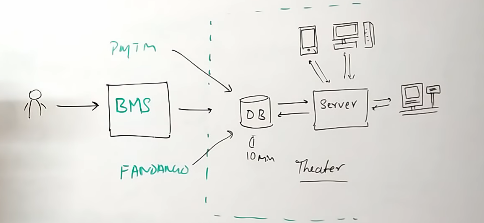
# Movie Ticket Booking System Design

Let’s start with the obvious fact how we will show the available and reserved seats for a particular theater.

There are two ways

1. We will directly connect to their DB (Very bad way)
2. Every theatre will be exposing an API and will connect to that API to get our information.

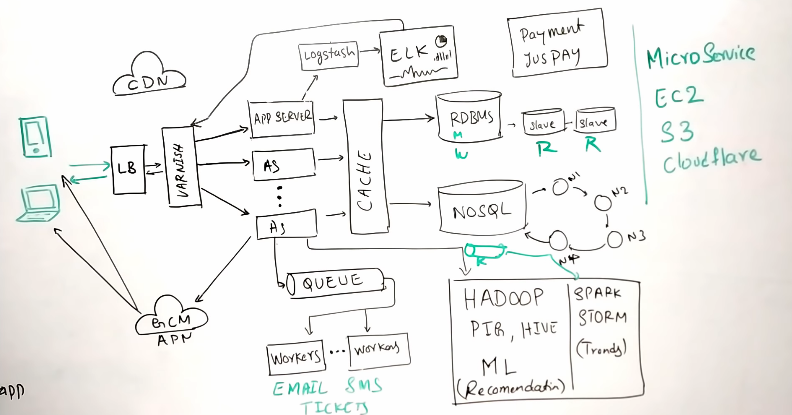


The most important part will be locking the seat, which will be done from the theater API perspective so anyone booking from same or any other site the same seat shouldn’t be booked for many users.

## Requirements to support

* Highly concurrent
* Responsive UI
* Multiple cities
* Payments
* Movie suggestions
* Comments & Ratings
* Movie info
* Send ticket by SMS Mail or WhatsApp.

## Architecture



**Load balancer** **(NgNix)** has its own strategy to distribute the load

* Concurrent hash ring
* Round robin
* Weighted round robin
* Least connection based strategy

**Varnish** is the frontend caching also makes request collapsing which will reduce the load on backend infrastructure.

**CDN** can be used to cache pages, images, video content etc. (Static data)

We need to scale the APP Server horizontally to handle the heavy load or highly concurrent request(s).

To support search query API, dump all the data to **Elastic Search** It is fast distributed and restful search API available in its system. It also can be used as analytics engine.

Next thing is cache to store Movie, Seats etc. can be used MemCachedb or Redis.

For Theatre, Seat bookings we can use **RDBMS** which also gives ACID transaction guarantee. In RDBMS we should store the data which is not growing too fast and also if we can establish a relationship around it. We also need to shard our database based on region (may be) where **Slaves** nodes will be used for reading and **Master** node will be for writing.

For data relates to movies, actors, crews, ratings, comments we can use NoSql database like MongoDB, Casandra which is designed for scale. We can set the replication factor and consistency level.

### Async workers

Once a user booked a movie we also need to do many other works like generate PDF, sent message, Email or send ticket via whatsapp. All these tasks are like making third party API calls these are network I/O which adds lots of latency. So in the same API request for ticket booking it’s better not to do all these tasks else we will lose performance of our application.

As soon as application server completes ticket booking, It can add an event to ticket booking **Queue** and from there **Workers** will take the task and perform asynchronously.

### Business intelligence and Recommendation engine

For this we need Hadoop platform. All the logs, user activities, information be dumped to Hadoop HDFS and on top of that we can run Pig or Hive queries to extract the information to train our machine learning model on user behavior.

We also can place **Kafka** queue in between and feed the data to **Spark** and **Storm** to do real time analysis for latest trends or user activity behavior or fraud detection.

### Payment gateway

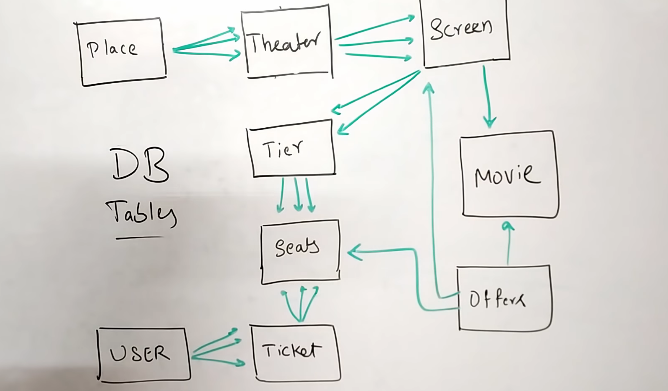
We can use 3rd party payment gateway.

### Location detection

For Mobile we can use GPS for location identification and for Desktop or Laptop we can use ISP based IP based strategy to get the location for user.

## Database design

### RDBMS tables



### Nosql tables

