# CHAPTER ONE INTRODUCTION

## **Background of Study**

Streaming TV advertising is advertising that runs within TV content watched through an internet-connected device rather than ads you may see on traditional linear television. These streaming service ads will appear either before or during streaming TV content found on ad-supported networks and apps (Greenfield, 2022).

To aid with the challenges of video marketing, various sources offer different kinds of solutions and reliefs. According to Lessard (2017), video will transform from an outsourced production to an in-house production. This means that businesses will start to establish their own video production teams within the company, who make the marketing videos rather than outsourcing the production to a dedicated video production company. Lessard (2017) writes, that according to Halligan buyers want short, easily consumed and informative content due to short attention spans and high cynicism. Therefore, videos should forward a feeling that they are created by real people. In-house video productions allow more control over the process, from the presentation of interactive features and analytics to automated marketing tools and customer relationship management. Making videos in-house makes it easier to distinguish a brand from an increasingly crowded market.

Murdico (2017), predicts that video production will become cheaper already during 2017. Murdico declares that when the content is creative, it has a targeted audience and a call to action, the production doesn’t necessarily need expensive cameras and a large crew. According to Murdico, marketers should start to think about continuity in video production, like they do in TV series and magazines. When customers’ needs are matched in the video

production, it will more likely be efficient.

As far as content strategy goes, Murdico (2017) writes that creativity becomes more and more important for video production and marketing in general. These days anyone can shoot a high-quality video so the content and its creativity are what matter the most. There are three things involved in creative video productions; addressing targeted consumers’ needs, doing it in a clever way to catch their attention and by that to give them a call to action. The most effective ways to gain viewership are to use humour, emotion, value and immediacy. Succeeding in video production generates soft video marketing goals, which means for example increases in social media interaction, though the ultimate factor for video marketing success are sales. There should be a rise in sales with quality video content, and at the end of the day, that is what matters the most. Furthermore, to get the most out of video marketing, marketers should remember to add a call for action alongside the video. Regardless of the video content, it turns out the placement of online video advertising has a considerable impact on viewership. If an online video is paid to be shown as an advertisement among other content - where the advert is shown greatly affects the videos completion rate.

The focus of this project is create a platform that advertise product as video on demand (VoD) as guide to consumers and customers on how to operate the products like Netflix.

## **Statement of the Problem**

Understanding a customer base is why companies advertise their product but to gauge customer engagement with a product will require the customer understanding of the product and that’s portrayed through advertisements. Clients and customers engagement to a product is influenced by advertisement and with 2-Dimentional (2D) advertising customer are left to their imaginations to gauge the applicable areas of products and their usage which limit engagement but with 3-Dimentional (3D) or video streaming, products can be advertised through professional usage in their areas of application and method of usage.

## **Aim and Objectives of the Study**

By developing a video streaming application companies can engage user with creative content on how to use product in multiple ways including how to maintain the product with care. And also, how professionals handle said product. Another advantage is customers can see best ways to use the products thereby reducing time exploring best ways of using the product.

The objectives are:

1. An engaging and easy to use UI with good UX using Html and Tailwind.
2. The logic will be handled and the data will be stored using Django.
3. Evaluating and verifying the app.

## **Scope of the Project**

The area covered in the project focus on video streaming of advertisement product by a company using an online platform powered by Django. The project will not facilitate live streaming.

## **Limitation of the Study**

The platform is limited to Video on Demand (VoD) approach meaning no live streaming, will have minimum functionality with no payment gateway due to resource and time constraint.

## **Significance of the Study**

The platform or web app stands to benefit the industry with a choice of diverse advertising for different products and a way for customers to see how products are used and their relevance thereby allowing customers to make better choice in choosing of product and brand.

## **Project Organization**

The project is divided into five chapters. The outlines are presented below:

**Chapter One: Introduction**

Chapter one introduces this project work, the study's background, the problem statement, the purpose and objectives, the scope of the study, the constraints of the study, the relevance of the study, the project organization, and the definition of terms.

**Chapter Two: Literature review**

This chapter focuses on the literature review, and the contributions of other scholars on the subject matter being discussed.

**Chapter Three: Methodology and Design**

This chapter is concerned with the presentation of the results of system analysis and design. It presents the research methodology used in the development of the system to facilitate an understanding and effective future implementation of the system.

**Chapter Four: System Implementation Evaluation**

This chapter describes the system implementation and documentation, analysis of modules, and system requirements for implementation.

**Chapter Five: Summary, Conclusion, and** **Recommendation**

The chapter provides a summary of major findings, conclusions, and recommendations based on the study conducted

# CHAPTER TWO LITERATURE REVIEW



## **Introduction**

A look at what the programming language and database used the project are and a review of related literature.

## **Programming Languages Used**

### **Tailwind CSS**

Tailwind CSS can be used to make websites in the fastest and the easiest way.

Tailwind CSS is basically a utility-first CSS framework for rapidly building custom user interfaces (sapna & sachin, 2021). It is a highly customizable, low-level CSS framework that gives you all of the building blocks you need to build bespoke designs without any annoying opinionated styles you have to fight to override (sapna & sachin, 2021).

The beauty of this thing called tailwind is it doesn’t impose design specification or how your site should look like, you simply bring tiny components together to construct a user interface that is unique (sapna & sachin, 2021). What Tailwind simply does is take a ‘raw’ CSS file, processes this CSS file over a configuration file, and produces an output (sapna & sachin, 2021).

#### **Why Tailwind CSS?**

Faster UI building process, It is a utility-first CSS framework which means we can use utility classes to build custom designs without writing CSS as in traditional approach (sapna & sachin, 2021).

Advantages of Tailwind CSS:

1. No more silly names for CSS classes and Id’s.
2. Minimum lines of Code in CSS file.
3. We can customize the designs to make the components.
4. Makes the website responsive.
5. Makes the changes in the desired manner. CSS is global in nature and if make changes in the file the property is changed in all the HTML files linked to it. But with the help of Tailwind CSS we can use utility classes and make local changes (sapna & sachin, 2021).

### **Django**

Django is a high-level Python web framework that enables rapid development of secure and maintainable websites. Built by experienced developers, Django takes care of much of the hassle of web development, so you can focus on writing your app without needing to reinvent the wheel. It is free and open source, has a thriving and active community, great documentation, and many options for free and paid-for support (hamishwillee & ozgurturkiye, 2022).

Django helps you write software that is:

**Complete**

Django follows the "Batteries included" philosophy and provides almost everything developers might want to do "out of the box". Because everything you need is part of the one "product", it all works seamlessly together, follows consistent design principles, and has extensive and up-to-date documentation (hamishwillee & ozgurturkiye, 2022).

**Versatile**

Django can be (and has been) used to build almost any type of website — from content management systems and wikis, through to social networks and news sites. It can work with any client-side framework, and can deliver content in almost any format (including HTML, RSS feeds, JSON, and XML) (hamishwillee & ozgurturkiye, 2022).

Internally, while it provides choices for almost any functionality you might want (e.g. several popular databases, templating engines, etc.), it can also be extended to use other components if needed (hamishwillee & ozgurturkiye, 2022).

**Secure**

Django helps developers avoid many common security mistakes by providing a framework that has been engineered to "do the right things" to protect the website automatically. For example, Django provides a secure way to manage user accounts and passwords, avoiding common mistakes like putting session information in cookies where it is vulnerable (instead cookies just contain a key, and the actual data is stored in the database) or directly storing passwords rather than a password hash (hamishwillee & ozgurturkiye, 2022).

A password hash is a fixed-length value created by sending the password through a [cryptographic hash function](https://en.wikipedia.org/wiki/Cryptographic_hash_function). Django can check if an entered password is correct by running it through the hash function and comparing the output to the stored hash value. However due to the "one-way" nature of the function, even if a stored hash value is compromised it is hard for an attacker to work out the original password (hamishwillee & ozgurturkiye, 2022).

Django enables protection against many vulnerabilities by default, including SQL injection, cross-site scripting, cross-site request forgery and [clickjacking](https://developer.mozilla.org/en-US/docs/Glossary/Clickjacking) (hamishwillee & ozgurturkiye, 2022).

**Scalable**

Django uses a component-based "shared-nothing" architecture (each part of the architecture is independent of the others, and can hence be replaced or changed if needed). Having a clear separation between the different parts means that it can scale for increased traffic by adding hardware at any level: caching servers, database servers, or application servers. Some of the busiest sites have successfully scaled Django to meet their demands (e.g. Instagram and Disqus, to name just two) (hamishwillee & ozgurturkiye, 2022).

**Maintainable**

Django code is written using design principles and patterns that encourage the creation of maintainable and reusable code. In particular, it makes use of the Don't Repeat Yourself (DRY) principle so there is no unnecessary duplication, reducing the amount of code. Django also promotes the grouping of related functionality into reusable "applications" and, at a lower level, groups related code into modules (along the lines of the Model View Controller (MVC) pattern) (hamishwillee & ozgurturkiye, 2022).

**Portable**

Django is written in Python, which runs on many platforms. That means that you are not tied to any particular server platform, and can run your applications on many flavors of Linux, Windows, and macOS. Furthermore, Django is well-supported by many web hosting providers, who often provide specific infrastructure and documentation for hosting Django sites (hamishwillee & ozgurturkiye, 2022).

In a traditional data-driven website, a web application waits for HTTP requests from the web browser (or other client). When a request is received the application works out what is needed based on the URL and possibly information in POST data or GET data. Depending on what is required it may then read or write information from a database or perform other tasks required to satisfy the request. The application will then return a response to the web browser, often dynamically creating an HTML page for the browser to display by inserting the retrieved data into placeholders in an HTML template (hamishwillee & ozgurturkiye, 2022).

Django web applications typically group the code that handles each of these steps into separate files:

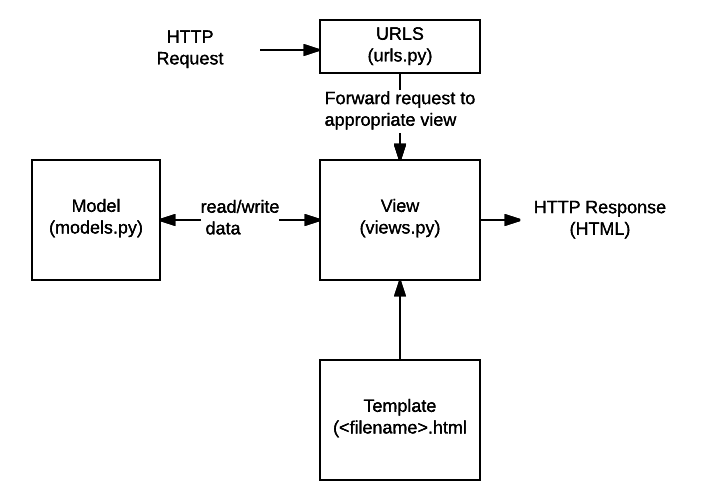


Figure 2.1: Structure of Django File (hamishwillee & ozgurturkiye, 2022)

1. **URLs:** While it is possible to process requests from every single URL via a single function, it is much more maintainable to write a separate view function to handle each resource. A URL mapper is used to redirect HTTP requests to the appropriate view based on the request URL. The URL mapper can also match particular patterns of strings or digits that appear in a URL and pass these to a view function as data (hamishwillee & ozgurturkiye, 2022).
2. **View:** A view is a request handler function, which receives HTTP requests and returns HTTP responses. Views access the data needed to satisfy requests via *models*, and delegate the formatting of the response to *templates* (hamishwillee & ozgurturkiye, 2022).
3. **Models:** Models are Python objects that define the structure of an application's data, and provide mechanisms to manage (add, modify, delete) and query records in the database (hamishwillee & ozgurturkiye, 2022).
4. **Templates:** A template is a text file defining the structure or layout of a file (such as an HTML page), with placeholders used to represent actual content. A *view* can dynamically create an HTML page using an HTML template, populating it with data from a *model*. A template can be used to define the structure of any type of file; it doesn't have to be HTML! (hamishwillee & ozgurturkiye, 2022).

## **Related Literature**

Goel, (2013) investigates cloud centered video streaming methods particularly from the mobile viewpoint. The qualitative part of the research contains explanations of current video development methods, streaming methods and third celebration cloud centered streaming solutions for different mobile which shows my realistic work relevant to streaming methods with RTMP protocols family and solutions for iPhone, Android, Smart mobile phones, Window and BalackBerry phones.

Adimoolam et al., (2013) paper makes use of an Android based mobile phone as the embedded platform, connecting to an IP network using its built-in Wi-Fi interface. The phone’s camera performs the video capturing function, whose output is encoded into MPEG4 format. And it is streamed over the Wi-Fi network using the LIVE555 media server. Then the video can be observed at remote stations by adopting the inbuilt media player. And it also gives a study of Android framework and how it functions.

Suganthi et al., (2013) propose a new mobile video streaming framework consist of AMES cloud, which is dubbed with: adaptive mobile video streaming and efficient social video sharing these construct a private agent to provide video streaming efficiently for each mobile user. AMoV uses the scalable video coding technique to adjust the streaming. ESoV monitors the social network interactions among mobile users and as to prefetch video in advance. the private agents in the clouds can effectively provide the adaptive streaming, and perform video sharing based on the social network.

Londhe et al., (2016) paper briefly described & analyzed the role of cloud computing in mobile devices. But Mobile multimedia still poses many challenges in efficient video streaming due to the form of mobile devices. In mobile cloud video rendering and encoding is performed on cloud servers, with the resulting video streamed over wireless networks to mobile devices. For providing higher user experience in mobile devices higher bit rate is required to improve video quality and response time in video.

According to Marshall (2016) there were little under 8 million videos uploaded to Facebook in November 2016 and those videos generated 229 billion views. Smith (2016) states that in March 2016, Instagram’s top three most watched video creators generated circa 250 million views in that month.

Kalle & Kalle, (2017) research was to examine the online video marketing habits and attitudes of small and medium-sized enterprises in Jyväskylä for RecOn Productions Oy, a local audio-visual production company. The findings of the research provide RecOn Productions Oy with an understanding of what type of businesses would be interested in their services and what are their video marketing needs, but also serves anyone who is interested in the video marketing trends in the area.

Online video is not yet a ready-made vehicle for marketing and it will keep on evolving as Lessard (2017) suggests in his article on Marketing profs website. Video will transform from passive to interactive experience, where the viewer doesn’t just sit and watch the video passively but can take part in surveys, click links or see the video in 360° material. As the viewer interaction increases, the back end can also track more meaningful metrics and provide more meaningful analytics. Another aspect of video development is the hyper personalization of videos. This means that video will not remain only as one size-fits-all proposition, but a personalized message just for the viewer. A targeted and persona-based video requires more content, but will yield a better return on investment (ROI). As the personalization increases, viewers will become more engaged with content and the story of a company.

Khan et al., (2022) paper presents a comprehensive survey of recent developments in MEC-enabled video streaming bringing unprecedented improvement to enable novel use cases. A detailed review of the state-of-the-art is presented covering novel caching schemes, optimal computation offloading, cooperative caching and offloading and the use of artificial intelligence (i.e., machine learning, deep learning, and reinforcement learning) in MEC-assisted video streaming services.

# CHAPTER THREE

**METHODOLOGY AND DESIGN**



## **Introduction**

This chapter covers the analysis models (data flow diagram, sequence, class and entity relationship diagram) of the project and the higher-level solution (programming language) approach used.

## **Analysis Models**

Analysis models is a technical representation of the system. It acts as a link between the system description and the design model. In Analysis Modelling, information, behavior, and functions of the system are defined and translated into the architecture, component, and interface level design in the design modelling.

### **Use Case Diagram**

Use-case diagrams **describe the high-level functions and scope of a system**. These diagrams also identify the interactions between the system and its actors. The use cases and actors in use-case diagrams describe what the system does and how the actors use it, but not how the system operates internally.

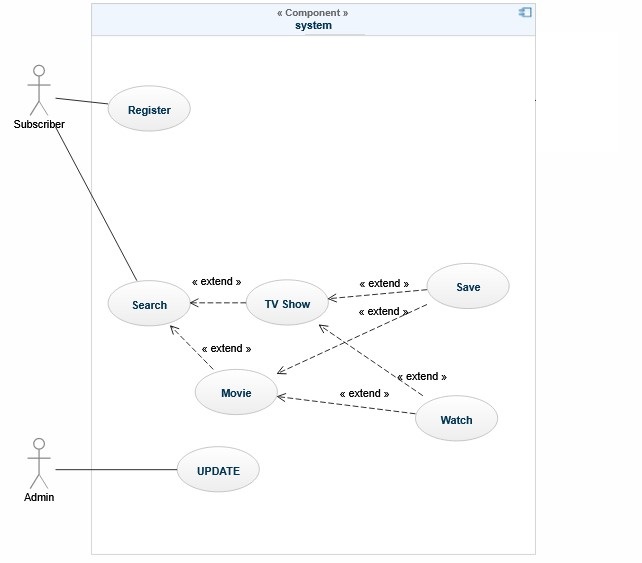


Figure 3.1: Use Case Diagram

### **Data Flow Diagram**

A data flow diagram (DFD) maps out the flow of information for any process or system. It graphically represents the flow of data in a business information system.

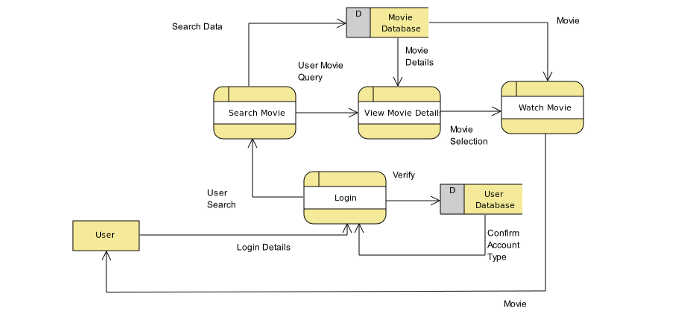


Figure 3.2: Data Flow Diagram

### **Sequence Diagram**

Sequence Diagrams are interaction diagrams that detail how operations are carried out. They capture the interaction between objects in the context of a collaboration.

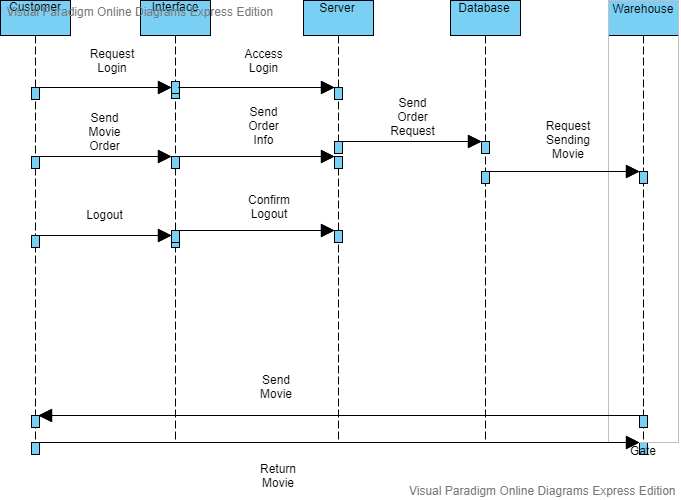


Figure 3.3: Sequence Diagram

### **Entity-Relationship Diagram**

An entity-relationship diagram (ERD) shows the relationships of entity sets stored in a database. An entity in this context is an object, a component of data. An entity set is a collection of similar entities. These entities can have attributes that define its properties.

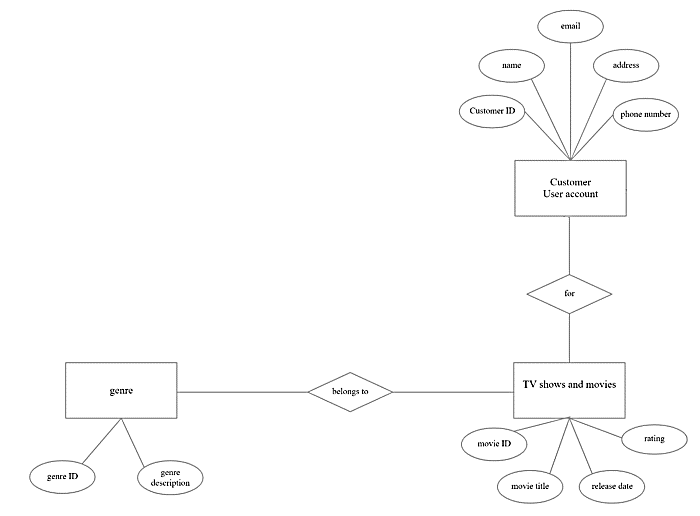


Figure 3.4: E.R Diagram

### **Architecture Diagram**

Class diagram is basically a graphical representation of the static view of the system and represents different aspects of the application. A collection of class diagrams represents the whole system. The name of the class diagram should be meaningful to describe the aspect of the system.

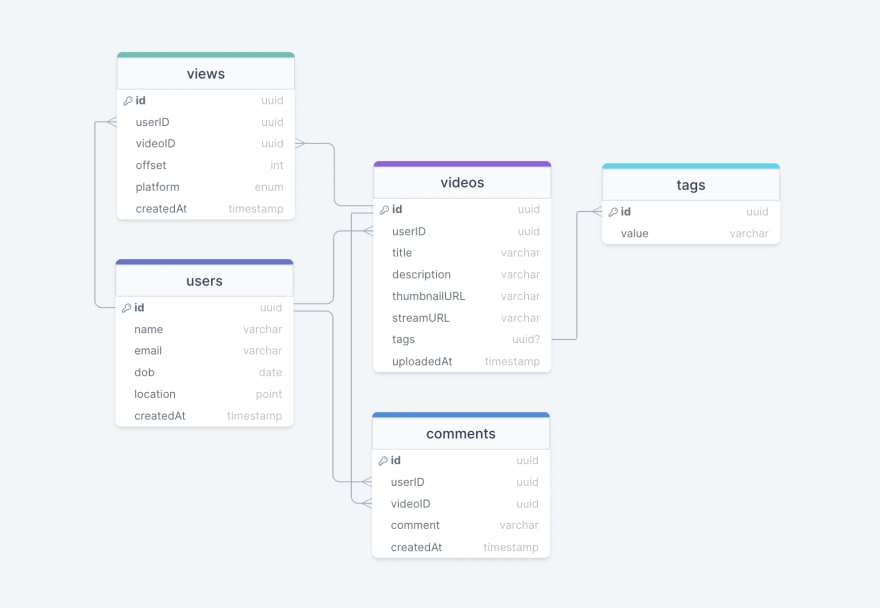


Figure 3.5: Class Diagram

## **Programming languages used**

1. Tailwind CSS
2. Python

**CHAPTER FOUR**

**SYSTEM IMPLEMENTATION EVALUATION**

**4.1 Introduction**

This section provides a comprehensive explanation of the implementation process for the new system, highlighting its efficiency and effectiveness. It presents practical instances of the functional aspects of the system and outlines the steps involved in its implementation.

* 1. **System Testing and Evaluation**

Testing the developed system is crucial for several reasons. One key purpose is to uncover any potential flaws within the system and devise appropriate solutions. In this project, a combination of unit and integration testing was employed to verify the effectiveness and efficiency of the design, ensuring that the new system fulfills its functional requirements without any errors.

**Unit Testing**

This part examines specific units or single components of the system individually to confirm that specific phases function properly and without problems.

**Integration Testing**

Integration testing was performed on the software, wherein all components were brought together and operated as a unified system. The objective of this testing was to validate the connectivity and proper integration of the various parts, ensuring seamless collaboration among the units.

**4.3 System Installation**

In order to use the proposed application on any computer system, the following steps need to be taken:

1. Make sure, pip, pipenv, and python3 or greater are installed on the system.
2. Copy your project folder to any location of your choice.
3. Open project folder in Visual Studio Code
4. On the terminal run “pipenv install -r requirements.txt”
5. On the terminal run “python manage.py runserver”
6. Open any browser on the system example Chrome, Microsoft Edge, or Mozilla Firefox.
7. On the address bar, type <http://127.0.0.1> and press the enter key the site should be loaded.

**4.4 Security Measures**

The application has a public scope, allowing all users to access the available information. However, certain functionalities are restricted to authenticated users, for example, the admin can create advertisements, create and manage the user’s account, etc. Access to these restricted functionalities is protected by passwords, ensuring that only authorized individuals can access the admin pages. Additionally, certain functionalities within the application may be restricted based on the specific user type, providing tailored access and permissions as needed.

**4.5 Sample Outputs**

These describe and give the pictorial representation of the program or software; it shows and gives a clear understanding of the design, and displays all the interfaces.

**Homepage**

The image provided illustrates the homepage, which serves as the initial page and serves as a gateway to navigate and explore the various sections and functionalities of the website.

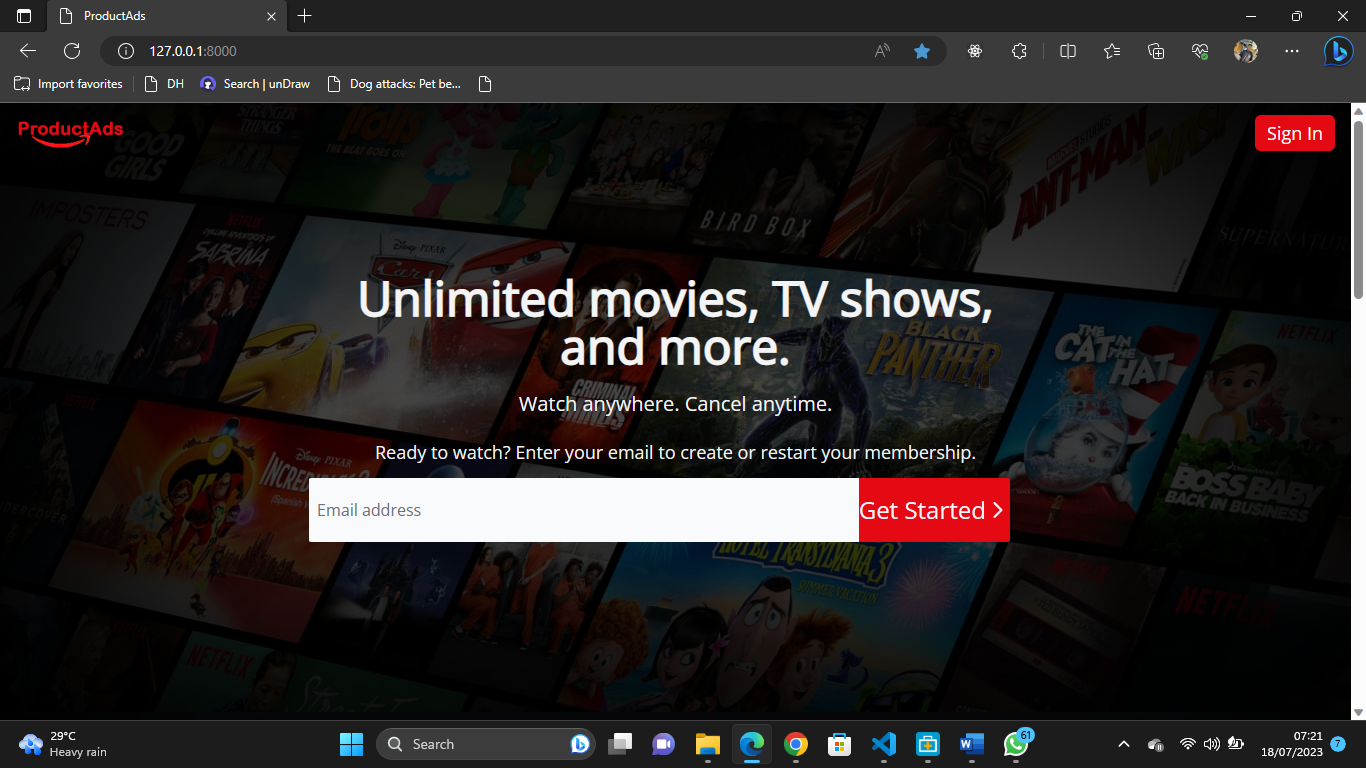


Fig 4.5.1 Homepage

**User Login**

This is a page that grants users (lecturers, students, and admin) access to the system only if the correct credentials are provided.

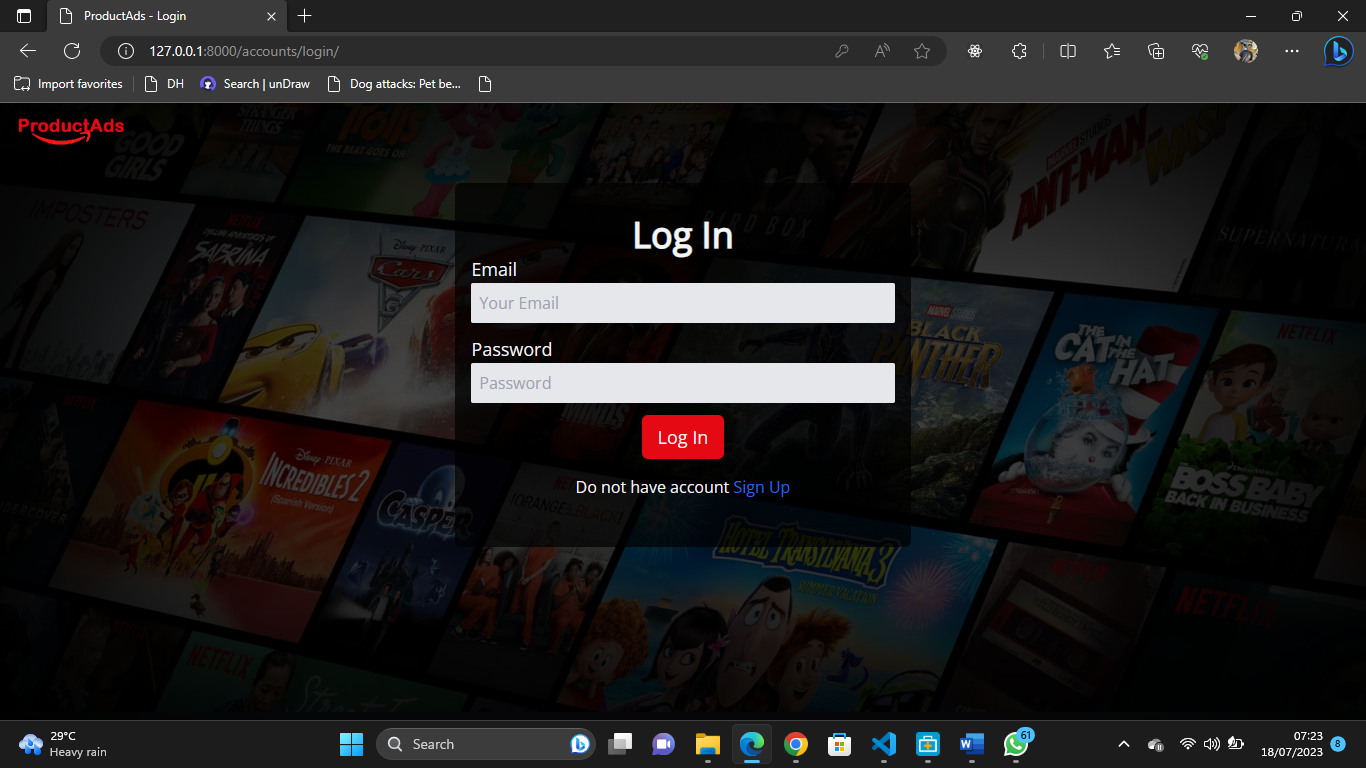


Fig 4.5.2 Login

**User Registration**

The registration page is only for users, they can create an account to access video streaming adverts.

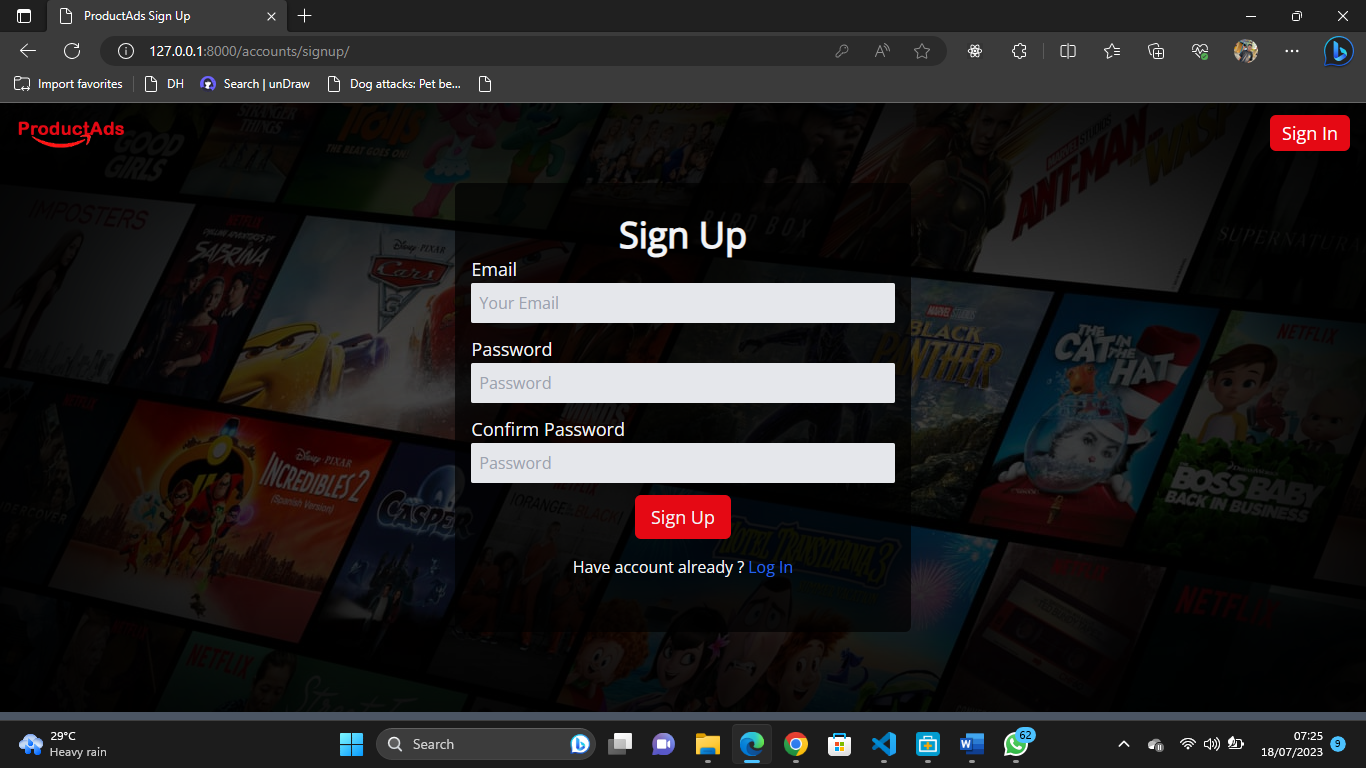


Fig 4.5.3 User Registration

**User Dashboard**

This is the user dashboard, the page shows the available functionality for the users

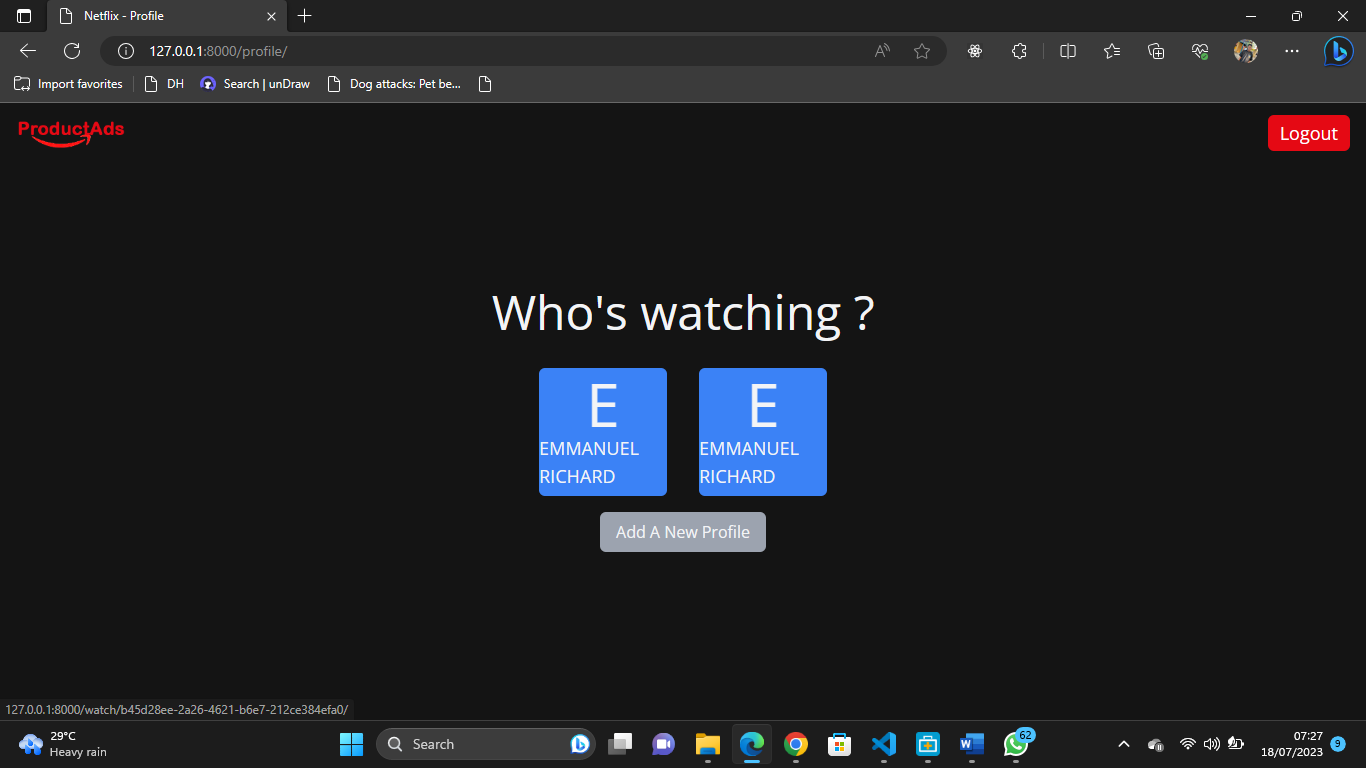


Fig 4.5.4 User Dashboard

**User Profile**

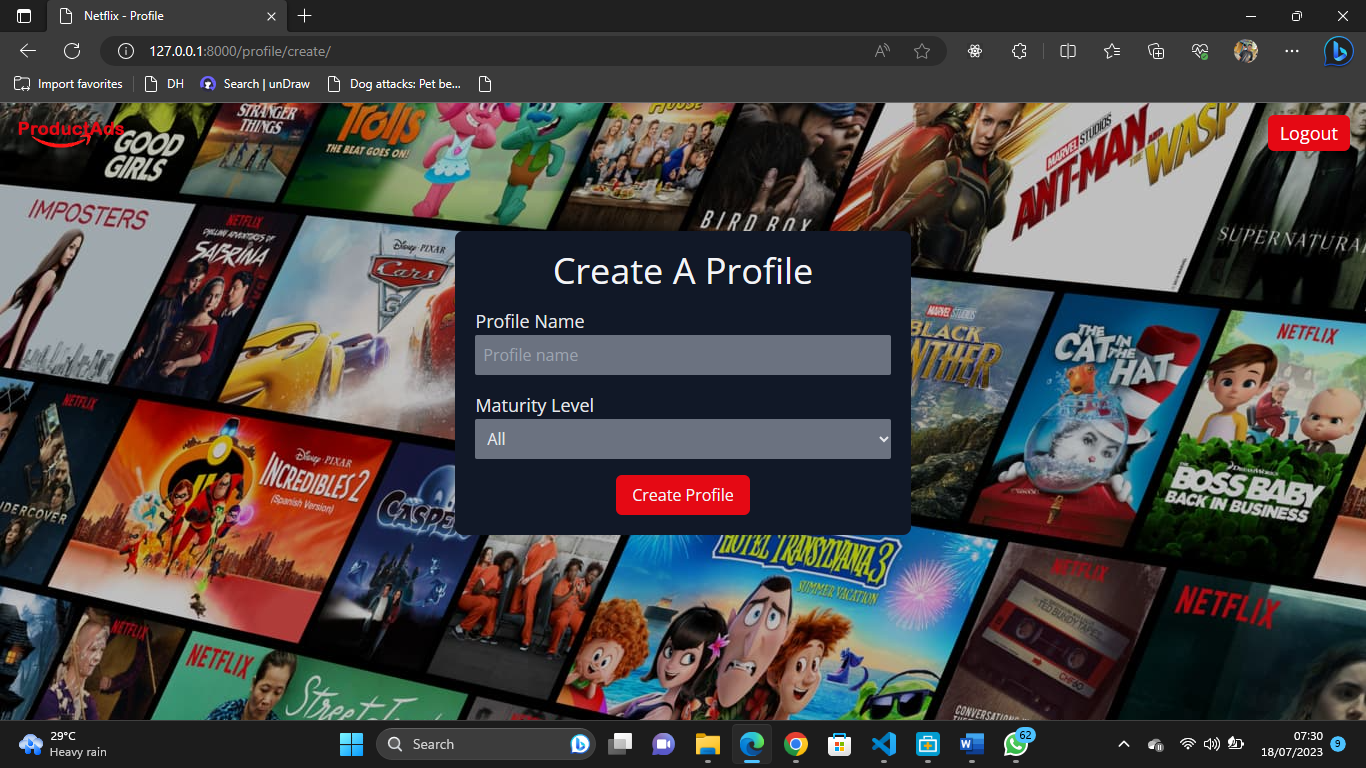
The authenticated user can make adjustments to his or her profile, select the maturity level of adverts that can be displayed to the user authenticated

Fig 4.5.5 User Profile

**Video Streaming**

The authenticate user can stream a video online and videos advert will be displayed at different intervals of the movie

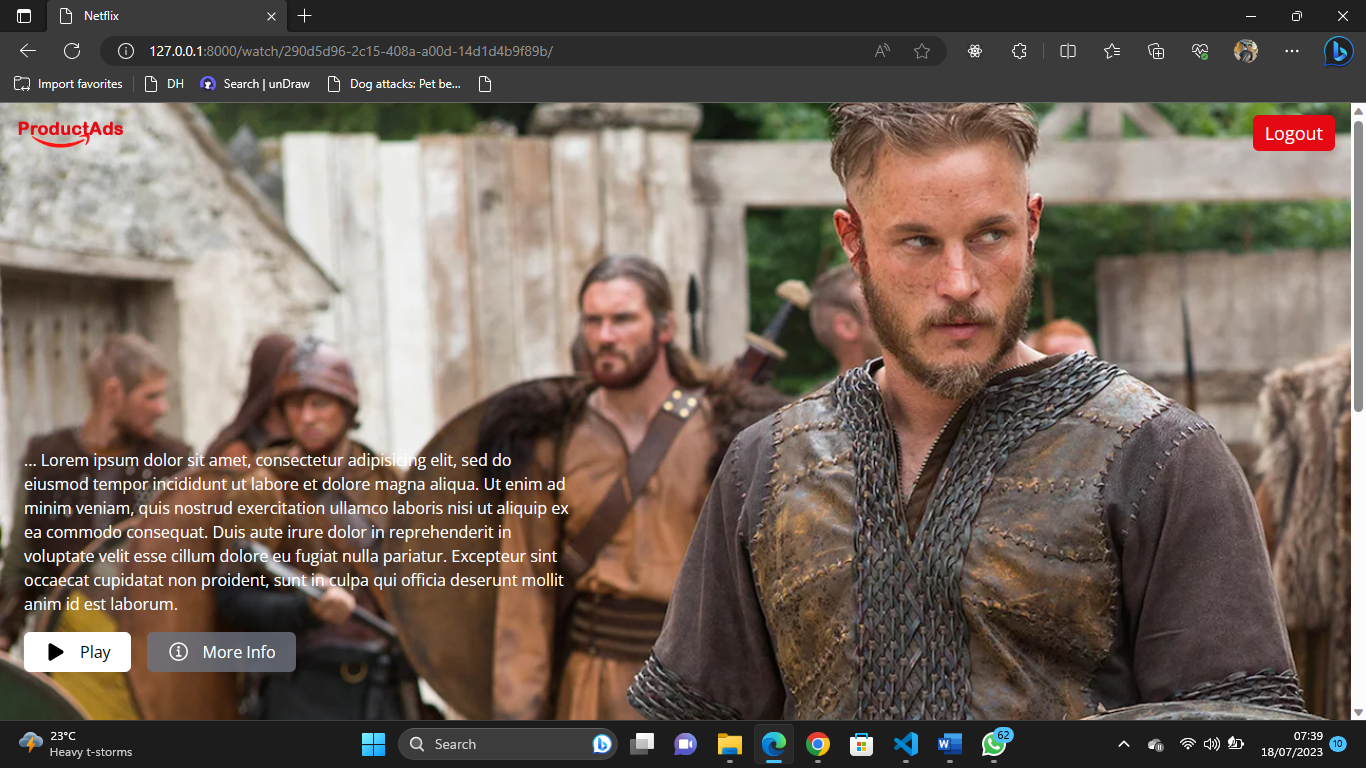


Fig 4.5.6 Video Streaming

**CHAPTER FIVE**

**SUMMARY CONCLUSION AND RECOMMENDATION**

**5.1 Summary**

This study addresses the challenges of video marketing and proposes solutions for effective advertising in the digital age. It highlights the shift from outsourced video production to in-house production, allowing businesses to have greater control over the content creation process. Creativity plays a crucial role in video production, with a focus on addressing targeted consumers' needs and using clever techniques to capture their attention and prompt a call to action. The study emphasizes the importance of humor, emotion, value, and immediacy in gaining viewership and achieving marketing goals. The placement of online video advertising also significantly impacts viewership. The project aims to create a video streaming platform for advertising products, providing users with creative content and guides on product usage. The platform's focus is on Video on Demand (VoD) functionality, with an engaging user interface and logic handled by Django. Limitations include no live streaming and absence of a payment gateway. The significance of the study lies in offering diverse advertising options and empowering customers to make informed choices.

**5.2 Conclusion**

In conclusion, this study highlights the evolving landscape of video marketing and the need for businesses to adapt their advertising strategies in the digital era. The shift towards in-house video production allows companies to have greater control over content creation, fostering creativity and enabling a more personalized approach to engage targeted consumers. By addressing consumers' needs and utilizing clever techniques to capture attention, videos can generate increased viewership and achieve marketing goals. The development of a video streaming platform for advertising products, with a focus on Video on Demand (VoD), offers the opportunity to provide users with engaging and informative content on product usage. While the project has limitations such as the absence of live streaming and a payment gateway, it still holds significance by offering diverse advertising options and empowering customers to make informed decisions. Moving forward, it is crucial to continue exploring innovative approaches in video marketing to effectively engage audiences and drive sales in an increasingly competitive digital landscape.

**5.3 Recommendation**

Based on the findings and objectives of this study, the following recommendations are proposed:

1. Continuously Improve Content Creativity: Emphasize the importance of creativity in video production. Encourage the in-house video production team to think outside the box and deliver content that captivates viewers' attention, utilizing humor, emotion, value, and immediacy. Regularly evaluate and refine the content strategy to ensure it remains relevant and engaging.
2. Optimize Video Placement: Pay attention to the placement of online video advertising. Consider strategic placements that align with the target audience's preferences and online viewing habits. Optimizing the placement will maximize viewership and completion rates, ultimately driving better results for advertisers.
3. Incorporate Analytics and User Feedback: Implement robust analytics tools to gather insights on user behavior, engagement, and preferences. Regularly analyze the data to gain a deeper understanding of user interactions with the platform and content. Additionally, actively seek and incorporate user feedback to continuously improve the user experience and address any pain points or suggestions.

By implementing these recommendations, the video streaming platform can further enhance its functionality, user experience, and advertising effectiveness. These improvements will contribute to the platform's success in engaging users, promoting products, and ultimately driving sales for the businesses utilizing the platform.

**REFERENCES**

Goel, S. (2013). Cloud-based mobile video streaming techniques. *International Journal of Wireless & Mobile Networks*, *5*(1), 85–92. https://doi.org/10.5121/ijwmn.2013.5107

Greenfield, I. (2022, December 9). *Streaming TV advertising: The Definitive Guide for Marketers*. MNTN. Retrieved December 20, 2022, from https://mountain.com/blog/streaming-tv-advertising/

hamishwillee, kruschk, & ozgurturkiye, ranganathan. (2022). *Django Introduction - Learn Web Development: MDN*. Learn web development | MDN. Retrieved December 13, 2022, from https://developer.mozilla.org/en-US/docs/Learn/Server-side/Django/Introduction

Hemachandran, K., & Gunashanthi, M. (2013). Creating A Video Streamer App On Android Mobile Phones. *International Journal of Engineering Research & Technology (IJERT)*, *2*(1), 1–6.

Kalle, B., & Kalle, R. (2017). *Online Video as a Marketing Tool* (thesis). RecOn Productions Oy, Jyväskylä.

Khan, M. A., Baccour, E., Chkirbene, Z., Erbad, A., Hamila, R., Hamdi, M., & Gabbouj, M. (2022). A Survey on Mobile Edge Computing for Video Streaming: Opportunities and Challenges. *ArXiv*, *1*.

Lessard, T. (2017, January 10). *Three major shifts that will transform video marketing in 2017*. MarketingProfs. Retrieved December 20, 2022, from https://www.marketingprofs.com/articles/2017/31378/threemajor-shifts-that-will-transform-video-marketing-in-2017

Londhe , R. D., Thakare, V. M., & Sherekar , S. S. (2016). A Survey of Video Streaming in Mobile Cloud Computing. *International Journal of Scientific & Engineering Research*, *7*(2), 374–377.

Marshall, C. (2016). *The Top Facebook Video Publishers: November 2016 Most Popular*. Tubularinsight.com. Retrieved December 20, 2022, from http://tubularinsights.com/top-facebook-video-creators/

Murdico, D. (2017). *Video Marketing 2017 - Cheaper And More Frequent!*. Business2community.com. Retrieved December 20, 2022, from http://www.business2 community.com/video-marketing/video-marketing-2017-cheaper-frequent-01756781.

Sapna, & Sachin, Y. (2021). *Introduction to tailwind CSS*. GeeksforGeeks. Retrieved December 13, 2022, from https://www.geeksforgeeks.org/introduction-to-tailwind-css/

Smith, A. (2016). *The Most-Popular Instagram Video Creators: March 2016 Leaderboard*. Tubularinsights.com. Retrieved December 20, 2022, from <http://tubularinsights.com/> top-instagram-video-creators/

Suganthi, S., Sharmila, M. F., & Kannan, E. (2013). Literature Survey on Streaming and Sharing of Videos in Mobile Network. *International Journal of Engineering Research & Technology (IJERT)*, *2*(10), 1413–1416.

**APPENDIX**

**Views.py**

from django.shortcuts import redirect, render

from django.views import View

from django.contrib.auth.decorators import login\_required

from django.utils.decorators import method\_decorator

from .forms import ProfileForm

from .models import Movie, Profile

class Home(View):

    def get(self,request,\*args, \*\*kwargs):

        if request.user.is\_authenticated:

            return redirect(to='/profile/')

        return render(request,'index.html')

@method\_decorator(login\_required,name='dispatch')

class ProfileList(View):

    def get(self,request,\*args, \*\*kwargs):

        profiles=request.user.profiles.all()

        print(profiles)

        return render(request,'profileList.html',{

            'profiles':profiles

        })

@method\_decorator(login\_required,name='dispatch')

class ProfileCreate(View):

    def get(self,request,\*args, \*\*kwargs):

        form=ProfileForm()

        return render(request,'profileCreate.html',{

            'form':form

        })

    def post(self,request,\*args, \*\*kwargs):

        form=ProfileForm(request.POST or None)

        if form.is\_valid():

            print(form.cleaned\_data)

            profile = Profile.objects.create(\*\*form.cleaned\_data)

            if profile:

                request.user.profiles.add(profile)

                return redirect(f'/watch/{profile.uuid}')

        return render(request,'profileCreate.html',{

            'form':form

        })

@method\_decorator(login\_required,name='dispatch')

class Watch(View):

    def get(self,request,profile\_id,\*args, \*\*kwargs):

        try:

            profile=Profile.objects.get(uuid=profile\_id)

            movies=Movie.objects.filter(age\_limit=profile.age\_limit)

            try:

                showcase=movies[0]

            except :

                showcase=None

            if profile not in request.user.profiles.all():

                return redirect(to='core:profile\_list')

            return render(request,'movieList.html',{

            'movies':movies,

            'show\_case':showcase

            })

        except Profile.DoesNotExist:

            return redirect(to='core:profile\_list')

@method\_decorator(login\_required,name='dispatch')

class ShowMovieDetail(View):

    def get(self,request,movie\_id,\*args, \*\*kwargs):

        try:

            movie=Movie.objects.get(uuid=movie\_id)

            return render(request,'movieDetail.html',{

                'movie':movie

            })

        except Movie.DoesNotExist:

            return redirect('core:profile\_list')

@method\_decorator(login\_required,name='dispatch')

class ShowMovie(View):

    def get(self,request,movie\_id,\*args, \*\*kwargs):

        try:

            movie=Movie.objects.get(uuid=movie\_id)

            movie=movie.videos.values()

            return render(request,'showMovie.html',{

                'movie':list(movie)

            })

        except Movie.DoesNotExist:

            return redirect('core:profile\_list')

Index.html

{% extends 'base.html' %}

{% load static %}

{% block title %}

    ProductAds

{% endblock title %}

{% block others %}

<script src="https://unpkg.com/ionicons@5.4.0/dist/ionicons.js"></script>

{% endblock others %}

{% block content %}

    {% include "partials/navbar.html" %}

    <header class="min-h-screen bg-black bg-opacity-75 showcase border-gray-600 border-b-8 flex justify-center items-center bg-no-repeat"  style="background-image: url('{% static 'background\_netflix.jpg' %}');">

        <div class="z-10 h-full py-32 md:py-0 text-gray-100 text-center flex items-center flex-col">

            <h1 class="font-semibold text-4xl md:text-5xl w-10/12 md:w-3/4 text-center">

                Unlimited movies, TV shows, and more.

            </h1>

            <h3 class="text-xl md:text-xl py-5">

                Watch anywhere. Cancel anytime.

            </h3>

            <h4 class="text-sm md:text-lg pb-3">

                Ready to watch? Enter your email to create or restart your membership.

            </h4>

            <div class="w-full flex justify-center">

                <div class="flex justify-center items-center w-11/12 md:w-10/12">

                    <input type="email" name="" id="" placeholder="Email address" class="md:py-5 py-4 px-2 placeholder-gray-500 rounded-tl-sm rounded-bl-sm outline-none  text-gray-900 bg-gray-50 block w-8/12 md:w-3/4">

                    <a href='{% url "account\_signup" %}' class='md:w-1/4'>

                        <button class="bg-primary\_red font-medium md:font-normal flex justify-center items-center py-4 px-2 md:px-0 rounded-br-sm rounded-tr-sm md:text-2xl">

                            <span>

                                Get Started

                            </span>

                            <ion-icon name="chevron-forward-outline" class="text-2xl"></ion-icon>

                        </button>

                    </a>

                </div>

            </div>

        </div>

    </header>

    <section class='flex flex-col md:flex-row items-center border-gray-900 border-b-8' style="min-height: 70vh;">

        <div class="md:w-1/2 flex flex-col justify-center text-center md:text-left p-5 md:p-10 ">

            <h2 class="md:text-5xl text-4xl font-semibold mb-4">

                Enjoy on your TV.

            </h2>

            <h5 class="md:text-xl text-lg">

                Watch on Smart TVs, Playstation, Xbox, Chromecast, Apple TV, Blu-ray players, and more.

            </h5>

        </div>

        <div class='md:w-1/2 flex justify-center items-center'>

            <img src="{% static 'tv.png' %}" class="w-8/12" alt="">

        </div>

    </section>

    <section class='flex flex-col md:flex-row items-center border-gray-900 border-b-8' style="min-height: 70vh;">

        <div class='md:w-1/2 flex justify-center items-center '>

            <img src="{% static 'mobile-0819.jpg' %}" class="w-8/12" alt="">

        </div>

        <div class="md:w-1/2 flex flex-col justify-center text-center md:text-left p-5 md:p-10">

            <h2 class="md:text-5xl text-4xl font-semibold mb-4">

                Download your shows to watch offline.

            </h2>

            <h5 class="md:text-xl text-lg">

                Save your favorites easily and always have something to watch.

            </h5>

        </div>

    </section>

    <section class='flex flex-col md:flex-row items-center border-gray-900 border-b-8' style="min-height: 70vh;">

        <div class="md:w-1/2 flex flex-col justify-center text-center md:text-left p-5 md:p-10">

            <h2 class="md:text-5xl text-4xl font-semibold mb-4">

                Create profiles for kids.

            </h2>

            <h5 class="md:text-xl text-lg">

                Send kids on adventures with their favorite characters in a space made just for them—free with your membership.

            </h5>

        </div>

        <div class='md:w-1/2 flex justify-center items-center'>

            <img src="{% static 'netflix\_kid.png' %}" class="w-8/12" alt="">

        </div>

    </section>

    <footer class=" px-5 py-10 ">

        <ul class="flex justify-between items-center">

            <li>&copy; 2021</li>

            <li> Made with Django and Tailwind</li>

            <li>By Abdulazeez</li>

        </ul>

    </footer>

{% endblock content %}

Models.py

from django.db import models

from django.contrib.auth.models import AbstractUser

import uuid

AGE\_CHOICES=(

    ('All','All'),

    ('Kids','Kids')

)

MOVIE\_TYPE=(

    ('single','Single'),

    ('seasonal','Seasonal')

)

class CustomUser(AbstractUser):

    profiles=models.ManyToManyField('Profile')

class Profile(models.Model):

    name=models.CharField(max\_length=225)

    age\_limit=models.CharField(max\_length=5,choices=AGE\_CHOICES)

    uuid=models.UUIDField(default=uuid.uuid4,unique=True)

    def \_\_str\_\_(self):

        return self.name +" "+self.age\_limit

class Movie(models.Model):

    title:str=models.CharField(max\_length=225)

    description:str=models.TextField()

    created =models.DateTimeField(auto\_now\_add=True)

    uuid=models.UUIDField(default=uuid.uuid4,unique=True)

    type=models.CharField(max\_length=10,choices=MOVIE\_TYPE)

    videos=models.ManyToManyField('Video')

    flyer=models.ImageField(upload\_to='flyers',blank=True,null=True)

    age\_limit=models.CharField(max\_length=5,choices=AGE\_CHOICES,blank=True,null=True)

class Video(models.Model):

    title:str = models.CharField(max\_length=225,blank=True,null=True)

    file=models.FileField(upload\_to='movies')

Advert Detail Page

{% extends 'base.html' %}

{% load static %}

{% block title %}

Detail Page

{% endblock title %}

{% block others %}

<script src="https://unpkg.com/ionicons@5.4.0/dist/ionicons.js"></script>

{% endblock others %}

{% block content %}

{% include 'partials/navbar.html' %}

<main class='bg-primary\_black min-h-screen w-full'>

<section class="h-screen relative bg-no-repeat pb-8 pl-6 flex items-end bg-cover" style="background-image: url('{{movie.flyer.url}}');">

<div>

<h2 class="text-6xl font-semibold text-white my-4">

{{movie.title}}

</h2>

<p class=" w-11/12 md:w-5/12 font-medium text-white my-4">

{{movie.description |slice:":50"}}...

{% lorem 1 %}

</p>

<div class="flex my-4">

<a href="{% url 'core:play' movie\_id=movie.uuid %}">

<button class="flex items-center bg-white py-2 px-5 rounded-md" style='color:black;'>

<ion-icon name="play" class="text-2xl"></ion-icon>

<span class="ml-3 font-medium">

Play

</span>

</button>

</a>

</div>

</div>

</section>

<section class='bg-primary\_black min-h-screen flex justify-center py-10'>

<div class="p-8 w-10/12 relative" >

{% if movie.type == 'seasonal' %}

<div style='margin-top:2rem;margin-bottom:2rem;'>

<h2 class="text-gray-200 text-3xl font-medium">

Episode

</h2>

{% for episode in movie.videos.all %}

<h3 class='text-lg'>

<a href="{% url 'core:play' movie\_id=movie.uuid %}?epi={{forloop.counter0}}">

{{forloop.counter}} - {{episode.title}}

</a>

</h3>

{% endfor %}

</div>

{% endif %}

<h2 class="text-gray-200 text-3xl font-medium">

Description

</h2>

<div class="flex justify-center items-center min-w-full movie\_lis">

<p class="text-gray-100 text-lg">

{{movie.description}}

</p>

</div>

</div>

{% endblock content %}