Requirements

Select the city.	Display Theater	Select Seats
Display movies	Select ShowTime	Payment

Consistent so ACID may be used

Traffic Capacity Estimation

DAU - 1 Mn, Peak Traffic -10x of total traffic

SEARCHING -> 5 req per user per day = 5* 1Mn= 5Mn req = Peak req = 50 Mn req per day

SEAT SELECITON -> 1 req per user per day = 1* 1Mn = 1Mn req = Peak req = 10Mn req/day

Booking Tickets -> 0.2 req per user per day = 200,000 => Peak req = 2 Mn req /day

Payments ->

Storage Capacity Estimation

User - id, name, email, phone etc.. => 1Kb * 1Mn Dau = 1GB => 10times growth over time so 10 * 1GB=10GB

EventDetails -> eventId, date, name, artist, trailer etc..= 10Kb* 10000 movies= 100 MB

Booking Detail -> id, date, eventId, noOfSeat, seatNo,timing etc.. = 200000 * 0.5Mb=100MB/day => In 1 year = 365*100Mb = 36.5GB

API's

Login -> Post /api/login

UserInfo -> Get /api/details/{id}

FetchMovie -> Get /api/movies/{city}

FetchTheater -> Get /api/theater/{city}/{movie}

Booking -> Post /api/booking -> in ReqBody we send the details

Services

SEARCH SERVICE-> MAP<SHOWID, <BOOKINGID, TIMESTAMP>

PAYMENT SERVICE

BOOKING SERVICE

Database Design

User -> userId, name, password, email

City -> name, State

Cinema -> location, noOfScreen, city(FK)

Theatre-> id, name, seats, cinemaId (FK)

Movie -> id, name, actor,

Show -> id, movieId(FK), cinemaId(Fk), timings

Booking -> id, noOfSeats, showId(FK), userId(FK)

BookingId will be best for Partitioning

CONCURRENCY

Out of 4 transactions - ReadUncomitted, ReadComitted, Non RepeatableReads, Serializable, we'll use **Serializable.**

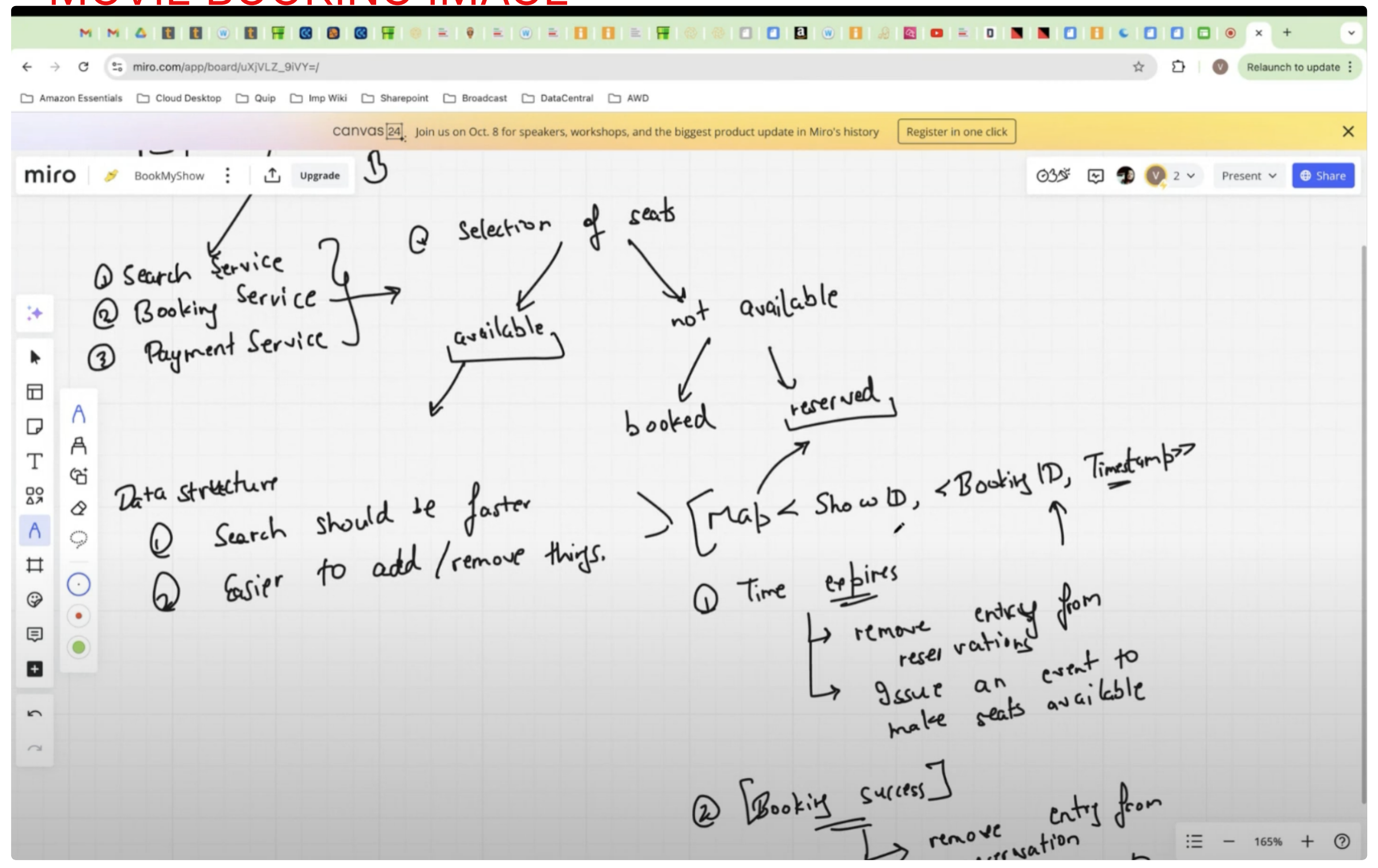
This is how we''l write the DB :-

Set Transaction Isolation Level Serializable;

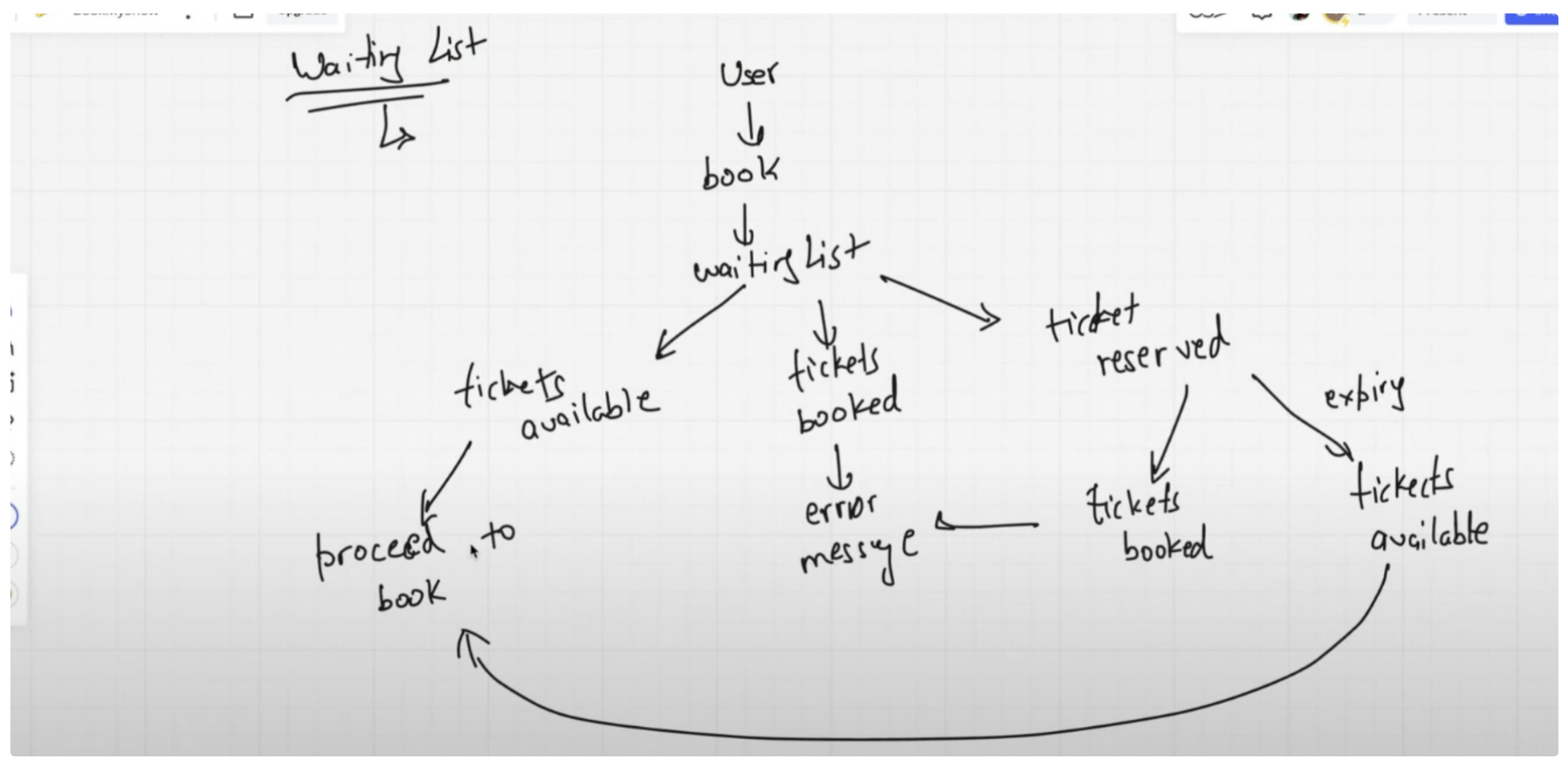
BEGIN
- **set of commands to run**
- **like select seats which is selected and reserve it.**
-

COMMIT TRANSACTION
END

MOVIE BOOKING IMAGE



miro.com/app/board/uXjVLZ_9iVY=/
Amazon Essentials
Cloud Desktop
Quip
Imp Wiki
Sharepoint
Broadcast
DataCentral
AWD
Relaunch to update
canvas 24 Join us on Oct. 8 for speakers, workshops, and the biggest product update in Miro's history
Register in one click
miro
BookMyShow
Upgrade
Present
Share
165%
① Search Service
② Booking Service
③ Payment Service
② Selection of seats
available
not available
booked
reserved
Data structure
① Search should be faster
② Easier to add / remove things.
[Map < ShowID, <Booking ID, Timestamp>>
① Time expires
remove entry from reservations
Issue an event to make seats available
② [Booking success]
remove entry from reservation



Waiting List
User
book
waiting List
tickets available
tickets booked
ticket reserved
proceed to book
error message
tickets booked
expiry
tickets available