1. Title, Project Summary, and Group Members

Title: Movie Spotlight

Summary: Our website titled Movie Spotlight aims to improve the experience of people who frequently watch movies as a hobby. The site offers a way for the user to browse a collection of movies. The user can read certain details about a selected movie, like its plot synopsis, as well as reviews of the movie written by other users of the site. If the user decides that they want a reminder to watch a certain movie later, they can add the movie to their queue, which is found under their user profile. After watching the movie, they can move the movie to their watch history list and are encouraged to write a review for the movie. The watch history list serves as a way for users to keep track of the movies that they have seen and reviewed. The user can also view the profiles of the other users on the site.

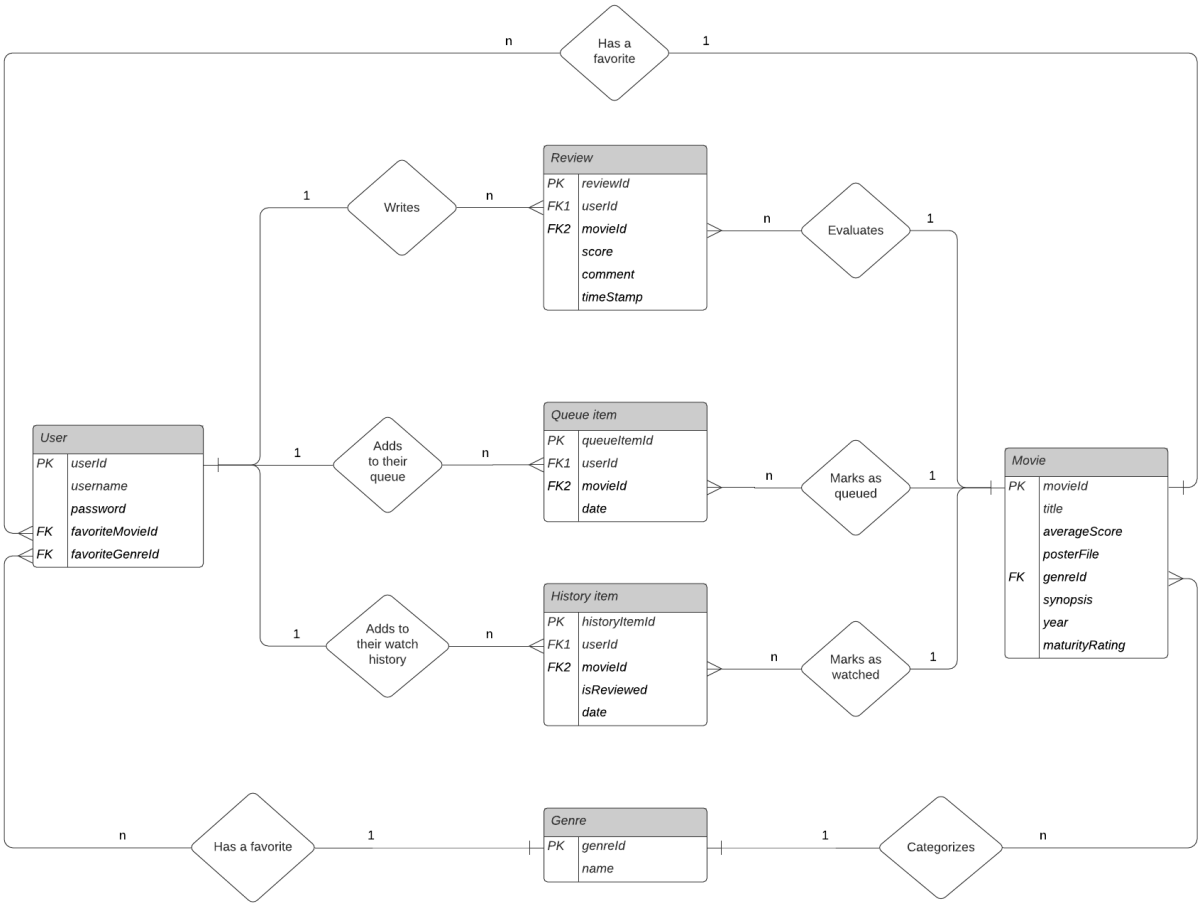
Group Members: Drew Pavlik, Badri Manishankar, Lap Pham, Matthew Funkhouser

2. How the work was divided

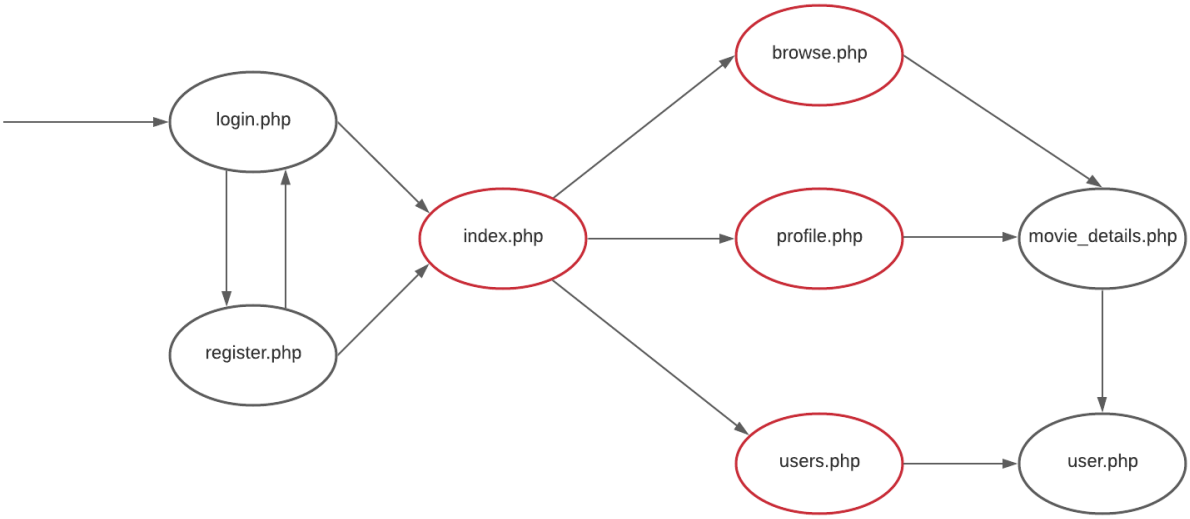
Create design documents (Use Case, ER-Diagram, CRUD Matrix, etc.) **Group Members:** Badri, Drew, Lap, Matthew
Define the basic structure of our web pages with HTML **Group Members:** Drew, Matthew
Create the database and populate the movies table with data for 70 movies **Group Members:** Badri, Drew, Lap, Matthew
Implement logging in with session variables and cookies to keep track if the user is logged in **Group Members:** Matthew, Drew
Dynamically populate web pages with information retrieved from the database about users, movies, and reviews **Group Members:** Drew
Create forms allowing the user to insert, update, or delete certain records in the database, like movie reviews, user profile details, and queue/watch history entries **Group Members:** Drew
Use JavaScript to validate user input on the login, registration, and movie review forms **Group Members:** Badri
Apply a consistent style across all web pages **Group Members:** Lap

3. Use-Case Diagram

4. ER-Diagram


Has a favorite
n
1
Writes
1
n
Review
PK reviewId
FK1 userId
FK2 movieId
score
comment
timeStamp
Evaluates
n
1
User
PK userId
username
password
FK favoriteMovieId
FK favoriteGenreId
Adds to their queue
1
n
Queue item
PK queueItemId
FK1 userId
FK2 movieId
date
Marks as queued
n
1
Movie
PK movieId
title
averageScore
posterFile
FK genreId
synopsis
year
maturityRating
Adds to their watch history
1
n
History item
PK historyItemId
FK1 userId
FK2 movieId
isReviewed
date
Marks as watched
n
1
Has a favorite
n
1
Genre
PK genreId
name
Categorizes
1
n

5. Navigational Chart



Note: The web pages circled in red can be navigated to at any point after logging in via the site's navigation bar. The user must be logged into their account to access any of the pages to the right of login.php and register.php, otherwise they will be redirected to login.php.

6. CRUD Matrix

Tables Forms	User	Movie	Review	Queue item	History item	Genre
login.php	R					
register.php	CR					
index.php	R					
browse.php		R				R
movie_details.php	R	RU	CRUD	CRD	CRD	R
profile.php	U	R	R	RD	CRD	R
users.php	R					
user.php	R	R	R	R	R	R

Note: No forms can be used to create records in the "Movie" and "Genre" tables. The data for these tables was put there in advance by the developers with direct access to the database. Users are not allowed to create movie records because there is no way to guarantee that the information they enter for a movie, like its title and plot synopsis, is correct. And if no new movies are being added by users, then no new genres will need to be created either because the developers already added all the genres that our list of movies requires.

7. CREATE TABLE Statements

Please use the file "OTHER_DELIVERABLES/movie_site.sql" to import our database. It contains all of the necessary statements to create the tables and populate them with required data.

10. External Documentation

Please see the "README.md" file for a guide of how to get the website running, as well as other important information, such as which browsers the site has been tested on.

11. Major problems and how they were resolved

Our first major problem encountered was figuring out how we could share the database among ourselves when XAMPP only offered a local MySQL server. At first, we thought that the database only needed to exist on one of our machines, and the other group members could connect to this person's local server remotely whenever they needed to work with the database. But this meant that the owner of the server would have to have their computer on and database server running whenever anyone wanted to access the database at any given time, which may not always be possible for them. Instead, it was much easier for everyone to maintain their own copy of the database on their own device. Using this method just meant that everyone had to remember to always export their copy of the database to our GitHub repository after making any changes to it. They also had to notify the other members that they made changes to the database and that the other members should import this latest database dump to their local server in order to be working with an up-to-date copy of the database.

None of us had any experience working with PHP and using it to interact with a database. To learn how we should even begin with PHP, we turned to Murach's PHP and MySQL textbook. Chapter 2 gave us a grasp on the basics of PHP, and chapter 4 explained how to use prepared statements in PHP to execute SQL statements. After reading up on prepared statements and learning about their performance and SQL injection security benefits, working with the database from our web application became easy since we already knew how to write SQL commands, and now we had a way of executing them via PHP.

We first tried validating user input for the login and registration pages by filtering out problematic characters one by one in an incredibly long if-statement that looped through the input once for every disallowed character we were checking for. This was incredibly tedious and inefficient because we would not be able to list every single obscure Unicode character that we did not want to be included in the input when there are potentially hundreds of these characters. We solved this by learning about how to use regular expressions in JavaScript, which gave us a way to concisely specify exactly what kind of input should be accepted or rejected.

12. Lessons Learned

We did not make a schedule for ourselves until Group Assignment #4 was assigned in late October. We wish that we had made a schedule sooner because it helped us stay on pace to complete the project in time by giving us deadlines for when specific features would need to be completed. The schedule made it easy to tell if we were ever falling behind and how much work we still had to do. It also eliminated confusion between group members by outlining the tasks that each group member would be responsible for. When we are working on future programming projects in a team setting, coming up with a schedule that breaks up the project into smaller parts with specific deadlines should be one of the first things that we do.

On a similar note, we learned just how helpful it is to spend a lot of time creating design documents for an application before actually starting work on the implementation. If we had not made a use-case diagram and an ER diagram to carefully plan out all the features of the site and the structure of the database required to support these features, we just know that we would have constantly been making changes to the database and to our PHP scripts as we operated without a clear picture of what we wanted our site to be. By putting a lot of thought into the design and knowing the concrete details of how the site should behave, this made the actual implementation so much easier since everyone knew exactly what to implement.