iMovies

Online Movie Reservation System

A PROJECT REPORT

Submitted for the partial fulfillment of the project for the subject, Database Management Systems

By

K GOKUL RAJ - 19BCE0522

RAHUL SANJEEV - 19BCE0952

MOHIT SINHA - 19BCE0947

Under the guidance of

Prof. KURUVA LAKSHMANNA

VIT®
UNIVERSITY
(Estd. u/s 3 of UGC Act 1956)

School of Computer Science and Engineering October,2020

ABSTRACT

Our project 'Online Movie Reservation System' focuses on creating and easy, user friendly system to book movies. Entertainment industry with movies being such an important part of our lives we are always excited to go to the theatres. We want to create an easy online site that makes the movie-booking experience wholesome.

The program when runs can be used by two individuals. We have named them 'Admins' and 'Users'. The program would allow users to select a movie from the different ones that are played on a daily basis in different theatres. Users can choose the theatre of their choice and select a time slot available. The availability will be based on the Seat Availability of that slot in the theatre of their choice. After choosing these, they would move into payments and hence pay for their tickets

Admins are allowed to add a new movie in a theatre. They are allowed to add a new theatre location. Admins are also allowed to choose the number of seats in the theatre and also the pricing range for the seats in their theatre. Hence, admins can create new theatre locations that the users can choose from.

Program starts off with a login page. Users select the option and is asked to choose a location. This opens up a new option to choose theatres that have a wide range of movie options, mainly 'Currently Running', 'Coming Soon' and 'Closed'. Users would then choose their desired movie. This would show a short description of the movie and there would be a 'Book Now' option that they would choose to move forward. On clicking this option, they have to choose the Date and time of the movie based on the seat availability. The seats have been divided into 4 categories of different prices; they are – 'Normal', 'Deluxe', 'Gold' and 'Platinum'. These are of varying price ranges and users must choose their desired seat choice and give the number of seats they want. The users have to put in the details of each booked participant (i.e. Name, Age and Gender). After this they would move to the Payment gateway, Users can choose to pay through- 'Credit Card', 'Debit Card', 'UPI' and 'Net Banking'. They have to then pay and then 'PAYMENT SUCEEDED' or 'PAYMENT FAILED' option would come up. Users also have choice of putting their movie in a cart and pay for it at a later time.

When the admins log in, they must first put in the Movie Name. They should fill in the details of the Movie- Title, Director Name, Cast Details and a small description of the movie. They should fill in the Theatre ID and Movie ID. When they want to fill in a new Theatre location, admins must fill in location details and also the seat numbers and prices. These prices and the number of seats is under the admin's discretion.

This was a brief abstract of the working model of this project- 'Online Movie Reservation System'.

ACKNOWLEDGEMENT

First of all, we thank the almighty who showered his immense blessings on us, that has helped us to complete this project with success.

Our heartful thanks to Prof.Kuruva Lakshmanna for giving us the necessary environment to acquire knowledge and skills.

Our sincere and warmest thanks for his valuable and inspiring guidance and encouragement given throughout the period of this project. We feel indebted to him for guiding our project with reviews, evaluations, guidance and suggestions offered throughout this project.

We also express my whole hearted thanks to our parents for their encouragement to bring this project to a successful completion.

1. INTRODUCTION

Introduction to Online Movie Reservation System:

Welcome to newly designed website movie ticket booking is a faster, cleaner and a tad more personal website, specially designed to make your booking experience better. Log on, navigate and find out for yourselves and if time permits leave your valuable feedback. Customers may view the contents of any movie show at any time and may book any movie ticket as needed. The program automatically calculates the subtotal and grand total. When a visitor decides to finally book the ticket, the order information including the buyer's name, address and billing instruction is stored in the database securely and payment has been made. The combo booking is also provided at the time of booking the ticket and there's a wonderful facility of delivering the combos at your seat when you are watching the movie.
You need to register a new user whenever you have first visited or site then for future it will be stored in our database permanently and you can book you movie ticket at any time you want with this username and password.

Online Movie Reservation System is a user-friendly site that allows users to choose their favourite movie at their desired time. They may choose the type of seats they want and also choose the type of Payment Gateway they would like to use for booking the tickets. We have also given an Admin Login that allows admins to add a new movie or a new theatre location that users can choose from later.

1.1. Purpose Scope and Applicability of Project:

The main purpose of this project is to provide a reliable, secure, efficient and user friendly environment to the customers and management authorities. Also benefit to the customer with efficient and faster service.

The Project "Online Movie Ticket Booking System" as a wide scope as it is generalized software and can be easily used in any ticket booking process system with little or no change. The Changes in software can be easily accommodated. The addition and deletion of the modules in software can be easily adjusted. This project has a lot of scope for further enhancement too. This project can save money and efforts in managing the record, just a mouse click can make the task easy and faster

1.2. Objective of Project

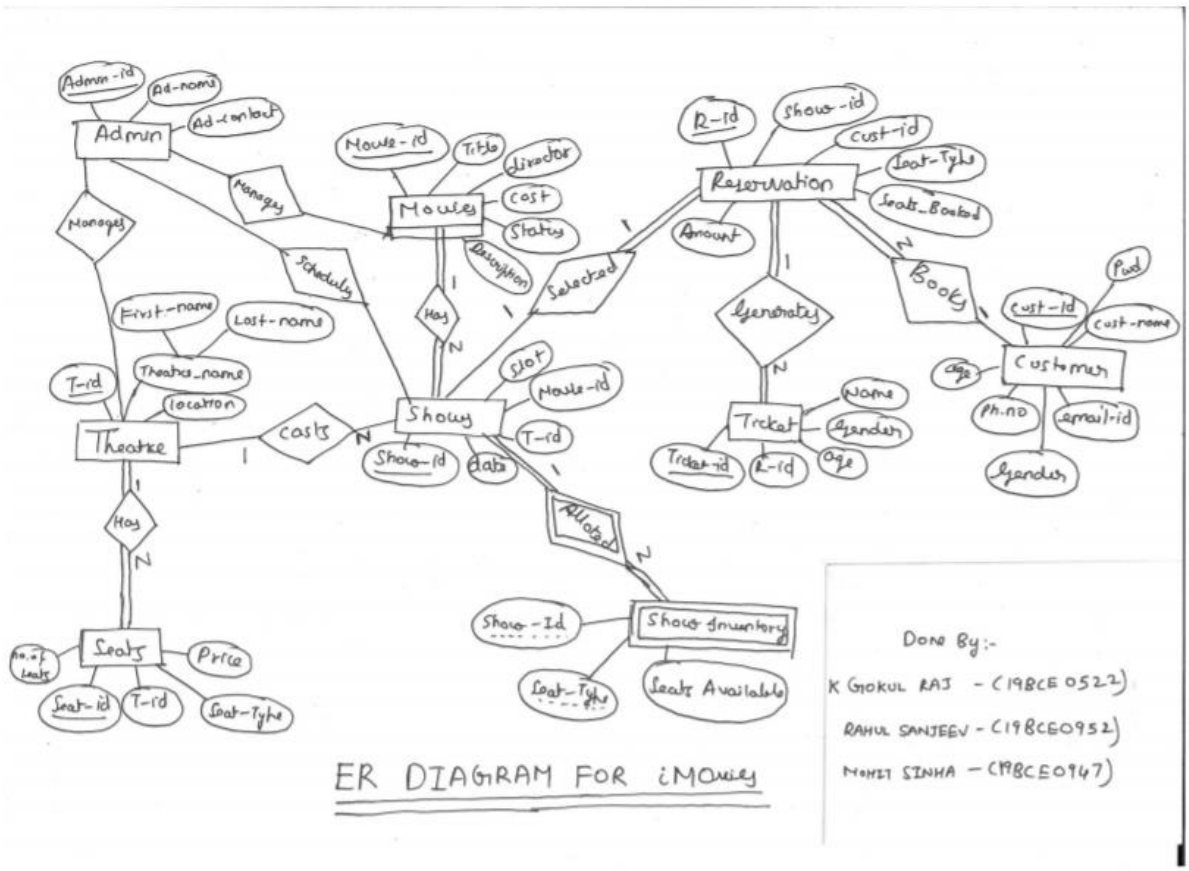
The project "Online Movie Ticket Booking System" is dedicated to the general Requirements of multiplex theaters. The main objective of the project is to create an Online Movie Ticket Booking processing that allows customers to know about new movies, their schedules, cinema locations, class and ticket price etc. In the booking process when customer selects his city then cinemas of that city are filtered. In next step he/she selects his desired cinema where he/she wish to see movie, then selects movie and other details like show date, show time, class and no of tickets. Based on given parameters a graphical layout of seat status is visible to the customer. Now customer can select his desired seat location and number of seats. The Administrator will be able to see all booked and canceled tickets.

The main objectives of "Online Movie Ticket Booking System" project are as follows:

1) Facility to store the information of new customer, different types of movie show timing, ticket rates of different types on show class etc.

2) And Give the admin the function to add movie, theatre, seats and create shows and manage shows.

ER model



Tables and Constraints (Generated by DBeaver software):

Key : PRI = PRIMARY KEY

MUL = FOREIGN KEY

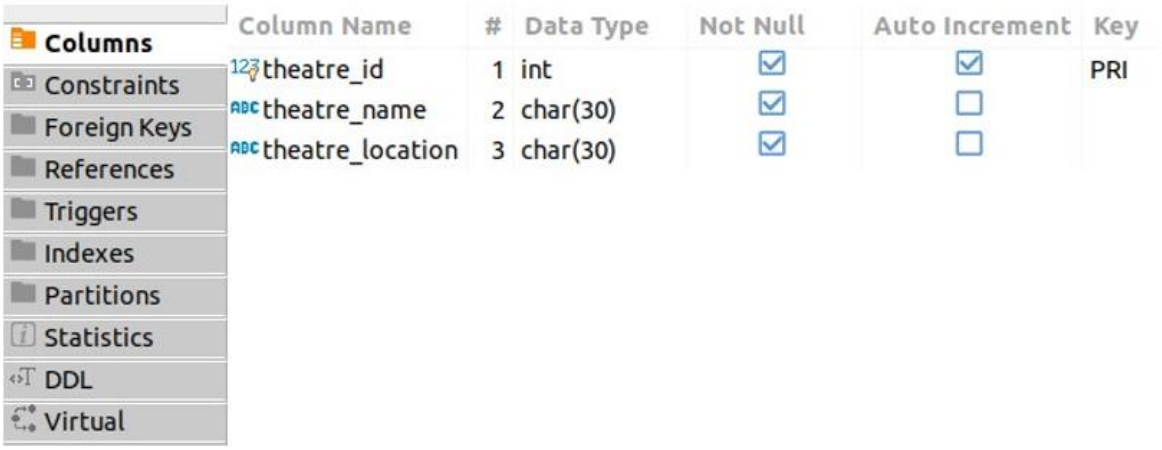
WE HAVE HOSTED OUR PROJECT IN INTERNET:

LINK FOR THE WEBSITE: iMovies2020.epizy.com

CLOUD WEBSITE USED FOR HOSTING – infintiyfree.net

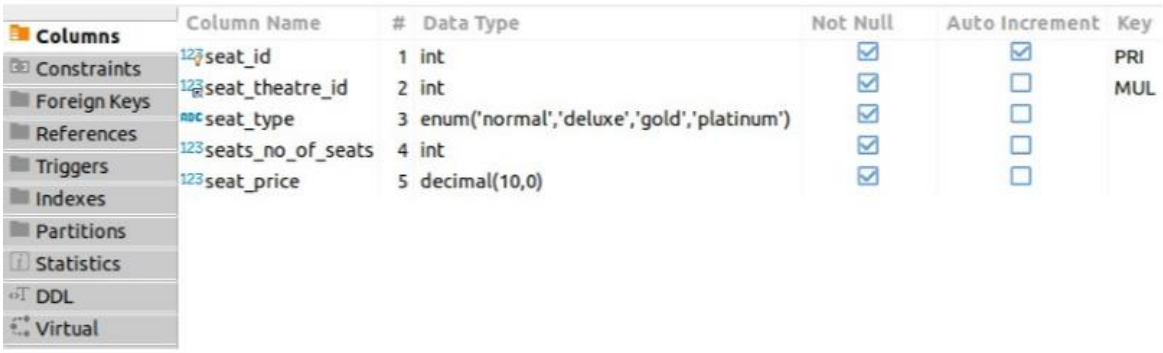
2. Tables Used:

- **Theatre Table**



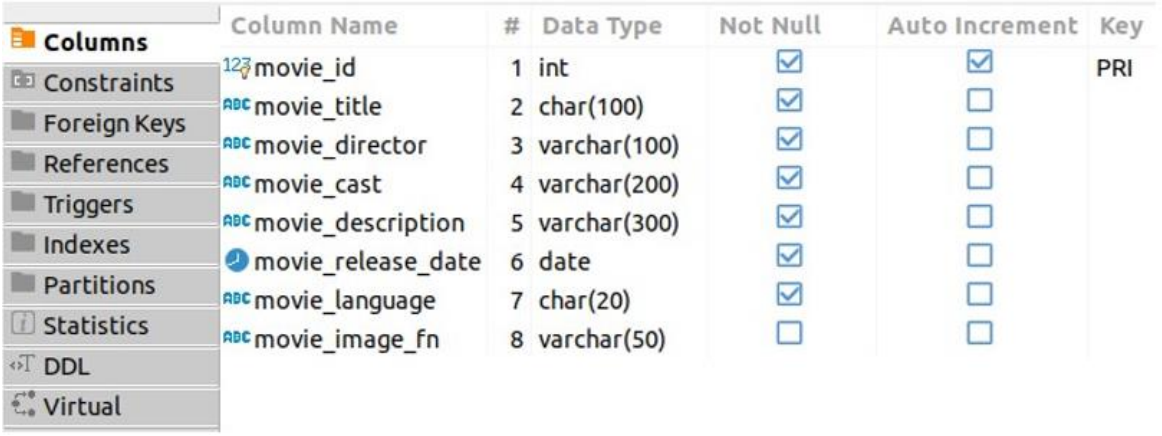
Column Name	#	Data Type	Not Null	Auto Increment	Key
theatre_id	1	int	☑	☑	PRI
theatre_name	2	char(30)	☑	☐	
theatre_location	3	char(30)	☑	☐	

- **Seats Table**



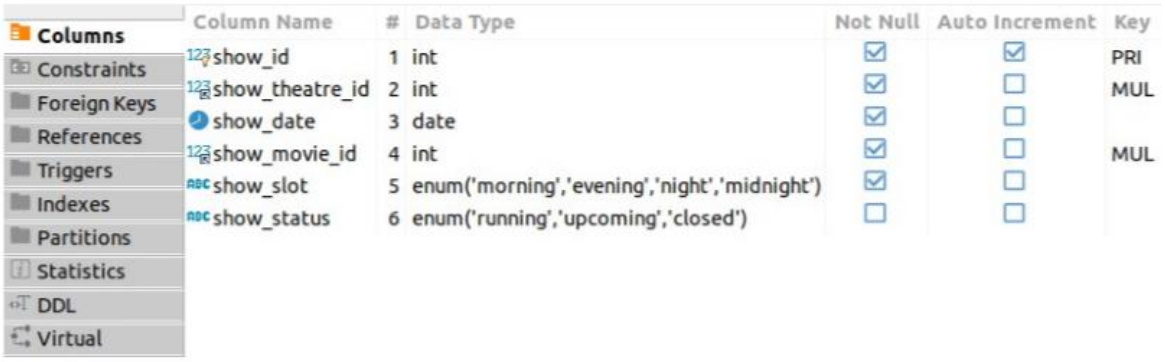
Column Name	#	Data Type	Not Null	Auto Increment	Key
seat_id	1	int	☑	☑	PRI
seat_theatre_id	2	int	☑	☐	MUL
seat_type	3	enum('normal','deluxe','gold','platinum')	☑	☐	
seats_no_of_seats	4	int	☑	☐	
seat_price	5	decimal(10,0)	☑	☐	

- **Movies Table**



Column Name	#	Data Type	Not Null	Auto Increment	Key
movie_id	1	int	☑	☑	PRI
movie_title	2	char(100)	☑	☐	
movie_director	3	varchar(100)	☑	☐	
movie_cast	4	varchar(200)	☑	☐	
movie_description	5	varchar(300)	☑	☐	
movie_release_date	6	date	☑	☐	
movie_language	7	char(20)	☑	☐	
movie_image_fn	8	varchar(50)	☐	☐	

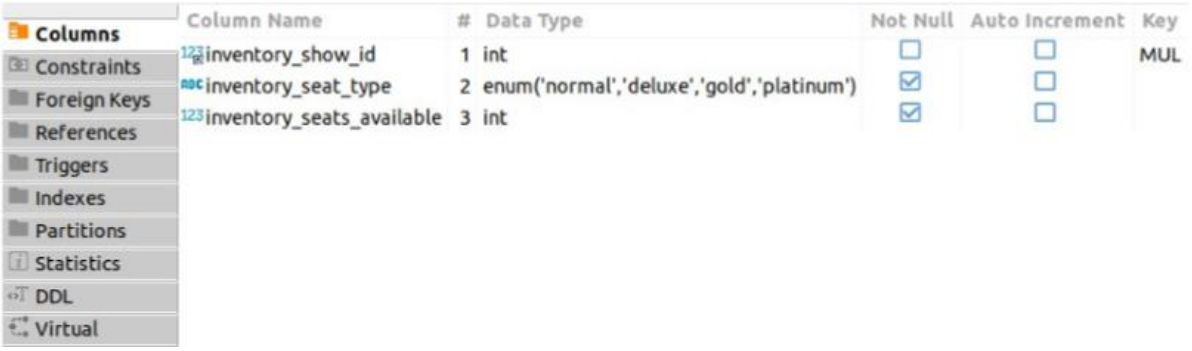
- **Shows Table**



Columns
Constraints
Foreign Keys
References
Triggers
Indexes
Partitions
Statistics
DDL
Virtual

Column Name	#	Data Type	Not Null	Auto Increment	Key
show_id	1	int	☑	☑	PRI
show_theatre_id	2	int	☑	☐	MUL
show_date	3	date	☑	☐	
show_movie_id	4	int	☑	☐	MUL
show_slot	5	enum('morning','evening','night','midnight')	☑	☐	
show_status	6	enum('running','upcoming','closed')	☐	☐	

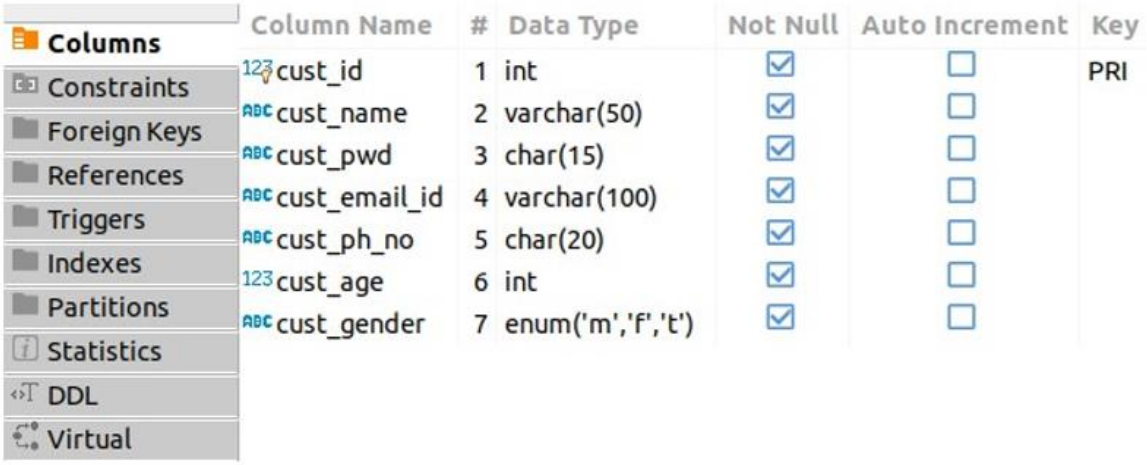
- **Show Inventory Table**



Columns
Constraints
Foreign Keys
References
Triggers
Indexes
Partitions
Statistics
DDL
Virtual

Column Name	#	Data Type	Not Null	Auto Increment	Key
inventory_show_id	1	int	☐	☐	MUL
inventory_seat_type	2	enum('normal','deluxe','gold','platinum')	☑	☐	
inventory_seats_available	3	int	☑	☐	

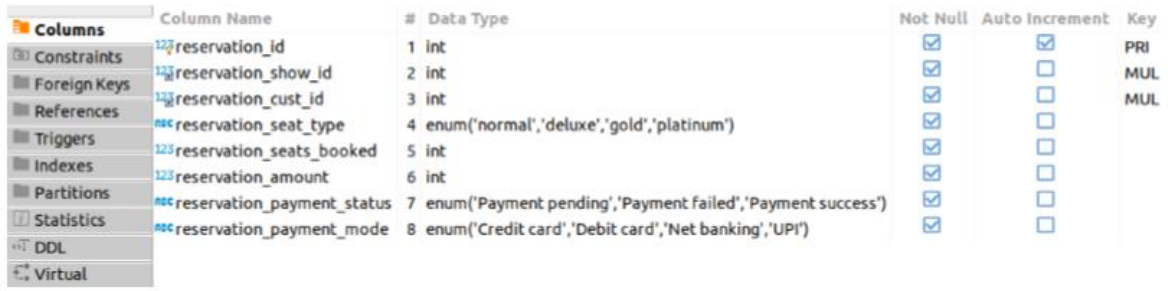
- **Customer Table**



Columns
Constraints
Foreign Keys
References
Triggers
Indexes
Partitions
Statistics
DDL
Virtual

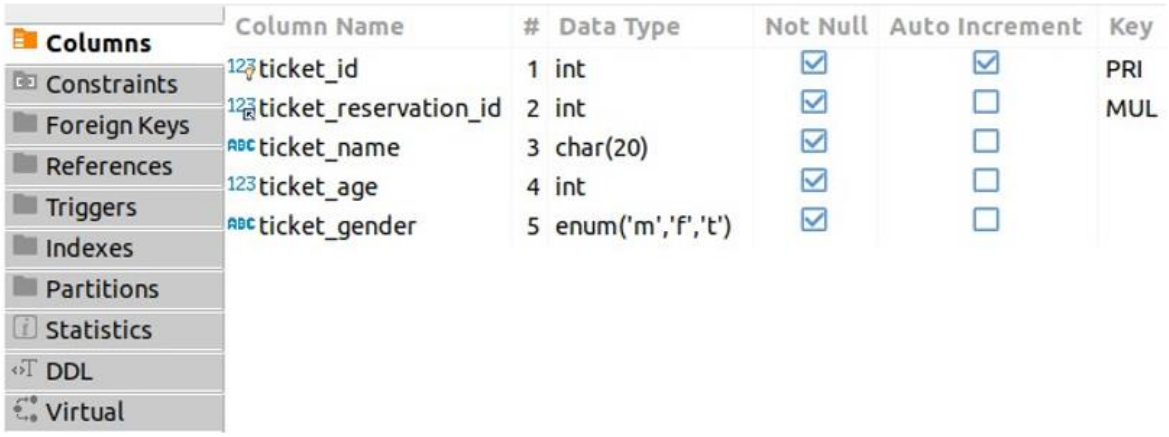
Column Name	#	Data Type	Not Null	Auto Increment	Key
cust_id	1	int	☑	☐	PRI
cust_name	2	varchar(50)	☑	☐	
cust_pwd	3	char(15)	☑	☐	
cust_email_id	4	varchar(100)	☑	☐	
cust_ph_no	5	char(20)	☑	☐	
cust_age	6	int	☑	☐	
cust_gender	7	enum('m','f','t')	☑	☐	

- **Reservation Table**



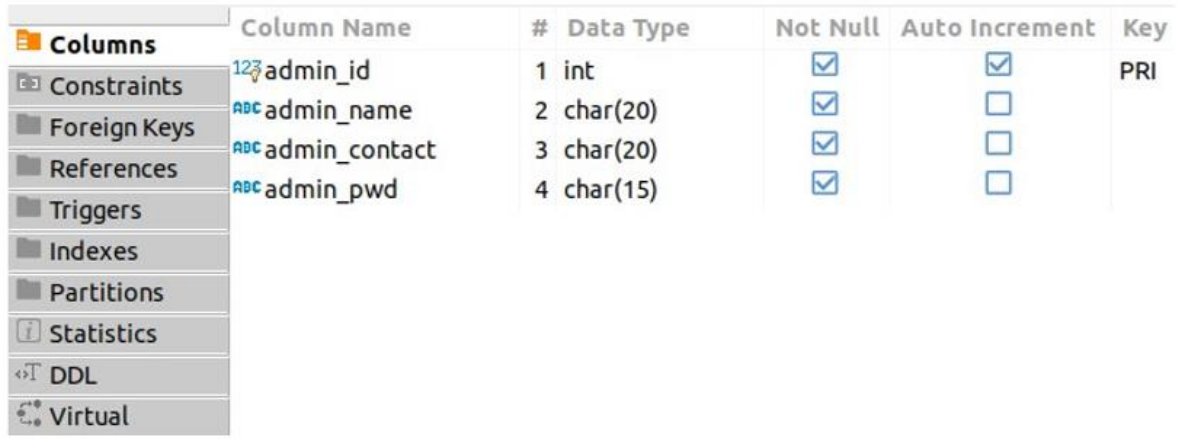
Column Name	#	Data Type	Not Null	Auto Increment	Key
reservation_id	1	int	☑	☑	PRI
reservation_show_id	2	int	☑	☐	MUL
reservation_cust_id	3	int	☑	☐	MUL
reservation_seat_type	4	enum('normal','deluxe','gold','platinum')	☑	☐	
reservation_seats_booked	5	int	☑	☐	
reservation_amount	6	int	☑	☐	
reservation_payment_status	7	enum('Payment pending','Payment failed','Payment success')	☑	☐	
reservation_payment_mode	8	enum('Credit card','Debit card','Net banking','UPI')	☑	☐	

- **Ticket Table**



Column Name	#	Data Type	Not Null	Auto Increment	Key
ticket_id	1	int	☑	☑	PRI
ticket_reservation_id	2	int	☑	☐	MUL
ticket_name	3	char(20)	☑	☐	
ticket_age	4	int	☑	☐	
ticket_gender	5	enum('m','f','t')	☑	☐	

- **Admin Table**



Column Name	#	Data Type	Not Null	Auto Increment	Key
admin_id	1	int	☑	☑	PRI
admin_name	2	char(20)	☑	☐	
admin_contact	3	char(20)	☑	☐	
admin_pwd	4	char(15)	☑	☐	

Step by step ER to relational mapping

STEP 1: **Mapping of Regular Entity Type.**

For each regular entity type in the ER schema, create a relation R that includes all the simple attributes of the entity.

Choose one of the key attributes of the entity as the primary key for the relation, R.

If the chosen key of the entity is composite, the set of simple attributes will together form the primary key of R.

STEP2: **Mapping of Weak Entity Type**

For each weak entity type (W) in the ER schema with its owner entity type i.e E, create a relation R & then include all simple attributes of W as attributes of the relation, R.

Include a foreign key attributes of Relation, R as the primary key attribute of the relation that correspond to the owner entity type.

The primary key of relation, R is the combination of the primary key of the owner and the partial key of the weak entity type, W, if any.

STEP3: **Mapping of Binary 1:1 Relation Type**

For each binary 1:1 relationship type R in the ER schema, identify the relations S and T that correspond to the entity types that participates in the relation, R.

STEP4: **Mapping of Binary 1:N Relationship Type.**

For regular binary 1:N relationship type R, identify the relation, let's say, S that represents the participating entity type at the N-side of the binary relationship type.

Include a foreign key in S as the primary key of the relation T that represents other entity type participating in R.

Include the simple attributes of the 1:N relation as the attributes of S.

STEP5: **Mapping of Binary M:N Relationship Types.**

For regular binary M:N relationship type R, create another relation S to represent the relation, R.

Include a foreign key attribute in S as the primary key of the relations that represent the participating entity types and then their combination will be the primary key.

Also include the simple attribute of the M:N relationship type as attribute of S.

STEP6: **Mapping of Multivalued attribute.**

For each multivalued attribute A, create a relation R.

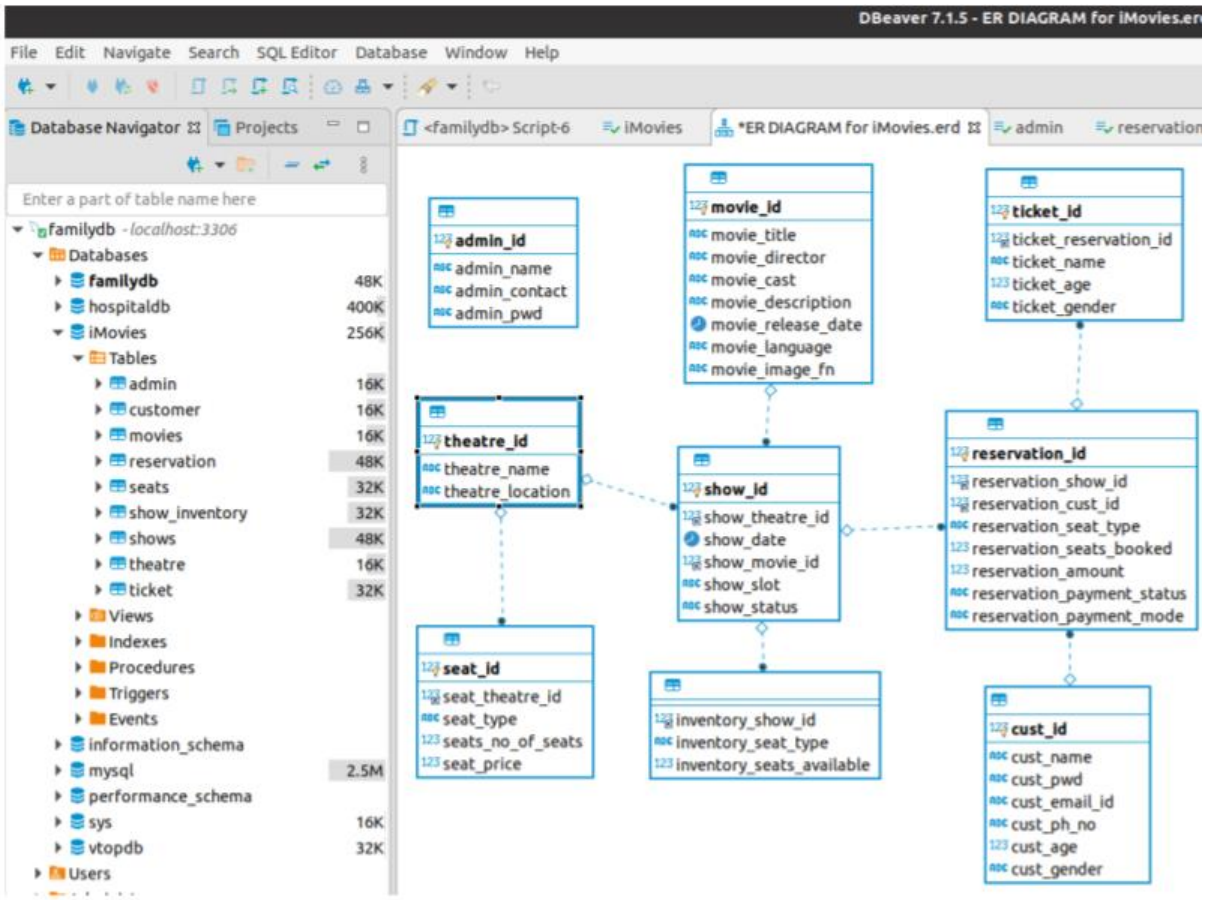
This relation R will include the attributes corresponding to A, also add the primary key attribute K as the foreign key in R of the relation.The primary key of R is the combination of A and primary key, K. If the multivalued attribute is composite, we include their simple components.

STEP7: **Mapping of N-ary Relationship Type.**

For each n-ary relationship type R, where n>2, we create a relationship S to represent R. Include a foreign key attribute in S as the primary key of the relation that represent the participating entity types.

Also include the simple attributes of the N-ary relationship as the attributes of S.

Relational Scheme Diagram –



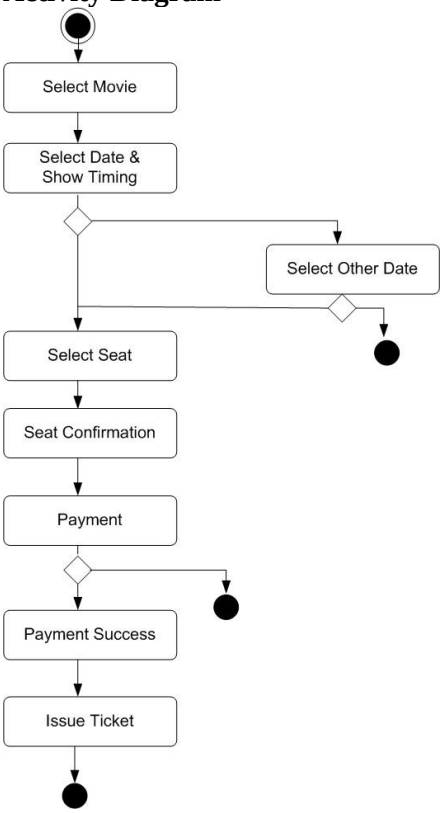
3.SYSTEM DESIGN

Systems design is simply the design of systems. It implies a systematic and Rigorous approach to design—an approach demanded by the scale and complexity of many Systems problems. The main purpose of the system is to design a security and efficient System for online movie ticket booking portal. It must be utilize very low system resources.

Online Movie Ticket Booking Process

The “Online Movie Ticket Booking System” manages all the movie ticket booking process and ticket amount payment process. This maintains information of list of all currently running movies in different multiplex of the same company in various locations. The process of booking movie tickets online is kept very simple. The customer can go to the online ticket booking portal where they can see list of all movies running currently in different cinemas situated in various location and cities, different show timing etc. The customer selects their desired cinema, movie, preferred date, show timing, desired no of tickets and see the booking status / available seats. Then customer register it self on the portal and reserve his desired no of seats. In the next step system ask customer for payment online. After successful payment the system will generate movie ticket.
All registered customers can login to the ticket booking portal and see their previous ticket booking status.

Activity Diagram



Description of model

SCENARIOS:

Customer Registration Scenario

All customers' detailed information will be stored in the database and they will be given unique account numbers. The system will provide facility of adding new customer record, updating old customer records and deleting any customer record .This information is further used by all the modules as required. The customer record is very important in the project. This customer record will be used by the project every where it is needed.

Security & Movie Ticket Management Scenario

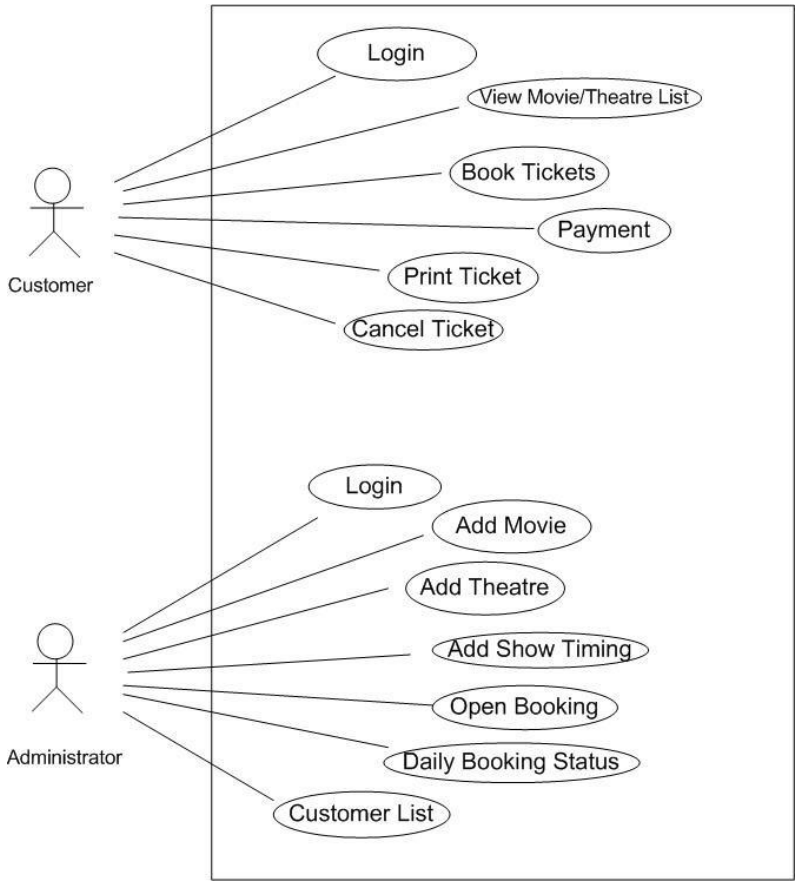
In this scenario we will manages all information of all the master records of the project such as cinema master, movie master, screen master, show timing etc. This master information is used by the project as required by the project. This module also sends the book information to all the modules as needed. Online Movie Ticket Booking System & Cinema schedule maker module uses this information for preparing cinema schedule.

Movie Show Timing Management Scenario

In this scenario of the project online movie ticket booking project. As the name indicates we will manage all the records related with the making a movie show timing schedule for showing different movies in different screens and time, in different cinemas and their show timings etc. When a new movie has to be added in cinema, this module is used to prepare a daily show plan for this movie. The running show booking can be controlled by this module. When a movie has completed enough business, its booking can be closed by this module. Show booking for next date begins when we add next date in this module.

Seat Confirmation & Payment Scenario

All seat allocation and ticket booking is done here. This module take customer details and show date time and allocate seats and calculate booking amount, which is further used for payment & ticket printing.
In payment & ticket print scenario the system prepares ticket based on the information given by the seat allocation & ticket booking module. This is the final process of ticket booking process.



USE CASE DIAGRAM

Important SQLs used in this project

Purpose	Populate landing page with latest movies details
Pages Impacted	Index2.php, adminIndex.php
Brief description	Fetch latest 6 movies and related details like movie images to display as carousel
SQL statement	#1

select movie_id, movie_title, movie_image_fn, movie_cast

```
"select movie_id,movie_title,movie_image_fn,movie_cast
        from $t_Movies order by movie_id desc limit 6"
```

Purpose	Validate login credentials for customer and admin
Pages Impacted	Index2.php, adminIndex.php
Brief description	Fetch user id and password from customer or admin table to compare with HTML form variables
SQL statement	#2

```
"Select * from $t_customer where cust_username =  '$uid' and cust_pwd = '$pwd'";
```

Purpose	Add new theatre and related details
Pages Impacted	addTheatre.php, insertTheatre.php
Brief description	Based on the HTML form variables, insert new theatre details, followed by inserting seat types, # of seats and price details
SQL statement	#3

Insert new theatre into theatre table:

```
"INSERT INTO $t_theatre (theatre_name, theatre_location) VALUES ('$tName','$tCity')"
```

Based on the theatre id (auto generated), insert seat details

```
"INSERT INTO $t_seats (seat_theatre_id, seat_type,seats_no_of_seats,seat_price)
        VALUES ($t_theatre_id,'$sType',$sNumbers,$sPrice)"
```

Purpose	Add new movie and related details
Pages Impacted	addMovie.php, insertMovie.php
Brief description	Based on the HTML form variables, insert new movie and related details
SQL statement	#4

```
"INSERT INTO $t_movies (movie_title, movie_cast, movie_director, movie_image_fn, movie_language, movie_release_date, movie_description) VALUES ('$iname','$icast' , '$idirect','$ifn', '$ilang', date('$ireldate'),'$idesc' )"
```

Purpose	Add new show and modify the inventory
Pages Impacted	addShow.php and insertShow.php
Brief description	Fetch city theatre , the movie , status and the running dates from their respective tables to compare with HTML form variables and add it if there is no conflict (existing show)
SQL statement	#5

Select distinct city location:

```
"SELECT distinct theatre_location from $t_theatre") as $rs1){
                $cities[] = array( "tcity" => $rs1['theatre_location']
```

Based on the selected loaction filter and display the theatres:

```
"SELECT theatre_location, theatre_id, theatre_name from $t_theatre") as $rs2){
                $theatres[$rs2['theatre_location']][] = array("tid" => $rs2['theatre_id'], "tname" => $rs2['theatre_name']
```

To check the conflict if any existing shows are present for the given combination of dates,slot and theatres.

```
"SELECT b.theatre_name, b.theatre_location, a.show_date, c.movie_title, a.show_slot
                from $t_shows as a, $t_theatre as b, $t_movies as c where a.show_theatre_id = b.theatre_id
                and a.show_movie_id = c.movie_id and a.show_theatre_id = $showTheatre and a.show_date = '$checkDate' and
                a.show_slot = '$showSlots[$i]' and a.show_status <> 'closed' "
```

Insert into shows database if there is no conflict:

```
"INSERT INTO $t_movies (movie_title, movie_cast, movie_director, movie_image_f
n, movie_language,
            movie_release_date, movie_description)  VALUES ('$iname', '$icast'
 , '$idirect','$ifn', '$ilang',
            date('$ireldate'),'$idesc' )"
```

Purpose	Manage the shows (closing shows of running movies)
Pages Impacted	manageShow.php manageShow2.php manageShow3.php manageShow4.php
Brief description	Fetch city theatre, movie scheduled all the shows will be displayed and admin can close the shows as per his convenience
SQL statement	#6

Select the city where movies are scheduled:

```
"SELECT distinct theatre_location from $t_theatre") as $rs1){
                $cities[] = array( "tcity" => $rs1['theatre_location']
```

Select the theatres in city where movies are scheduled:

```
"SELECT theatre_location, theatre_id, theatre_name from $t_theatre") as $rs2){
                $theatres[$rs2['theatre_location']][] = array("tid" => $rs2['t
heatre_id'], "tname" => $rs2['theatre_name']
```

Purpose	To display all the reservations booked in a show
Pages Impacted	ticketShow.php ticketShow2.php ticketShow3.php
Brief description	Fetch city theatre , movie and using the show id find the number of seats in reservation table and names of each customer in tickets table
SQL statement	#7

To show all running shows of selected slot:

```
"select show_id, show_date, show_slot, show_status from $t_shows
        where show_theatre_id = $theatreID and show_movie_id=$tMovieID and sho
w_status <> 'closed'"
```

To show number of tickets:

```
"select sum(reservation_seats_booked) as booked from $t_reservation a,
                                $t_shows b where a.reservation_show_id=$tshowi
d and a.reservation_show_id=b.show_id
                                and b.show_date='$tshowdate' and b.show_slot =
 '$tshowslot'
                                group by reservation_show_id")-
>fetch()['booked']
```

Normalization:

1NF(RULES):

- Each column should contain atomic values.
- A column should contain values that are of the same type.

• Each column should have a unique name.

As our table follows these conditions our table would look like:

Functional Dependencies:
Tables:

admin:
FD1: admin_id -> admin_name, admin_contact, admin_pwd, admin_username

customer:
FD1: id -> cust_name, cust_pwd, cust_email_id, cust_ph_no, cust_age, cust_gender, cust_username

movies:
FD1: movie_id -> movie_title, movie_director, movie_cast, movie_description, movie_release_date, movie_language, movie_image_fn

reservation:
FD1: reservation_id -> reservation_show_id, reservation_cust_id, reservation_seat_type, reservation_seats_booked, reservation_amount, reservation_payment_status, reservation_payment_mode

ticket:
FD1: ticket_id -> ticket_reservation_id, ticket_name, ticket_age, ticket_gender

schedule:
FD1: schedule_id -> schedule_screen_id, schedule_date, schedule_movie_id, schedule_slot, schedule_status

FD2 : schedule_seat_id -> schedule_seat_type, schedule_seats_available

screens:
FD1: screen_id -> screen_name, screen_location

FD2: screen_seat_id -> seat_type, seats_no_of_seats, seat_price

2NF(RULES):
• It should be in the 1 st normal form.

• It should not have Partial Dependencies.

All tables are in 2NF form so no change in schema

3NF(RULES):
• It should be in 2 nd normal form.
• It should not have Transitive Dependencies.

Functional Dependencies:
Tables:

admin:
FD1: admin_id -> admin_name, admin_contact, admin_pwd, admin_username

customer:
FD1: id -> cust_name, cust_pwd, cust_email_id, cust_ph_no, cust_age, cust_gender, cust_username

movies:
FD1: movie_id -> movie_title, movie_director, movie_cast, movie_description, movie_release_date, movie_language, movie_image_fn

reservation:
FD1: reservation_id -> reservation_show_id, reservation_cust_id, reservation_seat_type, reservation_seats_booked, reservation_amount, reservation_payment_status, reservation_payment_mode

ticket:
FD1: ticket_id -> ticket_reservation_id, ticket_name, ticket_age, ticket_gender

schedule:
FD1: schedule_id -> schedule_screen_id, schedule_date, schedule_movie_id, schedule_slot, schedule_status

FD2 : schedule_seat_id -> schedule_seat_type, schedule_seats_available

As we can see schedule has transitive dependency nature it further gets splitted to following tables

Column Name	#	Data Type	Not Null	Auto Increment	Key	Default	Extra
schedule_id	1	int	☑	☑	PRI		auto_increme
schedule_screen_id	2	int	☑	☐	MUL		
schedule_date	3	date	☑	☐			
schedule_movie_id	4	int	☑	☐	MUL		
schedule_slot	5	enum('morning','	☑	☐			
schedule_status	6	enum('running','u	☐	☐			
schedule_seat_type	7	enum('normal','d	☑	☐			
schedule_seats_availa	8	int	☑	☐			
schedule_seat_id	9	int	☑	☐			

==>>

Column Name	#	Data Type	Not Null	Auto Increment	Key	Default	Extra
show_id	1	int	☑	☑	PRI		auto_increme
show_theatre_id	2	int	☑	☐	MUL		
show_date	3	date	☑	☐			
show_movie_id	4	int	☑	☐	MUL		
show_slot	5	enum('morning','	☑	☐			
show_status	6	enum('running','u	☐	☐			

screens:
FD1: screen_id -> screen_name, screen_location

FD2: screen_seat_id -> seat_type, seats_no_of_seats, seat_price

+

Column Name	#	Data Type	Not Null	Auto Increment	Key
inventory_show_id	1	int	☐	☐	MUL
inventory_seat_type	2	enum('normal','d	☑	☐	
inventory_seats_available	3	int	☑	☐	

schema for shows : (show_id, show_theatre_id, show_dates, how_movie_id, show_slotshow_status)

FD1 : show_id -> show_theatre_id, show_dates, how_movie_id, show_slotshow_status

schema for show_inventory:(inventory_show_id, inventory_seat_type, inventory_seats_available)

FD1 : inventory_show_id -> nventory_seat_type, inventory_seats_available

screens:
FD1: screen_id -> screen_name, screen_location

FD2: screen_seat_id -> seat_type, seats_no_of_seats, seat_price

As we can see screens has transitive dependency nature it further gets splitted to following tables

Column Name	#	Data Type	Not Null	Auto Increment	Key	Default	Extra
screen_id	1	int	☑	☑	PRI		auto_increme
screen_name	2	char(30)	☑	☐			
screen_location	3	char(30)	☑	☐			
seat_type	4	enum('normal','d	☑	☐			
seats_no_of_seats	5	int	☑	☐			
seat_price	6	decimal(10,0)	☑	☐			
screen_seat_id	7	int	☑	☐			

==>

Column Name	#	Data Type	Not Null	Auto Increment	Key	Default	Extra
theatre_id	1	int	☑	☑	PRI		auto_increme
theatre_name	2	char(30)	☑	☐			
theatre_location	3	char(30)	☑	☐			

+

Column Name	#	Data Type	Not Null	Auto Increment	Key	Default	Extra
seat_id	1	int	☑	☑	PRI		auto_increme
seat_theatre_id	2	int	☑	☐	MUL		
seat_type	3	enum('normal','d	☑	☐			
seats_no_of_seats	4	int	☑	☐			
seat_price	5	decimal(10,0)	☑	☐			

schema for theatre :(theatre_id, theatre_name, theatre_location)

FD1: theatre_id -> theatre_name, theatre_location

schema for seats : (seat_id, seat_theatre_id, seat_type, seats_no_of_seats, seat_price)

FD2: seat_id -> seat_theatre_id, seat_type, seats_no_of_seats, seat_price

BCNF(RULES):
• It should be in 3 rd normal form.
• No non-prime attribute should derive any prime attribute

All our tables satisfies all these conditions so it remains the same

4. Run-Through : iMovie Ticket Booking:

As a customer:

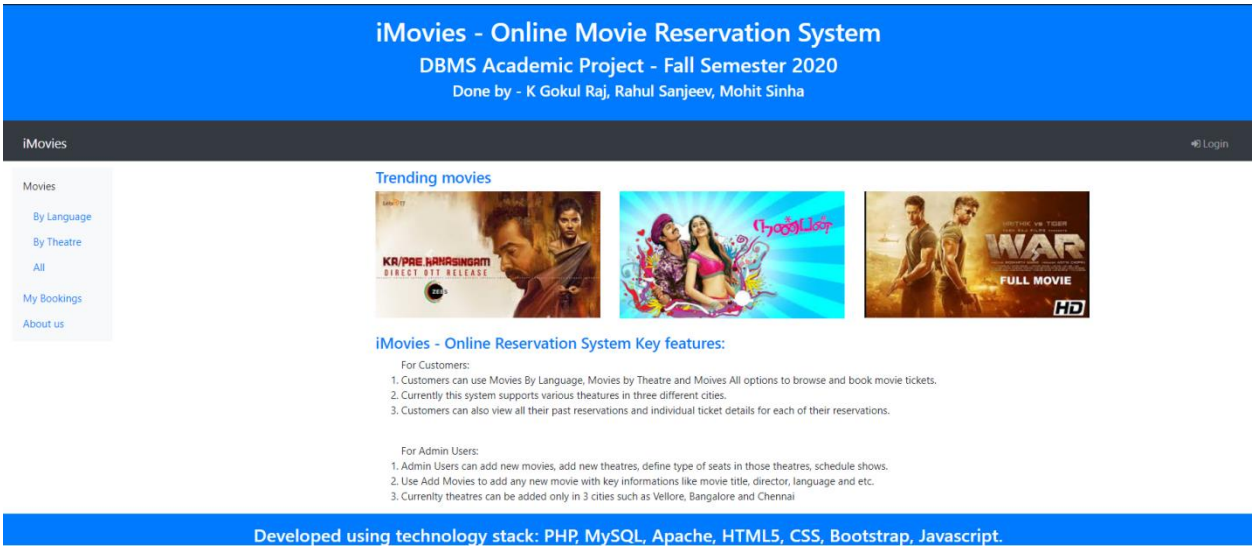
Index2.php Page (Landing Page):

After you enter the website URL you will be directed to index2.php page which consists of three main containers (segments) :

1) **Horizontal NAV BAR : which contains the menu for directing to index2.php – iMovies**
   Login – which directs the customer to login page

2) **Vertical NAV BAR : Contains the list of functions/options available for customer to make it more attractive, user friendly website and the list of options which user can make use of**

3) **Carousel : Top 6 trending movies will be displayed to user in carousel mode**

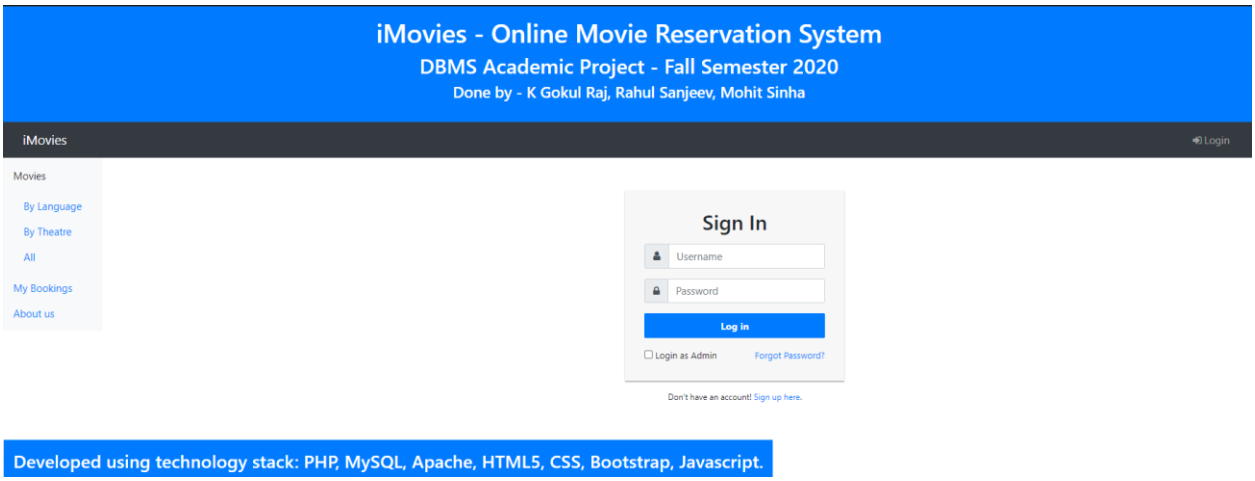


Registration.php Page:

1) **User can login with his credentials which will set the login menu in the NAV BAR with his name**

2) **If user has not yet registered he can do it by clicking on the Sign up here href link**

3) **If the user is a admin (which will be registered by an authorized person who has access to the respective databases directly.**

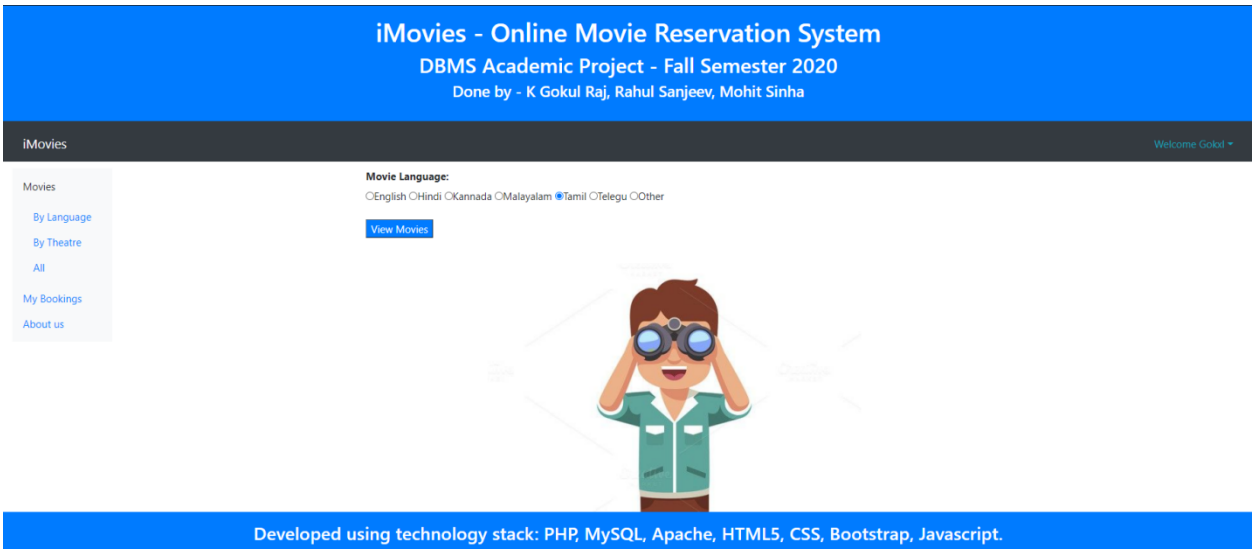


By Language:

viewLang.php:

1) **If the customer wants to see movies of specific movies he can make use of the By Language option.**

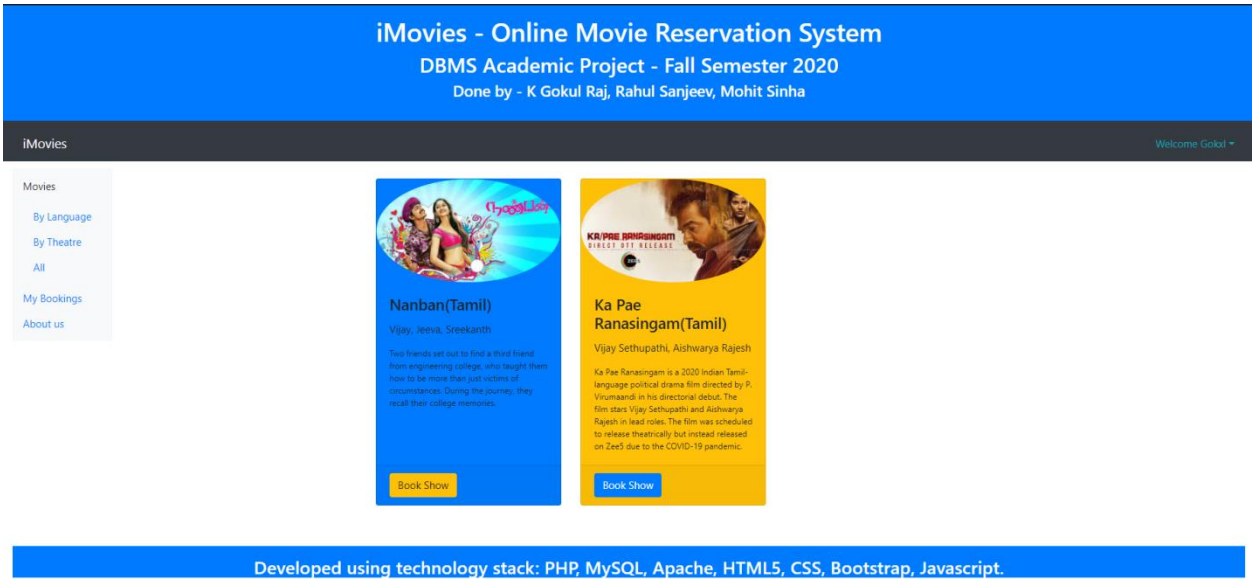
2) **He can select the language which he wants. The language id will be sent to the next page**



viewLang2.php:

1) **The movies of the selected language will be displayed and the user can click the Book Show button to proceed for the next step of booking**

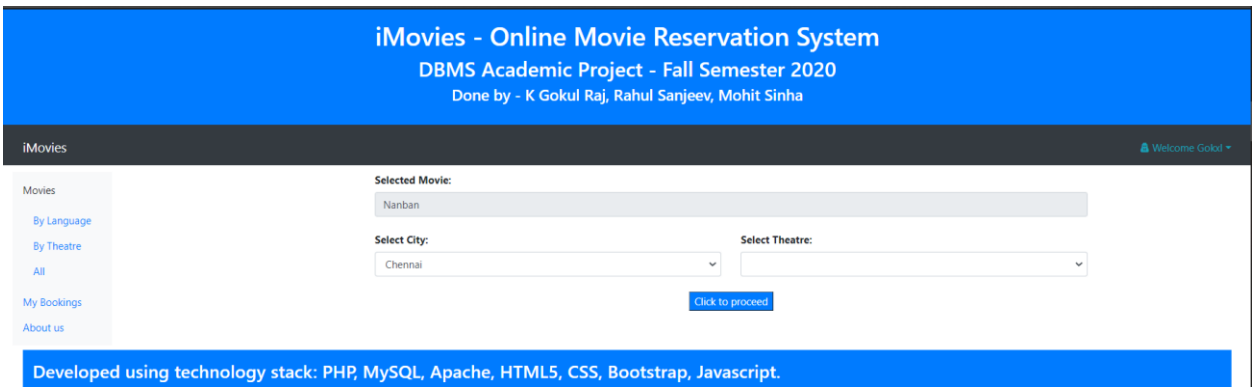
2) **The movie_id will be sent to the next page**



bookShow.php:

1) **The selected movie of selected language will be displayed as a read-only input so that data integrity is not violated, if the user wants to change his choice of language he has to again start from language.**

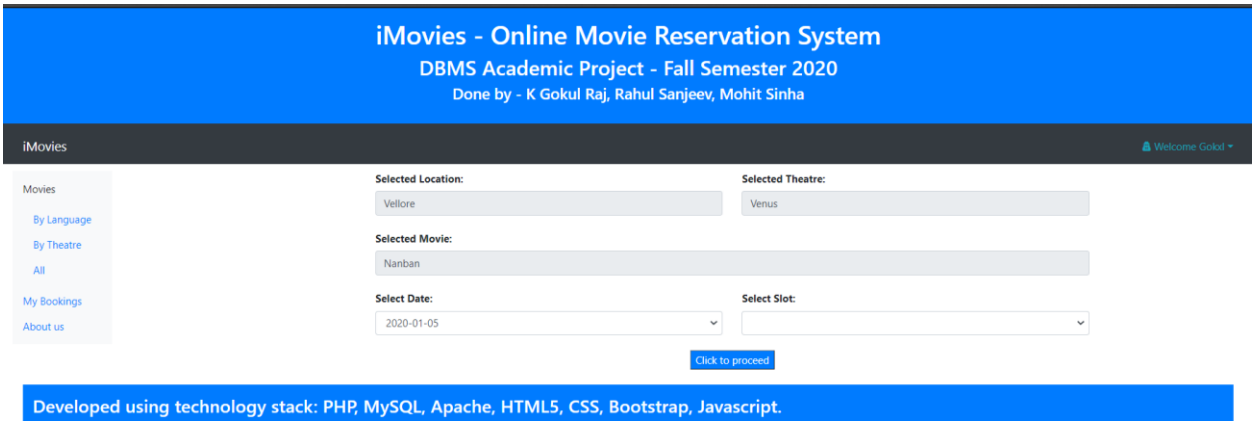
2) **Customer has to select the city and the respective theatre present in the selected city.**
   theatre_id will be sent to next page



bookShow2.php:

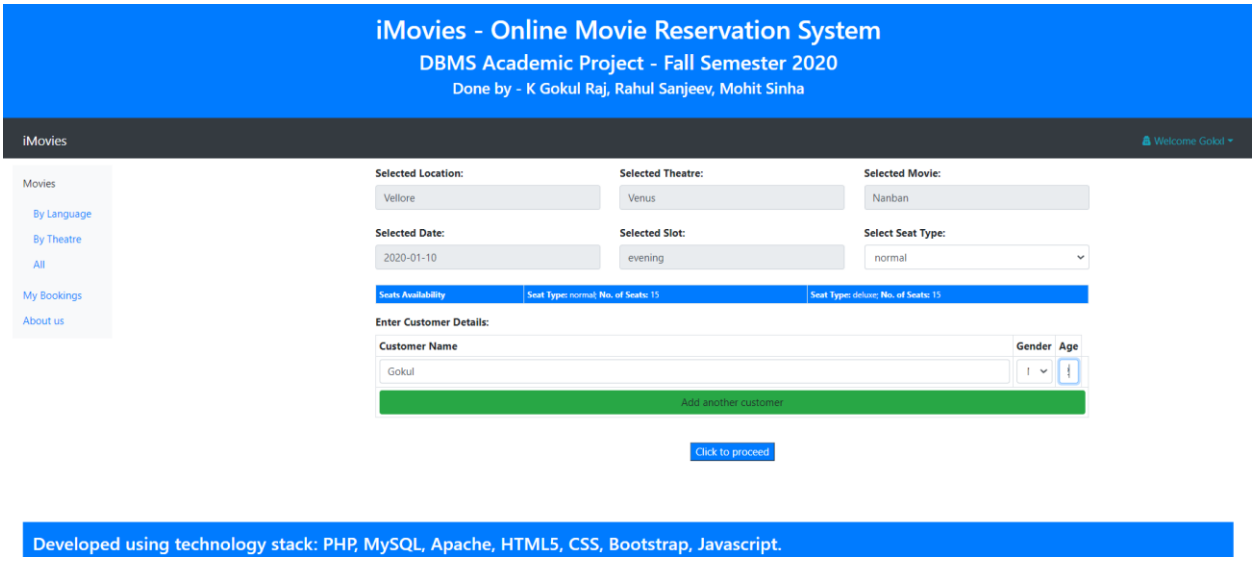
1) **Customer has to select the date and slot which are available for the respective date.**

2) **All the previous id's and the date and slot will be sent to next page**

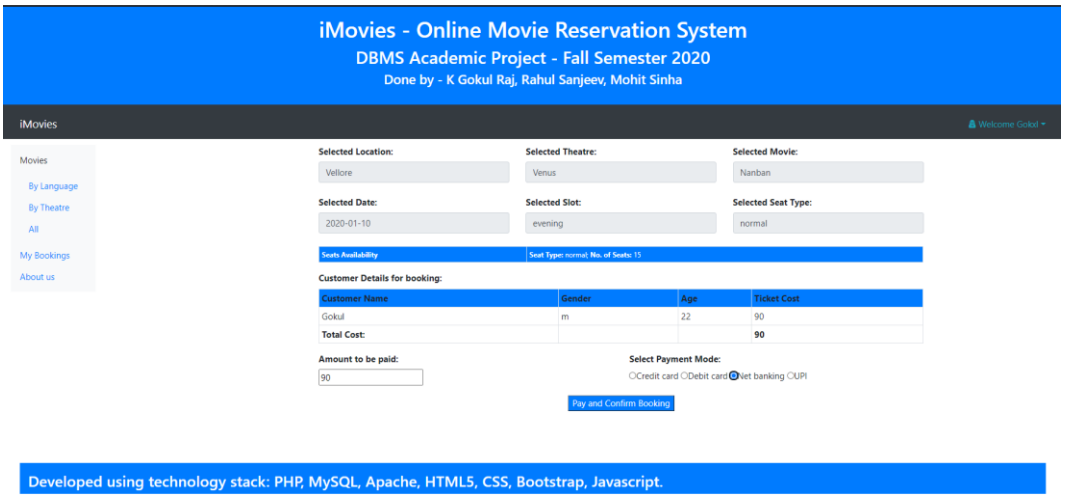


bookShow3.php:

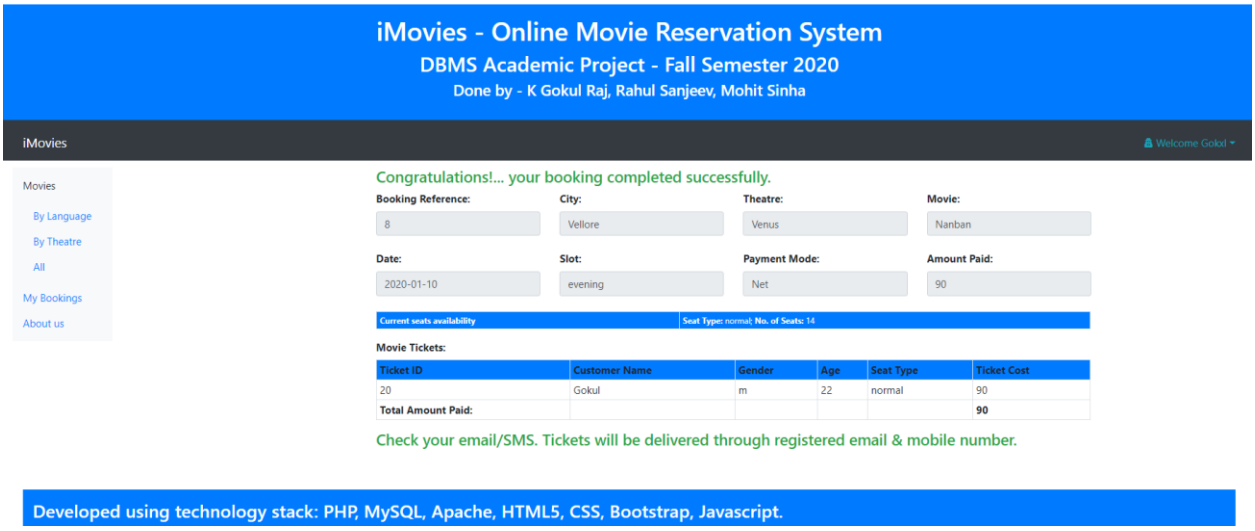
1) **Customer has to enter the details of the persons for whom he is booking tickets**



bookShow4.php: the committing page open where the user can select the mode of payment and commit all the options which he had selected



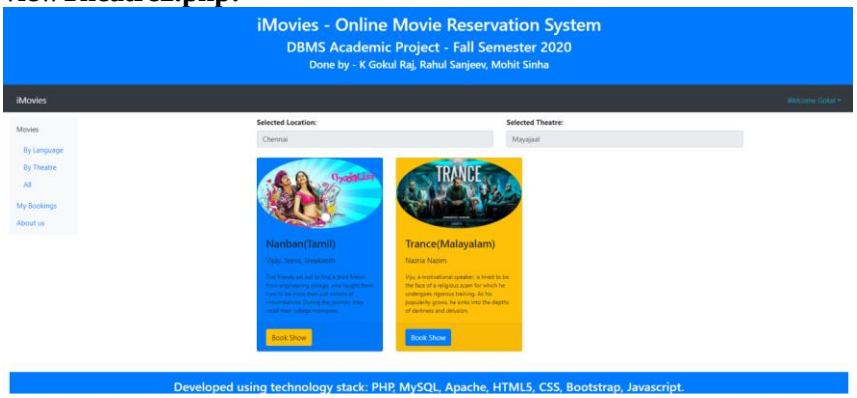
bookFinal.php:



By Theatre:

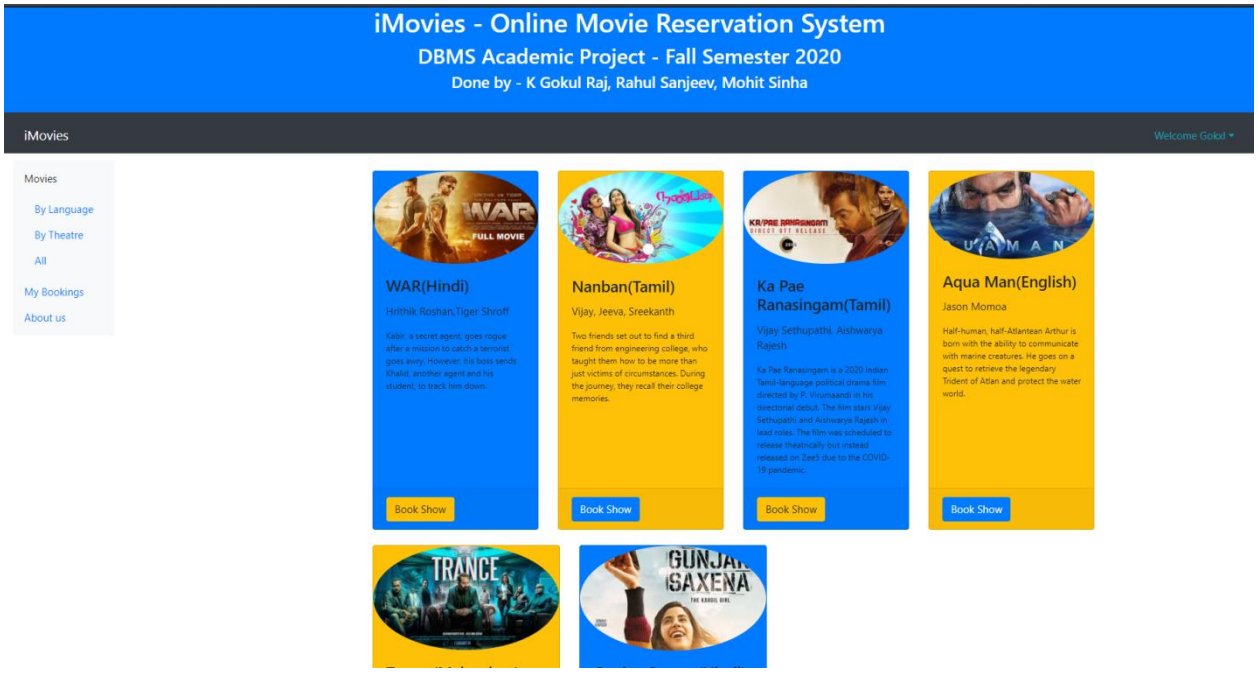
viewTheatre.php: To view movies based on the theatre you want to go and watch

viewTheatre2.php:

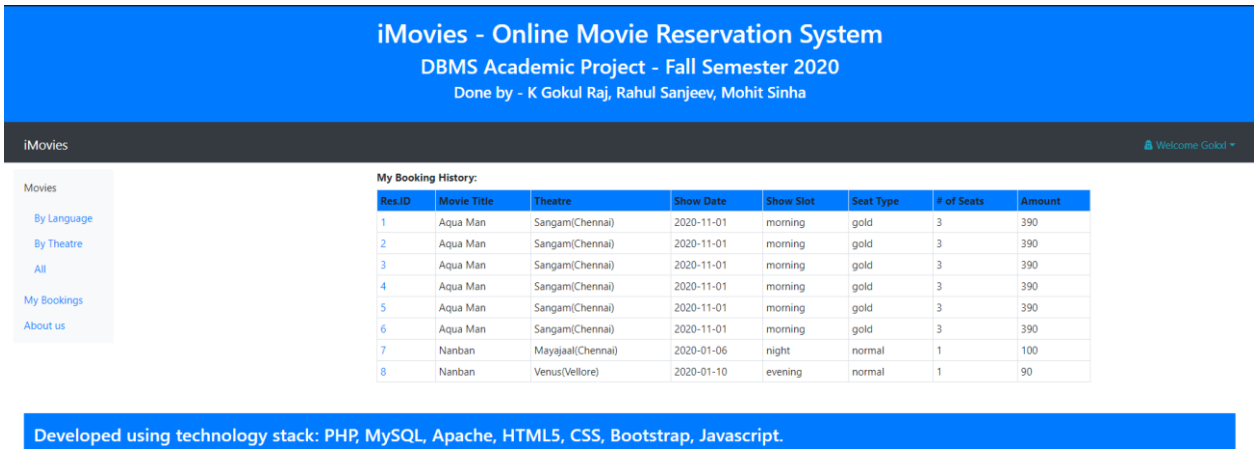


All:

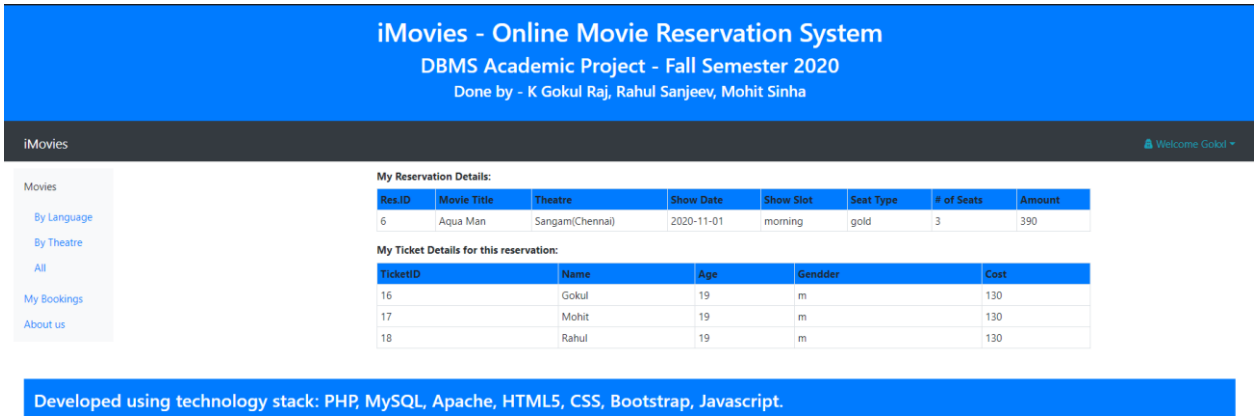
viewAll.php: Display all the movies added by the admin to this Booking system portal



myBookings.php: To display all the bookings of the user



myTicket.php:

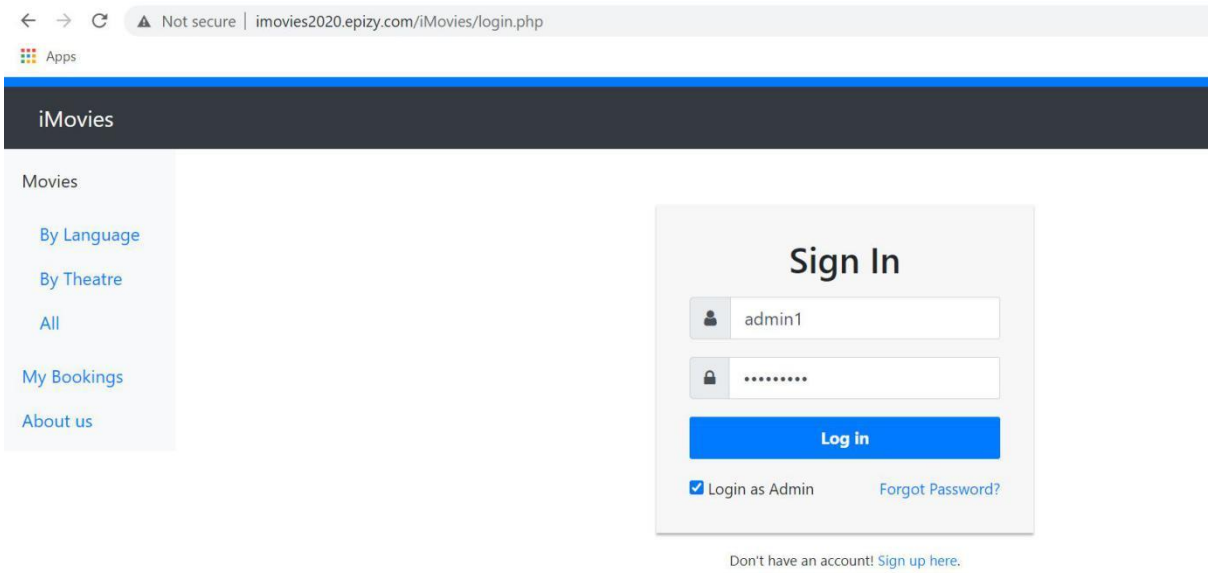


Login as admin:

login.php

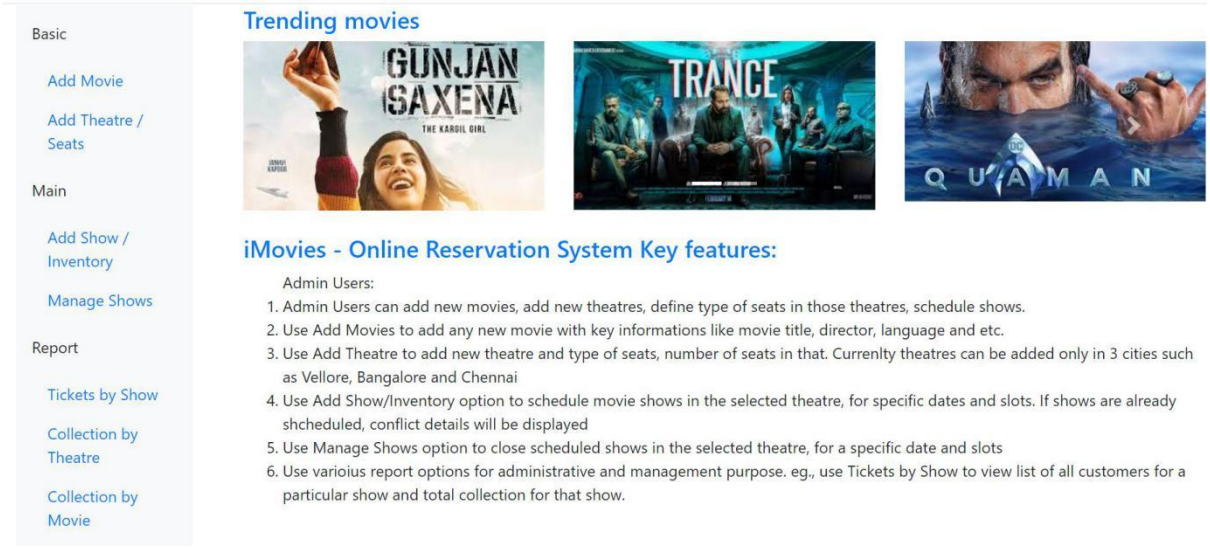
1) **user has to click on the login as admin checkbox and enter his credentials to enter as admin**

2) **admin credentials will be given by the manager of the portal and owner of the organization**



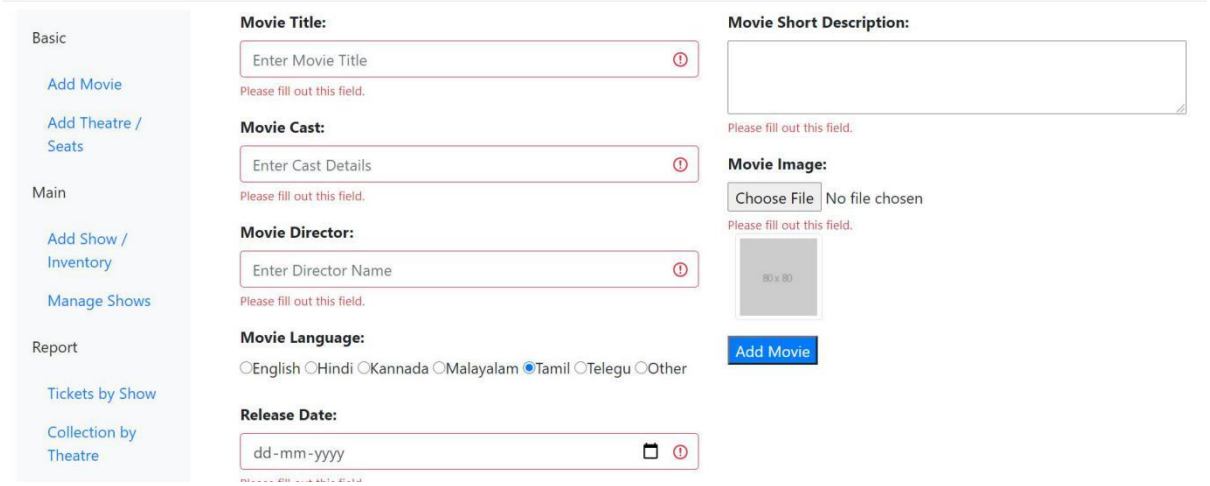
adminIndex.php

After logging as and admin you will be directed to a login home page. Here you will have option to add movies, add theaters, change show time and slot, modify seats etc.



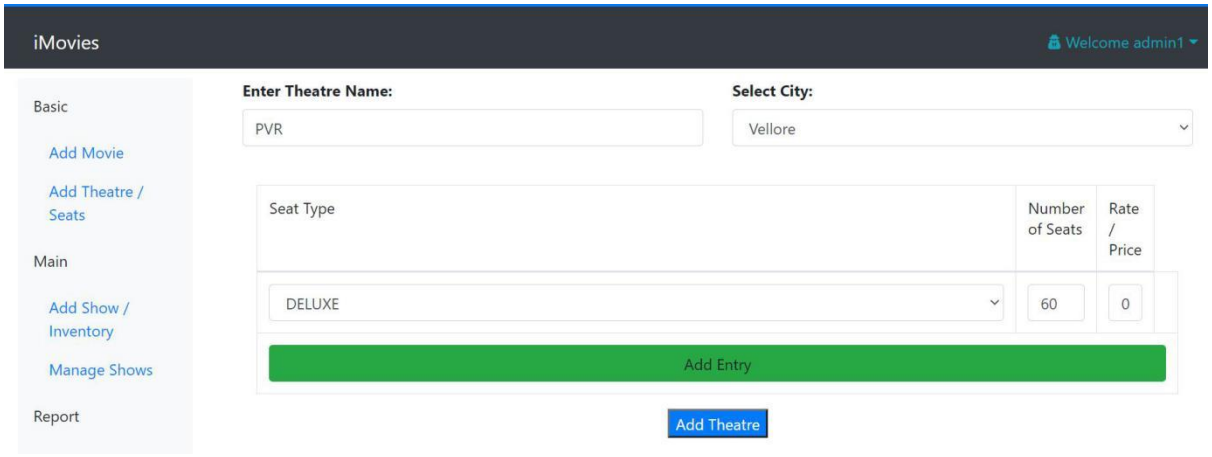
addMovies.php

In add Movies you can add movies by filling up the details like: title, description, cast, director, Image, language and release date.



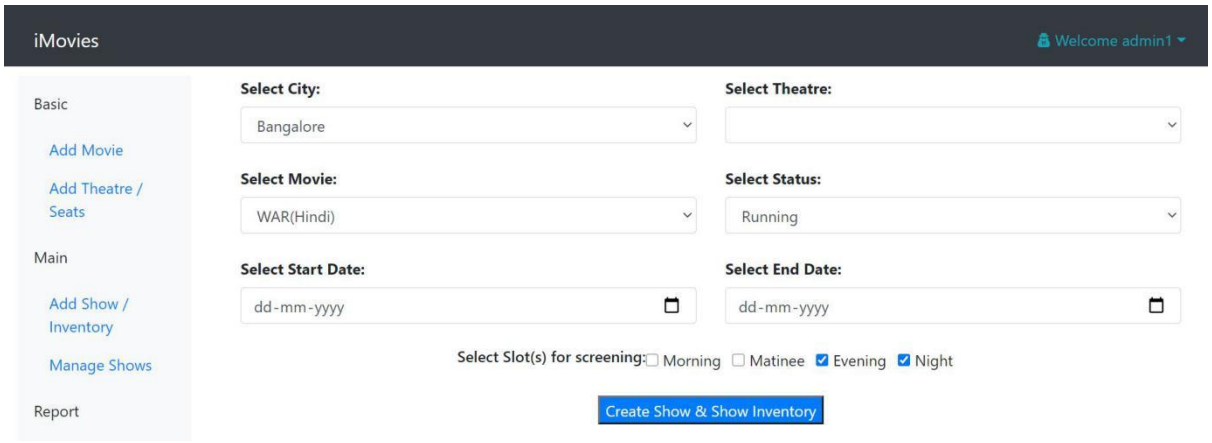
addTheatre.php

In add theater admin can add a new theater by filling theater name, city, seat type.



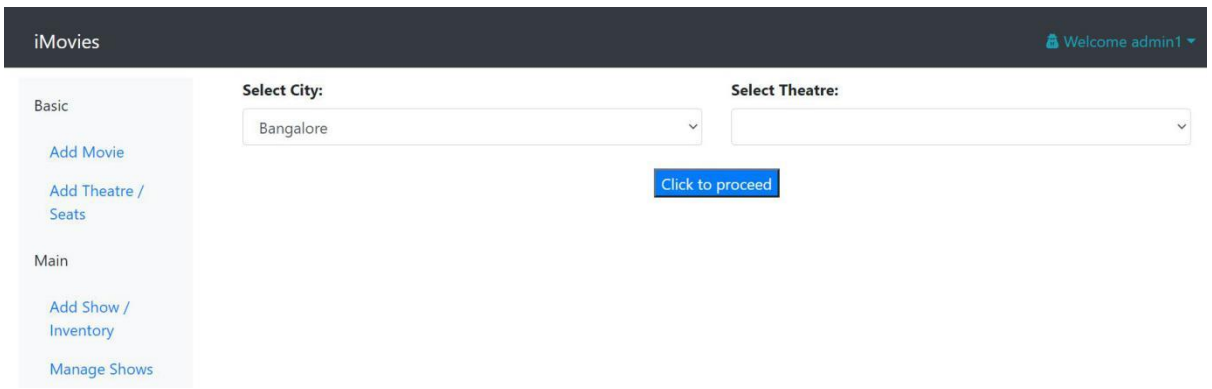
addShow.php

In add show admin can add a new show to the site by selecting city, theater, movies, status, start date, and end date.



manageShow..php

In this page admin can manage different theaters by selecting city and theater name.



TECHNOLOGY USED:

1. Front end:

- HTML- Easy to use for the designing aspect of our site
- Bootstrap - is the most popular HTML, CSS and Javascript Framework to develop easy-to-use websites

2. Back end:

- PHP - It is a general-purpose scripting language which is extensively used for Web Development. It is implemented mainly in C language with some components of C++. We used it for the backend programming of our project

3. Database Connection:

- MySQL server 8.0 - It is used for the Database Connection of all the tables in the site to the backend. It allows to navigate through databases easily

4. IDE for Database Connection:

- dBeaver - is a SQL client software application and a database administration tool. For relational databases it uses the JDBC application programming interface to interact with databases via a JDBC driver.

5. IDE for HTML and Bootstrap:

- Visual Studio Code - is a free source-code editor made by Microsoft for Windows, Linux and macOS. Features include support for debugging, syntax highlighting, intelligent code completion, snippets, code refactoring, and embedded Git.

6. Website used for hosting:

- Infinity free.net – It is a free hosting platform for websites with PHP and MySQL. We have uploaded all files and database to this.