VISVESVARAYA TECHNOLOGICAL UNIVERSITY

JnanaSangama, Belagavi-590018.

A Project Report on

"MOVIE TICKET MANAGEMENT SYSTEM"

Submitted in the partial fulfillment of the requirements for the award of the Degree of

Bachelor of Engineering

In

Computer Science and Engineering

by

Mohammed Suhail R (1OX20CS078)
Navya T L (1OX20CS084)

Under the guidance of
Dr Raghu R

Professor, Department of Computer Science & Engineering

Department of Computer Science and Engineering

THE OXFORD COLLEGE OF ENGINEERING

Bommanahalli, Bangalore 560068

2022-2023

THE OXFORD COLLEGE OF ENGINEERING

Hosur Road, Bommanahalli, Bengaluru-560068

(Affiliated to VISVESVARAYA TECHNOLOGICAL UNIVERSITY, Belagavi)

Department of Information Science and Engineering

CERTIFICATE

Certified that the project work entitled **"MOVIE TICKET MANAGEMENT SYSTEM"** carried out by, **Mohammed Suhail R (1OX20CS078), Navya T L(1OX20CS084),** bonafide students of **The Oxford College Of Engineering, Bengaluru** in partial fulfillment for the award of Degree of Bachelor of Engineering in Computer Sceince And Engineering if the **Visvesvaraya Technological University**, Belagavi, during the year **2022-2023**. It is certified that all corrections / suggestions indicated for Internal Assessment have been incorporated in the Report deposited in the departmental library. The project report has been approved as it satisfies the academic requirements in respect of Project work prescribed for the said Degree.

Dr Raghu R	**Dr. R.CH.A.NAIDU**	**Dr. N. Kannan**
Project Guide	Professor & Head of CSE	Principal

1. Internal Examiner:_______

2. External Examiner:______

THE OXFORD COLLEGE OF ENGINEERING

Hosur Road, Bommanahalli, Bengaluru-560068

(Affiliated to VISVESVARAYA TECHNOLOGICAL UNIVERSITY, Belagavi)

Department of Computer Science and Engineering

DECLARATION

We the students of Fifth semester B.E, at the Department of Computer Science and Engineering, **The Oxford College Of Engineering, Bengaluru** declare that the project entitled **"MOVIE TICKET MANAGEMENT SYSTEM"** has been carried out by us and submitted in partial fulfillment of the course requirements for the award of degree in Bachelor of Engineering in Computer Science and Engineering discipline of **Visvesvaraya Technological University, Belagavi** during the academic year **2022-2023.** Further, the matter embodied in dissertation has not been submitted previously by anybody for the award of any degree or diploma to any other university.

Name	USN	Signature
MOHAMMED SUHAIL R	1OX20CS078	
NAVYA TL	1OX20CS084	

Date: Place: Bangalore

AKNOWLEDGEMENT

The satisfaction and euphoria that accompany the successful completion of any task would be incomplete without the mention of people who made it possible whose constant guidance and encouragement crowned our effort with success.

We consider ourselves proud to be a part of The Oxford family, the institution that stood by our way in all our endeavours. We have a great pleasure in expressing our deep sense of gratitude to the founder chairman **late Sri S. Narasa Raju** and to our chairman **Sri S. N. V. L. Narasimha Raju** for providing us with a great infrastructure and well-furnished labs.

We would like to express our gratitude to **Dr. N. Kannan,** Principal, The Oxford College of Engineering for providing us a congenial environment and surrounding to work in.

Our hearty thanks to **Dr. R Ch A Naidu,** Professor & Head, Department of Computer Science and Engineering, The Oxford College of Engineering for his encouragement and support.

Guidance and deadlines play a very important role in successful completion of the project report on time. We convey our gratitude to (**Raghu R),** Professor, Department of Computer Science and Engineering for having constantly monitored the completion of the Project Report and setting up precise deadlines.

We also thank them for their immense support, guidance, specifications and ideas without which the Project Report would have been incomplete. Finally a note of thanks to the Department of Computer Science and Engineering, both teaching and non-teaching staff for their cooperation extended to us.

MOHAMMED SUHAIL R **(1OX20CS078)**
NAVYA TL **(1OX20CS084)**

ABSTRACT

Movie Ticket Booking System is a computerized solution for theatre setups, which will automate the process of Ticket Sale and Customer Bookings. This application will allow users to browse movies filtering them based on price, rating, genre, and timing. The home page will display login portal.When we login as a manager,we get the options to view bookings, insert movie,schedule shows and alter prices.First we insert a movie by giving it name,length,available languages,premier date, etc. Then we go to schedule shows and select the particulars for it and schedule a movie now you will be provided with the Show ID for reference.If needed the manager can also alter the price of the movie according to any criteria. When we login as a cashier,this is for the ones booking the movie tickets. Once you enter giving the login credentials you will be asked to pick a date, then it will show all the movies available for that day along with the time slots for it. . After picking a slot it will show the seat type and seat number.Select a seat number in your required seat type and press confirm booking. Your ticked will be booked and you will be provided with your ticket number. Through this project we aim at implementing various Database Management System concepts like transaction management and Database security while improving our software development skills.

TABLE OF CONTENTS

1.INTRODUCTION

These days there is increase in demand for the online bookings rather than going physically and waiting in queues for long time.People tend to opt for easy ways of booking in many fields such as movie tickets,flight tickets,bus tickets,etc.

Since movie ticket booking systems are used at such a large scale and have a lot of practical application as well as value, we picked this for our project. Our online movie ticket booking system, implemented in Python and MySQL allows users to easily make movie reservations. When we first visits the website it is redirected to a web page asking to login.Here if you are the one booking the tickets you have to login giving the cashier credentials and if you are the one scheduling movies you have to login giving the manager credentials.

Now when we login as a manager,we get the options to view bookings,insert movie,schedule shows and alter prices.First we insert a movie by giving it name,length,available languages,premier date, etc. Then we go to schedule shows and select the particulars for it and schedule a movie now you will be provided with the Show ID for reference.If needed the manager can also alter the price of the movie according to any criteria.The manager can also check the number of tickets booked by going to view bookings option. Here manager can find the number of tickets booked based on the dates selected. After picking a date it will show the ticked id with the seat type and seat number.Hence this will help for the manager to handle everything.

Now when we login as a cashier,this is for the ones booking the movie tickets. Once you enter giving the login credentials you will be asked to pick a date,then it will show all the movies available for that day along with the time slots for it. After picking a slot it will show the seat type and seat number.Select a seat number in your required seat type and press confirm booking. Your ticked will be booked and you will be provided with your ticket number. Hence this will be the procedure to book the tickets.

The users will be able to either book the tickets as cashier or schedule the movies as manager from our website. Which makes it a convenient way for ticket booking.

1.1 Preamble

Online movie ticket booking system keeps track of booking details and show details and other facilities. A database management system is necessary in order to reach and manage the data easily.it holds the vital information about status of the bookings and movie schedules. The customer can easily book the tickets from anywhere and manager can add the shows.

1.2 Problem Statement

A problem statement will outline the negative points of the correct situation and explain why this matters. It also serves as a great communication tool helping to get buy-in and support from others. One of the most important goals of any problem statement is to define the problem being addressed in a way that is clear and precise. Its aim is to focus the process improvement of the team's activities and steer the scope of the project.

i) The manger might sometimes feel difficult to keep a track on scheduling shows and keeping a count on tickets booked
ii) The cashier might be able to get the tickets physically or might not be able to check for available shows

1.3 Proposed Solution

The solution to the first problem is provided by organizing separate sections online for scheduling movies in different dates and different time slots.Also manager will be able to keep a track on number of tickets being booked

The solution to the second program is provided in same website through cashier login where the users can easily book their tickets accordingly.

2 ANALYSIS AND SYSTEM REQUIREMENTS

2.1 Existing System

In earlier days people used to book the tickets manually that is by standing in queues for a particular movie slot for hours together. Sometimes you might not even get the ticket even after waiting for so long hours. The crowd standing in queue would be very irritating and intolerant. Again the owner had to appoint a particular person to distribute the tickets and people to account the money all that would be very hectic.

So in order to reduce such efforts we have the easiest way of booking the tickets online through our website where both the customer and manager both are in advantage.
Our website not only reduces their waiting time also helps them to get the tickets in the convenient way.Also the manager will get the details of each and everything happening so that he need not be dependent on other person.

2.2 Hardware and Software Requirements

Hardware Requirements:

A minimum hard disk space of 20 Gigabytes(GB)..

RAM size of 1GB.

Intel core i3 processor and AMD processor.

Keyboard.

Mouse.

Software Requirements

Windows operating system such as Windows 2000, Windows XP, Windows Vista, Windows 7, Windows 10

Software: Xampp or Wamp

Front end: Python

Back end: MySQL

3 SYSTEM DESIGN AND MODELLING

Preliminary design

System design is an abstract representation of a system component and their relationship and which describe the aggregated functionally and performance of the system. It is also the overall plan or blueprint for how to obtain answer to the question being asked. The design specifies various type of approach.

Database design is one of the most important factors to keep in mind if you are concerned with application performance management. By designing your database to be efficient in each call it makes and to effectively create rows of data in the database, you can reduce the amount of CPU needed by the server to complete your request, thereby ensuring a faster application.

ER diagram

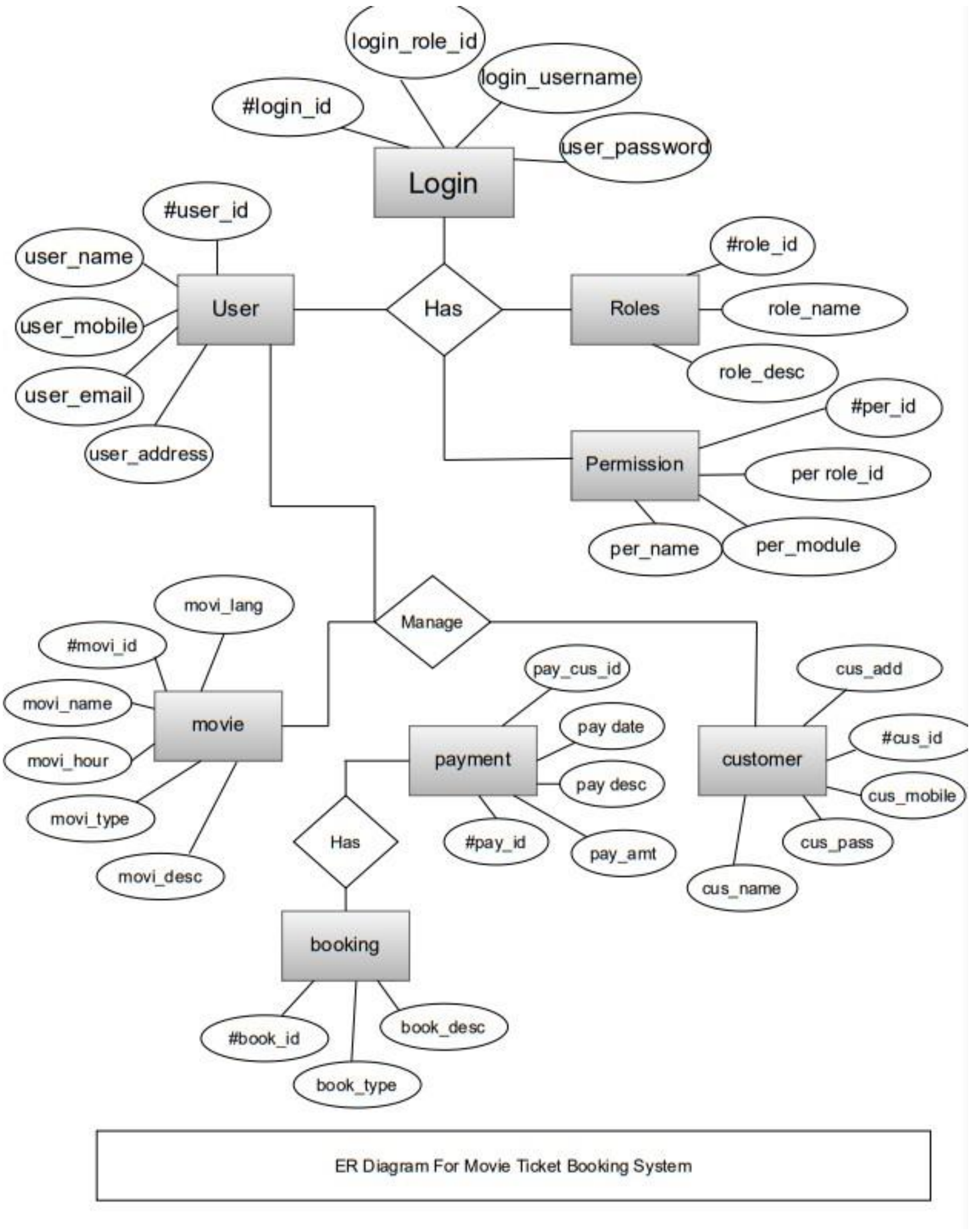
Entity-relationship diagram: This depicts relationship between data objects. The attribute of each data objects noted in the entity-relationship diagram can be described using a data object description.

Relationship: Data objects are connected to one another in a variety of different ways. We can define set of object relationship pairs that define the relevant relationships.

Cardinality ratio: The data model must be capable of representing the number of objects in a given relationship. The cardinality of an object relationship pair is I:N, 1:1, N:N, N:I.

Figure below describes the ER diagram of Movie Management System. It has 5 entities namely Student, Staff, Hostel, Examination and Finance. The entities have attributes which are primary, foreign and composite attributes. The primary attributes are underlined.

ER DIAGRAM



ER Diagram For Movie Ticket Booking System

Schema Diagram

Database schema is described as database connections and constraints. It contains attributes.

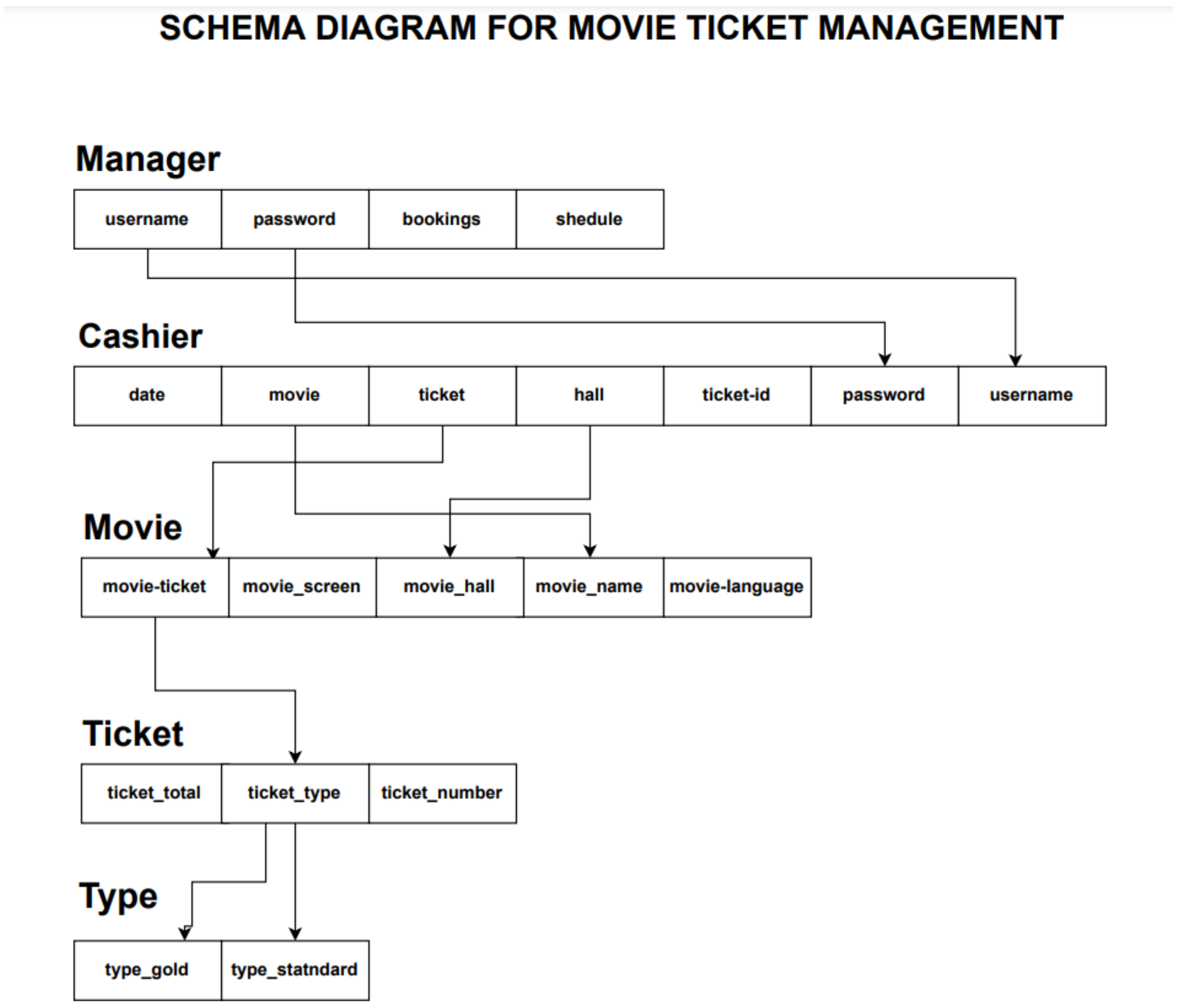
Every database has a state instances represent current set of database with values. There are different types of keys in a database schema.

A primary key is a table column that can be used to uniquely identify every row of the table. Any column that has this property, these columns are called candidate key. A composite primary key is a primary key consisting of more than one column. A foreign is a column or combination of columns that contains values that are found in the primary key of some table.

All the attributes of each table are interconnected by foreign key which is primary key in another column and composite key. Primary key cannot be null. The fact that many foreign key values repeat simply reflects the fact that its one-to-many relationship. In one-to-many Relationship the primary key has the one value and foreign key has many values.

The figure below is the schema diagram of Movie Ticket Booking System which has 5 tables Manager,Cashier,Movie,Ticket,Type along with some primary key or foreign key. The table Manager has 4 attributes username,password,booking and schedule. The table Cashier has 7 attributes date,movie,ticket,hall,ticket-id,password and username.The table Movie has 5 attributes movie-ticket,movie-screen,movie-hall,movie-name and movie-language.The table Ticket has 3 attributes ticket-total,ticket-type and ticket-number.The table Type has two attributes type-gold and type-standard

SCHEMA DIAGRAM



4 PROPOSED SYSTEM

Operations

Insert movie: this for the manger where he can add the movie name type,languages,duration,premier date,etc

Schedule Shows:here the shows are made available to the users by scheduling the date and time for the shows.

View Bookings:here the manager will be able to view all the tickets booked by the users along with their id and seat type.

Pick a date:this is available for the users where they can choose the date and go for further booking.

SQL Statements

Insert statement: The INSERT INTO statement is used to insert new records in a table. The INSERT INTO syntax would be as follows:

INSERT INTO table_name VALUES (value1, value2, value3,…);

The following SQL statement insert's a new record in the "movies" table:

Insert into movies values(194062403,'Uncharted',112,'English','2023-02-01','2023-02-15');

Update statement: An SQL UPDATE statement changes the data of one or more records in a table. Either all the rows can be updated, or a subset may be chosen using a condition.

The UPDATE syntax would be as follows: UPDATE table_name SET column_name value [column_name-value…] [WHERE condition).

The following updates a record in the "halls" table:

UPDATE halls SET no_of_seats=100 where class = 'standard';

Create statement:The CREATE TABLE Statement is used to create tables to movie ticket booking data. Integrity Constraints like primary key, unique key, foreign key can be defined for the columns while creating the table.

The CREATE syntax would be as follows:

CREATETABLE table_name(column1 datatype,column2 datatype,column3 Datatype…..columnN datatype,PRIMARY KEY(one or more columns));

The following SQL statement creates a table "halls":

```
Create table halls (`hall_id` int(11) NOT NULL,`class` varchar(10) NOT
NULL,`no_of_seats` int(11) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

Trigger statement: A trigger is a special kind of stored procedure that automatically executes when an event occurs in the database server. The trigger syntax would be as follows:

Create trigger trigger_name before insert or update of <parameters> on <table_name> for each row set <condition>;

Ex:

```
CREATE TRIGGER `get_price` AFTER INSERT ON `halls` FOR EACH ROW
```

Delimiter

```
UPDATE shows s, price_listing p
SET s.price_id=p.price_id
WHERE p.price_id IN
(SELECT price_id
FROM price_listing p
WHERE dayname(s.Date)=p.day AND s.type=p.type);

end
$$DELIMITER ;
```

IMPLEMENTATION

SQL CODE

```
-- phpMyAdmin SQL Dump
-- version 5.1.1
--  https://www.phpmyadmin.net/
--
-- Host: 127.0.0.1
-- Generation Time: Mar 10, 2022 at 02:17 PM
-- Server version: 10.4.22-MariaDB
-- PHP Version: 8.1.2

SET SQL_MODE =  "NO_AUTO_VALUE_ON_ZERO";
START TRANSACTION;
SET time_zone = "+00:00";


/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET
@OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS  */;
/*!40101   SET   @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION
*/;
/*!40101 SET NAMES utf8mb4 */;

--
-- Database: `dbtheatre`
DELIMITER $$
--
-- Procedures
--
CREATE  DEFINER=`root`@`localhost`  PROCEDURE  `delete_old`  ()   begin

        declare curdate date;
set curdate=curdate();

DELETE FROM shows
WHERE  datediff(Date,curdate)<0;

DELETE FROM shows
WHERE movie_id IN
(SELECT movie_id
FROM movies
WHERE  datediff(show_end,curdate)<0);

DELETE FROM movies
WHERE datediff(show_end,curdate)<0;

end$$
```

```
DELIMITER ;
-- Table structure for table `booked_tickets`
--

CREATE TABLE `booked_tickets` (
  `ticket_no` int(11) NOT NULL,
  `show_id` int(11) NOT NULL,
  `seat_no` int(11) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
-- Table structure for table `halls`
--

CREATE TABLE `halls` (
  `hall_id` int(11) NOT NULL,
  `class` varchar(10) NOT NULL,
  `no_of_seats` int(11) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
-- Dumping data for table `halls`
--

INSERT INTO `halls` (`hall_id`, `class`, `no_of_seats`) VALUES
(1, 'gold', 35),
(1, 'standard', 75),
(2, 'gold', 27),
(2, 'standard', 97),
(3, 'gold', 26),
(3, 'standard', 98);
-- Triggers `halls`
--
DELIMITER $$
CREATE TRIGGER `get_price` AFTER INSERT ON `halls` FOR EACH ROW begin

UPDATE shows s, price_listing p
SET s.price_id=p.price_id
WHERE p.price_id IN
(SELECT price_id
FROM price_listing p
WHERE dayname(s.Date)=p.day AND s.type=p.type);

end
$$
DELIMITER ;

-- Table structure for table `movies`
--

CREATE TABLE `movies` (
  `movie_id` int(11) NOT NULL,
  `movie_name` varchar(40) DEFAULT NULL,
  `length` int(11) DEFAULT NULL,
  `language` varchar(10) DEFAULT NULL,
  `show_start` date DEFAULT NULL,
  `show_end` date DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
-- Dumping data for table `movies`

INSERT INTO `movies` (`movie_id`, `movie_name`, `length`, `language`, `show_start`,
`show_end`) VALUES
(60146899, 'Demo Movie Name', 125, 'English', '2022-03-08', '2022-05-19'),
(194062403, 'Uncharted', 112, 'English', '2022-02-16', '2022-04-26'),
(476621797, 'The Batman', 176, 'English', '2022-03-03', '2022-05-17');
-- Table structure for table `price_listing`

CREATE TABLE `price_listing` (
  `price_id` int(11) NOT NULL,
  `type` varchar(3) DEFAULT NULL,
  `day` varchar(10) DEFAULT NULL,
  `price` int(11) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
-- Dumping data for table `price_listing`

INSERT INTO `price_listing` (`price_id`, `type`, `day`, `price`) VALUES
(1, '2D', 'Monday', 10),
(2, '3D', 'Monday', 12),
(3, '4DX', 'Monday', 15),
(4, '2D', 'Tuesday', 10),
(5, '3D', 'Tuesday', 12),
(6, '4DX', 'Tuesday', 14),
(7, '2D', 'Wednesday', 8),
(8, '3D', 'Wednesday', 10),
(9, '4DX', 'Wednesday', 12),
(10, '2D', 'Thursday', 10),
(11, '3D', 'Thursday', 12),
(12, '4DX', 'Thursday', 15),
(13, '2D', 'Friday', 12),
(14, '3D', 'Friday', 15),
(15, '4DX', 'Friday', 22),
(16, '2D', 'Saturday', 12),
(17, '3D', 'Saturday', 15),
(18, '4DX', 'Saturday', 20),
(19, '2D', 'Sunday', 10),
(20, '3D', 'Sunday', 13),
(21, '4DX', 'Sunday', 17);
-- Table structure for table `shows`
--

CREATE TABLE `shows` (
  `show_id` int(11) NOT NULL,
  `movie_id` int(11) DEFAULT NULL,
  `hall_id` int(11) DEFAULT NULL,
  `type` varchar(3) DEFAULT NULL,
  `time` int(11) DEFAULT NULL,
  `Date` date DEFAULT NULL,
  `price_id` int(11) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

-- Dumping data for table `shows`

INSERT INTO `shows` (`show_id`, `movie_id`, `hall_id`, `type`, `time`, `Date`, `price_id`)
VALUES
(288841196, 476621797, 3, '2D', 1900, '2022-03-10', 10),
(859902520, 194062403, 3, '4DX', 1945, '2022-03-10', 12),
(1095272239, 194062403, 2, '2D', 1900, '2022-03-10', 10),
(2001281728, 476621797, 2, '2D', 900, '2022-03-10', 10),
(2054684557, 60146899, 1, '4DX', 1900, '2022-03-10', 12);

-- Table structure for table `types`

CREATE TABLE `types` (
  `movie_id` int(11) NOT NULL,
  `type1` varchar(3) DEFAULT NULL,
  `type2` varchar(3) DEFAULT NULL,
  `type3` varchar(3) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

-- Dumping data for table `types`

INSERT INTO `types` (`movie_id`, `type1`, `type2`, `type3`) VALUES
(60146899, '4DX', 'NUL', 'NUL'),
(194062403, '2D', '3D', '4DX'),
(476621797, '2D', '3D', '4DX');
-- Indexes for dumped tables
-- Indexes for table `booked_tickets`
--
ALTER TABLE `booked_tickets`
  ADD PRIMARY KEY (`ticket_no`,`show_id`),
  ADD KEY `show_id` (`show_id`);

-- Indexes for table `halls`
--
ALTER TABLE `halls`
  ADD PRIMARY KEY (`hall_id`,`class`);

--
-- Indexes for table `movies`
--
ALTER TABLE `movies`
  ADD PRIMARY KEY (`movie_id`);
-- Indexes for table `price_listing`
```

```
--
ALTER TABLE `price_listing`
  ADD PRIMARY KEY (`price_id`);

--
-- Indexes for table `shows`
--
ALTER TABLE `shows`
  ADD PRIMARY KEY (`show_id`),
  ADD KEY `movie_id` (`movie_id`),
  ADD KEY `hall_id` (`hall_id`),
  ADD KEY `price_id` (`price_id`);

--
-- Indexes for table `types`
--
ALTER TABLE `types`
  ADD PRIMARY KEY (`movie_id`);

--
-- Constraints for dumped tables

-- Constraints for table `booked_tickets`
--
ALTER TABLE `booked_tickets`
  ADD CONSTRAINT `booked_tickets_ibfk_1` FOREIGN KEY (`show_id`) REFERENCES
`shows` (`show_id`) ON DELETE CASCADE;

--
-- Constraints for table `shows`
--
ALTER TABLE `shows`
  ADD CONSTRAINT `shows_ibfk_1` FOREIGN KEY (`movie_id`) REFERENCES
`movies` (`movie_id`),
  ADD CONSTRAINT `shows_ibfk_2` FOREIGN KEY (`hall_id`) REFERENCES `halls`
(`hall_id`),
  ADD CONSTRAINT `shows_ibfk_3` FOREIGN KEY (`price_id`) REFERENCES
`price_listing` (`price_id`) ON UPDATE CASCADE;

-- Constraints for table `types`
--
ALTER TABLE `types`
  ADD CONSTRAINT `types_ibfk_1` FOREIGN KEY (`movie_id`) REFERENCES
`movies` (`movie_id`) ON DELETE CASCADE;
COMMIT;

/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
```

PYTHON CODE

```
import mysql.connector,sys
import datetime
from mysql.connector import Error
from flask import Flask, request, jsonify, render_template
from random import randint

app = Flask(__name__)

@app.route('/')
def renderLoginPage():
   return render_template('login.html')

@app.route('/login', methods = ['POST'])
def verifyAndRenderRespective():
        username = request.form['username']
        password = request.form['password']

        try:
                if username == 'cashier' and password == 'cashier123':

                        res = runQuery('call delete_old()')
                        return render_template('cashier.html')

                elif username == 'manager' and password == 'Password@123':

                        res = runQuery('call delete_old()')
                        return render_template('manager.html')

                else:
                        return render_template('loginfail.html')
        except Exception as e:
                print(e)
                return render_template('loginfail.html')
# Routes for cashier
@app.route('/getMoviesShowingOnDate', methods = ['POST'])
def moviesOnDate():
        date = request.form['date']

        res = runQuery("SELECT DISTINCT movie_id,movie_name,type FROM movies
NATURAL JOIN shows WHERE Date = '"+date+"'")

        if res == []:
                return '<h4>No Movies Showing</h4>'
        else:
                return render_template('movies.html',movies = res)
@app.route('/getTimings', methods = ['POST'])
def timingsForMovie():
	date = request.form['date']
	movieID = request.form['movieID']
	movieType = request.form['type']

	res = runQuery("SELECT time FROM shows WHERE Date='"+date+"' and movie_id
= "+movieID+" and type ='"+movieType+"'")

	list = []

	for i in res:
		list.append( (i[0], int(i[0]/100), i[0]%100 if i[0]%100 != 0 else '00' ) )

	return render_template('timings.html',timings = list)
@app.route('/getShowID', methods = ['POST'])
def getShowID():
	date = request.form['date']
	movieID = request.form['movieID']
	movieType = request.form['type']
	time = request.form['time']

	res = runQuery("SELECT show_id FROM shows WHERE Date='"+date+"' and
movie_id = "+movieID+" and type ='"+movieType+"' and time = "+time)
	return jsonify({"showID" : res[0][0]})
@app.route('/getAvailableSeats', methods = ['POST'])
def getSeating():
	showID = request.form['showID']

	res = runQuery("SELECT class,no_of_seats FROM shows NATURAL JOIN halls
WHERE show_id = "+showID)

	totalGold = 0
	totalStandard = 0

	for i in res:
		if i[0] == 'gold':
			totalGold = i[1]
		if i[0] == 'standard':
			totalStandard = i[1]

	res = runQuery("SELECT seat_no FROM booked_tickets WHERE show_id =
"+showID)
	goldSeats = []
	standardSeats = []

	for i in range(1, totalGold + 1):
		goldSeats.append([i,''])

	for i in range(1, totalStandard + 1):
		standardSeats.append([i,''])
	for i in res:
		if i[0] > 1000:
			goldSeats[ i[0] % 1000 - 1 ][1] = 'disabled'
		else:
			standardSeats[ i[0] - 1 ][1] = 'disabled'
	return render_template('seating.html', goldSeats = goldSeats, standardSeats = 
standardSeats)
@app.route('/getPrice', methods = ['POST'])
def getPriceForClass():
	showID = request.form['showID']
	seatClass = request.form['seatClass']
	res = runQuery("INSERT INTO halls VALUES(-1,'-1',-1)");

	res = runQuery("DELETE FROM halls WHERE hall_id = -1")

	res = runQuery("SELECT price FROM shows NATURAL JOIN price_listing 
WHERE show_id = "+showID)
	if res == []:
		return '<h5>Prices Have Not Been Assigned To This Show, Please Try Again 
Later!</h5>'
	price = int(res[0][0])
	if seatClass == 'gold':
		price = price * 1.5
	return '<h5>Ticket Price: $ '+str(price)+'</h5>\
	<button onclick="confirmBooking()" class="btn-warning">Confirm 
Booking</button>'
@app.route('/insertBooking', methods = ['POST'])
def createBooking():
	showID = request.form['showID']
	seatNo = request.form['seatNo']
	seatClass = request.form['seatClass']

	if seatClass == 'gold':
		seatNo = int(seatNo) + 1000
	ticketNo = 0
	res = None
	while res != []:
		ticketNo = randint(0, 2147483646)
		res = runQuery("SELECT ticket_no FROM booked_tickets WHERE ticket_no 
= "+str(ticketNo))

	res = runQuery("INSERT INTO booked_tickets 
VALUES("+str(ticketNo)+","+showID+","+str(seatNo)+")")

	if res == []:
		return '<h5>Ticket Has Been Booked Successfully!</h5>\
		<h6>Ticket Number: '+str(ticketNo)+'</h6>'
# Routes for manager
@app.route('/getShowsShowingOnDate', methods = ['POST'])
def getShowsOnDate():
    date = request.form['date']

    res = runQuery("SELECT show_id,movie_name,type,time FROM shows NATURAL
JOIN movies WHERE Date = '"+date+"'")
    if res == []:
        return '<h4>No Shows Showing</h4>'
    else:
        shows = []
        for i in res:
            x = i[3] % 100
            if i[3] % 100 == 0:
                x = '00'
            shows.append([ i[0], i[1], i[2], int(i[3] / 100), x ])

        return render_template('shows.html', shows = shows)
@app.route('/getBookedWithShowID', methods = ['POST'])
def getBookedTickets():
    showID = request.form['showID']
    res = runQuery("SELECT ticket_no,seat_no FROM booked_tickets WHERE show_id
= "+showID+" order by seat_no")
    if res == []:
        return '<h5>No Bookings!!</h5>'
    tickets = []
    for i in res:
        if i[1] > 1000:
            tickets.append([i[0], i[1] - 1000, 'Gold'])
    else:
            tickets.append([i[0], i[1], 'Standard'])
    return render_template('bookedtickets.html', tickets = tickets)
@app.route('/fetchMovieInsertForm', methods = ['GET'])
def getMovieForm():
    return render_template('movieform.html')
@app.route('/insertMovie', methods = ['POST'])
def insertMovie():
    movieName = request.form['movieName']
    movieLen = request.form['movieLen']
    movieLang = request.form['movieLang']
    types = request.form['types']
    startShowing = request.form['startShowing']
    endShowing = request.form['endShowing']
    res = runQuery('SELECT * FROM movies')

    for i in res:
        if i[1] == movieName and i[2] == int(movieLen) and i[3] == movieLang \
         and i[4].strftime('%Y/%m/%d') == startShowing and
i[5].strftime('%Y/%m/%d') == endShowing:
            return '<h5>The Same Movie Already Exists</h5>'

    movieID = 0
    res = None
```

```
        res = runQuery("UPDATE price_listing SET price = "+str(newPrice)+" WHERE
price_id = "+str(priceID))

        if res == []:
                return '<h5>Price Updated Successfully</h5>\
                        <h6>Standard: $ '+newPrice+'</h6>\
                        <h6>Gold: $ '+str( int(int(newPrice) * 1.5) )+'</h6>'

        else:
                print(res)
        return '<h5>Something Went Wrong!!</h5>'


def runQuery(query):
        try:
                db = mysql.connector.connect(
                        host='localhost',
                        database='dbtheatre',
                        user='root',
                        password='')

                if db.is_connected():
                        print("Connected to MySQL, running query: ", query)
                        cursor = db.cursor(buffered = True)
                        cursor.execute(query)
                        db.commit()
                        res = None
                        try:
                                res = cursor.fetchall()
                        except Exception as e:
                                print("Query returned nothing, ", e)
                                return []
                        return res

        except Exception as e:
                print(e)
                return e

        finally:
                db.close()

        print("Couldn't connect to MySQL Database")
   #Couldn't connect to MySQL
        return None

if __name__ == "__main__":
   app.run(host='0.0.0.0')
```

TESTING

This gives the outline of all the testing methods that are carried out to get a bug free application. Quality can be achieved by testing the product using different techniques at different phases of the project development.

Testing process

Testing is an integral part of software development. Testing process, in a way certifies, whether the product, that is developed, compiles with the standards, that it was designed to. Testing process involves building of test cases, against which, the product has to be tested. In some cases, test cases are done based on the system requirements specified for the product/software, which is to be developed.

Testing Objective

The main objectives of testing process are as follows:

- Testing is a process of executing a program with the intent of finding an error.
- A good test case is one that has high probability of finding an as yet undiscovered error.
- A successful test is one that uncovers an as yet undiscovered error.

Levels of Testing

Different levels of testing are used in the testing process, each level of testing aims to test different aspects of the system. The basic levels are unit testing, integration testing,system testing and acceptance testing.

Unit Testing

Unit testing focuses verification effort on the smallest unit of software design the module. The software built, is a collection of individual modules.

In this kind of testing exact flow of control for each module was verified. With detailed design consideration used as a guide, important control paths are tested to uncover errors within the boundary of the module.

Negative test case for adding movies:

Function name	Input	Expected output	Error	Resolved
Create movie	Bahubali 2	Must take only Bahubali as input	Number is being taken as input for name	Consume()

Positive test case for adding movies:

Function name	Input	Expected output	Error	Resolved
Create movie	Bahubali	Expected output is seen	-	-

Integration Testing

The second level of testing is called integration testing. In this,many class tested modules are combined into subsystems, which are then tested. The goal here is to see if all the modules can be integrated properly. Error we have are identified and debugged.

Test case on the basis of shows available

Function name	Input	Expected output	Error	Resolved
Negative- Checking for movie using movie name	Kantara	Must display that movie name doesn't exist	Output not seen	Consume()
Positive- Checking for movie using movie name	Wednesday	Must display that name exists	-	-

System Testing

Here the entire application is tested. The reference document for this process is the requirement document, and the goal is to see IF the application meets its requirements. Each module and component of ethereal was thoroughly tested to remove bugs through a system testing strategy. Test cases were generated for all possible input sequences and the output was verified for its correctness.

Test cases for the project

Steps	Action	Expected Output
Step 1:	The screen appears hen the Users runs the program	A page with different tabbed panes appears.
Choice	i) If manager log in ii)If cashier log in	Manager panel opens Cashier panel opens

Step 2: **Selection**	The screen appears when the cashier logs in and selects any one of the show time. Movies showing	A window for selecting the seat appears
Step 3 **Selection**	The screen appears when the manager logs in and select any one of the tabbed panes from the click of the mouse. ● View Bookings ● Insert Movie ● Schedule Shows ● Alter Prices	A window for inserting the movie,scheduling time slots,adding duration,altering ticket price appears.

CONCLUSION

The project entitled as Movie Ticketing System is the system that deals with the issues related to a particular institution.

- The project is successfully implemented with all the features mentioned in the system requirements specification.
- The application provides appropriate information to users according to the chosen service.
- The project is designed keeping in view the day to day problems faced by a college.
- Deployment of our application will certainly help the college to reduce unnecessary wastage of time in personally going to each department for some information.

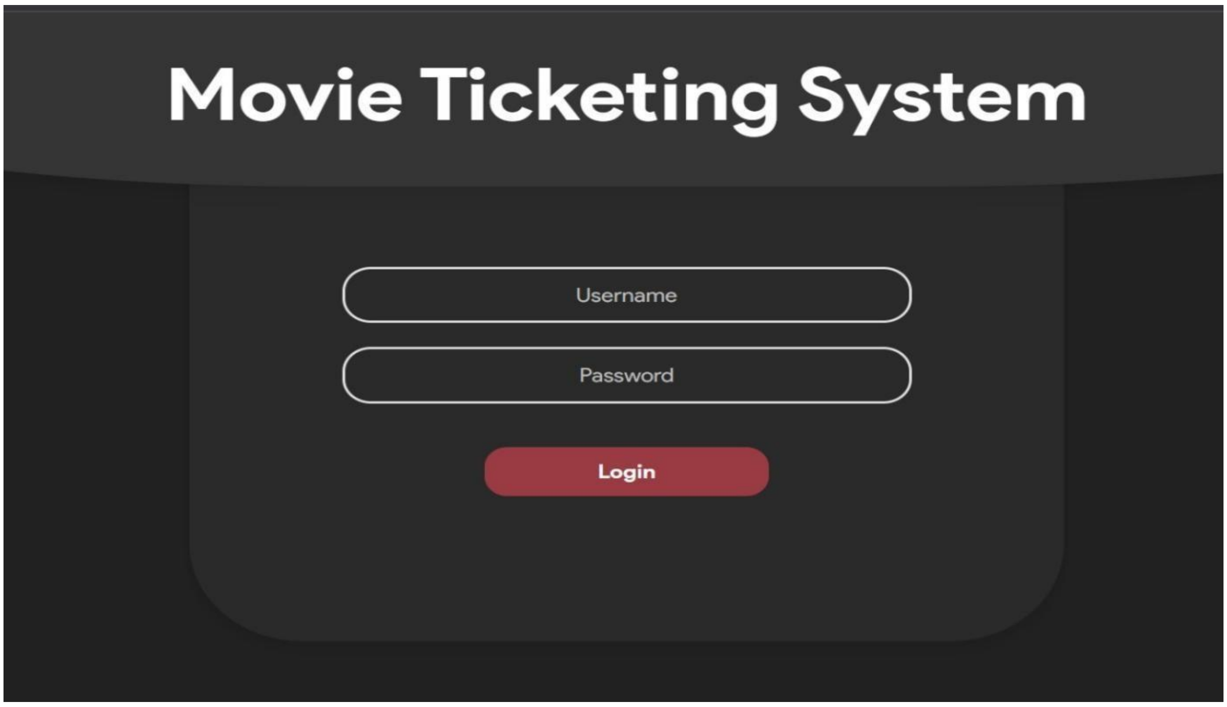
Awareness and right information about any college is essential for both the development of student as well as staff. So this serves the right purpose in achieving the desired requirements of both the communities.

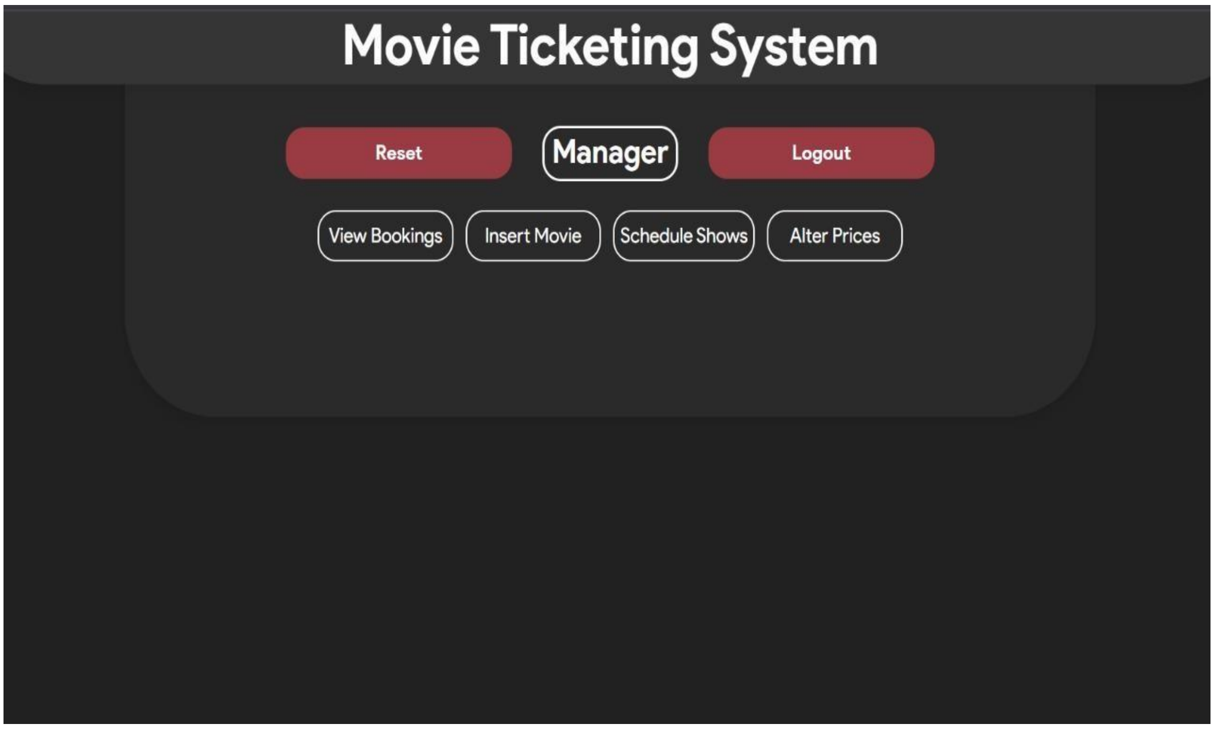
REFERENCES

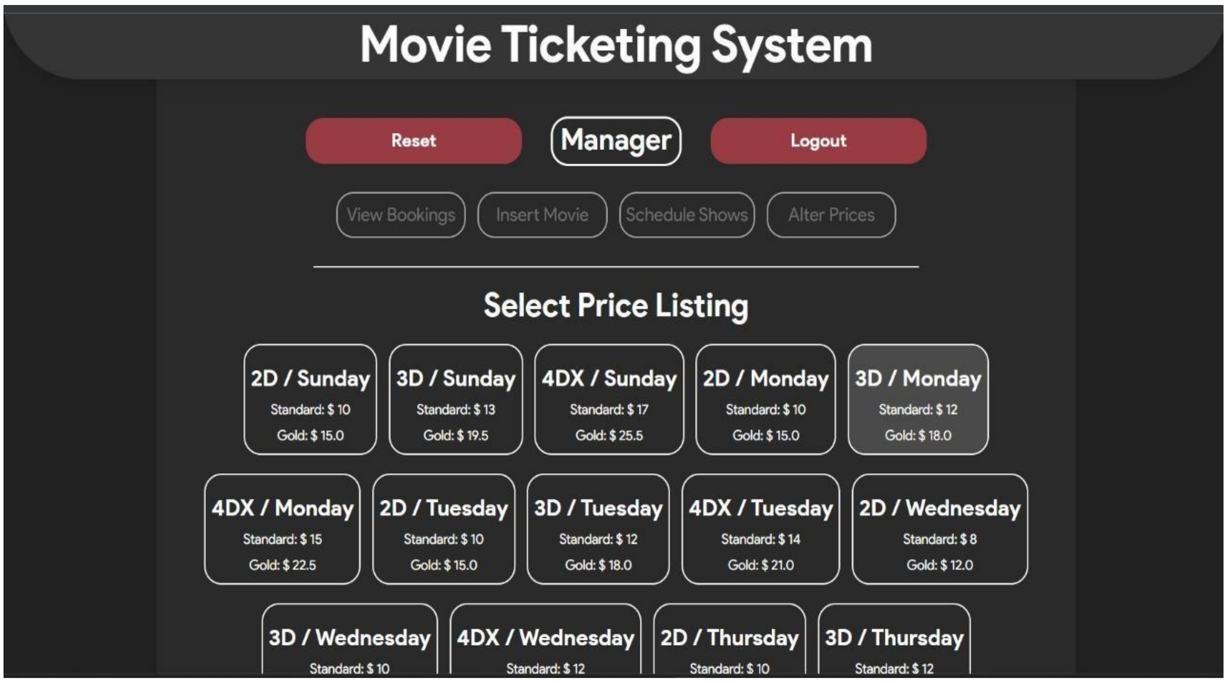
- www.itsourcecode.com
- www.wikipedia.com
- www.google.com
- www.youtube.com
- www.projectsworld.in
- Fundamentals of Database Systems,s Ramez Elmasri and Shamkant B. Navathe
- Database management systems, Ramakrishnan, and Gehrke

APPENDIX

Snapshots







Movie Ticketing System

Reset | Manager | Logout

View Bookings | Insert Movie | Schedule Shows | Alter Prices

Select Price Listing

2D / Sunday
Standard: $ 10
Gold: $ 15.0

3D / Sunday
Standard: $ 13
Gold: $ 19.5

4DX / Sunday
Standard: $ 17
Gold: $ 25.5

2D / Monday
Standard: $ 10
Gold: $ 15.0

3D / Monday
Standard: $ 12
Gold: $ 18.0

4DX / Monday
Standard: $ 15
Gold: $ 22.5

2D / Tuesday
Standard: $ 10
Gold: $ 15.0

3D / Tuesday
Standard: $ 12
Gold: $ 18.0

4DX / Tuesday
Standard: $ 14
Gold: $ 21.0

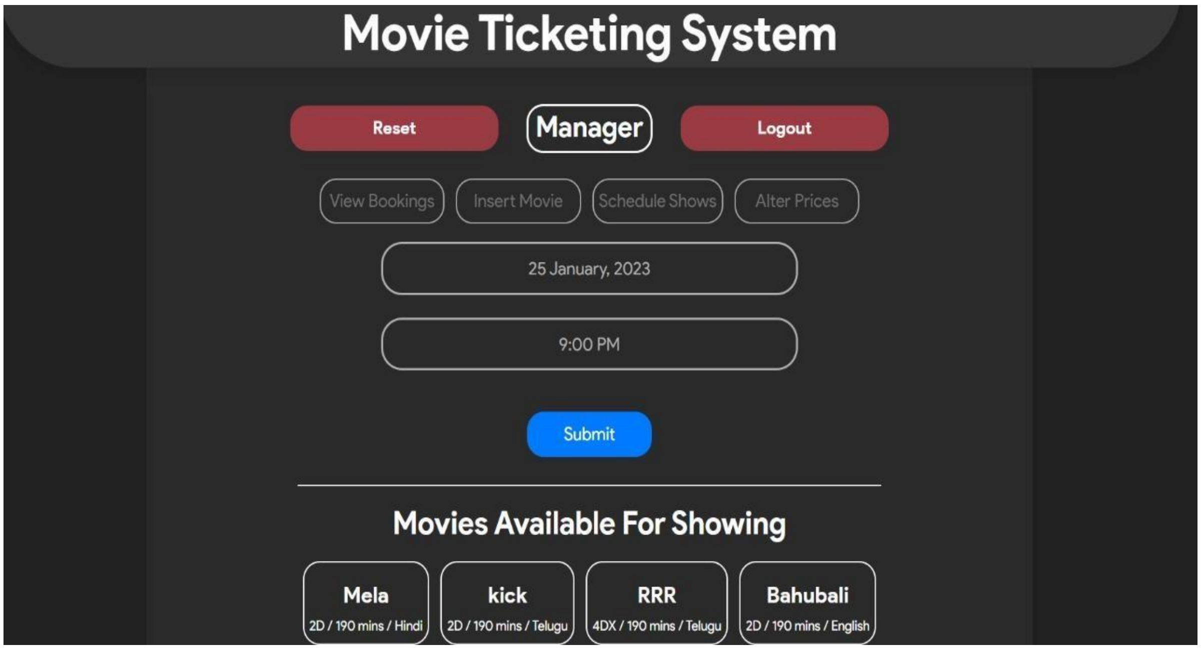
2D / Wednesday
Standard: $ 8
Gold: $ 12.0

3D / Wednesday
Standard: $ 10

4DX / Wednesday
Standard: $ 12

2D / Thursday
Standard: $ 10

3D / Thursday
Standard: $ 12



Movie Ticketing System

Reset | Manager | Logout

View Bookings | Insert Movie | Schedule Shows | Alter Prices

25 January, 2023

9:00 PM

Submit

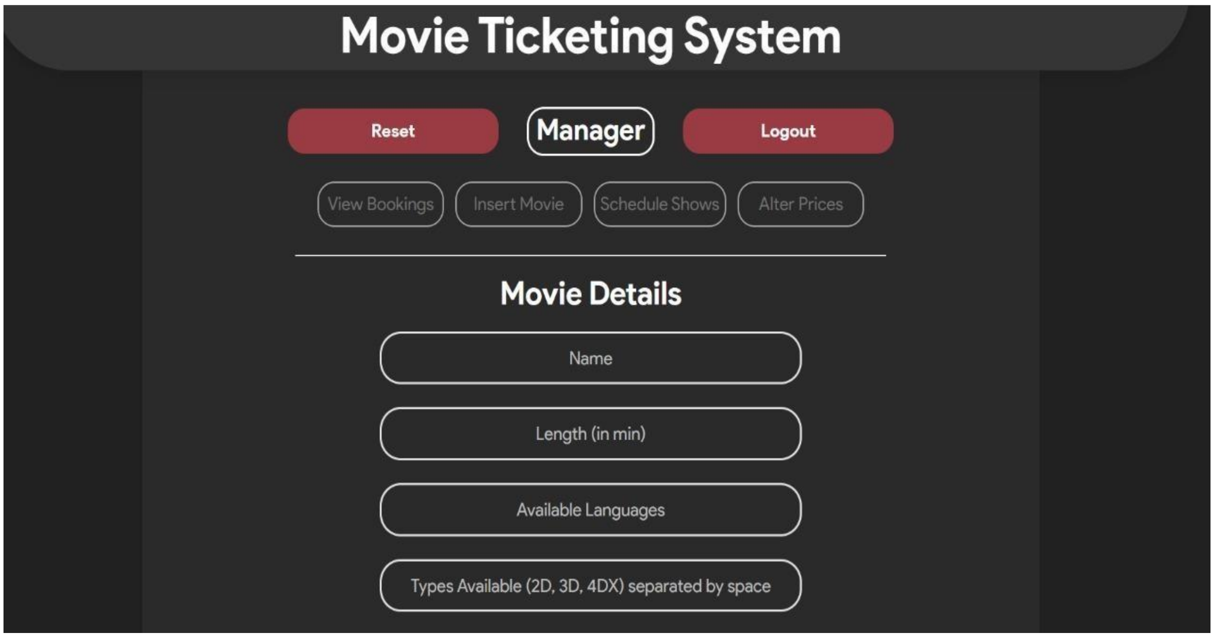
Movies Available For Showing

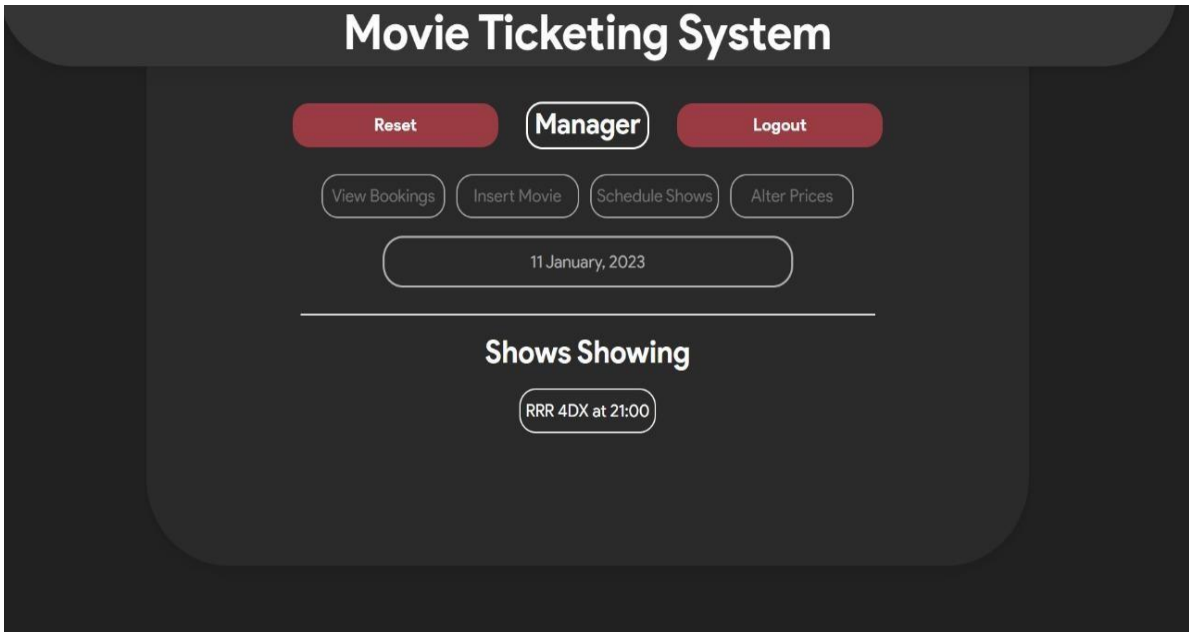
Mela
2D / 190 mins / Hindi

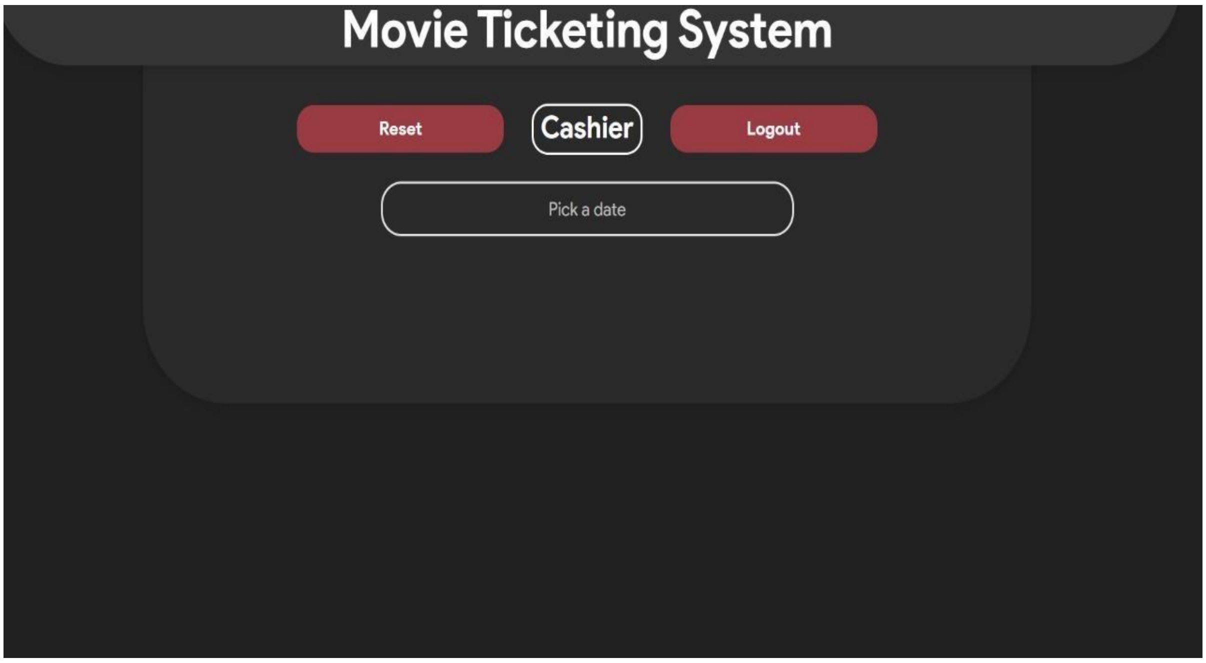
kick
2D / 190 mins / Telugu

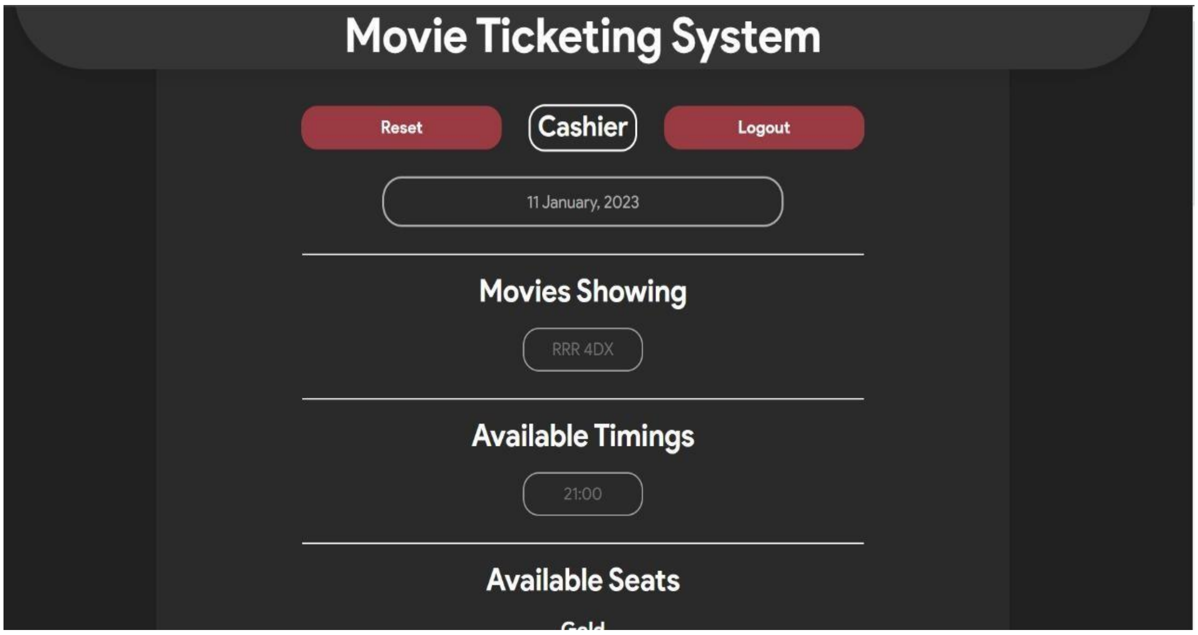
RRR
4DX / 190 mins / Telugu

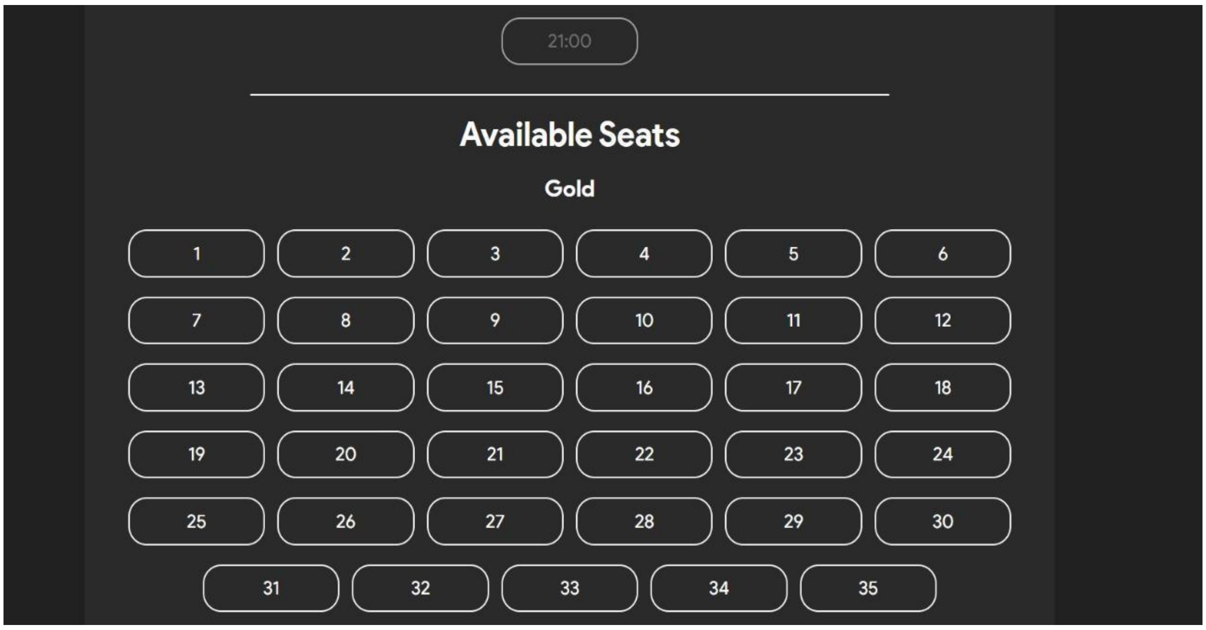
Bahubali
2D / 190 mins / English

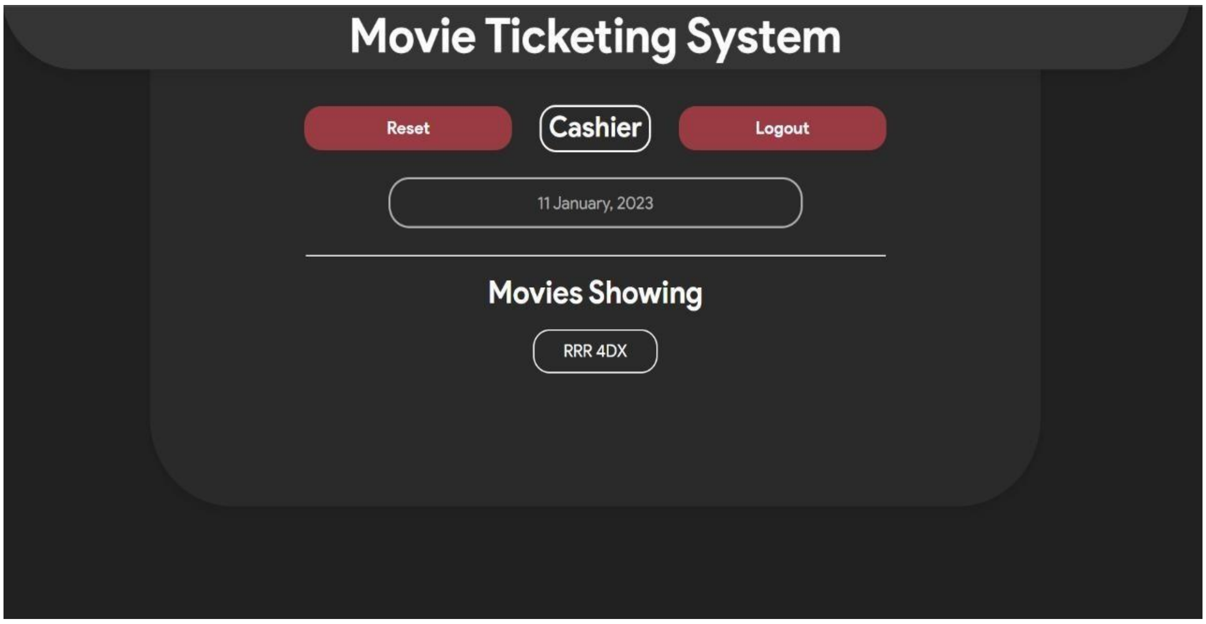

Movie Ticketing System
Reset
Manager
Logout
View Bookings
Insert Movie
Schedule Shows
Alter Prices
Movie Details
Name
Length (in min)
Available Languages
Types Available (2D, 3D, 4DX) separated by space


Movie Ticketing System
Reset
Manager
Logout
View Bookings
Insert Movie
Schedule Shows
Alter Prices
11 January, 2023
Shows Showing
RRR 4DX at 21:00


Movie Ticketing System
Reset
Cashier
Logout
Pick a date


Movie Ticketing System
Reset
Cashier
Logout
11 January, 2023
Movies Showing
RRR 4DX
Available Timings
21:00
Available Seats
Gold


21:00
Available Seats
Gold
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35


Movie Ticketing System
Reset
Cashier
Logout
11 January, 2023
Movies Showing
RRR 4DX