# VISVESVARAYA TECHNOLOGICAL UNIVERSITY

**"JNANA SANGAMA", BELAGAVI-590018**

**2019-2020**

**A Mini Project Report on**

# "Television Show Database Management"

**SUBMITTED IN PARTIAL FULFILMENT FOR 5TH SEMESTER**

**BACHELOR OF ENGINEERING**

**IN**

**INFORMATION SCIENCE AND ENGINEERING**

**SUBMITTED BY**


**SUPRAD S PARASHAR (1JB17IS096)**


**Under the Guidance of**


**Mr. Vinod H C**
Assistant Professor, Dept.
of ISE,
SJBIT, Bengaluru-60



**DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING**

**SJB INSTITUTE OF TECHNOLOGY**

**BGS HEALTH AND EDUCATION CITY, KENGERI, BENGALURU-560060**

**KARNATAKA, INDIA.**


**2019 - 2020**

||Jai Sri Gurudev||

Sri Adichunchanagiri Shikshana Trust ®

# SJB INSTITUTE OF TECHNOLOGY

BGS Health & Education City, Kengeri, Bengaluru – 60

## Department of Information Science & Engineering

## CERTIFICATE

Certified that the Mini Project entitled "**Television Show Database Management**" carried out by **Mr Suprad S Parashar (1JB17IS096)** is a bonafide work of the student of **SJB Institute of Technology** in partial fulfilment for the award of **BACHELOR OF ENGINEERING** in **INFORMATION SCIENCE AND ENGINEERING** as prescribed by **VISVESVARAYA TECHNOLOGICAL UNIVERSITY, BELAGAVI** during the academic year **2019-20**. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the Report deposited in the Departmental Library. The Mini Project report has been approved as it satisfies the academic requirements in respect of the Mini Project work prescribed for the said Degree.

| | |
|---|---|
| **Mr Vinod H C**<br>Asst. Professor<br>Dept. of ISE, SJBIT | **Dr MOHAN H.S**<br>Prof. & Head<br>Dept. of ISE, SJBIT |

**EXTERNAL VIVA**

| **Name of the Examiners** | **Signature with Date** |
|---|---|
| **1.....** | **.....** |
| **2.....** | **.....** |

# ACKNOWLEDGEMENT

I am grateful to His Divine soul **Padmabhushana Sri Sri Sri Dr Balagangadharanatha Maha Swamiji** and to His Holiness **Sri Sri Sri Jagadguru Dr Nirmalanandanatha Maha Swamiji,** for providing me with an opportunity to complete my academics in this esteemed college.

I would like to express my profound gratitude to his holiness **Revered Sri Sri Dr Prakashnath Swamiji, Managing Director**, SJBIT for providing an opportunity to complete my academics and present this Mini Project.

I am grateful to **Dr Puttaraju**, **Principal,** for his kind co-operation and encouragement and extremely grateful to **Dr Mohan H S**, **Professor and Head,** Department of Information Science and Engineering, for his motivation and encouragement.

I express my deepest gratitude and sincere thanks to **Mr. Vinod H C, Assistant Professor**, Department of Information Science and Engineering for his valuable guidance throughout my Mini Project. I am highly indebted to **Mr. Kiran Kumar V, Mini Project Coordinator** who has been a source of inspiration to me and has extended his support throughout the Mini Project duration.

I also thank all the staff members of the Department of Information Science and Engineering for their help during the course of this Mini Project. Last, but not least, I thank my parents, family members & friends, for their continuous support and encouragement throughout this Mini Project.

**Suprad S Parashar**

**(1JB17IS096)**

# Abstract

Let’s Watch is an online repository of various Television shows with details of each Television show in it. This project caters to the people who need organising and help track their watchlist. Let’s Watch holds the details of a Television Show in a database which enables it to provide users access to it from anywhere around the world.

Using Let’s Watch, users can browse through a list of Television shows, add them to their watchlist, keep track of other Television Shows, etc. The users can keep track of what Television Shows they are watching, and can also rate it after they complete watching the show. The rating is then reflected in the catalogue where different users can refer to it to help them choose the next show to binge on. The catalogue presents a list of available Television Shows which the user has not watched.

The admins of Let’s Watch are responsible to add new Television Shows to update the database. They can add and edit the Television Shows anytime on Let’s Watch. They are responsible to add Actors and Creators of different Television Shows too. They have the privilege of promoting existing users to admins of Let’s Watch.

Let’s Watch provides a clean and user-friendly interface to help users manage their watchlist. Let’s Watch aims to help users, all around the globe, manage their watchlist and help them find the next Television show to binge on.

# Table of Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

## 1.1 Description and Objective

Let's Watch is an online database of Television shows which is specifically designed keeping Television Show connoisseurs and binge-watchers in mind. It is an online repository of Television Shows which contain details of all Television Shows. Let's Watch is targeted toward young audiences and those people who would like to keep tabs on the new and latest Television Shows and to keep track of their watchlist. Let's Watch aims to help keep track of what Television Shows a user has watched, the Television Shows the user is currently watching and also the Television Shows which the user would like to watch in the future.

Let's Watch also displays the details of the Television Shows in the database to help the user make an informed decision on which Television Show to watch. Some of the details include –

1. Title of the Show
2. Synopsis or Description of the show.
3. The number of Seasons
4. The number of Episodes
5. Genre
6. The Actors who played in the Show.
7. The roles played by the actors
8. The Creators of the Show
9. The Air date of its first episode
10. The status of the show – Is it running or has it ended
11. The Channel
12. The End date (If ended)

## 1.2 Need for this Project

Many Television Show bingers face difficulty in keeping track of what Television Shows they have watched and mainly, which show to watch next. Let's Watch tries to solve this by providing a platform where users can add shows to their watchlist, mark shows which they are watching, also mark shows as finished which they have finished watching. This helps users to keep track of Television shows across multiple Channels such as CBS, BBC, The CW, Amazon Prime Videos, Netflix, AMC, etc.

Although many channels such as Netflix provide tracking, they do so only for the Television Shows broadcasted by that channel. Let's Watch solves this problem by taking into consideration all Television Shows across many Platforms.

With the previously watched shows, the ratings provided by other users to other shows, users can make an informed choice on what Television Show to watch next. This helps in providing a better experience to the user and helps them to 'Netflix and Chill'

# 1.3 Technologies Used

### 1.3.1 Front End

The Front end of the project comprises of the User Interface, where the user can directly interact with the Webpage. The front end is the part of the program is the main look of the program where the users can interact with various elements of the webpage. The Front End of the project is designed using tools such as the following -

#### 1.3.1.1 HyperText Markup Language (HTML)

Hypertext Markup Language (HTML) is the standard markup language for documents designed to be displayed in a web browser. It can be assisted by technologies such as Cascading Style Sheets (CSS) and scripting languages such as JavaScript.

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 <img /> and <input /> directly introduce content into the page. Other tags such as <p> surround and provide information about document text and may include other tags as sub-elements. 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 behaviour and content of web pages. Inclusion of CSS defines the look and layout of content.

#### 1.3.1.2 Cascading Style Sheets (CSS)

Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation of a document written in a markup language like 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, colours, 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 and reduce 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, in print, by voice (via speech-based browser or screen reader), and on Braille-based tactile devices. CSS also has rules 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.

#### 1.3.1.3 JavaScript (JS)

JavaScript, often abbreviated as JS, is a high-level, interpreted scripting language. JavaScript 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, and major web browsers have a dedicated JavaScript engine to execute it.

As a multi-paradigm language, JavaScript supports event-driven, functional, and imperative (including object-oriented and prototype-based) programming styles. It has APIs for working with text, arrays, dates, regular expressions, and the DOM, but the language itself does not include any I/O, such as networking, storage, or graphics facilities. It relies upon the host environment in which it is embedded to provide these features.

Initially only implemented client-side in web browsers, JavaScript engines are now embedded in many other types of host software, including server-side in web servers and databases, and in non-web programs such as word processors and PDF software, and in runtime environments that make JavaScript available for writing mobile and desktop applications, including desktop widgets.

Although there are similarities between JavaScript and Java, including language name, syntax, and respective standard libraries, the two languages are distinct and differ greatly in design. JavaScript was influenced by programming languages such as Self and Scheme. The JSON serialization format, used to store data structures in files or transmit them across networks, is based on JavaScript.

### 1.1.2 Back End

The Back end of the project comprises of the logic and the brains behind every interaction of the user with the webpage. The back end of the project is responsible to handle submission of forms, clicks of buttons and more. It is also responsible to store the user data in the database. Some of the technologies used in the back end are –

#### 1.3.2.1 PHP: Hypertext Preprocessor

PHP: Hypertext Preprocessor (or simply PHP) is a general-purpose programming language originally designed for web development. It was originally created by Rasmus Lerdorf in 1994; the PHP reference implementation is now produced by The PHP Group. PHP originally stood for Personal Home Page, but it now stands for the recursive initialism PHP: Hypertext Preprocessor.

PHP code may be executed with a command-line interface (CLI), embedded into HTML code, or used in combination with various web template systems, web content management systems, and web frameworks. PHP code is usually processed by a PHP interpreter implemented as a module in a web server or as a Common Gateway Interface (CGI) executable. The web server outputs the results of the interpreted and executed PHP code, which may be any type of data, such as generated HTML code or binary image data. PHP can be used for many programming tasks outside of the web context, such as standalone graphical applications and robotic drone control.

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.

#### 1.3.2.2 Structured Query Language

Structured Query Language (SQL) is a domain-specific language used in programming and designed for managing data held in a relational database management system (RDBMS), or for stream processing in a relational data stream management system (RDSMS). It is particularly useful in handling structured data, i.e. data incorporating relations among entities and variables.

SQL offers two main advantages over older read-write APIs such as ISAM or VSAM. Firstly, it introduced the concept of accessing many records with one single command. Secondly, it eliminates the need to specify how to reach a record, e.g. with or without an index.

Originally based upon relational algebra and tuple relational calculus, SQL consists of many types of statements, which may be informally classed as sublanguages, commonly: a data query language (DQL), a data definition language (DDL), a data control language (DCL), and a data manipulation language (DML). The scope of SQL includes data query, data manipulation (insert, update and delete), data definition (schema creation and modification), and data access control. Although SQL is essentially a declarative language (4GL), it includes also procedural elements.

SQL was one of the first commercial languages to utilize Edgar F. Codd's relational model. The model was described in his influential 1970 paper, "A Relational Model of Data for Large Shared Data Banks". Despite not entirely adhering to the relational model as described by Codd, it became the most widely used database language.

SQL became a standard of the American National Standards Institute (ANSI) in 1986, and of the International Organization for Standardization (ISO) in 1987. Since then, the standard has been revised to include a larger set of features. Despite the existence of such standards, most SQL code is not completely portable among different database systems without adjustments.

#### 1.3.2.3 Database

A database is an organized collection of data, generally stored and accessed electronically from a computer system. Where databases are more complex, they are often developed using formal design and modelling techniques.

The database management system (DBMS) is the software that interacts with end-users, applications, and the database itself to capture and analyse the data. The DBMS software additionally encompasses the core facilities provided to administer the database. The sum total of the database, the DBMS and the associated applications can be referred to as a "database system". Often the term "database" is also used to loosely refer to any of the DBMS, the database system or an application associated with the database.

Computer scientists may classify database-management systems according to the database models that they support. Relational databases became dominant in the 1980s. These model data as rows and columns in a series of tables and the vast majority use SQL for writing

and querying data. In the 2000s, non-relational databases became popular, referred to as NoSQL because they use different query languages.

# Chapter 2

# Literature Survey

Let’s Watch is an example of a Television Show Database Management Program where Television Shows from a smorgasbord of Channels such as CBS, BBC, The CW, Netflix, AMC, Amazon Prime Videos, etc. Although there are other platforms such as TV Time and Trakt.tv which provide the same functionality as Let’s Watch, Let’s Watch aims at a user-first approach where a user can request the addition of a new Television Show which does not already exist in the Let’s Watch Database.

# Chapter 3

# Requirement Specification

## 3.1 Minimum Hardware Specification

**CPU:** Intel Atom or Higher, AMD A2 or higher

**GPU:** Integrated Graphics or higher

**RAM:** 512 MiB

**Peripherals:** Keyboard, Mouse, Monitor

## 3.2 Minimum Software Requirements

**Operating System:** Windows XP or higher, macOS X Snow Leopard or higher, Linux based Distro

**Browser:** Any browser with support for HTML5, CSS3 and JavaScript

**Other Softwares:** MySQL or MariaDB Database for storing data, A localhost server with a compiler for PHP.

## 3.3 Installing and Executing the Program

### 3.3.1 Installing XAMPP

XAMPP is a free and open-source cross-platform web server solution stack package developed by Apache Friends, consisting mainly of the Apache HTTP Server, MariaDB database, and interpreters for scripts written in the PHP and Perl programming languages. Since most actual web server deployments use the same components as XAMPP, it makes transitioning from a local test server to a live server possible.

XAMPP's ease of deployment means a WAMP or LAMP stack can be installed quickly and simply on an operating system by a developer. With the advantage of common add-in applications such as WordPress and Joomla! can also be installed with similar ease using Bitnami.

### 3.3.2 Using Apache HTTP Server

The Apache HTTP Server, colloquially called Apache, is free and open-source cross-platform web server software, released under the terms of Apache License 2.0. Apache is developed and maintained by an open community of developers under the auspices of the Apache Software Foundation.

The vast majority of Apache HTTP Server instances run on a Linux distribution, but current versions also run on Microsoft Windows and a wide variety of Unix-like systems. Past versions also ran on OpenVMS, NetWare, OS/2 and other operating systems, including ports to mainframes.

Originally based on the NCSA HTTPd server, development of Apache began in early 1995 after work on the NCSA code stalled. Apache played a key role in the initial growth of the World Wide Web, quickly overtaking NCSA HTTPd as the dominant HTTP server, and has remained most popular since April 1996. In 2009, it became the first web server software to serve more than 100 million websites.

### 3.3.3 Using the MariaDB Database

MariaDB is a community-developed, commercially supported fork of the MySQL relational database management system (RDBMS), intended to remain free and open-source software under the GNU General Public License. Development is led by some of the original developers of MySQL, who forked it due to concerns over its acquisition by Oracle Corporation in 2009.

MariaDB intended to maintain high compatibility with MySQL, ensuring a drop-in replacement capability with library binary parity and exact matching with MySQL APIs

and commands. However, new features diverge more. It includes new storage engines like Aria, ColumnStore, and MyRocks.

Its lead developer/CTO is Michael "Monty" Widenius, one of the founders of MySQL AB and the founder of Monty Program AB. On 16 January 2008, MySQL AB announced that it had agreed to be acquired by Sun Microsystems for approximately $1 billion. The acquisition completed on 26 February 2008. MariaDB is named after Monty's younger daughter, Maria. (MySQL is named after his other daughter, My.)

### 3.3.4 Adding the Project Files to the Server.

To start using Let's Watch, we first need to load the project into the localhost server, powered by Apache HTTP Server, by moving the project files to –

1. Windows –
    a. Navigate to C > Xampp > htdocs.
    b. Create a new Folder called Let's Watch.
    c. Move the project files into Let's Watch.
2. macOS –
    a. Using Spotlight search find the htdocs folder.
    b. Create a new Folder called Let's Watch.
    c. Move the project files into Let's Watch.
3. Linux –
    a. Navigate to /opt/lampp/htdocs
    b. Create a new Folder called Let's Watch.
    c. Move the project files into Let's Watch.

To run Let's Watch,

1. Open the Xampp Control Panel
2. Start the Apache Server by pressing the start button next to it. The text 'Apache' will be highlighted in green.
3. Start the MySQL Server by pressing the start button next to it. The text 'MySQL' will be highlighted in green.
4. Open any browser and navigate to the URL, **localhost/Let's Watch/login.php**

## 3.4 Normalisation

The tables in the database are expected to be in the Third Normal Form.

Third normal form (3NF) is a normal form that is used in normalizing a database design to reduce the duplication of data and ensure referential integrity by ensuring that the entity is in second normal form, no non-prime (non-key) attribute is transitively dependent on any key i.e. no non-prime attribute depends on other non-prime attributes and all the non-prime attributes must depend only on the candidate keys.

3NF was designed to eliminate undesirable data anomalies, reduce the need for restructuring over time, make the data model more informative and make the data model neutral to different kinds of query statistics.

# Chapter 4

# System Design

The purpose of the design phase is to develop a clear understanding of what the developer wants people to gain from his/her project. As the developer works on the project, the test for every design decision should be "Does this feature fulfil the ultimate purpose of the project?"

A purpose statement affects the design process by explaining what the developer wants the project to do, rather than describing the project itself. The Design Document will verify that the current design meets all of the explicit requirements contained in the system model as well as the implicit requirements desired by the user.

## 4.1 Structure of Design Document

1. **Entity Relationship Diagram** –

An entity-relationship model (or ER model) describes interrelated things of interest in a specific domain of knowledge. A basic ER model is composed of entity types (which classify the things of interest) and specifies relationships that can exist between entities (instances of those entity types).

An entity-relationship diagram for an MMORPG using Chen's notation.

In software engineering, an ER model is commonly formed to represent things a business needs to remember in order to perform business processes. Consequently, the ER model becomes an abstract data model, that defines a data or information structure which can be implemented in a database, typically a relational database.

Entity-relationship modelling was developed for database and design by Peter Chen and published in a 1976 paper. However, variants of the idea existed previously. Some ER models show super and subtype entities connected by generalization-specialization relationships, and an ER model can be used also in the specification of domain-specific ontologies.

2. **Schema Diagram –**

In database terms, a schema is the organisation and structure of a database. Both schemas and schemata can be used as plural forms. A schema contains schema objects, which could be tables, columns, data types, views, stored procedures, relationships, primary keys, foreign keys, etc. A database schema can be represented in a visual diagram, which shows the database objects and their relationship with each other.

3. **System Architecture –**

A system architecture or systems architecture is the conceptual model that defines the structure, behaviour, and more views of a system. An architecture description is a formal description and representation of a system, organized in a way that supports reasoning about the structures and behaviours of the system.

## 4.2 Entity Relationship Diagram

This relationship diagram shows how the tables in the database are connected to each other and how the control flows from one table to another when some action is triggered by the user. It also shows the constraints on the database such as primary key constraints, foreign key constraints and procedures and triggers. Entity Relationship Diagram is also called an ER Diagram.

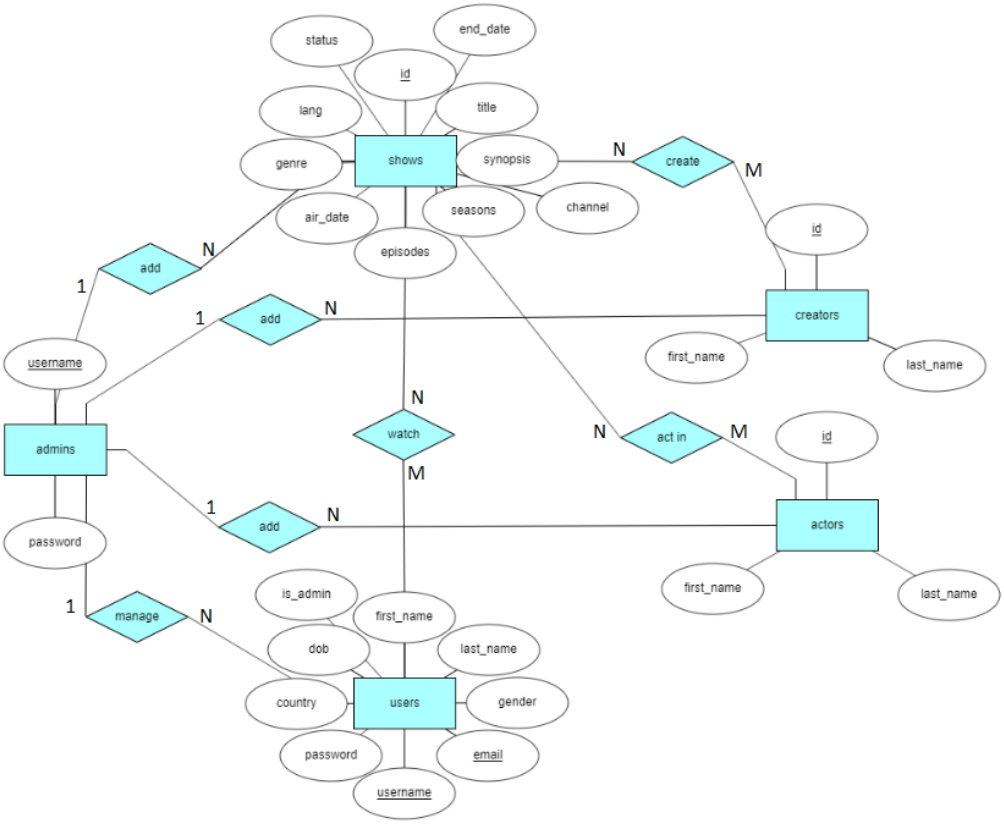



Figure 4.1 – Entity Relationship Diagram

## 4.3 Schema Diagram

The Schema Diagram gives us information about the attributes in the table of the database and how the given tables are related to each other.

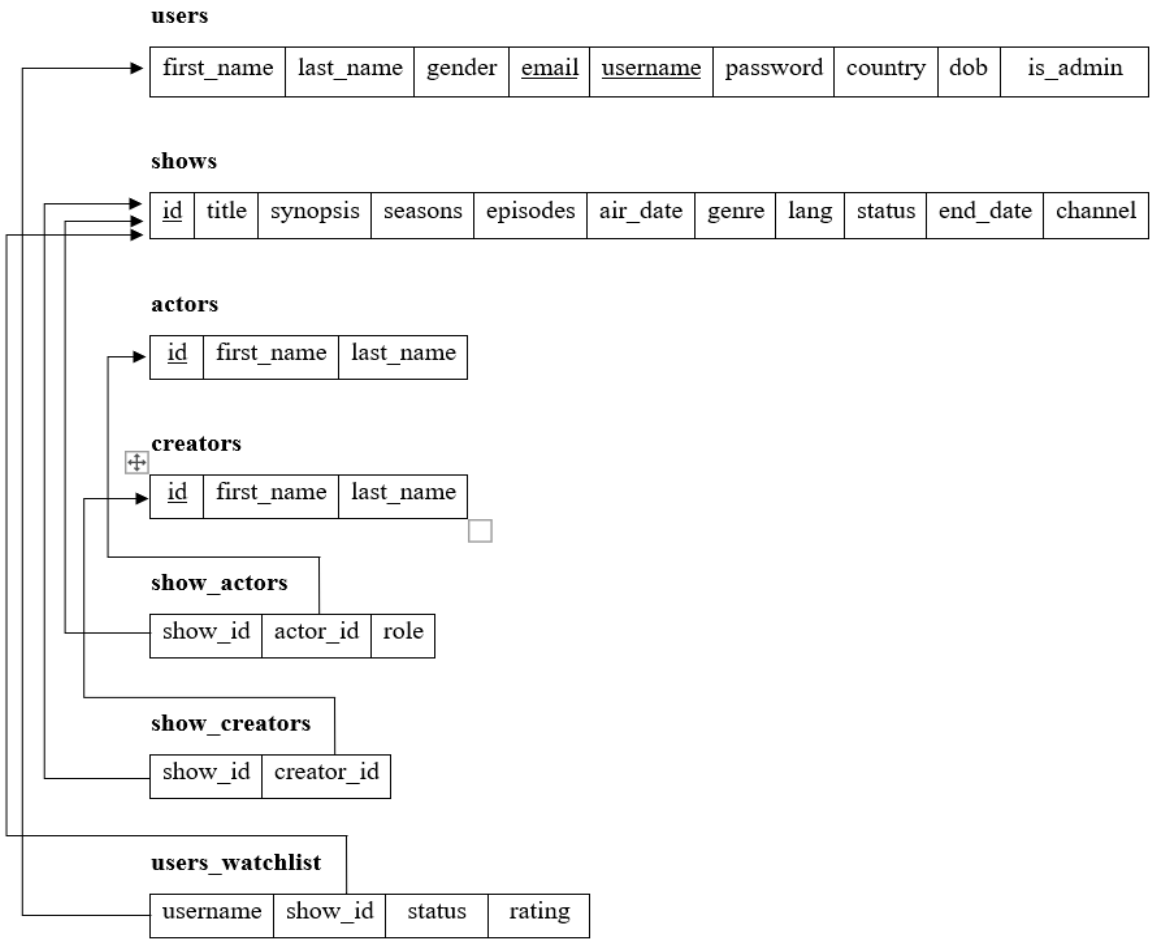



Figure 4.2 – Schema Diagram

## 4.4 Creating a Database

Once we start the Apache Server and the MySQL Server, the next step is using MySQL and creating a database and table which will hold information to be used by Let's Watch. The XAMPP package contains an application called phpMyAdmin which allows developers to administer and maintain MySQL databases. We will be using phpMyAdmin

to create a database and table, and enter test data. Before testing phpMyAdmin, make sure that both Apache and MySQL are running by opening their respective batch files: apache_start.bat and mysql_start.bat. After Apache and MySQL are running in the background, we navigate to **http://localhost/phpmyadmin/** using a web browser.

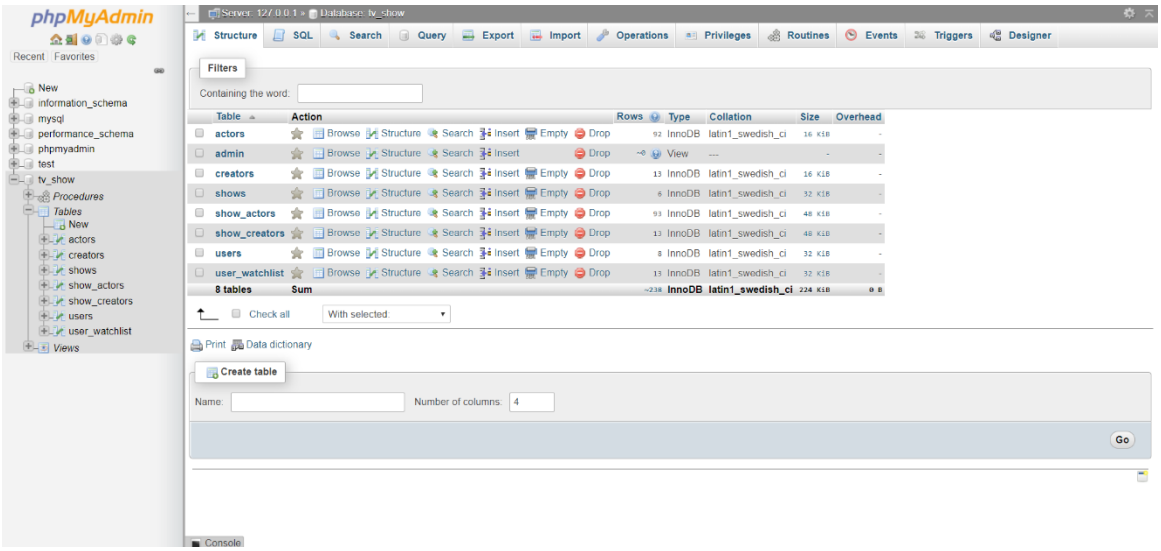



Figure 4.3 – PHPMyAdmin showing all Tables

The above picture shows how exactly the phpMyAdmin page looks like. All the SQL commands can be executed here.

## 4.5 Table Design

1. **Users Table Design**

The users table consists of details of all the users of the Let's Watch Program. The schema of the users table is as shown below.

| first_name | last_name | gender | email | username | password | country | dob | is_admin |
|---|---|---|---|---|---|---|---|---|

a. first_name is the First Name of the user. It is of type varchar.
b. last_name is the Last Name of the user. It is of type varchar.

c. gender is the Gender of the user. It is of type int. The gender of the user is stored as 0 for male, 1 for female and 2 for others.
d. email is the Email of the user. It is of type varchar. It is a primary key of this table. Hence, there can only be one user with a unique email.
e. username is the login id of the user. It is of type varchar. It is a primary key of this table. Hence, no two users can have the same username.
f. password is the Password used by the user to log into the application. It is of type varchar.
g. country is the Country in which the user resides. It is of type varchar.
h. dob is the user's date of birth. It is of type date.
i. is_admin is the status of the user. It is of type int. Here, 0 represents that the user is not an admin of Let's watch whereas 1 represents that the user is an admin of Let's Watch.

2. **Shows Table Design**

The shows table consists of details of all the shows present in the Let's Watch Program. The schema of the shows table is as shown below.

| id | title | synopsis | seasons | episodes | air_date | genre | lang | status | end_date | channel |
|---|---|---|---|---|---|---|---|---|---|---|

a. id is an identification number given to every show. It is of type int and is auto-incremented. It is the primary key of the table. Hence, each show is uniquely identified by its id.
b. title is the Title of the Television Show. It is of type varchar.
c. synopsis is the Description of the plot of the Television Show. It is of type varchar.
d. seasons is the number of seasons the show has completed. It is of type int.
e. episodes is the number of episodes aired in the Television Show. It is of type int.
f. air_date is the date on which the first episode of the Television Show aired. It is of type date.
g. genre is the Genre of the Television Show. It is of type varchar.
h. lang is the Language of the Television Show. It is of type varchar.
i. status represents the Status of the Television Show. It is of type int. Here, 0 represents that the show is Still Running, whereas 1 represents that it has ended.
j. end_date is the date of the last episode of the Television show. It is of type date.

k. channel is the channel on which the television show is broadcasted. It is of type varchar.

3. **Actors Table Design**

The actors table consists of the names of the actors. The schema of the actors table is as shown below.

| id | first_name | last_name |
|---|---|---|

a. id is the identification number given to an actor. It is of type int. It is the primary key of the table and is auto-incremented. Hence, the id uniquely identifies an actor.
b. first_name is the First Name of the Actor. It is of type varchar.
c. last_name is the Last Name of the Actor. It is of type varchar.

4. **Creators Table Design**

The creators table consists of the names of the creators. The schema of the creators table is as shown below.

| id | first_name | last_name |
|---|---|---|

a. id is the identification number given to a creator. It is of type int. It is the primary key of the table and is auto-incremented. Hence, the id uniquely identifies a creator.
b. first_name is the First Name of the Creator. It is of type varchar.
c. last_name is the Last Name of the Creator. It is of type varchar.

5. **show_actors Table Design**

The show_actors table maps the actors and the shows in which they have acted in. The schema of the show_actors table is as shown below.

| show_id | actor_id | role |
|---|---|---|

a. show_id represents the Id Number of the show. It is a foreign key to the table. It references the id attribute in the shows table. It is of type int.
b. actor_id represents the Id Number of the actor. It is a foreign key to the table. It references the id attribute in the actors table. It is of type int.
c. role represents the Name of the Character played by the Actor in a particular show. It is of type varchar.

6. **show_creators Table Design**

The show_creators table maps the creators and the shows they have created. The schema of the show_creators table is as shown below.

| show_id | creator_id |
|---|---|

a. show_id represents the Id Number of the show. It is a foreign key to the table. It references the id attribute in the shows table. It is of type int.
b. creator_id represents the Id Number of the creator. It is a foreign key to the table. It references the id attribute in the creators table. It is of type int.

7. **users_watchlist Table Design**

The users_watchlist table records the shows watched by the users. The schema of the users_watchlist table is as shown below.

| username | show_id | status | rating |
|---|---|---|---|

a. username represents the username of the User. It is a foreign key to the table. It references the username attribute in the users table. It is of type varchar.
b. show_id represents the Id Number of the show. It is a foreign key to the table. It references the id attribute in the shows table. It is of type int.
c. status represents the status of the show to the user. It is of type int. Here, 0 represents that the user has not yet started to watch the show, i.e., it is in his/hers watchlist, 1

represents that the user is currently watching the show and 2 represents that the user has completed watching the show.

d. rating is the Rating provided by the user to the show after he has completed watching it.

## 4.6 Views

This Project consists of one view called admins derived from the users table. The admin table contains the username and password of those users who have admin status.

# Chapter 5

# Implementation

The implementation of this project is divided into many Modules. Each module has its own feature and is responsible for it. This division of the program into smaller chunks or modules help to implement the program easily and also helps in the debugging of the code. The project is divided into the following Modules.

## 5.1 Module 1: Login

The Login Module is responsible for logging a user into the Let's Watch Application. It is also responsible to redirect the users to different pages such as Registration, Admin Login, etc.

The Login module is developed using HTML, CSS and PHP.

**login.php (Front End with HTML) –**

1. Consists of a form with input for username and password, where users can log into the Let's Watch Application.
2. Contains a link to redirect new users to the registration page.
3. Contains a link to the Admin Login module where admins can log into the Let's Watch Application.

**login.php (Back End for Database Connection) –**

1. Retrieves the information provided by the user in the login page and stores it in variables.
2. Creates a connection to the MariaDB Database and connects to the tv_show Database.
3. Calls a stored procedure which checks if the username and password entered matches with an entry in the users table of the database.
4. Redirects the user to the home page if successful, else returns to the login page.

**login.css –**

Responsible for styling the elements in the login.php (Front End) page.

## 5.2 Module 2: Registration

The Registration module is responsible to register new users and store their details in the database.

The Registration Module is developed using HTML, PHP and CSS.

**register.php (Front End with HTML) –**

1. Consists of a form with details such as Name, Email, Username, Password, etc where users can fill in the details and register themselves in the Let's Watch Application.
2. Contains a button to redirect users to the login page if they wish to cancel the registration.

**register.php (Back End for Database Connection) –**

1. Retrieves the information provided by the user in the registration page and stores it in variables.
2. Creates a connection to the MariaDB Database and connects to the tv_show Database.
3. Checks trigger before inserting to check if the username and email are already in use. If so, it redirects the user to the registration page with an error.
4. Executes an SQL Query which inserts the details of the user to the users table in the database.
5. Redirects the user to the login page if successful.

**login.css –**

Responsible for styling the elements in the register.php (Front End) page.

## 5.3 Module 3: Admin Login

The Admin Login Module is responsible for logging an admin into the Let's Watch Application.

The Admin Login module is developed using HTML, CSS and PHP.

**admin_login.php (Front End with HTML) –**

1. Consists of a form with input for username and password, where admins can log into the Let's Watch Application.
2. Contains a link to the Login module where users can log into the Let's Watch Application.

**admin_login.php (Back End for Database Connection) –**

1. Retrieves the information provided by the admin in the admin login page and stores it in variables.
2. Creates a connection to the MariaDB Database and connects to the tv_show Database.
3. Executes an SQL query which checks if the username and password entered matches with an entry in the admin view of the database.
4. Redirects the user to the home page if successful, else returns to the login page.

**login.css –**

Responsible for styling the elements in the admin_login.php (Front End) page.

## 5.4 Module 4: Dashboard

The dashboard module is responsible for showing the watchlist of the user. It is divided into three sections, namely –

1. Currently Watching
2. Finished Watching
3. Not Started Yet

The Dashboard module is implemented using HTML, CSS, PHP and JS.

**home.php (Front End with HTML) –**

1. Responsible to display the Television Shows based on their status in their respective sections.
2. Responsible to redirect the users to the Catalogue and Profile Tabs.
3. Responsible to log the user out.

**home.php (Back End for Database Connection) –**

1. Creates a connection to the MariaDB Database and connects to the tv_show Database.
2. Responsible for changing the status of the Television Shows to the user.

**navigation.css –**

Responsible to style the navigation bar.

**dashboard.css –**

Responsible to style the elements of the Dashboard tab.

### 5.4.1 Currently Watching

The currently watching section lists the Television Shows which are currently being watched by the user. It is implemented by a table which displays the title of the show and the options to the show. Clicking on the title redirects the user to the details of the Television Show.

The user is presented with two options – Completed Watching and Remove Show.

The Completed watching button changes the status of the show to Finished Watching and removes it from the Currently Watching List.

The Remove Show button removes the show from the user's watchlist.

### 5.4.2 Finished Watching

The finished watching section lists the television shows which have been watched by the user. It is implemented by a table which displays the title of the show, the rating of the user

to the show, and the options to the show. Clicking on the title redirects the user to the details of the Television Show.

The user is presented with two options – Save Rating and Remove Show.

The Save rating button saves the rating in the input field to the rating attribute in the user_watchlist database.

The Remove Show button removes the show from the user's watchlist.

### 5.4.3 Not Started Yet

The not started yet section lists the Television Shows which the user has not started to watch yet. It is implemented by a table which displays the title of the show and the options to the show. Clicking on the title redirects the user to the details of the Television Show.

The user is presented with three options – Start Watching, Completed Watching and Remove Show.

The Start Watching button changes the status of the show to Currently Watching and removes it from the Not Started Yet List.

The Completed watching button changes the status of the show to Finished Watching and removes it from the Currently Watching List.

The Remove Show button removes the show from the user's watchlist.

## 5.5 Module 5: Catalogue

The catalogue module is responsible to display the list of television Shows which the user has not yet started.

Here, the user is presented with the title of the show, the aggregate rating of the show given by other users and options to start watching the show or to add it to the user's watchlist.

**catalog.php (Front End with HTML) –**

1. Responsible for displaying the list of Television Shows the user has not watched in a table format.

2. The table also contains the links to the details of the Television Show.

**catalog.php (Back End for Database Connection) –**

1. Creates a connection to the MariaDB Database and connects to the tv_show Database.
2. Responsible for adding the Television Shows to the user's watchlist.

**navigation.css –**

Responsible to style the navigation bar.

**catalog.css –**

Responsible to style the elements of the Catalog tab.

## 5.6 Module 6: Profile

The profile module is responsible to display the details of the user and also to allow the user to change it whenever required.

The profile section contains links to edit the profile and to change the password.

The Profile Section displays the following details of the user –

1. Gender
2. Email
3. Country
4. Date of Birth

**profile.php (Front End with HTML) –**

1. Responsible to display the user data
2. Redirect users to Edit Profile and change Password sections.

**profile.php (Back End for Database Connection) –**

1. Creates a connection to the MariaDB Database and connects to the tv_show Database.
2. Responsible for saving changes to the profile section.

**navigation.css –**

Responsible to style the navigation bar.

## 5.7 Module 7: Show Details

This module is responsible to display the details of the Television Show to the user.

The details include the following –

1. Title
2. Description
3. Number of Seasons
4. Number of Episodes
5. Air Date
6. Genre
7. Channel
8. Status
9. Creators
10. Actors
11. Roles played by the Actors

## 5.8 Module 8: Admin Home (Users)

This module displays the list of users along with their names and email address. It enables the admins to make other users as admins or remove their admin status.

**admin_home.php**

Responsible to display the First Name, Last Name and the email address of users of Let's Watch along with the option to make a user as an admin or revoke their admin status. The details of the users are presented in a table format.

**admin_status.php**

Responsible for changing the admin status of the user in the database.

**navigation.css –**

Responsible to style the navigation bar.

**admin_home.css –**

Responsible to style the elements of the Admin Home (Users) tab.

## 5.9 Module 9: Admin Catalog

Responsible for displaying the list of Television Shows along with the links to edit them. The admins are presented with two options namely – To end an active show and to remove the show from the database.

The Admin Catalog contains three sub-sections which are responsible for adding Television Shows, Creators and Actors into the database.

**admin_catalog.php –**

Responsible for showing the table listing the Television Shows in the database.

**shows.php –**

Responsible for adding the shows to the database.

**creators.php –**

Responsible for adding the creators to the database.

**actors.php –**

Responsible for adding the actors to the database.

**navigation.css –**

Responsible to style the navigation bar.

**admin_catalog.css –**

Responsible to style the elements of the Admin Catalog tab.

**popup_box.css –**

Responsible to add the style to the popup windows in the Admin Catalog tab.

**popup.js –**

Responsible to show/hide the End Date field based on the status of the show.

### 5.9.1 Television Shows

The Television shows tab consists of three divisions to add the details of the television shows, to select the actors and enter their roles and, to select the creators of the television show.

### 5.9.2 Actors

The Actors tab contains a form to add actors into the database. It also displays the actors already present in the database, along with the option to remove them.

### 5.9.3 Creators

The Creators tab contains a form to add creators into the database. It also displays the creators already present in the database, along with the option to remove them.

## 5.10 Module 10: Edit Shows

This module is responsible to edit the Television Shows which are previously added into the database.

**show_edit.php –**

Responsible to display the form style approach to edit the Television Show.

**get_show_details.php –**

Responsible to obtain the details of the show and also to save the changes made to it into the database.

**navigation.css –**

Responsible to style the navigation bar.

**popup.js –**

Responsible to show/hide the End Date field based on the status of the show.

# 5.11 Table Creation

The following are the SQL Statements to create the tables.

## 5.11.1 Users Table

CREATE TABLE users (first_name VARCHAR(30), last_name VARCHAR(30), gender INT, email VARCHAR(50) PRIMARY KEY, username VARCHAR(30) PRIMARY KEY, password VARCHAR(30), country VARCHAR(30), dob DATE, is_admin INT);

| # | Name | Type | Collation | Attributes | Null | Default | Comments | Extra | Action |
|---|---|---|---|---|---|---|---|---|---|
| 1 | first_name | varchar(30) | latin1_swedish_ci | | No | None | | | Change Drop More |
| 2 | last_name | varchar(30) | latin1_swedish_ci | | No | None | | | Change Drop More |
| 3 | gender | int(11) | | | No | None | | | Change Drop More |
| 4 | email | varchar(50) | latin1_swedish_ci | | No | None | | | Change Drop More |
| 5 | username | varchar(30) | latin1_swedish_ci | | No | None | | | Change Drop More |
| 6 | password | varchar(30) | latin1_swedish_ci | | No | None | | | Change Drop More |
| 7 | country | varchar(30) | latin1_swedish_ci | | No | None | | | Change Drop More |
| 8 | dob | date | | | No | None | | | Change Drop More |
| 9 | is_admin | int(11) | | | No | 0 | | | Change Drop More |

Figure 5.1 – Users Table

## 5.11.2 Shows Table

CREATE TABLE shows (id INT PRIMARY KEY AUTO_INCREMENT, title VARCHAR(30), synopsis VARCHAR(1000), seasons INT, episodes INT, air_date DATE, genre VARCHAR(30), lang VARCHAR(30), status INT, end_date DATE, channel VARCHAR(30));

| # | Name | Type | Collation | Attributes | Null | Default | Comments | Extra | Action |
|---|---|---|---|---|---|---|---|---|---|
| 1 | id | int(11) | | | No | None | | | Change Drop More |
| 2 | title | varchar(30) | latin1_swedish_ci | | No | None | | | Change Drop More |
| 3 | synopsis | varchar(1000) | latin1_swedish_ci | | No | None | | | Change Drop More |
| 4 | seasons | int(11) | | | No | None | | | Change Drop More |
| 5 | episodes | int(11) | | | No | None | | | Change Drop More |
| 6 | air_date | date | | | No | None | | | Change Drop More |
| 7 | genre | varchar(30) | latin1_swedish_ci | | No | None | | | Change Drop More |
| 8 | lang | varchar(30) | latin1_swedish_ci | | No | None | | | Change Drop More |
| 9 | status | int(11) | | | No | None | | | Change Drop More |
| 10 | end_date | date | | | Yes | NULL | | | Change Drop More |
| 11 | channel | varchar(30) | latin1_swedish_ci | | No | None | | | Change Drop More |

Figure 5.2 – Shows Table

### 5.11.3 Actors Table

CREATE TABLE actors (id INT PRIMARY KEY AUTO_INCREMENT, first_name VARCHAR(30), last_name VARCHAR(30));

| # | Name | Type | Collation | Attributes | Null | Default | Comments | Extra | Action |
|---|---|---|---|---|---|---|---|---|---|
| 1 | id | int(11) | | | No | None | | AUTO_INCREMENT | Change Drop More |
| 2 | first_name | varchar(50) | latin1_swedish_ci | | No | None | | | Change Drop More |
| 3 | last_name | varchar(50) | latin1_swedish_ci | | No | None | | | Change Drop More |

Figure 5.3 – Actors Table

### 5.11.4 Creators Table

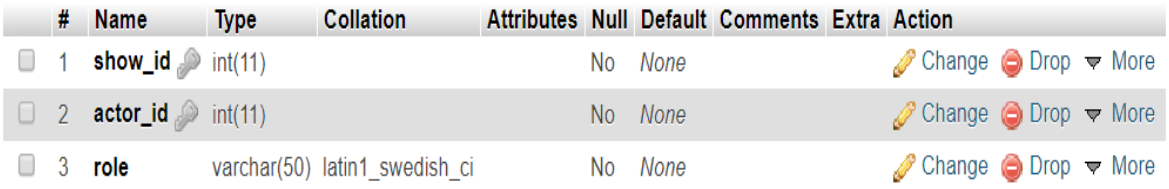
CREATE TABLE creators (id INT PRIMARY KEY AUTO_INCREMENT, first_name VARCHAR(30), last_name VARCHAR(30));

| # | Name | Type | Collation | Attributes | Null | Default | Comments | Extra | Action |
|---|---|---|---|---|---|---|---|---|---|
| 1 | id | int(11) | | | No | None | | AUTO_INCREMENT | Change Drop More |
| 2 | first_name | varchar(50) | latin1_swedish_ci | | No | None | | | Change Drop More |
| 3 | last_name | varchar(50) | latin1_swedish_ci | | No | None | | | Change Drop More |

Figure 5.4 – Creators Table

### 5.11.5 Show Actors Table

CREATE TABLE show_actors (show_id REFERENCES shows(id) ON DELETE RESTRICT, actor_id REFERENCES actors(id) ON DELETE RESTRICT, role VARCHAR(50));

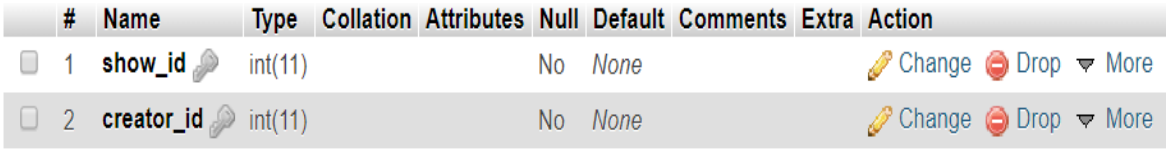


| # | Name | Type | Collation | Attributes | Null | Default | Comments | Extra | Action |
|---|---|---|---|---|---|---|---|---|---|
| 1 | show_id | int(11) | | | No | None | | | Change Drop More |
| 2 | actor_id | int(11) | | | No | None | | | Change Drop More |
| 3 | role | varchar(50) | latin1_swedish_ci | | No | None | | | Change Drop More |

Figure 5.5 – Show Actors Table

### 5.11.6 Show Creators Table

CREATE TABLE show_creators (show_id REFERENCES shows(id) ON DELETE RESTRICT, creator_id REFERENCES creators(id) ON DELETE RESTRICT);



| # | Name | Type | Collation | Attributes | Null | Default | Comments | Extra | Action |
|---|---|---|---|---|---|---|---|---|---|
| 1 | show_id | int(11) | | | No | None | | | Change Drop More |
| 2 | creator_id | int(11) | | | No | None | | | Change Drop More |

Figure 5.6 – Show Creators Table

### 5.11.7 User Watchlist Table

CREATE TABLE user_watchlist (username REFERENCES users(username) ON DELETE RESTRICT, show_id REFERENCES shows(id) ON DELETE RESTRICT, status INT, rating INT);

| # | Name | Type | Collation | Attributes | Null | Default | Comments | Extra | Action |
|---|---|---|---|---|---|---|---|---|---|
| 1 | username | varchar(30) | latin1_swedish_ci | | No | *None* | | | Change Drop More |
| 2 | show_id | int(11) | | | No | *None* | | | Change Drop More |
| 3 | status | int(11) | | | No | 0 | | | Change Drop More |
| 4 | rating | int(11) | | | No | 1 | | | Change Drop More |

Figure 5.7 – User Watchlist Table

## 5.12 Stored Procedure

A procedure (often called a stored procedure) is a subroutine like a subprogram in a regular computing language, stored in a database. There are many useful applications of SQL procedures within a database or database application architecture. SQL procedures can be used to create simple scripts for quickly querying transforming, updating data, generating basic reports, improve application performance, modularizing applications, and improve overall database design, and database security.

The Stored Procedure in this project takes two arguments that is the username and the password of the user. The Procedure checks if such a combination of username and password already exists and returns the number of combinations found.

**MariaDB SQL Statement to create a Stored Procedure -**

CREATE PROCEDURE `access` (IN `user` VARCHAR(30), IN `pass` VARCHAR(30)) DEFINER SELECT COUNT(*) AS tot FROM users WHERE username=user AND password=pass

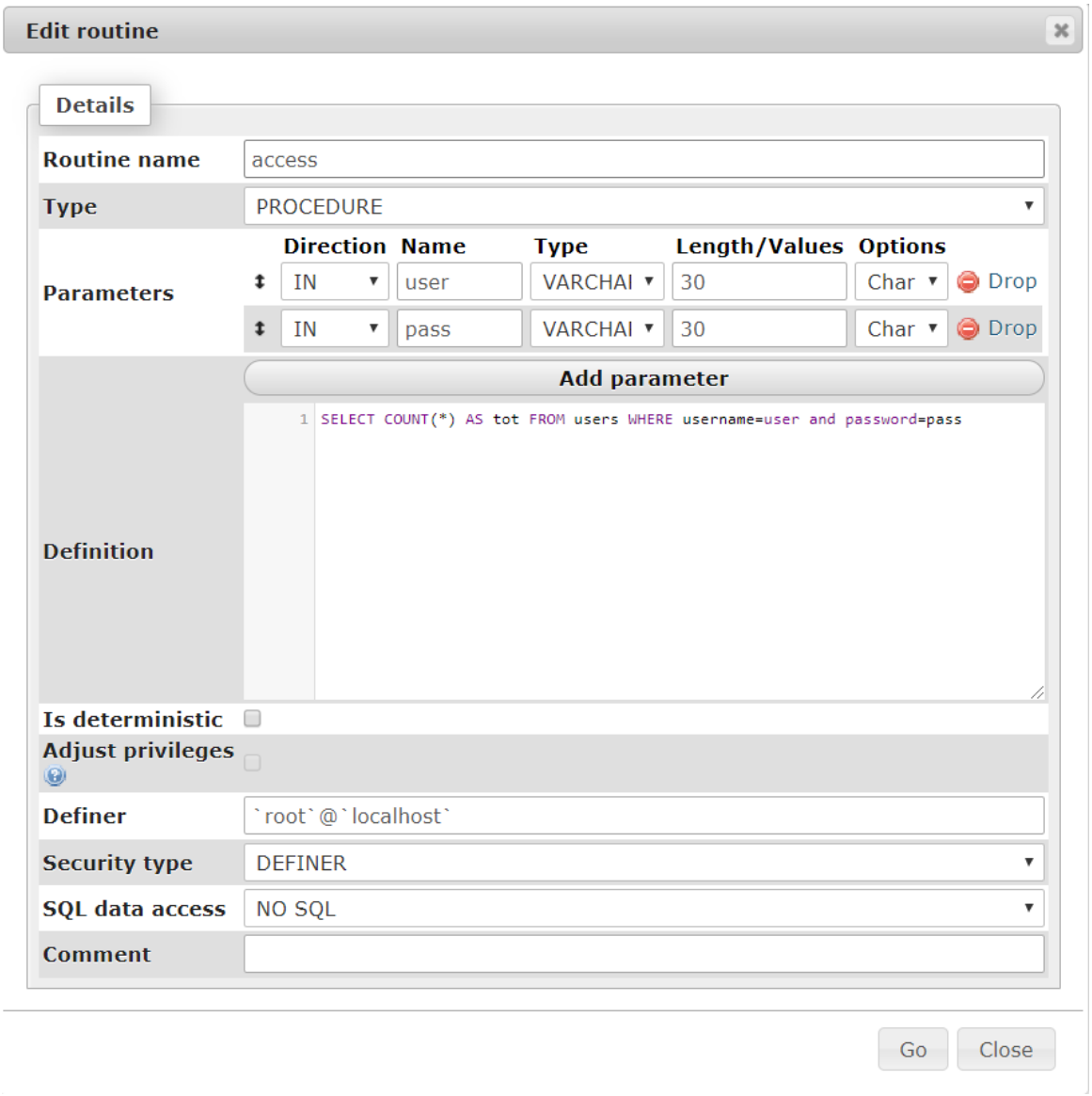



Figure 5.8 – Stored Procedure

## 5.13 Trigger

Triggers are stored programs, which are automatically executed or fired when some event occurs. Triggers are written to be executed in response to any of the following events. A database manipulation (DML) statement (DELETE, INSERT, or UPDATE). Database definition (DDL) statements (CREATE, ALTER, or DROP).

There is one trigger in this project. The trigger checks if there is a conflict with the email and the username before inserting it into the users table of the database.

```
CREATE TRIGGER `check_email` BEFORE INSERT ON `users` FOR EACH ROW IF (SELECT COUNT(*) FROM users WHERE email = new.email OR username = new.username) > 1 THEN SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'ERROR: CREDENTIALS ALREADY EXISTED!'; END IF
```

# Chapter 6

# System Testing

The aim of the system testing process was to determine all defects in our project. The program was subjected to a set of test inputs and various observations were made and based on these observations it will be decided whether the program behaves as expected or not. This Project went through two levels of testing –

1. Unit Testing
2. Integration Testing

| Functionality | Action | Expected Result | Actual Result | Test Result |
|---|---|---|---|---|
| Accept User input to log into the Let's Watch Application | The required details are entered and the login button is clicked. | Redirect the user to the dashboard section | Redirected the user to the dashboard section | PASS |
| Accept user input to go to the registration page. | The Click to register button is clicked. | The user is redirected to the registration page. | The user is redirected to the registration page. | PASS |
| Accept User input to create a new User. | The required details are entered and the register button is clicked. | A new user is created and the user is redirected to the login page. | A new user is created and the user is redirected to the login page. | PASS |
| Accept user input to go to the Admin Login page. | The Admin Login button is clicked. | The user is redirected to the admin login page. | The user is redirected to the admin login page. | PASS |

| | | | | |
|---|---|---|---|---|
| Accept User input to log into the Let's Watch Application | The required details are entered and the register button is clicked. | The user is verified as an admin and logged in as an admin and is redirected to the Admin Home section. | The user is verified as an admin and logged in as an admin and is redirected to the Admin Home section. | PASS |
| Accept user input to navigate to different areas of the dashboard. | Any of the links in the User Navigation Pane is clicked | The user is redirected to the respective page. | The user is redirected to the respective page. | PASS |
| Accept user input to change the status of the Television Show. | Any of the buttons – Start Watching and Completed Watching are clicked. | The status of the Television Show is changed to the user. | The status of the Television Show is changed to the user. | PASS |
| Accept user input to remove a Television Show from the watchlist. | The Remove Show button is clicked. | The television show is removed from the user's watchlist. | The television show is removed from the user's watchlist. | PASS |
| Accept user input to change and save the rating to a television show | The rating field is changed and the Save Rating button is clicked. | The rating is changed and updated in the Database. | The rating is changed and upda ted in the Database. | PASS |

| | | | | |
|---|---|---|---|---|
| Accept user input to view the Details of a Television Show. | The title of the television show is clicked. | The user is redirected to the details page of the television show. | The user is redirected to the details page of the television show. | PASS |
| Accept user input to edit the profile of the user. | The Edit Profile button is clicked. | The user is redirected to an edit profile page. | The user is redirected to an edit profile page. | PASS |
| Accept user input to change the details of the user. | The details required are entered and the Save button is clicked. | The details of the user are updated in the database. | The details of the user are updated in the database. | PASS |
| Accept user input to change the password of the user. | The Change Password button is clicked. | The user is redirected to a change password page. | The user is redirected to a change password page. | PASS |
| Accept user input to change the password of the user. | The details required are entered and the Save button is clicked. | The new password of the user is updated in the database. | The new password of the user is updated in the database. | PASS |
| Accept Admin input to add shows, creators and actors into the database. | The respective buttons are clicked and the required data is entered and submitted. | The respective data is added to the database. | The respective data is added to the database. | PASS |

| | | | | |
|---|---|---|---|---|
| Accept Admin input to change the status of the user. | The Make Admin or Remove Admin buttons are clicked. | The user is granted or revoked admin status. | The user is granted or revoked admin status. | PASS |

Table 6.1 – System Testing

# Chapter 7

# Snapshots

## 7.1 User Login

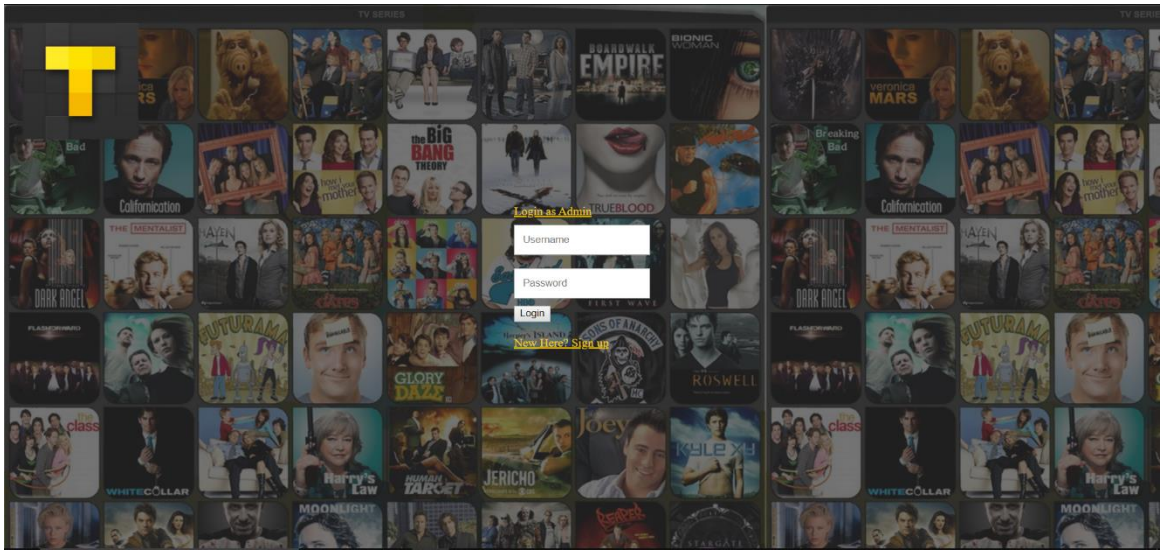



Figure 7.1 – User Login

## 7.2 Registration

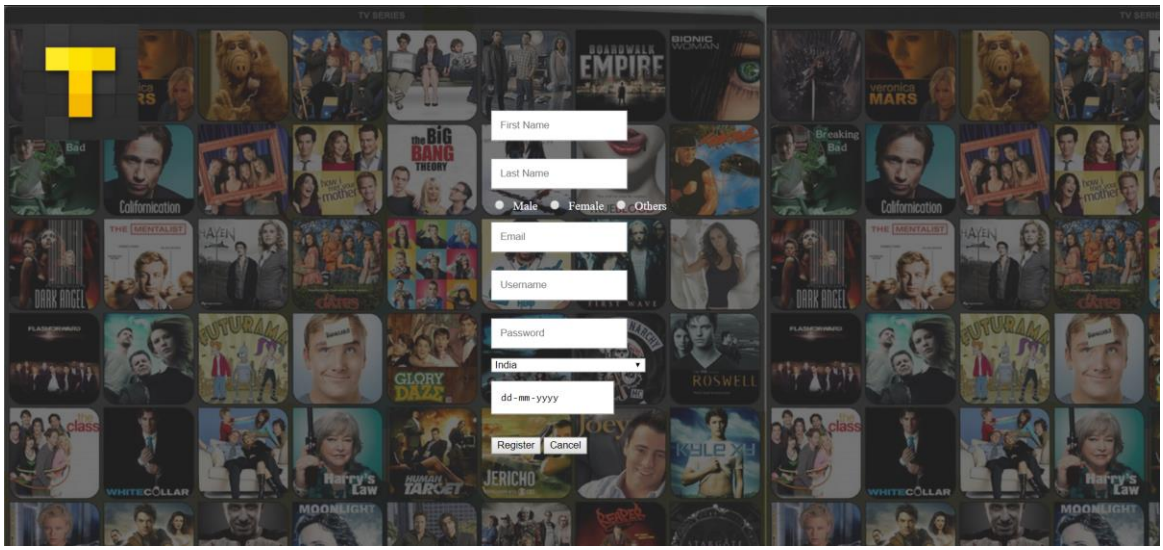



Figure 7.2 – Registration

## 7.3 Admin Login

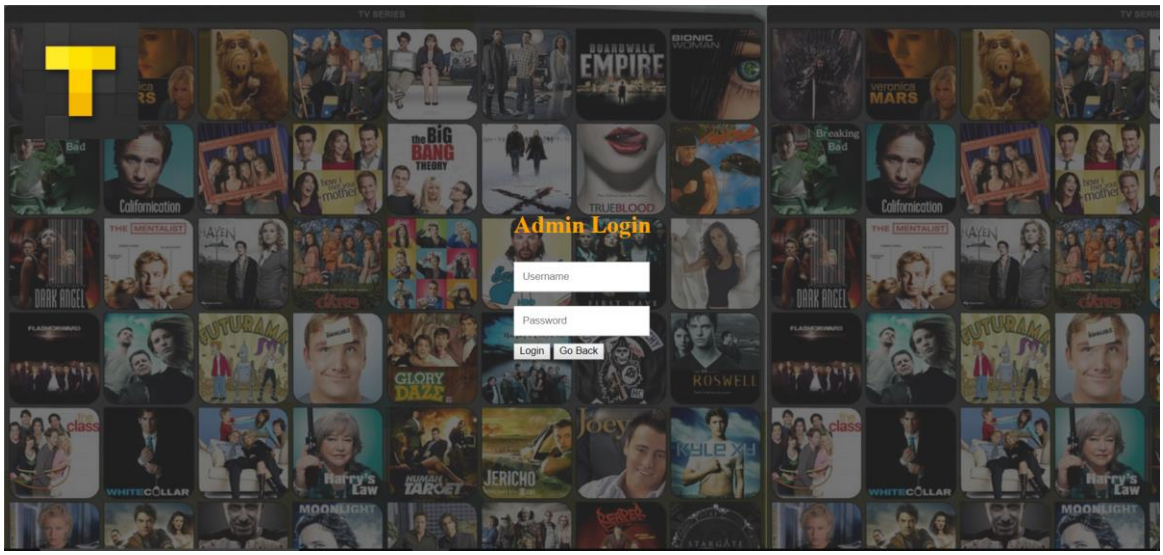



Figure 7.3 – Admin Login

## 7.4 Dashboard

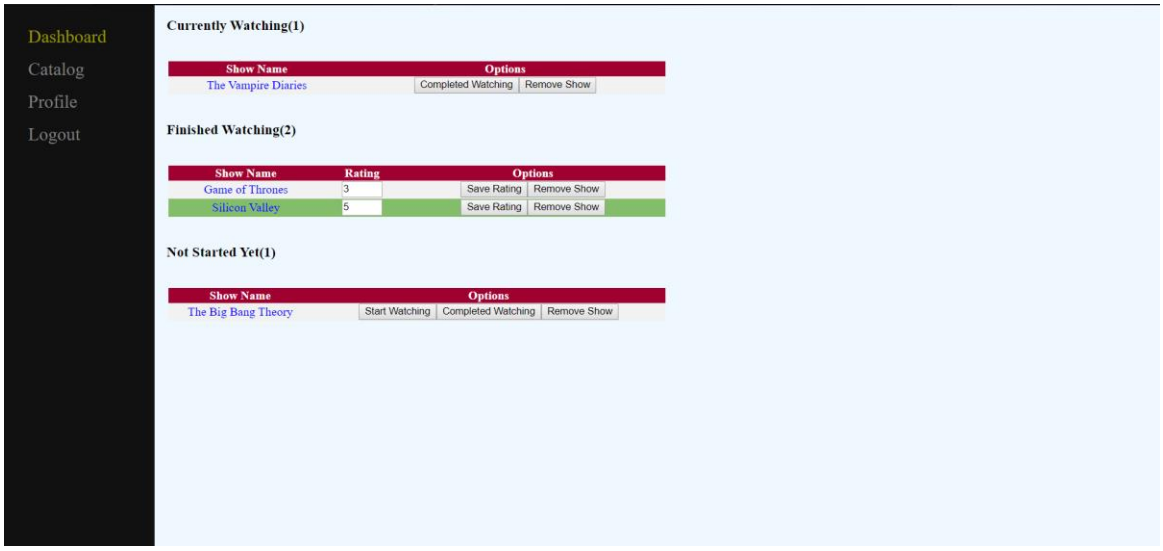



Figure 7.4 – Dashboard

## 7.5 User Catalog

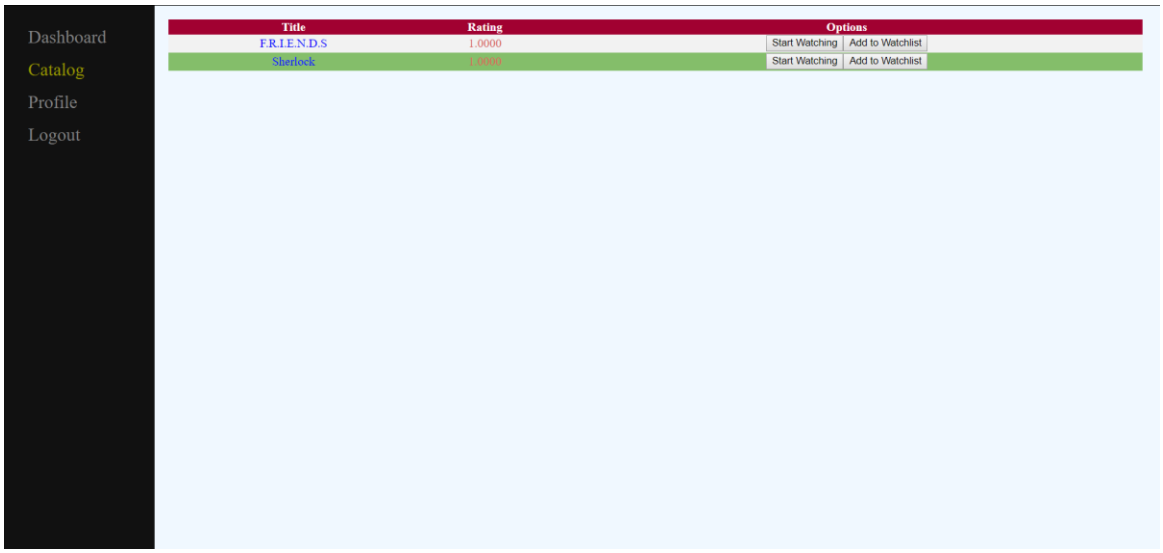



Figure 7.5 – User Catalog

## 7.6 Profile


Figure 7.6 – Profile

## 7.7 Show Details


Figure 7.7 – Show Details

## 7.8 Users (Admin)

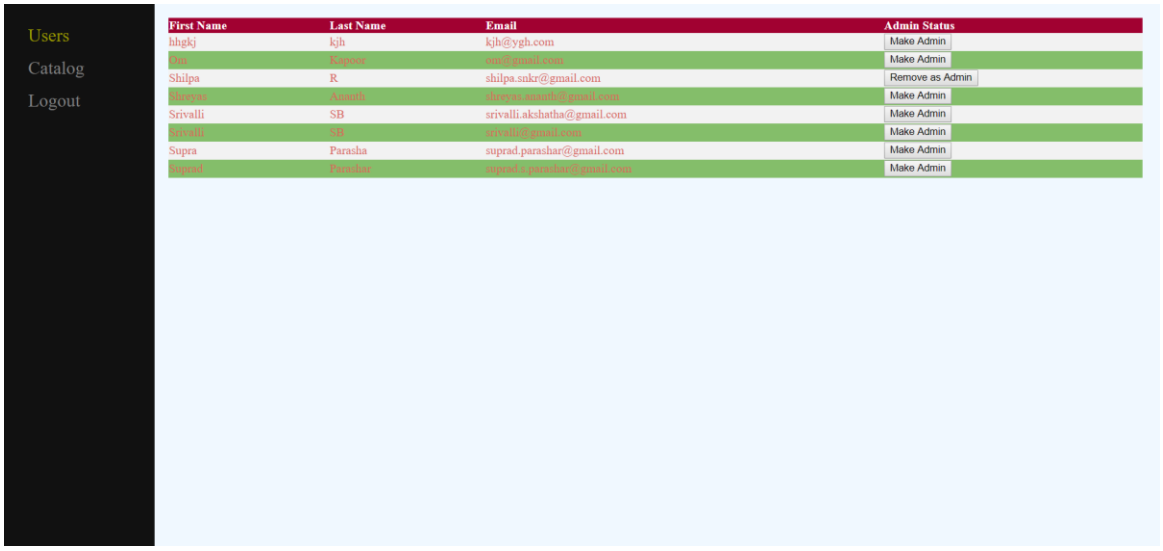



Figure 7.8 – Users (Admin)

## 7.9 Catalog (Admin)

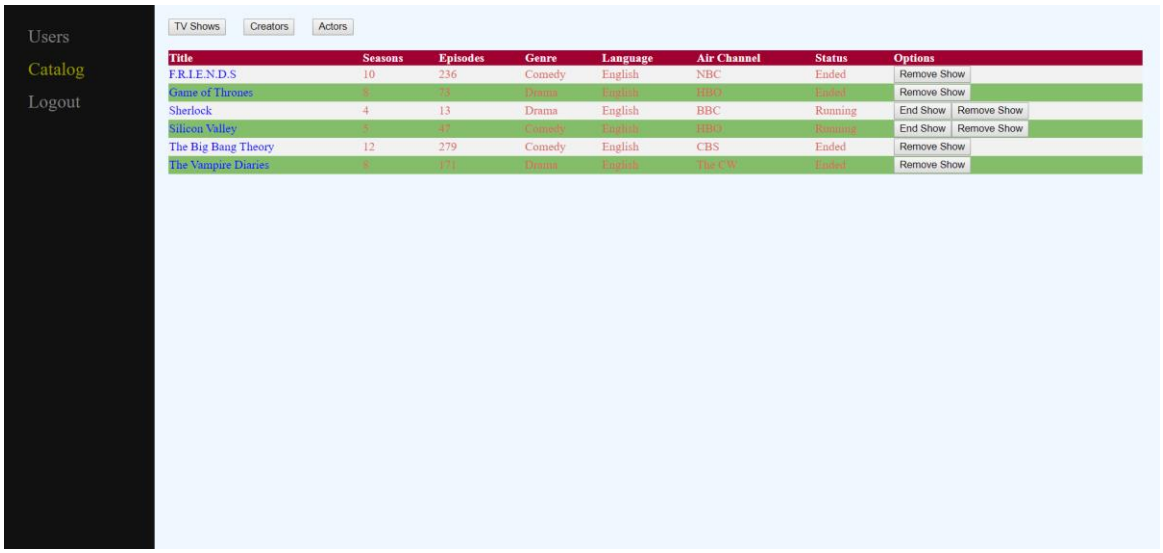



Figure 7.9 – Catalog (Admin)

## 7.10 Add TV Shows (Admin)

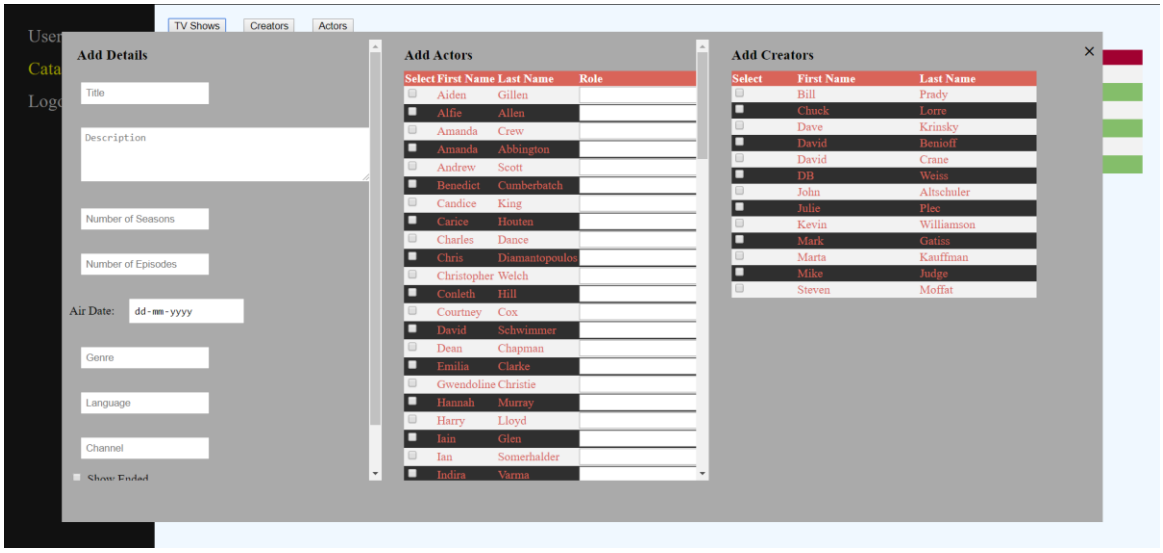



Figure 7.10 – Add TV Shows (Admin)

# 7.11 Add Creators (Admin)

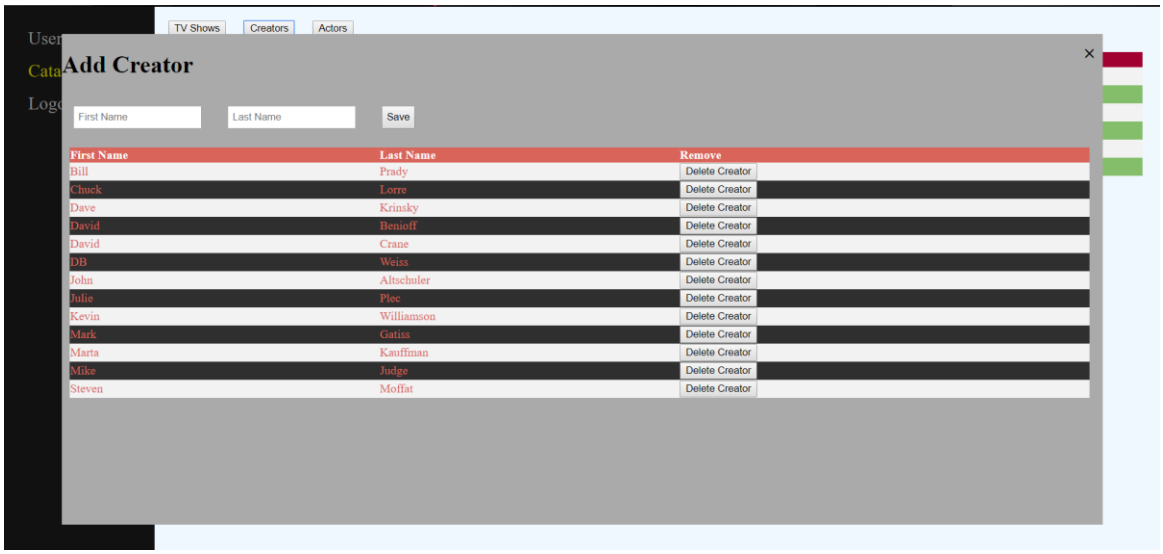



Figure 7.11 – Add Creators (Admin)

# 7.12 Add Actors (Admin)

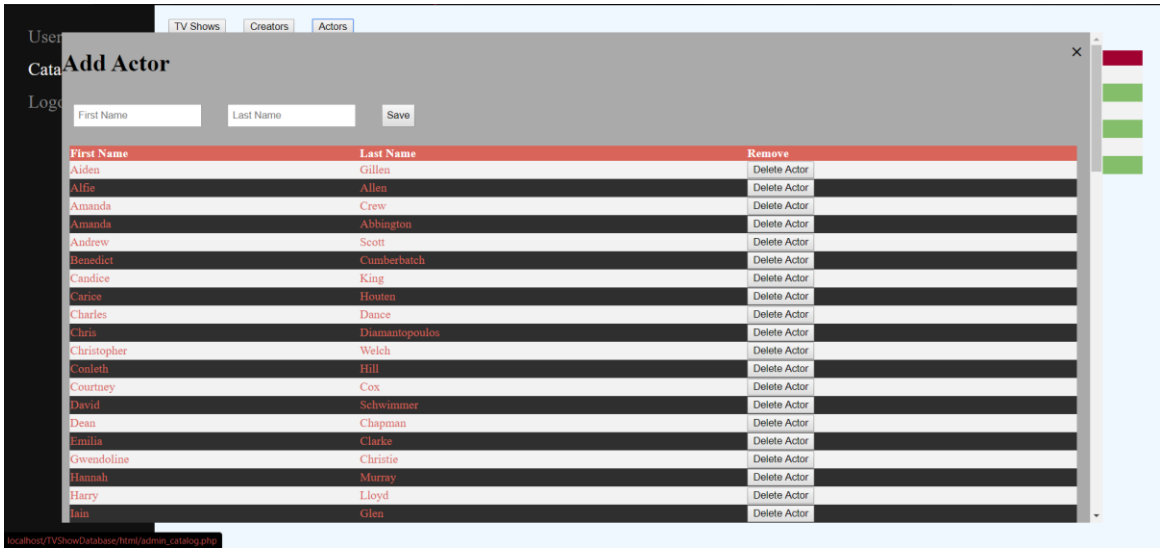



Figure 7.12 – Add Actors (Admin)

# 7.13 Edit Show (Admin)


Figure 7.13 – Edit Show (Admin)

# Chapter 8

# Conclusion and Future Enhancements

## 8.1 Conclusion

Let's Watch aims to help users track their Television shows across all channels and help them choose the best Television Show to binge on next. Let's Watch provides the aggregate rating of other users of a Television show to help the user make his/her decision about the next Television Show to binge on. Let's watch provides a clean and easy interface to users and admins to use the webpage.

## 8.2 Future Enhancements

1. Make the site more user-friendly and assistive by improving the design of the site.
2. Making our website responsive so that customers can use our site from devices with different screen resolution.
3. Add Television Shows every week to be updated with the latest and the greatest shows.
4. Add Session tracking using cookies.
5. Provide better security to the webpage to prevent stealing of user data.

## Chapter 9

# References

[1] W3Schools [https://www.w3schools.com/]

[2] Stack Overflow [https://stackoverflow.com/]

[3] Ramez Elmasri, Shamkant B. Navathe, Fundamentals of Database Systems, 7th Edition, Pearson

[4] TV Time [https://www.tvtime.com/]

[5] Lynn Beighley, Michael Morrison, Head First PHP and MySQL, 2nd Edition, O'Reilly