**Ticket Booking App**

This is a practice problem for you, and we'd love to see your solutions. You can share how you would design such a system in the discuss section. You can use draw.io, Excalidraw, or Google Diagrams to share your system design for this.

Problem Description

You need to build a Ticket booking app, for booking your seats for some movies, live shows, concerts, etc. We'll focus mainly on the user's and the theatre manager's experience. The user, here is a customer who wants to book the tickets for some movie let's say.

Some of the **functional requirements** that we know are:

Users should be able to:

1. Search for a movie or show by name.
2. See which movie theatres are showing the movie.
3. Choose from the available seats, and pay for the ticket.

The movie theatres should be able to:

1. See which seats are blocked and how many are available.
2. Reserve some seats for their staff.
3. Add new movies/shows that they would like to show in the future.

Feel free to add more features.

This needs to scale well, but we'll keep the non-functional requirements open-ended, and you can choose the scale/availability requirements by yourself.

Design Choices

Multiple Choice Question

In what parts of this system would you need to have immediate consistency when storing the data?

Adding new shows

Searching for newly added shows

Blocking seats

Making the payment

RedoSubmit

Correct.

Blocking the seats and payment would be individual systems, but within their own contexts, they should be immediately consistent, because you wouldn't want a situation where a user has blocked a seat, paid for it, and it's still available for other users.

Multiple Choice Question

What kind of a data store is best suited to power the search by movie names?

MySQL

Solr

Hadoop

Elasticsearch

RedoSubmit

Correct.

Solr and Elasticsearch have excellent text search capabilities and provide fuzzy search which could be valuable here.

Multiple Choice Question

What information would you want to cache in this system

Details of movie theatre, like name, address, etc.

Details about the movie, like cast, crew details, trailer link, etc.

Information about available seats

Price of the ticket

RedoSubmit

Correct.

Things that are usually constant, like details of the movie theatre, are good candidates for caching. Price and information about which seats are available can change very often can be made fuzzy by adding a surge component, which makes both of them bad candidates for caching.

Multiple Choice Question

When a user wants to block a seat, should you reserve the seat before the payment or after the payment?

Before the payment

After the payment

Either option can be done

RedoSubmit

Correct.

You can choose either of the approaches here, there is no right answer, and it depends on how you design the user journey. Usually the blocking seats before talking the payment works well in most cases, but in scenarios where you see very high payment failure rate from certain geographies/users, it's better to take the payment first and then block the seats.

Multiple Choice Question

When a seat is blocked, we might need to notify the movie theatre as well, in case they want to do something at that point. Should that notification be sent Synchronously or Asynchronously?

Synchronously

Asynchronously

RedoSubmit

Correct.

Since blocking the seat does not depend on Notification, we can make this interaction Asynchronous.

Multiple Choice Question

If you want to publish company-wide reports, like how much revenue is generated in a day, or which is the most-watched movie, which data store would you want to use to power these reports?

Orders Database

Some kind of a cache

Data Warehouse

RedoSubmit

Correct.

Usually, a data warehouse like Hadoop is best suited for offline reporting and analytics.

Is the Design Extensible?

Can you extend this design to provide search capability by the names of actors or directors?