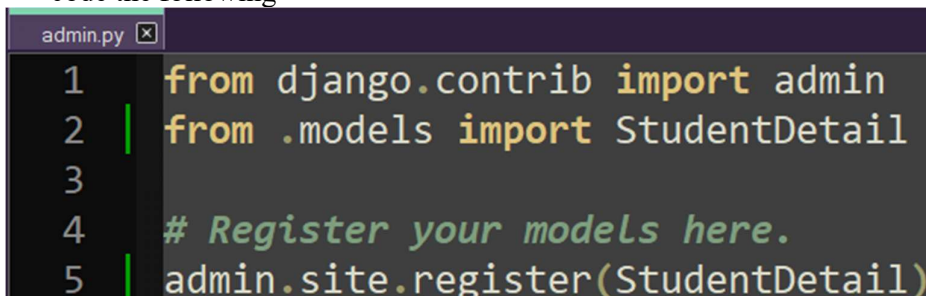
- In the models.py file, you can define model classes, such as a "Student" class, with attributes like student name, parent's name, student number, etc., which should inherit the base models.Model class.
- open **models.py** file in the app folder and code as below

```
models.py  ☒
1     from django.db import models
2     from django.urls import reverse
3
4     # Create your models here.
5    ⊟class StudentDetail(models.Model):
6         gender_choices=[('M','Male'),('F','Female')]
7         gender=models.CharField(choices=gender_choices,max_length=1,
8                             default=None,null=True)
9         first_name=models.CharField(max_length=250)
10        last_name=models.CharField(max_length=250)
11        student_id=models.SlugField(max_length=250,unique=True)
12        fathers_name=models.CharField(max_length=250)
13        mothers_name=models.CharField(max_length=250)
14        dob=models.DateTimeField()
15        objects=models.Manager() #The default manager.
16        std_choices=[('1','I'),('2','II'),('3','III'),('4','IV'),
17                    ('5','V'),('6','VI'),('7','VII'),
18                    ('8','VIII'),('9','IX'),('10','X')]
19        std=models.CharField(choices=std_choices, max_length=2,
20                        default=None,null=True)
21
22        class Meta:
23            verbose_name_plural="Student Details"
24
25        def get_absolute_url(self):
26            return reverse('student:viewStudentDetails',args=[self.student_id])
```

- Register your model to the Django admin by opening the **admin.py** file in your app folder and code the following

```
admin.py  ☒
1    from django.contrib import admin
2    from .models import StudentDetail
3
4    # Register your models here.
5    admin.site.register(StudentDetail)
```
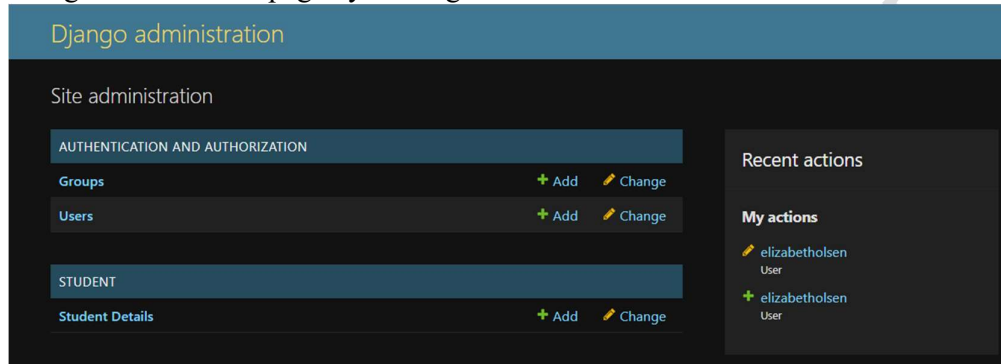
- Execute the following commands to tell the ORM to create a table in the database.

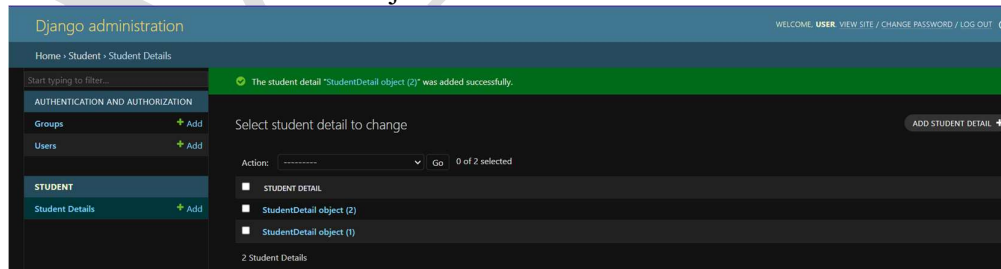## 20. Adding data to your database

Ans: Login to the admin page by starting the server and observe the STUDENT tab.

- Click the Student Details and then click on ADD STUDENT DETAIL to fill the details
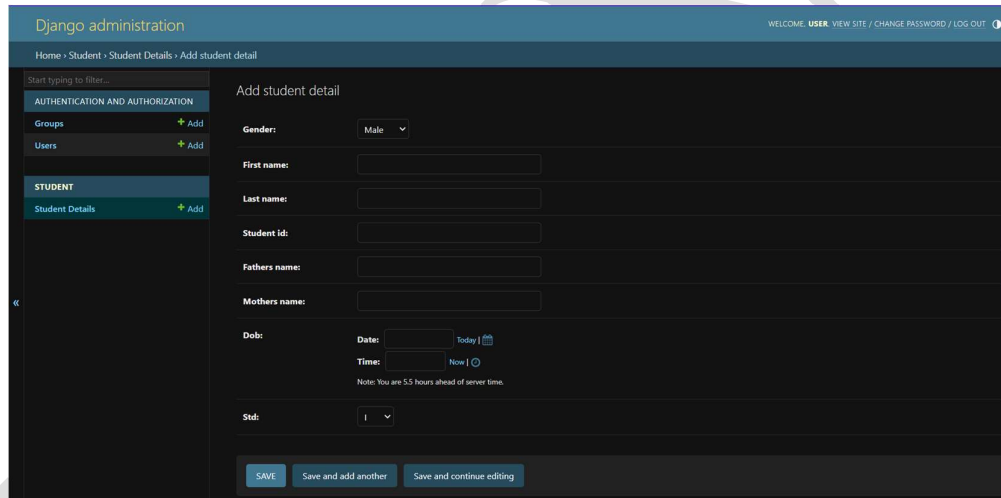
- You can observe a list of objects

21. Editing data in your database

Ans: Click on any of the objects and a form will pop up on the screen in which you can edit any of the fields.

- If you try to edit the Student ID field with a value already used by another student, an error will occur.

- The error occurs because the unique=True attribute is set in the class definition, which prevents duplicate values from being accepted for the field.