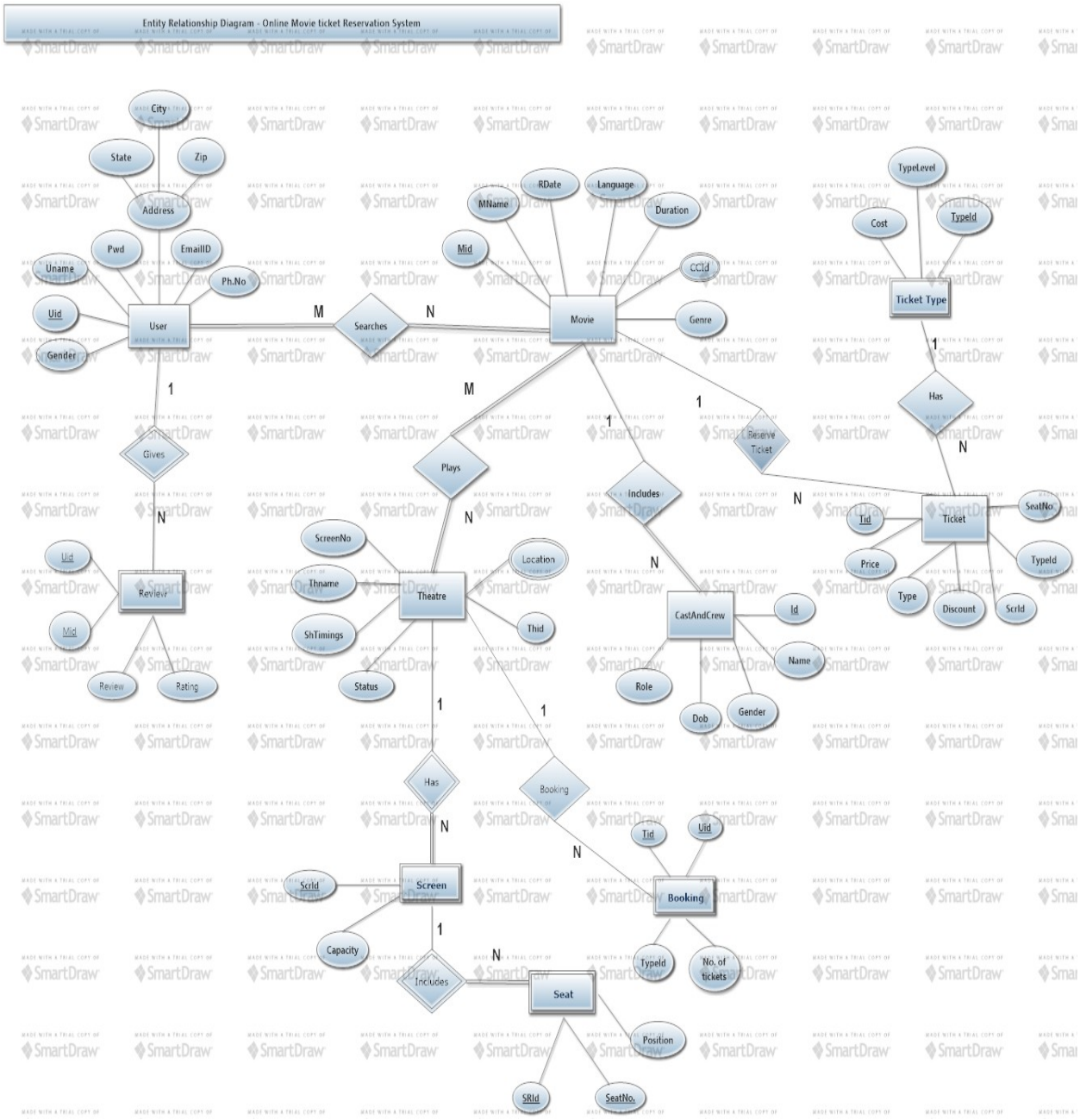
ONLINE MOVIE TICKET BOOKING

By

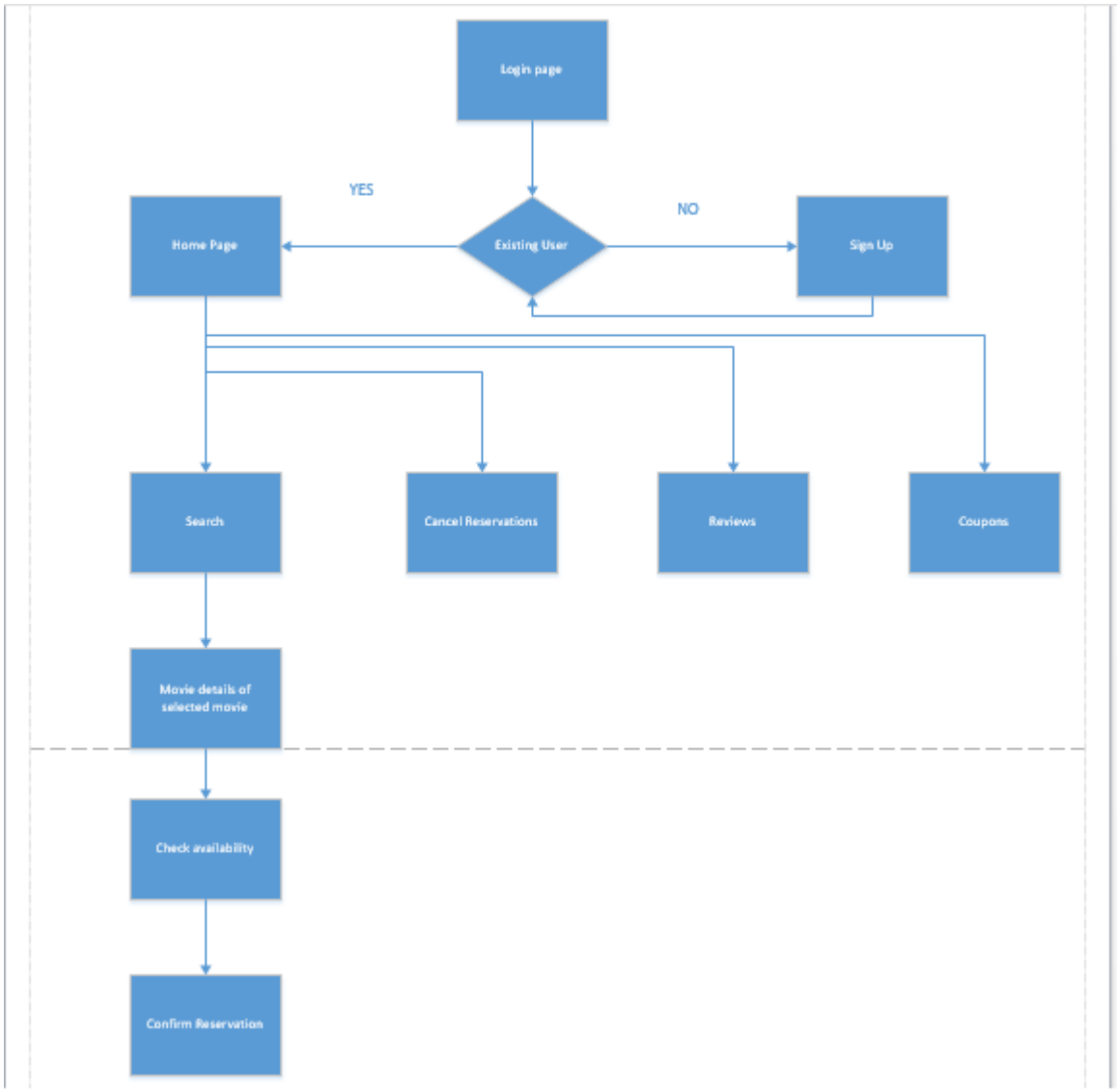
Ravi Krishna Duttaluru

E-R Diagram:



Tool Used: Smart Draw

<u>Front End Design Idea</u>:



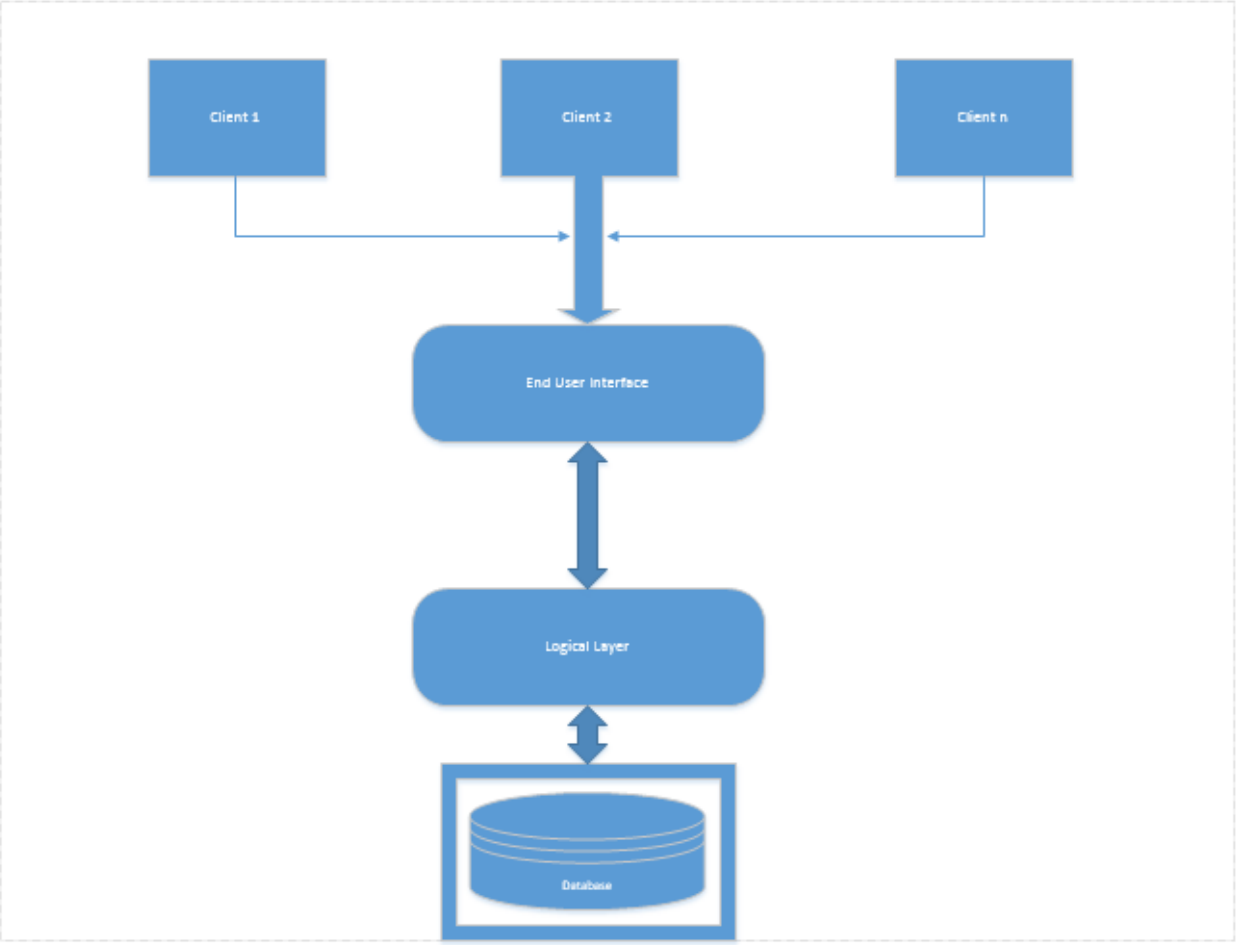
Tool Used : Microsoft Visio.

Online Movie Ticket booking, serves as an all in one system which deals with most of the aspects of current day online ticket booking. For any application the look and feel is very important and here goes the front end design idea for our application.

- As soon as the client enters the application the patron needs to login. If the patron is new then he first signs up and then gets logged in.
- In the home page, the client has various options to go through like booking ticket, cancelling ticket reservation , seeing reviews , coupons etc.,
- While booking the ticket, the front end will take the location and display the theaters accordingly. After that the client moves with the desired show and then the availability of seats will be shown.

- The wanted seats are hence reserved by the client.
- HTML5, JSP and CSS are used to design the webpages and javascript for validation purpose.
- We will have many dynamic pages in our application where the patrons will get up to date information.
- In the middleware we use Servlets.
- The JDBC-ODBC bridge drivers are used to connect to the backend database which is supposed to be Sql database.

Database Architecture:



Tool Used: Microsoft Visio

USER

UID	UNAME	EMAIL	GENDER	ADDRESS	DOB	PASSWORD	PASSWORD

PRIMARY KEY: UID

NOTE: We take only one Email, phone and address per person.

MOVIE

MID	MOVIENAME	CCID	LANGUAGE	MOVIE_RATING	RELEASE_DATE	DURATION	gen

PRIMARY KEY: MID

THEATER

THID	THNAME	MID	ScreeRoomID	SHOWTIMINGS	LOCATION	STATUS

PRIMARY KEY: THID, FOREIGN KEY: MID

SCREENROOM

SRID	CAPACITY

PRIMARY KEY: SRID

SEAT

SEATID	SRID	POSTION

PRIMARY KEY: {SEATID, SRID}

FOREIGN KEYS : SRID from SCREENROOM

TICKET

TID	MID	TYPE	DISCOUNTS	SRID	SEATID	TYPEID

PRIMARY KEY: TID

FOREIGN KEYS : MID is referenced from MOVIE table ; SRID is referenced from SCREENROOM table;SEATID is refrenced from SEAT;TYPEID is refrenced from Ticket_type.

TICKET_TYPE

TYPEID	TYPE_description	COST

PRIMARY KEY: {TYPEID}

BOOKING

BID	USERID	TID	TYPEID	No_Of_Tickets

PRIMARY KEY: BID

REVIEW

USERID	MID	REVIEW	RATING

CASTS_CREW

ID	NAME	DOB	GENDER	ROLE

PRIMARY KEY: ID

DATA DEPENDICIES

userID ->{username,address,emailID,gender,phone,password}

{mid,moviename}->{genre,ccid,language,movie_rating,release_date,duration)

Thid ->{mid,thname,srid,showtimings,location,status}

{seated,srid}->position

{bid,userid,tid) ->{typeid,no_of_tickets}

Typeid->{type_description,cost}

{userid,mid} ->{review,rating}

ID ->{name,dob,gender,role}

RELATIONAL ALGEBRAIC STATEMENTS:

- Search for movie using language as input

$\Pi_{MID,MOVIE\_NAME}(\sigma_{language='input'}(MOVIE))$

- Search for movies using location as input

  Result1 ->$\Pi_{mid}(\sigma_{location='input'}(Theater))$
  Result2->$\Pi_{mid,moviename}(Movie)$
  Result3->Result1 ∞ Result2
  Movie_location->$\Pi_{moviename}(Result3)$

- Search for movies using genre as input

  $\Pi_{moviename}(\sigma_{genre='input'}(Movie))$

- Retrieve movies using rating as input

  $\Pi_{moviename}(\sigma_{rating='input'}(Movie))$

- Search for movies along with duration

  $\Pi_{moviename,duration}(Movie)$

- Search for availability of movie in particular location

  Result1->$\Pi_{mid,moviename}(\sigma_{moviename='input'}(Movie))$
  Result2 ->$\Pi_{mid,status}(\sigma_{location='input'}(Theater))$
  Result3 ->Result1 ∞ Result2
  Availabilty_movie ->$\Pi_{moviename,status}(Result3)$

- Search for showtimings for particular movie

Result1->$\Pi_{mid,moviename}(\sigma_{moviename='input'}(Movie))$
Result2->$\Pi_{mid,showtimings}(Theater))$
Result3 ->Result1 ∞ Result2
Showtimings_movie->$\Pi_{moviename,showtimings}(Result3)$

- Search for theater w.r.t movie & location
  Result1->$\Pi_{mid}(\sigma_{moviename='input'}(Movie))$
  Result2 ->$\Pi_{mid,theatername}(\sigma_{location='input'}(Theater))$
  Result3 ->Result1 ∞ Result2
  Theaters ->$\Pi_{theaternames}(Result3)$

DDL & DML Commands:

DDL-> CREATE

Create table **movie**(mid varchar2(10),moviename varchar2(30),ccidint(10),release_datedate,durationtimestamp,PRIMARY KEY(mid))

Create table **theater**(thid varchar2(10),thname varchar2(30),mid varchar2(10),srid int(10),show_timings varchar2(30),location varchar2(50),status varchar2(10),PRIMARY KEY(thid) ,FOREIGN KEY(SRID) SCREENROOM(SRID))

Create table **screenroom**(srid int(10),capacity int(20) ,PRIMARY KEY(srid))

Create table **seat**(seatid int(10),srid int(10),postion int(20),PRIMARY KEY(srid),FOREIGN KEY(srid) SCREENROOM(srid))

Create table **ticket**(tid int(10),mid varchar(10),type varchar(10),discounts varchar(10),srid int(10), seatid int(10),typeid int(10) , FOREIGN KEY(mid) MOVIE(mid), FOREIGN KEY(SRID) SCREENROOM(SRID), FOREIGN KEY(seatid) SEAT(seatid), FOREIGN KEY(typeid) TICKET_TYPE(typeid))

Create table **ticket_type**(type_id int(10),type_description varchar(10),cost int(10), PRIMARY KEY(type_id int(10)))

Create table booking(bid int(10),userid int(10),tid int(10),typeid int(10),no_of_ticket,s int(10), PRIMARY KEY(type_id int(10)))

Similarly we need to create tables for ticket ,user ,ticket_type ,booking,Review ,Cast_crew

DML->{INSERT,UPDATE}

Insert into **movie** values(‘m1’,’2012’,’cc1’,’2012-01-01’,’02:36:55’)
Insert into **theater** values(‘th1’,’max’,’m1’,’s1’,’2:30-5:06:55’,’kc,mo 64112’,’available’)
Update **theater** set status=’close’ where thid=’th1’

In this manner we can insert and update tables based on requirement.

References:

http://www.cse.ohio-state.edu/~gurari/course/cse670/cse670Ch2.xht

http://en.wikipedia.org/wiki/Data_flow_diagram

http://www.imdb.com/

http://en.wikipedia.org/wiki/Codd's_12_rules