University of British Columbia, Department of Computer Science

CPSC 304

Winter 2016 Term 2

Project Part 4 (Completed Project)

Group Members (Team 34):

Name	Student Number	Unix ID	Email Address
Hyojin Yi (Ginnie)	32888125	x6w9a	hyojinyi@live.ca
Mike Yoon	14334149	x2v9a	mikeyoon@hotmail.ca
Sean Lennaerts	54179130	k4c9	seanlennaerts@me.com
Jeongwoo Park (David)	55148126	p5q8	jwpark511@gmail.com

By typing our names and student numbers in the above table, we certify that the work in the attached assignment was performed solely by those whose names and student IDs are included above.

In addition, we indicate that we are fully aware of the rules and consequences of plagiarism, as set forth by the Department of Computer Science and the University of British Columbia.

Project: GTheatre Accomplishments

This project has accomplished connecting a database to an application that can solve a real-life problem. This project creates a website that a movie theatre can use to sell tickets. The website offers multiple user options such as an Employee of the movie theatre or a Customer whose purpose is to buy tickets to a movie.

These different user types have different capabilities within the website. Customers are allowed to buy multiple tickets to a movie while bundling it with a food option. While employees are allowed to edit movie rating, movie released year, movie runtime and are allowed to remove a movie as movies become outdated and removed from the theatre which then is removed from the database. Also allows the Employee to add new movies as new movies are released.

This project also offers to show the top 5 movies available (the top 5 most bought movies). Also allows the Customers and Employees to filter through the current movies playing which is done with sql queries in the backend, allowing easier access to the wanted movie.

SQL Queries

Homepage - Hyojin

This is the initial start off point of the web application which shows certain links to pages depending on whether or not the user is logged in as a customer, employee or not at all. If the user is not signed in, the Login button and the Movies button are shown. If the user is signed in as a customer or employee, in addition to the Login and Movie buttons, an account button is shown with their name. Regardless of who's signed into the website a Top 5 movies banner that is dynamically changed is shown so users can see what are the most popular movies at the time. The banner will display at most 5 movies depending on how many people bought tickets to the movies and how many movies are in the database.

Render top 5 movies query:

```
SELECT Title, sum(Qty) as Total FROM associated_tickets
GROUP BY Title ORDER BY sum(Qty) DESC LIMIT 5
```

Login - Hyojin

The login page doubles as a sign-up/register page for new visitors to the web application. For users with accounts already, either Customer accounts or Employee accounts, they can just use their username and password to login. For new users, they can choose to register as a customer to buy tickets or as employees to edit movies.

Check to see if the database has a customer with that login if the user specified in the front-end that they were a customer:

```
SELECT * FROM Customers
WHERE Customer_Login = '$username'
AND Customer_Password = '$password'
```

Check to see if the database has an employee with that login if the user specified in the front-end that they were an employee:

```
SELECT * FROM Employees
WHERE Employee_Login = '$username' AND
Employee_Password = '$password'
```

Make sure that the username that the new customer chose is unique:

```
SELECT * FROM Customers WHERE Customer_Login = '$username'
```

Make sure that the username that the new employee chose is unique:

```
SELECT * FROM Employees WHERE Employee_Login = '$username'
```

Add a new customer:

```
INSERT INTO Customers(CreditCard, Customer_Login,
Customer_Password, FirstName)VALUE ('$credit', '$username',
'$password', '$name')
```

Add a new employee:

```
INSERT INTO Employees(SIN, Employee_Login, Employee_Password,
FirstName) VALUE ('$sinno', '$username', '$password', '$name' )
```

Account Page - Hyojin

This page is a page where users can change their personal information as well as see the tickets they have purchased.

Get the data from the database for an employee:

```
SELECT * FROM Employees WHERE Employee_Login = '$u'
```

Get the data from the database for a customer:

```
SELECT * FROM Customers WHERE Customer_Login = '$u'
```

Update all customer information:

```
UPDATE Customers SET FirstName = '$name', Customer_Password =
'$password', CreditCard = '$creditCard' WHERE Customer_Login =
'$username'
```

Update all employee information:

```
UPDATE Customers SET FirstName = '$name', Customer_Password =
'$password', SIN = '$sin' WHERE Employee_Login = '$username'
```

Buy Tickets Page - David

This page displays the general purchasing options. Customers are able to bundle their ticket with the given food options. They also able to choose the quantity of seats if they have companions. The food options are more of a reserve system for employees to see how much of a specific item they would need to have ready. Food prices did not make it into the interface.

Originally we planned to have the ticket number as a foreign key in bundle to reference the ticket, but due to bugs we instead show the start time, title and release year of the movie. Employees are still able to verify the specific movie and start time to match the associated food option to the ticket. Once the user purchases a ticket, it is displayed on the My Tickets Page with the relevant information.

Adding a new food bundle associated with a specific movie to the database:

```
INSERT INTO Bundle values('$ftype', '$title', '$ryear',
'$stime')
```

Adding Tickets to the database each with a unique ticket number:

```
INSERT INTO Associated_Tickets values('$title', '$ryear',
'$ticketno', '$qty', '$creditCard', '$login', '$tprice',
'$stime')
```

My Tickets Page - Hyojin

See all the tickets a user has bought.

Get all the tickets associated with customer username and credit card:

```
SELECT * FROM Plays p JOIN Associated_Tickets t
WHERE t.CreditCard = '$credit'
AND t.Login = '$u' AND t.Title = p.Title AND t.STime = p.STime
```

Showtimes Page - Sean

This page displays all the showtimes to the customer and employee. This includes seeing the title, release year, maturity rating, runtime, showtimes and number of tickets already sold. Both clients can filter the showtimes by title, maturity rating, and a range of release years. If signed in as an employee the client can also add, edit, and delete movies and showtimes.

Employee deleting a movie:

```
DELETE FROM movies
WHERE title = "$movie" AND ryear = "$year";
```

Employee updating a movie:

```
UPDATE movies
SET title = "$newtitle", ryear = "$newryear", mrating =
"$mrating", length = "$length"
WHERE title = "$oldtitle" AND ryear = "$oldryear";
```

Initializing the sold counter:

```
SELECT SUM(qty) AS qty
FROM associated_tickets
WHERE title = "$title" AND ryear = "$ryear" AND stime =
"$stime";
```

And

```
SELECT capacity
FROM theatrehalls
WHERE hnumber = "$hnumber";
```

Employee adding a new movie:

```
INSERT INTO movies
VALUES ("$title", "$ryear", "$mrating", "$length", "$tprice");
```

And for each showtime of that movie:

```
INSERT INTO plays
VALUES ("$start", "$end", "$hnumber", "$title", "$ryear");
```

Populate all movies with showtimes on page load:

```
SELECT m.title, m.ryear, mrating, length, tprice, stime, hnumber
FROM movies m, plays p
WHERE m.title = p.title AND m.ryear = p.ryear
```

```
ORDER BY m.title, stime;
```

Populate all theatre halls for add movie modal:

```
SELECT *
FROM theatrehalls
ORDER BY hnumber;
```

Filtering showtimes with all filters in use:

```
SELECT m.title, m.ryear, mrating, length, tprice, stime, hnumber
FROM movies m, plays p
WHERE m.title = p.title
     AND m.ryear = p.ryear
     AND m.title = "$movie"
     AND (m.rating = "$rating[$i]" OR m.mrating =
     "$rating[$i+1]")
     AND m.ryear >= $year[$min]
     AND m.ryear <= $year[$max]
ORDER BY m.title, stime;
```

Movie Stats Page - Sean/Hyojin

This page displays some statistics about the movies and customers watching the movies.

Who bought at least one ticket to all the movies

```
SELECT * FROM customers c
WHERE NOT EXISTS (SELECT * FROM movies m
WHERE NOT EXISTS (SELECT * FROM associated_tickets t
WHERE t.title=m.title AND t.login=c.customer_login))
```

Most popular movie:

```
SELECT max(AverageNumTicketsPerMovie.avgQty) AS max, stime,
title FROM (SELECT avg(qty) AS avgQty, title, stime
FROM associated_tickets GROUP BY title)AverageNumTicketsPerMovie
```

Least popular movie:

```
SELECT min(AverageNumTicketsPerMovie.avgQty) AS max, stime,
title FROM (SELECT avg(qty) AS avgQty, title, stime
FROM associated_tickets GROUP BY title)AverageNumTicketsPerMovie
```

Functional Dependencies

Attribute to Letter Mapping:

HNumber	A	Login	I
Capacity	B	Password	J
Title	C	FirstName	K
RYear	D	CreditCard	L
MRating	E	SIN	M
Length	F	Qty	N
STime	G	FType	P
ETime	H	FPrice	O

FDs of all Entities and Relationships:

HNumber -> Capacity Given a hall number, determine the capacity of that hall	A -> B
Title, RYear -> MRating, Length Given a movie title and the release year, the rating and length of the movie	CD -> EF
STime, HNumber -> ETime, Title, RYear Given a start time and the hall number of the movie, the end time, title and release year of that movie	AG -> CDH
ETime, HNumber -> STime Title, RYear Given an end time and the hall number of the movie, the start time, title and release year of that movie	AH -> CDG
SIN -> Login, Password, FirstName Given a social insurance number, the login, password and first name of an employee	M -> IJK
CreditCard -> Login, Password, FirstName Given a credit card number, the login, password and first name of a customer	L -> IJK