

# Grokking the System Design Interview

(/collection/5668639101419520/56490502)

87% completed

Q

Search Course

- (/courses/grokking-the-system-design-interview/B8R22v0wqJo)
- Designing Twitter  
(/courses/grokking-the-system-design-interview/m2G48X18NDO)
- Designing Youtube or Netflix  
(/courses/grokking-the-system-design-interview/xV26VjZ7yMI)
- Designing Typeahead Suggestion  
(/courses/grokking-the-system-design-interview/mE2XkgGRnmp)
- Designing an API Rate Limiter  
(/courses/grokking-the-system-design-interview/3jYKmrVAPGQ)
- Designing Twitter Search  
(/courses/grokking-the-system-design-interview/xV9mMjj74gE)
- Designing a Web Crawler  
(/courses/grokking-the-system-design-interview/NE5LpPrWrKv)
- Designing Facebook's Newsfeed  
(/courses/grokking-the-system-design-interview/gxpWJ3ZKYwl)
- Designing Yelp or Nearby Friends  
(/courses/grokking-the-system-design-interview/B8rpM8E16LQ)
- Designing Uber backend  
(/courses/grokking-the-system-design-interview/YQVkj548NM)
- Design Ticketmaster (\*New\*)  
(/courses/grokking-the-system-design-interview/YQyq6mBKq4n)
- Additional Resources  
(/courses/grokking-the-system-design-interview/JYILJB6DK7g)

## Glossary of System Design Basics

- System Design Basics  
(/courses/grokking-the-system-design-interview/B892KY261z2)
- Key Characteristics of Distributed Systems  
(/courses/grokking-the-system-design-interview/YQWGjZZVz9)
- Load Balancing  
(/courses/grokking-the-system-design-interview/...

# Design Ticketmaster (\*New\*)

Let's design an online ticketing system that sells movie tickets like Ticketmaster or BookMyShow.

Similar Services: bookmyshow.com, ticketmaster.com

Difficulty Level: Hard

We'll cover the following

- 1. What is an online movie ticket booking system?
- 2. Requirements and Goals of the System
- 3. Some Design Considerations
- 4. Capacity Estimation
- 5. System APIs
- 6. Database Design
- 7. High Level Design
- 8. Detailed Component Design
- 9. Concurrency
- 10. Fault Tolerance
- 11. Data Partitioning

## 1. What is an online movie ticket booking system?

A movie ticket booking system provides its customers the ability to purchase theatre seats online. E-ticketing systems allow the customers to browse through movies currently being played and to book seats, anywhere anytime.

## 2. Requirements and Goals of the System

Our ticket booking service should meet the following requirements:

### Functional Requirements:

- Our ticket booking service should be able to list different cities where its affiliate cinemas are located.
- Once the user selects the city, the service should display the movies released in that particular city.
- Once the user selects a movie, the service should display the cinemas running that movie and its available show times.



PK

educative

(/learn)

Explore

(/explore)

Tracks

(/tracks)

My Courses

(/mycourses)

Edpresso

(/edpresso)

Refer a Friend

(/refer-a-friend)

Grokking the System Design Interview

(/collection/5668639101419520/56490502)

87% completed

Search Course

Designing Twitter

(/courses/grokking-the-system-design-interview/B8R22v0wqJo)

Designing Youtube or Netflix

(/courses/grokking-the-system-design-interview/m2G48X18NDO)

Designing Typeahead Suggestion

(/courses/grokking-the-system-design-interview/xV26VjZ7yMI)

Designing Typeahead Suggestion

(/courses/grokking-the-system-design-interview/mE2XkgGRnmp)

Designing an API Rate Limiter

(/courses/grokking-the-system-design-interview/3jYKmrVAPGQ)

Designing Twitter Search

(/courses/grokking-the-system-design-interview/xV9mMjj74gE)

Designing a Web Crawler

(/courses/grokking-the-system-design-interview/NE5LpPrWrKv)

Designing Facebook's Newsfeed

(/courses/grokking-the-system-design-interview/gxpWJ3ZKYwl)

Designing Yelp or Nearby Friends

(/courses/grokking-the-system-design-interview/B8rpM8E16LQ)

Designing Uber backend

(/courses/grokking-the-system-design-interview/YQVkip548NM)

Design Ticketmaster (\*New\*)

(/courses/grokking-the-system-design-interview/YQyq6mBKq4n)

Additional Resources

(/courses/grokking-the-system-design-interview/JYILJB6DK7g)

Glossary of System Design Basics

System Design Basics

(/courses/grokking-the-system-design-interview/B892KY261z2)

Key Characteristics of Distributed Systems

(/courses/grokking-the-system-design-interview/YQWGjZZVz9)

Load Balancing

(/courses/grokking-the-system-design-interview/...

Timestamp, etc.) to store in the database. We would also need to

store information about movies and cinemas; let's assume it'll take 50 bytes. So, to store all the data about all shows of all cinemas of all cities for a day:

500 cities \* 10 cinemas \* 2000 seats \* 2 shows \* (50+50) bytes = 2GB

/ day

To store five years of this data, we would need around 3.6TB.

5. System APIs

#

We can have SOAP or REST APIs to expose the functionality of our service. The following could be the definition of the APIs to search movie shows and reserve seats.

SearchMovies(api\_dev\_key, keyword, city, lat\_long, radius, start\_datetime, end\_datetime, postal\_code, includeSpellcheck, results\_per\_page, sorting\_order)

Parameters:

api\_dev\_key (string):

The API developer key of a registered account. This will be used to, among other things, throttle users based on their allocated quota.

keyword (string):

Keyword to search on.

city (string):

City to filter movies by.

lat\_long (string):

Latitude and longitude to filter by.

radius (number):

Radius of the area in which we want to search for events.

start\_datetime (string):

Filter movies with a starting datetime.

end\_datetime (string):

Filter movies with an ending datetime.

postal\_code (string):

Filter movies by postal code / zipcode.

includeSpellcheck (Enum: "yes" or "no"):

Yes, to include spell check suggestions in the response.

results\_per\_page (number):

Number of results to return per page. Maximum is 30.

sorting\_order (string):

Sorting order of the search result. Some allowable values : 'name,asc', 'name,desc', 'date,asc', 'date,desc', 'distance,asc', 'name,date,asc', 'name,date,desc', 'date,name,asc', 'date,name,desc'.

Returns: (JSON)

Here is a sample list of movies and their shows:



Explore

(/explore)



Tracks

(/tracks)



My Courses

(/mycourses)




Edpresso

(/edpresso)



Refer a Friend

(/refer-a-friend)



Create Courses

(/courses/grokking-the-system-design-interview/YQYq6mB)

# Grokking the System Design Interview

(/collection/5668639101419520/56490502)

87% completed

Q

Search Course

- (/courses/grokking-the-system-design-interview/B8R22v0wqJo)
- Designing Twitter

(/courses/grokking-the-system-design-interview/m2G48X18NDO)
- Designing Youtube or Netflix

(/courses/grokking-the-system-design-interview/xV26VjZ7yMI)
- Designing Typeahead Suggestion

(/courses/grokking-the-system-design-interview/mE2XkgGRnmp)
- Designing an API Rate Limiter

(/courses/grokking-the-system-design-interview/3jYKmrVAPGQ)
- Designing Twitter Search

(/courses/grokking-the-system-design-interview/xV9mMjj74gE)
- Designing a Web Crawler

(/courses/grokking-the-system-design-interview/NE5LpPrWrKv)
- Designing Facebook's Newsfeed

(/courses/grokking-the-system-design-interview/gxpWJ3ZKYwl)
- Designing Yelp or Nearby Friends

(/courses/grokking-the-system-design-interview/B8rpM8E16LQ)
- Designing Uber backend

(/courses/grokking-the-system-design-interview/YQVkj548NM)
- Design Ticketmaster (\*New\*)

(/courses/grokking-the-system-design-interview/YQYq6mBKq4n)
- Additional Resources

(/courses/grokking-the-system-design-interview/JYILJB6DK7g)

## Glossary of System Design Basics

- System Design Basics

(/courses/grokking-the-system-design-interview/B892KY261z2)
- Key Characteristics of Distributed Systems

(/courses/grokking-the-system-design-interview/YQWGjZZVz9)
- Load Balancing

(/courses/grokking-the-system-design-interview/YQWGjZZVz9)

```
[
  {
    "MovieID": 1,
    "ShowID": 1,
    "Title": "Cars 2",
    "Description": "About cars",
    "Duration": 120,
    "Genre": "Animation",
    "Language": "English",
    "ReleaseDate": "8th Oct. 2014",
    "Country": "USA",
    "StartTime": "14:00",
    "EndTime": "16:00",
    "Seats": [
      {
        "Type": "Regular"
        "Price": 14.99
        "Status": "Almost Full"
      },
      {
        "Type": "Premium"
        "Price": 24.99
        "Status": "Available"
      }
    ]
  },
  {
    "MovieID": 1,
    "ShowID": 2,
    "Title": "Cars 2",
    "Description": "About cars",
    "Duration": 120,
    "Genre": "Animation",
    "Language": "English",
    "ReleaseDate": "8th Oct. 2014",
    "Country": "USA",
    "StartTime": "16:30",
    "EndTime": "18:30",
    "Seats": [
      {
        "Type": "Regular"
        "Price": 14.99
        "Status": "Full"
      },
      {
        "Type": "Premium"
        "Price": 24.99
        "Status": "Almost Full"
      }
    ]
  }
],
]
```

ReserveSeats(api\_dev\_key, session\_id, movie\_id, show\_id, seats\_to\_reserve[])

Parameters:



# Grokking the System Design Interview

87% completed

Q Search Course

Designing Twitter  
(/courses/grokking-the-system-design-interview/m2G48X18NDO)

Designing Typeahead Suggestion  
(/courses/grokking-the-system-  
design-interview/mE2XkgGRnmp)

Designing an API Rate Limiter  
(/courses/grokking-the-system-design-interview/3jYKmrVAPGQ)

Designing Twitter Search  
(/courses/grokking-the-system-  
design-interview/xV9mMjj74gE)

Designing a Web Crawler  
(/courses/grokking-the-system-  
design-interview/NE5LpPrWrKv)

Designing Facebook's Newsfeed  
(/courses/grokking-the-system-  
design-interview/gxpWJ3ZKYwl)

Designing Yelp or Nearby Friends  
(/courses/grokking-the-system-design-interview/B8rpM8E16LQ)

Designing Uber backend  
(/courses/grokking-the-system-design-interview/YQVkj548NM)

Design Ticketmaster (\*New\*)  
(/courses/grokking-the-system-design-interview/YQyq6mBKq4n)

Additional Resources  
(/courses/grokking-the-system-design-interview/JYLJB6DK7g)

# Glossary of System Design Basics

System Design Basics  
(/courses/grokking-the-system-design-interview/B892KY261z2)

## Key Characteristics of Distributed Systems

(/courses/grokking-the-system-design-interview/YQWGIlZZVz9)

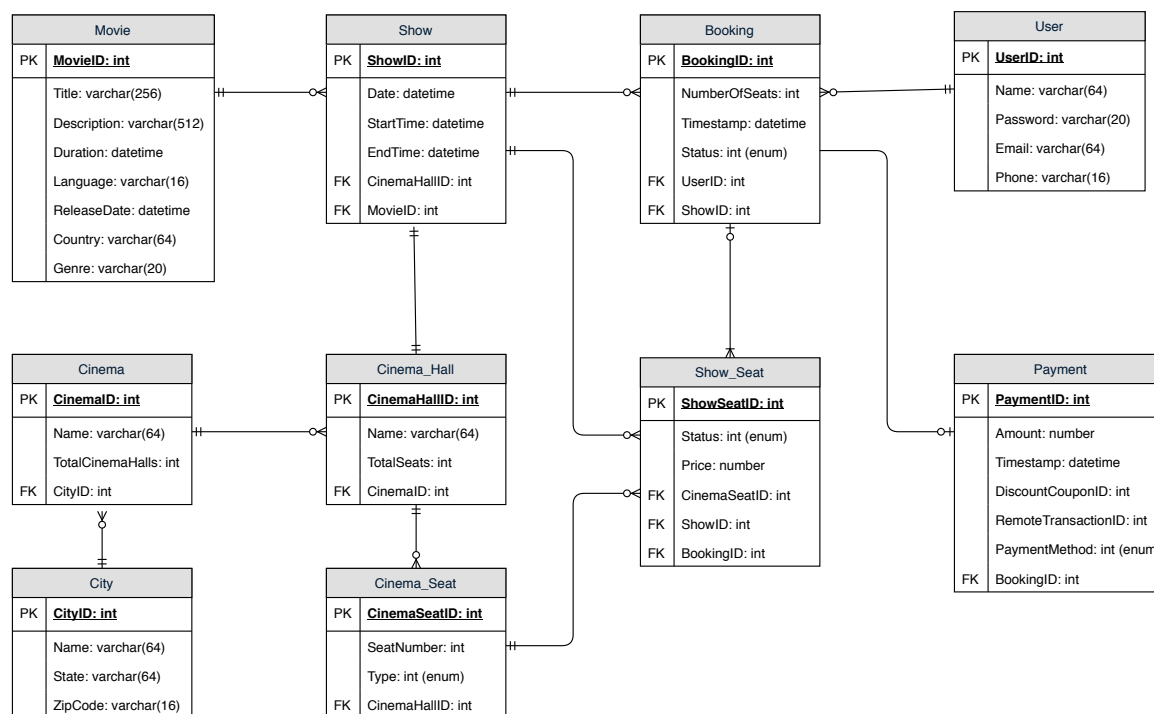
## Load Balancing

Returns the status of the reservation, which would be one of the following: 1) “Reservation Successful” 2) “Reservation Failed - Show Full,” 3) “Reservation Failed - Retry, as other users are holding reserved seats”.

## #

Here are a few observations about the data we are going to store:

1. Each City can have multiple Cinemas.
2. Each Cinema will have multiple halls.
3. Each Movie will have many Shows and each Show will have multiple Bookings.
4. A user can have multiple bookings.



## #


At a high-level, our web servers will manage users' sessions and application servers will handle all the ticket management, storing data in the databases as well as working with the cache servers to process reservations.




Explore  
(/explore)




Tracks  
(/tracks)



My Courses  
(/mycourses)



Edpresso  
(/edpresso)



Refer a  
Friend  
(/refer-a-  
friend)

# Grokking the System Design Interview

(/collection/5668639101419520/56490502)

87% completed



Q Search Course

(/courses/grokking-the-system-design-interview/B8R22v0wqJo)

Designing Twitter  
(/courses/grokking-the-system-design-interview/m2G48X18NDO)

Designing Youtube or Netflix  
(/courses/grokking-the-system-design-interview/xV26VjZ7yMI)

Designing Typeahead Suggestion  
(/courses/grokking-the-system-design-interview/mE2XkgGRnmp)

Designing an API Rate Limiter  
(/courses/grokking-the-system-design-interview/3jYKmrVAPGQ)

Designing Twitter Search  
(/courses/grokking-the-system-design-interview/xV9mMjj74gE)

Designing a Web Crawler  
(/courses/grokking-the-system-design-interview/NE5LpPrWrKv)

Designing Facebook's Newsfeed  
(/courses/grokking-the-system-design-interview/gxpWJ3ZKYwl)

Designing Yelp or Nearby Friends  
(/courses/grokking-the-system-design-interview/B8rpM8E16LQ)

Designing Uber backend  
(/courses/grokking-the-system-design-interview/YQVkj548NM)

Design Ticketmaster (\*New\*)  
(/courses/grokking-the-system-design-interview/YQyq6mBKq4n)

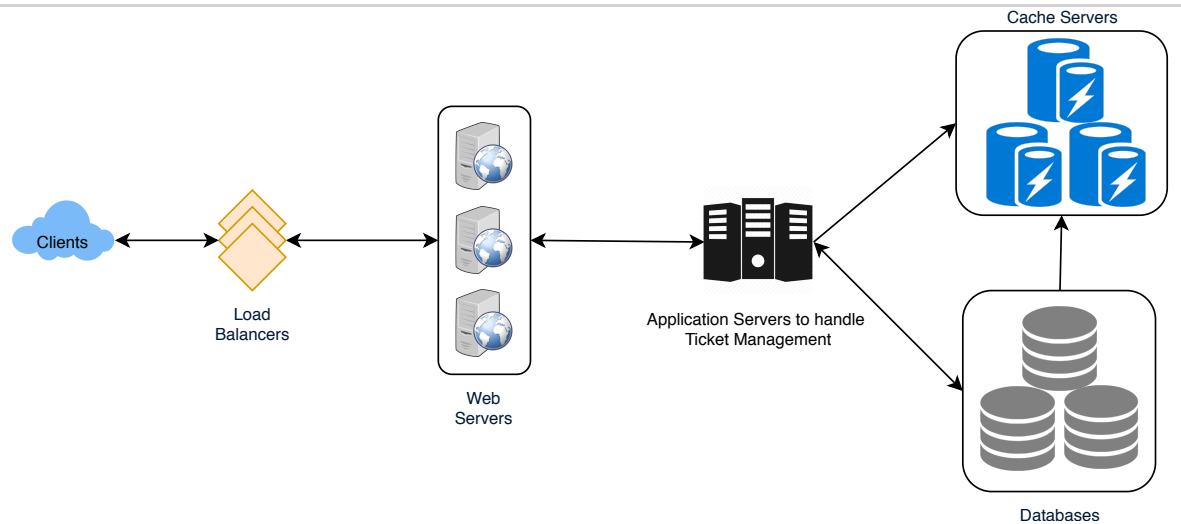
Additional Resources  
(/courses/grokking-the-system-design-interview/JYILJB6DK7g)

## Glossary of System Design Basics

System Design Basics  
(/courses/grokking-the-system-design-interview/B892KY261z2)

Key Characteristics of Distributed Systems  
(/courses/grokking-the-system-design-interview/YQWGjZZVz9)

Load Balancing  
(/courses/grokking-the-system-design-interview/...



## 8. Detailed Component Design

#

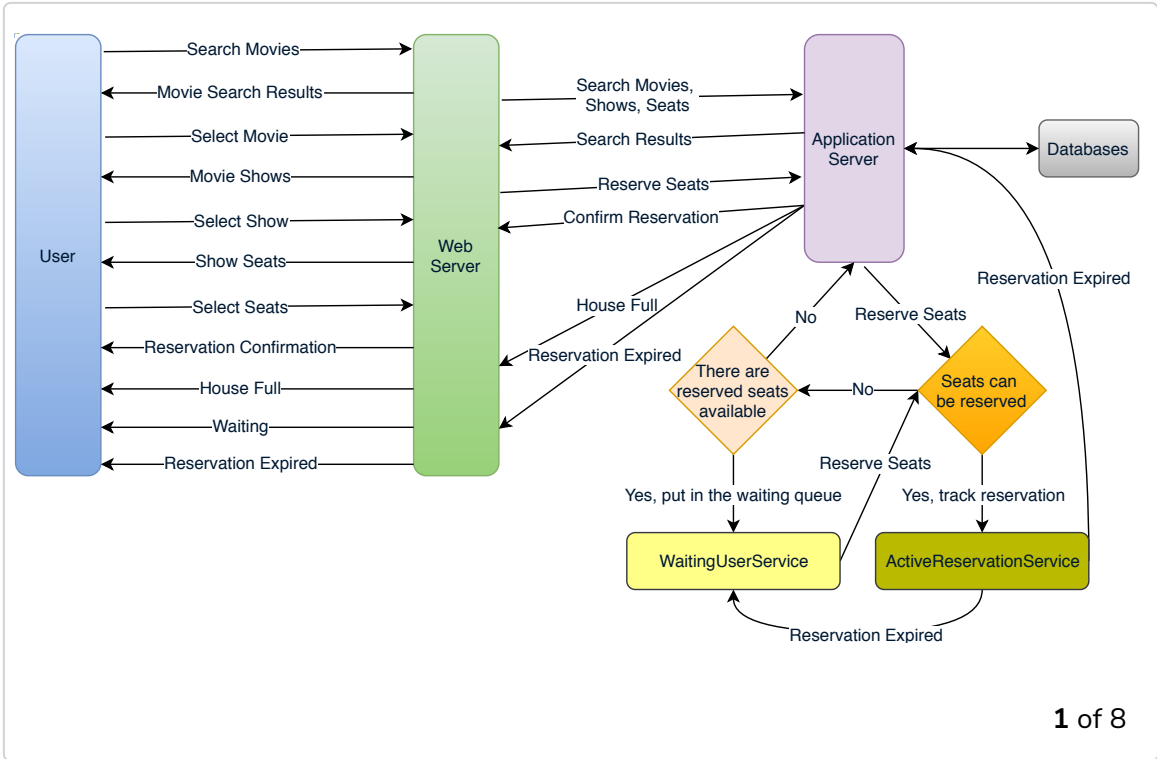
First, let's try to build our service assuming it is being served from a single server.

**Ticket Booking Workflow:** The following would be a typical ticket booking workflow:

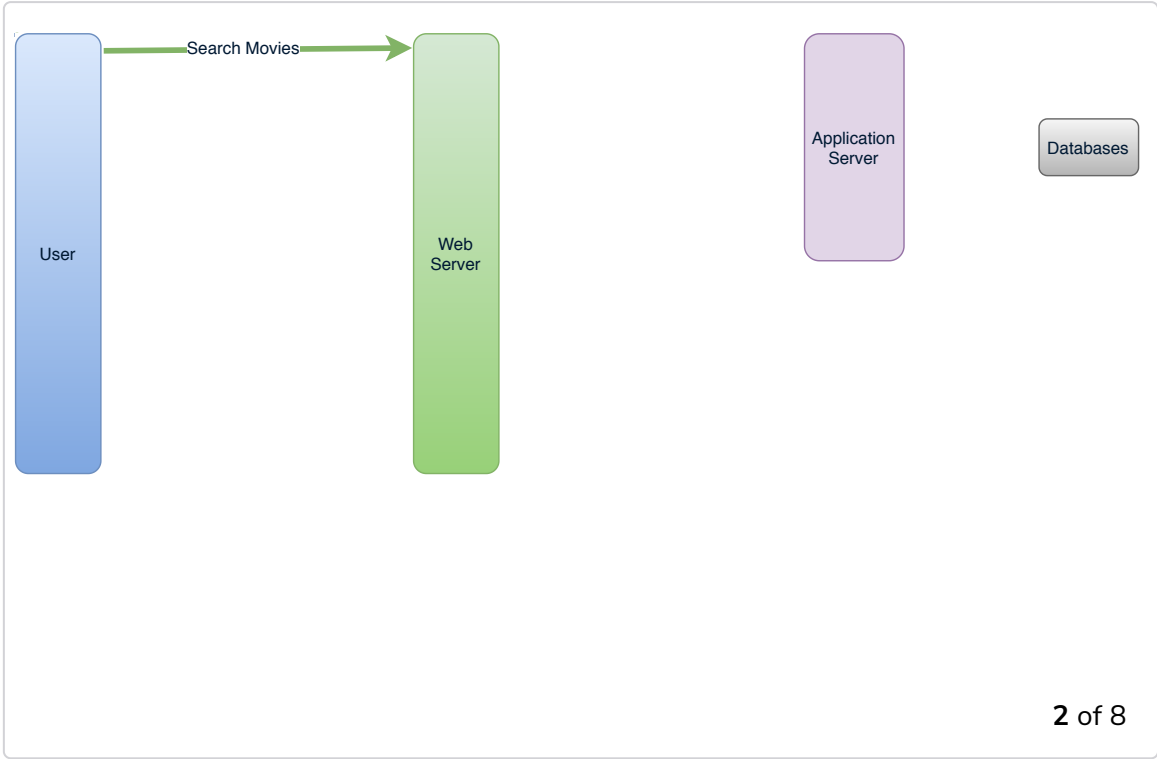
1. The user searches for a movie.
2. The user selects a movie.
3. The user is shown the available shows of the movie.
4. The user selects a show.
5. The user selects the number of seats to be reserved.
6. If the required number of seats are available, the user is shown a map of the theater to select seats. If not, the user is taken to 'step 8' below.
7. Once the user selects the seat, the system will try to reserve those selected seats.
8. If seats can't be reserved, we have the following options:
  - Show is full; the user is shown the error message.
  - The seats the user wants to reserve are no longer available, but there are other seats available, so the user is taken back to the theater map to choose different seats.
  - There are no seats available to reserve, but all the seats are not booked yet, as there are some seats that other users are holding in the reservation pool and have not booked yet. The user will be taken to a waiting page where they can wait until the required seats get freed from the reservation pool. This waiting could result in the following options:
    - If the required number of seats become available, the user is taken to the theater map page where they can choose seats.
    - While waiting, if all seats get booked or there are fewer seats in the reservation pool than the user intend to book, the user is shown the error message.
    - User cancels the waiting and is taken back to the movie search page.

- At maximum, a user can wait one hour, after that user’s session gets expired and the user is taken back to the movie search page.

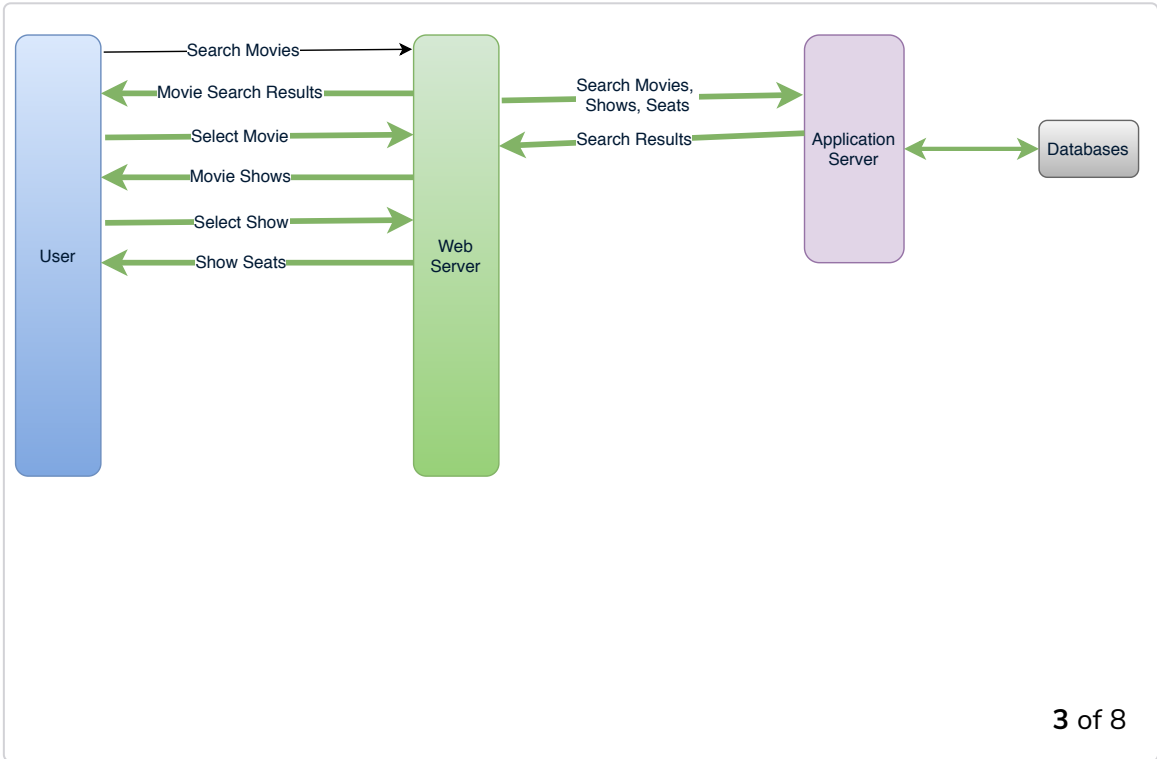
9. If seats are reserved successfully, the user has five minutes to pay for the reservation. After payment, booking is marked complete. If the user is not able to pay within five minutes, all their reserved seats are freed to become available to other users.



1 of 8



2 of 8



3 of 8

## Grokking the System Design Interview

(/collection/5668639101419520/56490502)

87% completed

Search Course

- (/courses/grokking-the-system-design-interview/B8R22v0wqJo)
- Designing Twitter (/courses/grokking-the-system-design-interview/m2G48X18NDO)
- Designing Youtube or Netflix (/courses/grokking-the-system-design-interview/xV26VjZ7yMI)
- Designing Typeahead Suggestion (/courses/grokking-the-system-design-interview/mE2XkgGRnmp)
- Designing an API Rate Limiter (/courses/grokking-the-system-design-interview/3jYKmrVAPGQ)
- Designing Twitter Search (/courses/grokking-the-system-design-interview/xV9mMjj74gE)
- Designing a Web Crawler (/courses/grokking-the-system-design-interview/NE5LpPrWrKv)
- Designing Facebook’s Newsfeed (/courses/grokking-the-system-design-interview/gxpWJ3ZKYwl)
- Designing Yelp or Nearby Friends (/courses/grokking-the-system-design-interview/B8rpM8E16LQ)
- Designing Uber backend (/courses/grokking-the-system-design-interview/YQVkip548NM)
- Design Ticketmaster (\*New\*) (/courses/grokking-the-system-design-interview/YQyq6mBKq4n)
- Additional Resources (/courses/grokking-the-system-design-interview/JYILJB6DK7g)

## Glossary of System Design Basics

- System Design Basics (/courses/grokking-the-system-design-interview/B892KY261z2)
- Key Characteristics of Distributed Systems (/courses/grokking-the-system-design-interview/YQWGjZZVz9)
- Load Balancing (/courses/grokking-the-system-design-interview/YQyq6mBKq4n)

Explore (/explore)

Tracks (/tracks)

My Courses (/mycourses)

Edpresso (/edpresso)

Refer a Friend (/refer-a-friend)

Create Courses (/courses/grokking-the-system-design-interview/YQyq6mBKq4n)




Explore  
(/explore)




Tracks  
(/tracks)



My Courses  
(/mycourses)



Edpresso  
(/edpresso)



Refer a Friend  
(/refer-a-friend)

# Grokking the System Design Interview

(/collection/5668639101419520/56490502)

87% completed

Q Search Course

(/courses/grokking-the-system-design-interview/B8R22v0wqJo)

Designing Twitter  
(/courses/grokking-the-system-design-interview/m2G48X18NDO)

Designing Youtube or Netflix  
(/courses/grokking-the-system-design-interview/xV26VjZ7yMI)

Designing Typeahead Suggestion  
(/courses/grokking-the-system-design-interview/mE2XkgGRnmp)

Designing an API Rate Limiter  
(/courses/grokking-the-system-design-interview/3jYKmrVAPGQ)

Designing Twitter Search  
(/courses/grokking-the-system-design-interview/xV9mMjj74gE)

Designing a Web Crawler  
(/courses/grokking-the-system-design-interview/NE5LpPrWrKv)

Designing Facebook's Newsfeed  
(/courses/grokking-the-system-design-interview/gxpWJ3ZKYwl)

Designing Yelp or Nearby Friends  
(/courses/grokking-the-system-design-interview/B8rpM8E16LQ)

Designing Uber backend  
(/courses/grokking-the-system-design-interview/YQVkj548NM)

Design Ticketmaster (\*New\*)  
(/courses/grokking-the-system-design-interview/YQyq6mBKq4n)

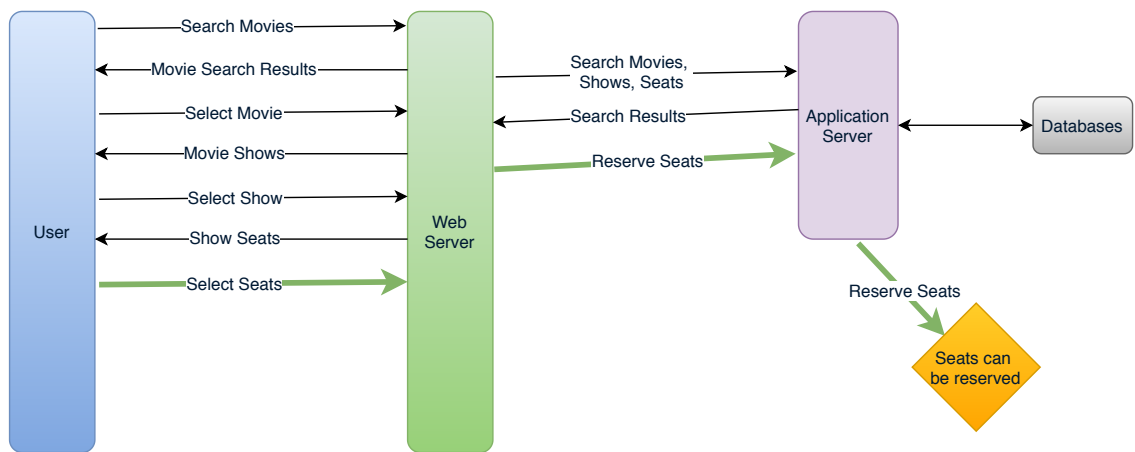
Additional Resources  
(/courses/grokking-the-system-design-interview/JYILJB6DK7g)

## Glossary of System Design Basics

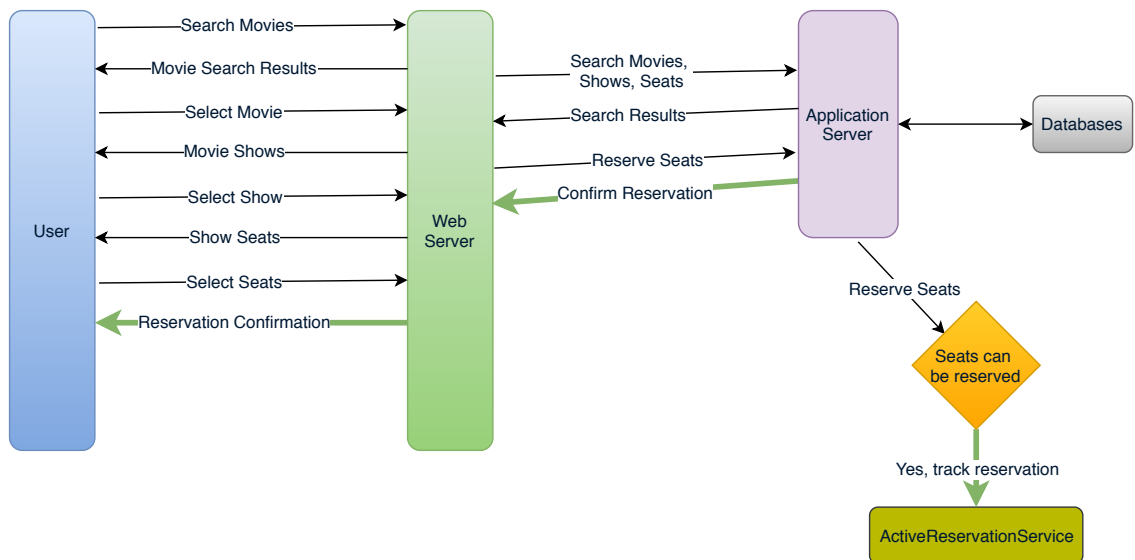
System Design Basics  
(/courses/grokking-the-system-design-interview/B892KY261z2)

Key Characteristics of Distributed Systems  
(/courses/grokking-the-system-design-interview/YQWGjZZVz9)

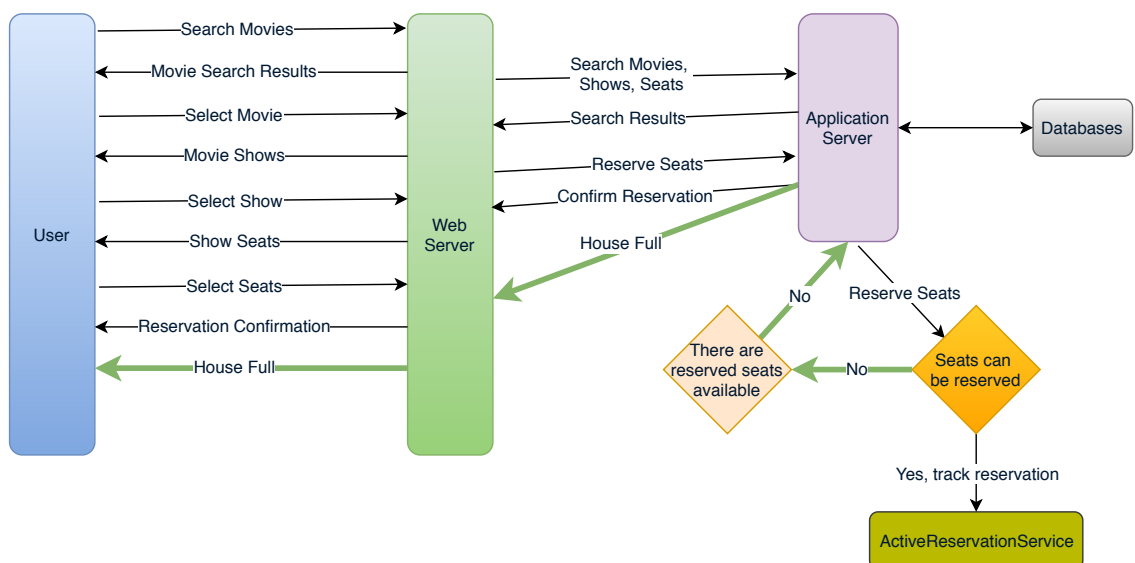
Load Balancing  
(/courses/grokking-the-system-design-interview/...



4 of 8



5 of 8



6 of 8





Create  
courses/grokking-  
system-design-  
iew/YQyq6mB




 Explore (/explore)

 Tracks (/tracks)

 My Courses (/mycourses)

 Edpresso (/edpresso)

 Refer a Friend (/refer-a-friend)

## Grokking the System Design Interview

(/collection/5668639101419520/56490502)

87% completed

 Search Course

(/courses/grokking-the-system-design-interview/B8R22v0wqJo)

Designing Twitter (/courses/grokking-the-system-design-interview/m2G48X18NDO)

Designing Youtube or Netflix (/courses/grokking-the-system-design-interview/xV26VjZ7yMI)

Designing Typeahead Suggestion (/courses/grokking-the-system-design-interview/mE2XkgGRnmp)

Designing an API Rate Limiter (/courses/grokking-the-system-design-interview/3jYKmrVAPGQ)

Designing Twitter Search (/courses/grokking-the-system-design-interview/xV9mMjj74gE)

Designing a Web Crawler (/courses/grokking-the-system-design-interview/NE5LpPrWrKv)

Designing Facebook's Newsfeed (/courses/grokking-the-system-design-interview/gxpWJ3ZKYwl)

Designing Yelp or Nearby Friends (/courses/grokking-the-system-design-interview/B8rpM8E16LQ)

Designing Uber backend (/courses/grokking-the-system-design-interview/YQVkj548NM)

Design Ticketmaster (\*New\*) (/courses/grokking-the-system-design-interview/YQyq6mBKq4n)

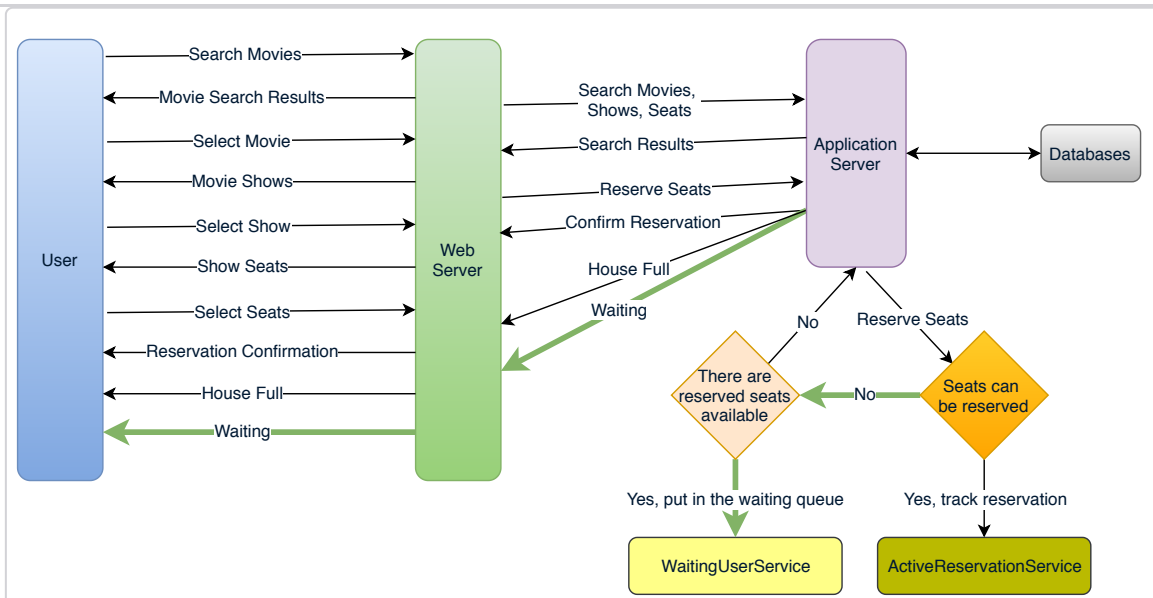
Additional Resources (/courses/grokking-the-system-design-interview/JYILJB6DK7g)

## Glossary of System Design Basics

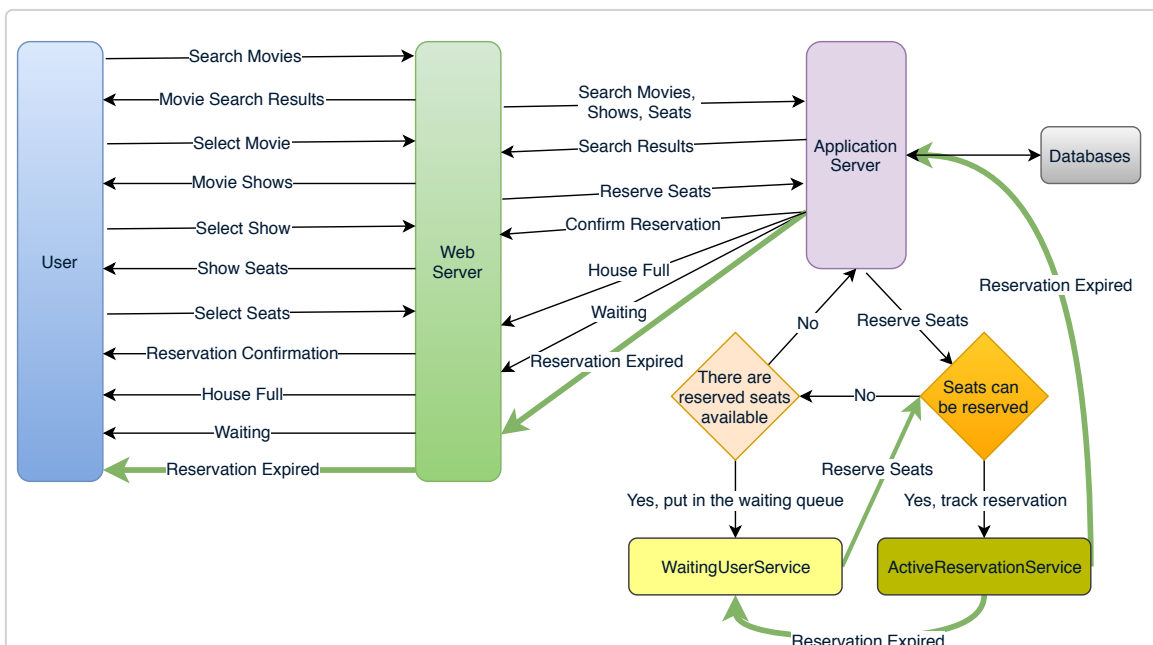
System Design Basics (/courses/grokking-the-system-design-interview/B892KY261z2)

Key Characteristics of Distributed Systems (/courses/grokking-the-system-design-interview/YQWGjZZVz9)

Load Balancing (/courses/grokking-the-system-design-interview/...



7 of 8



8 of 8

**How would the server keep track of all the active reservation that haven't been booked yet? And how would the server keep track of all the waiting customers?**

We need two daemon services, one to keep track of all active reservations and remove any expired reservation from the system; let's call it **ActiveReservationService**. The other service would be keeping track of all the waiting user requests and, as soon as the required number of seats become available, it will notify the (the longest waiting) user to choose the seats; let's call it **WaitingUserService**.

### a. ActiveReservationsService

We can keep all the reservations of a 'show' in memory in a data structure similar to Linked HashMap (<https://docs.oracle.com/javase/7/docs/api/java/util/LinkedHashMap.html>) or a TreeMap (<https://docs.oracle.com/javase/6/docs/api/java/util/TreeMap.html>) in addition to keeping all the data in the database. We will need a linked HashMap kind of data structure that allows us to jump to any reservation to remove it when the booking is complete. Also,







PK

Explore (/explore)

Tracks (/tracks)

My Courses (/mycourses)

Edpresso (/edpresso)

Refer a Friend (/refer-a-friend)

Create Courses (/courses/grokking-the-system-design-interview/YQyq6mB)

educative (/learn)

87% completed

Search Course

Grokking the System Design Interview

(/collection/5668639101419520/56490502)

87% completed

Designing Twitter (/courses/grokking-the-system-design-interview/m2G48X18NDO)

Designing Youtube or Netflix (/courses/grokking-the-system-design-interview/xV26VjZ7yMI)

Designing Typeahead Suggestion (/courses/grokking-the-system-design-interview/mE2XkgGRnmp)

Designing an API Rate Limiter (/courses/grokking-the-system-design-interview/3jYKmrVAPGQ)

Designing Twitter Search (/courses/grokking-the-system-design-interview/xV9mMjj74gE)

Designing a Web Crawler (/courses/grokking-the-system-design-interview/NE5LpPrWrKv)

Designing Facebook's Newsfeed (/courses/grokking-the-system-design-interview/gxpWJ3ZKYwl)

Designing Yelp or Nearby Friends (/courses/grokking-the-system-design-interview/B8rpM8E16LQ)

Designing Uber backend (/courses/grokking-the-system-design-interview/YQVkj548NM)

Design Ticketmaster (\*New\*) (/courses/grokking-the-system-design-interview/YQyq6mBKq4n)

Additional Resources (/courses/grokking-the-system-design-interview/JYILJB6DK7g)

Glossary of System Design Basics

System Design Basics (/courses/grokking-the-system-design-interview/B892KY261z2)

Key Characteristics of Distributed Systems (/courses/grokking-the-system-design-interview/YQWGjZZVz9)

Load Balancing (/courses/grokking-the-system-design-interview/...

active reservations from the booking table. Remember that we keep the 'Status' column as 'Reserved (1)' until a reservation is booked. Another option is to have a master-slave configuration so that, when the master crashes, the slave can take over. We are not storing the waiting users in the database, so, when WaitingUserService crashes, we don't have any means to recover that data unless we have a master-slave setup.

Similarly, we'll have a master-slave setup for databases to make them fault tolerant.

11. Data Partitioning

#

Database partitioning:

If we partition by 'MovieID', then all the Shows of a movie will be on a single server. For a very hot movie, this could cause a lot of load on that server. A better approach would be to partition based on ShowID; this way, the load gets distributed among different servers.

ActiveReservationService and WaitingUserService partitioning:

Our web servers will manage all the active users' sessions and handle all the communication with the users. We can use the Consistent Hashing to allocate application servers for both ActiveReservationService and WaitingUserService based upon the 'ShowID'. This way, all reservations and waiting users of a particular show will be handled by a certain set of servers. Let's assume for load balancing our Consistent Hashing (https://www.educative.io/collection/page/5668639101419520/5649050225344512/5709068098338816) allocates three servers for any Show, so whenever a reservation is expired, the server holding that reservation will do the following things:

1. Update database to remove the Booking (or mark it expired) and update the seats' Status in 'Show\_Seats' table.

2. Remove the reservation from the Linked HashMap.

3. Notify the user that their reservation has expired.

4. Broadcast a message to all WaitingUserService servers that are holding waiting users of that Show to figure out the longest waiting user. Consistent Hashing scheme will tell what servers are holding these users.

5. Send a message to the WaitingUserService server holding the longest waiting user to process their request if required seats have become available.

Whenever a reservation is successful, following things will happen:

1. The server holding that reservation sends a message to all servers holding the waiting users of that Show, so that those servers can expire all the waiting users that need more seats than the available seats.

2. Upon receiving the above message, all servers holding the



PK

educative

(/learn)

Explore

(/explore)

Tracks

(/tracks)

My Courses

(/mycourses)

Edpresso

(/edpresso)

Refer a Friend

(/refer-a-friend)

waiting users will query the database to find how many free seats are available now. A database cache would greatly help here to run this query only once.

3. Expire all waiting users who want to reserve more seats than the available seats. For this, WaitingUserService has to iterate through the Linked HashMap of all the waiting users.

Back

(/courses/grokking-the-system-design-interview/B8R22v0wqJo)

the-system-design-interview/YQVkj548NM)

design-Additional Resources interview/YILJB6DK7g)

Report an Issue

Ask a Question

(https://discuss.educative.io/c/grokking-the-system-design-interview-design-gurus/system-design-problems-design-bookmyshow-new)

Glossary of System Design Basics

System Design Basics

(/courses/grokking-the-system-design-interview/B892KY261z2)

Key Characteristics of Distributed Systems

(/courses/grokking-the-system-design-interview/YQWGjZZVz9)

Load Balancing

(/courses/grokking-the-system-design-interview/YQyq6mBKq4n)