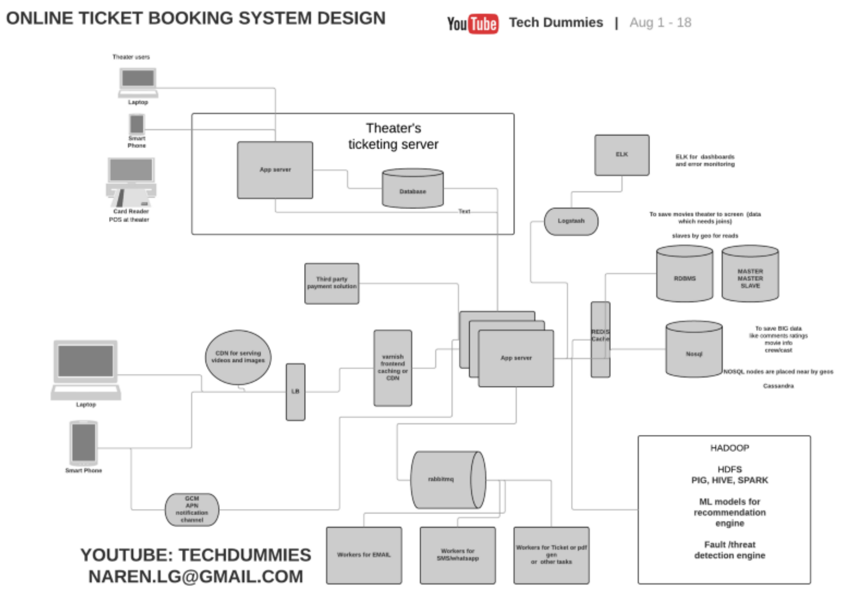
"woman standing near building" by Joshua Rawson-Harris on Unsplash

# BOOKMYSHOW system design !!

Narendra L
Sep 8, 2018 · 6 min read


ONLINE TICKET BOOKING SYSTEM DESIGN
YouTube Tech Dummies | Aug 1 - 18
Theater users
Laptop
Smart Phone
Card Reader POS at theater
Theater's ticketing server
App server
Database
Text
ELK
ELK for dashboards and error monitoring
Logstash
To save movies theater to screen (data which needs joins)
slaves by geo for reads
RDBMS
MASTER MASTER SLAVE
Third party payment solution
REDIS Cache
Nosql
To save BIG data like comments ratings movie info crew/cast
NOSQL nodes are placed near by geos
Cassandra
Laptop
CDN for serving videos and images
LB
varnish frontend caching or CDN
App server
Smart Phone
rabbitmq
GCM APN notification channel
HADOOP
HDFS
PIG, HIVE, SPARK
ML models for recommendation engine
Fault /threat detection engine
YOUTUBE: TECHDUMMIES
NAREN.LG@GMAIL.COM
Workers for EMAIL
Workers for SMS/whatsapp
Workers for Ticket or pdf gen or other tasks

If you want to watch instead of reading, I have made a video as well

Mainly there are two components to the system

## USER → (1)BMS—→ (2)Theater

Now the first question is how BMS talks to Theater?

Every movie theater which works along with BMS will have their own POS, Mobile App or Website from which users can get tickets. to support these systems every theater needs there own

1. server
2. DB

3. App and website

Without a server at theater, this its a lot difficult to build this system. Any third party application/movie tickets aggregator app should work along with Theater's Server to get seat availability information.

There are many strategies to get seats allocated to aggregators

1. Every aggregator will have designated rows of seats
2. works along with theater and other aggregators for available seats

A) In the first strategy, we don't need to keep updating the available seat info from all the theaters, as we have dedicated seats available for us

and we, in turn, offer these tickets to users

B) in the second strategy we need to keep updating the seats availability info and we, in turn, offer these tickets to users

## What are the possible way to sync the seat booking data?

1. Connecting to theater's DB
2. or keep syncing the seats of every theater always (not a good option) and book
3. request for available seats using Theater's API and book using APIs.

Now there can be two or more users trying to book the same tickets!!

How do we solve that? → In Theater side server we need to have a lock with a timeout and assign seat temporarily to the user until he books.

If he doesn't book within 10 min, release the seat.

If your design involves tons of requests or IO blocking calls, then it is advised to use async in python or go coroutines in go or Erlangs lightweight threads to get better performance.

here are the design goals

1. Highly concurrent -> Scale app server and scale workers

2. Responsive UI -> using react JS and Bootstrap
3. Supports multiple city > RDBMS
4. Support payment > Third party payment gateway to support all type of payments
5. support movies suggestions when login > We are using Hadoop and spark streaming with ML to get Recommendation engine
6. Comments and rating > Save in Cassandra
7. Movie information, trailers, and galleries >
8. Send ticket copy -> workers
9. Sms and notification > Workers and GCM

**Load Balancer: We can use Hardware or Software**

Here are the techniques that load balancers use:

1. Consistent Hashing
2. Round Robin
3. Weighted Round Robin
4. Least Connection

Varnish for frontend caching: API and Pages—BMS uses Cloudflare

A well-designed cache can not only improve your UX but also help relieve the pressure on your backend infrastructure.

Also, you can use CDN to cache your pages/API/Contents

App server: BMS uses Java, Spring Boot, Swagger, Hibernate on EC2

Either Java or Python based servers Which are horizontal scaled, which mostly servers APIS.

Whereas Frontend is Single page app / Android iOS Apps

To support search API, the Elastic search is a distributed, RESTful search and analytics engine which works as App level search engine. to answer all search needs.

**DataBase:**

In case of ticketing system.

1. We need ACID and relationship representation
2. we need transactions
3. We have tons of data like movies information, actors, crew, comments and reviews
4. We need to have faster search
5. We need to run queries for analytics

We can't just use RDMS or NOSQL alone, let's use BOTH

For 1st and 2nd scenario use RDMS

for 3rd scenario use—NOSQL

for 4th scenario use—Elastic search

for 5th scenario use—HDFS

The system is read heavy we will have sharded and master-master slave RDBMS where slaves are used for reads and master for writes. Master Master to avoid single point failure single point failure

Whereas NOSQL can be distributed and nodes are deployed in multiple regions to avoid latency.

NoSQL is good at holding BIGDATA for example comments, review of all the movies released so far.

**Log management:**

ELK stack can be used to visualize logs using Logstash. Logstash can consume data from all the servers via Files/syslog/socket/AMQP etc and can redirect logs based on different set of filters to Queue/File/Hipchat/Whatsapp/JIRA etc

**Caching:**

Redis is an open source (BSD licensed), in-memory data structure store, used as a database, cache and message broker also. It is used as a caching layer for heavily read data

REDIS can also be used as locks for various purpose E.g.: To Block tickets, use REDIS entry for every seat a user is trying to book with a specific TTL

**Async workers:**

Needs a Message Queueing System like Rabbitmq or Kafka and Workers to execute the async task like Python celery workers execute time-consuming tasks like Sending SMS/Email or Watsap messages or Generating Tickets as PDF or crawling if any needed etc..

Also to browser notification or phone Notification use GCM/ APN

**Business intelligence and ML:**

To do data analysis of business information we can use Hadoop in which HDFS, PIG/Hive, and Spark streaming for real-time analytics/trend

ML can be used to understand users behavior and to generate recommendation etc

**WORKING STEPS:**

- Customer Will check Application with Location, Filters, Sorting, Categories, etc. Find location from APP or Location from ISP from Browser(For this data provided from DB and ELK sometimes recommendation engine)
- Open a Movie—Check all Nearby theaters and there timing of Movie.
- Select a theater for a particular date and time. Now the seat information is fetched from the DB of the Movie theater which is cached for a few seconds also
- Open Grid/Arina and can view available, unavailable, or booked tickets. Now user selects the seats, meanwhile, BMS should lock the seats for next 10 minutes. It makes a call to the theater DB to block the Seats. Now the ticket is booked temporarily to the user trying to book so every other app/ theater / other aggregators will see that seat as booked, and no one is allowed to book until next 15 minutes except the current user.
- Check Invoice with payment options and pay via gateway. Request will be redirected to payment gateway

- Once the Payment succeeds theater gives a unique ID
- Customer will get a ticket with Invoice with Encrypt QR code, Basic details of Movie, Address of theater, Timing, Theater Number, etc
- A ticket will be generated then and there only using Unique Ticket ID
- few more async tasks were invoked to send SMS, EMAIL and WHATSAPP the Tickets
- Customer goes to the theater at Movie Time and theater manager scan QR code on booking ticket.
- Ticket code is retrieved using QR code scan

**APIS needed:**

API -> GetListOfCities()

API -> GetListOfEventsByCity(CityId)

API -> GetLocationsByCity(CityId)

API -> GetLocationsByEventandCity(cityid, eventid)

API -> GetEventsByLocationandCity(CityId, LocationId)

API -> GetShowTiming(eventid,locationid)

API -> GetAvailableSeats(eventid,locationid,showtimeid)

API -> VarifyUserSelectedSeatsAvailable(eventid,locationid,showtimeid,seats)

API -> BlockUserSelectedSeats()

API -> BookUserSelectedSeat()

API -> GetTimeoutForUserSelectedSeats()

## Actual technologies used by BMS

UI: ReactJS & BootStrapJS

Server: Java, Spring Boot, Swagger, Hibernate

Security: Spring Security

Database: MySQL

Server: Tomcat

Caching: In memory cache Hazelcast.

Notifications: RabbitMQ. A Distributed message queue for push notifications.

Payment API: Popular ones are Paypal, Stripe, Square

Deployment: Docker & Ansible

Code repository: Git

Logging: Log4J

Log Management: Logstash, ELK Stack

Load balancer: Nginx