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

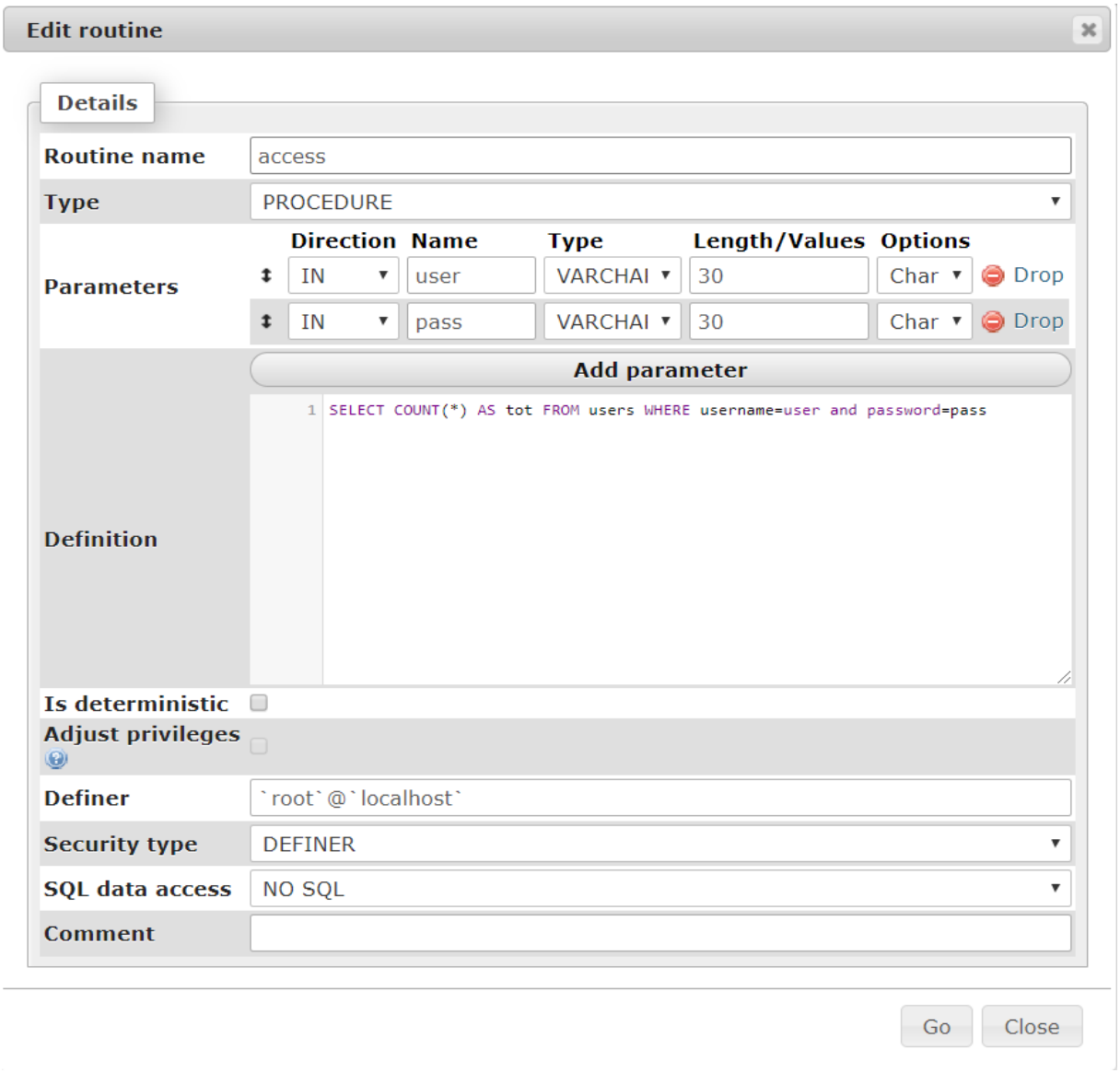

Figure 5.8 – Stored Procedure

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
```