

VISVESVARAYA TECHNOLOGICAL UNIVERSITY

**Jnana Sangama, Santhibastawad Road, Machhe
Belagavi - 590018, Karnataka, India**



**A PROJECT REPORT
ON
“MEDIA & ENTERTAINMENT
MANAGEMENT SYSTEM”**

Submitted in the partial fulfilment of the requirements for the award of the degree of

**BACHELOR OF ENGINEERING
IN
INFORMATION SCIENCE AND ENGINEERING**

For the Academic Year 2019-2020

Submitted by

**Akanksha
Pallavi Pandey**

**1JS18IS125
1JS17IS062**

Under the Guidance of

**Mrs. Punitha,
Dr. Sowmya K.N,**

Asst. Prof, Dept. of ISE, JSSATE



**DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING
JSS ACADEMY OF TECHNICAL EDUCATION**

JSS Campus, Dr.Vishnuvardhan Road, Bengaluru-560060
2020-2021

JSS MAHAVIDYAPEETHA, MYSURU
JSS ACADEMY OF TECHNICAL EDUCATION
JSS Campus, Dr.Vishnuvardhan Road, Bengaluru-560060

DEPARTMENT OF INFORMATION SCIENCE & ENGINEERING



CERTIFICATE

This is to certify that the project work entitled “**Media & Entertainment Management System**” is a bona fide work carried out by **Akanksha [1JS18IS125], Pallavi Pandey [1JS18IS062]** in partial fulfilment for the DBMS Laboratory with Mini Project (18CSL58) of 5th Semester **Bachelor of Engineering in Information Science and Engineering of Visvesvaraya Technological University Belagavi** during the year 2020-2021. It is certified that all corrections and suggestions have been incorporated in the report deposited in the department library.

Signature of the Guide	Signature of the Guide	Signature of the HoD
Dr. Sowmya K N Asst. Prof. Dept. Of ISE JSSATE, Bengaluru	Mrs. Punitha M Asst. Prof. Dept. Of ISE JSSATE, Bengaluru	Dr. Dayananda P Assoc. Prof & Head Dept. of ISE JSSATE, Bengaluru

Signature of Examiners

1. _____

2. _____

ACKNOWLEDGEMENT

The satisfaction and euphoria that accompany the successful completion of any task would be incomplete without the mention of the people who made it possible. So with gratitude, we acknowledge all those whose guidance and encouragement crowned my effort with success.

First and foremost, we would like to thank his **Holiness Jagadguru Sri Shivarathri Deshikendra Mahaswamiji** and **Dr. Mrityunjaya V Latte**, Principal, JSSATE, Bangalore for providing an opportunity to carry out the Project Work as a part of our curriculum in the partial fulfilment of the degree course.

We express our sincere gratitude for our beloved Head of the department, **Dr. Dayananda P**, for his co-operation and encouragement at all the moments of our approach.

It is our pleasant duty to place on record our deepest sense of gratitude to our respected guide **Punitha M, Sowmya K.N, Assistant Professor** for the constant encouragement, valuable help and assistance in every possible way.

We are thankful to the Project Coordinators **Dr. Sowmya K N**, Asst. Professor and **Mrs. Punitha M**, Asst. Professor, for their continuous co-operation and support.

We would like to thank all ISE Department Teachers, non-teaching staff and Library staff for providing us with their valuable guidance and for being there at all stages of our work.

Akanksha[1JS18IS125]
Pallavi Pandey[1JS18IS062]

Abstract

The purpose of Media & Entertainment Management system is to automate the existing manual system by the help of computerized equipment and full-fledged computer software. It provides streaming movies and shows to subscriber. In order to understand customer behavior, it needs to track its user, its content, and the content specific customers watch. Understanding which user watch which shows and movies will recommend similar content that user will also likely enjoy. Moreover, the database will track important metrics such as customer churn and poor performing content (content that receives poor ratings and content that is rarely streamed).

Thus, by this all it proves it is user- friendly. Media & Entertainment Management System, as described above, can lead to error free, secure, reliable and fast management system. It can assist the user to concentrate on their other activities rather to concentrate on the record keeping. Thus, it will help organization in better utilization of resources. This is designed to assist in strategic planning, and will help you ensure that your organization is equipped with the right level of information and details for your future uses. These systems will ultimately allow you to better manage resources.

Table of contents

Acknowledgement	5
Abstract	6
Table of contents	7
Chapter 1 Introduction	
1.1 Introduction to DBMS	9
1.1.1 History to DBMS	10
1.2 Project Introduction	11
1.3 Objectives	12
Chapter 2 Literature Survey	
2.1 Problem statement	12
2.2 HTML	13
2.3 CSS	13
2.4 JavaScript	13
2.5 PHP	13
2.6 Normalization	14
2.7 Requirement Specification	
2.7.1 Software Specification	15
2.7.2 Hardware Specification	15
2.8 Functional Requirements	15
Chapter 3 System Design	
3.1 Introduction	16
3.2 ER Diagram	17
3.3 Relational Schema Diagram	18

Chapter 4 Implementation

4.1 Project Implementation	19
4.2 Pseudo Code	
4.2.1 HTML Code	20
4.2.2 JavaScript Code	21
4.2.3 PHP Code	22
4.3 Source Code for Back-End	
4.3.1 Stored Procedure	26
4.3.2 Triggers	27

Chapter 5 Testing

5.1 Introduction	28
5.2 Types of testing	
5.2.1 Unit testing	29
5.2.2 Integration testing	29
5.2.3 Acceptance testing	29
5.3 Test results	29

Chapter 6 Results

6.1 Snapshots	30
---------------	----

Chapter 7 Conclusion

7.1 Conclusion	36
7.2 Future Enhancement	36

Chapter 8 References

8.1 Book references	37
8.2 Web references	38

Chapter 1

Introduction

1.1 Introduction to DBMS

A database is a collection of related data which represents some aspect of the real world. A database system is designed to be built and populated with data for a certain task.

Database Management System (DBMS) is a software for storing and retrieving users' data while considering appropriate security measures. It consists of a group of programs which manipulate the database. The DBMS accepts the request for data from an application and instructs the operating system to provide the specific data. In large systems, a DBMS helps users and other third-party software to store and retrieve data. DBMS allows users to create their own databases as per their requirement. The term “DBMS” includes the user of the database and other application programs. It provides an interface between the data and the software application.

The common features of a DBMS include:

- capacity for large amount of data
- an easy to use interface language (SQL-structured query language)
- efficient retrieval mechanisms
- multi-user support
- security management
- concurrency and transaction control
- persistent storage with backup and recovery for reliability

The users of a database assume difference roles such as

- end user - application programmers that use the DB as a storage subsystem
- designer - application programmers and/or business analysts who design the layout of the DB
- administrator - operators who maintain the health and efficiency of the DB
- implementor - programmers who maintain and develop the DBMS

The key concepts of database include:

- schema - the structure and the constraints of data
- data - the actual content of the DB representing information
- data definition language - used to specify the schema
- data manipulation and query language - used to change the data and query them

Schema are meta-data that describe data. Such meta-data can describe the structure of the data which ranges from strictly enforced structure (relational) to semi-structure (XML) and free-structured data (text files). Before we define the schema we must decide on a model of the data - a metaphor. For relational database n-ary relation is used to model data.

1.1.1. History of DBMS

In the ancient times, elaborate database systems were developed by government offices, libraries, hospitals, and business organizations, and some of the basic principles of these systems are still being used today.

Computerized database started in the 1960s, when the use of computers became a more cost-effective option for private organizations. There were two popular data models in this decade: a network model called CODASYL and a hierarchical model called IMS. One database system that proved to be a commercial success was the SABRE system that was used by IBM to help American Airlines manage its reservations data.

E.F. Codd published an important paper to propose the use of a relational database model, and his ideas changed the way people thought about databases. In his model, the database's schema, or logical organization, is disconnected from physical information storage, and this became the standard principle for database systems.

Two major relational database system prototypes were created between the years 1974 and 1977, and they were the Ingres, which was developed at UBC, and System R, created at IBM San Jose. Ingres used a query language known as QUEL, on the other hand, System R used the SEQUEL query language, and it contributed to the development of SQL/DS, DB2, Allbase, Oracle, and Non-Stop SQL. It was also in this decade that Relational Database Management System, or RDBMS, became a recognized term.

A new database model called Entity-Relationship, or ER, was proposed by P. Chen in 1976. This model made it possible for designers to focus on data application, instead of logical table structure. Structured Query Language, or SQL, became the standard query language.

Relational database systems became a commercial success in the 1980s as the rapid increase in computer sales boosted the database market, and this caused a major decline in the popularity of network and hierarchical database models.

A number of tools for personal productivity, such as ODBC and Excel/Access, were also developed. Prototypes for Object Database Management Systems, or ODBMS, were created in the early 1990s.

Although the Internet industry experienced a decline in the early 2000s, database applications continue to grow. New interactive applications were developed for PDAs, point-of-sale transactions, and consolidation of vendors. Presently, the three leading database companies in the western world are Microsoft, IBM, and Oracle.

Today, databases are everywhere and are used to enhance our day-to-day life. From personal cloud storage to predicting the weather, many of the services we utilize today are possible due to databases. Presently, there are many new players in the non-relational database space offering specific solutions. Some of the current relational databases include giants such as Oracle, MySQL, and DB2. We're also seeing new trends emerging that focus on making powerful technology accessible to everyone.

1.2 Project Introduction

Data is an important resource in today's age where most, if not all applications, run on gathering and applying the data in a productive way. Storing years and years' worth of data always pays off in the long run, regardless of what type of data one is storing and what the data pertains to. There is always much knowledge to be gained by the stored data.

The "Media & Entertainment Management System" has been developed to override the problems prevailing in the practicing manual system. This webpage is supported to eliminate and, in some cases, reduce the hardships faced by this existing system. Moreover, this system is designed for the particular need of the user to carry out operations in a smooth and effective manner. The application is reduced as much as possible to avoid errors while entering the data. Thus, by this all it proves it is user- friendly.

User can search movie according to their preferences and watch the trailer of the movie. This project provides a centralized database that offers the basic information about the movies such as the genre, director name, artist name, description of movie and year of release, poster as well as trailer. User can also customize their watchlist according to their need.

1.3 Objectives

- Provide a centralized database to store information about the movie.
- Allow playback of the selected movie trailer, provided that the links to a streaming site, such as YouTube are provided
- Additional features such as allowing the searching movies according to our preferences.
- Allow logging in of the user and sort their choice.
- Allow viewing trailers of all movies.
- Allow new users to create their own access account by entering their email and their desired password
- Store the user details in the database as well
- Validate the entered data and error check everything
- Allow user to customize their watchlist

Chapter 2

Literature Survey

2.1 Problem Statement

There has been an absence of a preference-based movie system till date and the application deals with having a system that can provide the necessary details about a particular movie, from a list entered by admin, obtained from database.

2.2 HTML (HyperText Markup Language)

Hypertext Markup Language (HTML) is the standard markup language for documents designed to be displayed in a web browser. Web browsers receive HTML documents from a web server or from local storage and render the documents into multimedia web pages. HTML describes the structure of a web page semantically and originally included cues for the appearance of the document.

HTML elements are the building blocks of HTML pages. With HTML constructs, images and other objects such as interactive forms may be embedded into the rendered page. HTML provides a means to create structured documents by denoting structural semantics for text such as headings, paragraphs, lists, links, quotes and other items. HTML elements are delineated by tags, written using angle brackets. Tags such as `<p>`, `<input>`, etc. Browsers do not display the HTML tags, but use them to interpret the content of the page.

HTML can embed programs written in a scripting language such as JavaScript, which affects the behavior and content of web pages. Inclusion of CSS defines the look and layout of content. The World Wide Web Consortium (W3C), former maintainer of the HTML and current maintainer of the CSS standards, has encouraged the use of CSS over explicit presentational HTML since 1997.

2.3 CSS (Cascading Style Sheets)

Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation of a document written in a markup language such as HTML. CSS is a cornerstone technology of the World Wide Web, alongside HTML and JavaScript.

CSS is designed to enable the separation of presentation and content, including layout, colors, and fonts. This separation can improve content accessibility, provide more flexibility and control in the specification of presentation characteristics, enable multiple web pages to share formatting by specifying the relevant CSS in a separate .css file which reduces complexity and repetition in the structural content. Separation of formatting and content also makes it feasible to present the same markup page in different styles for different rendering methods, such as on-screen on a computer monitor but also for alternate formatting if the content is accessed on a mobile device.

The name cascading comes from the specified priority scheme to determine which style rule applies if more than one rule matches a particular element. This cascading priority scheme is predictable.

The CSS specifications are maintained by the World Wide Web Consortium (W3C). The W3C operates a free CSS validation service for CSS documents

2.4 JavaScript (JS)

JavaScript often abbreviated as JS, is a programming language that conforms to the ECMAScript specification. JavaScript is high-level, often just-in-time compiled, and multi-paradigm. It has curly-bracket syntax, dynamic typing, prototype-based object-orientation, and first-class functions.

Alongside HTML and CSS, JavaScript is one of the core technologies of the World Wide Web. JavaScript enables interactive web pages and is an essential part of web applications. The vast

majority of websites use it for client-side page behavior, and all major web browsers have a dedicated JavaScript engine to execute it.

As a multi-paradigm language, JavaScript supports event-driven, functional, and imperative programming styles. It has application programming interfaces (APIs) for working with text, dates, regular expressions, standard data structures, and the Document Object Model (DOM). However, the language itself does not include any input/output (I/O), such as networking, storage, or graphics facilities, as the host environment (usually a web browser) provides those APIs.

JavaScript engines were originally used only in web browsers, but they are now embedded in some servers, usually via Node.js. They are also embedded in a variety of applications created with frameworks such as Electron and Cordova.

2.5 PHP (Hypertext PreProcessor)

PHP is a widely-used open-source general-purpose scripting language that is especially suited for web development and can be embedded into HTML. PHP code is usually processed on a web server by a PHP interpreter implemented as a module, a daemon or as a Common Gateway Interface (CGI) executable. On a web server, the result of the interpreted and executed PHP code – which may be any type of data, such as generated HTML or binary image data – would form the whole or part of an HTTP response. Various web template systems, web content management systems, and web frameworks exist which can be employed to orchestrate or facilitate the generation of that response. Additionally, PHP can be used for many programming tasks outside of the web context, such as standalone graphical applications and robotic drone control. Arbitrary PHP code can also be interpreted and executed via command-line interface (CLI).

The standard PHP interpreter, powered by the Zend Engine, is free software released under the PHP License. PHP has been widely ported and can be deployed on most web servers on almost every operating system and platform, free of charge.

The PHP language evolved without a written formal specification or standard until 2014, with the original implementation acting as the de facto standard which other implementations aimed to follow. Since 2014, work has gone on to create a formal PHP specification

What distinguishes PHP from something like client-side JavaScript is that the code is executed on the server, generating HTML which is then sent to the client. The client would receive the results of running that script, but would not know what the underlying code was. You can even configure your web server to process all your HTML files with PHP, and then there's really no way that users can tell what is going on behind scene.

2.6 Normalization

Database normalization is the process of structuring a relational database in accordance with a series of so-called normal forms in order to reduce data redundancy and improve data integrity. Normalization entails organizing the columns (attributes) and tables (relations) of a database to ensure that their dependencies are properly enforced by database integrity constraints. It is accomplished by applying some formal rules either by a process of synthesis (creating a new database design) or decomposition (improving an existing database design).

The Theory of Data Normalization in SQL server is still being developed further. For example, there are discussions even on 6th Normal Form. However, in most practical applications, normalization achieves its best in 3rd Normal Form.

One of the driving forces behind database normalization is to streamline data by reducing redundant data. Redundancy of data means there are multiple copies of the same information spread over multiple locations in the same database. The drawbacks of data redundancy include:

1. Data maintenance becomes tedious – data deletion and data updates become problematic
2. It creates data inconsistencies
3. Insert, Update and Delete anomalies become frequent. An update anomaly, for example, means that the versions of the same record, duplicated in different places in the database, will all need to be updated to keep the record consistent
4. Redundant data inflates the size of a database and takes up an inordinate amount of space on disk

First Normal Form:

For a table to be in the First Normal Form, it should follow the following 4 rules:

- It should only have single(atomic) valued attributes/columns.
- Values stored in a column should be of the same domain
- All the columns in a table should have unique names.
- And the order in which data is stored, does not matter.

Second Normal Form:

For a table to be in the Second Normal Form,

- It should be in the First Normal form.
- And, it should not have Partial Dependency, as in if a table B has two primary keys 1 and 2, (ie, a candidate key), both of those must be foreign keys in table A. If only one of those is a foreign key, then partial dependency exists

Third Normal Form:

A table is said to be in the Third Normal Form when

- It is in the Second Normal form.
- And, it doesn't have Transitive Dependency, as in if a table A is dependent on table B and then table B is dependent on table C, an explicit relation from table A to C should not exist.

2.7 Requirement specification

2.7.1 Software Requirements

- HTML (HyperText Markup Language)

- CSS (Cascading Style Sheets)
- JavaScript
- PHP for server connectivity
- Text Editor such as Visual Studio Code, Atom, Sublime, etc
- MySQL server such as Apache, Nginx

The PHP, database GUI and the server required can be utilized in the form of a development stack such as MAMP (Mac, Apache, MySQL, PHP) or XAMPP (Cross Platform, Apache, MySQL/MariaDB, Perl, PHP)

2.7.2 Hardware Requirements

- Device with Intel i3, 7th gen processor, an equivalent AMD processor or higher
- At least 2 GB of RAM
- 2 GB or higher of long term storage space on a hard drive or solid state drive

2.8 Functional Requirements

- It should present the list of movies in the database to any visitor to view and choose according to their preference but not alter the contents
- It should allow users to log in
- It should allow a visitor to create a new authorization account
- It should allow user to search existing movie
- It should present the genre, director, artist, ratings-based searching for user
- It should allow play the trailer of movie
- It should validate and make sure that the data entered is of proper format

Chapter 3

System Design

3.1 Introduction

This project team decided to implement the core functionality first and later to attach additional functions. The core functionality is:

- Add and utilize user information to log in
- Add information about the movie, artists ,genre ,director ,ratings, poster, trailer, watchlist of user.

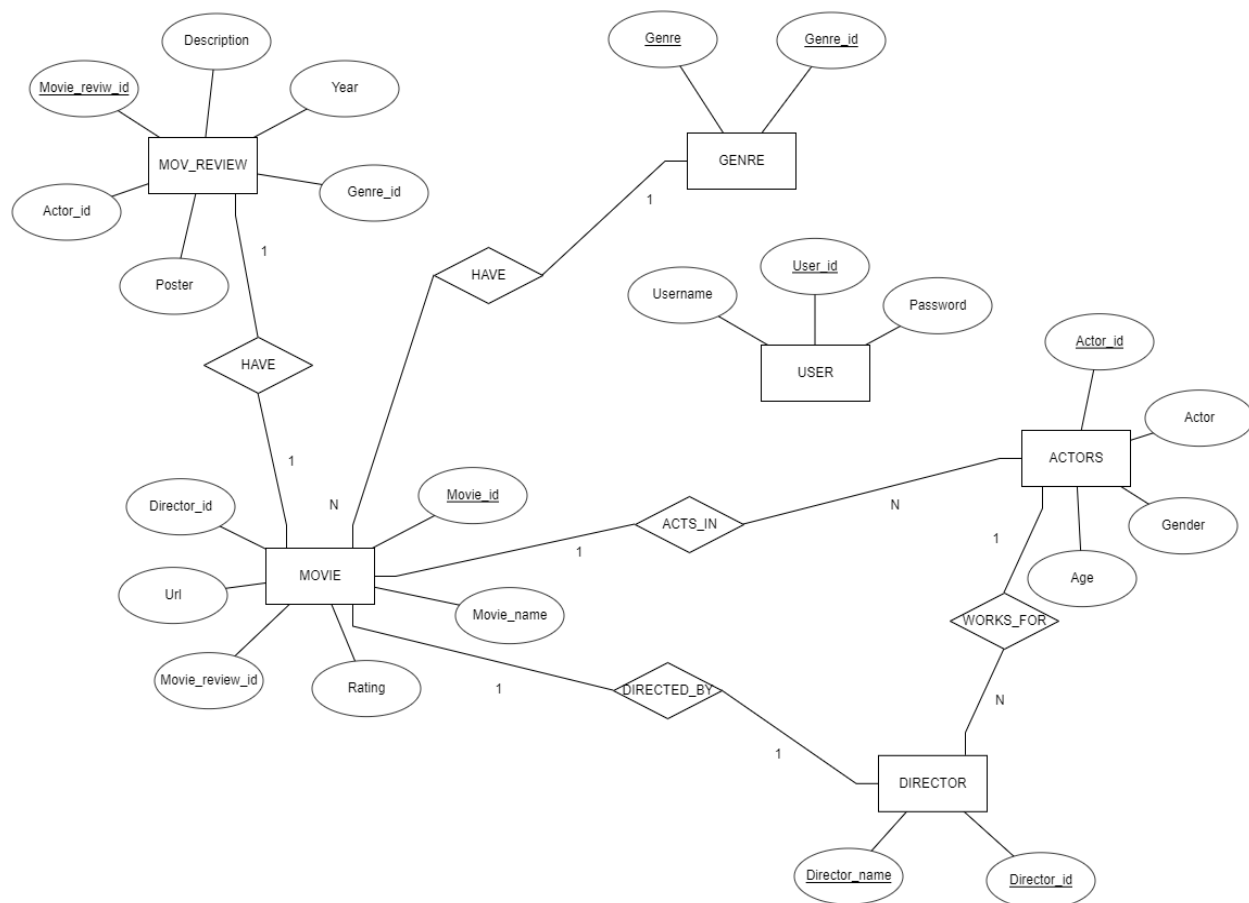
This database currently has 6 related tables and one user table to keep track of who has access to the stored information. The included tables are:

- Genre table to hold genre_id,
- Movie_review table holds information of movie poster, year of release and description.
- Movie table consist of trailer, rating of movie and director info.
- Director table consist of unique director_id.
- Actor table consist of actor_id, actor gender and age info.
- User table which holds the emails and passwords of the users who are allowed to access the database.

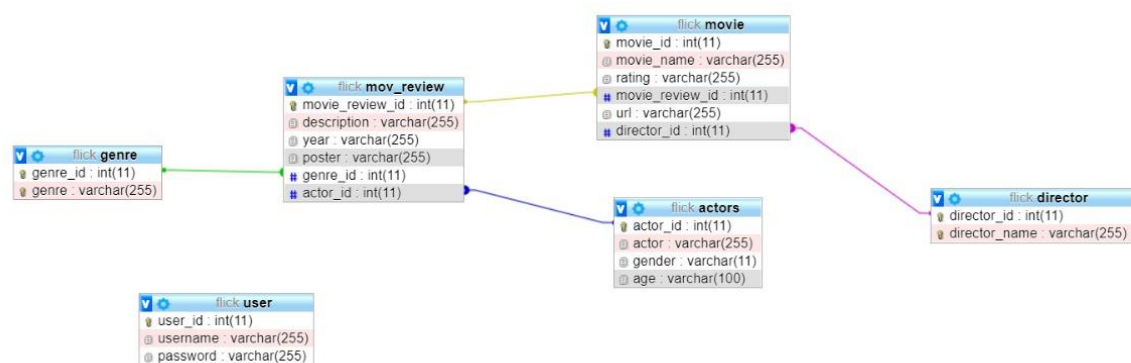
All the tables are normalized and each table has a been categorized to store a particular type of data. The genre table store genre_id assigned automatically by code, the mov_review table consist of movie_revie_id as primary key, description, year, poster, genre_id, actor_id and movie table consist of movie_id as primary key, movie_name, rating, movie_review_id, url and director_id, actors table consist of actor_id as primary key, actor, gender and age. Director table consist of director_id assigned automatically.

The User table just hold the access information of the users to make sure nothing about the database can be changed without an authorized user logging in.

3.2 ER Diagram



3.3 Relational Schema Diagram



Chapter 4

Implementation

4.1 Project Implementation

Create and use database named 'flick'

```
CREATE DATABASE IF NOT EXISTS `music`;  
USE `flick`;
```

Create the required tables, namely label, genre, user, artist, album and track tables and also define their primary and foreign keys

```
CREATE TABLE `actors` (  
  `actor_id` int(11) NOT NULL AUTO_INCREMENT,  
  `actor` varchar(255) NOT NULL,  
  `gender` varchar(11) NOT NULL,  
  `age` varchar(100) NOT NULL,  
  PRIMARY KEY (`actor_id`)  
)
```

```
CREATE TABLE `director` (  
  `director_id` int(11) NOT NULL AUTO_INCREMENT,  
  `director_name` varchar(255) NOT NULL,  
  PRIMARY KEY (`director_id`),  
  UNIQUE KEY `director_name` (`director_name`)  
)
```

```
CREATE TABLE `genre` (  
  `genre_id` int(11) NOT NULL AUTO_INCREMENT,  
  `genre` varchar(255) NOT NULL,  
  PRIMARY KEY (`genre_id`),  
  UNIQUE KEY `genre` (`genre`)  
)
```

```
CREATE TABLE `movie` (  
  `movie_id` int(11) NOT NULL AUTO_INCREMENT,  
  `movie_name` varchar(255) NOT NULL,  
  `rating` varchar(255) NOT NULL,  
  `movie_review_id` int(11) NOT NULL,  
  `url` varchar(255) NOT NULL,  
  `director_id` int(11) NOT NULL,  
  PRIMARY KEY (`movie_id`),  
  KEY `movie_ibfk_1` (`movie_review_id`),  
  KEY `movie_ibfk_2` (`director_id`)  
)
```

```
CREATE TABLE `mov_review` (  
  `movie_review_id` int(11) NOT NULL AUTO_INCREMENT,  
  `description` varchar(500) NOT NULL,  
  `year` varchar(255) NOT NULL,  
  `poster` varchar(255) NOT NULL,
```

```

`genre_id` int(11) NOT NULL,
`actor_id` int(11) NOT NULL,
PRIMARY KEY (`movie_review_id`),
KEY `genre_id` (`genre_id`),
KEY `mov_review_ibfk_2` (`actor_id`)
)

CREATE TABLE `user` (
  `user_id` int(11) NOT NULL AUTO_INCREMENT,
  `username` varchar(255) NOT NULL,
  `password` varchar(255) NOT NULL,
  PRIMARY KEY (`user_id`)
)

```

4.2 Pseudo Code

4.2.1 HTML Code

The project is based on a page-by-page website design. Hence the HTML code is bath explicitly written and generated by the PHP and JavaScript blocks. For example, the HTML code in add.php has explicitly written HTML that generates a form-based input to enter new values in the tables in the database

HTML Code of Admin.php

```

<!DOCTYPE html>
<html>
<head>
<title>new movie entry</title>

<!-- Latest compiled and minified CSS -->
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">

<!-- jQuery library -->
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>

<!-- Popper JS -->
<script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.16.0/umd/popper.min.js"></script>

<!-- Latest compiled JavaScript -->
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>

<link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.4.0/css/bootstrap.min.css" integrity="sha384-SI27wrMjH3Z789r4o+fgIJtnzk" >
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/rateYo/2.3.2/jquery.rateyo.min.css">

</head>
<body>

  <div class="container">
    <br>
    <h1 class="text-white bg-dark text-center">
      create new entry
    </h1>
    <div class="col-lg-8">
      <form action="upload.php" method="post" enctype="multipart/form-data">

        <div class="form-group">
          <label for="user"> MOVIE </label>
          <input type="text" name="movie" id="user" class="form-control">
        </div>

```

```

    <div class="rateyo" id="rating"
    data-rateyo-rating="4"
    data-rateyo-num-stars="5"
    data-rateyo-score="3">
    </div>

    <span class='result'>0</span>
    <input type="hidden" name="rating">

<script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/rateYo/2.3.2/jquery.rateyo.min.js"></script>

<script>

    $(function () {
        $(".rateyo").rateYo().on("rateyo.change", function (e, data) {
            var rating = data.rating;
            $(this).parent().find('.score').text('score :'+ $(this).attr('data-rateyo-score'));
            $(this).parent().find('.result').text('rating :'+ rating);
            $(this).parent().find('input[name=rating]').val(rating); //add rating value to input field
        });
    });

</script>

    <div class="form-group">
    <label for="user"> EMBED CODE </label>
    <input type="text" name="url" id="user" class="form-control">
    </div>

    <div class="form-group">
    <label for="user"> MOVIE DESCRIPTION </label>
    <input type="text" name="description" id="user" class="form-control">
    </div>

```

```

<div class="form-group">
<label for="user"> GENRE </label>
<input type="text" name="genre" id="user" class="form-control">
</div>

<div class="form-group">
<label for="user"> ACTOR </label>
<input type="text" name="actor" id="user" class="form-control">
</div>

<div class="form-group">
<label for="user"> GENDER</label>
<input type="text" name="gender" id="user" class="form-control">
</div>

<div class="form-group">
<label for="user"> AGE </label>
<input type="text" name="age" id="user" class="form-control">
</div>

<div class="form-group">
<label for="user"> DIRECTOR </label>
<input type="text" name="director" id="user" class="form-control">
</div>

<div class="form-group">
<label for="file"> IMAGE </label>
<input type="file" name="file" id="file" class="form-control">
</div>
<input type="submit" name="submit" value="submit" class="btn btn-success">
</form>
</div>
</div>
</body>
</html>

```

4.2.2 JavaScript Code

Similarly, in both add.php and in edit.php, a JavaScript function is used to dynamically write HTML code to generate extra fields for the addition of new tracks in the database. In add_user.php, there is another JavaScript function that is used to validate the data of the new user authorization being added.

JavaScript code in search.php

```

(function(document) {
    'use strict';

    var LightTableFilter = (function(Err) {

        var _input;

        function _onInputEvent(e) {
            _input = e.target;
            var tables = document.getElementsByClassName(_input.getAttribute('data-table'));
            Err.forEach.call(tables, function(table) {
                Err.forEach.call(table.tBodies, function(tbody) {
                    Err.forEach.call(tbody.rows, _filter);
                });
            });
        }

        function _filter(row) {
            var text = row.textContent.toLowerCase(),
                val = _input.value.toLowerCase();
            row.style.display = text.indexOf(val) === -1 ? 'none' : 'table-row';
        }

        return {
            init: function() {
                var inputs = document.getElementsByClassName('light-table-filter');
                Err.forEach.call(inputs, function(input) {
                    input.oninput = _onInputEvent;
                });
            }
        };
    })(Array.prototype);

    document.addEventListener('readystatechange', function() {
        if (document.readyState === 'complete') {
            LightTableFilter.init();
        }
    });
})(document);

```

4.2.3 PHP Code

The core of the application, apart from the database in itself, is the PHP that is utilized to enter data into the database through the front-end. The PHP code handles all the preparing, and execution of SQL statements as well as run validations for the data entered.

Snippet of PHP code

```
<?php

//header('location:login.php');

$con = mysqli_connect('localhost','root','');
mysqli_select_db($con,'flicks');

$name = $_POST['user'];
$pass = $_POST['password'];

$s = "select * from usertable where name='$name'";

$result = mysqli_query($con,$s);

$number = mysqli_num_rows($result);

if($number == 1){
    echo"username already taken";
}else{
    $reg="insert into usertable(name,password) values ('$name','$pass')";
    mysqli_query($con,$reg);
    echo"registration successful";
}
```

Queries:


```

<?php

//header('location:login.php');

$con = mysqli_connect('localhost','root','');
mysqli_select_db($con,'flick');

$name = $_POST['user'];
$pass = $_POST['password'];

$s = "select * from user where username='$name'";

$result = mysqli_query($con,$s);

$number = mysqli_num_rows($result);

if($number == 1){
    echo"username already taken";
}else{
    $reg="insert into user (username,password) values ('$name','$pass)";
    mysqli_query($con,$reg);
    echo"registration successful";
}

~~~~~
<?php

$con = mysqli_connect('localhost','root','','flick');

        $row = mysqli_query($con,"Select m.movie_name,a.actor,r.description,r.year,r.poster,m.url from movie m inner join  mov_review r
        on m.movie_review_id = r.movie_review_id inner join actors a on a.actor_id = r.actor_id ");

        while($result = mysqli_fetch_array($row)){

            ?>

            <tr>
            <td> <?php echo $result['movie_name']; ?> </td>
            <td> <?php echo $result['actor']; ?> </td>
            <td> <?php echo $result['description']; ?> </td>
            <td> <?php echo $result['year']; ?> </td>
            <td> </td>
            <td> <?php echo $result['url']; ?> </td>
            </tr>

            <?php
        }

```

```

<?php

$con = mysqli_connect('localhost','root','','flick');

    $row = mysqli_query($con,"Select m.movie_name,d.director_name,r.description,r.year,r.poster,m.url from movie m inner join mov_review r
    on m.movie_review_id = r.movie_review_id inner join director d on d.director_id = m.director_id ");

    while($result = mysqli_fetch_array($row)){

        ?>

        <tr>
        <td> <?php echo $result['movie_name']; ?> </td>
        <td> <?php echo $result['director_name']; ?> </td>
        <td> <?php echo $result['description']; ?> </td>
        <td> <?php echo $result['year']; ?> </td>
        <td> </td>
        <td> <?php echo $result['url']; ?> </td>
        </tr>

    <?php

<?php

$con = mysqli_connect('localhost','root','','flick');

    $row = mysqli_query($con,"Select m.movie_name,g.genre,r.description,r.year,r.poster,m.url from movie m inner join mov_review r
    on m.movie_review_id = r.movie_review_id inner join genre g on g.genre_id = r.genre_id ");

    while($result = mysqli_fetch_array($row)){

        ?>

        <tr>
        <td> <?php echo $result['movie_name']; ?> </td>
        <td> <?php echo $result['genre']; ?> </td>
        <td> <?php echo $result['description']; ?> </td>
        <td> <?php echo $result['year']; ?> </td>
        <td> </td>
        <td> <?php echo $result['url']; ?> </td>
        </tr>

    <?php

<?php

$con = mysqli_connect('localhost','root','','flick');

    $row = mysqli_query($con,"Select m.movie_name,m.rating,r.description,r.year,r.poster,m.url from movie m inner join mov_review r
    on m.movie_review_id = r.movie_review_id inner join actors a on a.actor_id = r.actor_id ");

    while($result = mysqli_fetch_array($row)){

        ?>

```

```

<?php

$con = mysqli_connect('localhost','root','');
mysqli_select_db($con,'flick');

$name = $_POST['user'];
$pass = $_POST['password'];

$s = "select * from user where username='$name' && password = '$pass'";

$result = mysqli_query($con,$s);

$number = mysqli_num_rows($result);

if($number == 1){
    $_SESSION['username'] = $name;
    header('location:option.html');
}else{
    echo"unable to login";
}

?>

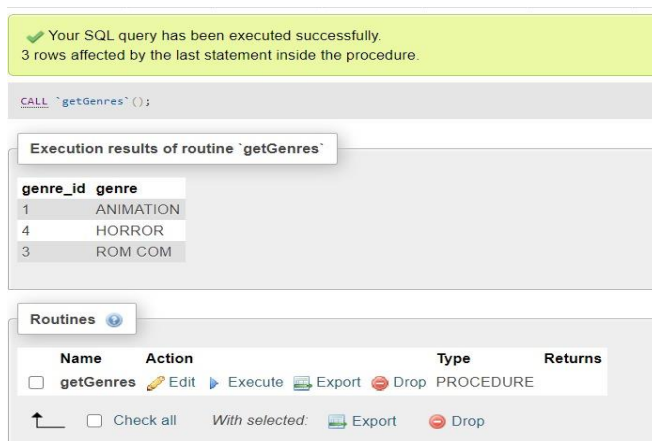
```

Source Code: [DBMS Mini Project\(https://github.com/Akanksha740/website.git\)](https://github.com/Akanksha740/website.git)

4.3 Source Code for Backend

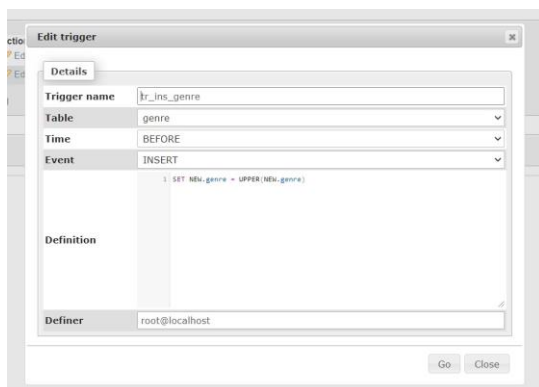
4.3.1 Stored Procedures

A stored procedure is a set of Structured Query Language (SQL) statements with an assigned name, which are stored in a relational database management system as a group, so it can be reused and shared by multiple programs. Stored Procedures are used in the database to run a certain line or lines of code multiple times by calling that procedure in any number of .php files, any number of times.



4.3.2 Triggers

A trigger is a special type of stored procedure that automatically runs when an event occurs in the database server. DML triggers run when a user tries to modify data through a data manipulation language (DML) event. DML events are INSERT, UPDATE, or DELETE statements on a table or view. The trigger used in our database is used to convert the genre entered in any way to upper case which is displayed everywhere.



Chapter 5

Testing

5.1 Introduction

Method Accepted for system testing

System testing is the stage of implementation, which is aimed at ensuring that the system works accurately and efficiently before leave operation commences. An elaborate testing of data is prepared and the system is based using the test data. While testing errors noted and corrections are measured. The users are trained to operate the developed system.

Testing objectives: Testing is a process of executing a program with the intent of finding an error. Good test case is one that has a probability of finding an as yet undiscovered error. A successful test is one that uncovers an uncovered error.

Testing Principle: All tests should be traceable to end user requirements.

Tests should be planned before long test begins.

Testing should begin on small scale and progress towards testing in large.

Exhaustive testing is not possible.

To be most effective, testing should be conducted by an independent party.

Testing strategies: A strategy for software testing integrates software test cases into a series of well-planned steps that result in a successful construction of software. Software testing is a broader topic for what is referred to as verification and validation. Verification refers to the set of activities that ensure that the software that has been built is traceable to the requirements

Testing Steps:

- Unit Testing
- Integration Testing
- Acceptance Testing

5.2 Types of Testing

5.2.1 Unit Testing

Unit testing means testing each units of design separately. Here in this project we tested each unit of design separately and verify that there were no errors. For this testing each design is run individually. After executing each page if there any error occurs correction mechanism is done instantly. XDebug was used as a debugging extension for Visual Studio Code for unit testing.

5.2.2 Integration Testing

In our project we combine many unit modules to form a sub system. These sub systems are then tested directly on the browser, in our case, Google Chrome. This is done to see whether the modules can be integrated properly. Based on integration testing some changes made to the design.

5.2.3 Acceptance testing

The goal of acceptance testing is to see if the software meets all the requirements as needed. The testing was performed by data of all the tables of the system. It was found that the software meets all the requirements of the tables such as movie, actor, director etc. as needed, such as the validation for each input of every table while the user enters them to make sure that the datatype of the information matched the required datatype and the length of the value did not exceed the desired length as well.

5.3 Test Results

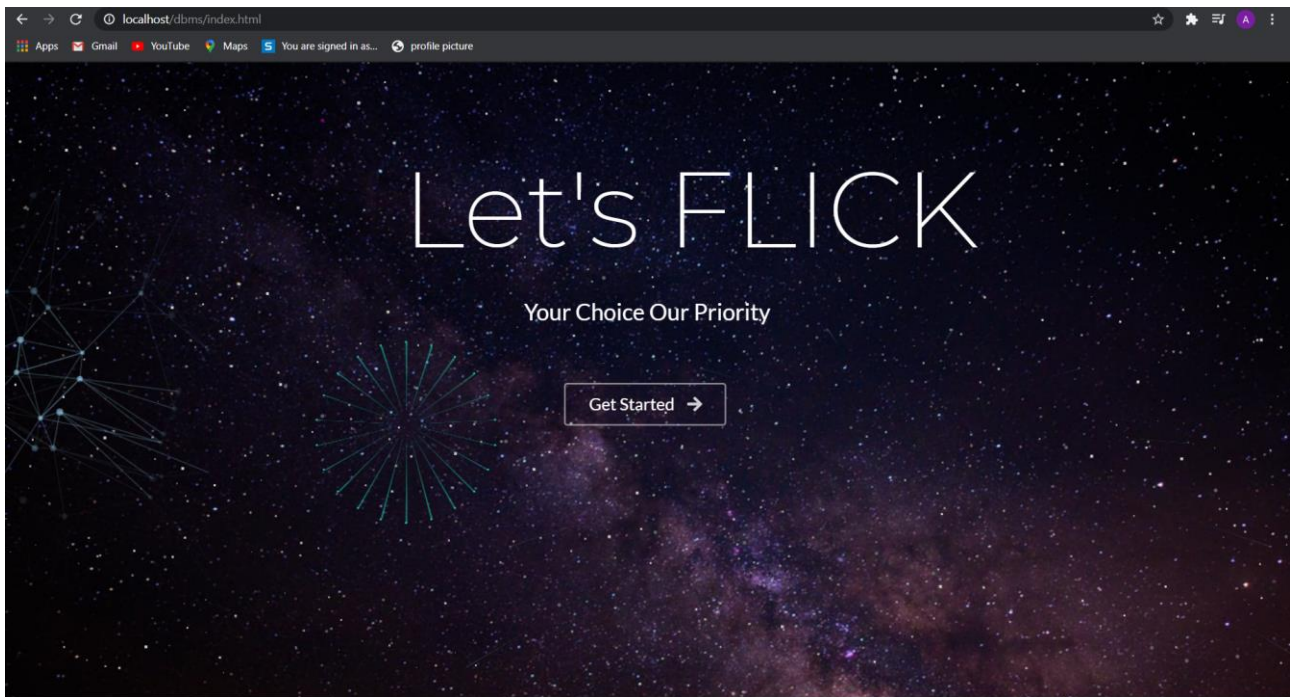
- Some errors were encountered during validation of the input, such as even an empty input being accepted. This was corrected
- Multiple PDOExceptions were encountered due to improper connection being made and consequently solved
- HTML and SQL injections were easy to do which is why functions such as prepare (), execute() and htmlentities() are now being used
- The value from minutes to seconds was not being converted properly. This was solved by changing the datatype of the length attribute in track table to DECIMAL (11,2)
- User password was not being properly hashed and hence often resulted in a mismatch of password entered in the database and password entered in the login. This was found to be due to the different salts being used to hash and was thus corrected
- Many random and out of place elements were found during the CSS styling which were all promptly corrected
- Some features were restructured from the first prototype to the final version to allow better flow of operations
- Playback through YouTube link embedded in the page was added after much trial and error with regular expressions

Chapter 6

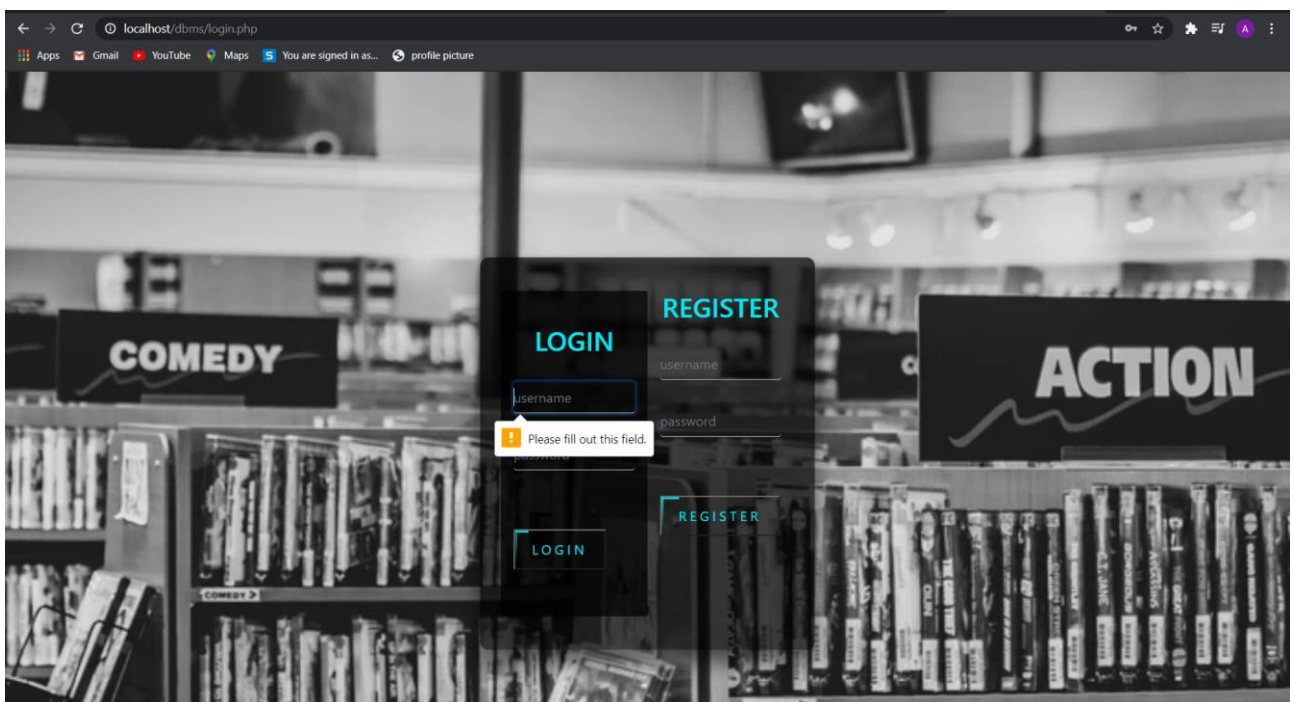
Results

6.1 Screenshots

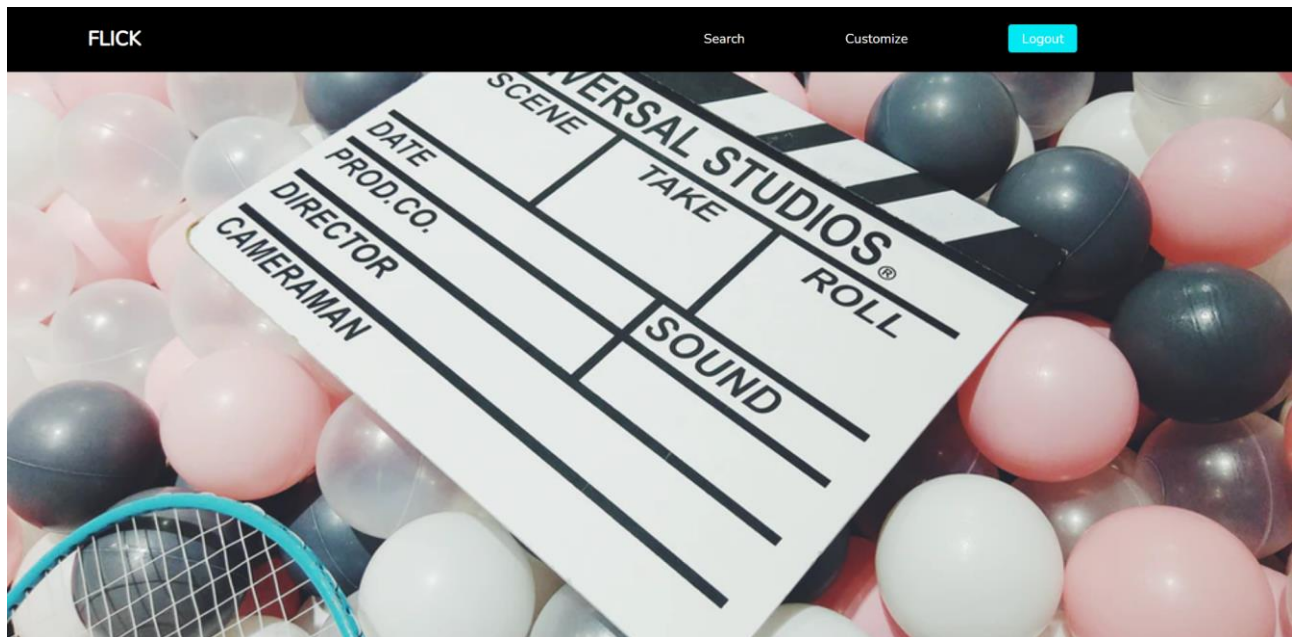
Home Page



Log in page






Home Page



Customize

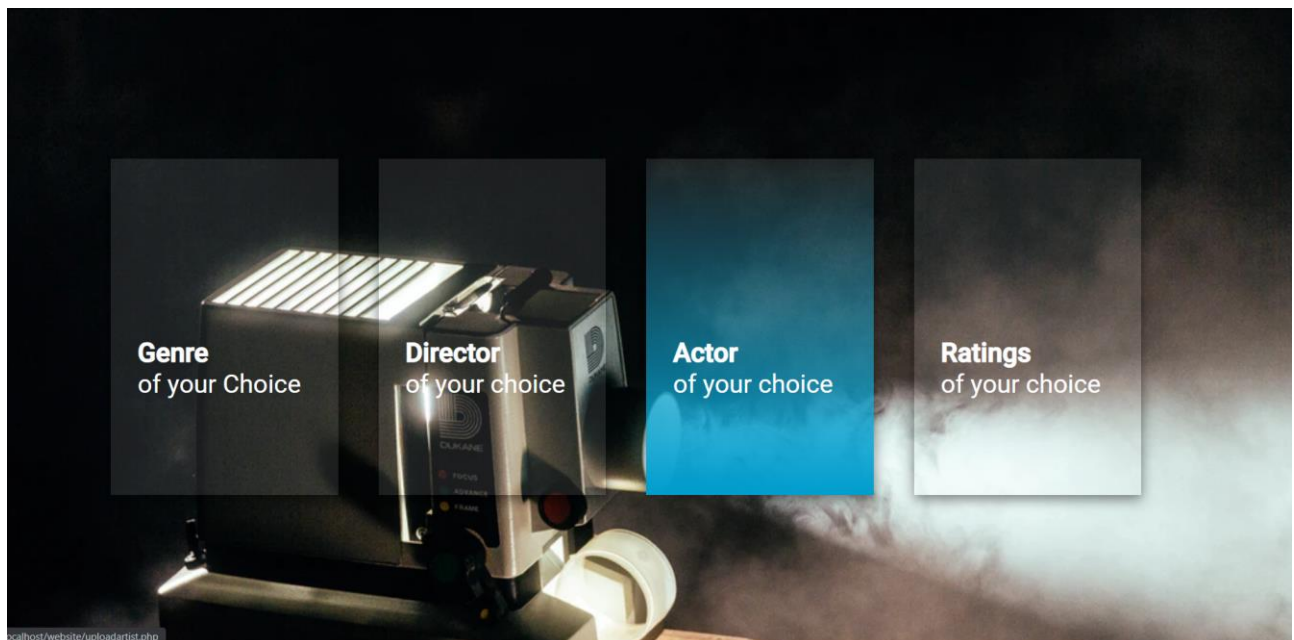
YOUR WATCHLIST...!

		
Ice Age 2 ANIMATION	Kung Fu Panda ANIMATION	Mean Girls ROM COM
4.4	4.7	3.7
add_to_watchlist	add_to_watchlist	add_to_watchlist

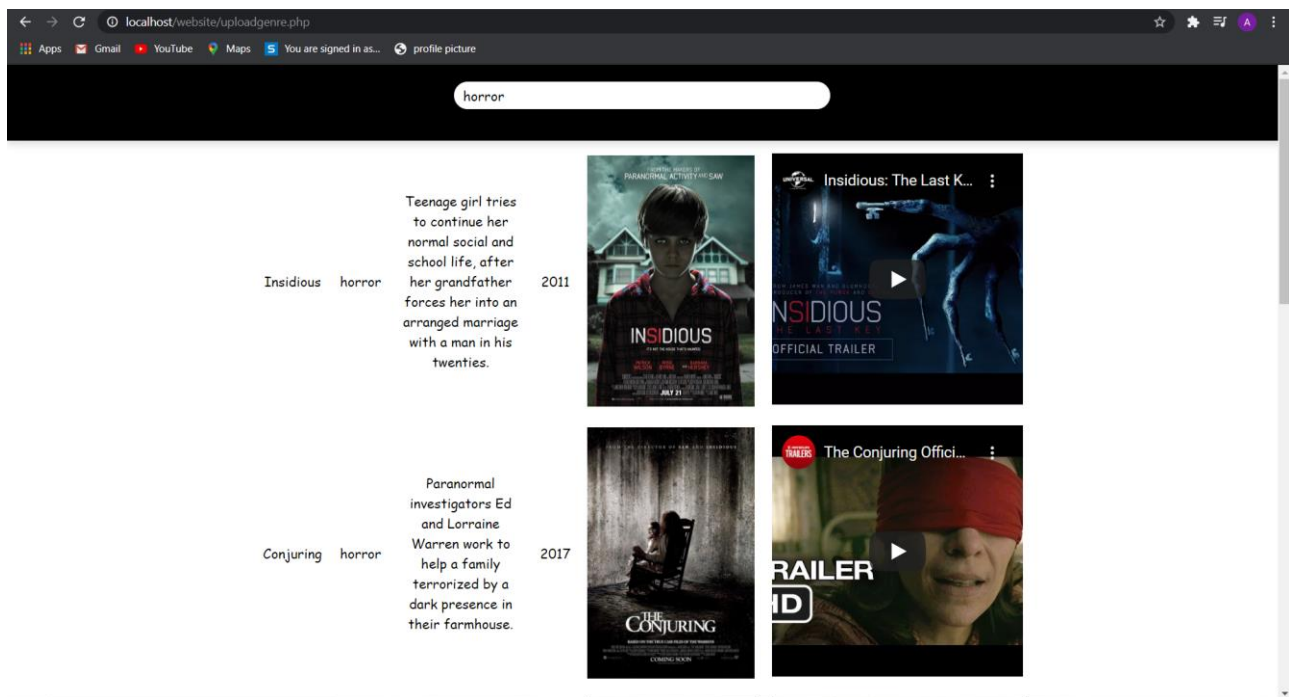
your watchlist

movie_id	movie Name	rating
1	Ice Age 2	4.4

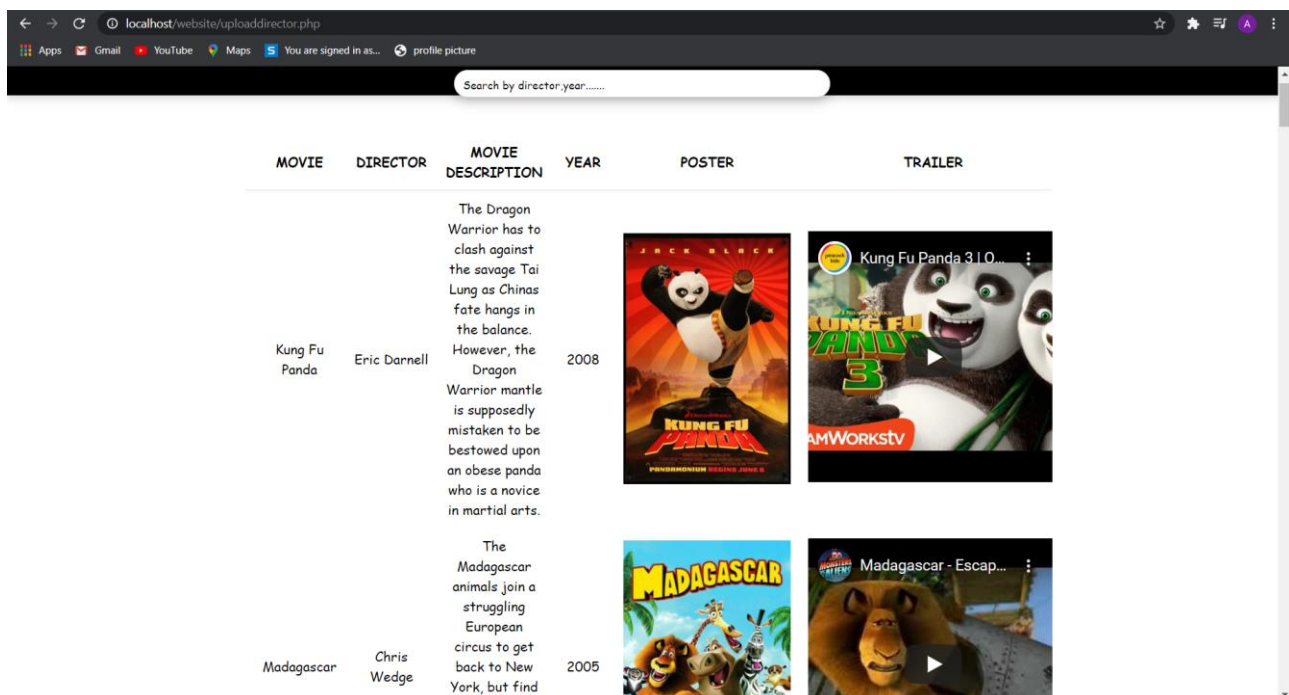
Preferences selection page







Uploaded genre.php






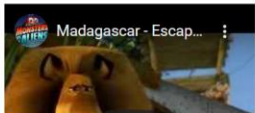
Uploaded director.php



Uploaded artist.php


Search by artist,year,.....					
MOVIE	ARTIST	MOVIE DESCRIPTION	YEAR	POSTER	TRAILER
Kung Fu Panda	Eric Darnell	The Dragon Warrior has to clash against the savage Tai Lung as Chinas fate hangs in the balance. However, the Dragon Warrior mantle is supposedly mistaken to be bestowed upon an obese panda who is a novice in martial arts.	2008		
Madagascar	Eric Darnell	The Madagascar animals join a struggling European circus to get back to New York, but find	2005		

Upload ratings.php

Search by ratings,year,.....					
MOVIE	RATINGS	MOVIE DESCRIPTION	YEAR	POSTER	TRAILER
Kung Fu Panda	★★★★☆	The Dragon Warrior has to clash against the savage Tai Lung as Chinas fate hangs in the balance. However, the Dragon Warrior mantle is supposedly mistaken to be bestowed upon an obese panda who is a novice in martial arts.	2008		
Madagascar		The Madagascar animals join a struggling European circus to get			

create new entry

MOVIE


rating :4

EMBED CODE

MOVIE DESCRIPTION

YEAR

GENRE

ACTOR

GENDER

AGE

Chapter 7

Conclusion

7.1 Conclusion

- Thus, what this project has allowed us to do, is understand the structuring of data in a database and also how that data can be worked on.
- It has allowed us to see the interaction of the database with the front-end through the server and server code.
- We were also able to pick up on the certain basic of web design such as the placements of HTML tags and how they are reflected to the user and how certain CSS values affect certain tags in HTML
- We have also learnt the importance and working of JavaScript and its oldest framework, jQuery.
- In regards to PHP, the potential of the language was realized, mainly in database connectivity, powerful but simplistic code and multiple suitable methods and classes that mesh perfectly with what we wanted to do with the project.
- Security of the application was also taken into consideration to prevent HTML and SQL injection.
- Validation of input both in the front and the back end was taken into consideration
- The result of this project was a stylish yet simplistic site that can now be used to store data

about movies and also information related to that such as genre, artist, director, imbd ratings etc.

7.2 Future Enhancements

- Addition of a search bar to enable sorting and searching of the movies by movie names, director, ratings, genre and artist names.
- Restructuring of the database to include more complex data such as movies preferences etc.
- Migration of the code from “vanilla” or core PHP to a framework such as CodeIgniter or Laravel which is MVC based, or the system of Model-View-Controller
- Page based structural design is used currently and as simple and easy it is to write code in such a way, and also to debug since all the related code in the same file, moving to the MVC system will allow for better flexibility and better reusability of code
- From what we have noticed now, the application is not exactly scalable, both due to software and hardware limitations, this is something that needs to be worked on.
- Security was a property taken into consideration hence, the existence of both user authentication and the presence of certain functions to prevent HTML and SQL injection but it is still not enough. More layers of security must be included to make the data tighter and more strictly controlled calls.
- The GUI can be bettered as well, to enable the CSS script to be more dynamic and include responsive design
- As of now, there is no real way to remove a user account since doing so deletes all the data input by the user or makes it impossible to reach without going into the database manually. This needs to be modified.

Chapter 8

References

8.1 Book References

- Database management systems, Ramakrishnan, and Gehrke, 3rd Edition, 2014, McGraw Hill

8.2 Web References

- <https://stackoverflow.com/>
- <https://www.youtube.com/>
- <https://www.php.net/>
- <https://www.php.net/manual/en/book.pdo.php>
- <https://dev.mysql.com/doc/refman/8.0/en/>
- <https://www.w3schools.com/html/default.asp>
- <https://www.w3schools.com/css/default.asp>
- <https://www.w3schools.com/js/default.asp>

- <https://www.w3schools.com/php/default.asp>
- https://en.wikipedia.org/wiki/Main_Page
- <https://codepen.io/trending>