# **ERD Thought Process:**

We were thinking about the key parts of the Cinema System in interaction with its potential users and from this wet agreed upon the key parts of the specification:

Screen:

This role will store the details of the screens inside the cinema building, such as its screen number, seat capacity and how many of its seats are premium.

Film:

This role will keep track of the films the cinema has on offer with attributes such as the name, release date and audience age rating.

Showing:

This will keep track of all the films that are currently on show and what screens they are shown in. This will allow the customers to know where to go when they purchased a ticket and what time to go for their preferred film.

Ticket:

This table will keep track of the information that’s unique to each ticket, storing it under a unique ID, this includes whether it’s part of a booking or not, if it’s premium and whether or not it’s been sold yet. Seat numbers are also included to keep track of the seat it refers to.

It will also keep reference to which showing it grants access to.

Booking:

This table contains information on bookings made in advance for one or multiple tickets at a time, they can be made online or over the phone so the customers name and credit card number are recorded to verify identification upon collection.

After deciding the above were key information, which the group set as Tables, we decided to look for more key information inside the specification. This is to see what information we would need to make the system work. We looked for potential fields associated with the Tables we decided on. We also discussed what problems we may come across in the system that were not specified in the document, and how we could resolve it in our design.

For example, in our Film table, each film has a name, a release date, a duration, a priority rating, a film audience rating, a date where the cinema starts showing the film and a finish date. One problem with this is that there are a number of films that, over the years, have been redone or re-released with updated visuals and/or a brand new plot. So we decided to give each film an ID so we have a way of telling the difference between each film.

After looking into the specification to see what we can set as fields for each table, with every one of them having a primary key associated with them, we decided on what relationships to create between each table. To tackle this part of the assignment, we went through the table that appeared to have the least dependencies on other tables to do its tasks.

What we came up with was the following:

Film has a one to many relationship with Showing.

Showing has a one to one relationship with Film.

Screen has a zero to many relationship with Showing.

Showing has a one to one relationship with Screen.

Showing has a one to many relationship with Ticket.

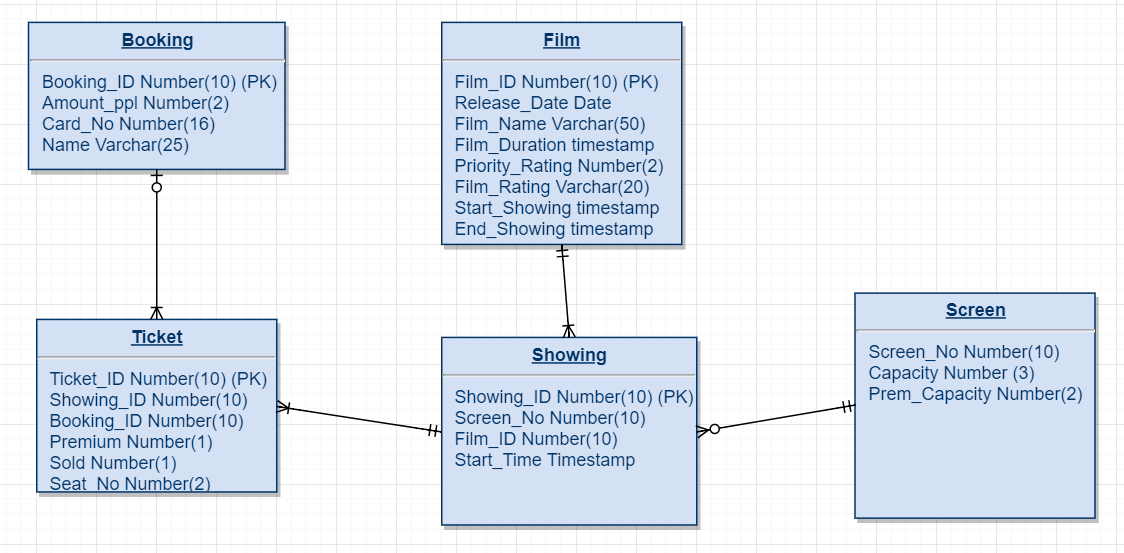
Ticket has a one to one relationship with Showing.

Booking has a one to many relationship with Ticket.

Ticket has a zero to one relationship with Booking.

Once all that was completed, we went through each variable to see how we would like to store each piece of information.  For example we agreed that names in each table would be stored as VARCHAR(25) or Strings, as that is necessary. In case of some ambiguous areas, we discussed how would we decide to store them (Ie. whether we would store ID’s as integers or as Strings).

In some cases, we came across problems associated with the input of the datatypes when we tried to execute the the design into the schema. We were able to patch it up quickly due to our overall design of the system and knowledge of alternative methods to get access to the same information and store it in the database. Once we were happy with our design, we sketched up a few different ERD’s, of which our final draft is laid out below:



The only thing left to do is to test the system. We connected to the appropriate schema, created the relational database and inserted random information, which we got from the spec document assigned to us or made up ourselves. Once the information was stored, we could test the attributes, and the relationships we had implemented, to insure their validity.

After testing the system, we discussed roles to do with the system and what they can do. The three roles we agreed that are associated with this system is the manager, a cashier staff and a user that wants to create a booking either online, or over the phone.

After making sure we understood the roles and the privileges associated with them, we discussed what roles each member of the team are to adapt to complete their individual part of the assignment. Once we finished that, we executed the sql code to allow all the roles and the system to exist with information stored.

# Transaction Thought Process

The privileges believed to be associated with these members are: The person can see everything but can only book. The staff cashier can go a step further and update the tickets or refund the ticket, as well as cancel the booking.

The manager has the decision to allow what film gets shown, what screens are displayed and even accept new films. So the privileges associated with the manager is superior to the other two. The only thing the manager cannot do is drop the systems tables. The manager may need to get access to previous information so the manager cannot delete information either, but is entitled to update the information.

Since I was assigned role of the manager, I knew I could get creative with the system. I decided to leave the booking parts of the system to my other teammates and decided to see what I could do with the other three systems. I decided to write a function that would take in a new film into the system. Since I was not writing to my own schema but a shared one, I couldn’t find a way to get access to the datatypes inside the fields. So, I got creative and used the inbuilt ones such as VARCHAR.

I would take input for the name, film rating, the film length, the start show date, the end show date and the released date. Apart from the film length, which I decided would be number(3), all of the fields was varchar. In the case of the date fields, I chose varchar to make it easier for conversion to either date to timestamp variables. I would let the system generate the ID and the priority rate in this version of the function. Once I get all the information, I would insert the new film into the database and commit immediately so the session does not get locked immediately and the others can see the updated information.

# Queries Thought Process

When it came to the queries, (with the group) went to make sure we all understood the terminology with the sql queries. After that we went to do our own approaches. Every had permission to view the tables, so it shouldn’t be a problem. After doing each query, I would think about which table I should work with for each query. In many cases there was multiple choices, so I tried to think of one that was not too obvious. Once I completed each part, I added comments at the top to explain what I was trying to achieve so people reading can understand what I was doing.

I also used some functions for the visibility to be more appealing rather than constantly long useless info or nulls, such as TO\_CHAR or NVL(). This is so the information that has been retrieved inside the system to be displayed makes sense of what we tried to accomplice inside the system we had designed.