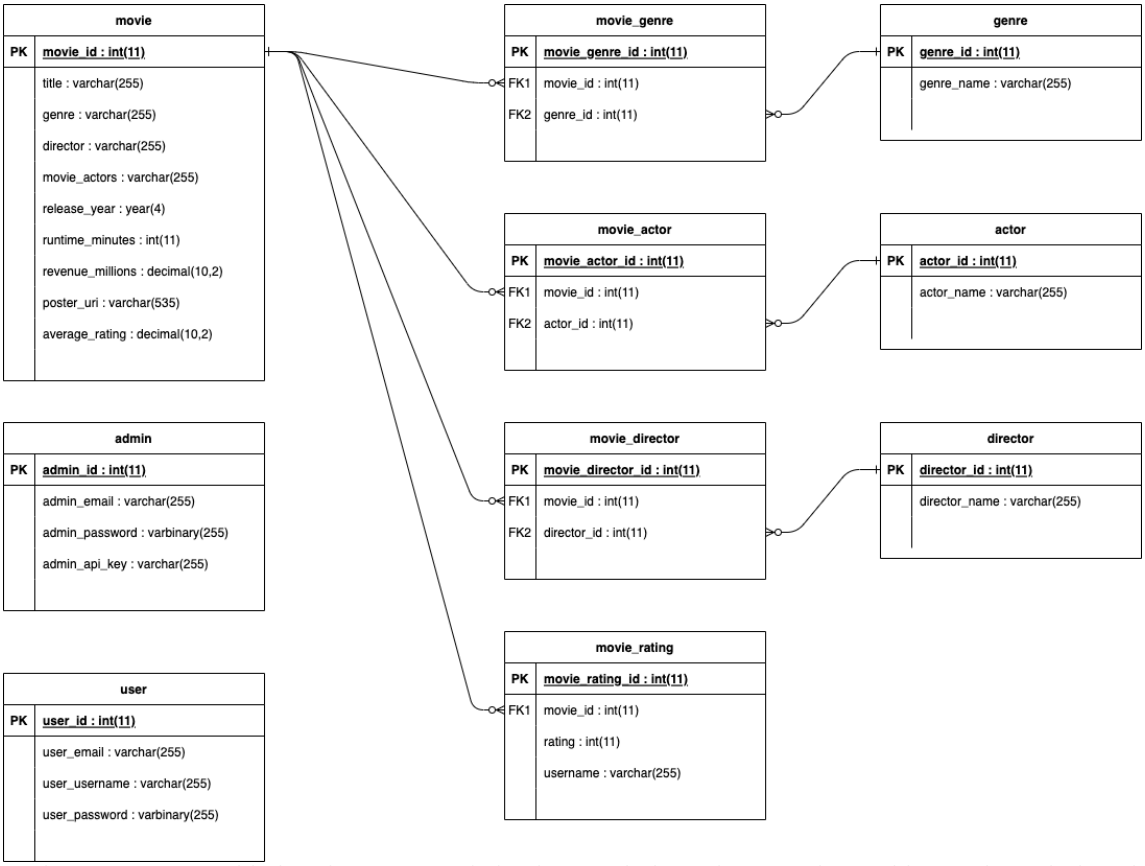
CSC7062 Web Development BingeSpark Project Report

Oisín Carlin (40201120)

This short report will explain the design and implementation of my BingeSpark Web Project and provide URIs and credentials for accessing it on the QUB web-hosted server.

Database Design and ER diagram.



When movies are added to the BingeSpark database including when an admin adds one through the admin (addmovie.php) page with the REST API, or CSV, all movie details are added to the 'movie' table. Collections of genres, actors and directors are added to the table as String lists separated by commas. They are then individually separated into dedicated tables ensuring no duplicates and associated with the movie through conjoining many-to-many tables taking foreign keys of movie_id and genre/actor/director_id allowing querying of movies associated with a particular genre/actor/director. The REST API (bingesparkapis) uses the title of the movie to query the OMDB API for an IMDB ID, which is used to query the TMDB API to source a poster URI for a movie and add it to the movie table.

The average_rating for a movie is updated each time the homepage is visited and will have no value when a movie is first added until a review is submitted. The update calculates the average of ratings in the movie_rating table with a particular movie_id stored as a foreign key. A user cannot submit a rating unless they are logged in; once they log in on the index page the username only is stored as a session variable - the user table is not referenced again during normal use of the website's functionality. For security, the user and admin tables are dissociated with the rest of the database. Moreover, the passwords are stored in varbinary format once encrypted using salted hashes with the SHA1 encryption algorithm.

BingeSpark Web-Hosted Project Links

BingeSpark Index Log-in Page:	https://ocarlin04.webhosting5.eeecs.qub.ac.uk/bingespark/index.php
BingeSpark Homepage (auto-assigned guest)	https://ocarlin04.webhosting5.eeecs.qub.ac.uk/bingespark/homepage.php
REST API	https://ocarlin04.webhosting5.eeecs.qub.ac.uk/bingesparkapis/index.php
Video Demonstration	https://web.microsoftstream.com/video/9037df52-e64f-4c6f-878f-acff81a4b64a

Authentication Credentials

Example Log-in of one registered user	oisin.carlin@queens.com	thisisapassword
Example Log-in of a registered admin	carlin.oisin@queens.com	thisisapassword

Project File Guide: (https://ocarlin04.webhosting5.eeecs.qub.ac.uk/bingespark/...[file.php])

adminlogin.php	Contains HTML form posting admin credentials to *adminloginprocess.php* to access admin dashboard. Receives and displays messages from *adminloginprocess.php* if inputted credentials contain incorrect password or no registered admin.
adminloginprocess.php	Verifies inputted credentials from *adminlogin.php.* Collects salt used to encrypt password on admin associated with email address and salted-hashed password stored. Uses salt to hash password inputted by user and compares that salted-hash to stored salted-hash – admin verified if values are equal. Redirects to *admin/dashboard.php* if correct storing admin API key as $_SESSION or *adminlogin.php* if incorrect (Easy Tutorials, 2018).
db.php	MySQLi connection to database. Included in all files where connection required.
deleterating.php	Uses $_SESSION username and $_GET movie_id from *movieinfo.php* to delete a movie from the movie_rating table in DB. Redirects back to *movieinfo.php*
homepage.php	Main homepage of website to browse movies. Displays username/guest stored as $_SESSION (Easy Tutorials, 2018). Buttons allow sorting of movies by changing SELECT statement from movie table. While-loop loops through selected movies and outputs details on individual Bootstrap cards. Average ratings calculated and displayed on each card (Mubin, 2015). Search bar allows querying of movies by title, genre, actors, and directors.
incorrectpassword.php	Returned page if user logs in with wrong password. Button to return to *index.php*
index.php	Landing page of BingeSpark. Form provided for credentials for user to log-in, or buttons to sign up page or continue as guest to homepage.
loginprocess.php	Verifies inputted credentials from *index.php.* Collects salt used to encrypt password on user associated with email address and salted-hashed password stored. Uses salt to hash password inputted by user and compares that salted-hash to stored salted-hash – user verified if values are equal. Redirects to *homepage.php* if correct storing username only as $_SESSION, *incorrectpassword.php* if incorrect password or *noregistereduser.php* if no user found with inputted email address (Easy Tutorials, 2018).
logout.php	Directed to this file if user clicks a 'log out' button. Destroys the session and redirects user to *index.php* (Easy Tutorials, 2018).

movieinfo.php	Provides information to user about a movie when they click the 'rate' button on a movie card on homepage. Movie_id passed to a movieinfo page through $_POST and $_GET, then used to query movie table and movie_actor/director/genre tables for details. Average rating calculated (Mubin, 2015). Logged in users can submit a rating and previous ratings listed in a table with rater's username. Guests cannot submit ratings.
noregistereduser.php	Returned page if user logs in with a non-registered email. Button to return to *index.php*
ratingprocess.php	Adds user ratings to movie_rating table in database. Takes $_SESSION username, $_GET movie_id and $_POST rating from a movie's *movieinfo.php* page and redirects to that page.
searchresult.php	Displays search results of movies as Bootstrap cards. Takes $_POSTs of search query and search selection which indicates which SQL SELECT query to use on which tables. E.g. 'search by actor' would SELECT from movie_actor table. Movie details displayed and average rating (Mubin, 2015).
signup.php	If user does not have account they can register using the form on this page, taking in e-mail, username and password. Details $_POST to *signupprocess.php.*
signupduplicateusername.php	Returned page if user signs up with an already registered username. Button to return to *signup.php*
signupprocess.php	Processes sign-up credentials to create registered user and add to user table in DB. Takes $_POST of email, username, and password from *signup.php.* Encrypts user's password using salting and hashing with SHA1 hashing algorithm. Redirects user to *index.php* with URL encoded message stating that account has been created and user can log-in (Iorga, 2012).
signupuserexists.php	Returned page if user signs up with an already registered e-mail. Button to return to *signup.php*

Project File Guide: (https://ocarlin04.webhosting5.eeecs.qub.ac.uk/bingespark/admin...[file.php])

addmovie.php	Form collecting details of a movie for REST API to process and add movie to database tables. Example input provided – admins should enter each actor/director/genre consecutively separated by a comma as in original CSV.
dashboard.php	Admin dashboard, potential to be further developed. Button links to *addmovie.php* form.
logout.php	All other pages on BingeSpark/admin redirect here if admin not logged in and admin API key not stored as $_SESSION. Also, directed to this file if admin clicks a 'log out' button. Destroys the session and redirects user to *index.php* (Easy Tutorials, 2018).
movieadded.php	Returned page after admin completes *addmovie.php* form, states whether addition to database successful. Button links back to *dashboard.php*.
processmovie.php	Collects $_POST information from *addmovie.php* form. Prepare information as arrays and HTTP query to send to REST API endpoint with admin API key which adds data to database.

Project File Guide: (https://ocarlin04.webhosting5.eeecs.qub.ac.uk/bingespark/images...[file.php])

BingeSparkLogo.png	BingeSpark logo linked to and found on all BingeSpark pages.

Project File Guide: (https://ocarlin04.webhosting5.eeecs.qub.ac.uk/bingesparkapis/...[file.php])

index.php	JSON-type secure REST API for adding movie information to database through $_POST. Will not work unless API key received as $_GET can be found in admin table. Inserts received movie information to movie table, including actors/directors/genres as string separated by comma. They are then individually separated into dedicated tables ensuring no duplicates and associated with the movie through conjoining many-to-many tables taking foreign keys of movie_id and genre/actor/director_id

	allowing querying of movies associated with a particular genre/actor/director. Uses received title to query OMDB API for a movie's IMDB ID, using this to query TMDB API for the movie's poster URI which is added to the movie table.

Base Data Entry Code Guide (For initial building of website. Not current source code. SQL Queries commented out to avoid unintended data entry.)

genrerateratings.php	Generates randomised 1-5 ratings for 10 registered users for every movie in database.
normalise	Loops through all movies in movies table and normalises actors, genres, and directors from movie table into dedicated tables ensuring no duplicates and associated with the movie through conjoining many-to-many tables taking foreign keys of movie_id and genre/actor/director_id allowing querying of movies associated with a particular genre/actor/director. Similar process when adding movies using *admin/addmovie.php* and REST API.
runonce.php	Adds movie details from CSV file to movie table, details normalised into separate tables using *normaliseactorandgenre.php.* Empty values in CSV replaced with 'n/a' or 0 (CodePudding, 2020). Some bad data corrected using if-statements (Clarke, 1st April 2022).
updatedposteruri.php	Loops through all movies in movie table and uses title to query OMDB API for a movie's IMDB ID, using this to query TMDB API for the movie's poster URI which is added to the movie table.

Bibliography

Clarke, C., 1st April 2022. *WrongTitlesPHP, WrongYearsPHP (used with permission).* s.l.:s.n.

CodePudding, 2020. *REPLACING EMPTY VALUES IN A CSV FILE USING PHP WHEN INSERTING DATA INTO A MYSQL DATABASE.* [Online]
Available at: https://www.codepudding.com/net/161746.html
[Accessed 19 April 2022].

Easy Tutorials, 2018. *How To Make Login & Registration Form In PHP And MySql, Create SignIn & SignUp Page.* [Online]
Available at: https://www.youtube.com/watch?v=NXAHkqiIepc&ab_channel=EasyTutorials
[Accessed 2nd May 2022].

Iorga, M., 2012. *PHP :passing message along header location.* [Online]
Available at: https://stackoverflow.com/questions/12246102/php-passing-message-along-header-location
[Accessed 4th May 2022].

Mubin, 2015. *How to find average from array in php?.* [Online]
Available at: https://stackoverflow.com/questions/33461430/how-to-find-average-from-array-in-php
[Accessed 4th May 2022].