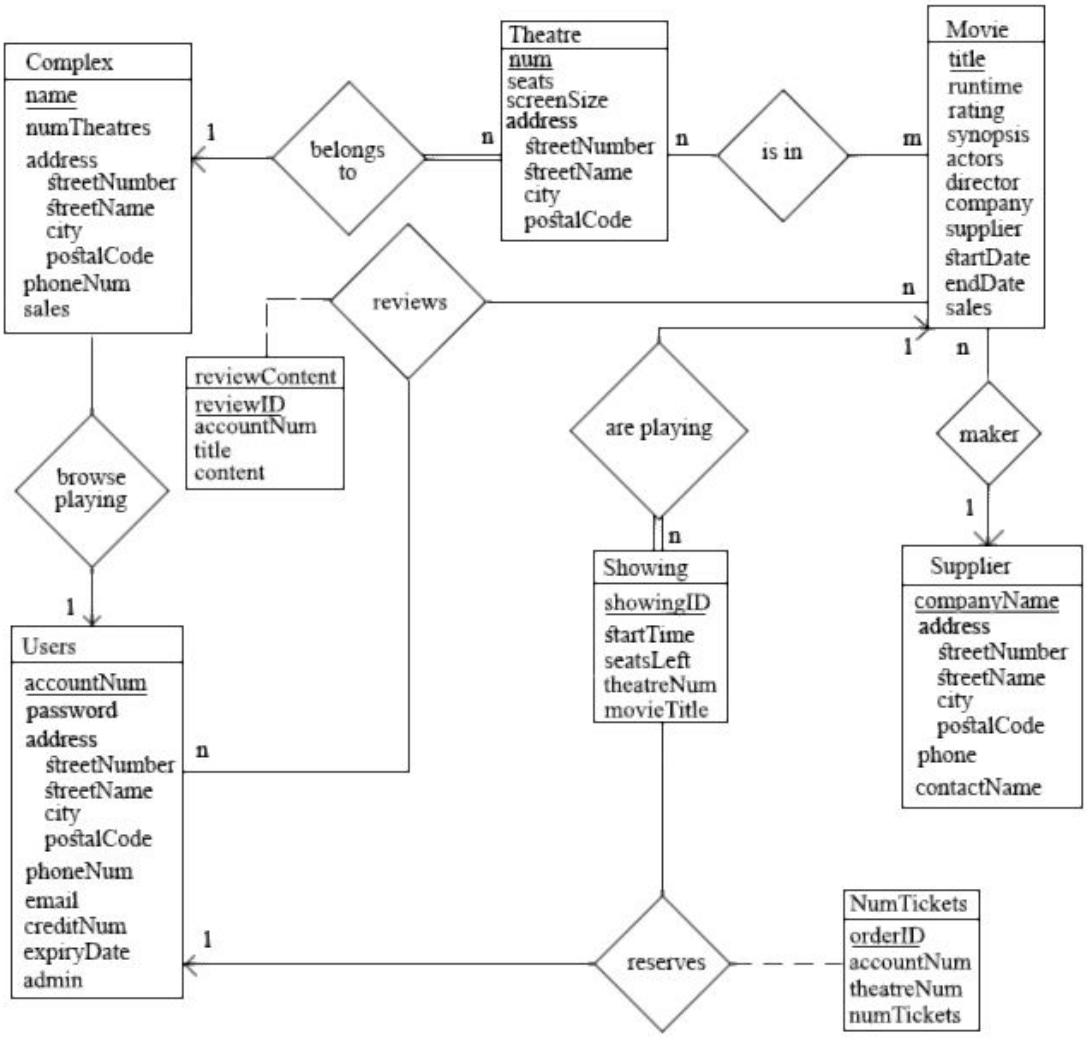
CISC-332 Project - Group 69
Deliverable 2: Application Demonstration / Final Report

Assumptions

- Every user must either login or register with the service before viewing any movies therefore first page user interacts with is a Login/Sign Up Page.
- Assume the user will always input the correct information. Therefore, we had minimal checking for proper input but if incorrect information is inputted user is notified.
- Assume the administrator knows the information needed to delete user and add/update movies, showings, theatres, and complexes. For example, while adding a movie, administrator already knows the existing suppliers.
- Assume that once user purchases a movie, no ticket is to be provided but can be viewed in “View Purchases”.
- Assume that a user can also purchase tickets for a movie that will be coming out soon.
- Assume that the user does not have to go back and look at their written review as it was not part of the required functionalities. However, we know our “Write a Review” functionality worked as it does appear in our database.
- Assume that the Theatre entity needs the address attributes as it was one of our suggested feedback after Deliverable 1.
- Assume that since the most popular Complex and Movie must be computed, the easiest way to implement this was to have a sales attribute within a Complex and Movie that tracks and counts the amount of tickets sold.
- Assume that the user does not need to create a username that associates with their account as their email acts as their unique username.
- Assume that the entities Showing, NumTickets, and ReviewContent need unique primary keys therefore an ID attribute is added.
- Assume that we do not have to provide sample output from queries as Professor stated.

Final ER Diagram



Final Relational Schema

Please refer to the attached .dml file for the complete and final relational schema.

SQL Interactions with Database

Below you will find the queries we had in our php code for the various functionalities in order to interact with our database. Keep in mind that these are only the query lines in the code to give you a brief overview of the queries we made.

Login Query:

```
$query="SELECT * FROM `Users` WHERE email='$email' AND password='$password'";
//where $email and $password were variables that held the input from the user and
we had an if statement that checked if admin flag was set to 1 for admin privileges
```

Sign Up Query:

```
$query1 = "SELECT accountNum FROM users ORDER BY accountNum DESC LIMIT 1";
$accountNum = ++$result['accountNum']; //increments new accountNum by 1
```

```
$query2 = "INSERT INTO users Values ('$accountNum', '$password', '$streetNumber',
'$streetName', '$city', '$postalCode', '$phoneNum', '$email', '$creditNum',
'$expiryDate', '0')";
```

Browse Movies

```
//currently playing query
$today = date("Y-m-d");
$query1 = "SELECT * FROM Movie WHERE (endDate > '$today' AND startDate <= '$today')
ORDER BY startDate DESC";
//coming soon query
$query2 = "SELECT * FROM Movie WHERE (startDate >'$today') ORDER BY startDate
DESC";
//previously played query
$query3 = "SELECT * FROM Movie WHERE (endDate < '$today') ORDER BY startDate DESC";
```

Purchase Movie Tickets

```
//lists all the currently playing/coming soon movies
$query1 = "SELECT * FROM Movie WHERE (endDate > '$today') ORDER BY startDate DESC";
//lists all the theatres playing the movie user selected
 $query2 = "SELECT complexName FROM Theatre JOIN
            (SELECT theatreNum FROM Showing WHERE movieTitle='$movie') AS T
             ON Theatre.num = T.theatreNum";
//lists all available times based on movie and theatre user selected
$query = "SELECT startTime FROM Showing WHERE movieTitle='$movie'";
```

Cancel Purchase

```
$query = "SELECT orderID FROM NumTickets WHERE orderID='$orderNum'";
$query1 = "DELETE FROM NumTickets WHERE orderID='$orderNum'";
```

Browse Previous Purchases

```
$query = "SELECT accountNum FROM Users WHERE email='$email'";
$query1 = "SELECT * FROM NumTickets WHERE accountNum='$account'";
$query2 = "SELECT movieTitle FROM Showing WHERE Showing.theatreNum = $num";
$query3 = "SELECT endDate FROM Movie WHERE title='$movieTitle'";
$query4 = "SELECT complexName FROM Theatre WHERE num='$num'";
```

Write a Review

```
$query1 = "SELECT `reviewID` FROM ReviewContent ORDER BY reviewID DESC LIMIT 1";
$query2 = "SELECT accountNum FROM Users WHERE email='$email'";
$query3 = "INSERT INTO ReviewContent VALUES ('$id', '$account', '$title',
'$content')";
```

Edit Profile

```
//checked when user input var is not null, update that specific attribute
$query = "SELECT accountNum FROM Users WHERE email='$newEmail'";
$query1 = "UPDATE Users SET email = '$newEmail' WHERE accountNum='$accountNum'";
$query2 = "UPDATE Users SET password ='$newPassword' WHERE
accountNum='$accountNum'";
$query3 = "UPDATE Users SET streetNumber='$newStreetNumber' WHERE
accountNum='$accountNum'";
$query3 = "UPDATE Users SET streetName ='$newStreetName' WHERE
accountNum='$accountNum'";
```

```
$updateStreetName = mysqli_query($conn, $query3) or die(mysqli_error($conn));
$query4 = "UPDATE Users SET phoneNum ='$newPhoneNum' WHERE
accountNum='$accountNum'";
$query5 = "UPDATE Users SET city='$newCity' WHERE accountNum='$accountNum'";
$query6 = "UPDATE Users SET postalCode ='$newPostalCode' WHERE
accountNum='$accountNum'";
$query7 = "UPDATE Users SET creditNum ='$newCreditNum' WHERE
accountNum='$accountNum'";
$query8 = "UPDATE Users SET expiryDate ='$newExpiryDate' WHERE
accountNum='$accountNum'";
```

List Members

```
$result = $conn->query("SELECT * FROM `users`");
```

Remove Members

```
$query = "DELETE FROM users WHERE accountNum = '$accountNum'";
```

Add/Update Theatre Info (Complex has same idea)

```
//Similar structure to "Edit Profile" where it updates the attribute when the user
input var is not NULL
$query1 = "UPDATE theatre SET seats ='$newSeats' where num = '$num'";
$query2 = "UPDATE theatre SET streetNumber='$newStreetNumber' where num = '$num'";
$query3 = "UPDATE theatre SET streetName ='$newStreetName' where num = '$num'";
$query4 = "UPDATE theatre SET screenSize ='$newScreenSize' where num = '$num'";
$query5 = "UPDATE theatre SET city='$newCity' where num = '$num'";
$query6 = "UPDATE theatre SET postalCode ='$newPostalCode' where num = '$num'";
```

Add Movies

```
$query2 = "INSERT INTO movie Values ('$title', '$runtime', '$rating', '$synopsis',
'$director', '$company', '$supplier', '$startDate', '$endDate', '$sales')";
```

Update Showings

```
//again similar structure to "Edit Profile" where it updates the attribute when the
user input var is not NULL
$query1 = "UPDATE showing SET theatreNum ='$newTheatreNum' where showingID ='$id'";
$query2 = "UPDATE showing SET movieTitle='$newMovieTitle' where showingID = '$id'";
$query3 = "UPDATE showing SET startTime ='$newStartTime' where showingID = '$id'";
$query4 = "UPDATE showing SET seatsLeft ='$newSeatsLeft' where showingID = '$id'";
```

Show Rental History

```
$query1 = "SELECT * FROM  numTickets WHERE accountNum = '$accountNum'";
$query = "SELECT movieTitle FROM Showing WHERE Showing.theatreNum = $num";
```

Most Popular Movie (Complex has same idea)

```
$result = $conn->query("SELECT MAX(sales) as maxSales FROM `movie`");
$title = $conn->query("SELECT title from movie where sales = $sales");
```

Discussion

While designing and implementing the website for the Online Movie Ticket System, we knew that there will need to be many web pages developed in order to differentiate between the User and Administrator as well as their functionalities. The first point of contact the user encounters with the OMTS is the Login / Sign Up page. This page prompts the user to Login with their email and password. If the credentials used to log in are not valid, the user would be prompted about it and must fill out the Sign Up form.

Once they login, depending on the admin flag used in the database, it will direct the user to the appropriate page. We decided that the best approach to implement these pages is to create a simple, concise, and elegant home page that would list all the functionalities the member/admin can do and link the corresponding pages. The design of the home page is kept simple by using a box-grid system, where each box listed is a functionality. In addition, depending on the login credentials entered, the member/admin will see different amounts of boxes indicating their ability within the OMTS. When listing information to the user such as in browsing movies or listing members, we grouped related info together and separated it from others with spacing. For example, when listing members, each user is divided into its' own block of text based on their account number (primary key). When prompting user for input, we used a form that enables the user to input text from what is asked. For example, in "Edit Profile", in order to keep it straight-forward, we listed out all the possible info the user can modify, and they would only enter into the text-boxes they wanted to update. In contrast, for other functionality such as "Purchase Movie", the user must enter into the text-box from one of the listed options.

The web pages were defined through HTML, styled with CSS, and functionality was provided through PHP. These files were written in a text-editor and our .dml file was imported into phpMyAdmin to help us manage the database. We used the XAMPP (SQL/Apache) Servers where all the corresponding files were saved into htdocs. By doing so, we were able to connect to our database while running the website in the browser. To ensure our website is user-friendly, we also used Bootstrap to help us, especially for the home page grid system. Furthermore, we used software such as Photoshop to help us mockup a logo for the web pages and favicon to establish a professional looking website.

We chose to use these technologies and tools since it was what was familiar to us. In class, it was suggested to use XAMPP and following the provided instructions, it was very easy to install and use. As for experience, we both had no prior experience working with databases, and we had no formal knowledge of web development as it was all self taught. In order to ensure that we fully grasped the tools we were using, we asked the course's teaching assistants as well as friends with experience to help guide us. For example, after writing the .dml file, it was not clear how to implement it onto a database system. Therefore, a friend in the course walked us through how to access and use phpMyAdmin, and import our corresponding .dml file. Although there

were many frustrations while developing, we are both very proud and satisfied with our final design and implementation as prior to this project, we had limited knowledge and experience.

While implementing the OMTS, there were many problems that arose since it was still a learning curve for the two of us. We both had experience developing with HTML and CSS but not with PHP, therefore to be able to execute it with a working website took some time. The problems we mainly encountered were debugging issues and minor syntax mistakes. For example, writing HTML code within the PHP code left room for silly mistakes, as it was all confined within an echo. Furthermore, while making queries and accessing the results from it, we either forgot to fetch the result into a variable or ran into runtime issues as the result was an object, and we forgot to fetch the object's contents through iteration. Lastly, a problem we faced was that between the two of us, we both developed on different environments. For example, we divided the work by Member and Administrator functionalities. While putting the system all together onto one computer, we saw styling differences since one of us developed on a Mac and the other PC. We both also had different screen resolutions, and saw minor differences between the two outcomes, in which we had to make small styling fixes before the final demo.

If we had the chance to go back and do it differently, we would add more realistic functionalities for the Members and Administrators. For example, with the listing of all the movies, we could attach a movie poster and the corresponding written reviews for the movie to give the user more info and visual aspects. In addition, while browsing and purchasing a movie, allow the user to be able to search by either the Movie or Complex as this would be a more realistic and user-friendly approach. Furthermore, we would make the login page more dynamic and interesting by using a "carousel" of movie posters available in bootstrap. With these suggested changes, we believe our Online Movie Ticket System will enhance the user's experience with our system, and resemble more likely to a real-world movie ticket system.

User's Guide

Shown below.