# Project Description Web-Technologies WS 22

As you have mastered web development fundamentals, you are now asked to implement an online system for the local cinema.
From a customer's perspective going to the cinema is described as follows:
Currently, tickets can only be bought offline at the cinema shop. A customer buys several tickets for a specific movie at a specific date and time and for one seat in a specific theater. A cinema worker has to check for each customer if a requested number of seats is available in a specific row, and if not, provide alternative seats. If the customer is happy with the provided seats, he/she pays the ticket price (either in cash or with a card). After the payment, the customer receives a number of printed tickets. Right before entering the cinema, a cinema worker checks if the customer has a valid ticket (by scanning the QR code).

While this approach is typically accepted, it imposes several problems:
- The process is labor-intensive
- Customers often have to wait a long time in a queue before they can buy tickets
- Therefore, customers have to come in advance before the movie starts
- Printing tickets is not resource friendly

We, therefore, aim to automate this process:
The customers should be able to buy tickets either remotely (e.g., from their smartphones) or locally via special terminals. In both cases, the customer uses the same web service interface.

With this interface, the customer must be able to:

- Browse a list of movies currently available,
- see for each movie the time and date on which it is shown,
- check which seats are currently available for a specific movie at a specific date and time,
- buy tickets,
- make a user account to see bought tickets, request QR codes, and write reviews.

After payment, the customer receives a QR code on the smartphone. Upon entering the respective cinema, the customer is asked to scan the QR code at the entrance.

## Teams

Each team consists of up to 5 students responsible for **both** the frontend and backend of the application.
The application has to provide the following functionality via its frontend:

**1. Management view**

From the management view, cinema management can:

a) Create, update, and delete theaters:
 · Each theater has a number of seats
 · Each seat has a unique number and row in a specific theater
 · Seats can be either normal seats, deluxe seats or removable (for wheelchairs)
 · Different theaters have different features (3D, 4D, Dolby Atmos...)

b) Create, update and delete movies
 · Each movie has a name, description, duration, and a minimum age
 · Each movie also has a set of customer reviews that must be updated regularly

c) Create and manage the schedule
 · Each movie can be shown multiple times in different theaters and at different times and dates

d) Sell tickets
 · Avoid clashes (it should not be possible to book the same seat at the same time twice)
 · The ticket price depends on the factors like time, date, duration, theater, seat type, and row
 · It should be possible for a customer to return the bought ticket 1 hour before the movie starts. If that happens, that seat must be made available for customers again.

**2. Customer View**

From the customer view, the customer can:

a) Browse current movies and check the reviews and the rating (the number of stars)
b) Check which seats are available for a specific movie
c) Provide a graphical view of the seating plan
d) Buy tickets
e) Manage bought tickets
 · return tickets (until some deadline), see above
 · Request QR code
f) Write reviews and give points to contribute to the rating (only for watched movies)

It is recommended to make at least one person responsible for backend, one – for the management view, and one – for the customer view.

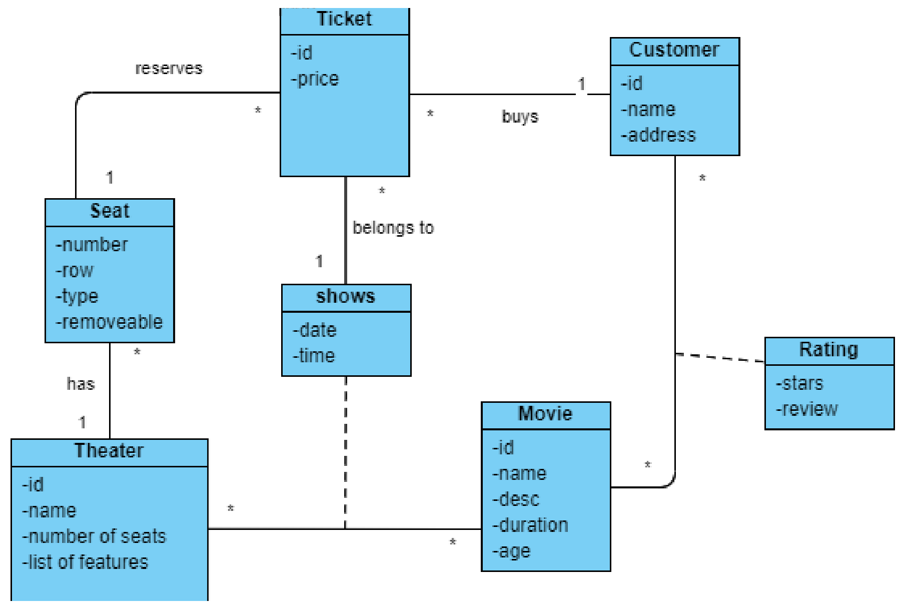
## Requirements for all Teams

A. The application should
 a) store the data in the database (the usage of PostgreSQL is recommended)
 b) implement backend in Node.js, expose backend functionality via the REST API
 c) use Angular for the frontend, access the REST API from the Angular code
 d) support user authentication
 e) allow the users to see only their own data
 f) be responsive and have a coherent UI
 g) not be easily hackable (use prepared statements and filter all inputs, details on the lecture session on security).

B. The given requirements are just starting points. If additional functionality is required for the successful execution of the entire process, it must also be included. If you propose alternative functionality for the same application, this can be negotiated with the teacher.

C. Be sure that the application for customers can be intuitively used on mobile phones.

## Example database schema


Ticket
-id
-price
Customer
-id
-name
-address
reserves
*
1
*
buys
1
*
belongs to
1
*
Seat
-number
-row
-type
-removeable
shows
-date
-time
Rating
-stars
-review
*
has
1
Theater
-id
-name
-number of seats
-list of features
Movie
-id
-name
-desc
-duration
-age
*
*
*