CMPE 332 | Course Project

Online Movie Ticket Service

Group Number: 53

Robert Saunders, 10194030, 15rws@queensu.ca

Mariano Elia, 10197755, 15mre@queensu.ca

Jack Naylor, 10183805, 14jjn2@queensu.ca

Date: April 6th, 2018

Table of Contents

[Introduction 2](#_Toc510820074)

[Assumptions 2](#_Toc510820075)

[Design Phase 2](#_Toc510820076)

[ER Diagram 3](#_Toc510820077)

[Relational Schemas 4](#_Toc510820078)

[Development Phase 4](#_Toc510820079)

[Development Tools and Technologies 4](#_Toc510820080)

[Interacting with the Database 4](#_Toc510820081)

[Future Considerations 5](#_Toc510820082)

[User Guide 5](#_Toc510820083)

[OTMS Members 6](#_Toc510820084)

[OTMS Admins 9](#_Toc510820085)

[Appendix A 11](#_Toc510820086)

[Appendix B 18](#_Toc510820087)

Table of Figures

[Figure 1 - Entity Relationship Diagram for OTMS 3](#_Toc510814936)

# Introduction

The Online Movie Ticket Service (OMTS), is an application for the advance purchase of movie tickets from any local theatre. Customers use the service to find out information about movies currently playing in their city and to order advance tickets for specific showings of the movies. This report outlines the design, development and implementation details of the OMTS. A detailed list of the projects data and functional requirements can be found on the project repository at: <https://github.com/RobertWSaunders/databases-class-project>

This project has given the team an opportunity to participate in all phases of the development of a database application. The development of the application has required us to use ER modeling techniques, relational modeling techniques, SQL, and to access your database via a web application.

## Assumptions

A comprehensive list of assumptions we made can be seen below:

* Each theatre complex has a unique name
* Each movie has a unique title
* Each supplier company has a unique name
* Only one company can supply a movie
* An account can only review a single movie once
* Every theatre complex has at least one theatre
* The OMTS is only being operated in one country
* Theatre numbers will never change

# Design Phase

During the design phase we focused on simplicity both for how we model our data and how our users interact with the application. Placing an importance on simplicity allowed us to focus on creating a great application rather than overly complex intricacies. When designing how we planned on modeling our data that made sense to someone that wasn’t familiar with complex database systems. As you will see from the diagram below the tables and attribute names are very explicit in what they represent. In addition to well named tables the relationships are defined to be as explicitly as possible as well and to allow for flexibility with regards to the participation of entities (i.e. a customer is able to make multiple reservations. For relations that did not require large amounts of properties we opted to create definitive attributes on the relationships themselves (i.e. movie run, review, reservation) we found this greatly improved readability.

## ER Diagram

Below you can see the entity relationship diagram we designed for this project:

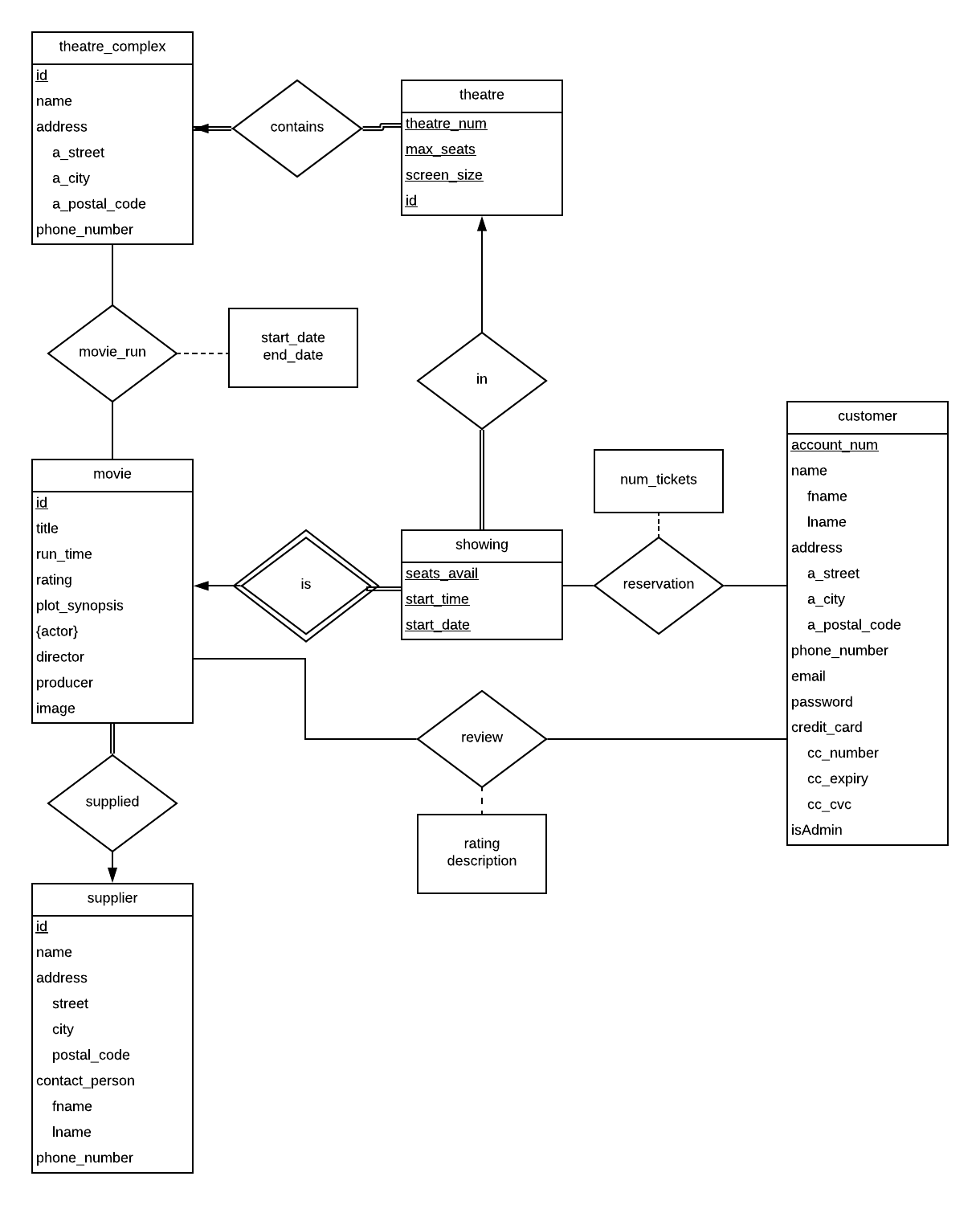


Figure 1 - Entity Relationship Diagram for OTMS

## Relational Schemas

In order to make the entity relationship diagram defined above into an operational database system we utilized SQL and a set of translation rules to create relational schema. Through the use of data definition language and data manipulation language the code in Appendix A will create all tables, relationships and seed data for the OTMS database.

# Development Phase

Having completed a well thought out design for our application it was time to implement it. Throughout the development of the application we made tough decisions, encountered issues, used many tools and interacted with our database right from the code powering the application. Our largest problem encountered with the development of the application was purely inexperience with PHP. As a result of the lack of experience we spent a lot of time trying to figure out how to do basic operations. Once we overcame that and became more proficient with PHP we made good progress and were able to achieve the goals of the project.

## Development Tools and Technologies

Throughout the development of the application we used lots of tools to help us. See the comprehensive list of tools and technologies below:

**PHP –** Programming language of choice.

**SQL –** Querying language used to interface with the database.

**Atom –** The integrated development environment our code was written in.

**MAMP Pro –** Service used to run our Apache server and MySQL database instance.

**PHP Data Object –** Used to interface directly with the database from our code.

Going into the project our team had no experience with any of the tools so the learning and onboarding time was significant, as mentioned in the problems we encountered above.

## Interacting with the Database

As mentioned above, our application uses PHP Data Objects to interact with the database. The actual commands and queries we used in our application to enable the functionality can be found in Appendix B.

## Future Considerations

There are many potential areas of both improvement and extension for our application.

One potential improvement would be the search functionality. Currently, the only way to search for movies and show times is by theatre complex. Some other possible ways to search could be by movie, allowing a user to select the movie they want to view and be provided with the theatre complex’s showing of the film and their times. Another possible search feature could be by show time. Many user’s when searching for movies have a specific time they are able to go, so this would allow users to look for all movies across all complexes by either the start time or over a possible time period. An extension of this function could be including the running time, so a user could state they want to watch a movie sometime between 6:30 and 9:30. This would bring up all of the movies that start and end within the given time period.

Some possible improvements also include providing more information on various movies and showings, such as the audience rating for a movie and the official film rating (e.g. PG-13). Each movie page could also provide a link to the film’s official trailer, so users can watch a sample of the movie. A possible extension to the standard search function would be implementing a map feature. This would allow the user to see their own location on a map and being able to view the location of all theatre complex’s, so they can look at movies at the best location for them instead of having to look up where each complex is separately.

Another extension of the search function is implementing filters. These filters could be used to only see movies of a certain rating, such as only looking at PG movies if the user is planning a family movie night for example. The filters could also be used to only see movies at a certain audience rating level, such as only seeing movies rating 80% or higher. A history extension could also be implemented into the application allowing users to see past movies that have played in theatres. Having access to all past information such as reviews and ratings would be very useful for those who did not manage to see a certain film in theatres and were considering viewing it once it becomes available on a streaming service or DVD.

On the administration side of the application, an improvement would be allowing administrator accounts to view possible customer analytics. Some of these could include number of reservations made, number of overall tickets purchased, and the ratio of reviews given to reservations made, so the admin can know how often a customer will leave a review for a movie. This can be used to implement a rewards program that gives benefits such as discounts to customers active in the reviewing community. Finally, one improvement to the applications security would be protection against SQL injections. This can be done by implementing an input sanitation script in order to ensure the various inputs for the database are safe and do not include any sort of security risks.

# User Guide

In the design phase of the application we focused a lot of our attention to making the user interface as intuitive as possible. We understand that some users may have difficulty using our application and therefore we have provided a guide for both types of our users below. Navigation to the section that applies to what you desire to do and follow the instructions.

## OTMS Members

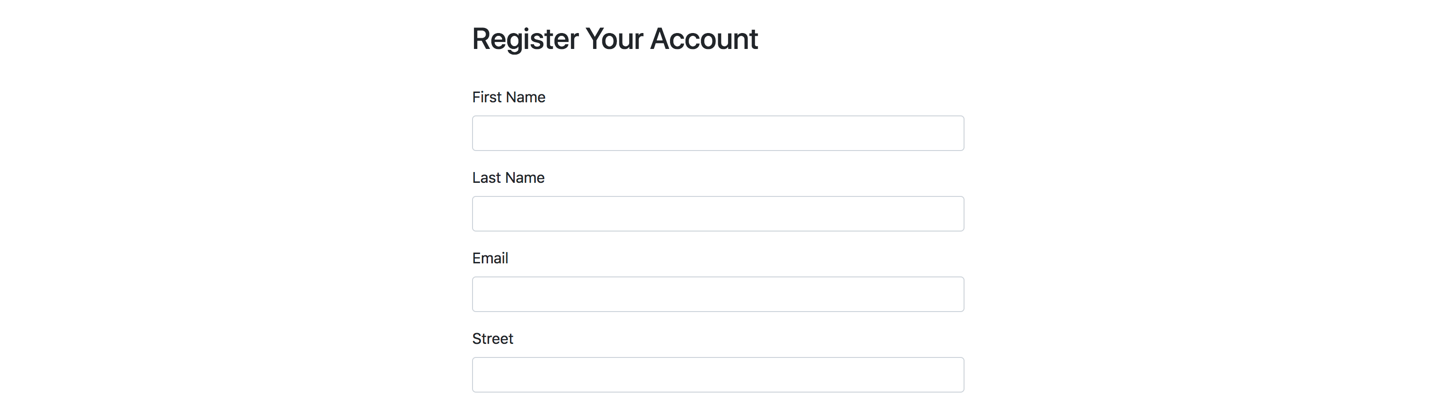
Members have the ability to do the following:

* Make an account including a login email and password
* Browse movies playing at the various theatre complexes
* Purchase some number of tickets for one or more movies showing at one or more theatres
* View their purchases
* Cancel a purchase
* Update their personal details
* Browse their past purchases
* Add a review for a movie

An outline how each of these can be done is below:

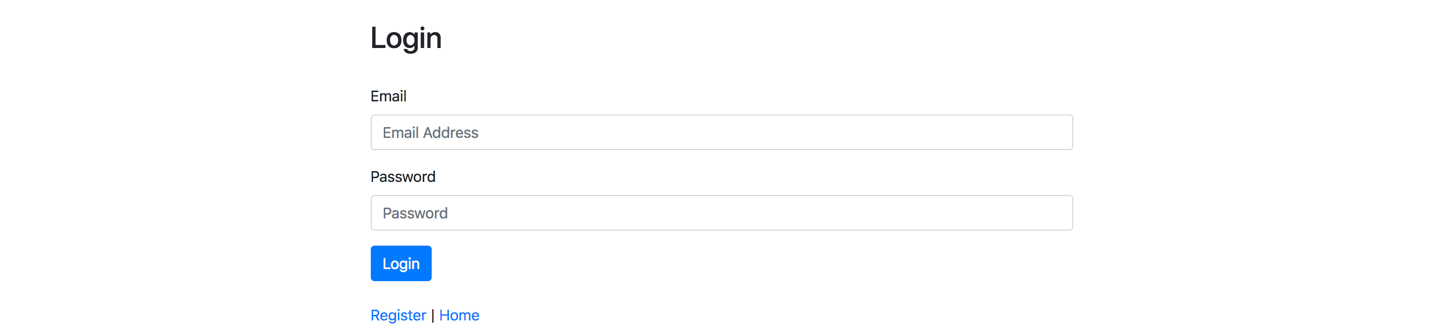
**Making an Account**

A user can make an account by navigating to the register page from the homepage, see below:



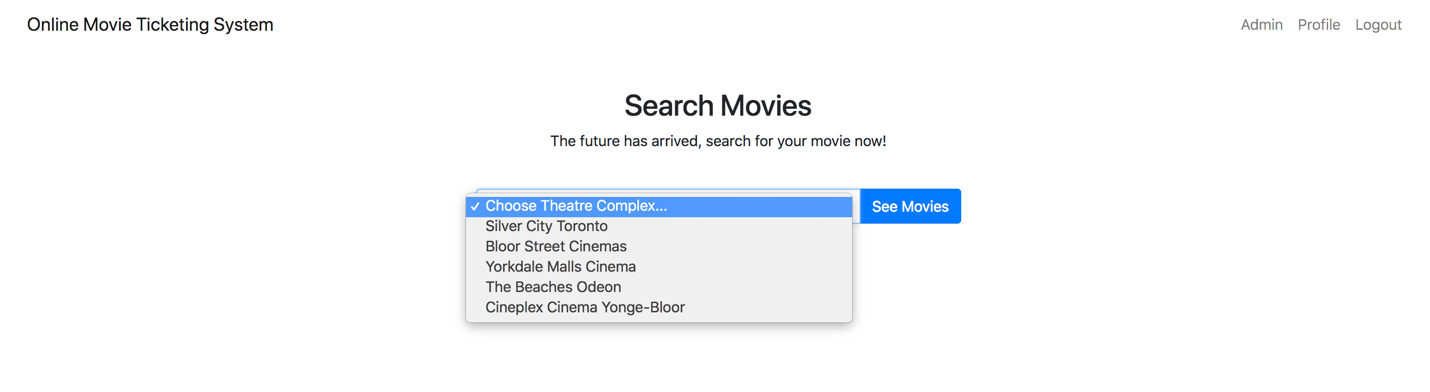
**Login to Account**

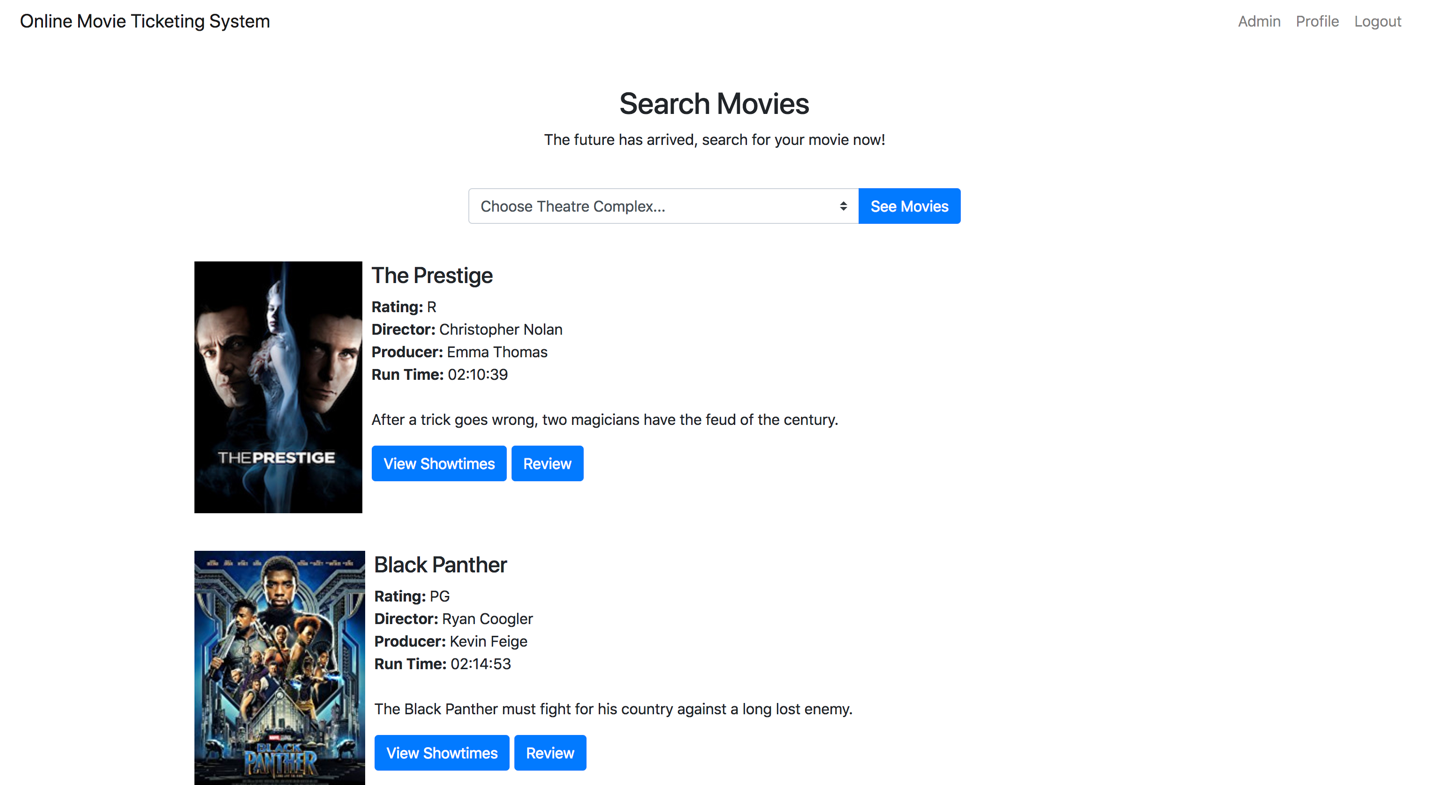
A user can login into their account by navigating to the login page from the homepage, see below:



**Browse Movies**

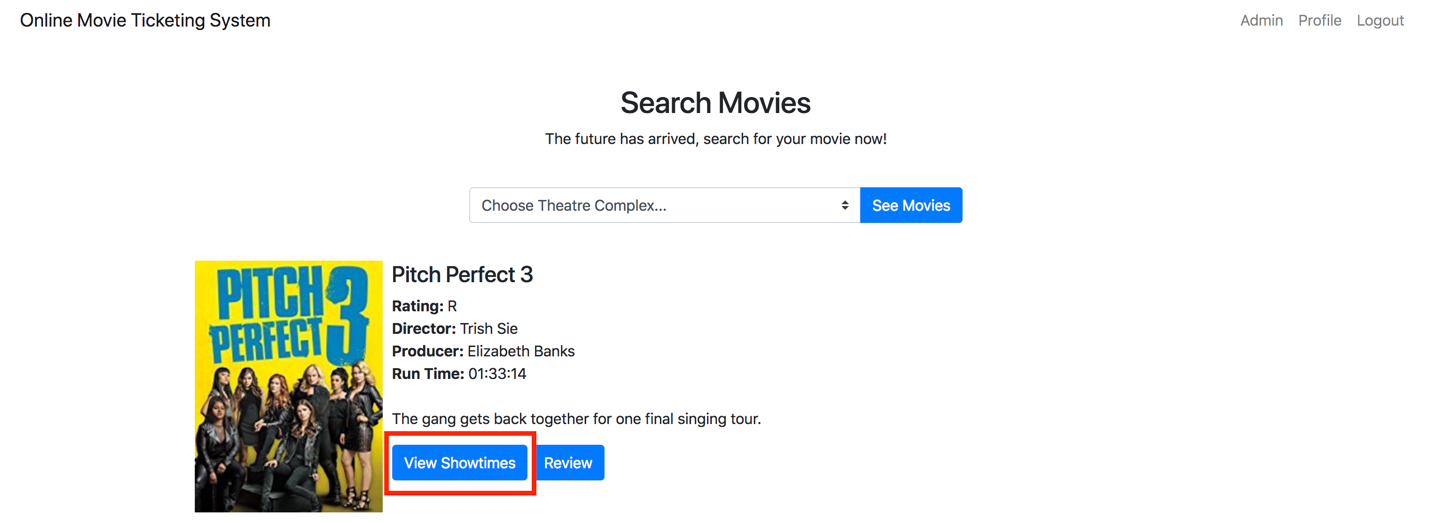
A user can browse movies by first selecting a theatre complex and clicking “See Movies”. This is an authenticated action so the user must be logged into the application as outlined above. See below:





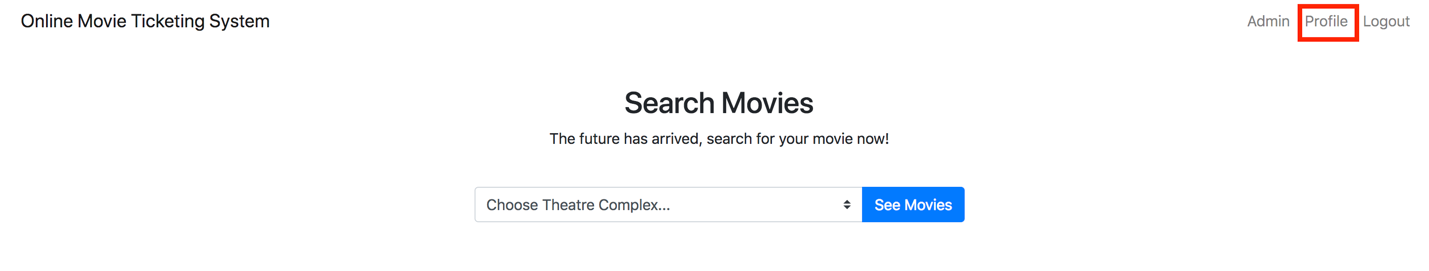
**Purchase Tickets**

A user can purchase tickets for a movie by selecting ‘View Showtimes” on a movie from the movie list and clicking “Reserve”. This is an authenticated action so the user must be logged into the application as outlined above.

****

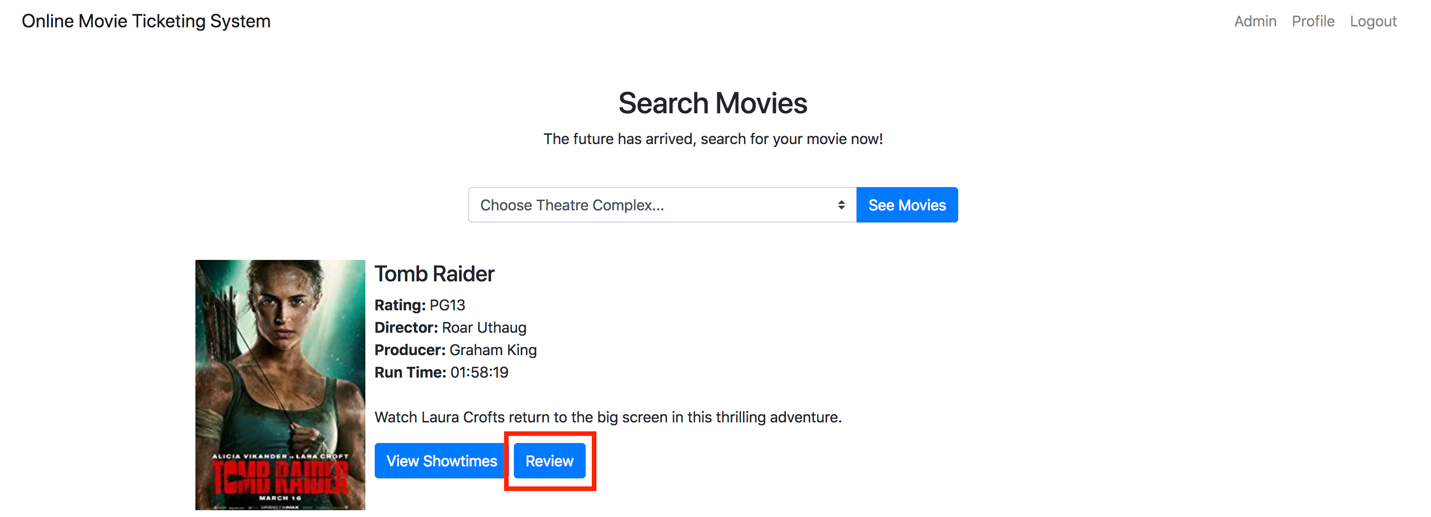
**Browse/View/Update Reservations and Update Profile**

A user can browse, view and update reservations from their profile page. From within the application you can navigate to your profile by selecting the menu item in the top right of the screen. In addition to viewing reservations user can also update their profile. This is an authenticated action so the user must be logged into the application as outlined above.

****

**Review a Movie**

A user can review a movie by selecting “Review” next to a movie from the movie list, see below. This is an authenticated action so the user must be logged into the application as outlined above.

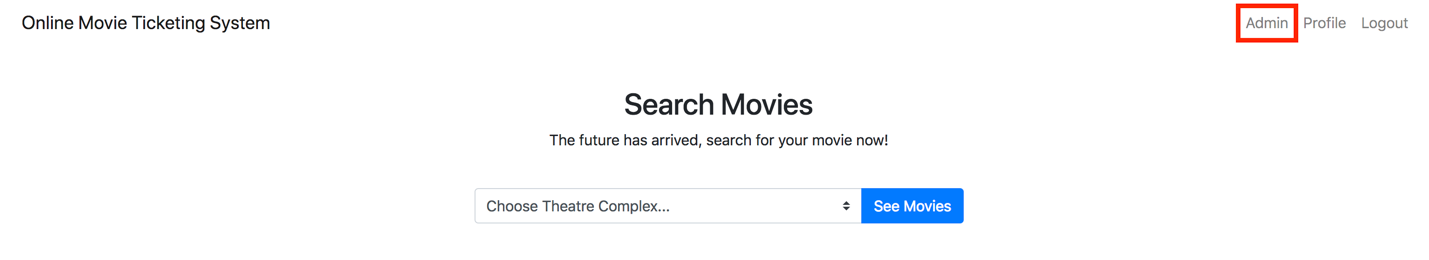


## OTMS Admins

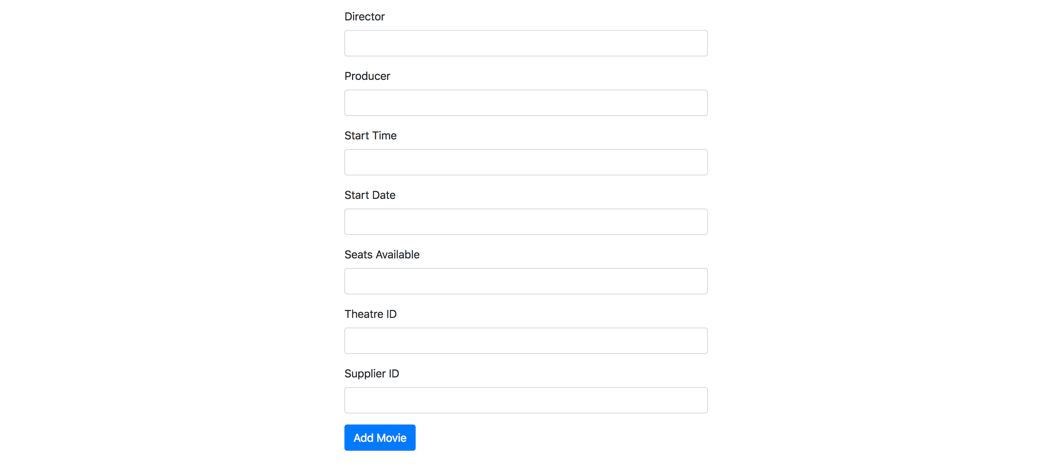
Admins have the ability to do the following:

* List all the members
* Remove a member
* Add or update the information for a theatre complex/theatre
* Add movies to the database
* Update where/when movies are showing
* For a particular customer, show their rental history
* Find the movie that is the most popular
* Find the theatre complex that is most popular

For admins the user interface is pretty intuitive. For users that are admins there will be an extra menu item in the menu bar at the top of the screen, see below:



Once the admin presses that menu item they will be taken to the admin page. From there admins have one click functionality to perform all of the operation listed above. They can select “List Users” to view a list of all the users in the system, and press on one to view additional information like their account info and rental history. Directly on the dashboard admins are able to see the most popular movie and theatre complexes. Similar to the user’s functionality admins can select to “List Theatre Complexes” to add to the list and update any existing ones.



# Appendix A

Please find our relational schema and our seed data below:

/\*

OTMS Database Definition

Group Number: 53

March 28th, 2018

\*/

-- Drop the database

DROP DATABASE omtsdb;

-- Create the database

CREATE DATABASE omtsdb;

-- Use the newly created database

USE omtsdb;

-- Table for theatre complexes

CREATE TABLE theatre\_complex (

id INTEGER NOT NULL AUTO\_INCREMENT,

name varchar(50) NOT NULL,

phone\_number varchar(30) NOT NULL,

a\_street varchar(100) NOT NULL,

a\_city varchar(100) NOT NULL,

a\_postal\_code varchar(7) NOT NULL,

PRIMARY KEY (id)

);

-- Table for theatres

-- Belongs to a theatre complex

CREATE TABLE theatre (

id INTEGER NOT NULL AUTO\_INCREMENT,

theatre\_num INTEGER NOT NULL,

max\_seats INTEGER NOT NULL,

screen\_size varchar(10) NOT NULL,

theatre\_complex\_id INTEGER NOT NULL,

PRIMARY KEY (id, theatre\_num),

FOREIGN KEY (theatre\_complex\_id) REFERENCES theatre\_complex(id)

);

-- Table for a movie supplier

CREATE TABLE supplier (

id INTEGER NOT NULL AUTO\_INCREMENT,

name varchar(100) NOT NULL,

a\_street varchar(100) NOT NULL,

a\_city varchar(100) NOT NULL,

a\_postal\_code varchar(7) NOT NULL,

contact\_fname varchar(50) NOT NULL,

contact\_lname varchar(50) NOT NULL,

phone\_number varchar(30) NOT NULL,

PRIMARY KEY (id)

);

-- Table for movies

-- Supplied from a supplier

CREATE TABLE movie (

id INTEGER NOT NULL AUTO\_INCREMENT,

title varchar(50) NOT NULL,

run\_time TIME NOT NULL,

rating varchar(10) NOT NULL,

plot\_synopsis varchar(300) NOT NULL,

director varchar(50) NOT NULL,

producer varchar(50) NOT NULL,

supplier\_id INTEGER NOT NULL,

image varchar(100),

PRIMARY KEY (id),

FOREIGN KEY (supplier\_id) REFERENCES supplier(id)

);

-- Table for a movies run

CREATE TABLE movie\_run (

id INTEGER NOT NULL AUTO\_INCREMENT,

start\_date DATE NOT NULL,

end\_date DATE NOT NULL,

movie\_id INTEGER NOT NULL,

theatre\_complex\_id INTEGER NOT NULL,

PRIMARY KEY (id, start\_date, end\_date),

FOREIGN KEY (movie\_id) REFERENCES movie(id),

FOREIGN KEY (theatre\_complex\_id) REFERENCES theatre\_complex(id)

);

-- Table for actors

-- Belong to a movie

CREATE TABLE actor (

name varchar(50) NOT NULL,

movie\_id INTEGER NOT NULL,

PRIMARY KEY (name),

FOREIGN KEY (movie\_id) REFERENCES movie(id)

);

-- Table for movie showing

CREATE TABLE showing (

id INTEGER NOT NULL AUTO\_INCREMENT,

seats\_avail INTEGER NOT NULL,

start\_time TIME NOT NULL,

start\_date DATE NOT NULL,

movie\_id INTEGER NOT NULL,

theatre\_id INTEGER NOT NULL,

PRIMARY KEY (id, seats\_avail, start\_time, start\_date),

FOREIGN KEY (movie\_id) REFERENCES movie(id),

FOREIGN KEY (theatre\_id) REFERENCES theatre(id)

);

-- Table for customers

CREATE TABLE customer (

account\_num INTEGER NOT NULL AUTO\_INCREMENT,

fname varchar(50) NOT NULL,

lname varchar(50) NOT NULL,

a\_street varchar(100) NOT NULL,

a\_city varchar(100) NOT NULL,

a\_postal\_code varchar(7) NOT NULL,

phone\_number varchar(30) NOT NULL,

email varchar(100) NOT NULL,

password varchar(30) NOT NULL,

cc\_number varchar(30) NOT NULL,

cc\_expiry varchar(10) NOT NULL,

cc\_cvc varchar(3) NOT NULL,

isAdmin BOOLEAN DEFAULT false,

PRIMARY KEY(account\_num, email)

);

-- Table for customer movie reservations

CREATE TABLE reservation (

reservation\_id INTEGER NOT NULL AUTO\_INCREMENT,

num\_tickets INTEGER NOT NULL,

showing\_id INTEGER NOT NULL,

customer\_id INTEGER NOT NULL,

PRIMARY KEY (reservation\_id),

FOREIGN KEY (showing\_id) REFERENCES showing(id),

FOREIGN KEY (customer\_id) REFERENCES customer(account\_num)

);

-- Table for custom movie reviews

CREATE TABLE review (

review\_id INTEGER NOT NULL AUTO\_INCREMENT,

rating INTEGER NOT NULL,

description INTEGER NOT NULL,

customer\_id INTEGER NOT NULL,

movie\_id INTEGER NOT NULL,

PRIMARY KEY (review\_id),

FOREIGN KEY (customer\_id) REFERENCES customer(account\_num),

FOREIGN KEY (movie\_id) REFERENCES movie(id)

);

-- Inserting some theatre complexes

INSERT INTO theatre\_complex VALUES

(1, 'Silver City Toronto','123-456-789', '1200 Yonge St.', 'Toronto', 'A1B 2C3'),

(2, 'Bloor Street Cinemas','123-456-789', '800 Bloor St.', 'Toronto', 'A1B 2C3'),

(3, 'Yorkdale Malls Cinema','123-456-789', '12 Yorkdale St.', 'Toronto', 'A1B 2C3'),

(4, 'The Beaches Odeon','123-456-789', '14 Beach St.', 'Toronto', 'A1B 2C3'),

(5, 'Cineplex Cinema Yonge-Bloor', '123-456-789', '10 Dundas St. E', 'Toronto', 'A1B 2C3');

-- Inserting some suppliers

INSERT INTO supplier VALUES

(1, 'Warner Bros.', 'Bay St.', 'Toronto', 'A1B 2C3', 'John', 'Smith', '123-456-789'),

(2, 'Entertainment One', 'Yonge St.', 'Toronto', 'A1B 2C3', 'Jane', 'Doe', '123-456-789'),

(3, 'Teletoon', 'Queen St.', 'Toronto', 'A1B 2C3', 'Ryan', 'Cooper', '123-456-789');

-- Inserting some theatres for the complexes

INSERT INTO theatre VALUES

(1, 1, 50, 'Large', 1),

(2, 2, 75, 'Large', 1),

(3, 3, 50, 'Small', 1),

(4, 4, 50, 'Medium', 1),

(5, 1, 25, 'Small', 2),

(6, 2, 50, 'Medium', 2),

(7, 3, 100, 'Large', 2),

(8, 4, 50, 'Medium', 2),

(9, 1, 75, 'Large', 3),

(10, 2, 50, 'Medium', 3),

(11, 3, 50, 'Medium', 3),

(12, 1, 200, 'Large', 4),

(13, 1, 75, 'Large', 5),

(14, 2, 25, 'Small', 5),

(15, 3, 150, 'Large', 5),

(16, 4, 50, 'Medium', 5);

-- Inserting some movies

INSERT INTO movie VALUES

(1, 'The Prestige', 21039, 'R', 'After a trick goes wrong, two magicians have the feud of the century.', 'Christopher Nolan', 'Emma Thomas', 1, 'assets/img/prestige.jpg'),

(2, 'Black Panther', 21453, 'PG', 'The Black Panther must fight for his country against a long lost enemy.', 'Ryan Coogler', 'Kevin Feige', 2, 'assets/img/panther.jpg'),

(3, 'Pitch Perfect 3', 13314, 'R', 'The gang gets back together for one final singing tour.', 'Trish Sie', 'Elizabeth Banks', 1, 'assets/img/perfect.jpg'),

(4, 'Pacific Rim Uprising', 15147, 'PG', 'The humans must once again use their giant robots to fight off enemy Precursors.', 'Steven S. DeKnight', 'John Boyega', 3, 'assets/img/uprising.jpg'),

(5, 'IT', 21557, 'R', 'Pennywise makes a horrifying return to the big screen in one of the best horror movies of the year.', 'Andy Muschietti', 'Roy Lee', 1, 'assets/img/it.jpg'),

(6, 'Peter Rabbit', 13511, 'PG13', 'Peter and his furry friends go on a fun adventure through the neighbourhood.', 'Will Gluck', 'Zareh Nalbandian', 3, 'assets/img/rabbit.jpg'),

(7, 'Tomb Raider', 15819, 'PG13', 'Watch Laura Crofts return to the big screen in this thrilling adventure.', 'Roar Uthaug', 'Graham King', 3, 'assets/img/tomb.jpg'),

(8, 'Shape of Water', 20309, 'R', 'An unsuspecting love story between a mute human and humanoid amphibian.', 'Guillermo del Toro', 'J. Miles. Dale', 2, 'assets/img/water.jpg');

-- Inserting some movie runs

INSERT INTO movie\_run VALUES

(1, 2018-02-16, 2018-04-18, 2, 1),

(2, 2018-03-15, 2018-05-27, 4, 2),

(3, 2017-12-21, 2018-04-07, 3, 4),

(4, 2018-01-12, 2018-04-01, 1, 3),

(5, 2018-02-16, 2018-04-18, 2, 2),

(6, 2018-03-15, 2018-05-27, 4, 1);

-- Inserting some actors

INSERT INTO actor VALUES

('John Boyega', 4),

('Hugh Jackman', 1),

('Christian Bale', 1),

('Chadwich Bosman', 2),

('Anna Kendrick', 3),

('Andy Serkis', 2),

('Jaeden Lieberher', 5),

('Rose Byrne', 6),

('Alicia Vikander', 7),

('Sally Hawkins', 8);

-- Inserting some showings

INSERT INTO showing VALUES

(1, 10, 20000, 2018-05-02, 1, 5),

(2, 33, 80000, 2018-04-01, 2, 11),

(3, 19, 120000, 2018-05-12, 3, 1),

(4, 9, 33000, 2018-04-29, 4, 3),

(5, 4, 24500, 2018-04-13, 2, 5),

(6, 41, 51500, 2018-05-22, 3, 8),

(7, 108, 40000, 2018-04-27, 4, 12),

(8, 20, 73000, 2018-05-02, 5, 10),

(9, 2, 110000, 2018-03-29, 8, 2),

(10, 31, 53000, 2018-06-02, 6, 4),

(11, 17, 21500, 2018-05-02, 5, 6),

(12, 49, 123000, 2018-04-02, 7, 9),

(13, 15, 80000, 2018-04-29, 8, 5);

-- Inserting some customers

INSERT INTO customer VALUES

(1, 'John', 'Dale', 'Avenue Rd.', 'Toronto', 'A1B 2C3', '123-456-789', 'jdale@rogers.com', 'moviefan', '5483-3912-4921-0819', '08/20','456', false),

(2, 'Jane', 'Steel', 'Yonge St.', 'Toronto', 'A1B 2C3', '123-456-789', 'jsteel@gmail.com', 'movieguy', '7356-1475-8416-1783', '04/19', '127', true),

(3, 'Emma', 'Johnson', 'Duffrin Ave.', 'Toronto', 'A1B 2C3', '123-456-789', 'ejohnson@yahoo.ca', 'ilikemovies', '7823-9463-1689-1635', '11/21', '724', false),

(4, 'Matthew', 'Brown', 'Avenue Rd.', 'Toronto', 'A1B 2C3', '123-456-789', 'mbrown@rogers.com', 'fanofmovies', '5425-1948-4612-5223', '10/18', '522', false),

(5, 'Becky', 'Davis', 'King St. W.', 'Toronto', 'A1B 2C3', '123-456-789', 'bdavis@gmail.com', 'password', '8934-2436-3547-8456', '07/19', '736', false),

(6, 'Jared', 'Miller', 'Lawrence Ave.', 'Toronto', 'A1B 2C3', '123-456-789', 'jmiller@rogers.com', 'dogsname', '2034-7456-8375-1648', '09/19', '536', true),

(7, 'Robert', 'Saunders', '19 Whithall Rd.', 'Toronto', 'A1B 2C3', '123-456-789', 'robert@admin.com', '12345', '123456789', '1234', '123', true),

(8, 'Robert', 'Saunders', '19 Whithall Rd.', 'Toronto', 'A1B 2C3', '123-456-789', 'robert@notadmin.com', '12345', '123456789', '1234', '123', false);

-- Inserting some reservations

INSERT INTO reservation VALUES

(1, 3, 7, 1),

(2, 1, 1, 4),

(3, 4, 4, 3),

(4, 1, 5, 4),

(5, 2, 2, 5),

(6, 10, 7, 3),

(7, 1, 6, 1),

(8, 3, 3, 3),

(9, 2, 1, 5);

-- Inserting some movie reviews

INSERT INTO review VALUES

(1, 91, 'Very cool, great acting, insane twist ending.', 1, 1),

(2, 87, 'Awesome effects with a very powerful message.', 1, 2),

(3, 42, 'Very disappointing compared to the first 2, was expecting more.', 3, 3),

(4, 65, 'Great action scenes, a good, mind numbing movie.', 4, 4),

(5, 22, 'Not a huge fan, was forced to go watch with my family.', 4, 3),

(6, 86, 'Really interesting, I love movies from this time period.', 3, 1),

(7, 30, 'Seemed pretty pointless with no real plot, just giants fighting.', 5, 4),

(8, 92, 'Incredible movie with an amazing cast, very excited for the sequel.', 3, 2),

# Appendix B

Please find the SQL commands we utilized within our application below:

NOTE: That attributes in square brackets within the commands represent placeholders.

**Administration Operations**

**Delete account:**

DELETE FROM customer WHERE account\_num = [account\_num];

**List users:**

SELECT \* FROM customer;

**View most popular movie:**

SELECT \*, (SELECT SUM(num\_tickets) FROM reservation, showing WHERE (

reservation.showing\_id = showing.movie\_id) AND (showing.movie\_id = movie.id)) sold\_tick

FROM movie ORDER BY sold\_tick DESC LIMIT 1;

**View most popular theatre complex:**

SELECT (

SELECT

SUM(num\_tickets)

FROM

reservation,

showing,

theatre

WHERE (

reservation.showing\_id = showing.movie\_id

) AND(

showing.theatre\_id = theatre.id

) AND(

theatre.theatre\_complex\_id = theatre\_complex.id

)

) sold\_tick

FROM

theatre\_complex

ORDER BY

sold\_tick

DESC

LIMIT 1

**Customer Operations**

**View show times of a given movie:**

**Insert reservation for a customer:**

**Delete reservation:**

**Insert review:**

**View customer reservations:**

**View all movies showing within a certain complex:**

**Customer Operations**

**View show times of a given movie:**

SELECT showing.start\_time, showing.seats\_avail, theatre\_complex.name

FROM showing

INNER JOIN movie

ON movie.id = showing.movie\_id

INNER JOIN theatre

ON theatre.id = showing.theatre\_id

INNER JOIN theatre\_complex

ON theatre\_complex.id = theatre.theatre\_complex\_id

WHERE

(movie.id = \*) AND

(theatre\_complex.id = ^);

//replace \* with movie\_id

//replace ^ with theatre\_complex\_id

**Insert reservation for a customer:**

INSERT INTO reservation

VALUES (reservation\_id, num\_tickets, showing\_id, customer\_id);

//reservation id auto increments, num tickets has to be an input, showing id is selected when they click the showing to book, cust id is selected with login

**Delete reservation:**

DELETE FROM reservation

WHERE reservation\_id = \*;

//get \* from the reservaton selected to be deleted

**Insert review:**

INSERT INTO review

VALUES (id, rating, description, customer\_id, movie\_id);

//id auto increments, rating and description must be added by customer, customer id gotten from login, movie id gotten from movie clicked on to give review

**Customer Operations**

**View customer reservations:**

SELECT \*

FROM reservation

INNER JOIN customer

ON customer.account\_num = reservation.customer\_id

WHERE customer.account\_num = \*;

**View all movies showing within a certain complex:**

SELECT movie.title

FROM movie

INNER JOIN showing

ON showing.movie\_id = movie.id

INNER JOIN theatre

ON theatre.id = showing.theatre\_id

INNER JOIN theatre\_complex

ON theatre\_complex.id = theatre.theatre\_complex\_id

WHERE theatre\_complex.id = \*;