ASHESI

ASHESI UNIVERSITY

A CASH AND TOKEN PAID ENTERTAINMENT CONTENT STREAMING SITE

APPLIED PROJECT

B.Sc. Computer Science

Angela Musangi Munyao

2020

ASHESI UNIVERSITY

A Tokenized Content Streaming Site with Ad Hosting

APPLIED PROJECT

Applied Project submitted to the Department of Computer Science, Ashesi University in partial fulfilment of the requirements for the award of Bachelor of Science degree in Computer Science

Angela Musangi Munyao

May 2020

DECLARATION

I hereby declare that this applied project is the result of my own original work and that no part of it has been presented for another degree in this university or elsewhere.

Candidate's Signature

………………………………………………………………………………………………………

Candidate's Name

………………………………………………………………………………………………………

Date:

………………………………………………………………………………………………………

I hereby declare that preparation and presentation of this applied project were supervised in accordance with the guidelines on supervision of applied project laid down by Ashesi University

Supervisor's Signature

………………………………………………………………………………………………………

Supervisor's Name

………………………………………………………………………………………………………

Date:

………………………………………………………………………………………………………

Acknowledgment

I want to thank my supervisor, Mr. David Hutchful, for his valuable insight into the project. I would also like to thank my family for encouragement and support.

Abstract

People that cannot afford subscription to paid and ad-free media hosting sites always have to deal with random and un-seamless popping up of ads as they browse, while viewing content on free media hosting sites, and mostly, these free media hosting sites lack current content such as recently released movies or songs.

The solution proposed in this paper is a cash and token paid entertainment content streaming site with undisruptive ad hosting. In this site, users can access up to date content (songs, movies, comedy clips, and TV Shows), all they need to do is pay for the content in tokens, and they obtain the tokens by either converting their cash into tokens, or by watching ads hosted on a specified page within the site (Users can make a choice to visit this ads page or not, and hence do not incur random popping up of ads while browsing). The aim of the solution is to accommodate users that cannot afford money-paid media hosting sites but desire to watch up to date content with minimum or no disruption from ads, and it achieves so by creating a fair playing field between ad-owners and content viewers by sharing some of the revenue generated from hosting ads with the ad-viewers.

.

Table of Contents

List of Figures

Chapter 1: Introduction

1.1 Introduction

Since the invention of the internet forty years ago [1], the world has seen a massive invention of online entertainment sites, and a good number of us rely on these sites to stream entertainment content, however, people that cannot afford subscription to paid and ad-free media hosting/entertainment sites mostly opt for free media hosting sites, where they always have to deal with lack of up to date content, and random and un-seamless popping up of ads as they browse. Moreover, because ads come up to computer users as a distraction, they mostly end up being skipped, leading to wastage of resources utilised in hosting them. Content creators hosting their content on these free sites hoping to earn from ad views occasionally end up losing out, because viewers skip the ads and navigate directly to the hosted content.

All people need to be given a chance at accessing up to date entertainment content irrespective of whether or not they can afford such content with monetary payment, and ads need to be promoted in a seamless manner that doesn't come up as a disruption to target viewers.

This paper introduces a system that ensures that, people that cannot afford cash subscription for content can get access to up to date content by viewing ads. The system aims at ensuring a fair playing field between ad owners and content viewers, by rewarding ad views with tokens. These ads are only accessible on a certain page within the system and hence do not randomly interrupt users while browsing. While this system also promotes viewing of ads and eliminates resource wastage, it also ensures content owners such as music artists and movie creators that post their content on the site are able to directly monitor their content earnings, because they will earn directly based on the number of views their content amasses.

1.2 Background

Streamed content is content sent in compressed form over the Internet and displayed by the viewer in real time [2]. A content streaming site is any site that internet users can view content such as songs, movies, pictures, etcetera. A good example is YouTube and Spotify, and other social media platforms like Facebook and Instagram. Content streaming sites are often either paid, where users pay for the content in monetary terms, or unpaid, where users do not pay, but may encounter several ads as they browse through such sites, where they may too, not find up to date or highly sought after content. Highly sought-after content could be a newly released film for example, that is most likely made available to the paid sites before later getting to the unpaid sites. Over a long span of time such users have often been compelled to watch interrupting adverts without any direct benefit of viewing such adverts, and as a result, their mentality of ads has over time taken a negative dive with most of them viewing ads as not only annoying, but also time consuming, and wasteful; this is also evident in questionnaires send out to users of different age groups and occupations, and most of them cited finding ads as a disruption while surfing most sites on the internet. Up to 50% of the average user's mobile data is for ads and trackers, costing as much as $23 a month. Ads use about 5 seconds of mobile load time on average, they decrease phone battery life by as much as 21% [3]. As a result, most ads do not get watched as expected. Over 600 million phones and desktops run ad-blocking [3]. Content creators hosting their content on these free sites hoping to earn from ad views end up losing out, because viewers skip the ads and navigate directly to the hosted content.

1.3 Motivation

According to my research, the desire of many online entertainment enthusiast (who are my main concern for his project) is that they would be capable of viewing any new and worthwhile content, at the moment it is made available online, seamlessly, without any disruption such as the one that random ads bring along. The desire of most ad owners is that their ads would be viewed by their target audience as many times as possible, and the desire of content creator e.g movie creators, is that they would be able to earn maximumly from their online content. Why not create a solution that strikes a fair balance between these three players, ensuring that each of them has their desires fulfilled? This motivated a further research and exploration into such a project as a probable solution.

1.4 Significance

Users can view up to date entertainment content without random and uncalled for disruption from ads, ad viewing will improve as ad-views will get rewarded in tokens, and content creators will get full credit for their content , as they will earn directly from the number of tokens amassed from views on their posts.

1.5 Related Work

1.5.1 BAT (Basic Attention Blockchain Tokenisation System) by Brave browser.

With an aim of ensuring users browsing the internet on the Brave browser do not feel hard hit by the disruption of ads within the browser, users are rewarded with BAT tokens for their attention on ads and content within the brave browser, content publishers receive BAT tokens based on users' attention on their content, and advertisers achieve increased target audience attention in the process. Users can make use of their BAT tokens to view on premium content on the Brave browser. [3]

The proposed system not only borrows from Brave's research, but also attempts a different model of user navigation, by ensuring advertisers decide on the number of tokens to reward their ad viewers per each ad posting, and content creators/publishers decide on the number of tokens to charge users wanting to view their posted content. The user navigates by viewing ads on the ads page so to earn tokens, then using earned tokens to view hosted content on the content page.

1.5.2 Spotify’s Thirty Free Minutes Incentives

Spotify is an audio music streaming site that also provides its regular users with an option to watch a video ad and receive thirty free minutes to stream audio content on the site, uninterrupted. This ensures that users that are freely subscribed enjoy ad-free music streaming without a monetary exchange. The user may watch as many video ads as they wish, so to earn more free minutes [4, 5].

The proposed system however aims at having more than audio content, as users wishing to stream non audio content ad-free and without a monetary exchange need to be covered too.

Chapter 2: Requirements

2.1 Overview

This chapter delves deep into analysing the requirements of the proposed system (Functional and non-functional), and how they were obtained, the target users, and an analysis of various scenarios of users using the proposed site.

2.2 Requirements Gathering & Analysis

Requirements for the proposed system were verified through questionnaires, interviews and from research. From interviews and questionnaires, it was notable that computer users do not enjoy popping up of adverts, and also notable that there are a good number of users that cannot afford access to paid content hosting sites.

According to a research done by Brave:

- Up to 50% of the average user’s mobile data is for ads and trackers, costing as much as $23 a month [3].
- Ads use about 5 seconds of mobile load time on average [3].
- Ads decrease phone battery life by as much as 21% [3].
- Over 600 million phones and desktops run ad-blocking [3].

2.3 User Identification

The target users include people looking out for entertainment through songs, movies, TV Shows and comedy clips, people looking to sell their music or movies or comedy, or TV Shows, and people looking to host advert (Ads) for marketing purposes.

Users need a computer and internet connection to be able to access the web application, and access to a credit card or a mobile money account for Ghanaian Nationals, an M-Pesa account for Kenyan nationals and a Ugandan Mobile money account for Ugandan nationals, should they need to convert cash into tokens or vice versa.

2.4 Scenarios

2.4.1 Scenario 1

Priscilla wishes to watch a certain movie released by her favourite actors. On searching through the proposed system, she finds it listed under movies with a requirement of a certain number of tokens for viewers to be able to see it. For her to watch it, she needs to pay some tokens, which she does not have. To gain tokens, she navigates to the ads section and scrolls through, watching adverts so to earn the reward tokens. She then uses them to pay for the movie and voila! she watches it.

2.4.2 Scenario 2

Shadrack is a talented musician looking to sell his music. He creates an account on the proposed system and logs in, then uploads his song with a restriction on the number of tokens that must be given to the song by a user willing to listen to it. He then sits back and waits for his song to earn tokens, which he can then withdraw as cash.

2.4.3 Scenario 3

Toni owns a firm offering financial consultancy services to start-ups. To advertise his firm, he creates an advert. He then creates an account on the proposed system and logs in. He purchases a desired number of tokens from the system, to be attached to his advert. He then uploads his advert and lists the number of tokens to be earned by any viewer that watches it.

2.5 Functional Requirements

2.5.1 User Requirements.

- All ad owners should be able to upload their ads.
- Ad Owners should be able to set the number of tokens to be awarded to anyone that views their advert.
- Entertainment content posters should be able to set the number of tokens to be awarded to their posts per each view.
- Any user owning some tokens should be able to cash them out if their number is up to the system's set limit.
- A user watching an advert should be rewarded in tokens.
- A user watching entertainment content must pay in tokens.
- All users should be able to purchase tokens from the system. (Top up their token balance with cash payment)

2.5.2 System Requirements.

- The system must make use of a search algorithm to make it easier for users to search entertainment content of their choice, and to search through ads should they prefer a certain ad.
- The system must present the token balance to users at all time.
- The system should always be able to pay out any user cashing out their tokens.

2.6 Non-Functional Requirements

- The system must be reliable.

- The system must always be available.
- The system must always be accurate.
- The system must provide up-to-date information.
- The system must be secure.
- The system must be cross platform.

Chapter 3: Architecture & Design

3.1 Project Domain

The project will be in the form of a web application, and this chapter describes the design and architecture used in its implementation. The web application will be developed using a model View controller architecture.

3.2 High-Level Architecture

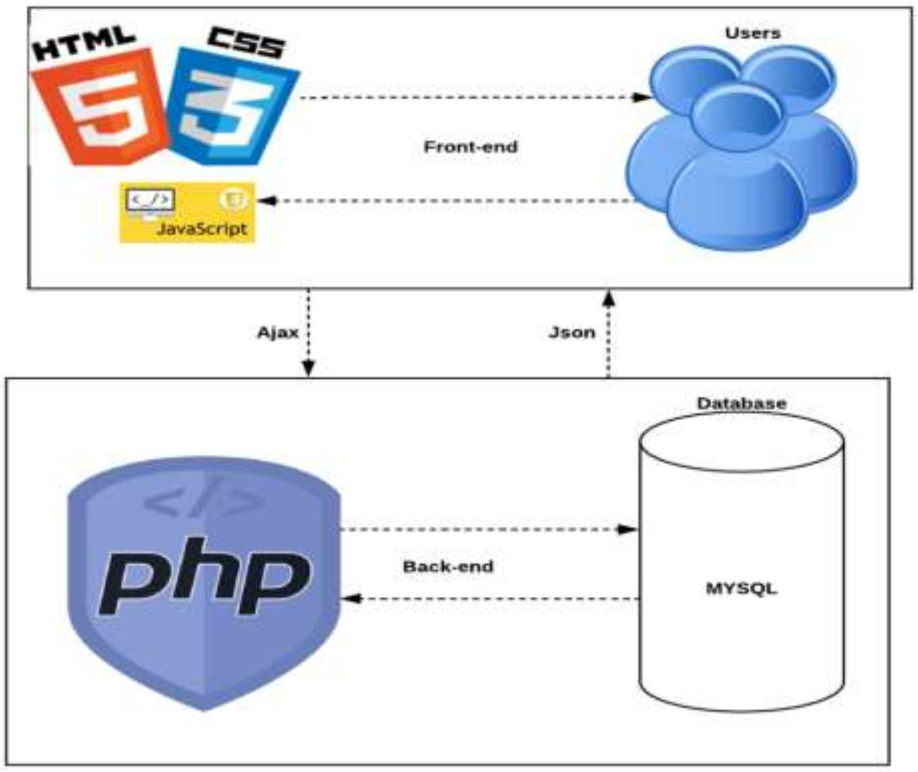


Figure 1: High-level architecture

3.3 Overview of the Model View Controller (MVC) Architecture

"The MVC is an architectural pattern that specifies that an application consists of a data model, presentation information (View), and control information (Controller), and the pattern requires that each of these be separated into different objects" [7].

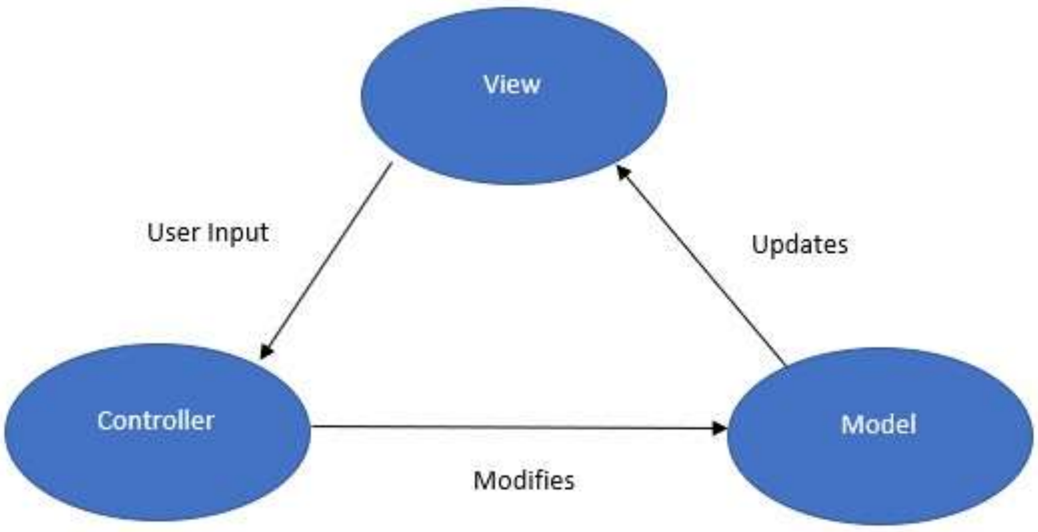


Figure 2: Overview of the MVC Architecture

3.3.1 The Model

This entails data related logic. It will define interaction with MYSQL database (SELECT, INSERT, UPDATE, DELETE) used in this application. It will communicate with the controller, and will sometimes, update the View [7].

3.3.2 The View

This is what the user will be seeing and will be made of HTML and CSS. It will communicate with the controller and will be able to pass values from the controller [7].

3.3.3 The Controller

It will receive input from View, process requests such as (GET, POST, PUT, DELETE), get data from the Model, and also pass this data to the View for display.

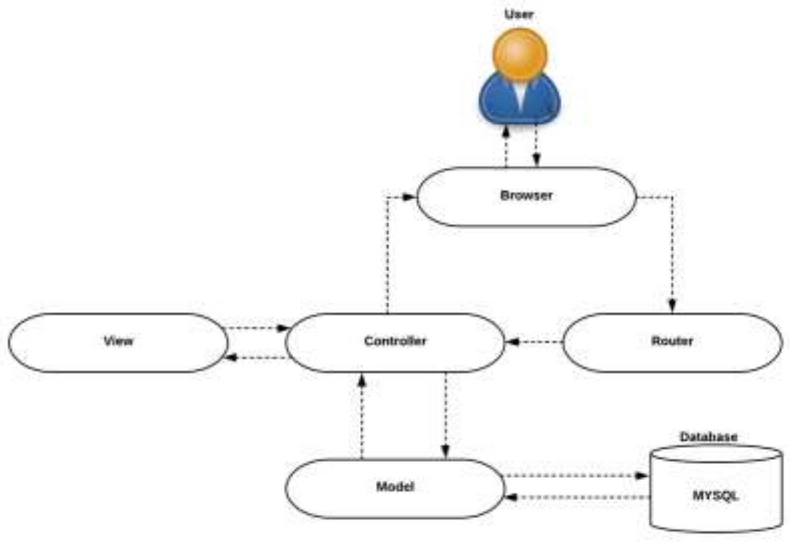


Figure 3: MVC Controller

3.4 Technologies & Frameworks

The technologies and frameworks used are:

HTML -It is the standard mark-up language for Web pages [8].

CSS -It describes how HTML elements are to be displayed on screen, paper, or in other media [9].

jQuery -It is a JavaScript Library that simplifies JavaScript programming [10].

JavaScript -It is a scripting language, primarily used on the Web. It is used to enhance HTML pages and is commonly found embedded in HTML code [11].

PHP -It is a server scripting language, and a usually powerful tool for making dynamic and interactive Web pages [12].

AJAX -It is a technology that entails exchanging data with a server, and updates parts of a web page, without reloading the whole page [13].

MySQL – It is an open Source Relational SQL Database Management System [14].

FlutterWave Payment API – This is a payment API developed by FlutterWave to enhance online payments [15].

3.5 Security & Backup

- All user passwords will be encrypted before being stored in the database.
- All Money transaction will be done through a trusted merchant (FlutterWave API), because it has put strong encryption mechanisms to ensure security of user credit cards.

Chapter 4: Implementation

4.1 Overview of Implementation

This chapter presents the various functionalities implemented in the web application, as well as the implementation process.

4.2 Structure of the Main Folder

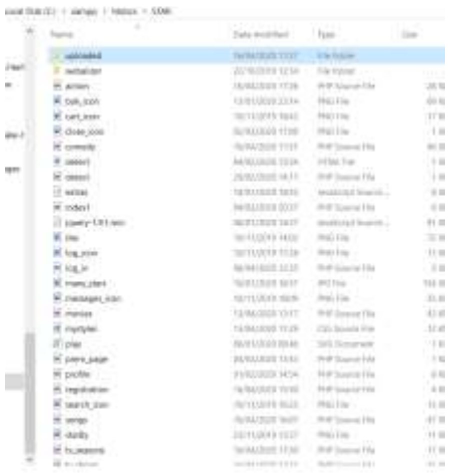
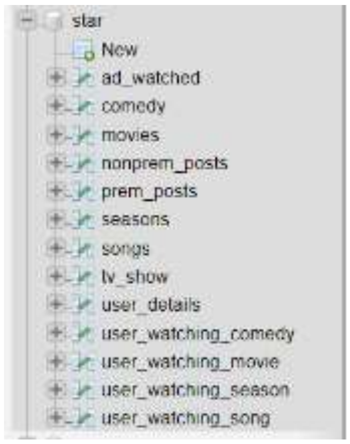


Figure 4: Main Folder Structure

4.3 Database Structure.



star
New
ad_watched
comedy
movies
nonprem_posts
prem_posts
seasons
songs
tv_show
user_details
user_watching_comedy
user_watching_movie
user_watching_season
user_watching_song

Figure 5: Database Structure

4.4 User Interface Design & Key Functionalities

4.4.1 Landing Page

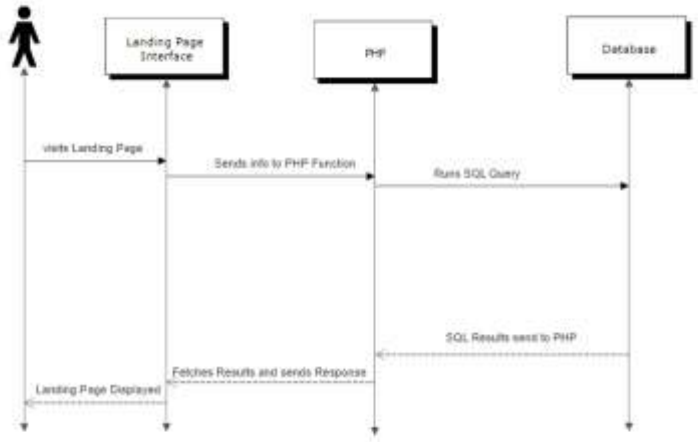


Figure 6: Landing Page Sequence Diagram

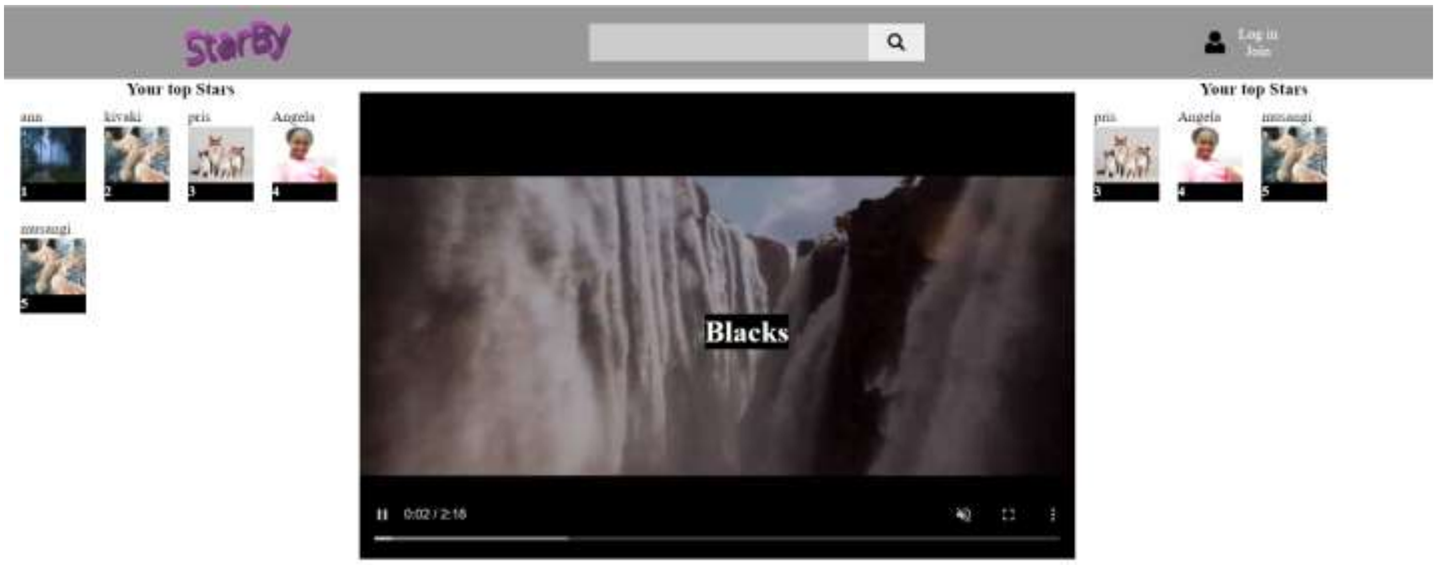


Figure 7: Landing Page Interface

4.4.2 Sign Up Page

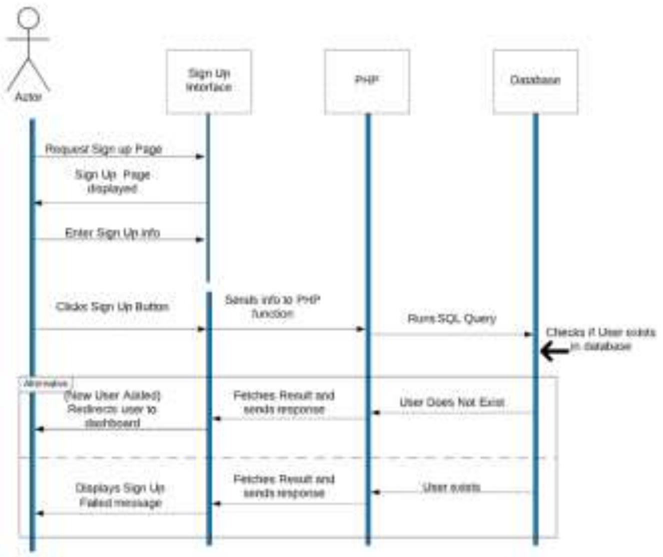
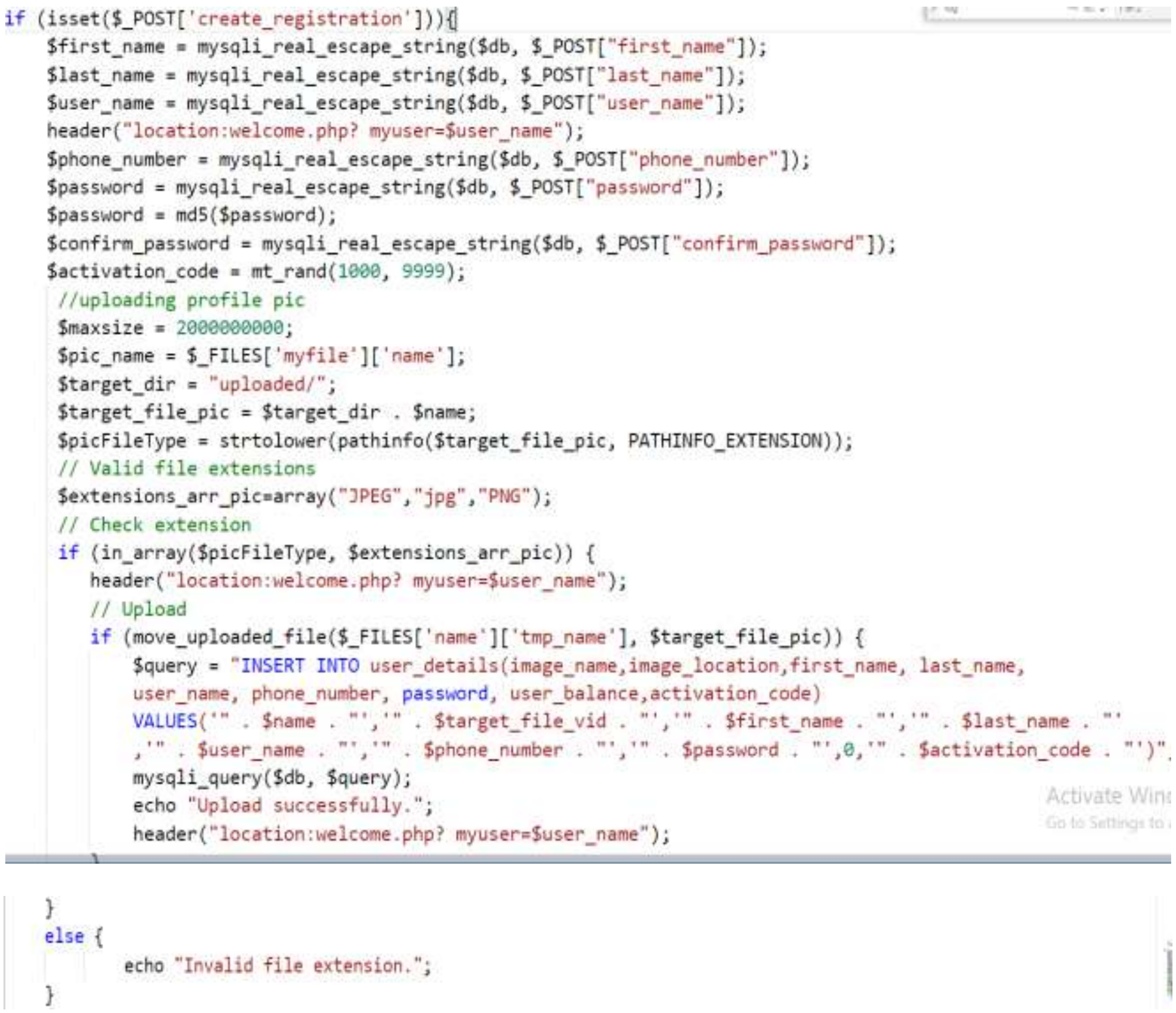

Actor
Sign Up Interface
PHP
Database
Request Sign up Page
Sign Up Page displayed
Enter Sign Up info
Clicks Sign Up Button
Sends info to PHP function
Runs SQL Query
Checks if User exists in database
Alternative
(New User Added) Redirects user to dashboard
Fetches Result and sends response
User Does Not Exist
Displays Sign Up Failed message
Fetches Result and sends response
User exists

Figure 8: Sign Up Page Sequence Diagram



```
if (isset($_POST['create_registration'])){
    $first_name = mysqli_real_escape_string($db, $_POST["first_name"]);
    $last_name = mysqli_real_escape_string($db, $_POST["last_name"]);
    $user_name = mysqli_real_escape_string($db, $_POST["user_name"]);
    header("location:welcome.php? myuser=$user_name");
    $phone_number = mysqli_real_escape_string($db, $_POST["phone_number"]);
    $password = mysqli_real_escape_string($db, $_POST["password"]);
    $password = md5($password);
    $confirm_password = mysqli_real_escape_string($db, $_POST["confirm_password"]);
    $activation_code = mt_rand(1000, 9999);
     //uploading profile pic
     $maxsize = 2000000000;
     $pic_name = $_FILES['myfile']['name'];
     $target_dir = "uploaded/";
     $target_file_pic = $target_dir . $name;
     $picFileType = strtolower(pathinfo($target_file_pic, PATHINFO_EXTENSION));
     // Valid file extensions
     $extensions_arr_pic=array("JPEG","jpg","PNG");
     // Check extension
     if (in_array($picFileType, $extensions_arr_pic)) {
        header("location:welcome.php? myuser=$user_name");
        // Upload
        if (move_uploaded_file($_FILES['name']['tmp_name'], $target_file_pic)) {
            $query = "INSERT INTO user_details(image_name,image_location,first_name, last_name,
            user_name, phone_number, password, user_balance,activation_code)
            VALUES('" . $name . "','" . $target_file_vid . "','" . $first_name . "','" . $last_name . "'
            ,'" . $user_name . "','" . $phone_number . "','" . $password . "',0,'" . $activation_code . "')"
            mysqli_query($db, $query);
            echo "Upload successfully.";
            header("location:welcome.php? myuser=$user_name");

    }
    else {
           echo "Invalid file extension.";
    }
```

Figure 9: Sign Up Function

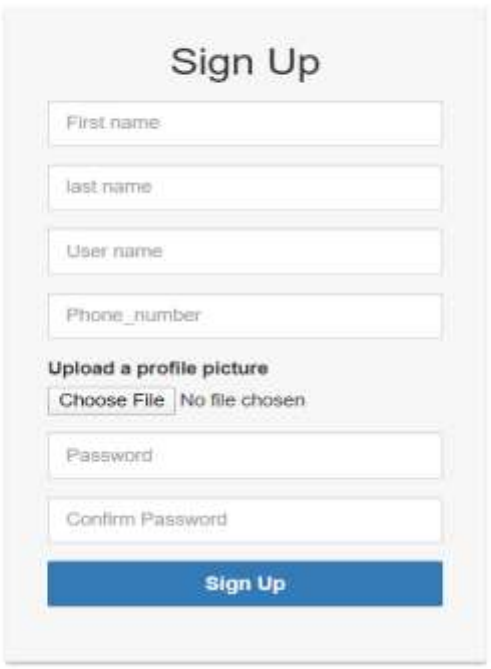


Figure 10: Sign Up Interface

4.4.3 Login

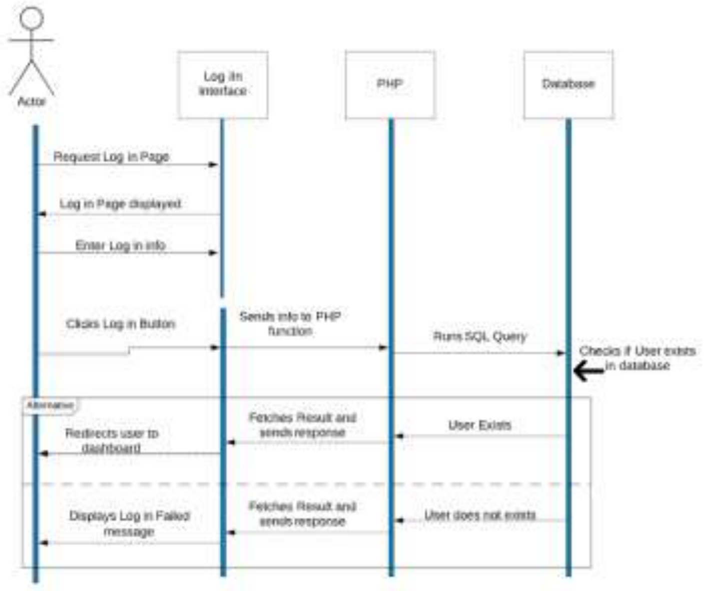
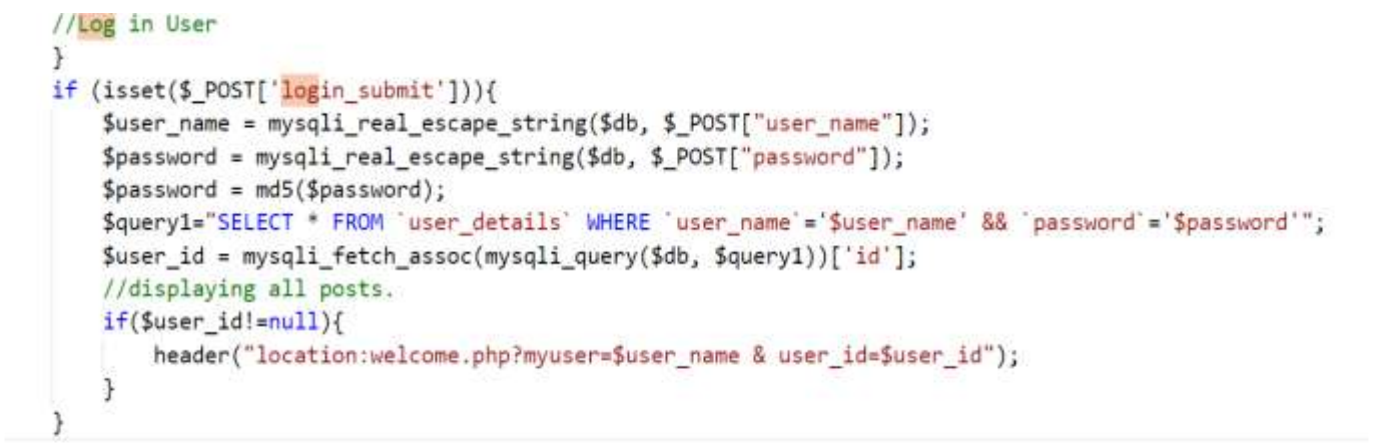

Actor
Log in interface
PHP
Database
Request Log in Page
Log in Page displayed
Enter Log in info
Clicks Log in Button
Sends info to PHP function
Runs SQL Query
Checks if User exists in database
Alternative
Redirects user to dashboard
Fetches Result and sends response
User Exists
Displays Log in Failed message
Fetches Result and sends response
User does not exists

Figure 11: Log in Sequence Diagram



```
//Log in User
}
if (isset($_POST['login_submit'])){
    $user_name = mysqli_real_escape_string($db, $_POST["user_name"]);
    $password = mysqli_real_escape_string($db, $_POST["password"]);
    $password = md5($password);
    $query1="SELECT * FROM `user_details` WHERE `user_name`='$user_name' && `password`='$password'";
    $user_id = mysqli_fetch_assoc(mysqli_query($db, $query1))['id'];
    //displaying all posts.
    if($user_id!=null){
        header("location:welcome.php?myuser=$user_name & user_id=$user_id");
    }
}
```

Figure 12: Query for Log In

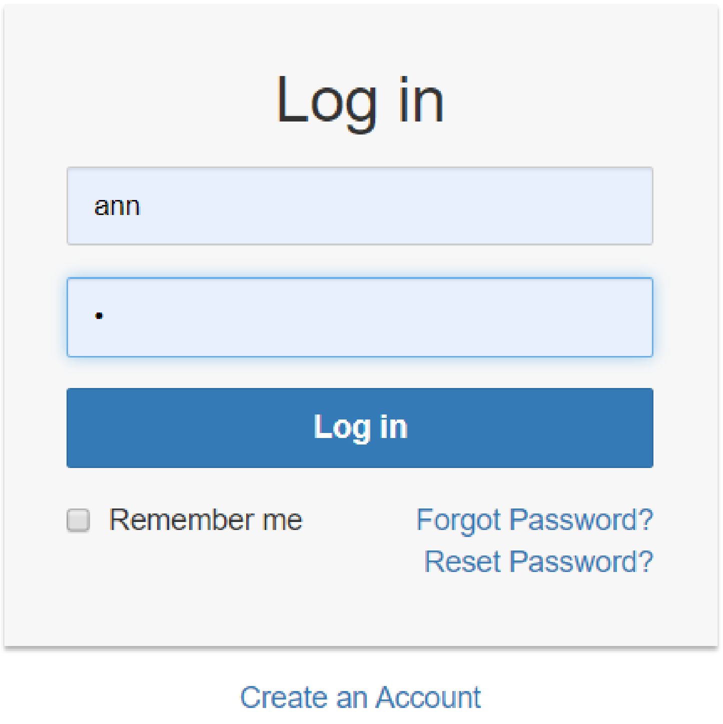


Figure 13: Log in Interface

4.4.4 Dashboard

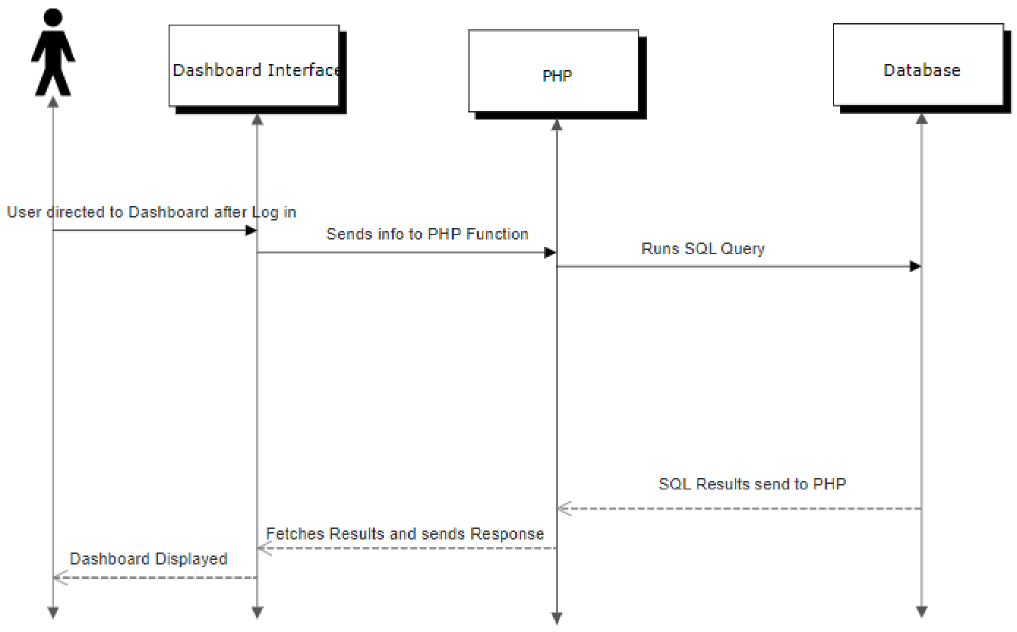

Dashboard Interface
PHP
Database
User directed to Dashboard after Log in
Sends info to PHP Function
Runs SQL Query
SQL Results send to PHP
Fetches Results and sends Response
Dashboard Displayed

Figure 14: Dashboard Sequence Diagram

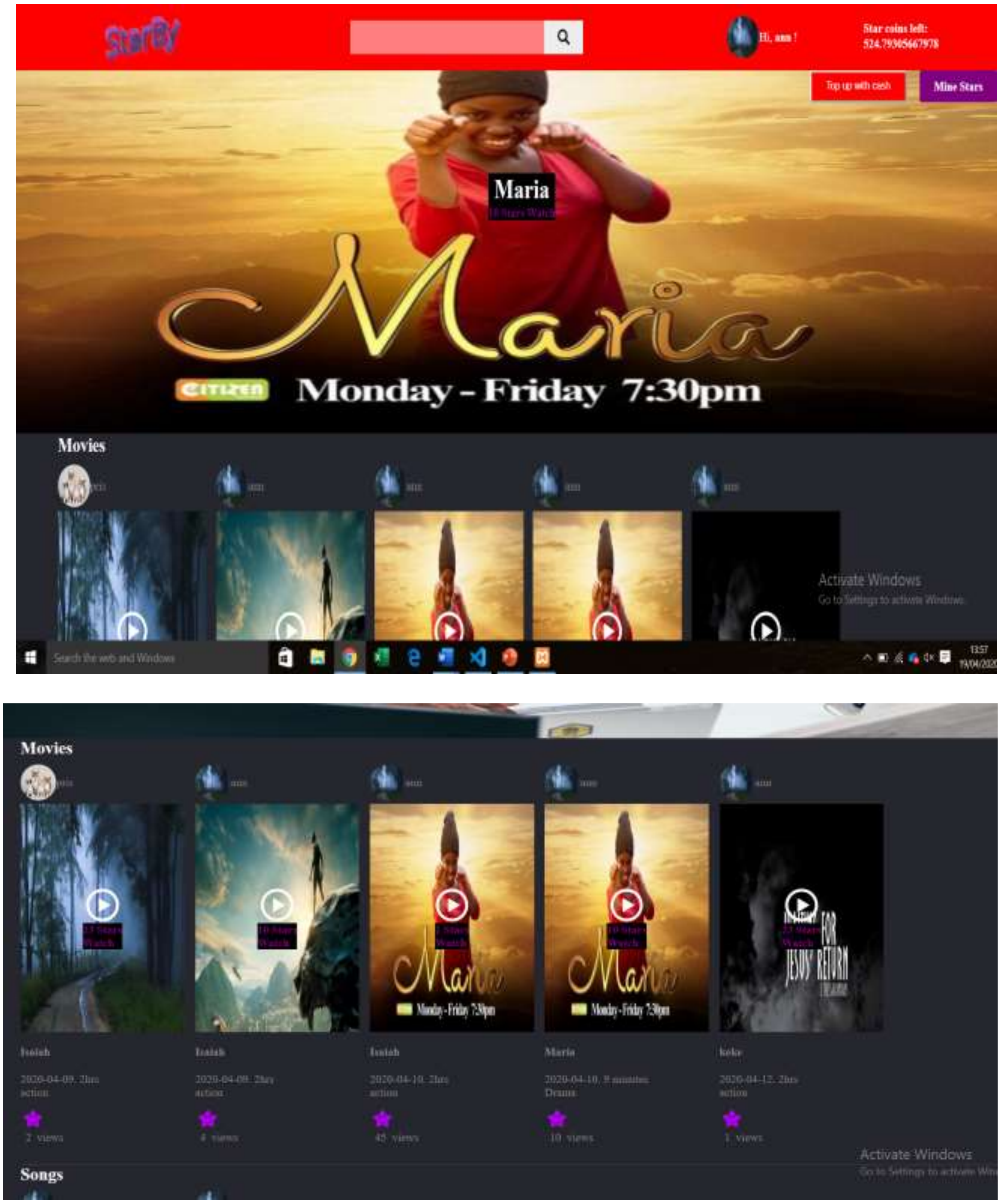


Figure 15: Dashboard Interface

4.4.5 Watching ads to earn Tokens (After clicking the 'Mine Stars' button towards the dashboard's left)

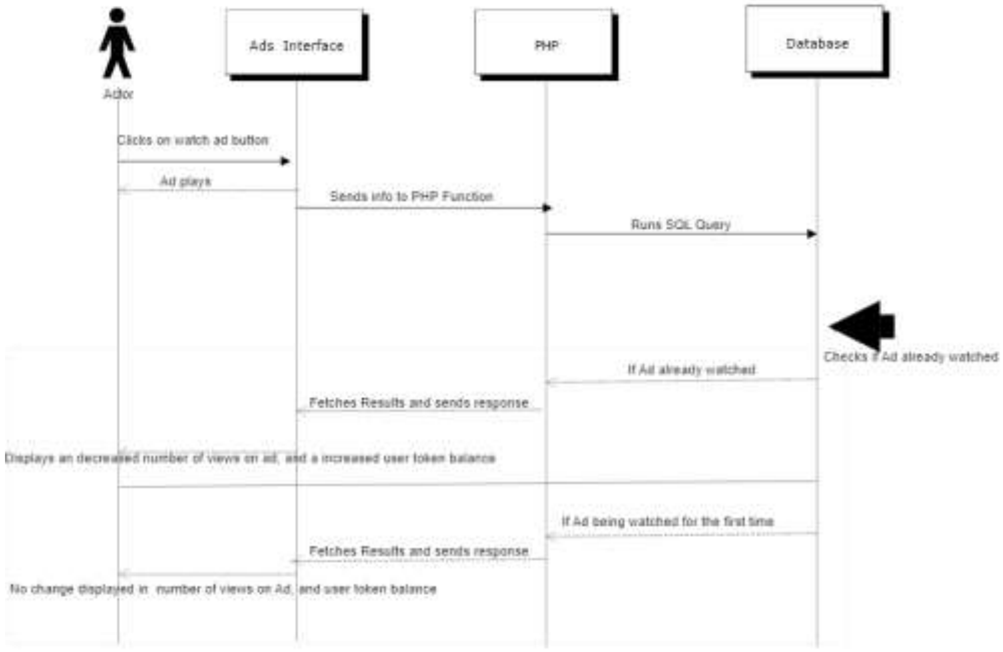

Actor
Ads Interface
PHP
Database
Clicks on watch ad button
Ad plays
Sends info to PHP Function
Runs SQL Query
Checks if Ad already watched
If Ad already watched
Fetches Results and sends response
Displays an decreased number of views on ad, and a increased user token balance
If Ad being watched for the first time
Fetches Results and sends response
No change displayed in number of views on Ad, and user token balance

Figure 16: Ad Watching Sequence Diagram

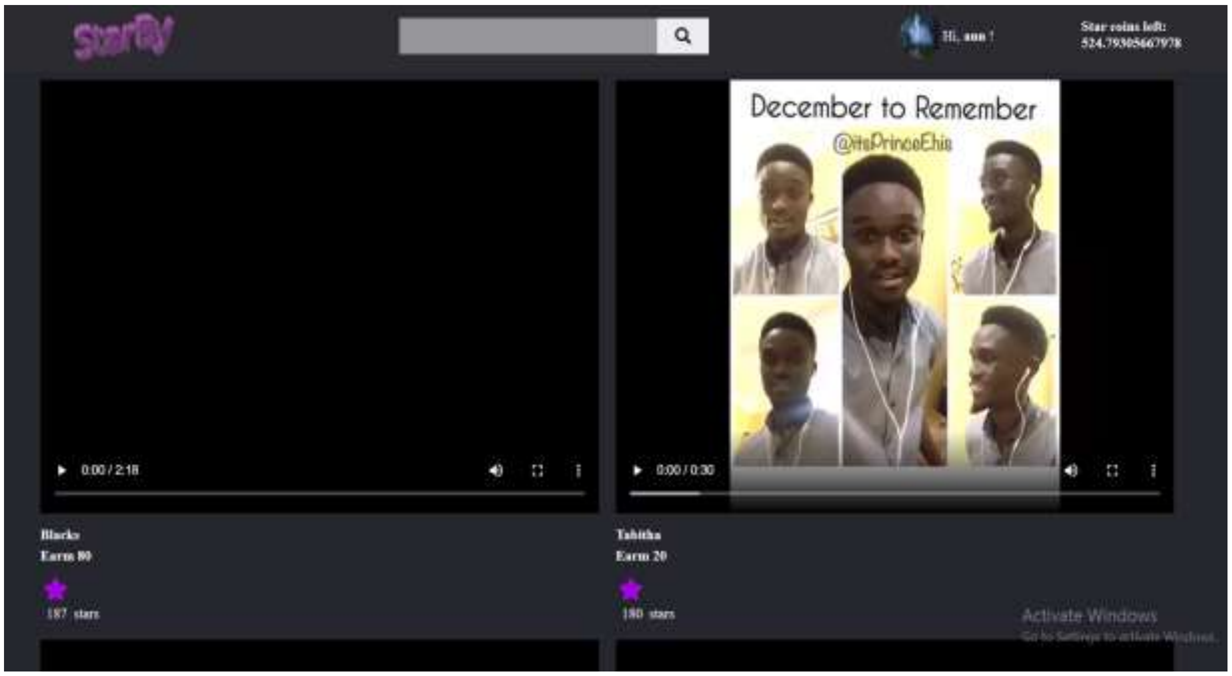
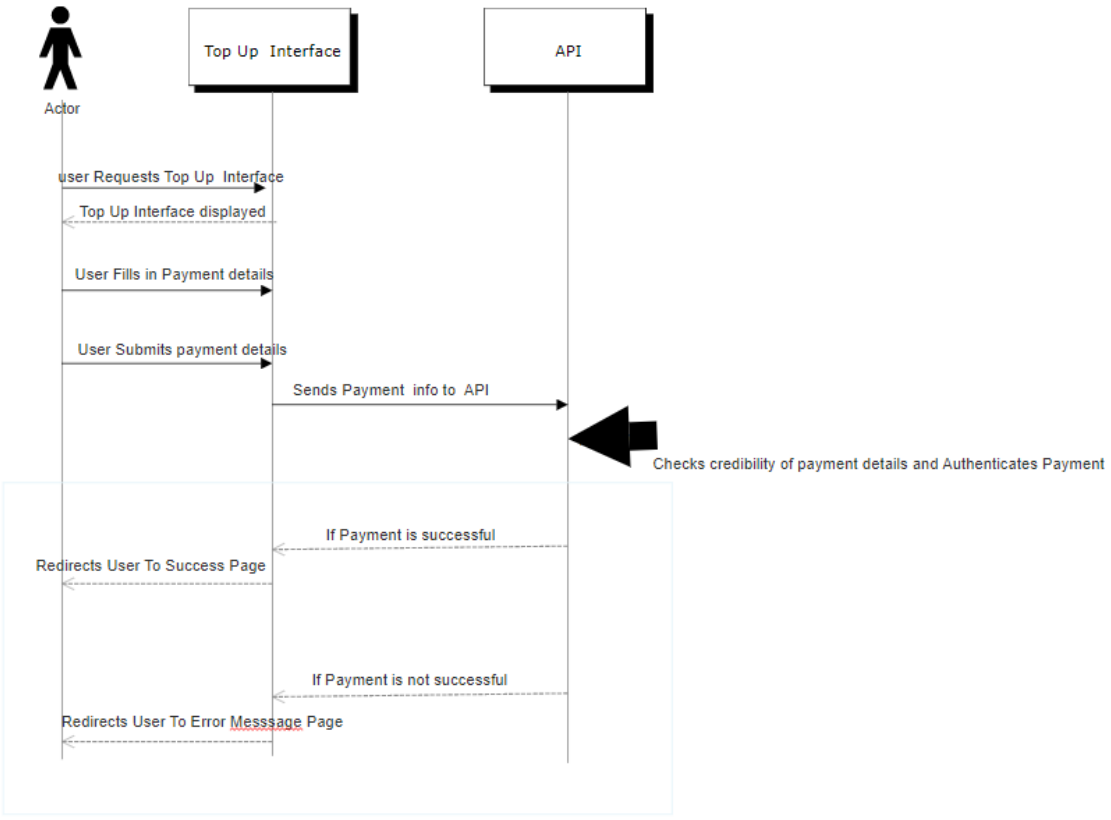


Figure 17: Ad Watching Interface

4.4.6 Topping up Token balance with Cash



Actor
Top Up Interface
API
user Requests Top Up Interface
Top Up Interface displayed
User Fills in Payment details
User Submits payment details
Sends Payment info to API
Checks credibility of payment details and Authenticates Payment
If Payment is successful
Redirects User To Success Page
If Payment is not successful
Redirects User To Error Messsage Page

Figure 18: Top Up Sequence Diagram

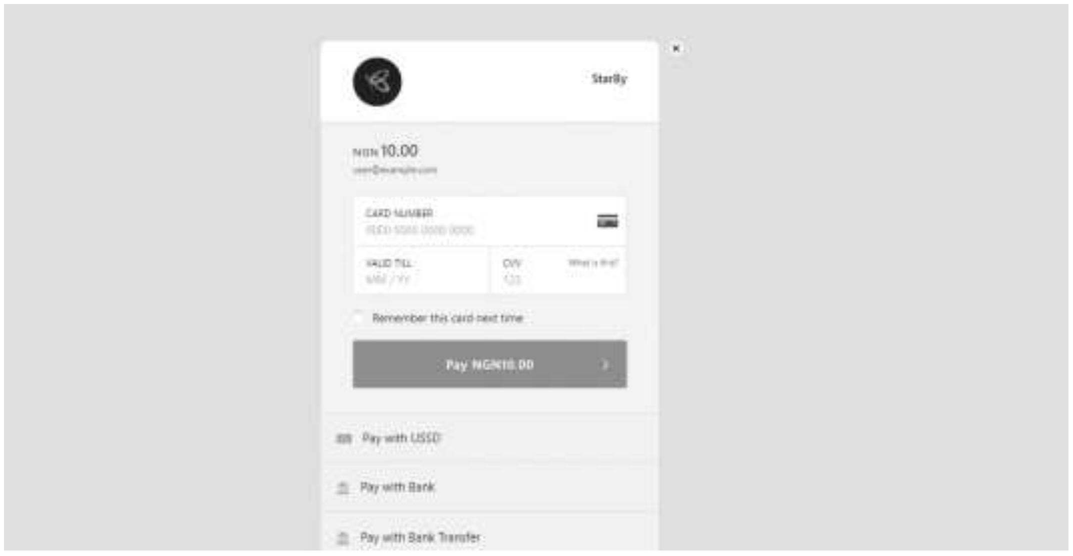


Figure 19: Top Up Interface

4.4.7 Watching a movie

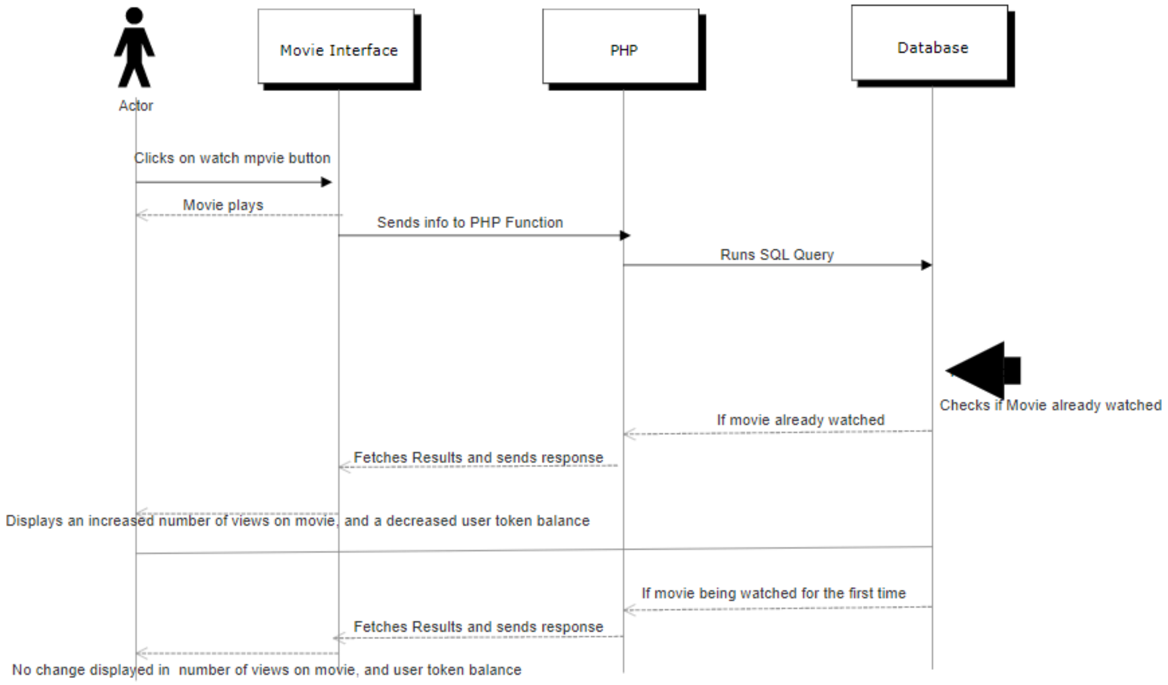

Actor
Movie Interface
PHP
Database
Clicks on watch mpvie button
Movie plays
Sends info to PHP Function
Runs SQL Query
Checks if Movie already watched
If movie already watched
Fetches Results and sends response
Displays an increased number of views on movie, and a decreased user token balance
If movie being watched for the first time
Fetches Results and sends response
No change displayed in number of views on movie, and user token balance

Figure 20: Movie Watching Sequence Diagram

Figure 21: Movie Watching Interface

4.4.8 Uploading Content

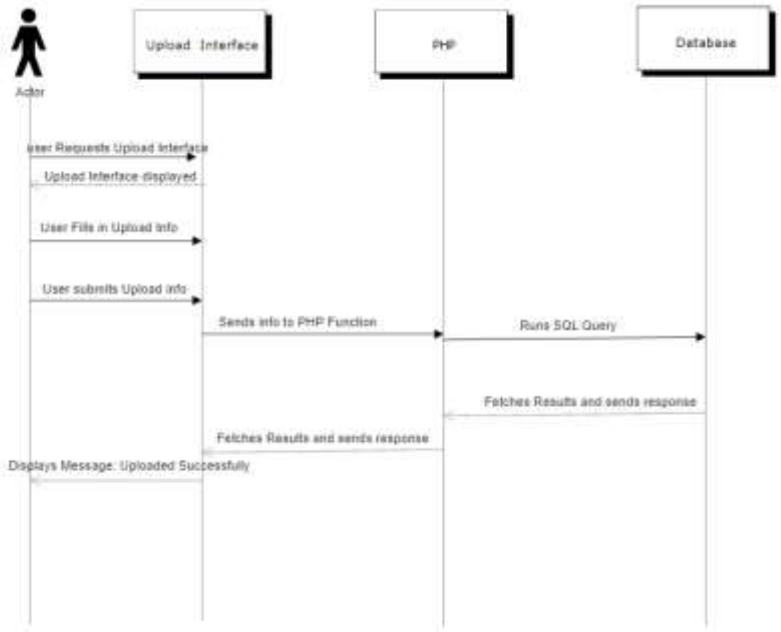
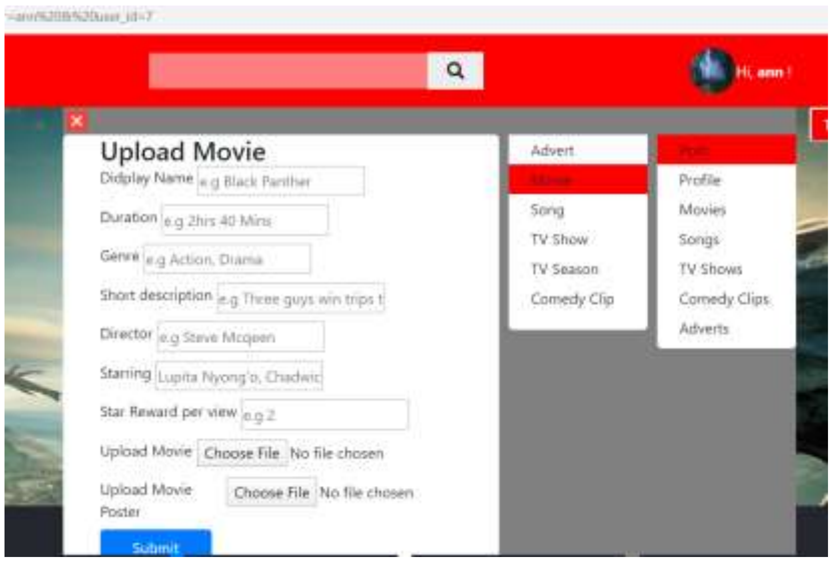

Actor
Upload Interface
PHP
Database
user Requests Upload Interface
Upload Interface displayed
User Fills in Upload Info
User submits Upload info
Sends info to PHP Function
Runs SQL Query
Fetches Results and sends response
Fetches Results and sends response
Displays Message: Uploaded Successfully

Figure 22: Upload Content Sequence Diagram



Upload Movie

Didplay Name	e.g Black Panther
Duration	e.g 2hrs 40 Mins
Genre	e.g Action, Drama
Short description	e.g Three guys win trips t
Director	e.g Steve Mcqeen
Starring	Lupita Nyong'o, Chadwic
Star Reward per view	e.g 2
Upload Movie	No file chosen
Upload Movie Poster	No file chosen

Figure 23: Upload Interface

4.4.9 Search results when searching (E.g Example below displays a search for a given movie)

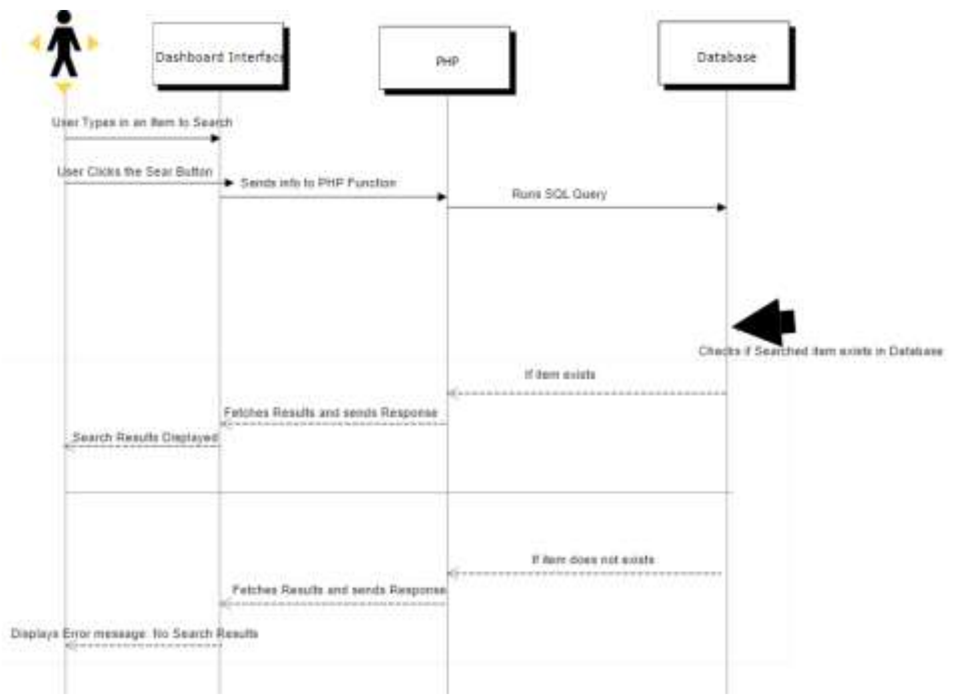

Dashboard Interface
PHP
Database
User Types in an Item to Search
User Clicks the Sear Button
Sends info to PHP Function
Runs SQL Query
Checks if Searched item exists in Database
If item exists
Fetches Results and sends Response
Search Results Displayed
If item does not exists
Fetches Results and sends Response
Displays Error message: No Search Results

Figure 24: Search Sequence Diagram

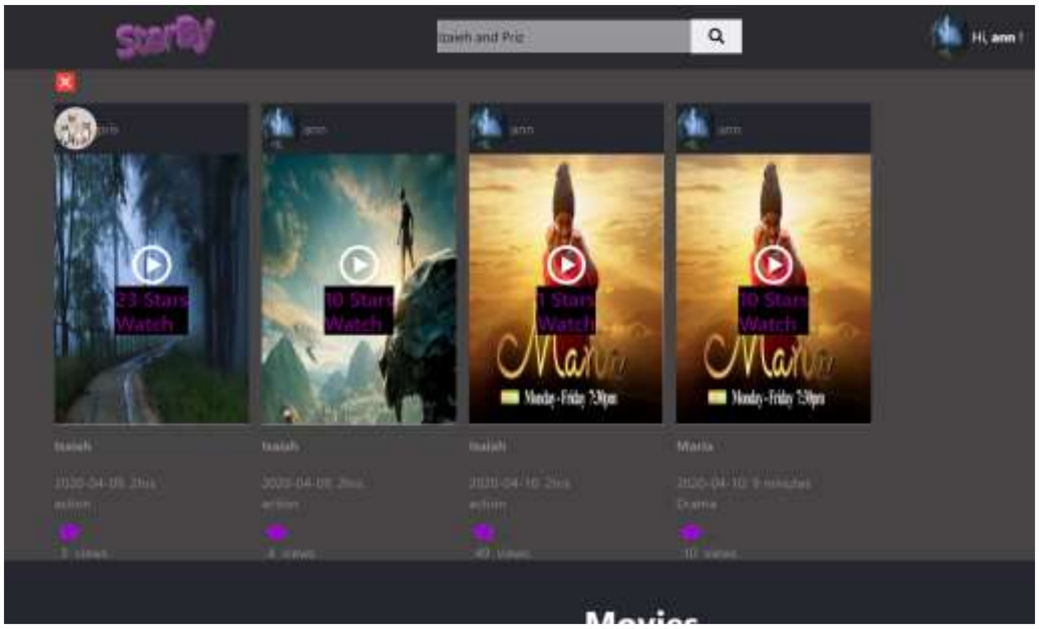


Figure 25: Search Interface

Chapter 5: Testing

5.1 Overview

This chapter delves deep into the development tests and user tests that were carried out to ensure this application functions as expected.

5.2 Developmental Testing

Development testing refers to tests carried out by the system developer to ensure that the system functions properly [16].

5.2.1 Unit Testing

Unit testing refers to a level of software testing where specific units of the system are tested [17]. In this case, different function units of the system were each tested, and they all functioned as expected. Such functionalities included login, log out, uploading a song, a movie, a comedy clip, a TV Show, different TV Seasons, and an advert, viewing any of the uploaded content by a logged in user, topping up token balance with cash, and searching for any of the uploaded content on the search bar.

5.2.2 System Testing

System testing is a level of software testing where the system is tested as a whole so to check if it functions as expected in terms of compatibility of its different components and a seamless execution of functions [17]. To perform this test, the application was hosted locally on localhost, and several devices all on the same network were able to access it simultaneously.

5.2.3 Compatibility Testing

Compatibility testing checks whether the application runs as expected on different environments, such as on different browsers, computers, etcetera [17]. This application was tested on Microsoft Edge, Opera, Chrome, and Mozilla, and on different computers (Mobile phones, PC, and a desktop computer), and all functionalities worked as expected.

5.3 User Testing

User testing checks to ensure that the system meets all user expectations, the users in this case being content creators, ad owners and users seeking entertainment [18].

5.3.1 Content creator Component Testing

A content creator created an account and was able to log in. They were able to upload their content (in this case, a song) stating the number of tokens to pay for the song per view, and they were able to log out.

5.3.1 Ad Owner Component Testing

An Ad Owner was able to create an account and log in, they were able to purchase tokens, upload an advert and were able to assign the number of tokens to earn per each full view on the advert.

5.3.1 A User seeking Entertainment Component Testing

A user seeking to be entertained was able to create an account and log in. Having a zero token balance in their account, they watched adverts and earned tokens, and were able to stream a song.

Chapter 6: Conclusion

6.1 Overview

This chapter delves deep into discussing the challenges, future works and the conclusion.

6.2 Challenges

The challenge initially faced was getting people to fill online questionnaires, and this resulted into live interviews during the requirements gathering stage. The other challenge was learning new technologies such as Ionic framework, and Block-Chain (To be used for handling payments), that did not get to be implemented due to time constraint.

The other challenge encountered in developing this project was having to trade off the fact that incentive driven viewing of ads may also lead to low-quality traffic as users may only view the ad for the token rewards without necessarily having a keen focus on the ad content [6].

6.3 Future Work

The only content available within the site is music, comedy clips, TV Shows, and movies. In future, the site could have a more diverse content such as text materials, audio books, podcasts, etcetera.

The current system is a web application, and for a better user experience, it would be highly recommendable to have native application versions of the system.

The system is currently hosted on a localhost, it would in future be hosted on a live server for users to be able to access it.

Moreover, all system files put on the web server will be encrypted to ensure no unauthorized users gain access, and a backup database would be created to track any changes made to the main database.

Currently, when a user views an Ad, the total number of tokens left attached to that Ad reduces, and this is not reflective to other logged in users unless they reload their pages. It would be of great benefit to make the system real-time in future.

6.4 Conclusion

This web application system makes life easier for content creators when it comes to tracking views of their content and the corresponding earnings they make. It does make life better for ad owners too, because their ads can receive more clicks as they no longer come up as disruptions, but as sources of token rewards. Not forgetting people looking for entertainment, they can access up to date content with or without cash affordability, and whenever they have to view ads, they do so at their own will, but are not randomly interrupted by ads popping up anyhow as they browse.

A significant portion of the intended solution in attempting this project has been achieved successfully, however, this system would have been better if it had an inbuilt recommender system where Ads are displayed to users based on the content that they have often been viewing within the site, and also one that can suggest possible likable content to users based on their browsing history.

References

[1] Tarnoff, Ben. ‘How the Internet Was Invented’. The Guardian, 15 July 2016, sec. Technology. https://www.theguardian.com/technology/2016/jul/15/how-the-internet-was-invented-1976-arpa-kahn-cerf.

[2] SearchUnifiedCommunications. ‘What Is Streaming Video? - Definition from WhatIs.Com’. Accessed20pril2020. https://searchunifiedcommunications.techtarget.com/definition/streaming-video.

[3] Basic Attention Token. ‘Home’. Accessed 16 December 2019. https://basicattentiontoken.org

[4] Sawers, Paul. ‘Spotify Launches Video Ads for Free Users’. The Next Web, 8 September 2014. https://thenextweb.com/insider/2014/09/08/spotify-will-now-serve-free-users-video-ads-desktop-mobile/.

[5] ‘30 Minutes Ad Free Is a Lie’, 28 February 2019. https://community.spotify.com/t5/iOS-iPhone-iPad/30-minutes-ad-free-is-a-lie/m-p/4692650#M109832.

[6] ‘Rewarded Video Ads: A Complete Overview with Benefits & Ad Specs’, 19 September 2018. https://instapage.com/blog/rewarded-video-ads.

[7] GeeksforGeeks. ‘MVC Design Pattern’, 18 August 2017. https://www.geeksforgeeks.org/mvc-design-pattern/.

[8] ‘HTML Tutorial - Tutorialspoint’. Accessed 11 May 2020. https://www.tutorialspoint.com/html/index.htm.

[9] ‘CSS Tutorial - Tutorialspoint’. Accessed 11 May 2020. https://www.tutorialspoint.com/css/index.htm.

[10] ‘JQuery Tutorial - Tutorialspoint’. Accessed 11 May 2020. https://www.tutorialspoint.com/jquery/index.htm.

[11] ‘Javascript Tutorial - Tutorialspoint’. Accessed 11 May 2020. https://www.tutorialspoint.com/javascript/index.htm.

[12] ‘PHP Tutorial - Tutorialspoint’. Accessed 11 May 2020. https://www.tutorialspoint.com/php/index.htm.

[13] ‘What Is AJAX? - Tutorialspoint’. Accessed 11 May 2020. https://www.tutorialspoint.com/ajax/what_is_ajax.htm.

[14] ‘MySQL Tutorial - Tutorialspoint’. Accessed 11 May 2020. https://www.tutorialspoint.com/mysql/index.htm.

[15] Flutterwave. ‘Welcome’. Accessed 11 May 2020. https://developer.flutterwave.com/docs.

[16] ‘What Is Development Testing (DevTest)? | NetApp’. Accessed 11 May 2020. https://www.netapp.com/us/info/what-is-development-testing-dev-test.aspx.

[17] ‘Software Testing Dictionary - Tutorialspoint’. Accessed 11 May 2020. https://www.tutorialspoint.com/software_testing_dictionary/.

[18] inoutput | Software Developer | Web & Mobile Application Development | Melbourne. ‘The Importance of User Testing during the Software Design and Development Process’. Accessed 11 May 2020. http://inoutput.io/articles/development/the-importance-of-user-testing-during-the-software-design-and-development-process.