AY 13/14 | SEM1

# CS2102 Project Report 1

Movie Ticket-Booking System & Database

**GROUP 42:** Lim Binjie Benjamin, Yang Xiaoyu, Yap Zi Xuan

## Project Overview

Our team is developing a movie ticket-booking system, Popcorn, for a local cineplex company. The company owns multiple cinemas across the country, and each cinema has multiple halls where movies are screened.

This booking system will comprise of 2 interfaces - one for customers, ticket bookers in this case, and the other for the cineplex's site admin:

1. Customer side: the former interface allows customers to create user accounts to make single and group bookings, make online payments* and cancel bookings prior to payment

2. Admin side: employees of the cineplex are able to login to access an admin panel that would allow them to insert, modify and remove movies, show timings. Other operations that the admin could perform include the ability to view sales data and retrieve booking information specific to a customer.

## Design Specifications

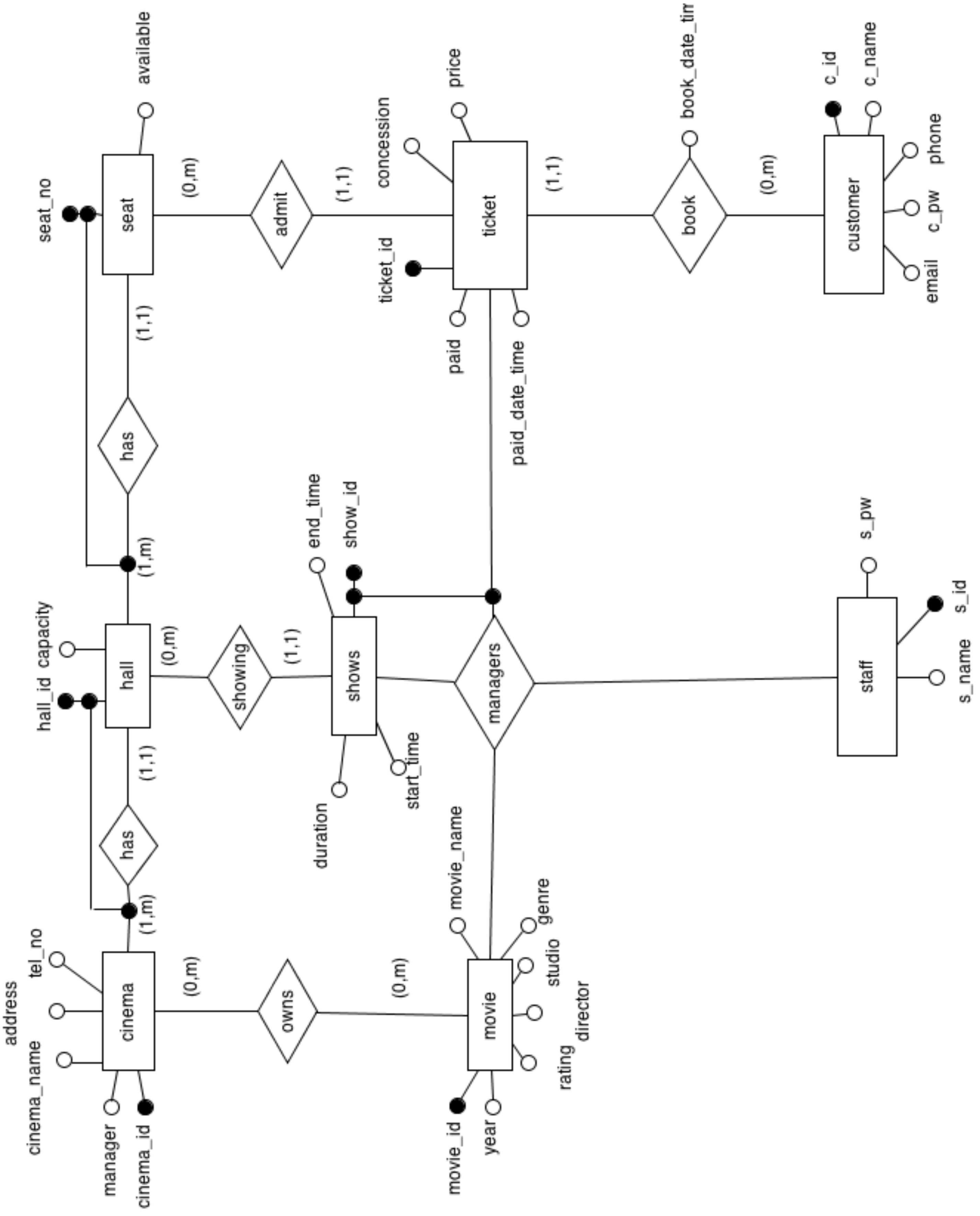
We are developing Popcorn from the following tools:

| | |
|---|---|
| Web Server: | Apache 2.4 |
| Web Server Languages: | HTML5, CSS3, PHP, JavaScript |
| DBMS: | MySQL 5.6.12 + phpmyadmin extension |

## Core Features

| Customer | Admin |
|---|---|
| 1) book multiple tickets for the same show in one booking | 1) view tickets booked by a certain customer with a particular customerID |
| 2) choose to pay online or pay at the cinema counter | 2) look at aggregate sales, or sales according to set categories (e.g. movieID) |
| 3) modify or delete bookings prior to payment | 3) insert, delete and update movies, show timings and hall locations |
| | 4) update 'paid' status of tickets (to reflect paid tickets during counter payments) |

# ER Diagram



seat
available
seat_no
(0,m)
admit
(1,1)
ticket
concession
price
ticket_id
paid
paid_date_time
(1,1)
book
book_date_tim
(0,m)
customer
c_id
c_name
phone
c_pw
email
(1,1)
has
(1,m)
hall
hall_id
capacity
(0,m)
showing
(1,1)
shows
end_time
show_id
duration
start_time
managers
staff
s_pw
s_id
s_name
(1,1)
has
(1,m)
cinema
address
tel_no
cinema_name
manager
cinema_id
(0,m)
owns
(0,m)
movie
movie_name
genre
studio
director
rating
movie_id
year

## Relation Schema

Here is our database schema in SQL DDL code:

```
DROP TABLE IF EXISTS ticket;
DROP TABLE IF EXISTS staff;
DROP TABLE IF EXISTS seat;
DROP TABLE IF EXISTS customer;
DROP TABLE IF EXISTS shows;
DROP TABLE IF EXISTS hall;
DROP TABLE IF EXISTS movie;
DROP TABLE IF EXISTS cinema;


CREATE TABLE cinema(
     cinemaID INT NOT NULL AUTO_INCREMENT CHECK (cinemaID
> 0 AND cinemaID < 1000) ,
     cinemaName VARCHAR(64) NOT NULL UNIQUE,
     address VARCHAR(256) NOT NULL,
     telNo INT,
     manager VARCHAR(256) NOT NULL,
     PRIMARY KEY(cinemaID)
) AUTO_INCREMENT=1;


CREATE TABLE movie(
     movieID INT NOT NULL AUTO_INCREMENT CHECK (movieID >
2000 AND movieID < 3000) ,
     movieName VARCHAR(256) NOT NULL,
     year DATE NOT NULL,
     genre CHAR(32),
     studio VARCHAR(256) NOT NULL,
     director VARCHAR(256),
     rating VARCHAR(4) NOT NULL CHECK (rating = ‘G’ OR
rating = ‘PG’ OR rating = ‘PG13’ OR rating = ‘NC16’ OR
rating = ‘M18’ OR rating = ‘R21’),
     posterLink VARCHAR(256),
     PRIMARY KEY(movieID)
)AUTO_INCREMENT=2001;


CREATE TABLE hall(
     hallID INT NOT NULL AUTO_INCREMENT CHECK (hallID >
1000 AND hallID < 2000) ,
     cinemaID INT,
     capacity INT CHECK(capacity > 0),
     FOREIGN KEY(cinemaID) REFERENCES cinema(cinemaID) ON
UPDATE CASCADE ON DELETE CASCADE,
     PRIMARY KEY(hallID)
)AUTO_INCREMENT=1001;
```

```
CREATE TABLE shows(
     showID INT NOT NULL AUTO_INCREMENT CHECK (showID >
3000 AND showID < 4000) ,
     movieID INT,
     hallID INT,
     duration INT,
     startTime DATETIME NOT NULL,
     endTime DATETIME NOT NULL,
     FOREIGN KEY(hallID) REFERENCES hall(hallID) ON
UPDATE CASCADE ON DELETE CASCADE,
     FOREIGN KEY(movieID) REFERENCES movie(movieID) ON
UPDATE CASCADE ON DELETE CASCADE,
     PRIMARY KEY(showID)
)AUTO_INCREMENT=3001;


CREATE TABLE customer(
     cID INT NOT NULL AUTO_INCREMENT CHECK (cID > 4000
AND cID < 5000) ,
     cName VARCHAR(256) NOT NULL,
     email VARCHAR(128) NOT NULL UNIQUE,
     cPw VARCHAR(64) NOT NULL,
     phone INT NOT NULL,
     PRIMARY KEY(cID)
)AUTO_INCREMENT=4001;



CREATE TABLE seat(
     seatNo CHAR(3),
     showID INT,
     hallID INT,
     available BOOLEAN NOT NULL DEFAULT 1,
     PRIMARY KEY(seatNo, showID),
     FOREIGN KEY (hallID) REFERENCES hall(hallID) ON
UPDATE CASCADE ON DELETE CASCADE,
     FOREIGN KEY(showID) REFERENCES shows(showID) ON
UPDATE CASCADE ON DELETE CASCADE
);


CREATE TABLE staff(
     sID INT NOT NULL AUTO_INCREMENT CHECK (sID > 5000
AND sID < 6000) ,
     sName VARCHAR(256) NOT NULL,
     sPw VARCHAR(64) NOT NULL,
     PRIMARY KEY(sID)
)AUTO_INCREMENT=5001;
```

```
CREATE TABLE ticket(
     ticketID INT NOT NULL AUTO_INCREMENT CHECK (ticketID
> 6000 AND ticketID < 7000) ,
     cID INT,
     showID INT,
     movieID INT,
     seatNo CHAR(3) NOT NULL,
     price INT NOT NULL,
     concession CHAR(5) NOT NULL DEFAULT 'ADULT', CHECK
(concession = 'CHILD' OR concession = 'ELDER' OR
concession = 'ADULT') ,
     paid BOOLEAN NOT NULL DEFAULT 0,
     bookDateTime DATETIME NOT NULL,
     paidDateTime DATETIME,
     FOREIGN KEY(cID) REFERENCES customer(cID) ON UPDATE
CASCADE ON DELETE CASCADE,
     FOREIGN KEY(showID) REFERENCES shows(showID) ON
UPDATE CASCADE ON DELETE CASCADE,
     PRIMARY KEY(ticketID)
)AUTO_INCREMENT=6001;
```

## Schema Notes*

1) In the Cinema, Hall, Movie and Shows relations described in the schema above, a boundary check is performed on their respective primary keys, such as this:

```
cinemaID INT(6) NOT NULL AUTO_INCREMENT CHECK
(cinemaID > 0 AND cinemaID < 1000),
```

Here, in the cinema relation, we are assuming that there will never be more than 999 cinemas existing in the database. The attribute has to be an INT type (instead of CHAR), so as to accommodate auto-incrementation. Each cinemaID has an auto-incremented value starting from 1, with the maximum value being 999. The key assumptions and rationale for the check applies similarly to the movie, show and hall relations.

2) In the customer relation, cID refers to customer ID, which also serves as the customer's username. cPW refers to their password. Both cID and cPW are required to login to their user account in Popcorn.

Similarly, for the staff relation, sID and sPW refer to the cineplex employees' username and password that is needed to access the admin panel.