



MBARARA UNIVERSITY OF SCIENCE AND TECHNOLOGY
FACULTY OF COMPUTING AND INFORMATICS

Course Unit: Web Development.

Academic Year: 2022/2023

Semester: Two

PROJECT

Student Name: NAMAZZI SOPHIE

Reg No.: 2021/BCS/103/PS

Title: Report on Cloning a Netflix-Inspired Website using Django

Abstract:

This report outlines the development process and technical details of a Django project aimed at creating a website inspired by the design and functionality of Netflix. The project involved implementing key features of a streaming platform, including user registration, movie and TV show listings, user authentication, and responsive UI design. The report covers the project's objectives, technical architecture, development process, challenges faced, and the outcomes achieved.

Table of Contents:

1. Introduction
2. Project Overview
3. Technical Details
4. Development Process
5. User Interface and Experience
6. Features and Functionality
7. Testing
8. Conclusion
9. Future Enhancements
10. References

1. Introduction:

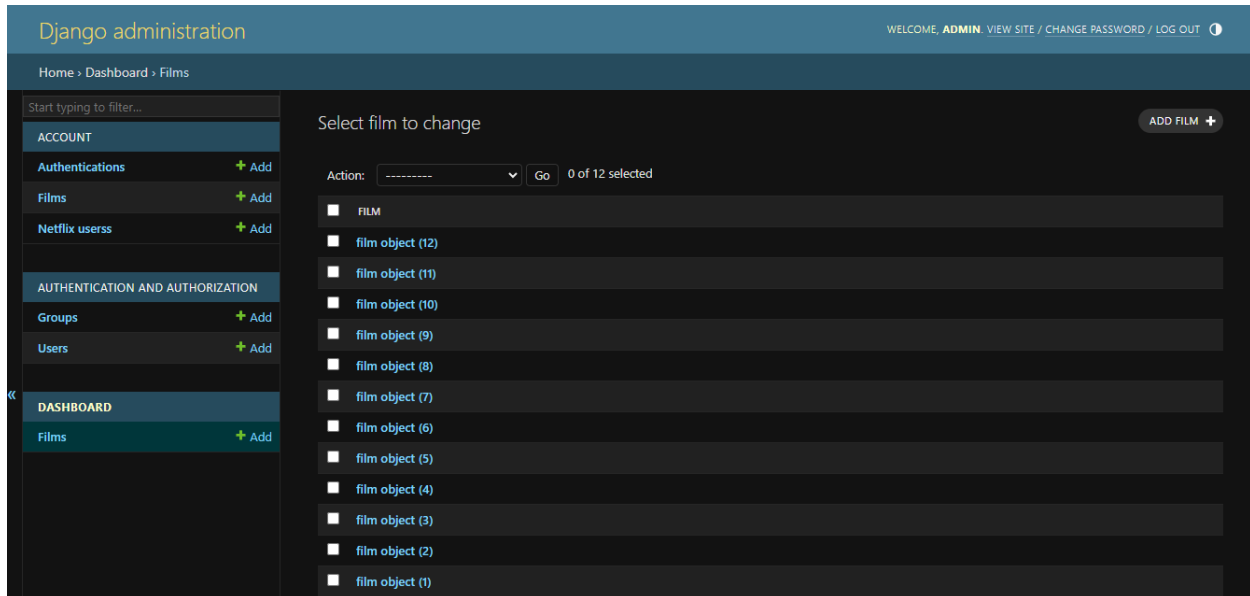
The aim of this project was to replicate the look and feel of a popular streaming platform, Netflix, using Django as the primary framework. By mimicking the UI and some core functionalities, the project aimed to provide users with a familiar experience while showcasing the capabilities of Django in web application development.

2. Project Overview:

The project focused on creating a web application with features such as user registration, user authentication, movie and TV show listings, search functionality, and a responsive user interface. The website would allow users to explore a catalog of content, view details about movies and TV shows, and manage their profiles.

3. Technical Details:

- Framework: Django (Python)



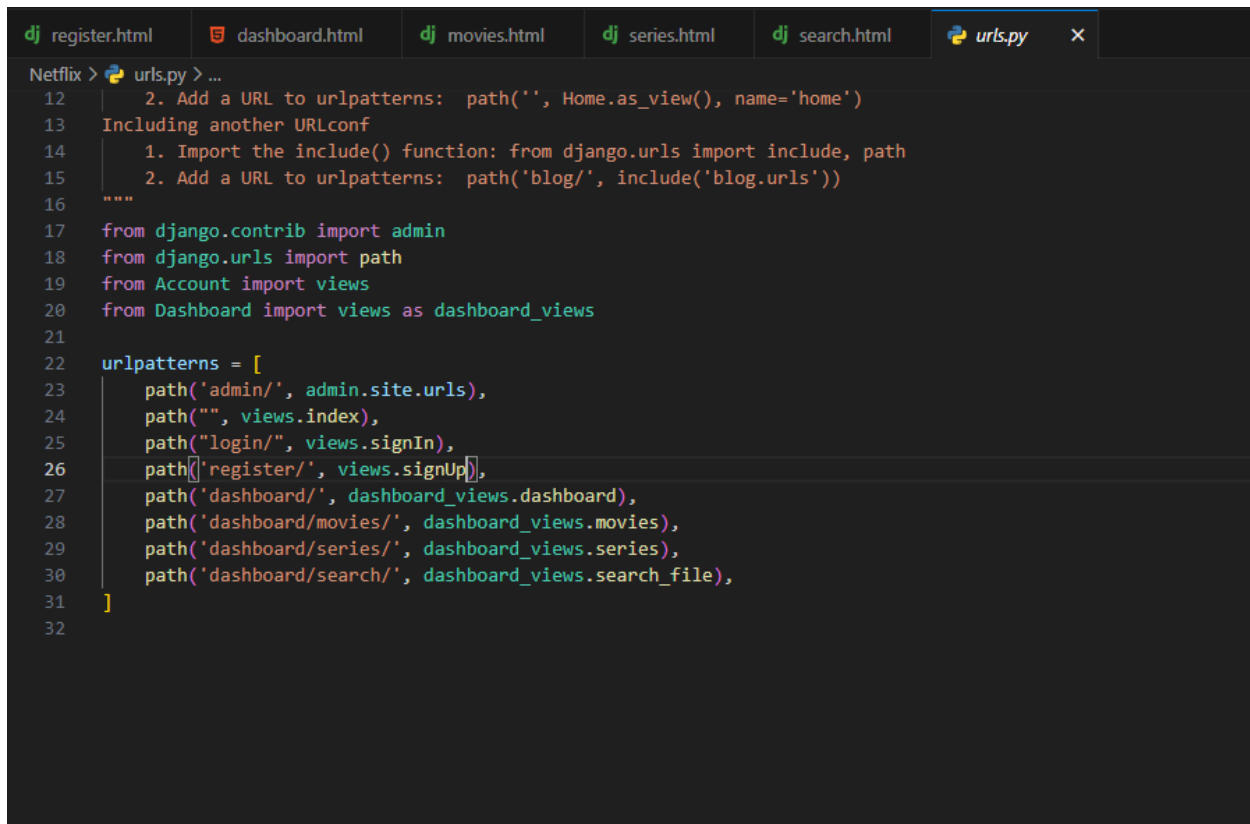
- Database: PostgreSQL

- Frontend: HTML, CSS, JavaScript like the home page



- Third-Party Libraries: Bootstrap for UI design, Pillow for image processing, Django Rest Framework for API development

-URLS;

A screenshot of a code editor with a dark theme. At the top, there are several tabs: 'dj register.html', 'dashboard.html', 'dj movies.html', 'dj series.html', 'dj search.html', and 'urls.py' (which is the active tab). The main area shows Python code for Django URL patterns. Line 12 has a comment '2. Add a URL to urlpatterns: path('', Home.as_view(), name='home')'. Line 13 has a comment 'Including another URLconf'. Line 14 has a comment '1. Import the include() function: from django.urls import include, path'. Line 15 has a comment '2. Add a URL to urlpatterns: path('blog/', include('blog.urls'))'. Line 16 is an empty line. Lines 17-20 are imports: 'from django.contrib import admin', 'from django.urls import path', 'from Account import views', and 'from Dashboard import views as dashboard_views'. Line 21 is an empty line. Line 22 starts the 'urlpatterns' list. Lines 23-30 are path entries: 'path('admin/', admin.site.urls)', 'path("", views.index)', 'path("login/", views.signIn)', 'path('register/', views.signUp)', 'path('dashboard/', dashboard_views.dashboard)', 'path('dashboard/movies/', dashboard_views.movies)', 'path('dashboard/series/', dashboard_views.series)', and 'path('dashboard/search/', dashboard_views.search_file)'. Line 31 closes the list with a bracket. Line 32 is an empty line.

```
Netflix > urls.py > ...
12 | 2. Add a URL to urlpatterns: path('', Home.as_view(), name='home')
13 | Including another URLconf
14 | 1. Import the include() function: from django.urls import include, path
15 | 2. Add a URL to urlpatterns: path('blog/', include('blog.urls'))
16 | """
17 | from django.contrib import admin
18 | from django.urls import path
19 | from Account import views
20 | from Dashboard import views as dashboard_views
21 |
22 | urlpatterns = [
23 |     path('admin/', admin.site.urls),
24 |     path("", views.index),
25 |     path("login/", views.signIn),
26 |     path('register/', views.signUp),
27 |     path('dashboard/', dashboard_views.dashboard),
28 |     path('dashboard/movies/', dashboard_views.movies),
29 |     path('dashboard/series/', dashboard_views.series),
30 |     path('dashboard/search/', dashboard_views.search_file),
31 | ]
32 |
```

4. Development Process:

- Requirement Analysis: Defined the scope and features of the project based on the Netflix website.
- Database Design: Designed the database schema for user accounts, content details, and user interactions.
- Backend Development: Implemented user authentication, content models, and API endpoints for data retrieval.
- Frontend Development: Created responsive UI templates for browsing content, viewing details, and managing user profiles.
- Integration: Integrated frontend and backend components to ensure seamless user interactions.
- Testing and Debugging: Conducted unit tests and resolved any bugs or issues.

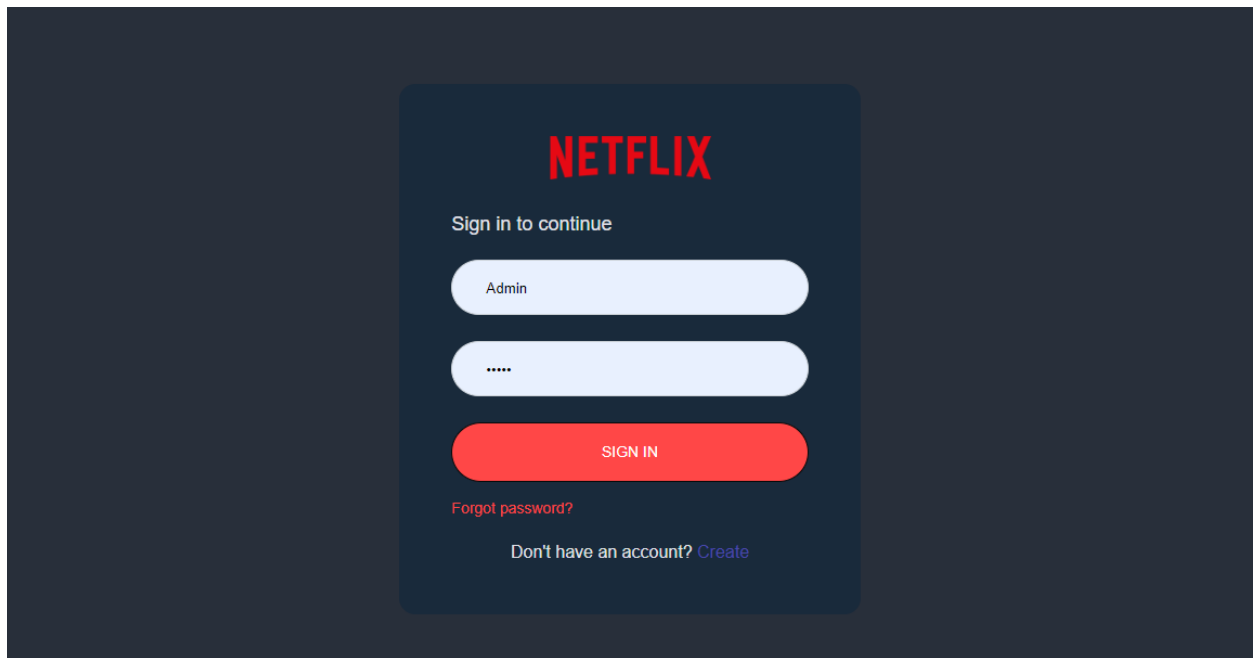
- Version Control: Utilized Git for version control and collaboration.

5. User Interface and Experience:

The UI was designed to resemble the Netflix interface, featuring a top navigation bar, content categories, responsive grids, and a visually appealing user interface. User experience was prioritized with smooth navigation, intuitive buttons, and consistent styling.

6. Features and Functionality:

- User Registration and Authentication



NETFLIX

New here?
Signing up is easy. It only takes a few steps

First Name

Last Name

mm/dd/yyyy

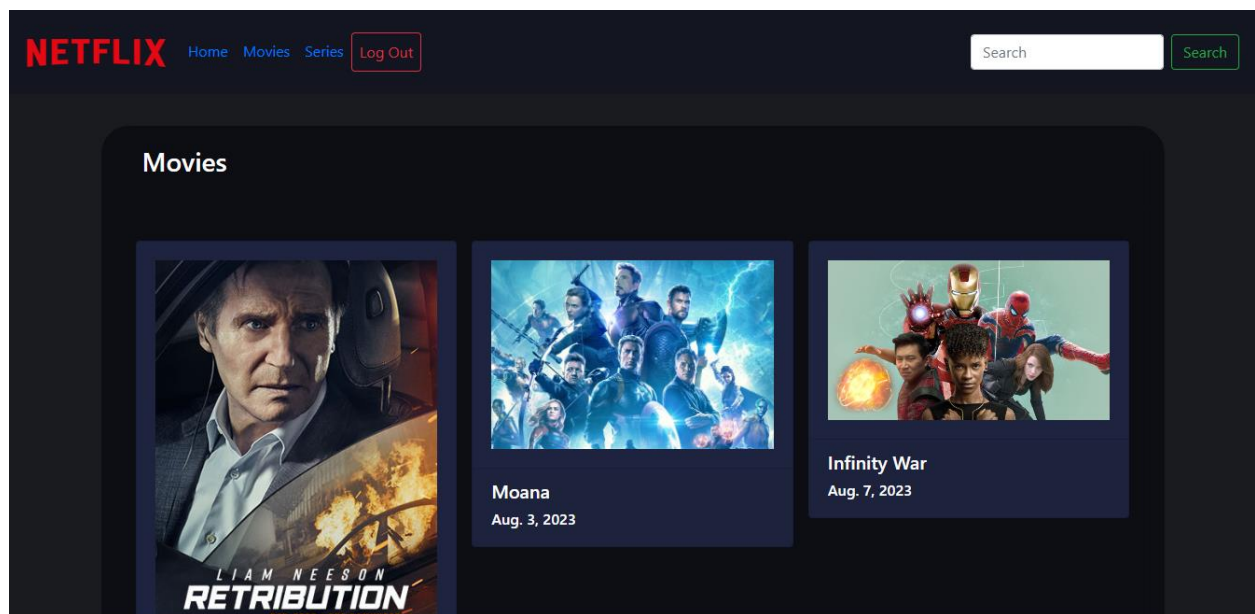
Admin

.....

Re-Enter Password

☐ By registering, you agree to the terms and conditions of usage of Netflix services

- Browse and Search Content
- View Movie/TV Show Details



- User Profile Management

- Play Movie Trailers (embedded videos)
- Responsive Design for Different Devices

7. Testing:

- Conducted unit tests for backend logic and API endpoints.
- Performed user testing for UI/UX and gathered feedback for improvements.
- Ensured cross-browser compatibility and responsiveness.

8. Conclusion:

The Django project successfully achieved its goal of creating a Netflix-inspired streaming platform. The implementation of key features, responsive design, and a user-friendly interface demonstrated the capabilities of Django in building robust web applications.

9. Future Enhancements:

Future enhancements could include:

- Implementing a recommendation system.
- Enabling user-generated reviews and ratings.
- Expanding content categories and adding sorting/filtering options.
- Enhancing security features and user privacy.

10. Challenges

- Implementing secure user authentication and managing user roles and permissions effectively
- Designing and implementing complex database relationships.

- Identifying and fixing bugs, ensuring code quality and performing thorough testing.

11. References:

<https://www.codewithrandom.com/2023/02/11/netflix-landing-page-clone-netflix-clone-source-code/>

<https://www.fr.freelancer.com/projects/twitter-bootstrap/html-bootstrap-netflix-clone-website>

<https://reactjsexample.com/netflix-clone-made-with-react-html-css/>