MBARARA UNIVERSITY OF SCIENCE AND TECHNOLOGY

FACULTY OF COMPUTING AND INFORMATICS

Course Unit: **CSC2207 Web Application development**

Course Year: **Two**

Name: **ARYATWIJUKA ALLAN**

Reg No: **2021/BCS/084/PS**

SYSTEM REPORT

INTRODUCTION

Netflix is one of the most popular streaming services in the world. It has got a lot of movies, documentaries and TV shows that are readily available for streaming on internet connected devices. With Netflix you can also download movies to your mobile device or tablet to watch offline. Users control what they want to watch and when they want it.

METHODOLOGY

I cloned Netflix web system/ application using the Python version 3.11.4 and Django framework version 4.2.2, SQLite 3, HTML5, CSS3.

The Django framework was used to create and manage the application's operations and navigation throughout the site, SQLite as a database server (store and retrieve records). Html was used for creating web pages, CSS for styling and designing web pages.

This web system is made up of a different categories of web pages which include home page, login page, signup page, profilecreate page, profilelist page, movielist page and playmovie page. The home page contains the signin button that links the user to the signin page. Below is how the home page looks like.

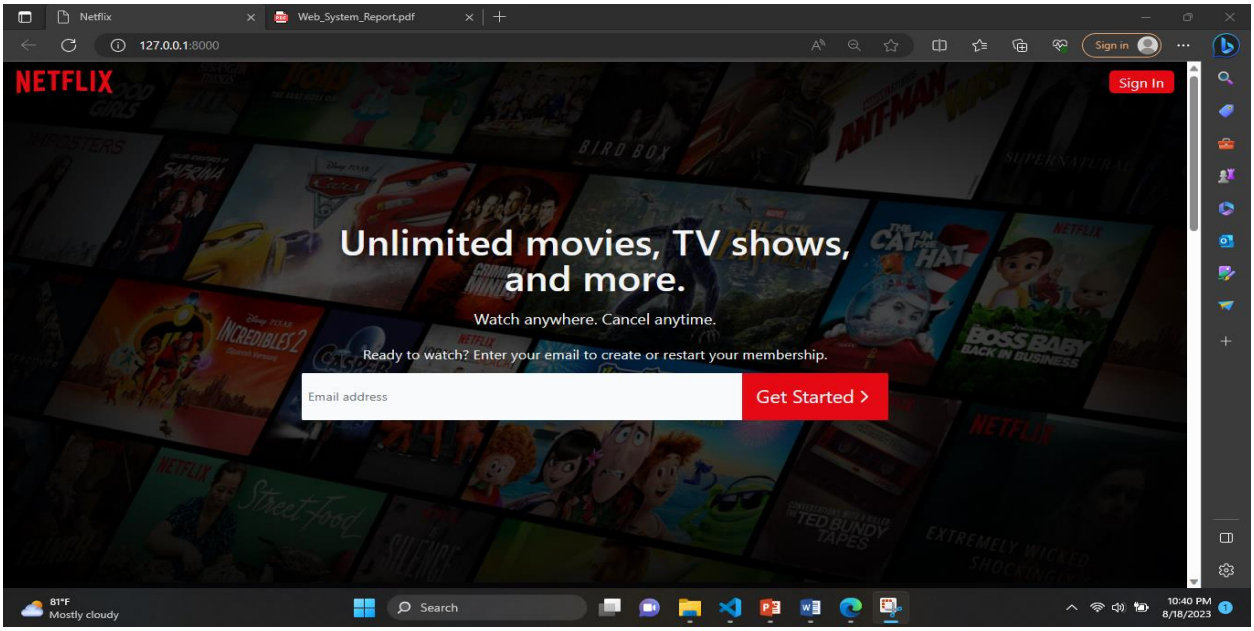

Figure 1: **Home page**

The button on the home page links the users of the web site to the log in /signin page below.

I installed a third party authentication package called Django all-auth. This was to help on account management e.g. change of email address, password and other account settings, user authentication, Registration, Email verification. This package was applied on the login, losignup pages

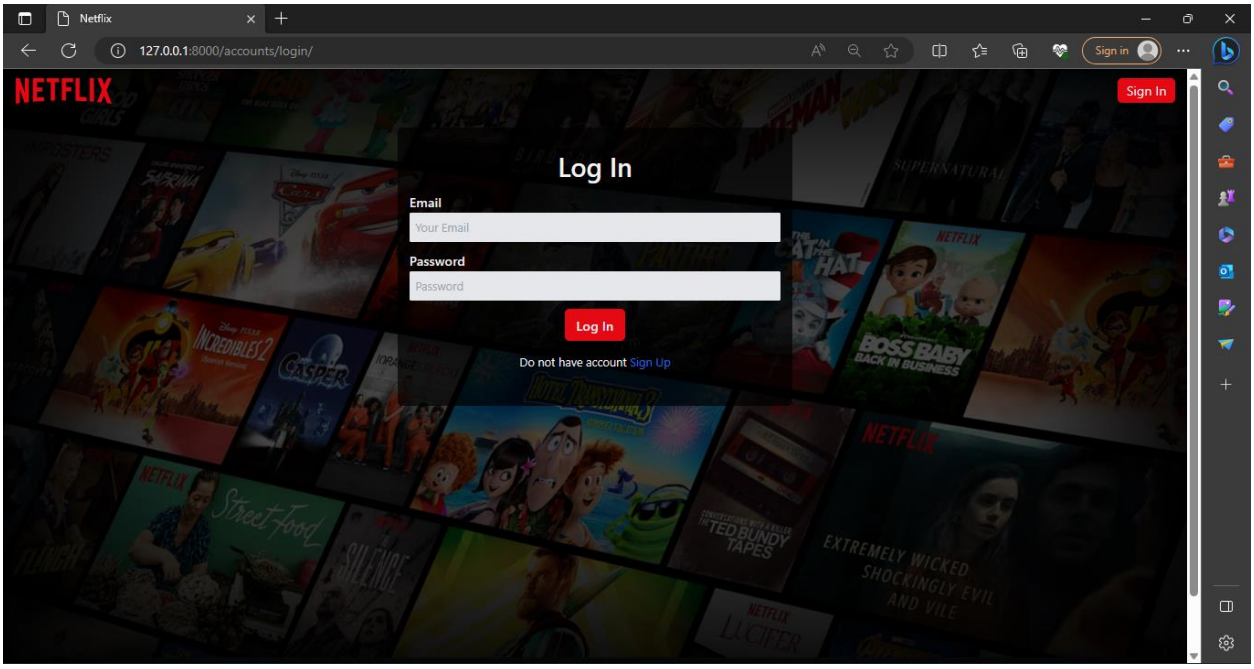

Figure 2: **Login page**

On the login page above, users will have to input their emails and their corresponding passwords to logged in or authenticated. If the user input credentials don't match, the Django all-auth I installed will do the validation on the input and display a corresponding error message. E.g. the error message "The email address and /or password you specified are not correct" was generated after entering wrong credentials

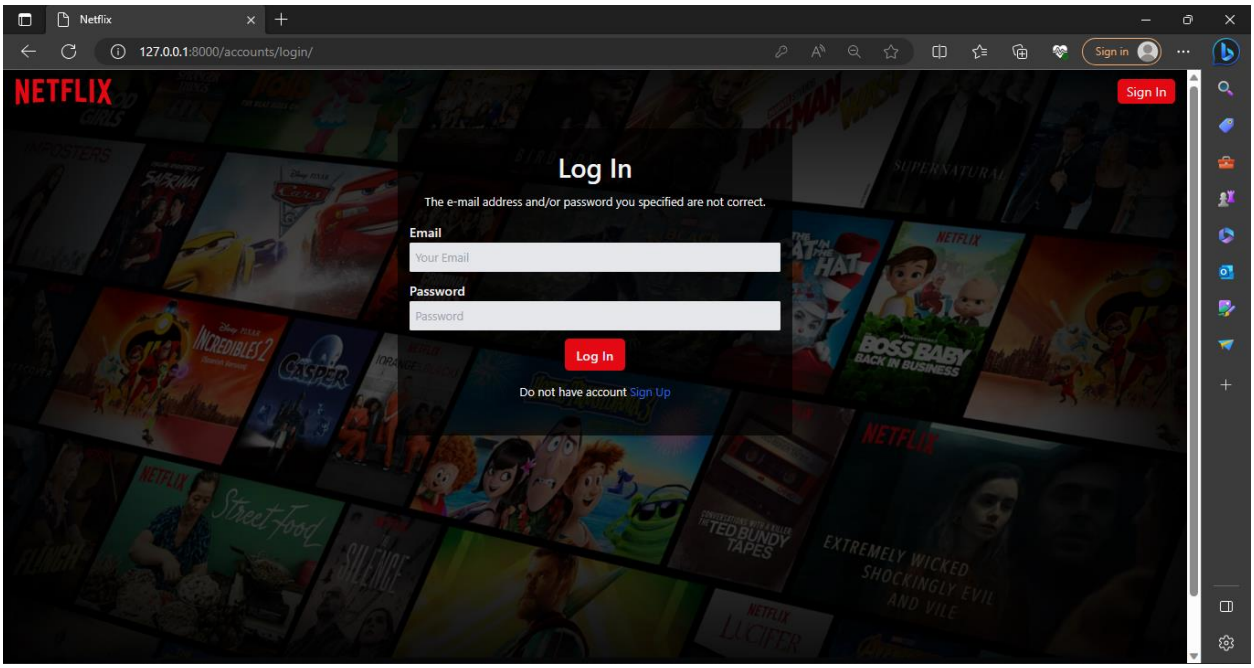

Figure 3: **Login Error**

Once the input credentials (password and e-mail) are correct or match the ones saved in the database. The user will be redirected to the profiles page where they will be able to create different profiles for different users.

If at all the user is new or has never signed up for an account, on the login page there is a signup link that enables the user to access the signup page below. The user will be required to input his email, password and confirm password. Django all-auth will do the validation on the user input and generate corresponding errors e.g. weak password won't be allowed, also passwords similar to email etc. if the user input meet the standard, the will be redirected to the profiles page where they will be able to create different profiles for different users. Below is the signup page.

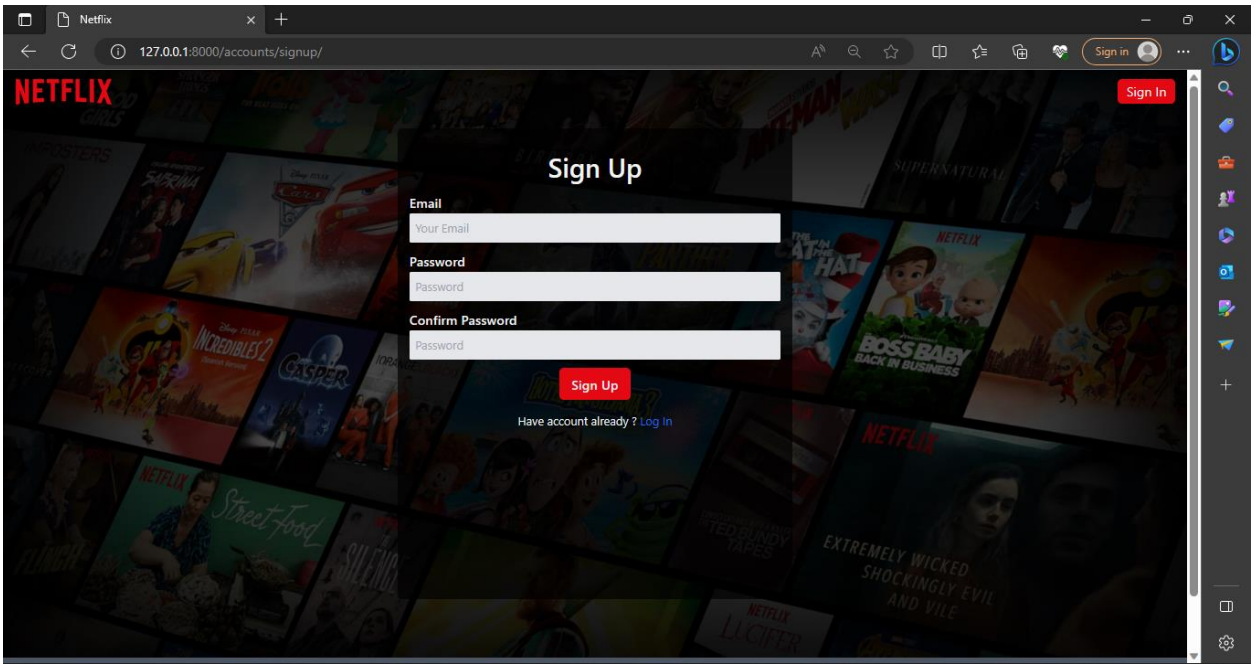

Figure 4: **Signup Page**

Once the email entered has already been used by another person or the password used is too weak, an error message is displayed. This helps the user to change to a new and unique email and also a strong password.

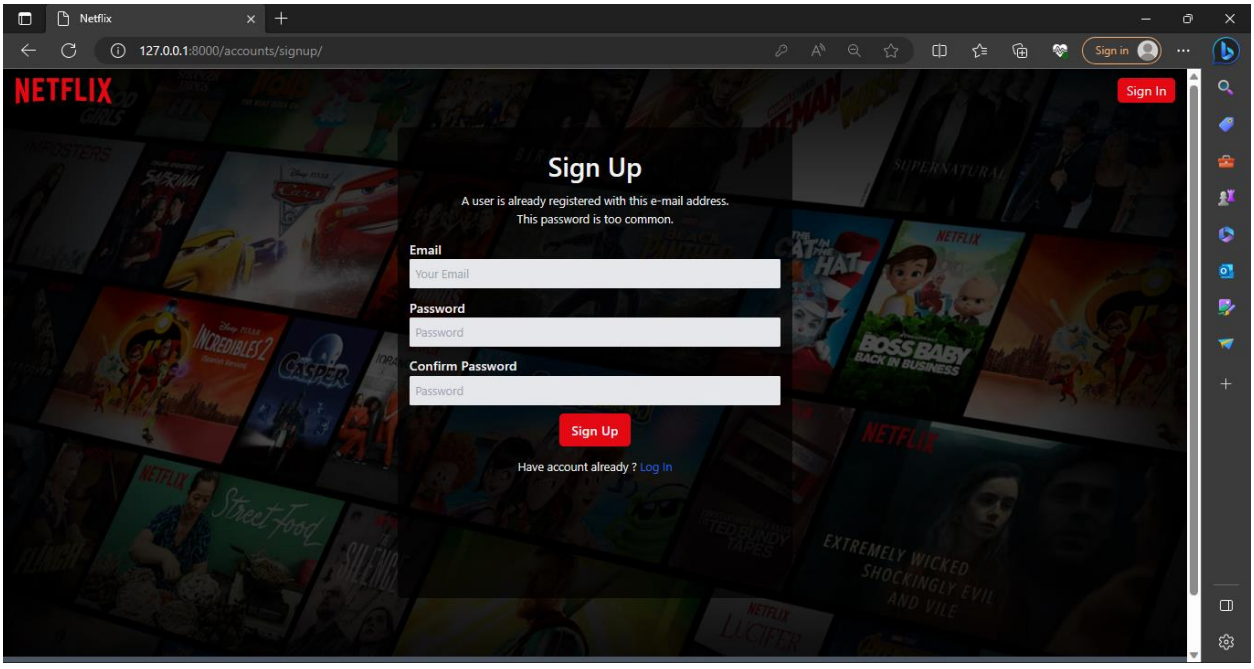

Figure 5: **Sign Up Error**

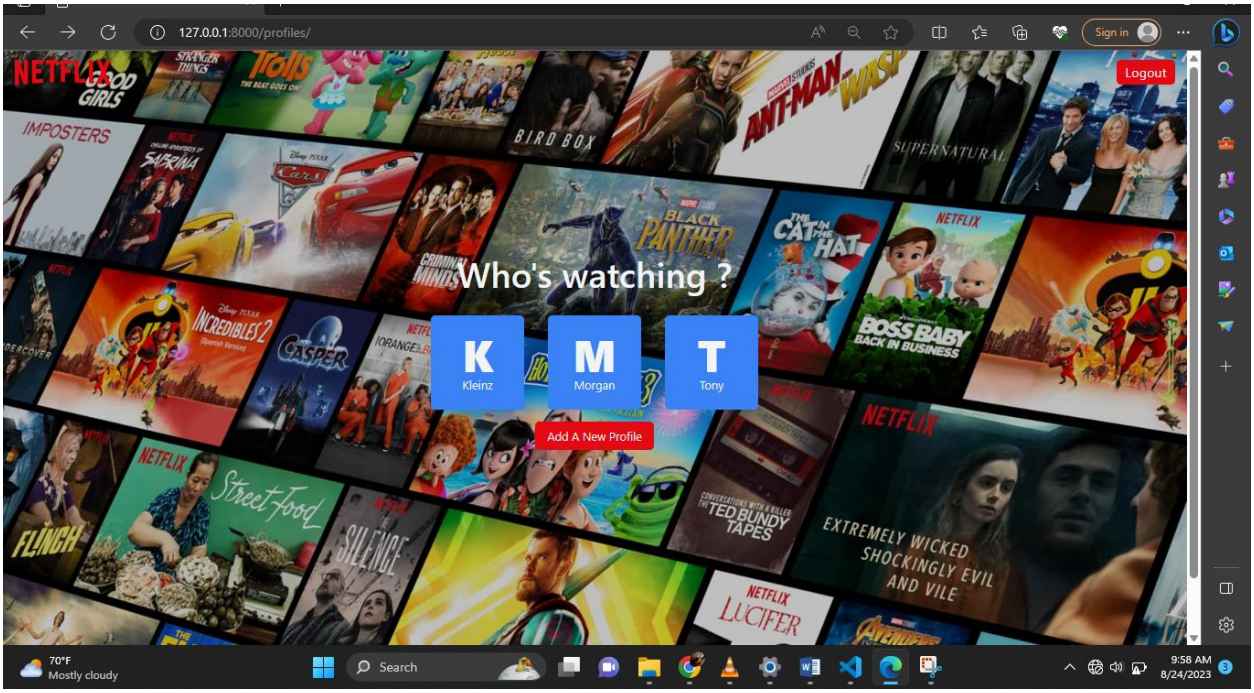

Figure 6: **Profilelist page**

From the above Profilelist page, a user can create a new profile for another person by clicking on the "Add New Profile" red button that redirects them to the profilecreate page below. From the page below, a new profile can be created by entering a profile name and choose whether a person for whom the profile is created is old or a kid. This helps to filter out kids' movies from old people's movies.

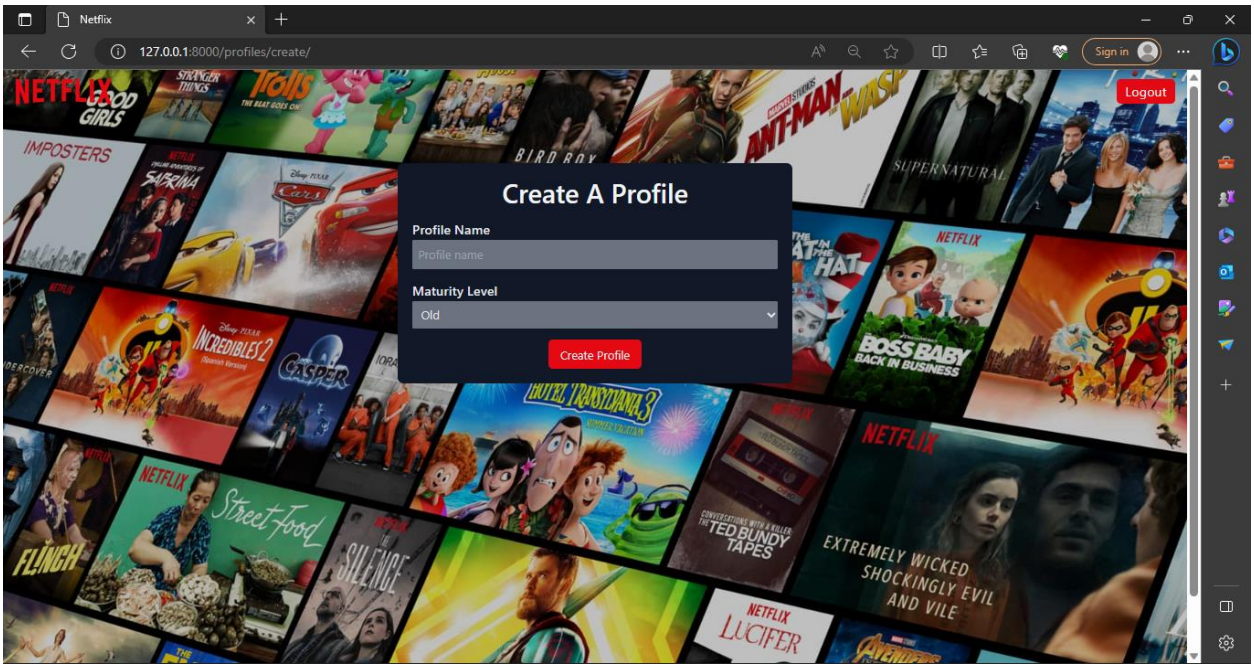

Figure 7: **Profilecreate page**

From the above Profilecreate page, a user is required to input the profile name they want to use, and select between two options old and kid for maturity level. When the user selects old as their maturity level, their profiles will only stream to movies for old people same as a user who selects kid as their maturity level will stream only kids' movies(cartoons). Once the profile is successfully created, the user is redirected back to the profilelist page with their profile already created.

Once the profile has been created, the user is required to click on their profile that redirects them to the movielist page where they can choose any movie they want to watch. Below is the movielist page.

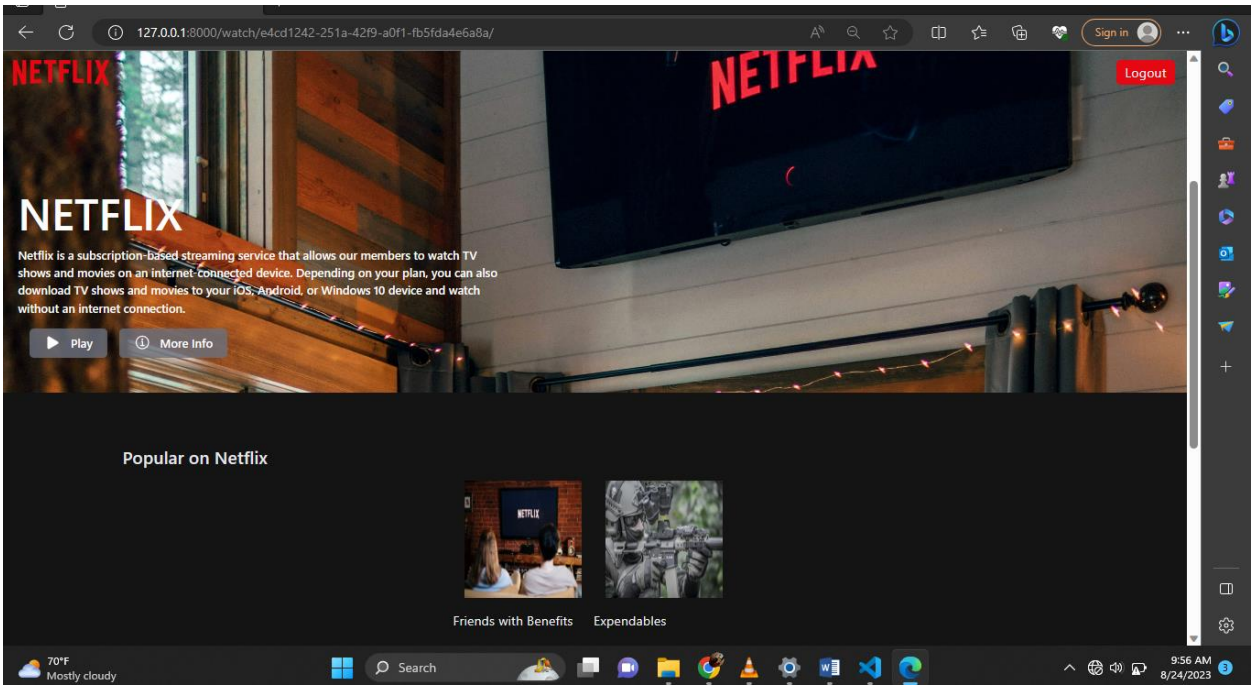

Figure 8: **Movielist page**

Django models have been used to create various tables in the database which are used to store information from the forms in the web pages. Below is an example of a model that stores profile, movie and custom user details in a database.

```python
from django.contrib.auth.models import AbstractUser
import uuid


AGE_CHOICES =(
    ('All', 'All'),
    ('Kids','Kids')
)

MOVIE_CHOICES =(
    ('seasonal', 'Seasonal'),
    ('single','Single')
)
class CustomUser(AbstractUser):
    profiles = models.ManyToManyField('Profile', blank =True)

class Profile(models.Model):
    name = models.CharField(max_length=1000)
    age_limit = models.CharField(choices=AGE_CHOICES,max_length=10)
    uuid = models.UUIDField(default=uuid.uuid4)

    def __str__(self):
        return self.name

class Movie(models.Model):
    title =models.CharField(max_length=1000)
    description = models.TextField(blank =True, null= True)
    created = models.DateTimeField(auto_now_add = True)
    uuid = models.UUIDField(default=uuid.uuid4)
    type = models.CharField(choices=MOVIE_CHOICES, max_length=10)
    video = models.ManyToManyField('video')
    image = models.ImageField(upload_to='covers')
    age_limit = models.CharField(choices=AGE_CHOICES, max_length=10)

    def __str__(self):
        return self.title

```

Figure 9: **Models**

I used the Django views to render and redirect to different web pages on request by the user. More so, they have been used with the help of “POST” method to transfer data from a webpage (forms) to the models for storage. Below is a sample of a view used to post data from the forms in the web pages.

Figure 10: **Views**

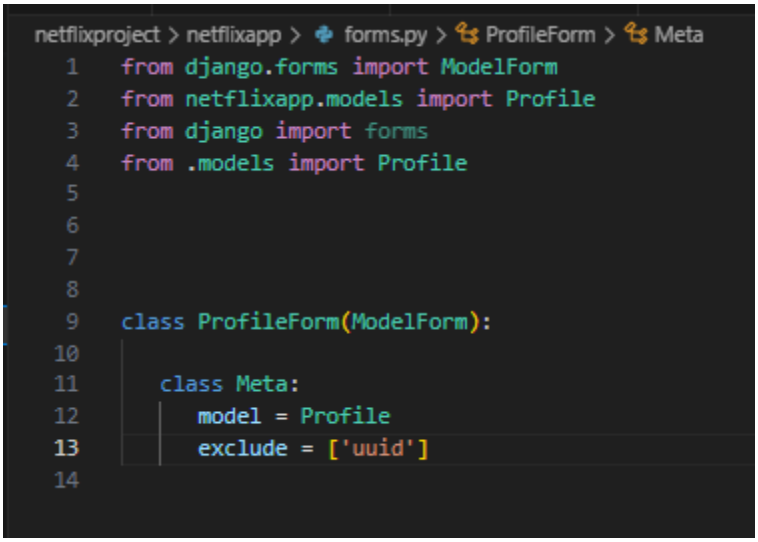

Figure 11: **Form**

PROBLEMS

- Designing the website to be responsive across different devices and screen sizes.

- Managing a large database of movies and TV shows and optimizing queries for seamless user interactions.
- Ensuring that the cloned website maintains visual consistency with the original Netflix interface can be challenging.

CONCLUSION

Cloning the Netflix website using Django presents a comprehensive learning experience in web development, backend integration, and frontend design. The project encompasses several critical aspects of building a sophisticated web application, including user authentication, content management, streaming capabilities, and responsive design.