



CISC 332 Project - Final Report

**Database Design and Implementation, SQL Queries,
Assumptions, User Guide, Discussion**

Submitted by:

Emily Bao 10103388

Yuhan Wang 10195692

Chantal Montgomery 10191890

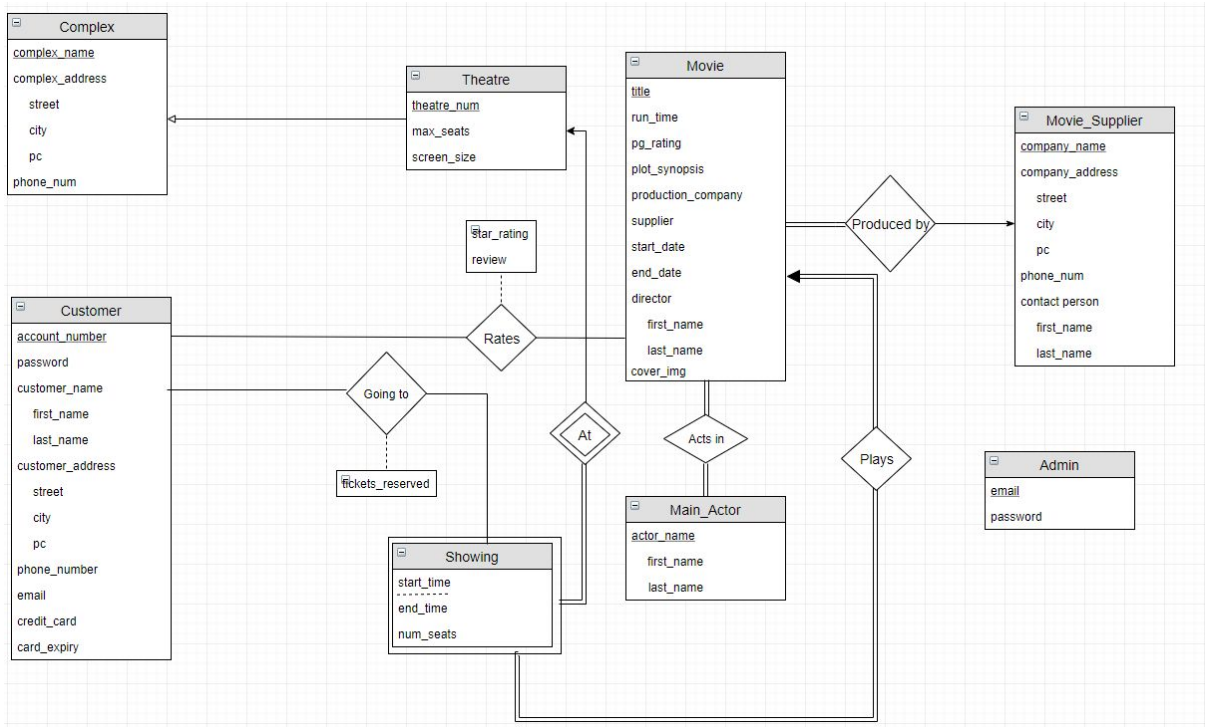
CONTENTS

Assumptions	2
ER Diagram	3
Relational Schema	4
SQL from Application	13
Discussion	26
User Guide	31

Assumptions

1. A theatre cannot exist without a complex
2. A complex cannot exist without a theatre
3. A movie supplier can exist without having a movie at any theatre
4. Every movie needs to have a supplier
5. Movies cannot have the same name, they must all be unique
6. Showings do not get deleted from the database even if they have passed, therefore each movie has at least one showing
7. No actors have the same first and last name, they must all be unique combinations of first and last name
8. All actors in the database act in at least one movie
9. Admins have been given their email and password information and they are not allowed to create new admin accounts

ER Diagram:



```

CREATE TABLE Theatre(
  complex_name VARCHAR(50) NOT NULL,
  street VARCHAR(100),
  city VARCHAR(100),
  pc CHAR(6),
  phone_num CHAR(10),
  theatre_num INTEGER NOT NULL,
  max_seats INTEGER NOT NULL,
  screen_size CHAR(1) NOT NULL,
  PRIMARY KEY(complex_name, theatre_num)
);
CREATE TABLE Customer(
  account_number CHAR(16) NOT NULL,
  password VARCHAR(30) NOT NULL,
  first_name VARCHAR(20) NOT NULL,
  last_name VARCHAR(20) NOT NULL,
  street VARCHAR(100),
  city VARCHAR(100),
  pc CHAR(6),
  phone_number CHAR(10),
  email VARCHAR(30) NOT NULL,
  credit_card VARCHAR(64) NOT NULL,
  card_expiry TIMESTAMP NOT NULL,
  PRIMARY KEY(account_number),
  UNIQUE(email)
);
CREATE TABLE Movie(
  title VARCHAR(100) NOT NULL,
  run_time INTEGER NOT NULL,
  pg_rating VARCHAR(4),
  plot_synopsis VARCHAR(1000),
  production_company VARCHAR(100),
  supplier VARCHAR(200) NOT NULL,
  start_date DATE NOT NULL,
  end_date DATE NOT NULL,
  director_fname VARCHAR(20) NOT NULL,
  director_lname VARCHAR(20) NOT NULL,
  Cover_img VARCHAR(1000),
  FOREIGN KEY(supplier) REFERENCES Movie_Supplier(company_name),
  PRIMARY KEY(title)
);
CREATE TABLE Movie_Supplier(
  company_name VARCHAR(100) NOT NULL,
  street VARCHAR(100) NOT NULL,
  city VARCHAR(100) NOT NULL,
  pc CHAR(6) NOT NULL,

```

```

    phone_num CHAR(10) NOT NULL,
    contact_fname VARCHAR(20),
    contact_lname VARCHAR(20),
    PRIMARY KEY(company_name)
);
CREATE TABLE Main_Actor(
    first_name VARCHAR(20) NOT NULL,
    last_name VARCHAR(20) NOT NULL,
    PRIMARY KEY(first_name, last_name)
);
CREATE TABLE Acts_In(
    actor_fname VARCHAR(20) NOT NULL,
    actor_lname VARCHAR(20) NOT NULL,
    movie_title VARCHAR(100) NOT NULL,
    FOREIGN KEY(actor_fname, actor_lname) REFERENCES Main_Actor(first_name,
last_name),
    FOREIGN KEY(movie_title) REFERENCES Movie(title),
    PRIMARY KEY(
        actor_fname,
        actor_lname,
        movie_title
    )
);
CREATE TABLE Showing(
    movie_title VARCHAR(100) NOT NULL,
    start_time DATETIME NOT NULL,
    end_time DATETIME NOT NULL,
    theatre_num INTEGER NOT NULL,
    num_seats INTEGER NOT NULL,
    complex_name VARCHAR(50) NOT NULL,
    FOREIGN KEY(movie_title) REFERENCES Movie(title),
    FOREIGN KEY(complex_name, theatre_num) REFERENCES Theatre(complex_name,
theatre_num),
    PRIMARY KEY(
        start_time,
        theatre_num,
        complex_name
    )
);
CREATE TABLE Rates(
    customer_account CHAR(16) NOT NULL,
    movie_title VARCHAR(100) NOT NULL,
    star_rating TINYINT NOT NULL,
    review VARCHAR(1000),
    FOREIGN KEY(customer_account) REFERENCES Customer(account_number),

```

```
FOREIGN KEY(movie_title) REFERENCES Movie(title),
PRIMARY KEY(customer_account, movie_title)
);
```

```
create table Going_To(
customer_account char(16) not null,
movie_title varchar(100) not null,
start_time datetime not null,
theatre_num integer not null,
complex_name varchar(50) not null,
tickets_reserved integer not null,
foreign key(movie_title) references Movie(title),
foreign key(customer_account) references Customer(account_number),
foreign key(start_time, theatre_num, complex_name) references Showing(start_time,
theatre_num, complex_name),
primary key(customer_account, start_time, theatre_num)
);
```

```
CREATE TABLE Admin(
email varchar(30) NOT NULL PRIMARY KEY,
password varchar(30) NOT NULL,
unique(email)
);
```

```
INSERT INTO Admin
VALUES('admin001@test.com', 'cisc332'),('admin002@test.com', 'cisc332');
```

INSERT into Theatre values

```
('Cineplex','55 Bloor St','Toronto','M4W1A5','4169616304',1,280,'l'),
('Cineplex','55 Bloor St','Toronto','M4W1A5','4169616304',2,300,'l'),
('Cineplex','55 Bloor St','Toronto','M4W1A5','4169616304',3,160,'m'),
('Cineplex','55 Bloor St','Toronto','M4W1A5','4169616304',4,140,'m'),
('Cineplex','55 Bloor St','Toronto','M4W1A5','4169616304',5,50,'s'),
('The Royal','608 College St','Toronto','M6G1B4','4164664400',1,300,'l'),
('The Royal','608 College St','Toronto','M6G1B4','4164664400',2,150,'m'),
('The Royal','608 College St','Toronto','M6G1B4','4164664400',3,170,'m'),
('Alliance','1651 Queen St','Toronto','M4L1G5','4166991327',1,270,'l'),
('Alliance','1651 Queen St','Toronto','M4L1G5','4166991327',2,170,'m'),
('Alliance','1651 Queen St','Toronto','M4L1G5','4166991327',3,150,'m'),
('Alliance','1651 Queen St','Toronto','M4L1G5','4166991327',4,150,'m');
```

INSERT into Customer values

('3369814532808088','dnanfkvn','Robin','Goodwin','99 Colborne Street',
'Kingston','K1L2W1','3568656000','rgoodwin@live.com','3116713299240001','2019-02-01'),
('3869839532808088','dalfvmiaeo','Amy','Hamilton','100 Colborne Street',
'Kingston','K1L2W2','3269000777','ahamilton@live.com','3716711119246871','2021-08-01'),
('3612814532808088','dmsaldaf','Paul','Rondo','101 Colborne Street',
'Kingston','K1L3W1','3069656777','prondo@live.com','3563513299246871','2022-01-05'),
('3941814532808088','dnalvvas','Tony','Ken','28 First Ave',
'Kingston','K1L2W1','3769656888','tken@live.com','3870713299246871','2025-01-08'),
('3201814532808021','fdnsalkvm','Kim','Ayeno','56 Duncan Street',
'Kingston','K1M2W1','3569656651','kayeno@live.com','3210386299246871','2021-09-08'),

('4369814532808088','nvalmvd','Jeremy','Gordon','99 Quebec Street',
'Kingston','K3K2W1','4169656000','jgordon@live.com','4716713299240001','2019-11-01'),
('4869839532808088','dasfnlnda','James','Curry','100 Quebec Street',
'Kingston','K3K2W2','4169000777','jcurry@live.com','4716711119246871','2020-06-01'),
('4612814532808088','nvalmvd','Lebron','Kenedy','101 Quebec Street',
'Kingston','K3K3W1','4169656777','lkenedy@live.com','4023513299246871','2022-01-01'),
('4941814532808088','dnofajda','Chris','Nash','36 First Ave',
'Kingston','K2K2W1','4169656888','cnash@live.com','4010713299246871','2025-01-01'),
('4201814532808021','adffaadada','Jason','Smith','12 Codd Street',
'Kingston','K5L2W1','4169656651','jsmith@live.com','4010386299246871','2021-09-01'),

('6046462851066361','as33dasd','Melinda','Gill','222 Stuart Street',

'Kingston','K7L2W1','4169656777','mthurn@live.com','4716713299246871','2018-12-01'),
('9982907981088662','muEW8YEK','Juana','Bowman','3978 Gorham
Street','London','N0N0N0','6135550157','rgarcia@optonline.net','4253402729496939','2020-
03-01'),
('5723962851066688','jguje6Jy','Bill','Henry','500 Kingston
Rd','Toronto','M4L1V3','6134550001','webdragon@comcast.net','4532817688666953','2022-
05-01'),
('7865462856766321','5NZkJLkj','Dolores','Wood','315 St Germain
Ave','Toronto','M4E3K7','6479368855','crandall@sbcglobal.net','4539287669568813','2022-0
8-01'),
('8874672851857461','F39DmRgh','Carl','Riley','26 Goodwood Park Cres East
York','Toronto','M4C2G5','9058789933','fangorn@hotmail.com','4539410702754915','2019-0
8-01'),
('8674462864456324','gEKNXSRx','Guadalupe','Houston','48 St Clair Ave
W','Toronto','M4V2Z2','6136665633','mxiao@yahoo.com','5536652401578877','2020-12-01')
,
('8563462851068567','f9srHP5d','Ian','Pratt','42 Balsam
Ave','Toronto','M4E3B4','6478290964','jguyer@aol.com','5229766204193627','2021-01-01'),
('7755462662366001','pju9y2ey','Javier','Hubbard','101 Hillingdon
Ave','Toronto','M4C3H9','4168988888','drezet@me.com','5353895832119767','2019-11-01'),

('8587462776065566','SysFfWRq','Ester','Bridges','304 Berkeley St','Toronto','M5A2X5','9056678443','euice@outlook.com','5299770036729702','2023-03-01'),

('6556462822226333','GPDzmK9u','Jeffrey','Greene','300 Silver Birch Ave','Toronto','M4E3L5','6137777707','firstpr@att.net','5194659098906983','2021-01-01'),

('0000111122223333','test123','Emily','Bao','5735 Tayside Crescent','Mississauga','L5M5J4','6477795377','emily@test.com','1234567829496939','2020-03-01'),

('0000111122220000','test123','Chantal','Montgomery','5711 Tayside Crescent','Mississauga','L5M5J4','6477799999','chantal@test.com','1234567829496939','2020-03-01'),

('0000111122221111','test123','Yuhan','Wang','5712 Tayside Crescent','Mississauga','L5M5J4','6477791111','yuhan@test.com','1234567829487939','2020-04-01'),

('0000111122223331','test123','Melinda','Gill','5713 Tayside Crescent','Toronto','L5M5J4','6473391111','melinda@gmail.com','1234567829496939','2020-07-01'),

('0000443122221111','test123','Melinda','Gill','5712 Windsor Crescent','Mississauga','L5M5J4','6477791111','m.gill@hotmail.com','1234567339496939','2020-12-01');

INSERT into Movie values

('Beauty and the Beast', 129, 'PG', 'An adaptation of the fairy tale about a monstrous-looking prince and a young woman who fall in love.', 'Walt Disney Studios', 'Elevation Pictures', '2017-10-02', '2017-12-03', 'Bill', 'Condon', 'https://upload.wikimedia.org/wikipedia/en/d/d6/Beauty_and_the_Beast_2017_poster.jpg'),

('Black Panther', 134, 'PG13', 'T"Challa, the King of Wakanda, rises to the throne in the isolated, technologically advanced African nation, but his claim is challenged by a vengeful outsider who was a childhood victim of T"Challa"s father"s mistake.', 'Marvel Studios', 'Elevation Pictures', '2018-02-03', '2018-04-01', 'Ryan', 'Coogler', 'https://13thdimension.com/wp-content/uploads/2018/02/Black-Panther-poster-main-xl-580x859.jpg'),

('Jumanji: Welcome to the Jungle', 119, 'PG13', 'Four teenagers are sucked into a magical video game, and the only way they can escape is to work together to finish the game.', 'Columbia Pictures', 'Brightlight Pictures', '2018-02-22', '2018-04-05', 'Jake', 'Kasdan', 'https://assets.voxcinemas.com/posters/P_HO00004797.jpg'),

('Maze Runner: The Death Cure', 141, 'PG13', 'Young hero Thomas embarks on a mission to find a cure for a deadly disease known as the Flare.', 'Gotham Group', 'Entertainment One', '2018-03-01', '2018-04-28', 'Wes', 'Ball', 'https://assets.voxcinemas.com/posters/P_HO00005023.jpg'),

('Winchester', 99, 'PG13', 'Ensnared in her sprawling California mansion, eccentric firearm heiress Sarah Winchester believes she is haunted by the souls of people killed by the Winchester repeating rifle.', 'Bullit Entertainment', 'Entertainment One', '2018-03-07', '2018-04-01', 'Michael', 'Spierig', 'http://www.espaciomediterraneo.com/uploads/cine/8934_winchester-395060487-large.jpg'),

('A Wrinkle in Time', 109, 'PG', 'After the disappearance of her scientist father, three peculiar beings send Meg, her brother, and her friend to space in order to find him.', 'Walt Disney Studios', 'Entertainment One', '2018-03-16', '2018-06-01', 'Ava', 'DuVernay', 'https://www.cinelandia.com.co/archivos/proximo/Afiche_Nuevos_viaje.jpg');

INSERT into Acts_In values

('Emma', 'Watson', 'Beauty and the Beast'), ('Dan', 'Stevens', 'Beauty and the Beast'), ('Chadwick', 'Boseman', 'Black Panther'), ('Lupita', 'Nyong'o', 'Black Panther'), ('Dwayne', 'Johnson', 'Jumanji: Welcome to the Jungle'), ('Kevin', 'Hart', 'Jumanji: Welcome to the Jungle'), ('Jack', 'Black', 'Jumanji: Welcome to the Jungle'), ('Dylan', 'O'Brien', 'Maze Runner: The Death Cure'), ('Ki Hong', 'Lee', 'Maze Runner: The Death Cure'), ('Kaya', 'Scodelario', 'Maze Runner: The Death Cure'), ('Helen', 'Mirren', 'Winchester'), ('Storm', 'Reid', 'A Wrinkle in Time'), ('Oprah', 'Winfrey', 'A Wrinkle in Time'), ('Reese', 'Witherspoon', 'A Wrinkle in Time'), ('Mindy', 'Kaling', 'A Wrinkle in Time');

INSERT into Movie_Supplier values

('Elevation Pictures', '166 Pearl St West Suite 300', 'Toronto', 'L2M5J8', '4167789876', 'Catherine', 'Simmonds'), ('Brightlight Pictures', '543 River Street', 'Vancouver', 'E9R3K1', '7053332647', 'Shawn', 'Westman'), ('Entertainment One', '499 Dundas Street West', 'Toronto', 'L2J3H8', '4167681004', 'Dan', 'Johnson'), ('Warner Bros.', '4000 Warner Blvd', 'Vancouver', 'V7G5X2', '7053245432', 'Scott', 'Martin');

INSERT into Main_Actor values

('Emma', 'Watson'), ('Dan', 'Stevens'), ('Chadwick', 'Boseman'), ('Lupita', 'Nyong'o'), ('Dwayne', 'Johnson'), ('Kevin', 'Hart'), ('Jack', 'Black'), ('Dylan', 'O'Brien'), ('Ki Hong', 'Lee'), ('Kaya', 'Scodelario'), ('Helen', 'Mirren'), ('Storm', 'Reid'), ('Oprah', 'Winfrey'), ('Reese', 'Witherspoon'), ('Mindy', 'Kaling');

INSERT into Showing values

('Maze Runner: The Death Cure', '2018-03-06 19:30:00', '2018-03-06 21:51:00', 1, 275, 'Cineplex'),
('Maze Runner: The Death Cure', '2018-03-06 13:00:00', '2018-03-06 15:21:00', 1, 279, 'Cineplex'),
('Maze Runner: The Death Cure', '2018-03-20 19:30:00', '2018-03-20 21:51:00', 1, 277, 'The Royal'),
('Maze Runner: The Death Cure', '2018-03-15 20:00:00', '2018-03-15 22:21:00', 3, 149, 'Alliance'),
('Jumanji: Welcome to the Jungle', '2018-04-01 20:00:00', '2018-04-01 21:59:00', 3, 148, 'Alliance'),
('Jumanji: Welcome to the Jungle', '2018-03-23 18:30:00', '2018-03-23 20:29:00', 2, 147, 'The Royal'),
('Jumanji: Welcome to the Jungle', '2018-03-25 20:00:00', '2018-03-25 21:59:00', 1, 280, 'Cineplex'),
('Beauty and the Beast', '2017-10-22 19:00:00', '2017-10-22 21:09:00', 1, 275, 'Cineplex'),
('Winchester', '2018-03-12 22:00:00', '2018-12-22 23:39:00', 4, 150, 'Alliance'),
('A Wrinkle in Time', '2018-04-10 14:00:00', '2018-04-10 15:49:00', 1, 300, 'The Royal'),
('Winchester', '2018-03-30 22:00:00', '2018-12-22 23:39:00', 4, 150, 'Alliance'),
('Winchester', '2018-03-30 20:00:00', '2018-12-22 21:39:00', 1, 280, 'Cineplex'),
('A Wrinkle in Time', '2018-04-05 22:00:00', '2018-12-22 23:49:00', 3, 160, 'Cineplex'),
('A Wrinkle in Time', '2018-04-05 22:00:00', '2018-12-22 23:39:00', 2, 300, 'Alliance'),
('Black Panther', '2018-03-24 19:00:00', '2018-03-10 21:14:00', 4, 150, 'Alliance'),
('Black Panther', '2018-03-26 19:00:00', '2018-03-10 21:14:00', 3, 80, 'Cineplex'),
('Black Panther', '2018-03-29 17:00:00', '2018-03-10 19:14:00', 1, 50, 'Alliance'),
('Winchester', '2018-04-20 16:00:00', '2018-12-22 17:39:00', 3, 144, 'Cineplex'),
('Winchester', '2018-04-20 20:00:00', '2018-12-22 21:39:00', 2, 300, 'Cineplex'),
('A Wrinkle in Time', '2018-04-05 22:00:00', '2018-12-22 23:49:00', 4, 150, 'Alliance'),
('A Wrinkle in Time', '2018-04-05 22:00:00', '2018-12-22 23:39:00', 1, 218, 'Cineplex');

INSERT into Going_To values

('0000111122220000','A Wrinkle in Time','2018-04-05 22:00:00',1,'Cineplex',5),
('6046462851066361', 'A Wrinkle in Time', '2018-04-05 22:00:00', 1, 'Cineplex', 8),
('9982907981088662', 'A Wrinkle in Time', '2018-04-05 22:00:00', 1, 'Cineplex', 3),
('5723962851066688', 'A Wrinkle in Time', '2018-04-05 22:00:00', 1, 'Cineplex', 1),
('7865462856766321', 'A Wrinkle in Time', '2018-04-05 22:00:00', 1, 'Cineplex', 2),
('8874672851857461', 'A Wrinkle in Time', '2018-04-05 22:00:00', 1, 'Cineplex', 4),
('8563462851068567', 'A Wrinkle in Time', '2018-04-05 22:00:00', 1, 'Cineplex', 3),
('8674462864456324', 'A Wrinkle in Time', '2018-04-05 22:00:00', 1, 'Cineplex', 2),
('3369814532808088', 'A Wrinkle in Time', '2018-04-05 22:00:00', 1, 'Cineplex', 3),
('3869839532808088', 'A Wrinkle in Time', '2018-04-05 22:00:00', 1, 'Cineplex', 1),
('3612814532808088', 'A Wrinkle in Time', '2018-04-05 22:00:00', 1, 'Cineplex', 5),
('3941814532808088', 'A Wrinkle in Time', '2018-04-05 22:00:00', 1, 'Cineplex', 3),
('3201814532808021', 'A Wrinkle in Time', '2018-04-05 22:00:00', 1, 'Cineplex', 2),
('4369814532808088', 'A Wrinkle in Time', '2018-04-05 22:00:00', 1, 'Cineplex', 6),
('4869839532808088', 'A Wrinkle in Time', '2018-04-05 22:00:00', 1, 'Cineplex', 1),

('4612814532808088','A Wrinkle in Time','2018-04-05 22:00:00',1,'Cineplex',5),
 ('4941814532808088','A Wrinkle in Time','2018-04-05 22:00:00',1,'Cineplex',2),
 ('4201814532808021','A Wrinkle in Time','2018-04-05 22:00:00',1,'Cineplex',3),
 ('0000111122220000','Maze Runner: The Death Cure','2018-03-06 19:30:00',1,'Cineplex',3),
 ('6046462851066361','Maze Runner: The Death Cure','2018-03-06 19:30:00',1,'Cineplex',2),
 ('9982907981088662','Maze Runner: The Death Cure','2018-03-06 13:00:00',1,'Cineplex',1),
 ('5723962851066688','Maze Runner: The Death Cure','2018-03-15 20:00:00',3,'Alliance',1),
 ('7865462856766321','Maze Runner: The Death Cure','2018-03-20 19:30:00',1,'The
 Royal',2),
 ('8874672851857461','Jumanji: Welcome to the Jungle','2018-04-01 20:00:00',3,'Alliance',1),
 ('8674462864456324','Jumanji: Welcome to the Jungle','2018-04-01 20:00:00',3,'Alliance',1),
 ('8563462851068567','Jumanji: Welcome to the Jungle','2018-03-23 18:30:00',2,'The
 Royal',1),
 ('0000111122220000','Jumanji: Welcome to the Jungle','2018-03-23 18:30:00',2,'The
 Royal',2),
 ('3369814532808088','Maze Runner: The Death Cure','2018-03-20 19:30:00',1,'The
 Royal',3),
 ('3869839532808088','Maze Runner: The Death Cure','2018-03-20 19:30:00',1,'The
 Royal',5),
 ('3612814532808088','Maze Runner: The Death Cure','2018-03-20 19:30:00',1,'The
 Royal',1),
 ('3941814532808088','Maze Runner: The Death Cure','2018-03-20 19:30:00',1,'The
 Royal',2),
 ('3201814532808021','Maze Runner: The Death Cure','2018-03-20 19:30:00',1,'The
 Royal',10),
 ('4369814532808088','Winchester','2018-04-20 16:00:00',3,'Cineplex',6),
 ('4869839532808088','Winchester','2018-04-20 16:00:00',3,'Cineplex',2),
 ('4612814532808088','Winchester','2018-04-20 16:00:00',3,'Cineplex',4),
 ('4941814532808088','Winchester','2018-04-20 16:00:00',3,'Cineplex',3),
 ('4201814532808021','Winchester','2018-04-20 16:00:00',3,'Cineplex',1),
 ('4369814532808088','Black Panther','2018-03-26 19:00:00',3,'Cineplex',3),
 ('4869839532808088','Black Panther','2018-03-26 19:00:00',3,'Cineplex',7),
 ('4612814532808088','Black Panther','2018-03-26 19:00:00',3,'Cineplex',2),
 ('4941814532808088','Black Panther','2018-03-26 19:00:00',3,'Cineplex',5),
 ('4201814532808021','Black Panther','2018-03-26 19:00:00',3,'Cineplex',3),
 ('7755462662366001','Beauty and the Beast','2017-10-22 19:00:00',1,'Cineplex',1),
 ('8587462776065566','Beauty and the Beast','2017-10-22 19:00:00',1,'Cineplex',3),
 ('6556462822226333','Beauty and the Beast','2017-10-22 19:00:00',1,'Cineplex',1),
 ('6046462851066361','Jumanji: Welcome to the Jungle','2018-04-01 20:00:00',3,'Alliance',1),
 ('0000111122223331','Jumanji: Welcome to the Jungle','2018-03-23 18:30:00',2,'The
 Royal',1),
 ('0000111122223331','A Wrinkle in Time','2018-04-05 22:00:00',1,'Cineplex',3),
 ('6046462851066361','Winchester','2018-04-20 16:00:00',3,'Cineplex',2),
 ('0000443122221111','Winchester','2018-04-20 16:00:00',3,'Cineplex',4),
 ('0000443122221111','A Wrinkle in Time','2018-04-05 22:00:00',1,'Cineplex',2);

INSERT into Rates values

('6046462851066361','Maze Runner: The Death Cure',9,'And yet as the final act succumbed to dull, apocalyptic formula, I saw an entire sub-genre slip away with it: The Death Cure is a grim, half-hearted farewell to this wave of young-adult dystopias.'),

('9982907981088662','Maze Runner: The Death Cure',8,'Everything else in The Death Cure moves along assuredly and relentlessly, but like so many final installments of series, it has a hard time letting go.'),

('5723962851066688','Maze Runner: The Death Cure',6,'The Death Cure serves as a dissatisfying ending to an only passably serviceable franchise.'),

('7865462856766321','Maze Runner: The Death Cure',7,'Better than expected is the best thing that can be said about this movie. '),

('7755462662366001','Maze Runner: The Death Cure',7,'The Death Cure is a grim, half-hearted farewell to this wave of young-adult dystopias.'),

('8874672851857461','Jumanji: Welcome to the Jungle',8,"This crowd-pleasing reboot may not be earth-shatteringly good, but it benefits from its stars' irresistible comedic and action charm."),

('86744628644563241','Jumanji: Welcome to the Jungle',7,'A consistently inventive and chucklesome reinvention of the Jumanji concept. Okay, so it coasts on the charm of its lead quartet, but when there's this much charm, that's no bad thing.'),

('8563462851068567','Jumanji: Welcome to the Jungle',9,"Welcome to the Jungle is an entertaining - if shallow - return to the world of Jumanji that's intended more for youngsters than nostalgic adults."),

('7755462662366001','Beauty and the Beast',9,"Watson is an ideal Belle in this wonderful remake that's at once nostalgic and new, bringing to life the musical both for kids and life-long adult fans."),

('8587462776065566','Beauty and the Beast',9,"A touching, eminently watchable, at times slightly awkward experience that justifies its existence yet never totally convinces you it's a movie the world was waiting for."),

('6556462822226333','Beauty and the Beast',5,"As a whole, the production doesn't hold a candelabrum to Kenneth Branagh's lovely 2015 version of Cinderella, to say nothing of the 1991 Beauty original."),

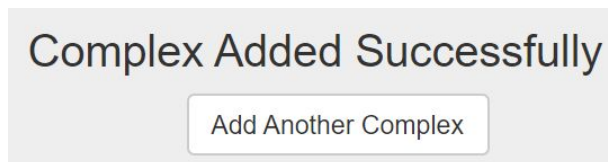
('5723962851066688','Beauty and the Beast',8,"From the poor provincial town to the dazzling ballroom, the movie is a larger-than-life feast for the eyes. Emma Watson is Belle is smart, progressive and not a princess.");

Administrator Side:

1. Add a new complex (A complex cannot exist without a theatre)

```
INSERT INTO theatre VALUES ('$complex_name', '$street', '$city', '$pc', $phone_num, $theatre_num, $max_seats, '$screen_size');
```

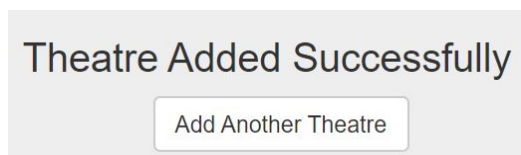
Output:



2. Add a new theatre (A theatre cannot exist without a complex)

```
INSERT INTO theatre VALUES ('$complex_name', '$street', '$city', '$pc', $phone_num, $theatre_num, $max_seats, '$screen_size');
```

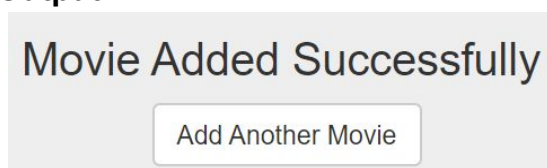
Output:



3. Add a new movie

```
INSERT INTO movie VALUES ('$title', $run_time, '$pg_rating', '$plot_synopsis', '$production_company', '$supplier', '$start_date', '$end_date', '$director_fname', '$director_lname', '$Cover_img');
```

Output:



4. Add a new showing

```
INSERT INTO showing VALUES('$movie_title', '$start_time', '$end_time', $theatre_num, $num_seats, '$complex_name');
```

Output:

Showing Added Successfully

Add Another Showing

5. Find the most popular complex

```
SELECT
    complex_name
FROM
    ( SELECT complex_name,
        SUM(tickets_reserved) AS A1
      FROM going_to
      GROUP BY complex_name
    ) AS T
WHERE A1 IN
    (SELECT MAX(A1) AS A2
     FROM
        ( SELECT
            complex_name,
            SUM(tickets_reserved) AS A1
          FROM going_to
          GROUP BY complex_name
        ) AS T);
```

Output:

The most popular complex is Cineplex

6. Calculate the number of movie sold in the most popular complex

```
SELECT MAX(A1) AS A2
FROM
    ( SELECT complex_name, SUM(tickets_reserved) AS A1
      FROM going_to
      GROUP BY complex_name
    ) as T;
```

Output:

— Number of movie tickets sold is: 117

Complete output of 5 and 6 that will shown on our admin main page:

Complex Analytics

The most popular complex is Cineplex

— Number of movie tickets sold is: 117

7. Find the most popular movie

```
SELECT movie_title
FROM
  ( SELECT movie_title, SUM(tickets_reserved) AS A1
    FROM going_to
    GROUP BY movie_title
  ) AS T
where A1 IN
  ( SELECT MAX(A1) AS A2
    FROM
      ( SELECT movie_title, SUM(tickets_reserved) AS A1
        FROM going_to
        GROUP BY movie_title
      ) as T);
```

Output:

The most popular movie is A Wrinkle in Time

8. Find the number of tickets sold of the most popular movie

```
SELECT MAX(A1) AS A2
FROM
  ( SELECT movie_title, SUM(tickets_reserved) AS A1
    FROM going_to
    GROUP BY movie_title
  ) as T;
```

Output:

— Number of movie tickets sold is: 64

Complete output of 7 and 8 that will shown on our admin main page:

Movie Analytics

The most popular movie is A Wrinkle in Time

— Number of movie tickets sold is: 64

9. Update complex information

```
UPDATE theatre SET street = '$street', city = '$city', pc = '$pc',  
phone_num = '$phone_num' WHERE complex_name = '$complex_name';
```

Output:

Complex Updated Successfully

10. Update theatre information

```
UPDATE theatre SET max_seats = $max_seats, screen_size =  
'$screen_size' WHERE complex_name = '$complex_name' and theatre_num  
= $theatre_num;
```

Output:

Theatre Updated Successfully

11. Search members

(1) Search member by account number:

```
SELECT first_name, last_name, email FROM customer WHERE  
account_number = '" . $_POST['account'] . "';
```

Output (when we search account number = 8874672851857461):

Carl Riley fangorn@hotmail.com 8874672851857461

[View Member](#)

(2) Search member by first and last name:

```
SELECT account_number, email FROM customer WHERE first_name =  
'"._POST['firstname']."' AND last_name = '"._POST['lastname']."';
```

Output (when we enter “Robin” in first name field and “Goodwin” in the last name field):

Robin Goodwin rgoodwin@live.com 3369814532808088

[View Member](#)

(3) List all members. When all three search boxes (first name, last name and account number) are empty, all members will be shown:

```
SELECT first_name, last_name, account_number, email FROM customer;
```

Output (Part of all results):

Chantal Montgomery chantal@test.com 0000111122220000

[View Member](#)

Yuhan Wang yuhan@test.com 0000111122221111

[View Member](#)

Melinda Gill melinda@gmail.com 0000111122223331

[View Member](#)

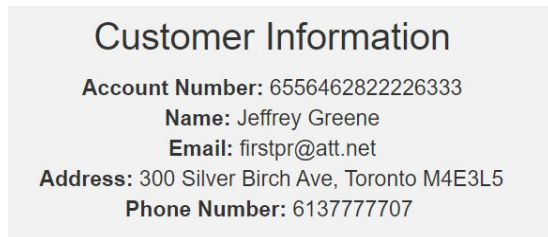
Emily Bao emily@test.com 0000111122223333

[View Member](#)

12. Display customer's information:

```
SELECT first_name, last_name, street, city, pc, phone_number, email  
FROM customer WHERE account_number = ".$account_number.";
```

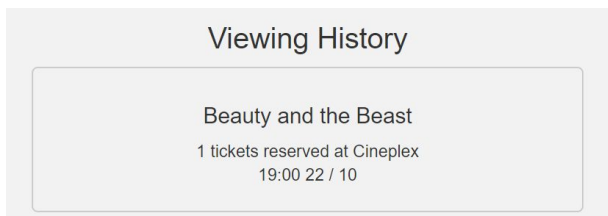
Output:



13. Display the viewing history of a customer:

```
SELECT movie_title, start_time, theatre_num, complex_name,  
tickets_reserved FROM going_to WHERE customer_account =  
".$account_number.";
```

Output:

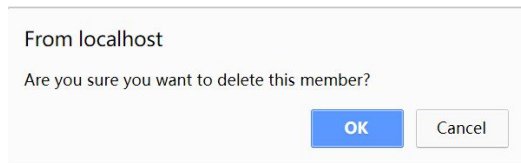


14. Delete a customer from the database (Delete in 3 separate queries):

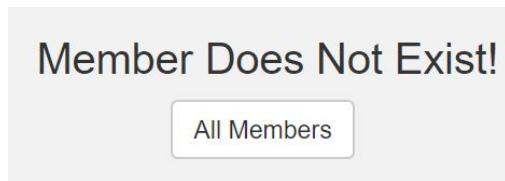
```
DELETE FROM `going_to` WHERE `customer_account` =  
'".$_POST['account'].'';  
DELETE FROM rates WHERE `customer_account` =  
'".$_POST['account'].'';  
DELETE FROM customer WHERE `account_number` =  
'".$_POST['account'].'';
```

Output:

(1) A pop up message appears for confirmation of deleting the selected member information



(2) After pressing "OK", it will shown the deleted member is not in the database anymore:

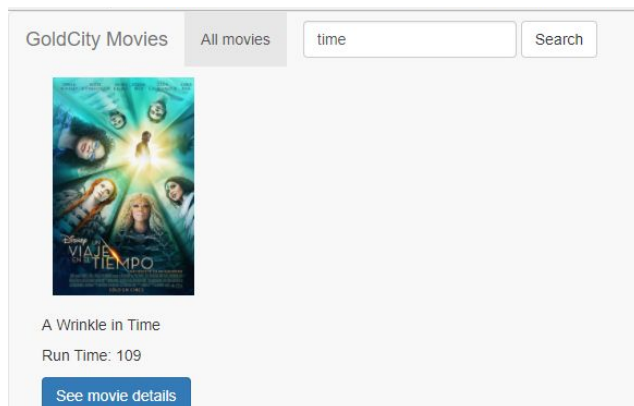


User (Customer) Side:

15. Search for movie/keyword in the header bar

```
SELECT title, run_time, plot_synopsis, Cover_img FROM Movie WHERE (title LIKE '%.$movie_title.%') OR (plot_synopsis LIKE '%.$movie_title.%');
```

Output: The movie you search shows up



16. Update customer's information

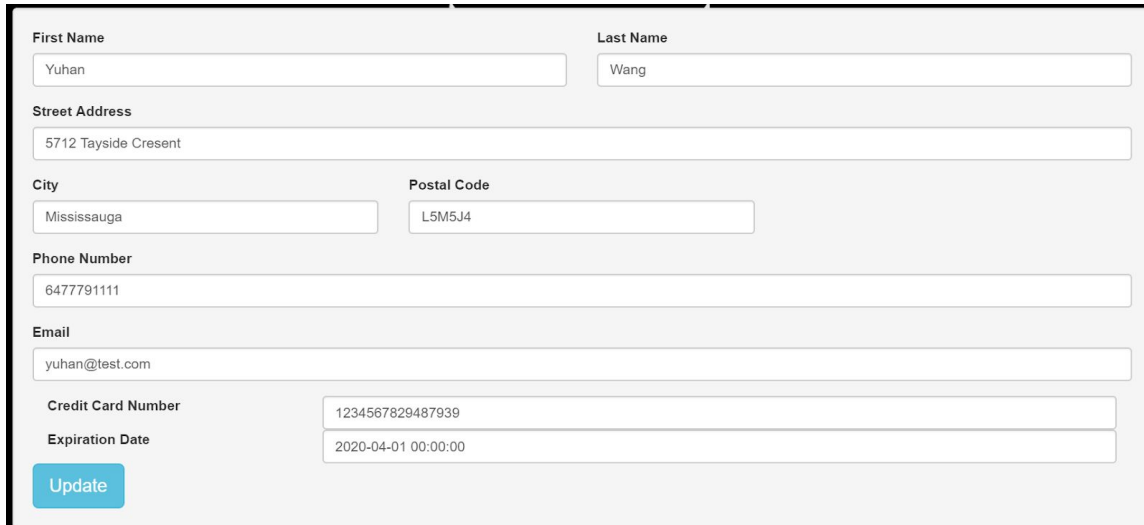
```
UPDATE customer SET first_name = '$fname.', last_name = '$lname.', street = '$street.', city = '$city.', pc = '$pc.', phone_number = '$phone_number.', credit_card = '$ccard.', card_expiry = '$card_expiry.', email = '$email.' WHERE account_number = '$account_number.';
```

Output: None, customer's information will be updated in the database.

17. Auto fill in customer's information from the database

```
SELECT first_name, last_name, street, city, pc, phone_number, email,
credit_card, card_expiry FROM customer WHERE account_number =
".$SESSION['account_number'].";
```

Output: Information is auto-filled in the input fields



The screenshot shows a web form with the following fields and values:

Field	Value
First Name	Yuhan
Last Name	Wang
Street Address	5712 Tayside Crescent
City	Mississauga
Postal Code	L5M5J4
Phone Number	6477791111
Email	yuhan@test.com
Credit Card Number	1234567829487939
Expiration Date	2020-04-01 00:00:00

An "Update" button is located at the bottom left of the form.

18. In the user login page, select all the attributes from the account (email) the user enters.

```
SELECT * FROM Customer WHERE email='$email';
```

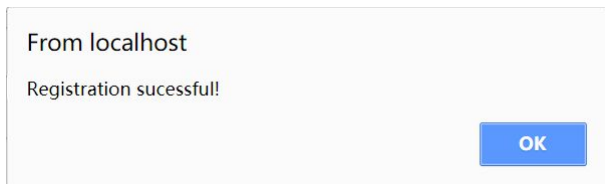
19. In the admin login page, select all the attributes from the account(email) the user enters.

```
SELECT * FROM Admin WHERE email='$email';
```

20. Register new user account

```
INSERT into customer VALUES ('$account_number', '$password',
'$first_name', '$last_name', '$street', '$city', '$pc',
'$phone_number', '$email', '$credit_card', '$card_expiry');
```

Output: New account information is inserted into the database and a pop-up message appears.



21. Insert customer's review and star ratings into the database

```
INSERT INTO `rates` VALUES ('$account_number', '$movie_title',  
$star_rating, '$review');
```

22. Select the average star rating of a movie on the movie information page

```
SELECT AVG(star_rating) FROM rates WHERE movie_title =  
'".$movie_title."' GROUP BY movie_title;
```

23. Show the reviews of a movie on the review page

```
SELECT first_name, city, star_rating, review FROM rates JOIN  
customer ON customer_account = account_number WHERE movie_title =  
'".$movie_title."';
```

Output:

Beauty and the Beast

8/10 Bill from Toronto

From the poor provincial town to the dazzling ballroom, the movie is a larger-than-life feast for the eyes. Emma Watson is Belle is smart, progressive and not a princess.

5/10 Jeffrey from Toronto

As a whole, the production doesn't hold a candle to Kenneth Branagh's lovely 2015 version of Cinderella, to say nothing of the 1991 Beauty original.

9/10 Javier from Toronto

Watson is an ideal Belle in this wonderful remake that's at once nostalgic and new, bringing to life the musical both for kids and life-long adult fans.

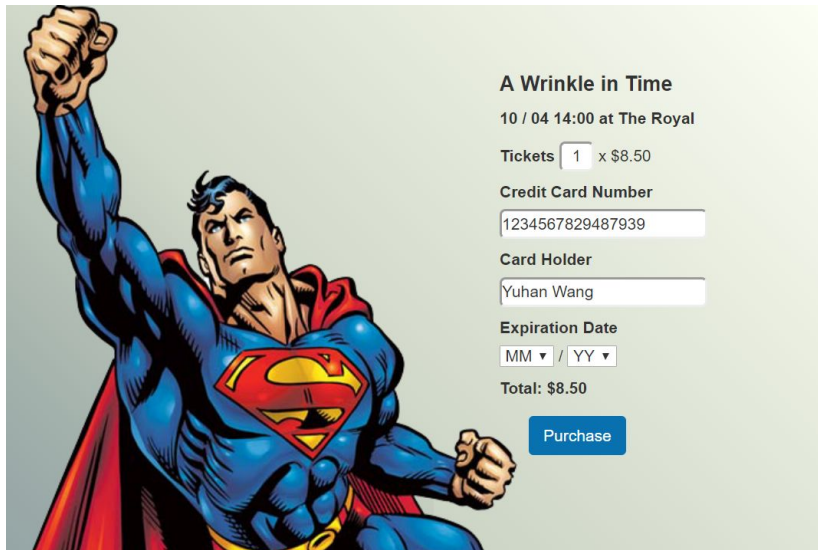
9/10 Ester from Toronto

A touching, eminently watchable, at times slightly awkward experience that justifies its existence yet never totally convinces you it's a movie the world was waiting for.

24. Auto-fill in the fields (not including the expiration date) in the ticket order page

```
SELECT first_name, last_name, credit_card FROM customer WHERE  
account_number = ".$SESSION['account_number'].";
```

Output:



A Wrinkle in Time
10 / 04 14:00 at The Royal

Tickets x \$8.50

Credit Card Number

Card Holder

Expiration Date
MM ▾ / YY ▾

Total: \$8.50

25. Select the movie title and number of seats left

```
SELECT movie_title, num_seats FROM showing WHERE start_time =  
'".$start_time."' and theatre_num = ".$theatre_num." and  
complex_name = '".$complex_name.'";
```

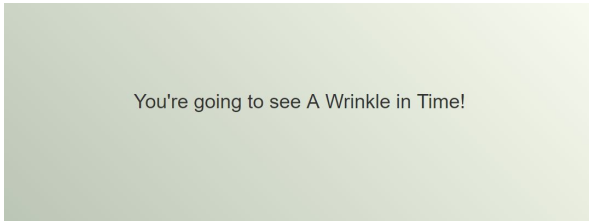
26. Update the number of seat left after a customer purchases tickets

```
UPDATE showing SET num_seats = num_seats - ".$numTickets." where  
start_time = '".$start_time."' and theatre_num = '".$theatre_num."'  
and complex_name = '".$complex_name.'";
```

27. Insert customer's ticket information into going_to table

```
INSERT INTO `going_to` VALUES ('$account_number', '$movie_title',  
'$start_time', $theatre_num, '$complex_name', $numTickets);
```

Output:




28. Select attributes of a movie from rates and movie tables

```
SELECT AVG(star_rating), run_time, pg_rating, plot_synopsis,  
production_company, start_date, end_date, director_fname,  
director_lname, cover_img FROM rates RIGHT JOIN movie ON movie_title  
= title WHERE title = '". $movie_title.'" GROUP BY movie_title;
```

29. Select actor's first name and last name

```
SELECT actor_fname, actor_lname FROM acts_in WHERE movie_title =  
'". $movie_title.'';
```

Output: (27, 28 combined)



Beauty and the Beast

☆☆☆☆☆☆☆☆☆☆

[Full Reviews](#)

PG

Duration: 129 min

Release Date: 2017-10-02 (CAN)

Director: Bill Condon

Starts: Dan Stevens, Emma Watson

30. Display the showing information of a movie

```
SELECT start_time, complex_name, num_seats, city, theatre_num FROM  
showing NATURAL JOIN theatre WHERE movie_title = '". $movie_title.'';
```

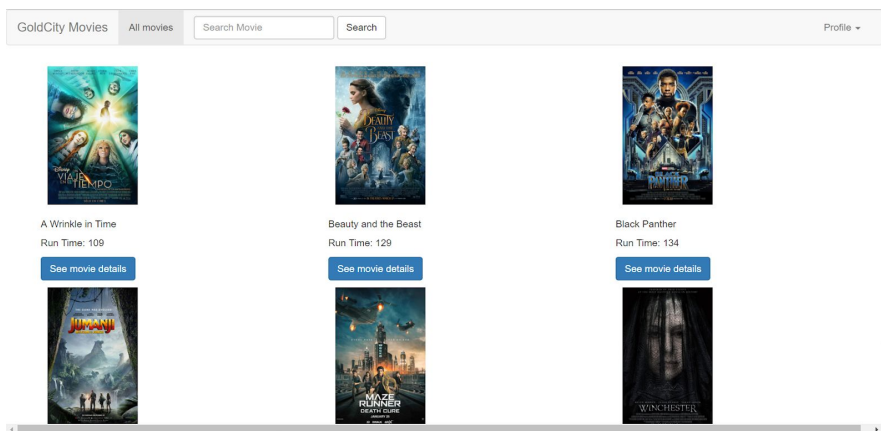

Output:



31. SELECT title, run_time, Cover_img FROM Movie

```
SELECT title, run_time, Cover_img FROM Movie;
```

Output: Movies are shown on the page home page.



32. Delete reservation

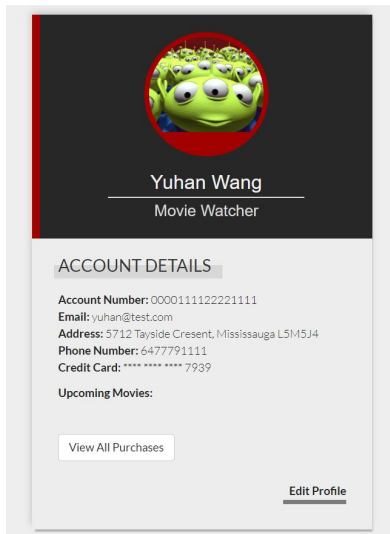
```
DELETE FROM `going_to` WHERE start_time = ' ".$start_time."' AND  
theatre_num = ' ".$theatre_num."' AND customer_account =  
' ".$account_number.'';
```

Output: Reservation is removed

33. Show profile information

```
SELECT first_name, last_name, street, city, pc, phone_number, email,  
credit_card FROM customer WHERE account_number =  
" ".$SESSION['account_number'].";
```

Output:

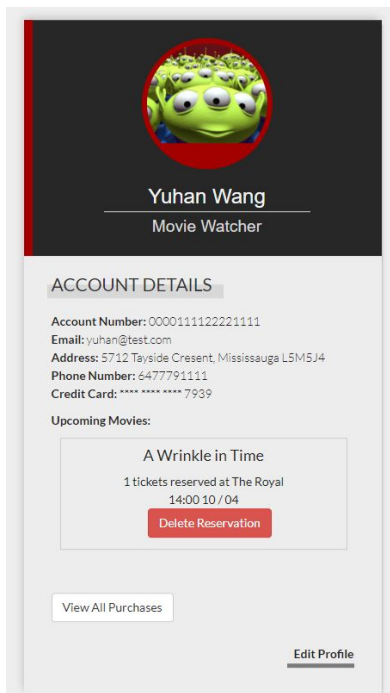


The image shows a user profile page for Yuhan Wang, a Movie Watcher. The profile includes a circular profile picture of a green alien character. Below the name and role, there is an 'ACCOUNT DETAILS' section with the following information: Account Number: 0000111122221111, Email: yuhan@test.com, Address: 5712 Tayside Crescent, Mississauga L5M5J4, Phone Number: 6477791111, and Credit Card: **** * 7939. There is also a section for 'Upcoming Movies' with a 'View All Purchases' button and an 'Edit Profile' link.

34. Show reservation on the profile page

```
SELECT movie_title, start_time, theatre_num, complex_name,  
tickets_reserved FROM going_to WHERE customer_account =  
"}.${_SESSION['account_number']}";
```

Output:



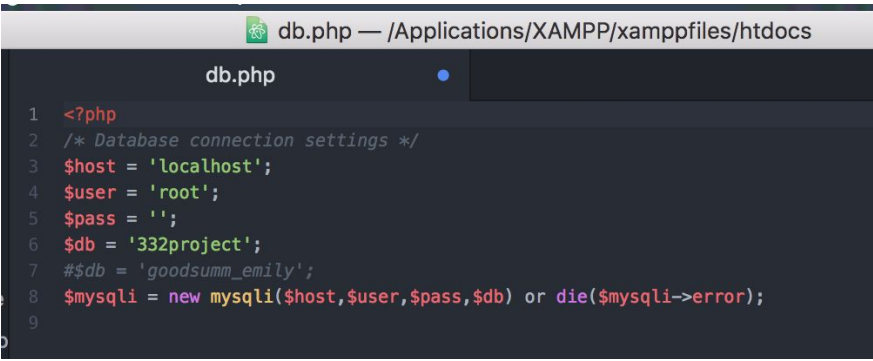
The image shows the same user profile page for Yuhan Wang, but with a reservation displayed under the 'Upcoming Movies' section. The reservation is for the movie 'A Wrinkle in Time' at 'The Royal' theatre, with 1 ticket reserved for 14:00 on 10/04. There is a 'Delete Reservation' button next to the reservation details. The 'View All Purchases' button and 'Edit Profile' link are also present.

Problems Encountered During Development

Writing the queries and implementing this project's visual was an educational process. One of the initial problems we encountered during the development phase was trying to connect to the database. Upon some research we figured out that the reason we could not connect was because the `mysql_connect()` extension has been deprecated, we are utilizing PHP 7, and to run database queries in PHP 7, we should use MySQLi.

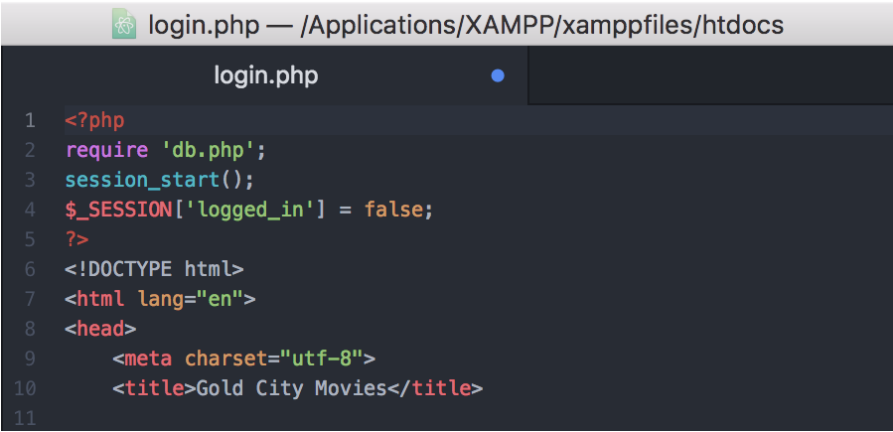
Ex. `$mysqli = new mysqli($host,$user,$pass,$db) or die($mysqli->error);`

We also found it repetitive to keep writing the full database connection in each of our scripts. A solution we found to this is to create the connection in one file (Figure 1), and require this file in the beginning of all php scripts that needs database reading/writing (Figure 2).



```
db.php — /Applications/XAMPP/xamppfiles/htdocs
db.php
1 <?php
2 /* Database connection settings */
3 $host = 'localhost';
4 $user = 'root';
5 $pass = '';
6 $db = '332project';
7 # $db = 'goodsumm_emily';
8 $mysqli = new mysqli($host,$user,$pass,$db) or die($mysqli->error);
9
```

Figure 1. Making connection to database



```
login.php — /Applications/XAMPP/xamppfiles/htdocs
login.php
1 <?php
2 require 'db.php';
3 session_start();
4 $_SESSION['logged_in'] = false;
5 ?>
6 <!DOCTYPE html>
7 <html lang="en">
8 <head>
9     <meta charset="utf-8">
10    <title>Gold City Movies</title>
11
```

Figure 2. Require the connection

Another problem we encountered was determining how to remember and verify whether a user was logged in while they moved from one page to another. This is important because a user who is logged in should be able to view their account information as well as purchase tickets and create reviews, but non-logged-in members should not have access to this functionality. We originally started with two pages, one as the logged-in view and one as the logged-out view. We soon realized that the problem with this implementation is that it is very insecure. A user can access the logged-in view if they had knowledge of our logged-in PHP file name. We switched to using a PHP session variable which stores information

across multiple pages until the user exits the application. At the start of all pages we used: `session_start()` which loaded the session variables previously set on other pages. We created a session variable: `$_SESSION['logged_in']` which would be set to `true` if the user logged-in successfully and `false` otherwise. When the `logged_in` variable is set to `true`, we grant the user access to certain pages. We checked this through an *if statement* at the top of our pages.

We also encountered problems with the SQL statements we wrote in our PHP files. There were some pages which did not give any errors, but no data was loaded from the database. We solved this by testing our queries on phpMyAdmin and printing the variables that we were inserting into the SQL queries on our webpage to ensure we were inserting the correct data.

Lastly, we found the hardest part of the project to be CSS debugging and browser compatibility. Sometimes, changes to our CSS file did not take effect in our browser or would work differently on separate pages. In some cases, it is because the CSS layout is already stored in cache and pre-processed by the browser. A resolution to this was to clear our browser's cache so that new CSS changes would show. Since we had multiple CSS files, we also ensured that the CSS links in our PHP files were ordered from lowest to highest priority. We also made sure to test our pages on different browsers such as Chrome, Safari, and IE to ensure our CSS would be compatible for most users.

Important design and implementation decisions

We continuously made small changes to improve our design. Some important design and implementation decisions include allowing users to only update theatre information for theatres that exists in the database. For example, a user cannot pick theatre number 5 for Alliance because it only contains 4 theatres or a user cannot pick theatre number 6 for Cineplex as it only has 5 theatres (Figure 3). This design structure populates the list of allowed response from users in the form of radio buttons and ensures the correctness of entry to our database. It also alleviate the need for an admin to go back and search through each complex to see how many theatres there are. This is a good implementation practice as it saves time for the user and protects from loss of integrity of data.

The figure shows two side-by-side screenshots of a web form titled "Update Theatre Information".

The left screenshot shows the form for the "Alliance" theatre chain. At the top, there are radio buttons for "Alliance" (selected) and "Cineplex". Below that are radio buttons for "The Royal". Under the heading "Theatre Number", there are four radio buttons labeled 1, 2, 3, and 4. Below that is a "Max Seats" label and a text input field containing the word "number". Under "Screen Size", there is a text input field containing "s / m / l". At the bottom is a green "Update" button.

The right screenshot shows the form for the "Cineplex" theatre chain. At the top, there are radio buttons for "Alliance" and "Cineplex" (selected). Below that are radio buttons for "The Royal". Under the heading "Theatre Number", there are five radio buttons labeled 1, 2, 3, 4, and 5. Below that is a "Max Seats" label and a text input field containing the word "number". Under "Screen Size", there is a text input field containing "s / m / l". At the bottom is a green "Update" button.

Figure 3. Updating theatre information

Another important design decision is the format of expiry date of credit card on the registration page. It auto divides the data and parses it into DD/MM/YY format when our PHP script grabs the POST data. This is very useful for our implementation, as it resolves problems such as month and date confusion with other formats of date entry (Eg. 2018-01-20, 01/20/2018). This ensures the integrity of the data by asking users to enter date in a specific predefined form that we can measure.

Since we want to minimize the amount of people who have access to the Admin page, we did not create an admin registration page. We believe that too much control from different sources can be messy. Having a list of admin accounts pre-created with usernames and passwords given to the user would be a better practice for users with elevated privileges to manage the system. This protects the confidentiality of the data because admin accounts have access to member information which regular users should not be able to see.

The last important design and implementation decision is letting our edit profile page auto fill the elements of all member information into the text label. This makes the flow of the website very natural and intuitive, so the users don't need to think what to do next when they want to edit or add to their profile information.

Tools Used in Development

The front end of our project consists of HTML, CSS, Bootstrap and Javascript. We started with simple HTML pages that contained the core content of the page. After all the core content was written out, we used Bootstrap as our UI framework base to deliver good layout display and match the need of current browser trends. CSS was then used on top of Bootstrap when we wanted to modify, remove, or add to the elements and styles from Bootstrap that we didn't like. CSS was used to separate style from structure and content.

We learned during the development of the project that having a framework like Bootstrap is very useful. Although there was a bit of a learning curve at the beginning, we felt that by the end of the project, not only did Bootstrap help us create consistent, minimalistic design layouts, it also saved us the effort and time of writing out the pages completely from scratch.

We hosted our PHP scripts through XAMPP - a free, easy to install apache distribution containing MySQL, PHP and Perl, and used MySQL and PHP to query statements from our database. PHP is a server-side scripting language designed for web development, but can also be used as general purpose programming language. We enjoyed having XAMPP on our computers because it was simple to add our files to the htdocs folder and be able to run our PHP code on our computer like a server. phpMyAdmin was helpful in checking our queries to ensure they were written correctly, and returned our desired results before adding them to our PHP documents. We learned that testing our SQL queries on phpMyAdmin first was a lot faster than trying to figure out the errors after running our application.

On some of our pages, we used JavaScript to create more dynamic, user friendly content. For example, on our member profile page, initially only upcoming reservations will be visible. We have already loaded all the information onto the page from our database query, but have made the div with past reservations hidden. Using JavaScript enabled us to have a button on our page which when clicked calls on a JavaScript function to make the div visible. Another example is on the ticket purchase page; the total price is updated

dynamically as the user enters how many tickets they would like to buy. JavaScript also allowed us to check that the entered value was a valid integer, and was less than the number of seats left for the showing.

Since our group members have different operating systems, we utilized two different IDE environments. Scripts on windows were written using Notepad++ and scripts on MAC were written in ATOM. We choose these two editors because of their minimalist nature and also because they can handles large amount of files very well. They are both free editors that supports several programming languages.

Our project is maintained through Github. We choose to use Github because it is a great way for us to collaborate and see each other's code. This way, we could ensure that all of our code worked well together and revision on one part did not cause unwanted results on another part. Github allowed us to handle and maintain code with the ability to undo any unwanted actions. It helped us maintain best code practices as well as showcase our recent work through our online profile.

Lessons Learned and Enhancements

During this project, our group felt we worked very well together. As a group we decided to start working on the project early. However, by getting an early start, the first time we met to collaborate on the web design, we were a bit scattered on where to start.

Individually, when writing our SQL statements, we should have created the query statements before developing each webpage instead of trying to call the database with incorrect statements that did not show up in our UI. Had we created these query statements earlier though phpMyAdmin, we would have spent less time figuring out our errors in our PHP scripts.

If we were to change anything, we would have liked to have more facts displayed on our admin home page, as well as more security for passwords. The admin homepage currently only has two facts - the most popular complex with the amount of tickets sold in that complex, and the most popular movie with the amount of tickets sold for that movie. Given more time, we would have liked to create better visual representation of the data with more insight. Eg. Histogram chart that displays the amount of tickets sold for each complex, the age and gender group for each movie, the top movies from each complex and more.

The password is currently stored directly in the database as plain-text. We understand that in the real world, the most important aspect of a user account system is the confidentiality of user data. User account databases are hacked frequently, so with more time, we would like to protect our user passwords by employing salted password hashing (with phpass) if our website is ever breached. We protect confidentiality currently by only allowing administrators to view member data (not including credit card data), but we would like to add PHP functions which better sanitize and validate form field data to prevent SQL injection.

As future developers working on a large scale project, we learned from this project that documentation in code is very important. The documentation and comments for each script should be clear from the beginning of the development. This will ensure that when other team members work on the code, there won't be any miscommunication. Having good documentation also makes our code easy to maintain. As we worked on this project over

several weeks, our comments helped us to remember what our code was doing and improved readability.

Overall it was a very fun experience to develop the Movie ticket website, and it gave us insight on how to develop, design, retrieve data across the database and design a platform to showcase it.