

MOVIE TICKET BOOKING SYSTEM

A PROJECT COMPONENT REPORT

Submitted by

VIGNESH. T (Reg. No. 201904168)

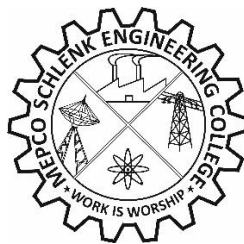
MANOJ KUMAR. D (Reg. No. 201904087)

*for the Theory Cum Project Component
of*

19CS692 – C# AND .NET

during

VI Semester – 2021 – 2022



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

MEPCO SCHLENK ENGINEERING COLLEGE, SIVAKASI

(An Autonomous Institution affiliated to Anna University Chennai)

April 2022

MEPCO SCHLENK ENGINEERING COLLEGE, SIVAKASI

(An Autonomous Institution affiliated to Anna University Chennai)

Department of Computer Science and Engineering

BONAFIDE CERTIFICATE

Certified that this project component report titled **MOVIE TICKET BOOKING SYSTEM** is the bonafide work of **T.VIGNESH (Reg. No. 201904168)**, and **D.MANOJ KUMAR (Reg. No. 201904087)** who carried out this work under my guidance for the Theory cum Project Component course “**19CS692 – C# AND .NET**” during the sixth semester.

Dr. M. S. BHUVANESWARI, M.E.,Ph.D.
Assistant Professor (Sl. Grade)
Course Instructor
Department of Computer Science & Engg.
Mepco Schlenk Engineering College
Sivakasi.

Dr. J. RAJA SEKAR, M.E.,Ph.D.
Professor
Head of the Department
Department of Computer Science &
Engg. Mepco Schlenk Engineering
College
Sivakasi.

Submitted for viva-Voce Examination held at **MEPCO SCHLENK ENGINEERING COLLEGE (Autonomous), SIVAKASI** on/...../.... **20.....**

Internal Examiner

External Examiner

ABSTRACT

Booking ticket in theatre by waiting in queue wastes a lot of time. During this pandemic period, it is necessary to maintain social distance. So, the application we developed helps customers book the movie tickets online. The customer signs up to the account by entering the customer's name, mobile number, age, maid-id and password if he/she is new user or he/she can login to the account using mobile number and password. Homepage displays the movies that are currently screening in the theatre. The customer can view the details about the movie like movie name, language, director, cast, CBFC certificate etc. The customer can also view or check the availability of seats, price of ticket and can book the tickets based on availability of seats. Once the tickets have been booked, the user can view the booked tickets and also cancel tickets. The admin maintains the system by updating customer details, movie and deleting the tickets which was canceled by customer.

ACKNOWLEDGEMENT

First and foremost, we thank the **LORD ALMIGHTY** for his abundant blessings that is showered upon our past, present and future successful endeavors.

We extend our sincere gratitude to our college management and Principal **Dr. S. Arivazhagan M.E., Ph.D.**, for providing sufficient working environment such as systems and library facilities. We also thank him very much for providing us with adequate lab facilities, which enable us to complete our project.

We would like to extend our heartfelt gratitude to **Dr. J. Raja Sekar M.E., Ph.D.**, Professor and Head, Department of Computer Science and Engineering, Mepco Schlenk Engineering College for giving us the golden opportunity to undertake a project of this nature and for his most valuable guidance given at every phase of our work.

We would also like to extend our gratitude and sincere thanks to **Dr. M.S. Bhuvaneswari M.E., Ph.D.**, Assistant Professor (Sl. Grade), Department of Computer Science and Engineering, Mepco Schlenk Engineering College for being our Project Mentor. She has put her valuable experience and expertise in directing, suggesting and supporting us throughout the Project to bring out the best.

Our sincere thanks to our revered **faculty members and lab technicians** for their help over this project work.

Last but not least, we extend our indebtedness towards our beloved family and our friends for their support which made the project a successful one.

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	ABSTRACT	ii
	LIST OF TABLES	v
	LIST OF FIGURES	vii
	LIST OF ABBREVIATIONS	ix
1	INTRODUCTION	1
2	REQUIREMENTS DESCRIPTION	2
	2.1 Functional Requirements	2
	2.2 Non-Functional Requirements	2
3	SYSTEM DESIGN	3
	3.1 ER Diagram for movie ticket booking system	3
	3.2 Design Components	5
	3.3 Database Description	5
	3.4 Low Level design	8
	3.5 User Interface design	12
4	SYSTEM IMPLEMENTATION	20
5	RESULTS AND DISCUSSION	24
6	CONCLUSION AND FUTURE ENHANCEMENT(S)	37
APPENDIX –A	SYSTEM REQUIREMENTS	38
APPENDIX –B	SOURCE CODE	39
	REFERENCES	53

LIST OF TABLES

Table No.	Table Caption	Page No.
3.1	Customer Structure	5
3.2	Movie Structure	6
3.3	Theatre Screens Structure	6
3.4	Show Structure	7
3.5	Ticket Structure	7
3.6	Login Details	8
3.7	Register user	8
3.8	Book Tickets	9
3.9	View booked Tickets	9
3.10	Change Password	9
3.11	Delete User	10
3.12	Update User	10
3.13	Update Movies	10
3.14	Update Show	11
3.15	Delete Ticket	11
5.1	Positive Test Case and result for Login	24
5.2	Negative Test Case and result for Login	24
5.3	Positive Test Case and result for Register user	25
5.4	Negative Test Case and result for Register user	25
5.5	Positive Test Case and result for Booking Tickets	26
5.6	Negative Test Case and result for Booking Tickets	26
5.7	Positive Test Case and result for Cancelling Tickets	27
5.8	Negative Test Case and result for Cancelling Tickets	28
5.9	Positive Test Case and result for Changing the Password	28
5.10	Negative Test Case and result for Changing the Password	29

5.11	Positive Test Case and result for Admin deleting the Customer	30
5.12	Negative Test Case and result for Admin deleting the Customer	30
5.13	Positive Test Case and result for Admin Updating the movie	31
5.14	Negative Test Case and result for Admin Updating the movie	31
5.15	Positive Test Case and result for Admin Updating the show	32
5.16	Negative Test Case and result for Admin Updating the show	33
5.17	Positive Test Case and result for Admin Updating the ticket	33
5.18	Negative Test Case and result for Admin Updating the ticket	34
5.19	Positive Test Case and result for Admin Updating customer details	35
5.20	Negative Test Case and result for Admin Updating customer details	35

LIST OF FIGURES

Figure No.	Figure Caption	Page No.
3.1	ER Diagram for online movie ticket booking system	3
3.2	User Login page	12
3.3	Register page	12
3.4	Home page	13
3.5	Movie details page	13
3.6	Seat details page	14
3.7	Book tickets page	14
3.8	Success page	15
3.9	Myticket page	15
3.10	Change password page	16
3.11	Error page	16
3.12	Admin view customer page	17
3.13	Admin view movie page	17
3.14	Admin view show page	18
3.15	Admin view Ticket page	19
5.1	Invalid user login	25
5.2	Successful Registration	24
5.3	Ticket Booking	27
5.4	No tickets booked	28
5.5	Successful password change	29
5.6	Successfully deleted customer	30
5.7	Successfully updated movie details	32
5.8	Successfully updated show details	33
5.9	Successfully deleted ticket	34

LIST OF ABBREVIATIONS

ABBREVIATION	DESCRIPTION
CBFC	Central Board For Film Certification

CHAPTER 1

INTRODUCTION

1.1 PERSPECTIVE

The purpose to introduce the movie ticket booking system is to save the precious time of the customer. Movie ticket booking system is an application which is used to provide the information about movies screening in theatre and availability of seats. The customer can login to the account using mobile number and password. If he/she is new to this they can sign up to create an account and login to the system. Dashboard displays the movies that are currently screening in the theatre. The customer can view the details about the movie like movie name, cast, rating etc.by clicking the movie. The customer can also view the available seats, ticket price and can book the tickets based on availability of seats. After booking, the user can view and even cancel the tickets they don't want.

1.2 OBJECTIVE

The main objective of this system is to let the customers book the ticket online which helps to save the customer's time. This application allows customer to view the screening movie and book the tickets.

1.3 SCOPE

This system would be very popular among people who don't like to waste their time. Also, this system would be perfect for use during the pandemic times where people must follow social distancing even after the end of lockdowns.

CHAPTER 2

REQUIREMENT DESCRIPTION

2.1 FUNCTIONAL REQUIREMENTS

The functional requirement in Movie ticket booking system is the collective information about what are the operations available in the system.

Authorization:

It should provide the functionality of authentication for the valid users.

View Movies:

It should provide the functionality of viewing the movie details to the customers

Select number of seats and book tickets:

It should provide the functionality for booking the tickets to the customers

View Tickets and cancel tickets:

The system must allow the user to view and cancel the tickets they have booked.

Change Password:

The system must allow the user to change their password.

Admin Rights:

The admin must have the provision to view and delete the users and tickets while also being able to update customer details and also update the movie and show details of the theatre.

2.2 NON-FUNCTIONAL REQUIREMENTS

The non-functional requirement describes about platform and physical resource required for building the Movie ticket booking system.

- The details of the customers and their login details are stored in database server. So the information will be reliable and must be stored efficiently in safe manner.
- The application must be user friendly and must be very interactive to the customer.
- The customer must have the internet availability while using this application.

CHAPTER 3

SYSTEM DESIGN

3.1 ER Diagram for Movie ticket booking system:

The ER Diagram for Movie Ticket Booking System depicts all the entities, relationships and attributes that represent the data about movies, customers, shows and tickets. Without access to this data, the system won't be able to function.

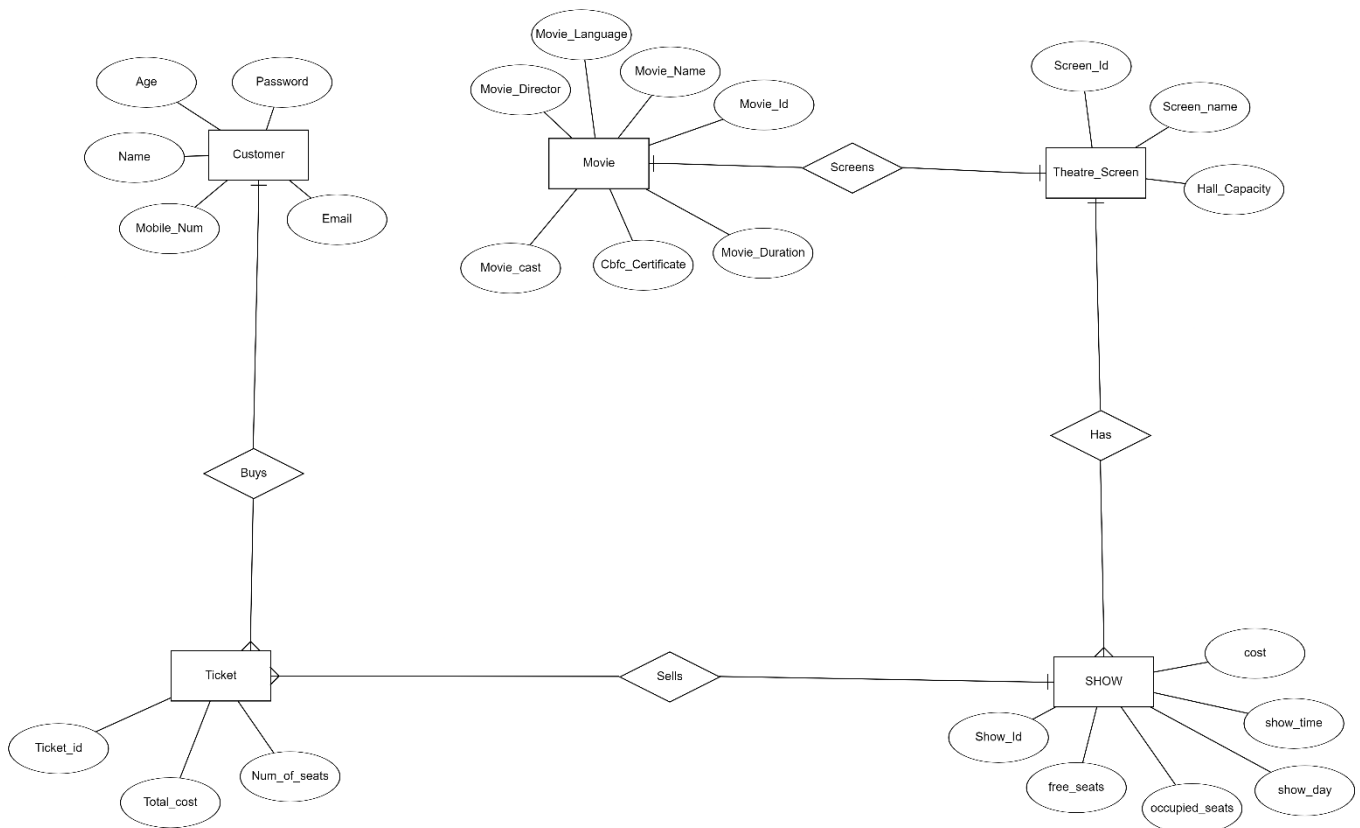


Figure 3.1: ER Diagram of Movie ticket booking system

From the **Figure 3.1** we can clearly see all the attributes in each entity used in the system. The ticket entity is linked to movies entity through the Show and Theatre_Screens entities. The details about all attributes, their data types, constraints and functionality are discussed in detail below.

The Movie Ticket Booking System consists of five entities: Customer, Movie, Theatre_Screens, Show and Ticket.

The Customer Entity contains Customer's details such as their registered number, name, age and email. The registered number is used as the primary key as each phone number is unique. The name and age may not be unique. Whenever a customer books a ticket, the booking is done using the registered mobile number of that customer.

The Movie Entity contains all the details about the movies being screened in the three different screens of the theatre. The details saved include Movie name, director, language, duration, CBFC certificate and cast. Since the Movie name may not be unique, we use the Movie ID as the primary key in this entity.

The Theatre_Screens entity contains details such as the Screen name, screen capacity and Movie ID. The movie id is a foreign key from the Movie entity. It represents which movie is being screened in which screen. The primary key of this entity is Screen_Id. The hall capacity represents the maximum number of seats in the hall and hence the total number of tickets available in that screen.

The Show entity contains attributes such as Occupied seats, free seats, Show day, Show time and Ticket Cost. The primary key of this entity is Show_Id. It also contains a foreign key from Screen entity called Screen_Id. Each movie is screened in a screen for four shows a day. Hence, this entity stores details about all those shows uniquely.

The Ticket entity contains details about each ticket booked in the theatre by the customers. The details include Ticket Id, User Id, Show Id, Number_of_seats, Total_Cost. The Ticket_Id acts as the primary key in this entity. Show Id is a foreign key that is derived from the Show entity.

This is used to display the Show day and time and from Show_Id other information such as Movie Details can also be derived from the Theatre_Screen entity which links Show with Movie entity. The other foreign key used in this entity is User_Id which is the Mobile_Num attribute derived from the Customer entity.

The entities Customer is related to only one other entity: Ticket. The relationship between these 2 entities is "Customer buys tickets" and the cardinality is "One to Many". That is, each Customer can have many tickets while each ticket can be associated with only one Customer.

Similarly, the entity Theatre_Screens is related to the entity Movie in the "Theatre Screens Movies" relationship with "One to One" cardinality. This pertains that each Movie will be screened in exactly one Screen and vice versa.

Next, the Theatre_Screen entity and Show entity are related in the “Screen has Shows” relationship with cardinality “One to Many”. That is, each Screen has multiple Shows (in this case, 4 shows per day). However, each Show can be associated with only one screen, even if the Show time and date are the same.

The last relationship in this system is between the Ticket entity and the Show entity. The relationship is “Show sell Tickets”. These two entities share a “One to Many” cardinality signifying the fact that each Ticket can be associated with exactly one Show whereas each Show can have many tickets associated with it.

When these relationships are joined together, from the Ticket, using the Show ID, we can access Theatre_Screen from which we get the Movie Id in order to get the details about the Movie for which the tickets have been purchased.

3.2 DESIGN COMPONENTS

3.2.1 Front End:

The Movie ticket booking system uses ASP.NET for developing interactive pages along with Bootstrap 5.

3.2.2 Back End:

Uses Microsoft SQL database for back end to store data. Validation is done using C#.

3.3 DATABASE DESCRIPTION

Listed below gives a description of database document schemas used for Movie ticket booking system

3.3.1 Customer Structure

As shown in **Table 3.1**, customer structure contains the details of the customers.

Table 3.1: Customer Description

Attribute Name	Type	Width	Constraint(s)	Description
Mobile_Num	Numeric	18,0	Primary Key, Not Null	Phone number of the user
Name	Varchar	50	Not Null	Name of the user
Age	Int		>=13	User’s Age

Password	Varchar	50	Not Null	Password for user login
Email	Varchar	50	Not Null	Email Address of user

3.3.2 Movie Structure

As shown in **Table 3.2**, movie structure contains the details of the movies.

Table 3.2: Movie Description

Attribute Name	Type	Width	Constraint(s)	Description
Movie_Id	Int		Primary Key, Not Null	ID of the movie
Movie_Name	Varchar	50	Not Null	Name of the movie
Movie_Language	Varchar	50	Not Null	Movie language
Movie_Director	Varchar	50	Not Null	Movie Director's name
Movie_Cast	Varchar	255	Not Null	Actors in the movie
Cbfc_Certificate	Varchar	50	U or U/A or A	Movie Certification
Movie_Duration	Int		Not Null	Length of the movie

3.3.3 Theatre Screens Structure

As shown in **Table 3.3**, theatre screens structure contains the details of the screens.

Table 3.3: Screen Description

Attribute Name	Type	Width	Constraint(s)	Description
Screen_Id	Int		Primary Key, Not Null	ID of the screen
Movie_Id	Int		Foreign Key, Not Null	ID of movie being screened
Screen_name	Varchar	50	Not Null	Name of the movie screen
Hall_Capacity	Int		Not Null	Total seats in the hall

3.3.4 Show Structure

As shown in **Table 3.4**, show structure contains the details of the show.

Table 3.4: Show Description

Attribute Name	Type	Width	Constraint(s)	Description
Show_Id	Int		Primary Key, Not Null	ID of the show
Screen_Id	Int		Foreign Key, Not Null	ID of the screen of the show
occupied_seats	Int		Not Null	Number of booked seats
free_seats	Int		Not Null	Number of seats available for booking
show_day	date		Not Null	Day of screening
Show_time	time	7	Not Null	Time of screening
cost	Int		Not Null	Cost per ticket

3.3.5 Ticket Structure

As shown in **Table 3.5**, ticket structure contains the details of the ticket.

Table 3.5: Ticket Description

Attribute Name	Type	Width	Constraint(s)	Description
Ticket_id	Int		Primary Key, Not Null	ID of tickets
Show_Id	Int		Foreign Key, Not Null	ID of the show
User_Id	Numeric	18,0	Foreign Key, Not Null	Mobile number of the user who booked the tickets
Num_of_seats	Int		Not Null	Number of tickets booked by the user
Total_cost	Int		Not Null	Total cost of all tickets

3.4 LOW LEVEL DESIGN

The following section illustrates the functionalities of the system. This includes login to the application, booking and canceling tickets.

3.4.1 Login

Table 3.6 shows the login details of the application.

Table 3.6 Login Details

Files used	Signin.aspx
Short Description	Allows the user to login to the application
Arguments	Phone Number, Password
Return	Success/Failure in login
Pre-Condition	The user must have an account
Post-Condition	The home page will be displayed
Exception	Invalid Phone Number and password
Actor	User, Admin

3.4.2 Register

Table 3.7 shows the register user page.

Table 3.7 Register user

Files used	Register.aspx
Short Description	Allows the user to register to the application
Arguments	Name, mail-id, age, Phone Number, Password
Return	Success/Failure in register
Pre-Condition	The user must have entered the details mentioned above
Post-Condition	The Login page will be displayed
Exception	Account already exists
Actor	User

3.4.3 Book Tickets

Table 3.8 shows the feature to book tickets for a movie.

Table 3.8 Book Tickets

Files used	selectSeats.aspx
-------------------	------------------

Short Description	Allows the user to check and book the tickets for the show
Arguments	Show ID
Return	Success/Failure in booking the tickets
Pre-Condition	The user must have an account and select the show they wish
Post-Condition	The success page will be displayed
Exception	Requested Seats are unavailable
Actor	User

3.4.4 View booked tickets

Table 3.9 shows the tickets booked by each user.

Table 3.9 View booked Tickets

Files used	myTickets.aspx
Short Description	Allows the user to view the details of booked show and allow to cancel the show
Arguments	User's mobile number
Return	Success/Failure in viewing my tickets
Pre-Condition	The user must have booked the tickets for the show
Post-Condition	The my tickets page will be displayed
Exception	Error in viewing my tickets
Actor	User

3.4.5 Change Password

Table 3.10 shows the change password feature.

Table 3.10 Change Password

Files used	changePwd.aspx
Short Description	Allows the user to change the password if they feel it insecure
Arguments	Registered phone number, old and new passwords
Return	Success/failure in changing password
Pre-Condition	The user must have an account to change the password
Post-Condition	Password updated in database
Exception	Passwords don't match
Actor	User

3.4.6 Delete User

Table 3.11 shows the delete user feature.

Table 3.11 Delete User

Files used	AdminViewCustomers.aspx
Short Description	Allows the admin to view and delete existing user accounts
Arguments	Registered phone number
Return	Success/failure in deleting account
Pre-Condition	Admin must select an account to delete
Post-Condition	The account gets removed
Exception	Invalid number
Actor	Admin

3.4.7 Update User

Table 3.12 shows the Update user feature.

Table 3.12 Update user

Files used	adminViewCustomers.aspx
Short Description	Allows the admin to view and update existing user accounts
Arguments	Registered phone number
Return	Success/failure in updating details
Pre-Condition	The user must request admin for changing details
Post-Condition	Details updated
Exception	The details are not sufficient
Actor	Admin

3.4.8 Update Movies

Table 3.13 shows the Update Movies feature.

Table 3.13 Update Movies

Files used	adminViewMovies.aspx
Short Description	Allows the admin to view and update movies
Arguments	Movie ID
Return	Success/failure in updating details

Pre-Condition	The admin must choose a movie to update
Post-Condition	Movie details updated
Exception	Insufficient details
Actor	Admin

3.4.9 Update Show

Table 3.14 shows the Update show feature.

Table 3.14 Update show

Files used	adminViewShows.aspx
Short Description	Allows the admin to view and update show
Arguments	Show ID
Return	Success/failure in updating details
Pre-Condition	The admin must choose a show to update
Post-Condition	Show details updated
Exception	Insufficient details
Actor	Admin

3.4.10 Delete Ticket

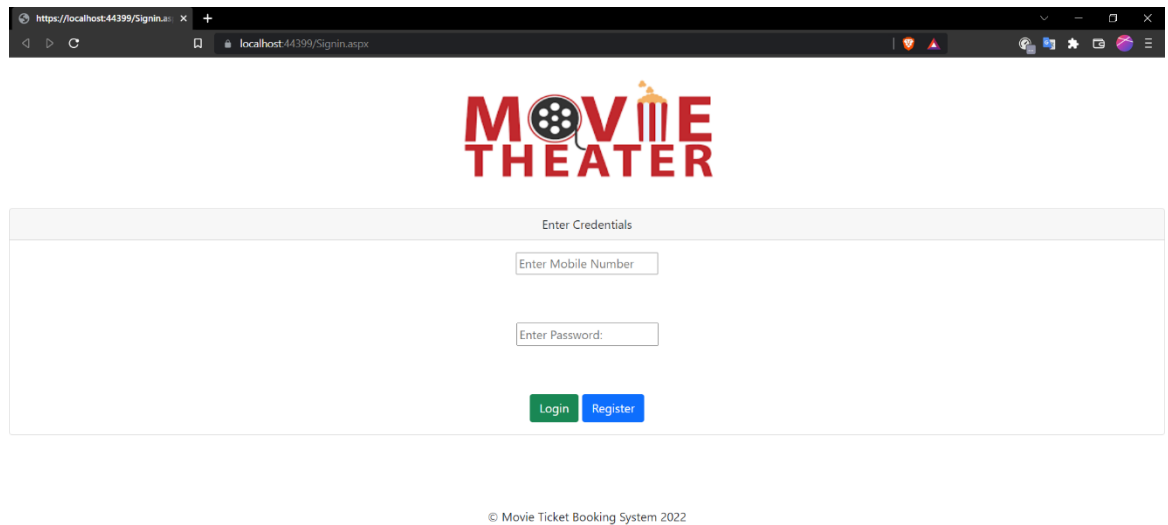
Table 3.15 shows the delete ticket feature.

Table 3.15 Delete Ticket

Files used	adminViewTickets.aspx
Short Description	Allows the admin to view and delete booked tickets
Arguments	Ticket ID
Return	Success/failure in deleting tickets
Pre-Condition	The admin must choose a ticket to delete
Post-Condition	Ticket deleted successfully
Exception	Invalid ticket id
Actor	Admin

3.5 USER INTERFACE DESIGN

3.5.1 User Login

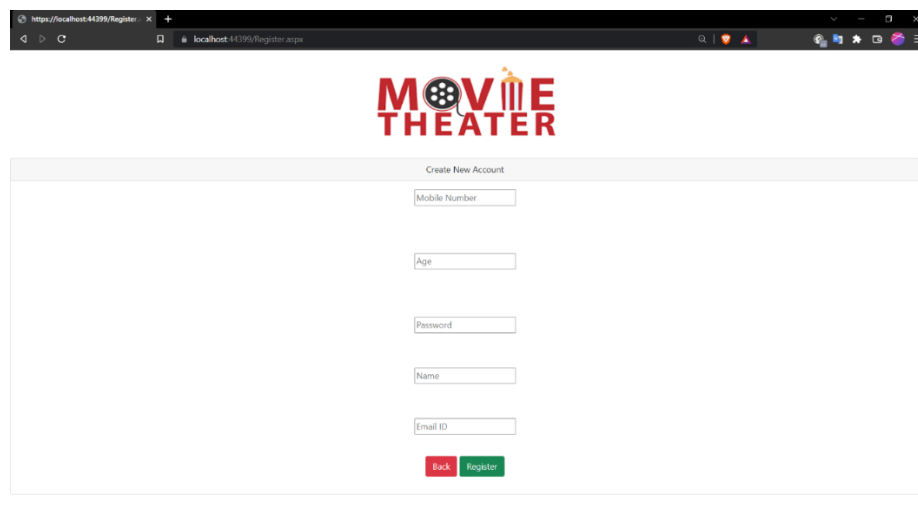


The screenshot shows a web browser window with the URL `https://localhost:44399/SignIn.aspx`. The page features the "MOVIE THEATER" logo at the top center. Below the logo is a form titled "Enter Credentials". The form contains two input fields: "Enter Mobile Number" and "Enter Password:". At the bottom of the form are two buttons: "Login" (green) and "Register" (blue). Below the form, the text "© Movie Ticket Booking System 2022" is displayed.

Figure 3.2: User Login

Figure 3.2 provide the interface for user login. The customer can login to the Movie ticket booking system by entering their registered mobile number and password. If they not have an account, they can click on register option to signup

3.5.2 Register page



The screenshot shows a web browser window with the URL `https://localhost:44399/Register.aspx`. The page features the "MOVIE THEATER" logo at the top center. Below the logo is a form titled "Create New Account". The form contains five input fields: "Mobile Number", "Age", "Password", "Name", and "Email ID". At the bottom of the form are two buttons: "Back" (red) and "Register" (green).

Figure 3.3: Register Page

Figure 3.3 provide the interface for Register page. The register page is for the first-time customers who register account for first time. The details to be filled by the customers are name, age, mail-id, phone number and set password for the account.

3.5.3 Home page

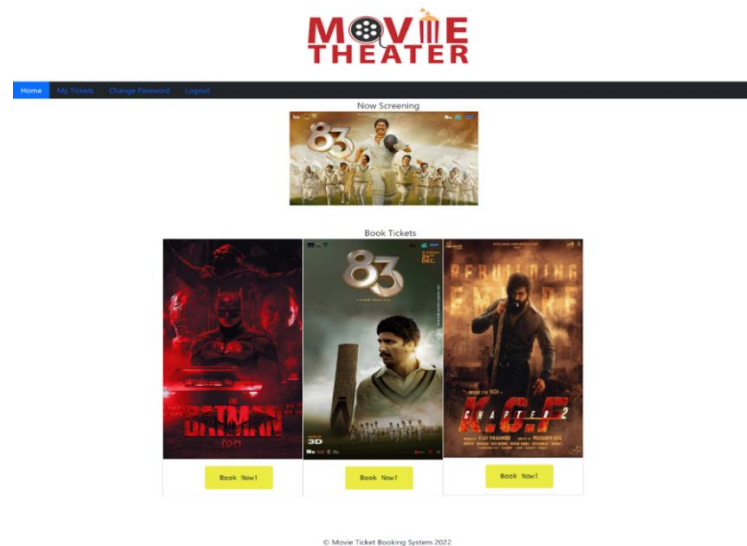


Figure 3.4: Home Page

Figure 3.4 provide the interface for Home page. After successful login the list of movies that are currently screening on theatres are available and to view the details about it click book now button.

3.5.4 Movie Details page

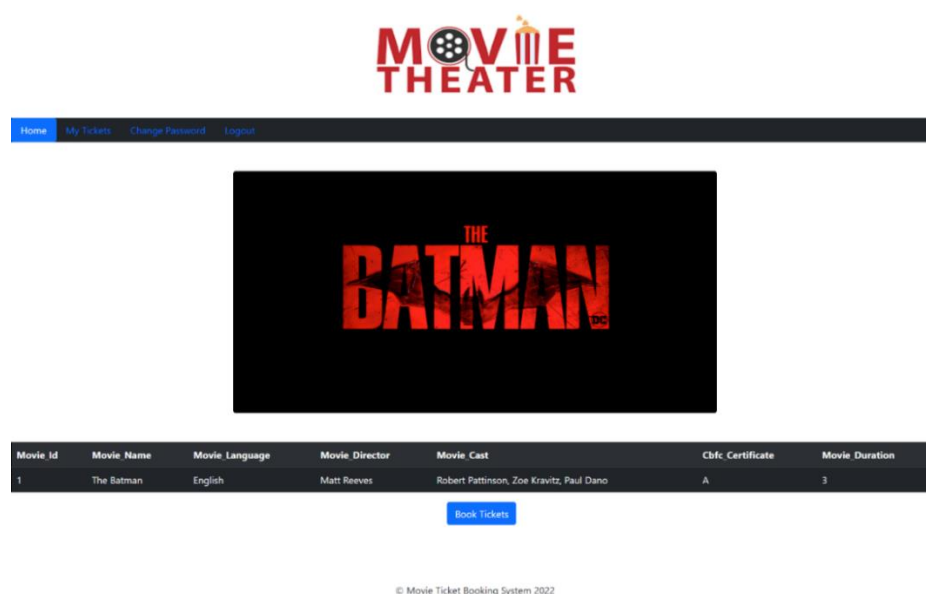


Figure 3.5: Movie Details Page

Figure 3.5 provide the interface for Movie Details page. After selecting the movie the movie details were displayed in the movie details page where we can view the movie details like cast, language, director, CBFC- certificate, duration of the movie and to book the tickets click on book Tickets.

3.5.5 Seat Details page



Home	My Tickets	Change Password	Logout
------	------------	-----------------	--------


Show ID	Booked Seats	Available Seats	Show Date	Show Time	
1	0	200	1 May 2022	10:00 AM	Book Tickets
4	0	200	1 May 2022	2:00 PM	Book Tickets

© Movie Ticket Booking System 2022

Figure 3.6: Seat Details Page

Figure 3.6 provide the interface for Seat Details page. After selecting and booking the show the seat availability will show in seats detail page. This page consists of booked seats, available seats, show date, show time. If the tickets are available then book the number of seats wanted and click on Book Tickets

3.5.6 Book Tickets page



Home	My Tickets	Change Password	Logout
------	------------	-----------------	--------

Select Seats

Number of Free Seats:

Cost per Ticket:

Enter Number of seats

Total Cost:

[Book Tickets](#)

© Movie Ticket Booking System 2022

Figure 3.7: Book Tickets Page

Figure 3.7 provide the interface for Book Tickets page. The book tickets page consists of cost per seats and total price of the number of seats entered. After viewing the price if it is affordable then click on Book Tickets.

3.5.7 Success page

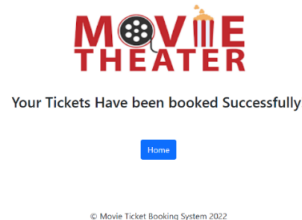


Figure 3.8 Success Page

Figure 3.8 provide the interface for Success page. This page is displayed after a customer selects their seats and clicks “Book Tickets” button. This is the confirmation page for the successful booking of tickets.

3.5.8 My Tickets page

Figure 3.9 provide the interface for My Tickets page

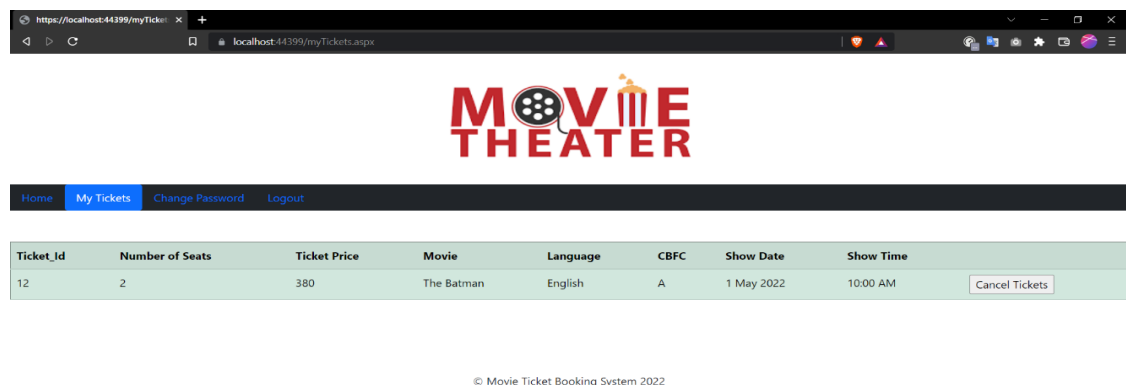
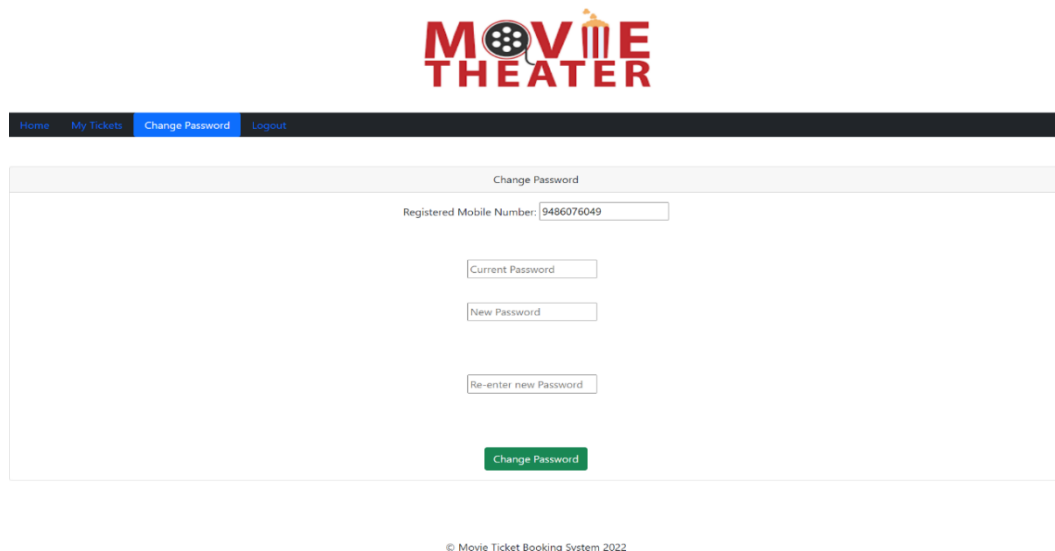


Figure 3.9: My Tickets Page

Figure 3.9 provide the interface for My Tickets page. My Ticket page displays the detail view about movie , booked seats , date, time , price for booked tickets. If the customer is not able to attend the show, they can cancel Tickets by click on Cancel Tickets.

3.5.9 Change Password page



The screenshot shows the 'Change Password' page of the 'MOVIE THEATER' application. At the top is the application logo. Below it is a navigation bar with links: Home, My Tickets, Change Password (highlighted), and Logout. The main content area is titled 'Change Password' and contains a form with the following fields: 'Registered Mobile Number' (with the value 9486076049), 'Current Password', 'New Password', and 'Re-enter new Password'. A green 'Change Password' button is at the bottom of the form. The footer of the page reads '© Movie Ticket Booking System 2022'.

Figure 3.10: Change Password Page

Figure 3.10 provide the interface for Change Password page. The customer can change their password by clicking change password and redirected to change password page the customer should change the password by giving current password and new password . To confirm the password customer have to reenter the password and change password button should be clicked.

3.5.10 Error page

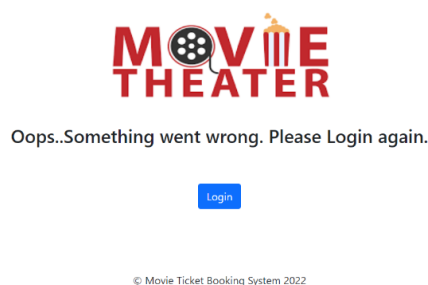



Figure 3.11 Error Page

Figure 3.11 provide the interface for Error page. This page is displayed, if in between the session times out or the user performs any undesired actions. This prevents any harmful events from happening such as accidental booking of duplicate tickets. Once this page appears, the user will have to log in again to continue their work.

3.5.11 Admin View Customers page



Customers	Movies	Shows	Tickets	Logout
Mobile Number	Age	Name	Email	
7904137677	20	vignesh	vignesh@gmail.com	
8489603365	21	Manoj	manoj@gmail.com	
8903143705	18	manoj	manoj@gmail.com	
9486076049	20	lanaard	lanaard@gmail.com	

Select Customer to delete:

Age:

Name:

Email:


Warning: Tickets already booked will be canceled

© Movie Ticket Booking System 2022

3.12 Admin View Customer page

Figure 3.12 provide the interface for Admin View Customers page. The Admin can view the details of the customer like registered mobile number, age, name, mail-id and They can also view the movies details booked by the customer.

3.5.12 Admin View Movies page



Customers	Movies	Shows	Tickets			
Movie ID	Movie	Language	Director	Cast	Certificate	Duration
1	The Batman	English	Matt Reeves	Robert Pattinson, Zoe Kravitz, Paul Dano	A	3
2	83	Hindi	Kabir Khan	Ranbir Singh, Jiiva, Deepika Padukone, Hardy Sandhu	U	3
3	KGF Chapter 2	Tamil	Prashanth Neel	Yash, Srinidhi Shetty, Sanjay Dutt, Raveena Tandon	U/A	3

Update Movies:

Movie Name:

Movie Language:

Movie Director:

Movie Cast:

Movie Certificate:

Movie Duration:

Warning: Tickets already booked will be canceled

© Movie Ticket Booking System 2022


Figure 3.13 Admin View Movies page

Figure 3.13 provide the interface for Admin View Movies page. The admin can update the movies details by entering the movie id in the drop down list box. Once that is done the details about the Movies in a screen are displayed in a textbox which can be edited. After editing, the Update button is pressed and the Ticket details will be updated.

Note:

Making changes to the Movie and Show tables leads to the canceling of tickets already booked for that Movie/Show.

3.5.13 Admin View Shows page



Customers	Movies	Shows	Tickets			
Show ID	Screen ID	Occupied Seats	Free Seats	Show Day	Show Time	Cost Per Ticket
1	1	0	200	1 May 2022	10:00 AM	190
2	2	0	150	1 May 2022	10:00 AM	150
3	3	0	100	1 May 2022	10:00 AM	120
4	1	0	200	1 May 2022	2:00 PM	190
5	2	0	150	1 May 2022	2:00 PM	150
6	3	0	100	1 May 2022	2:00 PM	120

Update Movies Select

Screen ID

Show Day

Show Time

Ticket Cost

Update

Warning: Tickets already booked will be canceled

© Movie Ticket Booking System 2022

Figure 3.14 Admin View Shows page

Figure 3.14 provide the interface for Admin View Shows page. The admin can view show pages and update the details of show like seats occupied, Free seats, show day, show time, cost per ticket. It can be done by entering the show id in the list box and details which is to be updated are filled then it is updated.

3.5.14 Admin View Tickets page

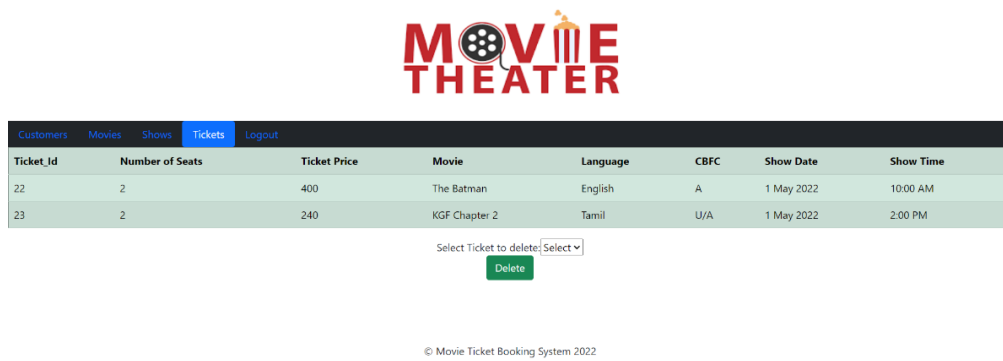


Figure 3.15 Admin View Tickets page

Figure 3.15 provide the interface for Admin View Tickets page. The admin can delete the booked ticket based on customer's request. If the customer requested to cancel the ticket then admin enter the ticket id in list box then tickets booked by the customer is deleted.

CHAPTER 4

SYSTEM IMPLEMENTATION

4.1 LOGIN IMPLEMENTATION

The login credentials are obtained. If the credentials are OK, then the user is redirected to the homepage

```
GET phone number, password
  IF phone number, password valid
    RETURN homepage
  ELSE
    RETURN Invalid Credential
```

4.2 SIGNUP IMPLEMENTATION

The form fields are obtained. If they are valid, then the customer details are added to the database.

```
GET requestFields
  IF requestFields valid
    RETURN added to db
  ELSE
    RETURN enter valid details
```

4.3 BOOKING IMPLEMENTATION

The form fields are obtained. If they are valid, then the booking details are listed.

```
GET countOfTheSeats
  IF countOfTheSeats valid
    RETURN seats are available
  ELSE
    RETURN requested seats are not available
```

4.4 CHANGE PASSWORD IMPLEMENTATION

The form fields are obtained. If they are valid, then the password is changed.

GET currentpassword

IF currentpassword valid

 GET newpassword,reenterpassword

 IF newpassword == reenterpassword

 RETURN password changed

 ELSE

 RETURN passwords don't match

ELSE

 RETURN current password is incorrect

4.5 CANCEL TICKET IMPLEMENTATION

Booked tickets can be canceled by the user.

GET userid

IF user has tickets

 DISPLAY ticketlist

 GET selectedticket

 RETURN deleted successfully

ELSE

 RETURN no tickets booked

4.6 ADMIN DELETE USER IMPLEMENTATION

Admin has the rights to view and delete existing users.

GET userlist

IF userlist is empty

 RETURN no users

ELSE

 GET userid

 DELETE user

 RETURN deleted successfully

4.7 ADMIN UPDATE MOVIES IMPLEMENTATION

Admin is responsible for updating the Movie details

GET movielist

DISPLAY movielist

GET movieid

DISPLAY moviedetails

GET updateddetails

IF any detail is empty

 RETURN Please enter all fields

ELSE

 GET tickets booked for that movie

 DELETE tickets

 UPDATE moviedetails

 RETURN movie details updated and tickets have been deleted

4.8 ADMIN UPDATE SHOWS IMPLEMENTATION

Admin is responsible for updating the Show details

GET showlist

DISPLAY showlist

GET showid

DISPLAY showdetails

GET updateddetails

IF any detail is empty

 RETURN Please enter all fields

ELSE

 GET tickets booked for that show

 DELETE tickets

 UPDATE showdetails

 RETURN show details updated and tickets have been deleted

4.9 ADMIN DELETE TICKETS IMPLEMENTATION

Admin has the rights to view and delete booked tickets.


```
GET ticketlist
IF ticketlist is empty
    RETURN no tickets
ELSE
    GET ticketid
    DELETE ticket
    RETURN deleted successfully
```

4.10 ADMIN UPDATE USER IMPLEMENTATION

Admin has the rights to view and update the existing user accounts.

```
GET userlist
DISPLAY userlist
GET mobile_num
DISPLAY userdetails
GET updateddetails
IF any detail is empty
    RETURN Please enter all fields
ELSE
    GET tickets booked by that user
    DELETE tickets
    UPDATE customerdetails
    RETURN customer details updated and tickets have been deleted
```

CHAPTER 5

RESULTS AND DISCUSSION

5.1 TEST CASES AND RESULTS

5.1.1 Test Cases and Results for Login function:

The **Table 5.1**, **Table 5.2** shows the possible test data for both the positive and negative test case given below, if the customer is already having an account, then the output is true otherwise it is false. After a successful login, the home page will be displayed.

Table 5.1: Positive Test Case and result for Login

Test Case ID	TC1
Test Case Description	It tests whether the given login details are valid or not
Test Data	8489803365,manojkumar
Expected Output	TRUE
Result	PASS

Table 5.2: Negative Test Case and result for Login

Test Case ID	TC2
Test Case Description	It tests whether the given login details are valid or not
Test Data	9434235443,kat
Expected Output	FALSE
Result	PASS



Enter Credentials

1234567890

Enter Password:

User doesn't exist

Login Register

© Movie Ticket Booking System 2022

Figure 5.1 Invalid user login

Figure 5.1 shows the system's response when the user tries to enter an invalid mobile number (not registered).

5.1.2 Test Cases and Results for Register function:

The **Table 5.3**, **Table 5.4** shows the possible test data for both the positive and negative test case given below, if the customer already has an account then, the output is true otherwise it is false.

Table 5.3: Positive Test Case and result for Register user

Test Case ID	TC1
Test Case Description	Creates new user
Test Data	9486076049, vignesh, helloworld, 20, vignesh@gmail.com
Expected Output	TRUE
Result	PASS

Table 5.4: Negative Test Case and result for Register user

Test Case ID	TC2
Test Case Description	Creates new user
Test Data	Fields are empty
Expected Output	FALSE
Result	PASS



Create New Account

8903143705

18

Password

manoj

manoj@gmail.com

Registration Successful

Back Register

© Movie Ticket Booking System 2022

Figure 5.2 Successful Registration

Figure 5.2 shows the output displayed when a new user is registered successfully. If any fields are empty, the user won't be registered

5.1.3 Test Cases and Results for Booking Tickets function:

The **Table 5.5**, **Table 5.6** shows that the possible test data for the both positive and negative test case given below, if the customer selects proper number of seats, then the output is true otherwise false.

Table 5.5: Positive Test Case and result for Booking tickets

Test Case ID	TC1
Test Case Description	Selecting show, seats and booking tickets
Test Data	Show:1, Seats: 5
Expected Output	TRUE
Result	PASS

Table 5.6: Negative Test Case and result for Booking tickets

Test Case ID	TC2
Test Case Description	Selecting show, seats and booking tickets

Test Data	Show:1, Seats: 250
Expected Output	FALSE
Result	PASS

Executed



[Home](#)
[My Tickets](#)
[Change Password](#)
[Logout](#)

Select Seats

Number of Free Seats:

Cost per Ticket:

Seat Limit exceeded

Total Cost:

Book Tickets

© Movie Ticket Booking System 2022

Figure 5.3 Ticket Booking

Figure 5.3 shows the ticket booking page after the user selects a valid number of seats for booking.

5.1.4 Test Cases and Results for Cancel Ticket function:

The **Table 5.7**, **Table 5.8** shows that the possible test data for the both positive and negative test case given below, if the customer already has tickets booked and chooses to cancel that, then the output is true otherwise false.

Table 5.7: Positive Test Case and result for Canceling ticket

Test Case ID	TC1
Test Case Description	Cancel booked tickets
Test Data	Ticket ID: 1
Expected Output	TRUE
Result	PASS

Table 5.8: Negative Test Case and result for Canceling ticket

Test Case ID	TC2
Test Case Description	Cancel booked tickets
Test Data	No ticket booked
Expected Output	FALSE
Result	PASS

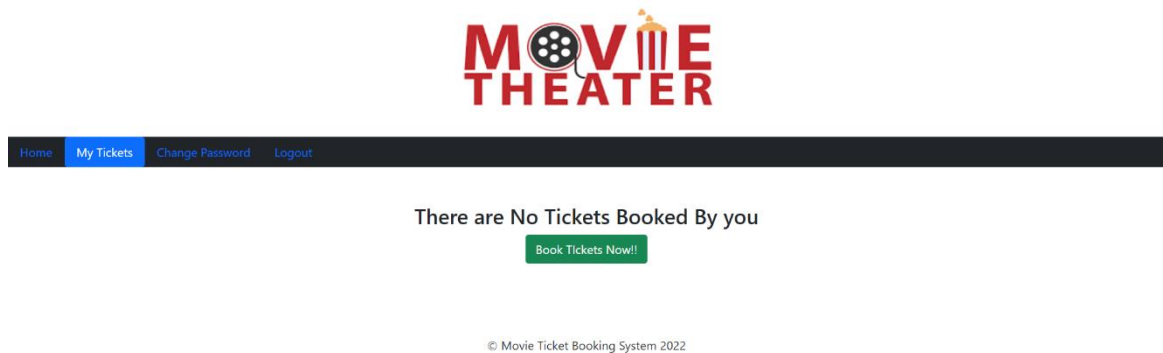


Figure 5.4 No tickets booked

Figure 5.4 shows the page that will be displayed when the user tries to delete or view their tickets but have no tickets booked to their registered mobile number.

5.1.5 Test Cases and Results for Change Password function:

The **Table 5.9**, **Table 5.10** shows that the possible test data for the both positive and negative test case given below, if the customer enters the current password correctly, then the output is true otherwise false.

Table 5.9: Positive Test Case and result for Changing Password

Test Case ID	TC1
Test Case Description	User changes their password
Test Data	Current password: helloworld, New password: newpassword
Expected Output	TRUE

Result	PASS
---------------	------

Table 5.10: Negative Test Case and result for Changing Password

Test Case ID	TC2
Test Case Description	User changes their password
Test Data	Current password: wrongpassword, New password: newpassword
Expected Output	FALSE
Result	PASS



[Home](#)
[My Tickets](#)
[Change Password](#)
[Logout](#)

Change Password

Registered Mobile Number:

Password changed successfully

[Change Password](#)

© Movie Ticket Booking System 2022

Figure 5.5 Successful password change

Figure 5.5 shows the page that will be displayed after the user changes their password correctly. For this, the user must enter their current password correctly and also the new password must be entered correctly twice.

5.1.6 Test Cases and Results for Admin Delete Customer function:

The **Table 5.11**, **Table 5.12** shows that the possible test data for the both positive and negative test case given below, if the admin enters a valid Customer ID, it can be deleted.

Table 5.11: Positive Test Case and result for Admin Deleting Customer

Test Case ID	TC1
Test Case Description	Admin has the rights to delete a customer
Test Data	Customer ID; 1
Expected Output	TRUE
Result	PASS

Table 5.12: Negative Test Case and result for Admin Deleting Customer

Test Case ID	TC2
Test Case Description	Admin has the rights to delete a customer
Test Data	Customer ID:100
Expected Output	FALSE
Result	PASS



Customers	Movies	Shows	Tickets	Logout
Mobile Number	Age	Name	Email	
7904137677	20	vignesh	vignesh@gmail.com	
8489803365	21	Manoj	manoj@gmail.com	
9486076049	20	lanaard	lanaard@gmail.com	

Select Customer to delete:

Warning: Tickets already booked will be canceled
Customer Deleted successfully

© Movie Ticket Booking System 2022

Figure 5.6 Successfully deleted customer

Figure 5.6 shows the response displayed by the server when the admin deletes a customer by choosing their Mobile Number from the drop-down list.

5.1.7 Test Cases and Results for Admin Update Movie Function:

The **Table 5.13**, **Table 5.14** shows that the possible test data for the both positive and negative test case given below, if the admin enters all fields properly after selecting a movie ID, the details will be updated properly.

Table 5.13: Positive Test Case and result for Admin Updating Movie

Test Case ID	TC1
Test Case Description	Admin can change the Movie details such as title, director, cast, certificate, cast, language, duration.
Test Data	Movie ID: 1, Movie Name: Beast, Movie Director: Nelson, Cast: Vijay, Pooja Certificate: U/A, Language: Tamil, Duration: 3
Expected Output	TRUE
Result	PASS

Table 5.14: Negative Test Case and result for Admin Updating Movie

Test Case ID	TC2
Test Case Description	Admin can change the Movie details such as title, director, cast, certificate, cast, language, duration.
Test Data	All fields are empty
Expected Output	FALSE
Result	PASS



Movie ID	Movie	Language	Director	Cast	Certificate	Duration
1	The Batman	English	Matt Reeves	Robert Pattinson, Zoe Kravitz, Paul Dano	A	3
2	83	Hindi	Kabir Khan	Ranbir Singh, Jhvi, Deepika Padukone, Hardy Sandhu	U	3
3	KGF Chapter 2	Tamil	Prashanth Neel	Yash, Srinidhi Shetty, Sanjay Dutt, Raveena Tandon	U/A	3

Update Movies Select

Movie Name
 Movie Language
 Movie Director
 Movie Cast
 Movie Certificate
 Movie Duration

Warning: Tickets already booked will be canceled

Movie Updated Successfully

© Movie Ticket Booking System 2022

Figure 5.7 Successfully updated movie details

Figure 5.7 shows the output displayed when the admin properly updates a movie's details by selecting proper id and entering all mandatory fields.

5.1.8 Test Cases and Results for Admin Update Show function:

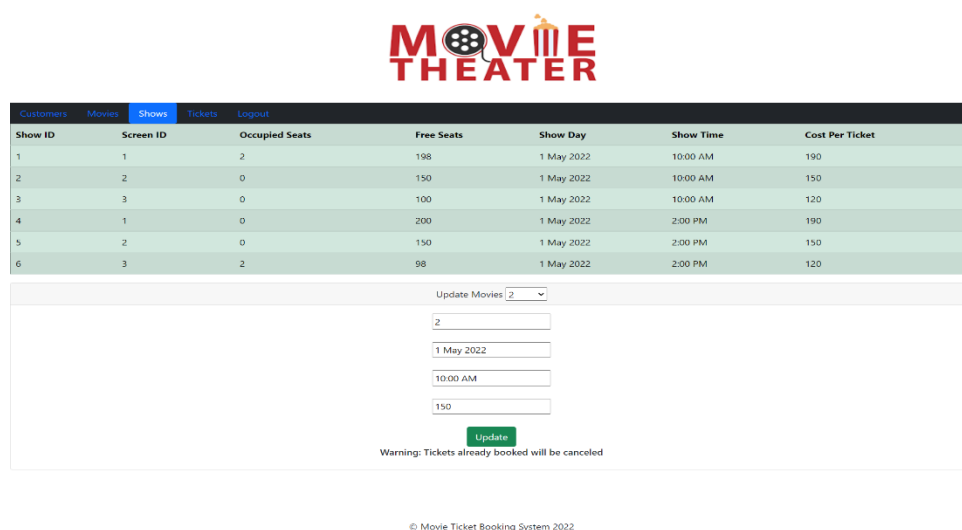
The **Table 5.15**, **Table 5.16** shows that the possible test data for the both positive and negative test case given below, if the admin enters all fields properly, then the details will be updated in the database.

Table 5.15: Positive Test Case and result for Admin updating shows

Test Case ID	TC1
Test Case Description	Admin can change the show details such as screen, date and time, movie
Test Data	Screen: 2, Date: May 1 2022, Time: 6 PM, Movie: 1
Expected Output	TRUE
Result	PASS

Table 5.16: Negative Test Case and result for Admin updating shows

Test Case ID	TC2
Test Case Description	Admin can change the show details such as screen, date and time, movie
Test Data	Empty fields
Expected Output	FALSE
Result	PASS



MOVIE THEATER

Customers Movies **Shows** Tickets Logout

Show ID	Screen ID	Occupied Seats	Free Seats	Show Day	Show Time	Cost Per Ticket
1	1	2	198	1 May 2022	10:00 AM	190
2	2	0	150	1 May 2022	10:00 AM	150
3	3	0	100	1 May 2022	10:00 AM	120
4	1	0	200	1 May 2022	2:00 PM	190
5	2	0	150	1 May 2022	2:00 PM	150
6	3	2	98	1 May 2022	2:00 PM	120

Update Movies | 2

2

1 May 2022

10:00 AM

150

Update

Warning: Tickets already booked will be canceled

© Movie Ticket Booking System 2022

Figure 5.8 Successfully updated show details

Figure 5.8 shows the page that will be displayed after the admin has successfully updated show details by selecting proper id from the drop-down list and entering all values in the textbox properly.

5.1.9 Test Cases and Results for Admin Delete Ticket function:

The **Table 5.17**, **Table 5.18** shows that the possible test data for the both positive and negative test case given below, if the admin enters a valid, existing ticket ID, it will be deleted from the database.

Table 5.17: Positive Test Case and result for Admin delete tickets

Test Case ID	TC1
Test Case Description	Admin can view and delete tickets that have been booked
Test Data	Ticket ID: 1

Expected Output	TRUE
Result	PASS

Table 5.18: Negative Test Case and result for Admin delete tickets

Test Case ID	TC2
Test Case Description	Admin can view and delete tickets that have been booked
Test Data	Ticket ID:100
Expected Output	FALSE
Result	PASS

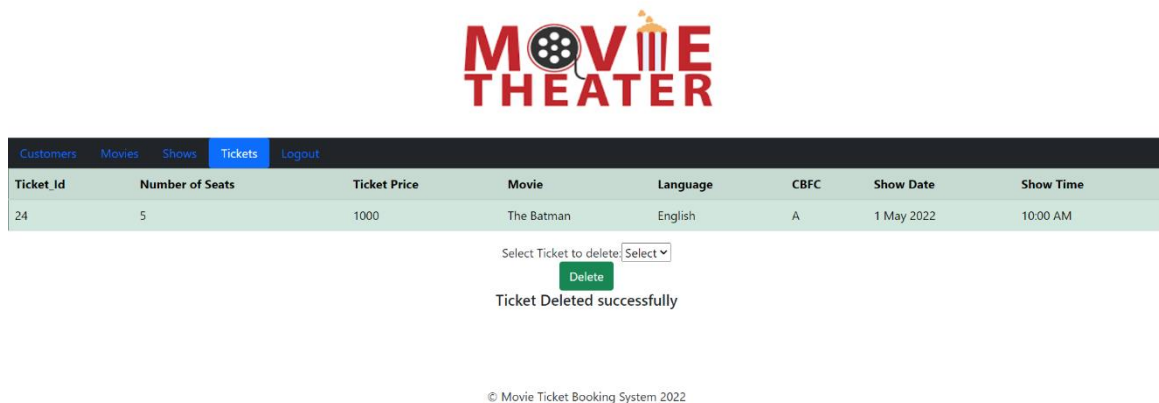


Figure 5.9 Successfully deleted ticket

Figure 5.9 shows the output page after the admin has successfully deleted a ticket from the list of booked tickets by properly selecting a valid id from the drop-down list.

5.1.10 Test Cases and Results for Admin Update Customer function:

The **Table 5.19**, **Table 5.20** shows that the possible test data for the both positive and negative test case given below, if the admin enters all fields properly, then the details of the customer will be updated in the database. If not, then the details won't be updated in the database. For example, if any field is left empty, then the details won't be updated in the database.

Table 5.19: Positive Test Case and result for Admin updating customers

Test Case ID	TC1
Test Case Description	Admin can change the customer details such as customer name, age and email id.
Test Data	Name: Manoj, Age: 21, Email ID: manoj@gmail.com
Expected Output	TRUE
Result	PASS

Table 5.20: Negative Test Case and result for Admin updating customers

Test Case ID	TC2
Test Case Description	Admin can change the customer details such as customer name, age and email id.
Test Data	Empty fields
Expected Output	FALSE
Result	PASS



Customers	Movies	Shows	Tickets	Logout
Mobile Number	Age	Name	Email	
7904137677	20	vignesh	vignesh@gmail.com	
8489803365	21	Manoj	manoj@gmail.com	
9486076049	20	Ianaard	ianaard@gmail.com	

Select Customer to delete:

Age

Name

Email

Warning: Tickets already booked will be canceled

Customer Updated Successfully

Figure 5.10 Successfully updated customer details

Figure 5.10 shows the page that will be displayed when the admin updates a customer's details successfully by selecting their mobile number from the drop-down list. An important feature to note is that both delete customer and update customer features are available on the same page and are dependent on the same drop-down list. Hence, the admin must be very careful to ensure that they are updating or deleting the proper id.

CHAPTER 6

CONCLUSION AND FUTURE ENHANCEMENT(S)

Movie ticket booking system is an application aimed at booking movie tickets online. The customers can book the ticket from any location by logging into their account, viewing movies being screened in home page, selecting the movie, checking the availability of seats and booking the tickets. In future the system can be expanded to display a greater number of theatres available across the country. Also, we could add the provision to choose the exact seat that the user wants. This app really helps the people who need to save their time by booking the tickets in online.

APPENDIX – A

SYSTEM REQUIREMENTS

HARDWARE REQUIREMENT:

Processor	:	Intel i3 or greater
RAM	:	8GB or greater
Storage	:	50 GB of free space required

SOFTWARE REQUIREMENT:

Operating System	:	Windows 8 or greater
DBMS	:	Microsoft SQL
IDE used	:	Visual Studio 2022
Language used	:	C#

APPENDIX – B

SOURCE CODE

Register.aspx:

```
<%@ Page Title="" Language="C#" MasterPageFile="~/Site1.Master" AutoEventWireup="true"
CodeBehind="Register.aspx.cs" Inherits="MovieSystem.WebForm1" %>

<asp:Content ID="Content2" ContentPlaceHolderID="ContentPlaceHolder1" runat="server">
    <form id="form1" runat="server">
        <div class="card text-center">
            <div class="card-header">
                Create New Account
            </div>
            <div class="card-body">

                <p><asp:TextBox ID="TextBox1" runat="server" TextMode="Number" placeholder="Mobile
Number"></asp:TextBox></p>
                <asp:RequiredFieldValidator ID="RequiredFieldValidator1" runat="server" ErrorMessage="*Enter
Mobile Number" ControlToValidate="TextBox1"
ValidationGroup="insertGroup"></asp:RequiredFieldValidator><br>
                <asp:RegularExpressionValidator ID="RegularExpressionValidator1" runat="server"
ErrorMessage="Invalid Mobile Number" ValidationExpression="[0-9]{10}"
ControlToValidate="TextBox1"></asp:RegularExpressionValidator>
                <br />
                <br />

                <p><asp:TextBox ID="TextBox2" runat="server" placeholder="Age"></asp:TextBox></p>
                <asp:RequiredFieldValidator ID="RequiredFieldValidator2" runat="server" ErrorMessage="*Enter Age"
ControlToValidate="TextBox2" ValidationGroup="insertGroup"></asp:RequiredFieldValidator><br>
                <asp:RangeValidator ID="RangeValidator1" runat="server" ErrorMessage="Age must be Between 13
and 80" ControlToValidate="TextBox2" MinimumValue="13" MaximumValue="80" Type="Integer"
></asp:RangeValidator>
                <br />
                <br />

                <p>
                    <asp:TextBox ID="TextBox3" runat="server" TextMode="Password"
placeholder="Password"></asp:TextBox> </p>
                    <asp:RequiredFieldValidator ID="RequiredFieldValidator3" runat="server" ErrorMessage="*Enter
Password" ControlToValidate="TextBox3"
ValidationGroup="insertGroup"></asp:RequiredFieldValidator><br>
                    <asp:RegularExpressionValidator ID="RegularExpressionValidator2" runat="server"
ErrorMessage="Invalid Password" ValidationExpression="[a-zA-Z]{6,13}" ControlToValidate="TextBox3"
ValidationGroup="insertGroup"></asp:RegularExpressionValidator>

                <asp:SqlDataSource ID="SqlDataSource1" runat="server" ConnectionString="<%$
ConnectionStrings:ConnectionString %>" SelectCommand="SELECT * FROM
[Customer]"></asp:SqlDataSource>

                <p><asp:TextBox ID="TextBox4" runat="server" placeholder="Name"></asp:TextBox></p>

                <asp:RequiredFieldValidator ID="RequiredFieldValidator4" runat="server" ErrorMessage="*Enter
Name" ControlToValidate="TextBox4" ValidationGroup="insertGroup"></asp:RequiredFieldValidator>
```

```

        <br />
        <br />
        <p><asp:TextBox ID="TextBox5" runat="server" placeholder="Email ID"
TextMode="Email"></asp:TextBox></p>

        <asp:RequiredFieldValidator ID="RequiredFieldValidator5" runat="server" ErrorMessage="*Enter
Email ID" ControlToValidate="TextBox5" ValidationGroup="insertGroup"></asp:RequiredFieldValidator>

        <center><div id="issuccess" runat="server" visible="false"><h5>Registration
Successful</h5></div></center>
        <p>
        <asp:Button ID="Button2" runat="server" OnClick="Button2_Click" Text="Back" CssClass="btn btn-
danger" />
        <asp:Button ID="Button1" runat="server" OnClick="Button1_Click" Text="Register" class="btn btn-
success" ValidationGroup="insertGroup"/>
        </p>
        </div>
        </div>
        </form>
</asp:Content>

```

Register.aspx.cs:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Data.SqlClient;
using System.Configuration;
using System.Data;

namespace MovieSystem
{
    public partial class WebForm1 : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {

        }

        protected void Button1_Click(object sender, EventArgs e)
        {
            try
            {
                SqlConnection conn = new
SqlConnection(ConfigurationManager.ConnectionStrings["ConnectionString"].ConnectionString);
                conn.Open();
                Int64 num = Int64.Parse(TextBox1.Text);
                int age = Int32.Parse(TextBox2.Text);
                string name = TextBox4.Text;
                string email = TextBox5.Text;
                string pwd = TextBox3.Text;
                SqlCommand cmd = new SqlCommand("insert into Customer values(" + num + "," + age + "," + pwd
+ "," + name + "," + email + ")", conn);
                cmd.ExecuteNonQuery();
                issuccess.Visible = true;
                conn.Close();
            }
        }
    }
}

```

```

        catch (Exception ex)
        {
            Response.Write("error" + ex.ToString());
        }
    }

    protected void Button2_Click(object sender, EventArgs e)
    {
        Response.Redirect("Signin.aspx");
    }

    protected void Button2_Click(object sender, EventArgs e)
    {
        try
        {
            SqlConnection conn = new
SqlConnection(ConfigurationManager.ConnectionStrings["ConnectionString"].ConnectionString);
            conn.Open();
            SqlCommand cmd = new SqlCommand("select * from movie", conn);
            SqlDataReader dr = cmd.ExecuteReader();
            DataTable dt = new DataTable();
            dt.Load(dr);
            GridView1.DataSource = dt;
            GridView1.DataBind();
            conn.Close();
        }
        catch (Exception ex)
        {
            Response.Write("error" + ex.ToString());
        }
    }

    protected void Button3_Click(object sender, EventArgs e)
    {
        try
        {
            SqlConnection conn = new
SqlConnection(ConfigurationManager.ConnectionStrings["ConnectionString"].ConnectionString);
            conn.Open();
            string pwd = TextBox5.Text;
            SqlCommand cmd = new SqlCommand("update customer set Password='" + pwd + "' where
Mobile_Num=9780970321", conn);
            cmd.ExecuteNonQuery();
            /*Response.Write("Password Changed Successfully");*/
            conn.Close();
        }
        catch (Exception ex)
        {
            Response.Write("error" + ex.ToString());
        }
    }
}
}
}

```

adminPageCustomers.aspx:

```
<%@ Page Title="" Language="C#" MasterPageFile="~/Site1.Master" AutoEventWireup="true"
CodeBehind="adminPageCustomers.aspx.cs" Inherits="MovieSystem.WebForm11" %>
<asp:Content ID="Content1" ContentPlaceHolderID="ContentPlaceHolder1" runat="server">
    <div>
        <ul class="nav nav-pills navbar-dark bg-dark">
            <li class="nav-item">
                <a class="nav-link active" aria-current="page" href="adminPageCustomers.aspx">Customers</a>
            </li>
            <li class="nav-item">
                <a class="nav-link" href="adminPageMovies.aspx">Movies</a>
            </li>
            <li class="nav-item">
                <a class="nav-link" href="adminPageShows.aspx">Shows</a>
            </li>
            <li class="nav-item">
                <a class="nav-link" href="adminPageTickets.aspx">Tickets</a>
            </li>
            <li class="nav-item">
                <a class="nav-link" href="Signin.aspx">Logout</a>
            </li>
        </ul>
    </div>
    <form id="form1" runat="server">
        <asp:SqlDataSource ID="SqlDataSource1" runat="server" ConnectionString="<%=
ConnectionStrings:ConnectionString %>" SelectCommand="SELECT * FROM
[Customer]"></asp:SqlDataSource>
        <asp:GridView ID="GridView1" CssClass="table table-success table-striped"
AutoGenerateColumns="false" runat="server" ShowHeader="true">
            <Columns>
                <asp:TemplateField HeaderText="Mobile Number">
                    <ItemTemplate>
                        <asp:Label ID="label8" Text="<%= # Eval("Mobile_Num") %>" runat="server" />
                    </ItemTemplate>
                </asp:TemplateField>
                <asp:TemplateField HeaderText="Age">
                    <ItemTemplate>
                        <asp:Label ID="label1" Text="<%= # Eval("Age") %>" runat="server" />
                    </ItemTemplate>
                </asp:TemplateField>
                <asp:TemplateField HeaderText="Name">
                    <ItemTemplate>
                        <asp:Label ID="label2" Text="<%= # Eval("Name") %>" runat="server" />
                    </ItemTemplate>
                </asp:TemplateField>
                <asp:TemplateField HeaderText="Email">
                    <ItemTemplate>
                        <asp:Label ID="label3" Text="<%= # Eval("Email") %>" runat="server" />
                    </ItemTemplate>
                </asp:TemplateField>
            </Columns>
        </asp:GridView>
        <center><div id="hascustomer" runat="server" visible="false"><h4>There are no registered
users.</h4></div></center>
    <br />
    <br />
    <br />
    <br />
</form>
</div>
```

```

        <center>Select Customer to delete:<asp:DropDownList ID="DropDownList1" AutoPostBack="true"
runat="server"
OnSelectedIndexChanged="DropDownList1_SelectedIndexChanged"></asp:DropDownList></center>
        <center>
            <br /><br />
            <asp:TextBox ID="agetf" runat="server" placeholder="Age"></asp:TextBox><br /><br />
            <asp:TextBox ID="nametf" runat="server" placeholder="Name"></asp:TextBox><br /><br />
            <asp:TextBox ID="mailtf" runat="server" placeholder="Email"></asp:TextBox><br /><br />
            <asp:Button ID="updateMovies" runat="server" Text="Update" CssClass="btn btn-primary"
OnClick="updateDetails_Click" />
            <asp:Button ID="deleteCustomers" runat="server" Text="Delete" CssClass="btn btn-danger"
OnClick="deleteCustomers_Click" />
            <center><div id="Div2" runat="server"><h6>Warning: Tickets already booked will be
canceled</h6></div></center>
            <div id="Div1" runat="server" visible="false"><h5>Please select ID</h5></div></center>
            <center><div id="delete" runat="server" visible="false"><h5>Customer Deleted
successfully</h5></div></center>
            <center><div id="update" runat="server" visible="false"><h5>Please enter all fields</h5></div></center>
            <center><div id="issuccess" runat="server" visible="false"><h5>Movie Updated
Successfully</h5></div></center>

        </form>
    </asp:Content>

```

adminPageCustomers.aspx.cs:

```

using System;
using System.Collections.Generic;
using System.Configuration;
using System.Data;
using System.Data.SqlClient;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

namespace MovieSystem
{
    public partial class WebForm1 : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            if (!IsPostBack)
            {
                //if (Session["user"] == null)
                //{
                //    Response.Redirect("error.aspx");
                //}
                setCustomerDetails();
            }
        }

        private void setCustomerDetails()
        {
            try
            {
                SqlConnection conn = new
SqlConnection(ConfigurationManager.ConnectionStrings["ConnectionString"].ConnectionString);
                conn.Open();

                SqlCommand cmd = new SqlCommand("select * from Customer", conn);
                SqlDataReader reader = cmd.ExecuteReader();
            }
        }
    }
}

```

```

        if (!reader.HasRows)
        {
            hascustomer.Visible = true;
        }
        else
        {
            DataTable dt = new DataTable();
            dt.Load(reader);

            GridView1.DataSource = dt;
            GridView1.DataBind();
            DropDownList1.DataSource = dt;
            DropDownList1.DataTextField = "Mobile_Num";
            DropDownList1.DataValueField = "Mobile_Num";
            DropDownList1.DataBind();
            DropDownList1.Items.Insert(0, "Select");
        }
        conn.Close();
    }
    catch (Exception ex)
    {
        Response.Write("error" + ex.ToString());
    }
}

protected void deleteCustomers_Click(object sender, EventArgs e)
{
    try
    {
        if (DropDownList1.SelectedValue.ToString() == "Select")
        {
            delete.Visible = false;
            Div1.Visible = true;
        }
        SqlConnection conn = new
SqlConnection(ConfigurationManager.ConnectionStrings["ConnectionString"].ConnectionString);
        conn.Open();
        SqlCommand cmd2 = new SqlCommand("delete from Customer where Mobile_Num=" +
DropDownList1.SelectedValue.ToString(), conn);
        int r = cmd2.ExecuteNonQuery();
        conn.Close();
        if (r == 1)
        {
            delete.Visible = true;
            Div1.Visible = false;
            clearText();
            setCustomerDetails();
        }
    }
    catch (Exception ex)
    {
        Response.Write("error" + ex.ToString());
    }
}

private void clearText()
{
    agetf.Text = "";
    nametf.Text = "";
    mailtf.Text = "";
}

```

```

protected void updateDetails_Click(object sender, EventArgs e)
{
    string n,a,m;
    n = nametf.Text;
    m = mailtf.Text;
    a = agetf.Text;

    if (n == "" || m == "" || a == "")
    {
        update.Visible = true;
        issuccess.Visible = false;
    }
    else
    {
        SqlConnection conn = new
SqlConnection(ConfigurationManager.ConnectionStrings["ConnectionString"].ConnectionString);
        conn.Open();

        SqlCommand cmd = new SqlCommand("update Customer set Age=" + a + ",Name=" + n + ",Email="
+ m + " where Mobile_Num=" + DropDownList1.SelectedValue.ToString(), conn);
        int r = cmd.ExecuteNonQuery();
        if (r == 1)
        {
            SqlCommand cmd2 = new SqlCommand("delete from Ticket where Show_Id in (select Show_Id
from Show where Screen_Id in (select Screen_Id from Theatre_Screens where Movie_Id=" +
DropDownList1.SelectedValue.ToString() + "))", conn);
            cmd2.ExecuteNonQuery();
            update.Visible = false;
            issuccess.Visible = true;
            clearText();
            setCustomerDetails();
        }
    }
}

protected void DropDownList1_SelectedIndexChanged(object sender, EventArgs e)
{
    try
    {
        if (DropDownList1.SelectedItem.Value.ToString() != "Select")
        {
            SqlConnection conn = new
SqlConnection(ConfigurationManager.ConnectionStrings["ConnectionString"].ConnectionString);
            conn.Open();

            SqlCommand cmd = new SqlCommand("select * from Customer where Mobile_Num=" +
DropDownList1.SelectedValue.ToString(), conn);
            SqlDataReader reader = cmd.ExecuteReader();

            DataTable dt = new DataTable();
            dt.Load(reader);
            reader.Close();
            foreach (DataRow dr in dt.Rows)
            {
                agetf.Text = dr["Age"].ToString();
                nametf.Text = dr["Name"].ToString();
                mailtf.Text = dr["Email"].ToString();
            }
            conn.Close();
        }
        else
        {
            clearText();
        }
    }
}

```

```

    }
}
catch (Exception ex)
{
    Response.Write("error" + ex.ToString());
}
}
}
}

```

bookTickets.aspx:

```

<%@ Page Title="" Language="C#" MasterPageFile="~/Site1.Master" AutoEventWireup="true"
CodeBehind="bookTickets.aspx.cs" Inherits="MovieSystem.WebForm5" %>
<asp:Content ID="Content1" ContentPlaceHolderID="ContentPlaceHolder1" runat="server">
    <div>
        <ul class="nav nav-pills navbar-dark bg-dark">
            <li class="nav-item">
                <a class="nav-link active" aria-current="page" href="dashboard.aspx">Home</a>
            </li>
            <li class="nav-item">
                <a class="nav-link" href="myTickets.aspx">My Tickets</a>
            </li>
            <li class="nav-item">
                <a class="nav-link" href="changePwd.aspx">Change Password</a>
            </li>
            <li class="nav-item">
                <a class="nav-link" href="Signin.aspx">Logout</a>
            </li>
        </ul>
    </div>
    <br />
    <br />
    <br />
    <form id="form1" runat="server">
        <div style="text-align:left;">
            <asp:SqlDataSource ID="SqlDataSource1" runat="server" ConnectionString="<%$
ConnectionStrings:ConnectionString %>" SelectCommand="SELECT [occupied_seats], [free_seats],
[show_day], [show_time], [Show_Id] FROM [SHOW]"></asp:SqlDataSource>
            <asp:GridView ID="GridView1" CssClass="table table-success table-striped"
AutoGenerateColumns="false" runat="server" EnablePersistedSelection="True"
OnRowCommand="GridView1_RowCommand" DataKeyNames="Show_Id">
                <Columns>
                    <asp:TemplateField HeaderText="Show ID">
                        <ItemTemplate>
                            <asp:Label ID="label8" Text="<%# Eval("Show_Id") %>" runat="server" />
                        </ItemTemplate>
                    </asp:TemplateField>
                    <asp:TemplateField HeaderText="Booked Seats">
                        <ItemTemplate>
                            <asp:Label ID="label1" Text="<%# Eval("occupied_seats") %>" runat="server" />
                        </ItemTemplate>
                    </asp:TemplateField>
                    <asp:TemplateField HeaderText="Available Seats">
                        <ItemTemplate>
                            <asp:Label ID="label2" Text="<%# Eval("free_seats") %>" runat="server" />
                        </ItemTemplate>
                    </asp:TemplateField>
                    <asp:TemplateField HeaderText="Show Date">

```



```

        <ItemTemplate>
            <asp:Label ID="label3" Text="<%# Eval("show_day") %>" runat="server" />
        </ItemTemplate>
    </asp:TemplateField>
    <asp:TemplateField HeaderText="Show Time">
        <ItemTemplate>
            <asp:Label ID="label4" Text="<%# Eval("show_time") %>" runat="server" />
        </ItemTemplate>
    </asp:TemplateField>

    <asp:TemplateField>

        <ItemTemplate>
            <asp:Button runat="server" CommandName="Select" CommandArgument="<%#
Container.DataItemIndex %>" Text="Book Tickets" CssClass="btn btn-success"/>
        </ItemTemplate>
    </asp:TemplateField>
</Columns>
</asp:GridView>

    <center><div runat="server" id="seatavailable" visible="false"><h5 forecolor="red">Selected Show
currently unavailable</h5></div></center>
</div>
</form>

</asp:Content>

```

bookTickets.aspx.cs:

```

using System;
using System.Collections.Generic;
using System.Configuration;
using System.Data;
using System.Data.SqlClient;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

namespace MovieSystem
{
    public partial class WebForm5 : System.Web.UI.Page
    {
        public string movie;
        protected void Page_Load(object sender, EventArgs e)
        {
            if (!IsPostBack)
            {
                if (Session["user"] == null)
                {
                    Response.Redirect("error.aspx");
                }
                movie = (string)Session["Movie"];
                setDetails();
            }
        }

        private void setDetails()
        {
            try
            {

```

```

        SqlConnection conn = new
SqlConnection(ConfigurationManager.ConnectionStrings["ConnectionString"].ConnectionString);
        conn.Open();

        int id;
        if (movie == "Batman")
        {
            id = 1;
        }
        else if (movie == "Kgf")
        {
            id = 2;
        }
        else
        {
            id = 3;
        }
        SqlCommand cmd = new SqlCommand("select Show_Id,occupied_seats, free_seats, show_day,
show_time from Show where Screen_Id=(select screen_id from Theatre_Screens where Movie_Id=" + id + ")",
conn);

        SqlDataReader reader = cmd.ExecuteReader();

        DataTable dt = new DataTable();
        dt.Load(reader);

        GridView1.DataSource = dt;
        GridView1.DataBind();
        conn.Close();
    }
    catch (Exception ex)
    {
        Response.Write("error" + ex.ToString());
    }
}

protected void GridView1_RowCreated(object sender, GridViewRowEventArgs e)
{
    if (e.Row.RowType == DataControlRowType.DataRow)
    {
        e.Row.Attributes["onclick"] =
            this.Page.ClientScript.
                GetPostBackClientHyperlink(this.GridView1, "Select$" + e.Row.RowIndex);
    }
}

protected void GridView1_SelectedIndexChanged1(object sender, EventArgs e)
{
    GridViewRow SelectedRow = GridView1.SelectedRow;
    string row = SelectedRow.Cells[0].Text;
    Response.Write(row);
}

protected void GridView1_RowCommand(object sender, GridViewCommandEventArgs e)
{
    if (e.CommandName == "Select")
    {
        int rowIndex = Convert.ToInt32(e.CommandArgument);

        GridViewRow SelectedRow = GridView1.Rows[rowIndex];
        int show = Convert.ToInt32(GridView1.DataKeys[rowIndex].Value.ToString());
        string available = (GridView1.Rows[rowIndex].FindControl("label2") as Label).Text.Trim();
        if (available == "0")
    }
}

```

```

    {
        seatavailable.Visible = true;
    }
    else
    {
        ClientScript.RegisterStartupScript(this.GetType(), "alert", "alert('show:') + show, true);
        Response.Write(show);
        Session["show"] = show;
        Response.Redirect("selectSeats.aspx");
    }
}
}
}
}
}

```

myTickets.aspx:

```

<%@ Page Title="" Language="C#" MasterPageFile="~/Site1.Master" AutoEventWireup="true"
CodeBehind="myTickets.aspx.cs" Inherits="MovieSystem.WebForm7" %>
<asp:Content ID="Content1" ContentPlaceHolderID="ContentPlaceHolder1" runat="server">
    <div>
        <ul class="nav nav-pills navbar-dark bg-dark">
            <li class="nav-item">
                <a class="nav-link" aria-current="page" href="dashboard.aspx">Home</a>
            </li>
            <li class="nav-item">
                <a class="nav-link active" href="myTickets.aspx">My Tickets</a>
            </li>
            <li class="nav-item">
                <a class="nav-link" href="changePwd.aspx">Change Password</a>
            </li>
            <li class="nav-item">
                <a class="nav-link" href="Signin.aspx">Logout</a>
            </li>
        </ul>
    </div>
    <br />
    <br />
    <div>
        <form id="form1" runat="server">
            <div runat="server" id="isticket" visible="false">
                <center><h3>There are No Tickets Booked By you</h3>
                <asp:Button runat="server" ID="booktickets" class="btn btn-success" Text="Book Tickets Now!!"
OnClick="booktickets_Click"/></center>
            </div>
            <%--<asp:SqlDataSource ID="SqlDataSource1" runat="server" ConnectionString="<%%$
ConnectionStrings:ConnectionString %>" SelectCommand="SELECT * FROM
[Ticket]"></asp:SqlDataSource>--%>
            <asp:GridView ID="GridView1" CssClass="table table-success table-striped"
AutoGenerateColumns="false" runat="server" OnRowDeleting="GridView1_RowDeleting"
DataKeyNames="Ticket_Id" ShowHeader="true">
                <Columns>
                    <asp:TemplateField HeaderText="Ticket_Id">
                        <ItemTemplate>
                            <asp:Label ID="label8" Text="<%# Eval("Ticket_Id") %>" runat="server" />
                        </ItemTemplate>
                    </asp:TemplateField>
                    <asp:TemplateField HeaderText="Number of Seats">
                        <ItemTemplate>
                            <asp:Label ID="label1" Text="<%# Eval("Num_of_seats") %>" runat="server" />
                        </ItemTemplate>
                    </asp:TemplateField>
                </Columns>
            </asp:GridView>
        </form>
    </div>

```

```

<asp:TemplateField HeaderText="Ticket Price">
    <ItemTemplate>
        <asp:Label ID="label2" Text=<%# Eval("Total_cost") %>' runat="server" />
    </ItemTemplate>
</asp:TemplateField>
<asp:TemplateField HeaderText="Movie">
    <ItemTemplate>
        <asp:Label ID="label3" Text=<%# Eval("Movie_Name") %>' runat="server" />
    </ItemTemplate>
</asp:TemplateField>
<asp:TemplateField HeaderText="Language">
    <ItemTemplate>
        <asp:Label ID="label4" Text=<%# Eval("Movie_Language") %>' runat="server" />
    </ItemTemplate>
</asp:TemplateField>
<asp:TemplateField HeaderText="CBFC">
    <ItemTemplate>
        <asp:Label ID="label5" Text=<%# Eval("Cbfc_Certificate") %>' runat="server" />
    </ItemTemplate>
</asp:TemplateField>
<asp:TemplateField HeaderText="Show Date">
    <ItemTemplate>
        <asp:Label ID="label6" Text=<%# Eval("show_day") %>' runat="server" />
    </ItemTemplate>
</asp:TemplateField>
<asp:TemplateField HeaderText="Show Time">
    <ItemTemplate>
        <asp:Label ID="label7" Text=<%# Eval("show_time") %>' runat="server" />
    </ItemTemplate>
</asp:TemplateField>
<asp:TemplateField>
    <ItemTemplate>
        <asp:Button runat="server" CommandName="Delete" Text="Cancel Tickets" class="btn btn-
danger" />
    </ItemTemplate>
</asp:TemplateField>
</Columns>

</asp:GridView>
<center><div id="issuccess" runat="server" visible="false"><h5>Ticket Canceled. Amount Will be
Refunded Soon.</h5></div></center>
</form>
</div>
</asp:Content>

```

myTickets.aspx.cs:

```

using System;
using System.Collections.Generic;
using System.Configuration;
using System.Data;
using System.Data.SqlClient;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

namespace MovieSystem
{
    public partial class WebForm7 : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)

```

```

{
    if (!IsPostBack)
    {
        if (Session["user"] == null)
        {
            Response.Redirect("error.aspx");
        }
        setDetails();
    }
}

private void setDetails()
{
    try
    {
        SqlConnection conn = new
SqlConnection(ConfigurationManager.ConnectionStrings["ConnectionString"].ConnectionString);
        conn.Open();

        SqlCommand cmd = new SqlCommand("select
a.Ticket_Id,a.Num_of_seats,a.Total_cost,d.Movie_Name,d.Movie_Language,d.Cbfc_Certificate,b.show_day,b.s
how_time from Ticket a inner join Show b on (a.Show_Id= b.Show_Id) inner join Theatre_Screens c on
(b.Screen_Id=c.Screen_Id) inner join Movie d on (c.Movie_Id=d.Movie_Id) where a.User_Id=" +
Session["user"], conn);
        SqlDataReader reader = cmd.ExecuteReader();

        if (!reader.HasRows)
        {
            isticket.Visible = true;
        }
        else
        {
            DataTable dt = new DataTable();
            dt.Load(reader);

            GridView1.DataSource = dt;
            GridView1.DataBind();
        }
        conn.Close();
    }
    catch (Exception ex)
    {
        Response.Write("error" + ex.ToString());
    }
}

protected void GridView1_RowDeleting(object sender, GridViewDeleteEventArgs e)
{
    try
    {
        Response.Write(GridView1.DataKeys[e.RowIndex].Value.ToString());
        SqlConnection conn = new
SqlConnection(ConfigurationManager.ConnectionStrings["ConnectionString"].ConnectionString);
        conn.Open();

        SqlCommand cmd1 = new SqlCommand("select Num_of_seats,Show_Id from Ticket where
Ticket_Id=" + Convert.ToInt32(GridView1.DataKeys[e.RowIndex].Value.ToString()), conn);
        int n = cmd1.ExecuteNonQuery();

        SqlDataReader reader = cmd1.ExecuteReader();
        DataTable dt = new DataTable();
        dt.Load(reader);
    }
}

```

```

        int seats=0,show=0;
        foreach(DataRow dr in dt.Rows)
        {
            seats = Int32.Parse(dr["Num_of_seats"].ToString());
            show = Int32.Parse(dr["Show_Id"].ToString());
        }
        Response.Write(seats+ show);
        SqlCommand cmd2 = new SqlCommand("delete from Ticket where Ticket_Id=" +
Convert.ToInt32(GridView1.DataKeys[e.RowIndex].Value.ToString()), conn);
        int r=cmd2.ExecuteNonQuery();
        if (r == 1)
        {
            issuccess.Visible=true;
            Response.Redirect("myTickets.aspx");
        }
        conn.Close();

    }
    catch (Exception ex)
    {
        Response.Write("error" + ex.ToString());
    }
}

protected void booktickets_Click(object sender, EventArgs e)
{
    Response.Redirect("dashboard.aspx");
}
}
}

```

REFERENCES

1. <https://www.c-sharpcorner.com/>
2. <https://stackoverflow.com/>
3. <https://www.geeksforgeeks.org/>
4. Herbert Schildt, “C# 4.0: The Complete Reference”, McGraw Hill Education, Indian Edition, 2010, Reprint 2017
5. Mark Michaelis, Eric Lippert, “Essential C# 6.0”, Pearson, 2015
6. S.Thamarai Selvi, R.Murugesan, “A textbook on C#”, Pearson, 2012